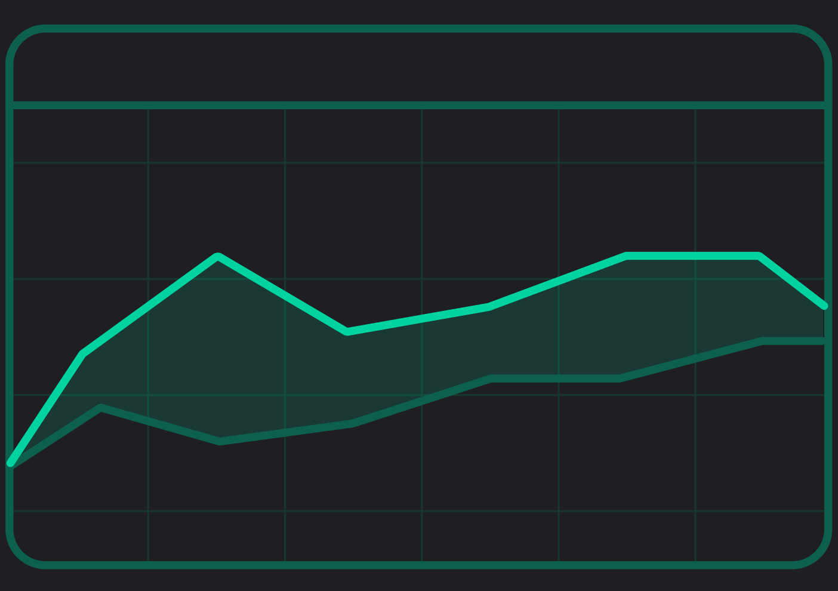
Arenadata QuickMarts

Arenadata QuickMarts (ADQM) — кластерная колоночная система управления базами данных, созданная на основе ClickHouse. С помощью ADQM можно в режиме реального времени генерировать аналитические отчеты разного плана, используя большие объемы информации, хранящиеся в витринах. По скорости работы ADQM многократно превосходит традиционные СУБД.

Версия CURRENT Язык: RU



Содержание

1. Концепции
   - Особенности ADQM
   - Архитектура ADQM
   - Шардирование
   - Типовой кластер
   - Геораспределенный кластер
   - Проксирование
2. Подготовка окружения
   - Требования к оборудованию
   - Требования к сети
   - Программные требования
3. Начало работы
   - Установка
     - Online-установка
       - Установка ADCM
       - Подготовка хостов
       - Установка кластера ADQM
         - Создание кластера
         - Добавление сервисов
         - Добавление хостов в кластер
         - Добавление компонентов
         - Настройка сервисов
         - Настройка кластера
         - Установка кластера
       - Установка мониторинга
     - Offline-установка
       - Установка ADCM
       - Подготовка хостов
       - Установка кластера Enterprise Tools
         - Создание кластера
         - Добавление сервисов
         - Добавление хостов в кластер
         - Добавление компонентов
         - Установка кластера
       - Установка кластера ADQM
         - Создание кластера
         - Добавление сервисов
         - Добавление хостов в кластер
         - Добавление компонентов
         - Настройка сервисов
         - Настройка кластера
         - Импорт настроек ET
         - Установка кластера
       - Установка мониторинга
   - Подключение к ADQM
     - Web-интерфейс ClickHouse
     - Клиент командной строки ClickHouse
     - JDBC-драйвер
     - Сторонние клиентские инструменты
       - DBeaver
4. Управление доступом
   - Аутентификация
     - По паролю
     - LDAP
     - По SSL-сертификату
     - Kerberos
       - MIT Kerberos
       - MS Active Directory
         - Проблема с типом шифрования в РЕД ОС
       - FreeIPA
   - Авторизация
   - Шифрование соединений с базой данных
5. Администрирование СУБД
   - Управление кластером через ADCM
     - Кластерные действия
     - Сервисные действия
     - Конфигурирование логических кластеров
     - Rolling upgrade
     - Режим обслуживания
   - Сервисы координации
     - ZooKeeper
     - ClickHouse Keeper
   - clickhouse-copier
   - Настройка хранилищ данных
   - Системные таблицы
   - Настройки
     - Управление настройками пользователя
     - Именованные коллекции
   - Метрики мониторинга
   - Логирование
6. Работа с данными
   - Обзор объектов базы данных
     - Базы данных
     - Таблицы
       - Движки таблиц
       - Обзор таблиц
       - Обновление и удаление строк
       - Сжатие данных
       - TTL
     - Индексы
     - Словари
     - Материализованные представления
     - Проекции
     - Пользовательские функции и таблицы Executable
   - Запросы к базе данных
     - Пример работы с таблицами в clickhouse-client
     - Типы запросов
       - JOIN
       - Подзапросы
       - Комбинирование запросов
       - Общие табличные выражения (CTE)
     - Использование функций
       - Агрегатные функции
         - flameGraph
       - Оконные функции
       - Табличные функции
     - Работа с комплексными типами данных
       - JSON
       - Вложенные структуры данных
     - Полнотекстовый поиск
     - Оптимизация запросов
       - Повышение производительности запросов
       - Анализ запросов
       - Семплирующий профилировщик запросов
     - Принудительное завершение запроса
   - Кодеки шифрования
   - Машинное обучение
7. Интеграция с внешними системами
   - JDBC Bridge
   - ADH
     - HDFS
     - Hive
   - ADS
     - Kafka
   - ADPG
   - S3
   - Импорт и экспорт в файлы
8. Справочные материалы
   - Конфигурационные параметры
9. Релизы
   - Релизы ADQM
   - Известные проблемы

Особенности ADQM

Елена Дворядкина

Назад к содержанию

Arenadata QuickMarts (ADQM) — это СУБД для выполнения аналитических запросов в режиме реального времени (OLAP) на структурированных больших данных. ADQM может применяться в различных сферах для решения задач, требующих высокой скорости обработки постоянно поступающей информации, например:

- аналитика веб-проектов и мобильных приложений;
- мониторинг технических и бизнес-метрик;
- быстрые витрины данных;
- работа со структурированными логами и событиями;
- мониторинг и анализ данных в режиме реального времени.

ADQM разработана на базе аналитической СУБД с открытым исходным кодом ClickHouse. Ключевые особенности ADQM/ClickHouse:

- **Колоночное хранение данных** — данные каждой колонки хранятся отдельно от других колонок. Такой формат хранения позволяет считывать данные быстрее (искать значения по отдельным колонкам, а не по всей таблице сразу) и эффективно сжимать однотипные данные.
- **Физическая сортировка данных по первичному ключу** — возможность быстро получать данные для конкретных значений или диапазонов значений ключа.
- **Векторные вычисления** — за счет обработки данных по векторам (участкам колонок) достигается высокая эффективность по CPU.
- **Распараллеливание операций** как в пределах одного сервера на несколько процессорных ядер, так и в рамках распределенных вычислений на кластере за счет механизма шардирования.
- **Поддержка приближенных вычислений** на части выборки — за счет снижения числа обращений к хранилищу повышается скорость обработки данных.
- **Линейная масштабируемость** — путем добавления новых узлов можно построить кластер очень большого размера, способный хранить и обрабатывать петабайты данных.
- **Работа на жестких дисках** — возможность дополнительно снизить стоимость хранения данных по сравнению с другими колоночными СУБД, которые работают только в RAM, так как жесткие диски дешевле оперативной памяти. SSD и оперативная память тоже могут полноценно использоваться, если доступны.
- **Асинхронная multi-master репликация** — после записи на любую доступную реплику, данные распространяются на все остальные реплики в фоновом режиме. Благодаря полной идентичности данных на разных репликах система может автоматически восстанавливать данные после большинства сбоев, обеспечивая тем самым высокую отказоустойчивость.
- **Язык запросов SQL** — ADQM/ClickHouse поддерживает диалект SQL, близкий к стандарту ANSI SQL, но содержащий различные расширения: массивы и вложенные структуры данных, вероятностные структуры, возможность подключить внешнее хранилище key/value, и другие.
- Поддержка различных **движков баз данных** (database engines) и **движков таблиц** (table engines) — все движки баз данных и таблиц, которые может использовать ADQM/ClickHouse, описаны в соответствующих разделах документации ClickHouse: Database Engines и Table Engines.
- **Быстрая вставка данных** — благодаря движку таблиц MergeTree обеспечивается высокая скорость записи данных в таблицы. Блокировки при добавлении новых данных отсутствуют.
- **Интеграция с множеством внешних источников** — ADQM/ClickHouse может взаимодействовать по специальным протоколам с различными внешними системами, такими как Kafka, RabbitMQ, Hadoop (HDFS), MySQL, PostgreSQL, MongoDB и другими.

ADQM/ClickHouse также обладает некоторыми особенностями, которые могут считаться ограничениями:

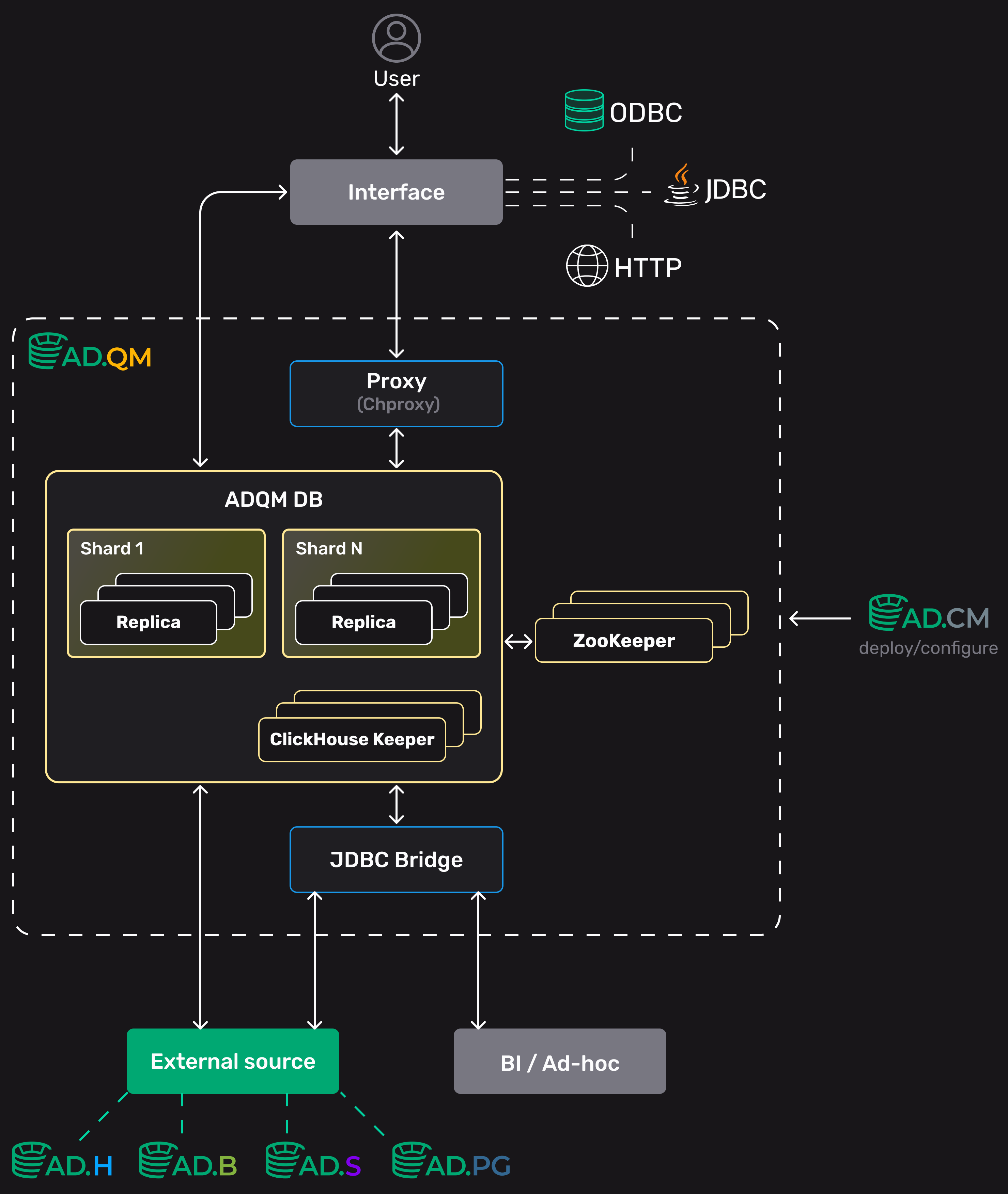
- Отсутствие полноценных транзакций — ADQM/ClickHouse является OLAP, а не OLTP-системой, и не поддерживает транзакционность записей, так как ориентирован, в первую очередь, на считывание данных.
- Изменение или удаление ранее записанных данных при большом количестве запросов происходит медленно.
- Низкая скорость точечного чтения одиночных строк по своим ключам из-за разреженного индекса.

Архитектура ADQM

Елена Дворядкина

Назад к содержанию

Обобщенно архитектура ADQM представлена на схеме ниже.



Архитектура ADQM

Особенности архитектуры ADQM:

- Для доступа к ADQM пользователи могут использовать драйверы JDBC и ODBC, HTTP-интерфейс, консольный клиент, а также множество "оберток" (wrappers) на Python, PHP, Node.js, Perl, Ruby, Go и многие другие. Дополнительная информация об интерфейсах для взаимодействия с базой данных ClickHouse доступна в разделе Drivers and Interfaces документации ClickHouse.
- Кластер ADQM включает следующие сервисы:
  - **ADQMDB** — колоночно-ориентированная СУБД ClickHouse. Данные могут быть расположены на разных шардах, где каждый шард — группа реплик. Репликация работает на уровне таблиц в рамках одного шарда независимо от других шардов.
  - **ZooKeeper**, **ClickHouse Keeper** — сервисы для координации репликации данных, не участвуют в процессе обработки и выполнения запросов.
  - **Chproxy** — HTTP-прокси и балансировщик нагрузки для базы данных ADQMDB/ClickHouse.

  Управление кластером ADQM можно выполнять с помощью специализированного модульного инструмента Arenadata Cluster Manager (ADCM), который позволяет быстро устанавливать, настраивать и обновлять data-сервисы через графический интерфейс.
- ADQM может подключаться к внешнему источнику данных (напрямую или через JDBC Bridge) и обращаться к его данным. Например, возможна интеграция ADQM с такими системами, как MySQL, MongoDB, Hadoop (HDFS), Apache Kafka, PostgreSQL и другими. Полный список поддерживаемых интеграций представлен в разделе Table Engines for Integrations.

  Для подключения к продуктам Arenadata — Arenadata Database (ADB), Arenadata Hadoop (ADH), Arenadata Streaming (ADS), Arenadata Postgres (ADPG) — доступна нативная интеграция в рамках единой платформы Arenadata Enterprise Data Platform (EDP).
- Для визуализации обработанных в ADQM данных и построения аналитических отчетов, могут использоваться любые BI-средства, предоставляющие доступ через JDBC/ODBC. Например, BusinessObjects, Power BI, Luxms, Visiology и другие.

Шардирование

Елена Дворядкина

Содержание

Назад к содержанию

- Распределенные таблицы
  - Создание распределенной таблицы
  - Конфигурация кластера
- Запись данных на кластер
  - Запись напрямую в локальные таблицы
  - Запись в распределенную таблицу
- Чтение из распределенной таблицы
- Перешардирование

Шардирование — это принцип проектирования базы данных, при котором части одной таблицы размещаются на разных шардах. **Шард** — узел кластера, который может состоять из одной или нескольких реплик. **Реплики** — это серверы, на которых дублируются данные в рамках шарда. Запрос на чтение или запись в шард может быть отправлен на любую его реплику, выделенного мастера нет.

Шардирование обычно используется в следующих случаях:

- Увеличивается количество данных и частота запросов к базе данных.
- Система требует все больше ресурсов, но вертикальное масштабирование кластера, которое предполагает обновление аппаратного обеспечения имеющихся серверов, не может быть использовано, так как существуют физические пределы — по количеству ядер на процессор, количеству процессоров, объему памяти и т.д.

Шардирование позволяет:

- **Преодолеть технические ограничения.** Если данные не помещаются на одном сервере или текущая инфраструктура хранения данных работает на пределе возможностей оборудования, можно горизонтально масштабировать кластер, то есть добавить новые серверы к существующему стеку и распределить данные по нескольким шардам.
- **Повысить доступность.** Шардирование позволяет смягчить последствия сбоев. Если данные хранятся на одном сервере, отказ этого сервера может привести к потере доступа ко всем данным. А отказ, например, одного шарда из пяти оставляет доступными 80% данных таблицы.
- **Увеличить скорость выполнения запросов.** Запросы конкурируют с друг другом за вычислительные ресурсы серверов кластера, что может приводить к снижению скорости обработки запросов. В шардированном кластере, где запросы к одной и той же таблице могут выполняться параллельно, устраняется конкуренция за общие ресурсы, что позволяет сократить время обработки запроса.

Распределенные таблицы

Шардирование данных реализуется с помощью **распределенных таблиц** — таблиц на движке Distributed. Распределенная таблица не хранит данные. Данные хранятся в локальных таблицах, которые находятся на серверах каждого шарда. Распределенная таблица обеспечивает маршрутизацию запросов к локальным таблицам, другими словами, позволяет обрабатывать запросы распределенно на нескольких серверах.

Создание распределенной таблицы

Распределенную таблицу можно создавать где угодно, в том числе на серверах, не входящих в шарды, на которых хранятся таблицы с данными. Но стандартный способ — размещение распределенной таблицы на всех шардах, на которых размещаются таблицы с данными.

Пример запроса на создание распределенной таблицы на всех серверах кластера:

```sql
CREATE TABLE distributed_table ON CLUSTER cluster_name AS table_name
ENGINE = Distributed(cluster_name, database, table_name [, sharding_key][, policy_name]);
```

При использовании движка Distributed требуется передать следующие параметры:

- имя кластера в конфигурационном файле сервера;
- имя базы данных для таблицы, к которой будет обращаться распределенная таблица (имя базы данных должны быть одинаковым на всех серверах кластера);
- имя таблицы, к которой будет обращаться распределенная таблица (имя таблицы должны быть одинаковым на всех серверах кластера);
- не обязательно — ключ шардирования, на основе которого данные будут распределяться по шардам при выполнении запросов `INSERT`;
- не обязательно — имя политики для хранения временных файлов при асинхронной отправке.

Конфигурация кластера

Движок Distributed требует описание кластера в конфигурационном файле каждого сервера (*config.xml*). Кластер, заданный в конфигурационном файле — это логическая сущность, объединяющая серверы. Один сервер может участвовать в нескольких логических кластерах. Таким образом обеспечивается гибкость распределения данных между серверами.

Кластеры задаются следующим образом:

```xml
<remote_servers>
    <cluster_name>
        <shard>
            <weight>2</weight>
            <internal_replication>false</internal_replication>
            <replica>
                <priority>1</priority>
                <host>host_01_1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host_01_2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <weight>1</weight>
            <internal_replication>false</internal_replication>
            <replica>
                <host>host_02_1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host_02_2</host>
                <port>9000</port>
            </replica>
        </shard>
    </cluster_name>
</remote_servers>
```

В данном примере задан распределенный кластер с именем `cluster_name`, состоящий из двух шардов. Каждый шард содержит две реплики. В качестве параметров для каждой реплики указываются `host` (адрес удаленного сервера) и `port` (TCP-порт для межсерверного взамодействия, обычно `9000`).

Кластер может включать один шард (в этом случае обработку запроса называют удаленной, а не распределенной) или несколько шардов. В каждом шарде можно указать от одной до произвольного числа реплик. Число реплик для каждого шарда в составе кластера может быть разным.

Для шарда могут быть заданы следующие необязательные параметры:

- `weight` (по умолчанию, `1`) — вес шарда при записи данных. Данные будут распределяться по шардам в количестве, пропорциональном весу шардов. Например, есть два шарда — у первого вес `2`, у второго вес `1`. В этом случае доля строк, отправляемых на первый шард, будет составлять `2/3`, а на второй — `1/3`.
- `internal_replication` — определяет, как записывать данные в реплики:
  - `true` — данные записываются в первую доступную реплику. Этот вариант следует использовать, если данные записываются в реплицируемую таблицу, которая будет сама управлять репликацией данных. Данные, записанные на реплику, будут скопированы на другие реплики шарда в асинхронном режиме.
  - `false` (по умолчанию) — данные записываются на все реплики, то есть распределенная таблица занимается репликацией данных самостоятельно. Это менее надежно, чем использование реплицируемых таблиц, так как в этом случае идентичность реплик не контролируется (со временем данные на них могут начать немного различаться).

Также можно настроить алгоритм балансировки нагрузки на реплики, то есть указать предпочтения, на какую из реплик в первую очередь отправлять запрос. Для этого можно задать параметр реплики `priority` в конфигурации кластера (меньшее значение — больший приоритет реплики, по умолчанию — `1`) или использовать настройку load_balancing.

Конфигурирование кластера через ADCM

Сконфигурировать логический кластер ADQM можно в интерфейсе ADCM, используя параметры в секции ***Cluster configuration*** на странице конфигурирования сервиса ADQMDB. Для этого можно:

- Указать значение ***Replication factor*** перед установкой ADQM — на основе указанного фактора репликации все хосты ADQM будут автоматически объединены в логический кластер.
- Вручную добавить и настроить необходимые кластеры с помощью визуальных элементов параметра ***Cluster Configuration***.

После установки ADQM или перезапуска сервиса ADQMDB описание кластера автоматически добавится в конфигурационный файл. Подробнее в статье Конфигурирование логических кластеров в интерфейсе ADCM.

Запись данных на кластер

Есть два способа записывать данные на кластер.

Запись напрямую в локальные таблицы

Вы можете самостоятельно определять, на какие серверы какие данные записывать, и выполнять запись непосредственно на каждый шард (то есть отправлять запросы `INSERT` напрямую в те таблицы, которые хранят данные). Это решение является наиболее гибким и оптимальным, поскольку вы можете использовать любую схему шардирования, отвечающую требованиям предметной области, и записывать данные на разные шарды полностью независимо.

Запись в распределенную таблицу

Вы можете отправлять запрос `INSERT` в распределенную таблицу. В этом случае таблица будет сама распределять вставляемые данные по серверам. Для этого необходимо задать **ключ шардирования**. Если шард всего один, то запись работает и без указания ключа шардирования, так как в этом случае он не имеет смысла.

Ключ шардирования может быть задан как произвольное выражение, возвращающее целое число. Например, в качестве ключа шардирования можно использовать:

- столбец таблицы `UserID` — для распределения данных по идентификатору пользователя (данные одного пользователя будут расположены на одном шарде);
- функцию toDayOfWeek — для распределения данных по шардам в зависимости от дня недели определенной даты (например, если в таблице есть поле `saleDate`, содержащее дату продажи, а кластер содержит 7 шардов);
- выражение `rand()` — для случайного распределения данных.

Для выбора на какой шард записывается строка данных, используется остаток от деления ключа шардирования на суммарный вес шардов. Строка данных отправляется на шард, если остаток от деления входит в интервал от `prev_weights` до `prev_weights + weight`, где `prev_weights` - сумма весов шардов с меньшим номером, а `weight` — вес этого шарда. Например, есть 2 шарда — у одного вес `2`, у другого вес `1`. Сумма весов всех шардов `2 + 1 = 3`. В этом случае возможные остатки от деления ключа шардирования на суммарный вес шардов — `0`, `1` и `2`. Данные будут записываться на первый шард, если остаток `0` или `1` (диапазон `[0, 2)`), и на второй шард — если остаток `2` (диапазон `[2, 3)`).

Простой остаток от деления является довольно ограниченным решением для шардирования и подходит не для всех случаев. Он подходит для среднего и большого объема данных (десятки серверов), но не для очень больших объемов данных (сотни серверов и больше). В последнем случае лучше использовать схему шардирования, учитывающую требования предметной области, и не использовать возможность записи в распределенные таблицы.

Чтение из распределенной таблицы

В отличие от запросов `INSERT`, запросы `SELECT` отправляются на все шарды кластера, вне зависимости от того, каким образом данные распределены по шардам (это поведение можно поменять с помощью настройки optimize_skip_unused_shards). При чтении для каждого из шардов выбирается одна из доступных реплик. Шарды обрабатывают запрос на чтение (максимум обработки выполняется на удаленных серверах, чтобы передавать меньше данных по сети) и отправляют частично агрегированные результаты в распределенную таблицу, которая объединит все полученные данные и отправит полный результат пользователю.

Перешардирование

Встроенной возможности автоматического перешардирования в ADQM нет. Но для перераспределения данных можно использовать функциональность clickhouse-copier (предоставляется пакетом adqm-clickhouse-copier в составе сервиса ADQMDB), которая позволяет копировать данные из таблиц одного кластера (source cluster) в таблицы другого кластера (destination cluster). Для этого необходимо:

1. Создать файл конфигурации заданий на копирование. В этом файле описать:
   - кластер с исходными данными и кластер с данными для перешардирования (в секции `remote_servers`);
   - задачи на шардирование (в секции `tables`);
   - новый ключ шардирования (в секции `sharding_key`).
2. Создать файл конфигурации с параметрами соединения с ZooKeeper.
3. Создать задачу в ZooKeeper.
4. Запустить clickhouse-copier для копирования данных.
5. Создать новую распределенную таблицу, переключиться со старого кластера на новый.

Для получения более подробной информации обратитесь к статье Примеры использования clickhouse-copier.

Типовой кластер

Елена Дворядкина

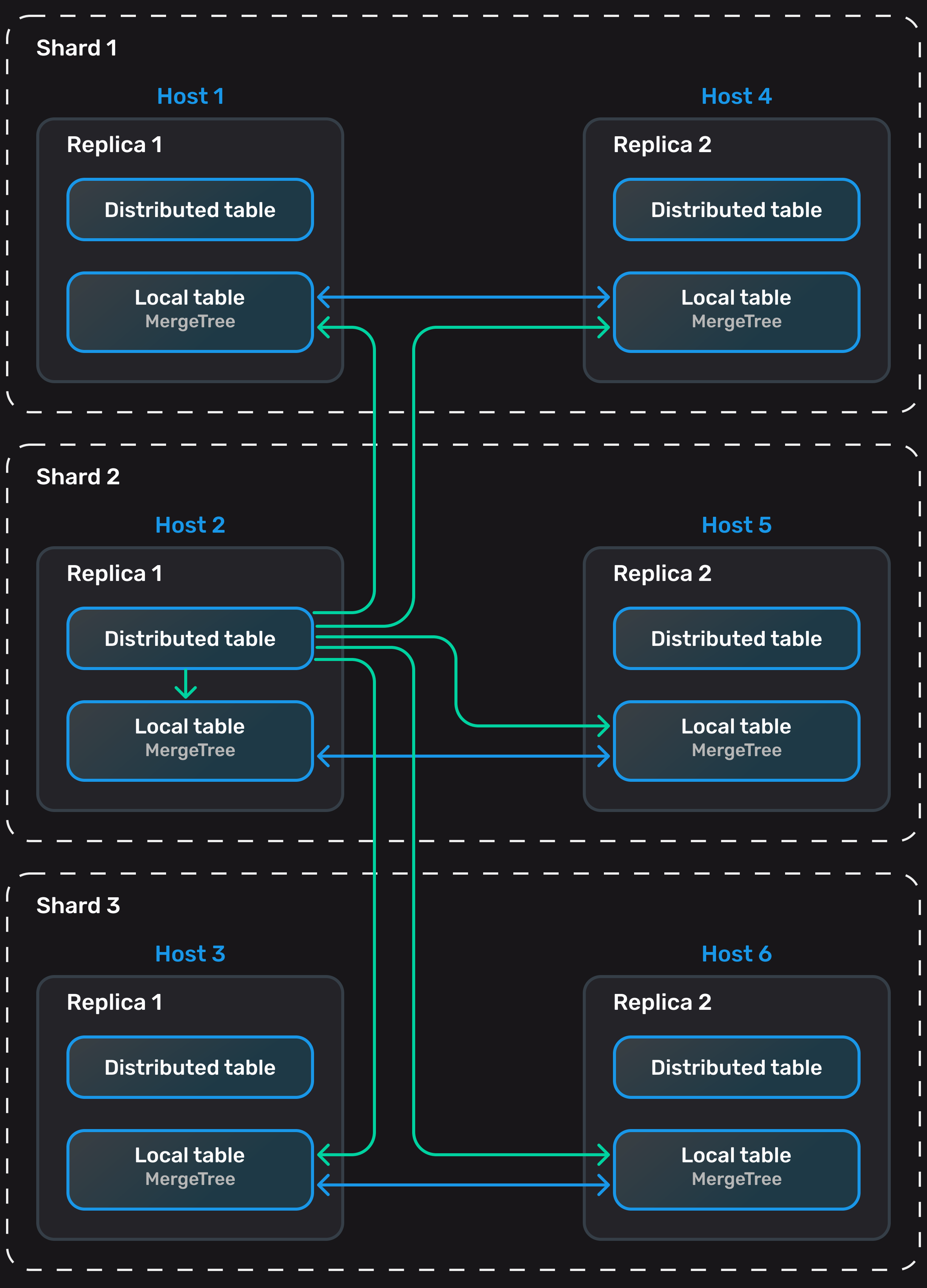
Содержание

Назад к содержанию

Создание реплицируемых таблиц
Создание распределенных таблиц

Кластер ADQM не имеет мастер-ноды и единой точки входа. Запросы могут отправляться к любому из хостов в кластере.

Рассмотрим типовой кластер ADQM — три шарда, каждый состоит из двух реплик (иными словами, распределенный кластер из 3 узлов с фактором репликации 2). Шарды — это серверы или группы серверов, на которых размещаются разные части одной базы данных. **Реплики** — дублирующие серверы в составе шарда, которые содержат одинаковые данные.



Пример типового кластера ADQM

Локальные таблицы — это реплицируемые таблицы, которые отвечают за хранение данных. Распределенные таблицы не хранят данные, они позволяют сделать запрос на несколько локальных таблиц, распределенных на хостах, которые объединены в виртуальный кластер.

В настройках каждого сервера (*config.xml*) должны присутствовать следующие секции:

- `remote_servers` — кластеры, которые используются для создания распределенных таблиц и выполнения распределенных запросов;
- `macros` — макросы для автоматической подстановки значений, соответствующих хосту, при создании реплицируемых таблиц.

Конфигурация `remote_servers` для представленного кластера:

```
<remote_servers>
  <default_cluster>
        <shard>
            <replica>
                <host>host_1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host_4</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <replica>
                <host>host_2</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host_5</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <replica>
                <host>host_3</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host_6</host>
                <port>9000</port>
            </replica>
        </shard>
  </default_cluster>
</remote_servers>
```

Здесь задан распределенный кластер с именем `default_cluster`, состоящий из трех шардов, каждый из которых содержит две реплики. В качестве параметров для каждого сервера указываются `host` (адрес удаленного сервера) и `port` (TCP-порт для межсерверного взаимодействия, обычно `9000`).

Настройка `macros` содержит идентификаторы шарда и реплики для каждого сервера. Например, макросы для `host-1`:

```
<macros>
    <replica>1</replica>
    <shard>1</shard>
</macros>
```

> **ПРИМЕЧАНИЕ**
>
> Чтобы объединить хосты ADQM в кластер, используйте интерфейс ADCM — подробнее в статье Конфигурирование логических кластеров в интерфейсе ADCM.

Создание реплицируемых таблиц

Пример создания реплицируемой таблицы:

```
CREATE TABLE test_local ON CLUSTER default_cluster
(
    id Int32,
    value_string String
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/test_local', '{replica}')
ORDER BY id;
```

Для создания реплицируемых таблиц используется движок ReplicatedMergeTree. Метаинформация, относящаяся к процессу репликации, хранится в ZooKeeper или ClickHouse Keeper. Соответственно, ReplicatedMergeTree имеет два обязательных параметра для формирования отдельного каталога в ZooKeeper/ClickHouse Keeper для каждой реплицируемой таблицы:

- путь к таблице ZooKeeper/ClickHouse Keeper;
- имя реплики в ZooKeeper/ClickHouse Keeper (идентифицирует разные реплики одной и той же таблицы).

В данном примере для создания реплицируемых таблиц используется распределенный DDL-запрос (`ON CLUSTER`) — запрос, который можно выполнить один раз на одном сервере, при этом необходимые таблицы будут созданы на всех хостах кластера. Идентификатор шарда и имя реплики, указанные как переменные `{shard}` и `{replica}` в параметрах ReplicatedMergeTree, заменяются на определенные значения из секции `macros` конфигурации каждого сервера. Таким образом для каждого экземпляра реплицируемой таблицы будет задан уникальный путь в ZooKeeper.

Если не использовать выражение `ON CLUSTER`, запрос на создание реплицируемой таблицы необходимо выполнить на каждом сервере отдельно. В этом случае можно не использовать макросы в параметрах, а явно указать соответствующие значения для каждого сервера при выполнении запроса. Однако при работе с большими кластерами рекомендуется использовать макросы, чтобы сократить вероятность ошибки.

Создание распределенных таблиц

Для создания распределенной таблицы используется движок Distributed, например:

```
CREATE TABLE test_distr ON CLUSTER default_cluster AS default.test_local
ENGINE = Distributed(default_cluster, default, test_local, rand());
```

В данном примере в движок Distributed передаются следующие параметры:

- кластер, на котором будет запущен запрос (имя кластера в `remote_servers` конфигурационного файла сервера);
- база данных для реплицируемой таблицы, к которой будет обращаться распределенная таблица;
- реплицируемая таблица, к которой будет обращаться распределенная таблица;
- ключ шардирования для вставки данных в распределенную таблицу (выражение `rand()` используется для случайного распределения данных по шардам).

Георaспределенный кластер

Елена Дворядкина

Содержание

Назад к содержанию

- Stretched-кластер
  - Конфигурация логического кластера
  - Создание распределенных таблиц
- Stretched-кластер с локальным чтением
  - Конфигурация логических кластеров
  - Создание распределенных таблиц
- Stretched-кластер с локальной записью
  - Конфигурация логических кластеров
  - Создание распределенных таблиц
- Асимметричный кластер
  - Конфигурация логических кластеров
  - Создание распределенных таблиц
- Кластер с локальным чтением без собственной георепликации
  - Конфигурация логических кластеров
  - Создание распределенных таблиц

Георaспределенный кластер — это кластер, серверы которого размещаются в нескольких географически удаленных дата-центрах. Такое решение позволяет обеспечить высокую надежность системы, которая будет сохранять работоспособность даже в случае выхода из строя одного из дата-центров.

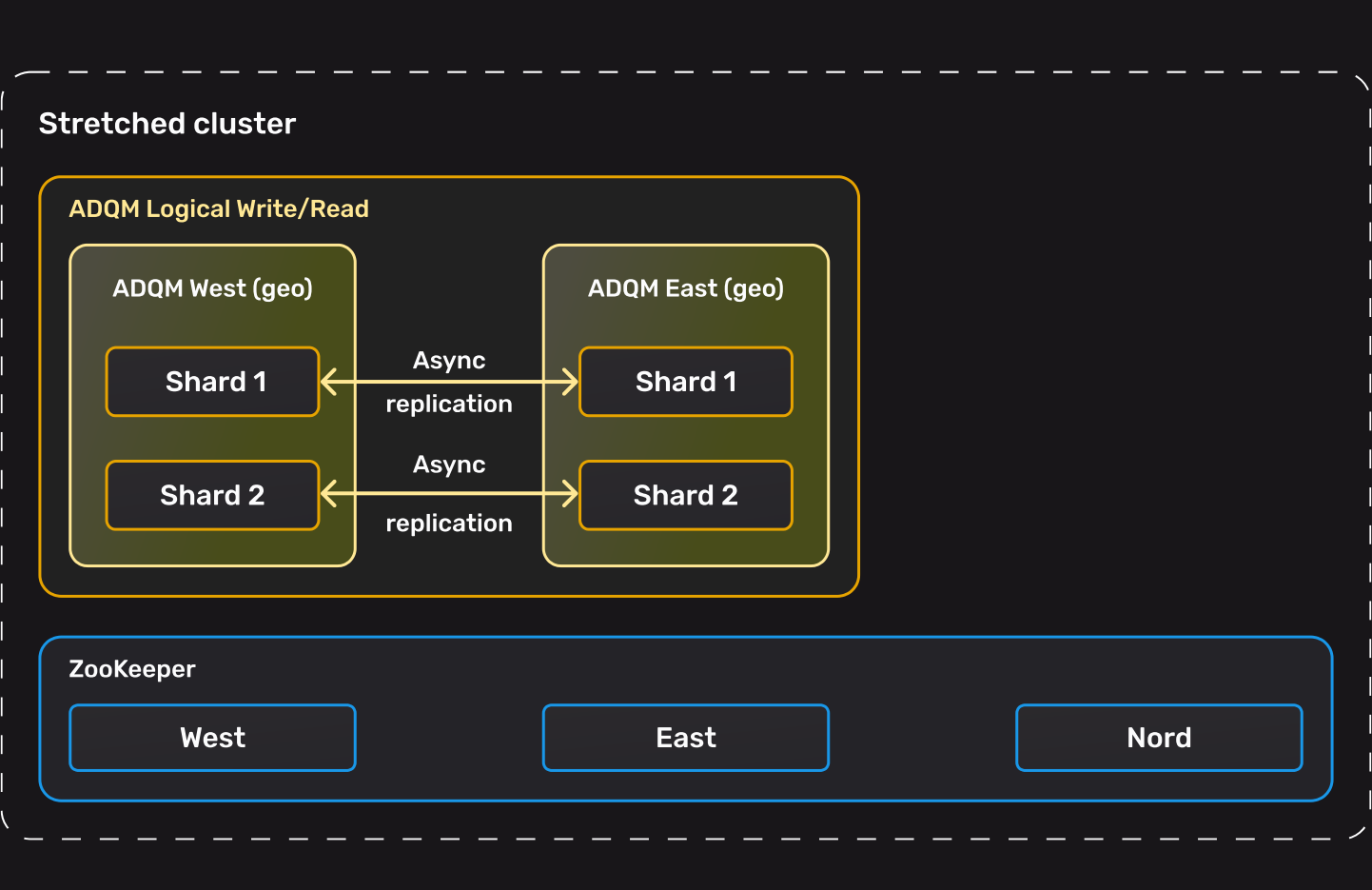
В этой статье приведены примеры возможных топологий георaспределенного кластера ADQM. Наиболее подходящая топология зависит от необходимого баланса в каждом конкретном случае между такими параметрами, как надежность, низкая задержка, дисковое хранилище и ремонтопригодность. Например, часто используются такие топологии, как Stretched-кластер, Stretched-кластер с локальным чтением и Stretched-кластер с локальной записью, обеспечивающие работоспособность кластера в условиях высокой латентности.

В каждом из приведенных ниже примеров серверы кластера расположены в двух дата-центрах — West и East.

Stretched-кластер

Особенности топологии:

- Запросы на чтение и вставку данных могут быть направлены в дата-центры West или East, при этом не гарантируется, что все операции чтения будут выполняться локально (возможно, запрос будет выполняться репликами в разных дата-центрах).
- Серверы ZooKeeper размещаются в дата-центрах West и East, а также в третьем дополнительном дата-центре Nord, так как ZooKeeper требует нечетного количества серверов.



Stretched-кластер

Конфигурация логического кластера

В настройках каждого сервера (*config.xml*) описывается кластер `stretched_cluster`, содержащий два шарда, каждый из которых состоит из двух реплик, находящихся в разных дата-центрах. Параметр шардов `internal_replication` устанавливается в `true` — в этом случае репликацию данных будет выполнять реплицируемая таблица, в которую данные записываются (то есть данные записываются на одну любую реплику шарда, вторая реплика получает данные автоматически).

config.xml

```xml
<remote_servers>
  <stretched_cluster>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>shard1.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>shard2.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </stretched_cluster>
</remote_servers>
```

Создание распределенных таблиц

На всех серверах кластера создается реплицируемая таблица `replicated_table` на базе движка ReplicatedMergeTree.

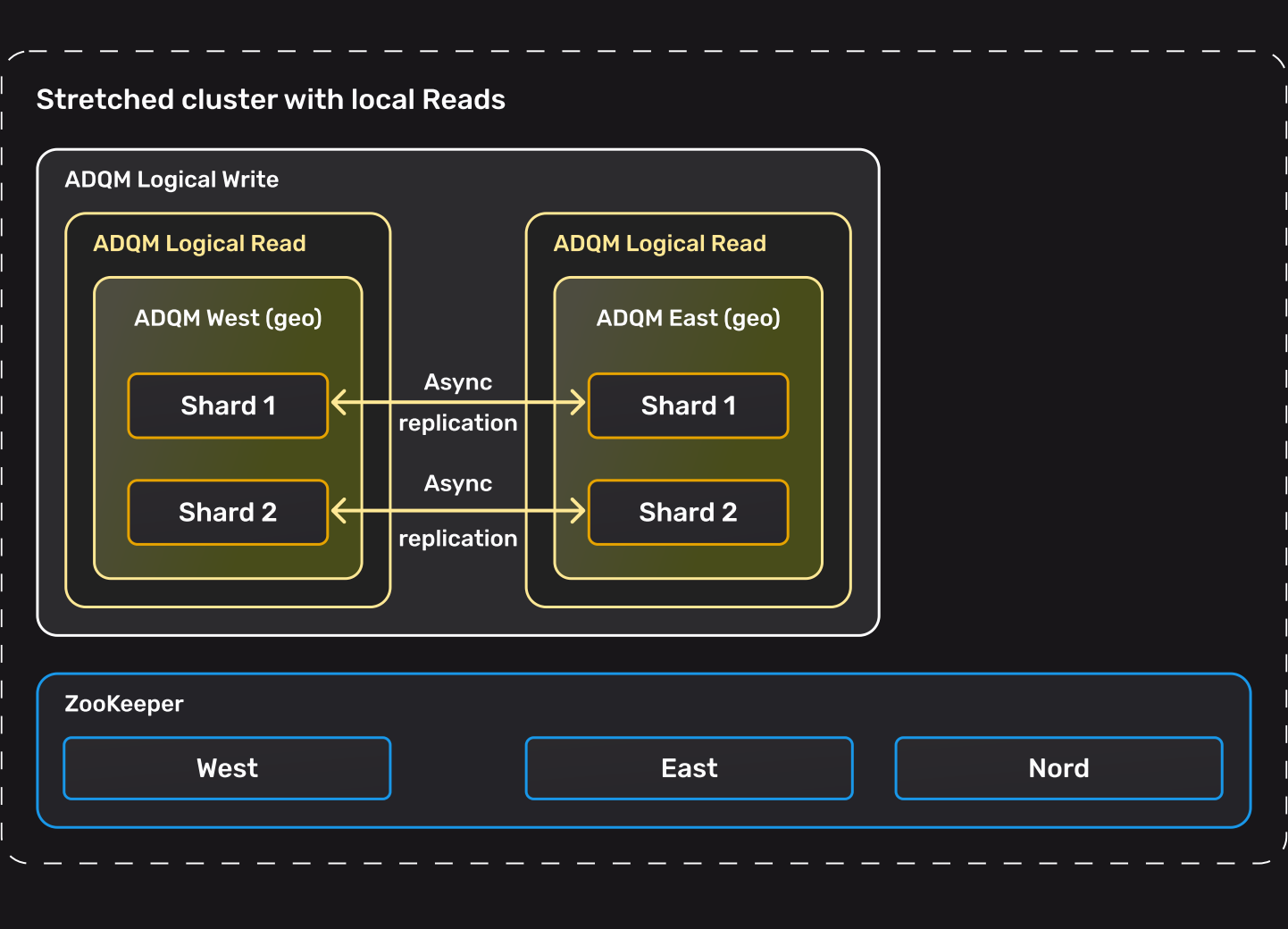
На всех серверах кластера на движке Distributed создается распределенная таблица, которая будет использоваться как для чтения, так и для записи. Для этого необходимо выполнить следующий запрос на одном из серверов:

```sql
CREATE TABLE table_distributed ON CLUSTER stretched_cluster AS replicated_table
ENGINE = Distributed(stretched_cluster, default, replicated_table);
```

Stretched-кластер с локальным чтением

Особенности топологии:

- Запросы на чтение и вставку данных могут быть направлены в дата-центры West или East, при этом гарантируется, что все операции чтения будут локальными.
- Серверы ZooKeeper размещаются в дата-центрах West и East, а также в третьем дополнительном дата-центре Nord, так как ZooKeeper требует нечетного количества серверов.



Stretched-кластер с локальным чтением

Конфигурация логических кластеров

В настройках каждого сервера (*config.xml*) описываются следующие кластеры:

- `stretched_cluster_writes` — включает серверы дата-центров West и East (два шарда, по две реплики в каждом), на которых будут выполняться операции записи данных;
- `stretched_cluster_west_reads` — включает серверы дата-центра West (два шарда, по одной реплике в каждом), на которых будут выполняться запросы на чтение;
- `stretched_cluster_east_reads` — включает серверы дата-центра East (два шарда, по одной реплике в каждом), на которых будут выполняться запросы на чтение.

Параметр шардов `internal_replication` устанавливается в `true` — в этом случае репликацию данных будет выполнять реплицируемая таблица, в которую данные записываются (то есть данные записываются на одну любую реплику шарда, вторая реплика получает данные автоматически).

config.xml

```xml
<remote_servers>
  <stretched_cluster_writes>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>shard1.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>shard2.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </stretched_cluster_writes>
  <stretched_cluster_west_reads>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
    </shard>
  </stretched_cluster_west_reads>
  <stretched_cluster_east_reads>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </stretched_cluster_east_reads>
</remote_servers>
```

Создание распределенных таблиц

На всех серверах кластера создается реплицируемая таблица `replicated_table` на базе движка ReplicatedMergeTree.

Распределенные таблицы для чтения создаются отдельно в каждом дата-центре:

- West:

```sql
CREATE TABLE table_distributed_west_reads ON CLUSTER stretched_cluster_west_reads AS replicated_table
ENGINE = Distributed(stretched_cluster_west_reads, default, replicated_table);
```

- East:

```sql
CREATE TABLE table_distributed_east_reads ON CLUSTER stretched_cluster_east_reads AS replicated_table
ENGINE = Distributed(stretched_cluster_east_reads, default, replicated_table);
```

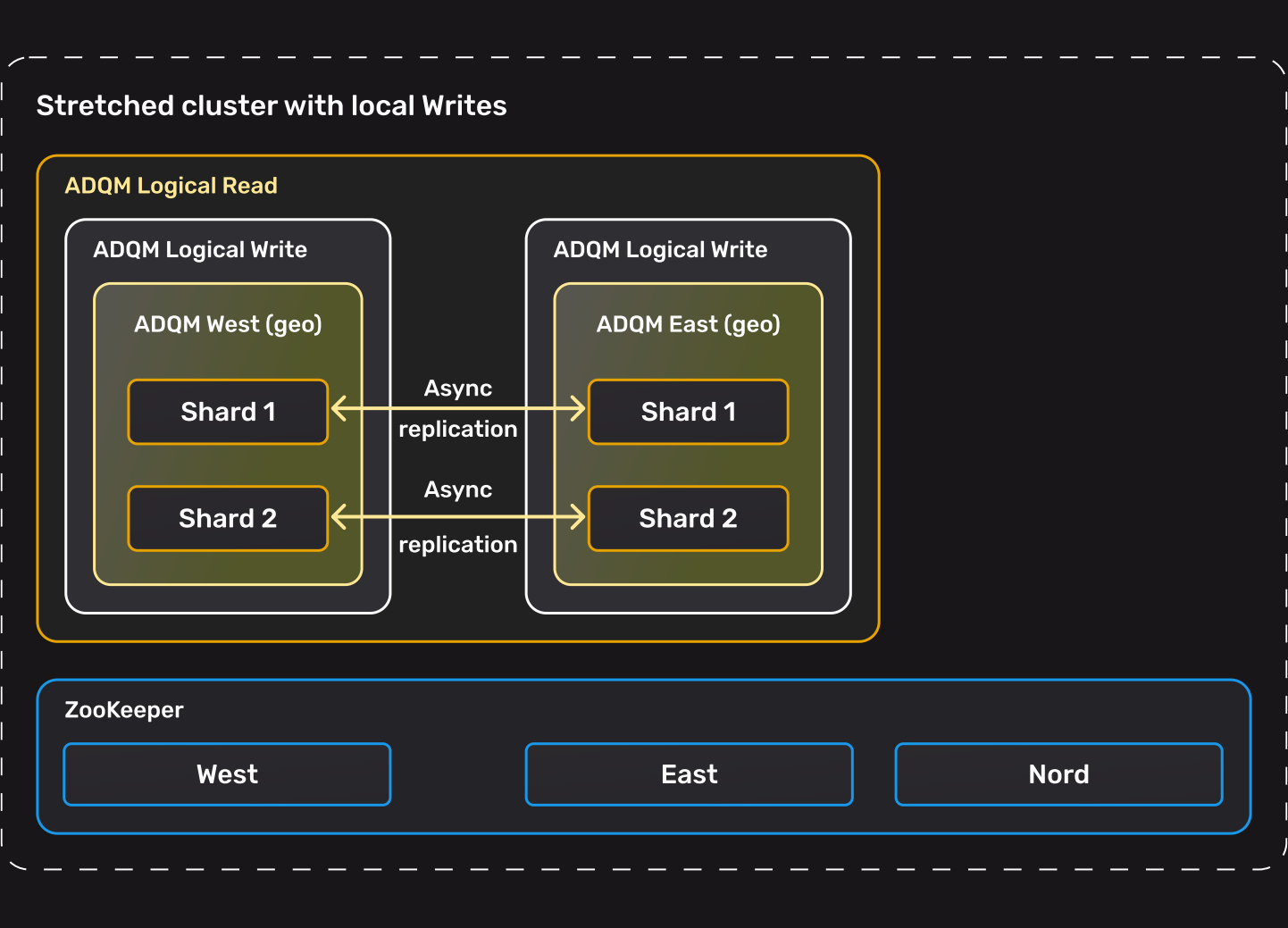
Для записи создаются идентичные распределенные таблицы в логическом кластере, объединяющем оба дата-центра:

```sql
CREATE TABLE table_distributed_writes ON CLUSTER stretched_cluster_writes AS replicated_table
ENGINE = Distributed(stretched_cluster_writes, default, replicated_table);
```

Stretched-кластер с локальной записью

Особенности топологии:

- Запросы на чтение и вставку данных могут быть направлены в дата-центры West или East, при этом гарантируется, что все операции записи данных будут выполняться локально.
- Серверы ZooKeeper размещаются в дата-центрах West и East, а также в третьем дополнительном дата-центре Nord, так как ZooKeeper требует нечетного количества серверов.



Stretched-кластер с локальной записью

Конфигурация логических кластеров

В настройках каждого сервера (*config.xml*) описываются следующие кластеры:

- `stretched_cluster_reads` — включает серверы дата-центров West и East (два шарда, по две реплики в каждом), на которых будут выполняться запросы на чтение;
- `stretched_cluster_west_writes` — включает серверы дата-центра West (два шарда, по одной реплике в каждом), на которых будут выполняться операции записи данных;
- `stretched_cluster_east_writes` — включает серверы дата-центра East (два шарда, по одной реплике в каждом), на которых будут выполняться операции записи данных.

Параметр шардов `internal_replication` устанавливается в `true` — в этом случае репликацию данных будет выполнять реплицируемая таблица, в которую данные записываются (то есть данные записываются на одну любую реплику шарда, вторая реплика получает данные автоматически).

config.xml

```xml
<remote_servers>
  <stretched_cluster_reads>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>shard1.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>shard2.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </stretched_cluster_reads>
  <stretched_cluster_west_writes>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
    </shard>
  </stretched_cluster_west_writes>
  <stretched_cluster_east_writes>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </stretched_cluster_east_writes>
</remote_servers>
```

Создание распределенных таблиц

На всех серверах кластера создается реплицируемая таблица `replicated_table` на базе движка ReplicatedMergeTree.

Распределенные таблицы для записи создаются отдельно в каждом дата-центре:

- West:

```sql
CREATE TABLE table_distributed_west_writes ON CLUSTER stretched_cluster_west_writes AS replicated_table
ENGINE = Distributed(stretched_cluster_west_writes, default, replicated_table);
```

- East:

```sql
CREATE TABLE table_distributed_east_writes ON CLUSTER stretched_cluster_east_writes AS replicated_table
ENGINE = Distributed(stretched_cluster_east_writes, default, replicated_table);
```

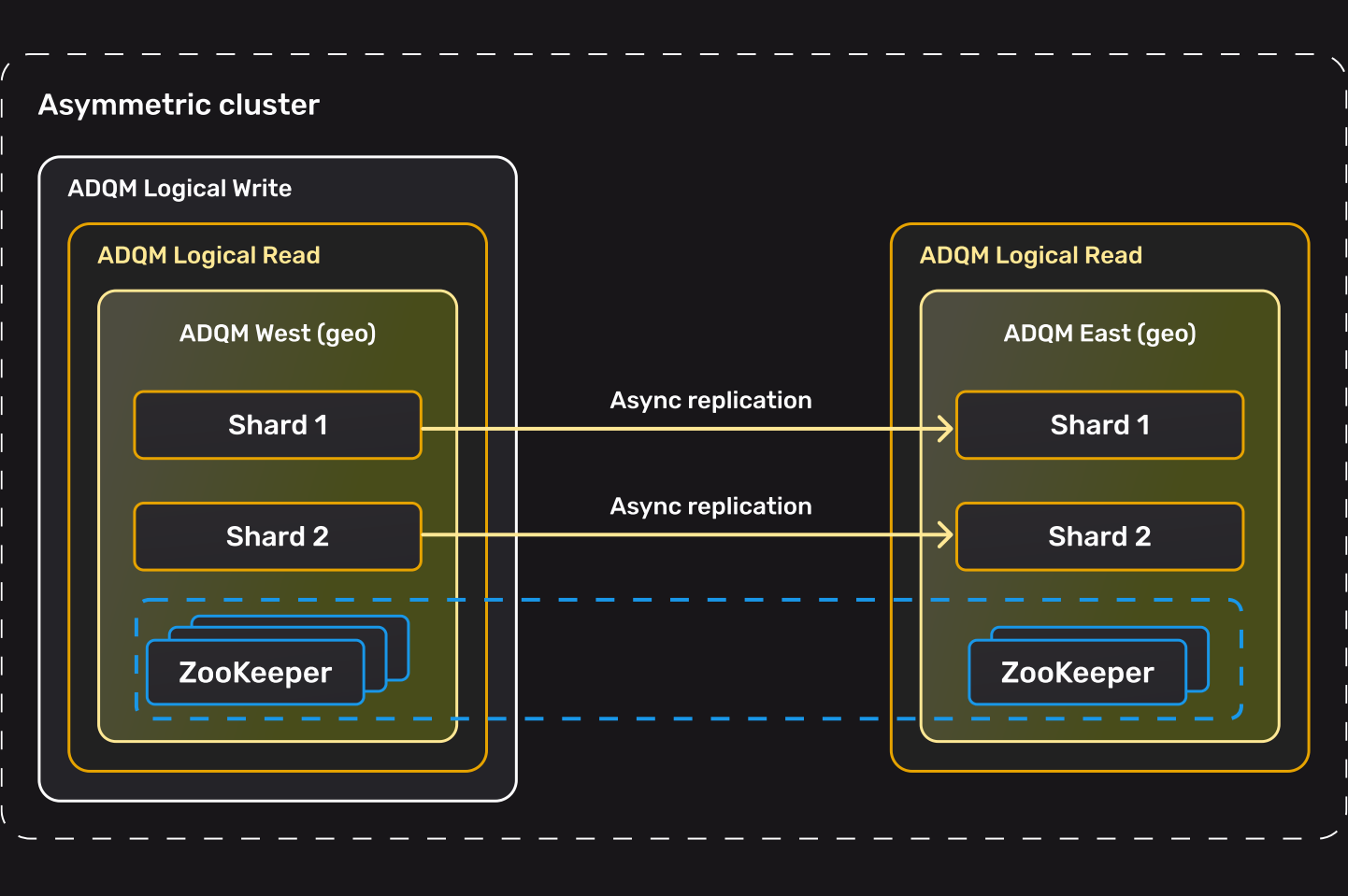
Для чтения создаются идентичные распределенные таблицы в логическом кластере, объединяющем оба дата-центра:

```sql
CREATE TABLE table_distributed_reads ON CLUSTER stretched_cluster_reads AS replicated_table
ENGINE = Distributed(stretched_cluster_reads, default, replicated_table);
```

Асимметричный кластер

Особенности топологии:

- Чтение данных возможно как в дата-центре West, так и в East, а запись — только в West.
- Серверы одного ансамбля ZooKeeper расположены в двух дата-центрах (третий дата-центр не нужен), при этом в кворуме участвуют только серверы дата-центра West. Реплики в дата-центре East не могут быть лидерами (это управляется настройкой `replicated_can_become_leader=0` в секции `merge_tree` файла *config.xml*).
- Параметр insert_quorum установлен в `1`, так как число серверов в кворуме должно быть равно или меньше, чем количество реплик в основном дата-центре (в данном примере это дата-центр West, в который записываются данные).



Асимметричный кластер

Конфигурация логических кластеров

В настройках каждого сервера (*config.xml*) описываются следующие кластеры:

- `asymmetric_cluster_west` — включает серверы дата-центра West, которые распределены в два шарда, по одной реплике в каждом;
- `asymmetric_cluster_east` — включает серверы дата-центра East, которые распределены в два шарда, по одной реплике в каждом.

Параметр шардов `internal_replication` устанавливается в `true` — в этом случае репликацию данных будет выполнять реплицируемая таблица, в которую данные записываются (то есть данные записываются на одну любую реплику шарда, вторая реплика получает данные автоматически).

config.xml

```xml
<remote_servers>
  <asymmetric_cluster_west>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
    </shard>
  </asymmetric_cluster_west>
  <asymmetric_cluster_east>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </asymmetric_cluster_east>
</remote_servers>
```

Создание распределенных таблиц

На всех серверах кластера создается реплицируемая таблица `replicated_table` на базе движка ReplicatedMergeTree.

Распределенные таблицы для чтения создаются в каждом дата-центре:

- West:

```sql
CREATE TABLE table_distributed_west_reads ON CLUSTER asymmetric_cluster_west AS replicated_table
ENGINE = Distributed(asymmetric_cluster_west, default, replicated_table);
```

- East:

```sql
CREATE TABLE table_distributed_east_reads ON CLUSTER asymmetric_cluster_east AS replicated_table
ENGINE = Distributed(asymmetric_cluster_east, default, replicated_table);
```

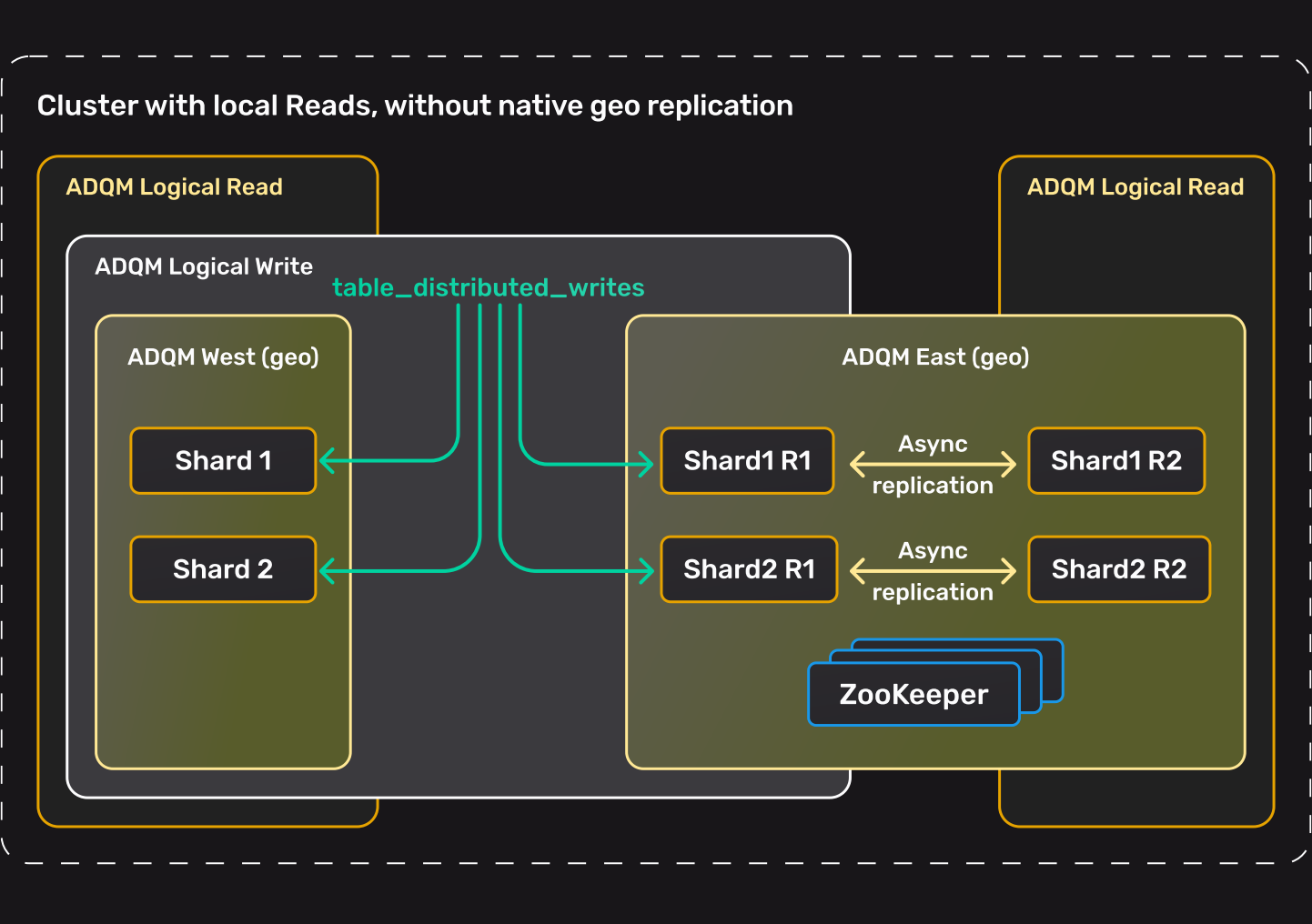
Распределенная таблица для записи создается только в дата-центре West:

```sql
CREATE TABLE table_distributed_west_writes ON CLUSTER asymmetric_cluster_west AS replicated_table
ENGINE = Distributed(asymmetric_cluster_west, default, replicated_table);
```

Кластер с локальным чтением без собственной георепликации

Особенности топологии:

- Не поддерживается нативная репликация ClickHouse между дата-центрами. Репликация между дата-центрами обеспечивается через распределенные таблицы.
- Параметр insert_quorum устанавливается равным `2`, чтобы при выполнении запроса `INSERT` распределенная таблица гарантированно записывала данные на две реплики, расположенные в разных дата-центрах.
- Запросы на чтение и вставку данных могут быть направлены в дата-центры East или West, при этом гарантируется, что все операции чтения будут локальными.
- Допускается большая задержка между West и East.
- ZooKeeper доступен только в одном дата-центре.



Кластер с локальным чтением без собственной георепликации

Конфигурация логических кластеров

В настройках каждого сервера (*config.xml*) описываются следующие кластеры:

- `cluster_writes` — включает серверы дата-центров West и East (два шарда, по две реплики в каждом), на которых будут выполняться операции записи данных;
- `cluster_west_reads` — включает серверы дата-центра West (два шарда, по одной реплике в каждом), на которых будут выполняться запросы на чтение данных;
- `cluster_east_reads` — включает серверы дата-центра East (два шарда, по одной реплике в каждом), на которых будут выполняться запросы на чтение данных.

Параметр `internal_replication` для шардов кластера `cluster_writes`, в который будут записываться данные, устанавливается в `false`, чтобы данные реплицировались через распределенную таблицу.

config.xml

```xml
<remote_servers>
  <cluster_writes>
    <shard>
      <internal_replication>false</internal_replication>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>sh1r1.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>false</internal_replication>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>sh2r1.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </cluster_writes>
  <cluster_west_reads>
    <shard>
      <replica>
        <host>shard1.west</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <replica>
        <host>shard2.west</host>
        <port>9000</port>
      </replica>
    </shard>
  </cluster_west_reads>
  <cluster_east_reads>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard1r2.east</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <internal_replication>true</internal_replication>
      <replica>
        <host>shard2r2.east</host>
        <port>9000</port>
      </replica>
    </shard>
  </cluster_east_reads>
</remote_servers>
```

Создание распределенных таблиц

На всех серверах кластера создается реплицируемая таблица `replicated_table` на базе движка ReplicatedMergeTree.

Распределенная таблица для чтения создается в дата-центре West на каждом сервере отдельно, так как чтобы работал механизм `ON CLUSTER`, требуется ZooKeeper. Для этого необходимо выполнить следующий запрос на всех серверах дата-центра West:

```sql
CREATE TABLE table_distributed_reads AS replicated_table
ENGINE = Distributed(cluster_west_reads, default, replicated_table);
```

Чтобы создать распределенную таблицу для чтения в дата-центре East, достаточно выполнить следующий запрос на любом сервере:

```sql
CREATE TABLE table_distributed_reads ON CLUSTER cluster_east_reads AS replicated_table
ENGINE = Distributed(cluster_east_reads, default, replicated_table);
```

Распределенная таблица для записи должна быть создана на серверах *shard1.west*, *shard2.west*, *shard1r1.east*, *shard2r1.east*:

```sql
CREATE TABLE table_distributed_writes AS replicated_table
ENGINE = Distributed(cluster_writes, default, replicated_table);
```

Проксирование

Елена Дворядкина

Содержание

Назад к содержанию

Установка Chproxy

Конфигурация Chproxy через ADCM

Для балансировки нагрузки и распределения доступа к базе данных ADQM можно использовать Chproxy — HTTP-прокси, который может обеспечить:

- равномерное распределение запросов между всеми узлами кластера ADQM (предполагается, что все узлы в кластере равноценны);
- проверку работоспособности узлов кластера и предотвращение отправки запросов на недоступные узлы;
- ограничение доступа по HTTP списком разрешенных IP-адресов/IP-масок;
- безопасную авторизацию пользователей — Chproxy может сопоставлять пользователей, отправляющих запросы, с пользователями кластера ADQM, не раскрывая реальные имена пользователей и пароли, используемые в ADQM.

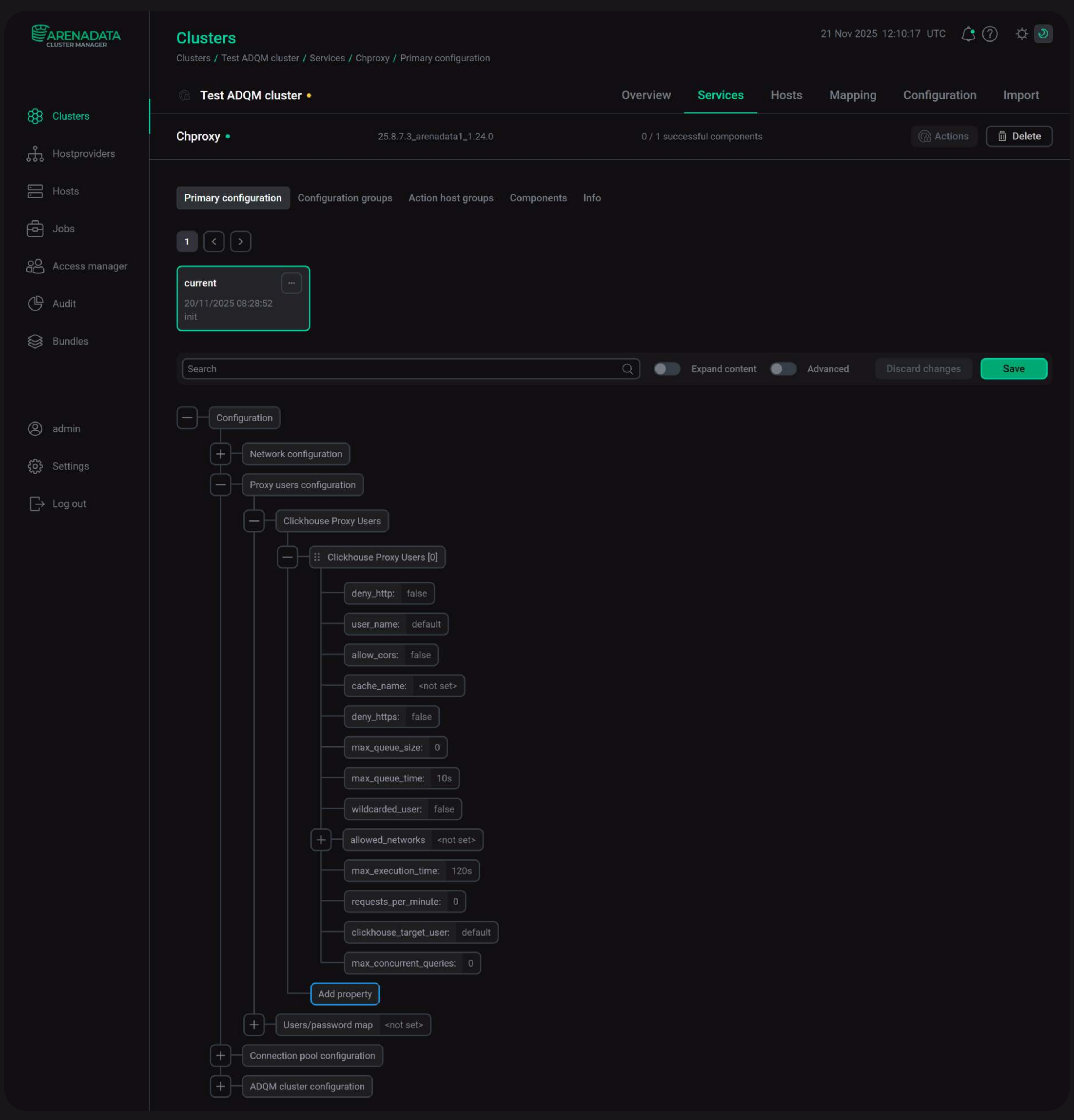
Установка Chproxy

Chproxy входит в список сервисов ADQM версии Enterprise (начиная с версии 2.3.7.28.1.b) и может быть установлен в кластер ADQM через интерфейс ADCM, как другие сервисы. Для этого необходимо:

1. обавить сервис Chproxy в класте
2. обавить компонент сервиса на один или несколько хостов кластер
3. астроить серви
4. становить сервис в класте

Конфигурация Chproxy через ADCM

Конфигурационные параметры сервиса Chproxy могут быть настроены в соответствующем окне интерфейса ADCM. Статья астройка сервисо содержит подробные инструкции как перейти к окну конфигурации сервиса.



ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Services / Chproxy / Primary configuration
21 Nov 2025 12:10:17 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Chproxy
25.8.7.3_arenadata1_1.24.0
0 / 1 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
1
current
20/11/2025 08:28:52
init
Search
Expand content
Advanced
Discard changes
Save
Configuration
Network configuration
Proxy users configuration
Clickhouse Proxy Users
Clickhouse Proxy Users [0]
Add property
Users/password map <not set>
Connection pool configuration
ADQM cluster configuration

deny_http: false
user_name: default
allow_cors: false
cache_name: <not set>
deny_https: false
max_queue_size: 0
max_queue_time: 10s
wildcarded_user: false
allowed_networks <not set>
max_execution_time: 120s
requests_per_minute: 0
clickhouse_target_user: default
max_concurrent_queries: 0

Конфигурация сервиса Chproxy

Для сервиса Chproxy доступны следующие параметры:

- *Network configuration*:
  - *Enable HTTP* — прослушивание HTTP-запросов (по умолчанию включено);
  - *HTTP listen port* — порт, на котором сервер слушает HTTP-запросы (по умолчанию `9090`);
  - *HTTP allowed networks* — список сетей, из которых разрешен доступ по HTTP (можно указать IP-адрес или маску подсети, по умолчанию добавлено значение `127.0.0.1`);
  - *Enable HTTPS* — прослушивание HTTPS-запросов (по умолчанию выключено);
  - *HTTPS listen port* — порт, на котором сервер слушает HTTPS-запросы (по умолчанию `8543`);
  - *HTTPS allowed networks* — список сетей, из которых разрешен доступ по HTTPS (можно указать IP-адрес или маску подсети, по умолчанию добавлено значение `127.0.0.1`);
  - *Certificate file* — путь к файлу серверного SSL-сертификата;
  - *Private key file* — путь к файлу закрытого ключа серверного SSL-сертификата.

- *Proxy users configuration*:
  - *Clickhouse Proxy Users* — конфигурации пользователей, запросы которых будут проксироваться в ADQM. С помощью параметров в этой секции для каждого пользователя можно настроить ограничение доступа по списку IP-адресов или масок подсетей, запретить или разрешить подключения по HTTP и HTTPS, включить кеширование ответов, установить ограничение продолжительности и частоты запросов, а также количества одновременно выполняемых запросов.

    Для сопоставления нескольких различных входящих пользователей с одним пользователем ADQM можно создать конфигурацию wildcarded-пользователя. Для этого необходимо включить опцию *wildcarded_user*, в качестве значения параметра *user_name* указать шаблон имени пользователя Chproxy вида `<prefix>*` или `*<suffix>` и в поле *clickhouse_target_user* указать пользователя ADQM. Например, если `user_name = dba_*` и `clickhouse_target_user = db_admin` (при этом `db_admin` — уществующий пользователь ADQ), то любой пользователь с именем, соответствующим этому шаблону (например, `dba_john` и `dba_mary`), будет авторизован в ADQM как `db_admin`.
  - *Users/password map* — пароли пользователей *Clickhouse Proxy Users*.

- *Connection pool configuration*:
  - *Max idle connections* — максимальное количество неиспользуемых соединений в пуле;
  - *Max idle connections per host* — максимальное количество неиспользуемых соединений на хост.

- *ADQM cluster configuration*:
  - *Interval* — интервал проверки всех узлов кластера на доступность (по умолчанию `5s`);
  - *Timeout* — время ожидания ответа от узлов кластера (по умолчанию `5s`).

  Эти параметры используются для проверки работоспособности узлов кластера. Запрос на проверку работоспособности отправляется компонентами сервиса Chproxy всем узлам кластера ADQM с периодичностью, которую определяет параметр *Interval*. Если на такой запрос не получен ответ от какого-то узла кластера в течение времени, заданного параметром *Timeout*, Chproxy прекратит отправку запросов на этот узел.

Требования к оборудованию

Елена Дворядкина

Содержание

Назад к содержанию

- ClickHouse-сервер
- Zookeeper-сервер
- Arenadata Enterprise Tools
- Arenadata Monitoring
- ADQM Control

ВАЖНО

Системные требования, приведенные ниже, являются **минимальными**. Целевой сайзинг необходимо рассчитывать исходя из требований вашей организации.

ClickHouse-сервер

Минимальные аппаратные требования к ClickHouse-серверам для установки кластера ADQM следующие:

- Физический сервер.
- Хранение: отформатированное в *ext4* устройство, примонтированное к файловой системе.
- CPU: от 8 ядер.
- RAM: от 64 ГБ.
- Сеть: сервера должны быть доступны по портам 9000, 8123, 9009 внутри кластера.

Для быстрых витрин (quick marts) подходит массив из SSD-дисков. Для хранения больших таблиц фактов (big tables with facts) предпочтителен RAID 10 из HDD. Рекомендуется использовать программный RAID в Linux (mdadm). Не рекомендуется использовать LVM. При создании RAID 10 необходимо выбрать far-расположение.

На каждом ClickHouse-сервере внутри кластера схема устройства, диски и точка монтирования должны быть одинаковыми.

Примеры требований к ClickHouse-серверам в зависимости от профиля нагрузки кластера ADQM показаны ниже.

Proof of concept

Количество узлов	Хранение	CPU	RAM	Сеть
1-4	1 x 500 ГБ HDD для ОС 4 x 1.2 ТБ SAS HDD RAID 10 для данных	8 ядер	96 ГБ	1 x 10 Гбит/с

Quick marts

Количество узлов	Хранение	CPU	RAM	Сеть
4	2 x 500 ГБ SSD RAID 1 для ОС 12 x 1 ТБ SAS12G SSD MixedUse RAID 10 (PCIe Gen3 x8) для данных	24 ядра (2 x 12)	256 ГБ (8 x 32)	4 x 25 Гбит/с (2 адаптера PCIe Gen4 x4)

Big tables with facts

Количество узлов	Хранение	CPU	RAM	Сеть
4	2 x 500 ГБ SSD RAID 1 для ОС 12 x 1.8 ТБ SAS10K HDD RAID 10 для данных	24 ядра	256 ГБ	2 x 25 Гбит/с (2 адаптера PCIe Gen3 x4)

Zookeeper-сервер

Минимальные аппаратные требования к серверам с Zookeeper кластера ADQM следующие:

- Физический или виртуальный сервер. Для пилотных инсталляций можно использовать те же сервера, что и для ClickHouse. При больших инсталляциях — минимум три железных сервера.
- Хранение: от 100 ГБ; для больших кластеров предпочтительны быстрый SSD-диск или RAID 1.
- CPU: от 4 ядер.
- RAM: от 4 ГБ.

В таблице ниже представлен пример конфигурации Zookeeper-сервера.

Количество узлов	Хранение	CPU	RAM	Сеть
3 и более	2 x 500 ГБ SSD RAID 1 для ОС 2 x 2.4 ТБ SSD RAID 1 для данных	4 ядра (P-cores)	8 ГБ	2 x 10 Гбит/с

ВАЖНО

Для обеспечения HA (High Availability) необходимо, чтобы количество экземпляров Zookeeper-серверов было нечетным.

Arenadata Enterprise Tools

Если вместе с вашим продуктом вы используете ET, примите во внимание следующие рекомендации:

- Виртуальный сервер.
- CPU: от 4 ядер.
- RAM: от 8 ГБ.
- Диск: от 50 ГБ.

Arenadata Monitoring

Если вместе с вашим продуктом вы используете ADM, примите во внимание следующие рекомендации:

- Виртуальный сервер.
- CPU: от 8 ядер.
- RAM: от 16 ГБ.
- Диск: от 10 ГБ на каждый отслеживаемый хост.

ADQM Control

Если вместе с вашим продуктом вы используете ADQM Control, примите во внимание следующие требования к хостам для установки компонентов ADQM Control:

- Backend:
  - CPU: от 4 ядер.
  - RAM: от 8 ГБ.
  - Диск: SSD от 50 ГБ.
- Хранилище данных (PostgreSQL):
  - CPU: от 4 ядер.
  - RAM: от 8 ГБ.
  - Диск: SSD от 50 ГБ в зависимости от нагрузки ADQMDB.

ПРИМЕЧАНИЕ

Поддерживается работа ADQM Control с кластерами ADQM версии не старше одного года.

Требования к сети

Елена Дворядкина

Назад к содержанию

- Требуется одна или несколько выделенных и обособленных одноранговых Interconnect-сетей под цели внутренней коммуникации кластера.
- К Interconnect-сетям должны быть подключены все серверы кластера.
- Все серверы должны иметь полные доменные имена (FQDN) и возможность обмениваться данными друг с другом на заданных скоростях.
- Должен быть известен IP-адрес каждого сервера.
- Должна быть обеспечена доступность Ping любого из серверов (ICMP).
- В случае инсталляции ADQM с доступом в Интернет требуется доступ к развернутому на выделенном сервере Arenadata Cluster Manager. Установка Arenadata Cluster Manager описана в статье Установка ADCM.
- В случае инсталляции ADQM без доступа в Интернет (offline) требуется доступ к развернутому на выделенном сервере Arenadata Enterprise Tools. Установка Enterprise Tools описана в разделе Установка кластера Enterprise Tools.
- На серверах, используемых для установки кластера, должны быть открыты порты, приведенные ниже.

ПРИМЕЧАНИЕ

Каждый порт относится к серверу, на который устанавливается конкретный сервис.

ADQM

Сервис	Порт	Назначение
Общие	22	Порт для подключения к SSH-провайдеру
	2015	TCP-порт для отправки метрик в кластер мониторинга ADM (depricated)
	2016	UDP-порт для отправки метрик в кластер мониторинга ADM (depricated)
ClickHouse	8123	HTTP-порт для клиентских или прокси-соединений
	8443	Порт для защищенных клиентских соединений через интерфейс HTTPS (см. Шифрование соединений с базой данных)
	9000	Порт для клиентских соединений по протоколу Native (TCP). Этот порт также используется для межсерверной коммуникации при выполнении распределенных запросов
	9440	TCP-порт для SSL-защищенных соединений с клиентами и межсерверной коммуникации при выполнении распределенных запросов
	9009	Порт для обмена данными между репликами
	9100	Порт для клиентских соединений через интерфейс gRPC
	9019	Порт JDBC Bridge
	9363	Порт доступа к метрикам ClickHouse в формате Prometheus
ZooKeeper	2181	Порт доступа к сервису
	2888 3888	Порты межсерверного взаимодействия для компонентов кворума
	2281	Порт для защищенных клиентских соединений
	9020	Порт доступа к метрикам ZooKeeper в формате Prometheus
ClickHouse Keeper	2129	TCP-порт для клиентских подключений
	9444	Рекомендуемый порт для соединений RAFT (также используется для безопасного обмена данными, если включена настройка `<secure>1</secure>`)
	9281	Рекомендуемый порт для SSL-защищенного взаимодействия
	9010	Порт доступа к метрикам ClickHouse Keeper в формате Prometheus
Monitoring	11210	Порт доступа к web-интерфейсу Grafana
	11203	Порт доступа к системным метрикам хоста ADQM в формате Prometheus
Chproxy	9090	HTTP-порт для подключения клиентов
	8543	HTTPS-порт для подключения клиентов

Monitoring

Порт	Назначение
3000	HTTP-порт доступа к web-интерфейсу Grafana. Значение по умолчанию может быть изменено при установке кластера мониторинга на странице конфигурации сервиса Grafana в разделе ***ip_and_ports***
80	TCP-порт доступа к web-интерфейсу Graphite. Значение по умолчанию может быть изменено при установке кластера мониторинга на странице конфигурации сервиса Graphite в разделе ***ip_and_ports***

Enterprise Tools

Порт	Назначение
443	TCP-порт, используемый Arenadata Enterprise Tools (служба реестра Docker)
81	TCP-порт Arenadata Enterprise Tools (доступ к репозиториям в случае offline-установки)

ADCM

Порт	Назначение
8000	TCP-порт для отправки статусов компонентов кластера в ADCM

Программные требования

Елена Дворядкина

Содержание

Назад к содержанию

Операционная система

Необходимая информация для установки кластера

Особенности установки ADQM

Операционная система

1. ADQM поддерживает только архитектуру x86_64 и следующие операционные системы:
   - Centos 7
   - RHEL 7
   - Alt Linux 8.4 SP (начиная с версии ADQM 22.3.2.2.1.b1), 10.2 (начиная с версии ADQM 25.3.2.39.1.b1)
   - Astra Linux SE 1.7 Орел с Axiom JDK (для Enterprise-версии ADQM, начиная с 22.8.12.45.1.b1), 1.7.6 Воронеж (начиная с версии ADQM 25.3.2.39.1.b1)
   - РЕД ОС 7.3 Сертифицированная редакция (для Enterprise-версии ADQM, начиная с 23.3.2.37.1.b1)
   - Ubuntu 22.04 (начиная с версии ADQM 24.3.2.23.2.b1)
2. Необходимо подготовить все зависимые репозитории дополнительно к выбранной операционной системе.
3. Со всех машин в кластере должен быть доступ к официальным репозиториям (или созданы локальные зеркала):
   - CentOS Extras
   - CentOS Updates
   - CentOS Base
4. В случае инсталяции ADQM с доступом в Интернет машины должны иметь доступ к репозиториям Arenadata, перечисленным на вкладке Configuration в меню кластера ADQM web-интерфейса ADCM. Для offline-установки ADQM необходим доступ к хосту, на котором развернут кластер Enterprise Tools.

Необходимая информация для установки кластера

Необходимая информация и условия для установки кластера ADQM:

- IP-адреса и hostnames всех серверов в кластере, сопоставленные с их ролями.
- Доступ к пользователю `root` или любому пользователю с правами `sudo` на каждом сервере внутри кластера.
- Правильно работающий DNS. Все hostname и FQDN должны одинаково резолвиться на всех серверах внутри кластера.

Особенности установки ADQM

1. В процессе установки модифицируются файлы, которые не должны контролироваться системой управления конфигурацией (при ее наличии):
   - системный файл */etc/hosts*;
   - системный файл */etc/selinux/config*;
   - системный файл */etc/sysctl.conf*.
2. В процессе установки создаются файлы в */usr/lib/systemd/system/*, которые не должны контролироваться системой управления конфигурацией (при ее наличии).
3. Сервисы, устанавливаемые в кластере, не должны контролироваться системой управления конфигурацией (при ее наличии).

Установка

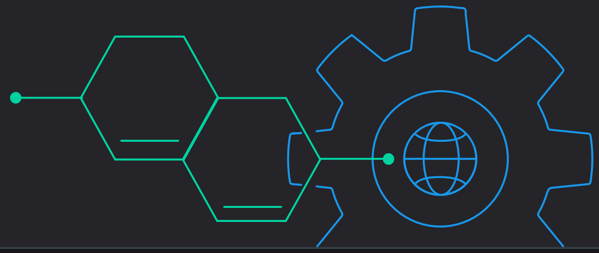
Назад к содержанию

Этот раздел содержит описание установки кластера Arenadata QuickMarts (ADQM). Для установки рекомендуется использовать Arenadata Cluster Manager (ADCM) — отдельный продукт Arenadata, предназначенный для простого, удобного и быстрого развертывания и дальнейшей эксплуатации программного обеспечения.

Возможны два способа установки ADQM через ADCM.

Online-установка

на хостах с доступом к Интернету



Offline-установка

на хостах без доступа к Интернету

Online-установка

Назад к содержанию

Этот раздел посвящен **online**-установке ADQM. Данный способ установки подходит для хостов с доступом к Интернету.

Процесс online-установки через ADCM включает следующие шаги.

Шаг 1.

Установка ADCM

Шаг 2.

Подготовка хостов

ARENADATA
QUICKMARTS

Шаг 3.

Установка кластера ADQM

Шаг 4.

Установка мониторинга

ARENADATA
ADQM CONTROL

Шаг 5.

Установка ADQM Control

Установка ADCM

Дарья Барышева

Содержание

Назад к содержанию

- Шаг 1. Предварительные требования
  - Требования к software и hardware
  - Установка Docker
  - Отключение SELinux
  - Остановка firewalld
  - Настройка доступа к Docker CLI без прав root
- Шаг 2. Установка ADCM
- Шаг 3. Запуск ADCM
- Шаг 4. Проверка установки
  - Проверка статуса Docker-контейнера
  - Проверка доступности порта 8000
  - Проверка web-интерфейса ADCM
  - Установка URL ADCM
- Шаг 5. Обновление ADCM

Для развертывания ADCM требуется отдельный хост. Перед началом установки рекомендуем ознакомиться с документацией ADCM.

Этапы установки описаны ниже.

Шаг 1. Предварительные требования

Требования к software и hardware

ADCM поставляется в виде Docker-образа. Для его установки необходимы:

- Одна из приведенных ниже операционных систем:
  - Ubuntu 22.04
  - CentOS 7
  - RHEL 7
  - Alt Linux SP 8
  - Alt Linux SP 10.2
  - Astra Linux SE 1.7 Орел
  - Astra Linux 1.7.6 Воронеж
  - РЕД ОС 7.3 Сертифицированная редакция
- Пользователь с правами `sudo`.
- Доступ к официальным репозиториям CentOS Extras и CentOS Base. По умолчанию, эти репозитории доступны в операционной системе. Однако, если они были отключены, потребуется их предварительное включение (например, путем редактирования конфигурационного файла в директории */etc/yum.repos.d/*).
- Менеджер программных пакетов YUM/APT.
- Хост, отвечающий следующим требованиям:

> ВАЖНО
>
> Системные требования, приведенные ниже, являются **минимальными**. Целевой сайзинг необходимо рассчитывать исходя из требований вашей организации.

  - **CPU**: 4 ядра CPU;
  - **RAM**: 16 ГБ;
  - **HDD**: 50 ГБ.

> ВАЖНО
>
> Начиная с версии ADCM 2.7.0 для установки ADPG без подключения к Интернету и использования его в качестве базы данных ADCM требуется любой Container Registry Client с загруженными образами ADPG, ADCM и ADCM Installer. Установка выполняется с помощью ADCM Installer с указанием URL локального хранилища образов ADCM и ADPG.

Чтобы пользователь обладал необходимыми правами для установки ADCM, выполните следующее:

1. Предоставьте пользователю привилегии `root`:

CentOS 7/RHEL 7/РЕД ОС 7.3/Alt Linux SP 8

```
$ sudo usermod -aG wheel <user>
```

где `<user>` — имя пользователя.

Ubuntu 22.04/Astra Linux SE 1.7 Орел

```
$ sudo usermod -aG sudo <user>
```

где `<user>` — имя пользователя.

2. Убедитесь, что в файле */etc/sudoers* есть запись, указанная ниже в зависимости от используемой операционной системы, с помощью следующей команды:

```
$ sudo vi /etc/sudoers
```

Чтобы вернуться в терминал, нажмите `Esc`, введите `:q!` и нажмите `Enter`.

CentOS 7/RHEL 7/РЕД ОС 7.3

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Ubuntu 22.04/Astra Linux SE 1.7 Орел

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Alt Linux SP 8

```
## Uncomment to allow members of group wheel to execute any command
WHEEL_USERS ALL=(ALL:ALL) ALL
```

Установка Docker

> ВНИМАНИЕ
>
> Установка Docker с официального сайта не рекомендуется. Вместо этого используйте официальные репозитории дистрибутива Linux. Репозитории, предоставляемые разработчиками операционных систем, как правило, более стабильны и безопасны.

Ниже приведены шаги, необходимые для установки Docker, на примере нескольких поддерживаемых в ADCM операционных систем. Для других версий операционных систем, отличных от указанных во вкладках, шаги могут отличаться.

CentOS 7/RHEL 7

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo yum update -y
```

2. Установите пакеты, необходимые для Docker:

```
$ sudo yum install -y yum-utils docker device-mapper-persistent-data lvm2
```

где:

- `yum-utils` — пакет, который предоставляет утилиту `yum-config-manager`;
- `device-mapper-persistent-data` и `lvm2` — пакеты, необходимые драйверу хранилища devicemapper.

3. Запустите Docker:

```
$ sudo systemctl start docker
```

4. Включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker
```

Ubuntu 22.04

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo apt-get update
```

2. Установите пакеты, необходимые для Docker:

```
$ sudo apt-get install ca-certificates curl
```

где:

- `ca-certificates` — пакет корневых сертификатов;
- `curl` — пакет для работы с веб-ресурсами.

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

3. Добавьте GPG-ключ для Docker-репозитория:

```
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

4. Добавьте Docker-репозиторий:

```
$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "jammy") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Обновите все программные пакеты, в том числе пакеты для добавленного Docker-репозитория:

```
$ sudo apt-get update
```

6. Установите Docker Engine:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

7. Запустите Docker:

```
$ sudo systemctl start docker
```

8. Включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker
```

Alt Linux SP 8

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo apt-get update
```

2. Установите Docker Engine:

```
$ sudo apt-get install docker-engine
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

3. Запустите и включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable --now docker
```

Astra Linux SE 1.7 Орел

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo apt-get update
```

2. Установите пакеты, необходимые для Docker:

```
$ sudo apt-get install ca-certificates curl net-tools
```

где:

- `ca-certificates` — пакет корневых сертификатов;
- `curl` — пакет для работы с веб-ресурсами;
- `net-tools` — пакет для управления сетевой подсистемой ядра Linux, в частности для запуска команды `netstat`.

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

3. Добавьте GPG-ключ для Docker-репозитория:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

4. Добавьте Docker-репозиторий:

```
$ echo "deb [arch=amd64] https://download.docker.com/linux/debian stretch stable" | sudo tee -a /etc/apt/sources.list
```

5. Обновите все программные пакеты, в том числе пакеты для добавленного Docker-репозитория:

```
$ sudo apt-get update
```

6. Установите Docker Engine:

```
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

7. Запустите Docker:

```
$ sudo systemctl start docker
```

8. Включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker
```

РЕД ОС 7.3

1. Установите Docker Engine:

```
$ sudo dnf install docker-ce docker-ce-cli
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

2. Запустите и включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker --now
```

Отключение SELinux

> ПРИМЕЧАНИЕ
>
> Отключение системы безопасности SELinux необходимо только в случае использования операционной системы CentOS 7 или RHEL 7.

Для долговременного отключения SELinux следует прописать `SELINUX=disabled` в конфигурационном файле */etc/selinux/config* и перезагрузить систему.

Файл может быть изменен с помощью редактора `vi` (или `vim`):

```
$ sudo vi /etc/selinux/config
```

Содержимое обновленного файла:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

Кроме этого, доступно временное отключение SELinux с помощью следующей команды:

```
$ sudo setenforce 0
```

Тем не менее рекомендуется отключить SELinux на постоянной основе, чтобы предотвратить ее перезапуск при каждой перезагрузке системы.

> ВНИМАНИЕ
>
> Не забудьте перезагрузить хост после отключения SELinux.

Остановка firewalld

Если на хосте используется сервис firewalld, требуется его остановка.

Перед остановкой firewalld убедитесь, что он установлен, запущен и включен, выполнив команду:

```
$ sudo systemctl status firewalld
```

Пример результата выполнения команды:

```
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-12-02 10:02:57 UTC; 3min 25s ago
     Docs: man:firewalld(1)
 Main PID: 3023 (firewalld)
    Tasks: 2
   Memory: 22.8M
   CGroup: /system.slice/firewalld.service
           └─3023 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid
```

Остановите firewalld с помощью команды:

```
$ sudo systemctl stop firewalld
```

Альтернативный вариант — отключить применение правил к сетевым пакетам при помощи команды `disable`:

```
$ sudo systemctl disable firewalld
```

Настройка доступа к Docker CLI без прав root

Для настройки доступа к Docker CLI без использования прав `root` необходимо создать группу пользователей `docker` (если она еще не существует) и добавить текущего пользователя в эту группу с определенными правами:

```
$ sudo groupadd docker
$ sudo usermod -a -G docker $USER
```

После запуска команд требуется перезайти в систему под текущим пользователем.

Шаг 2. Установка ADCM

> ПРИМЕЧАНИЕ
>
> - Рекомендуется самостоятельно настроить PostgreSQL в качестве внешней базы данных или развернуть встроенную базу данных с помощью ADCM Installer. Дополнительную информацию о настройке PostgreSQL можно получить в статье Работа с внешней базой данных.
> - Начиная с версии ADCM 2.5.0 минимальная требуемая версия PostgreSQL — 14.
> - Для просмотра полного списка доступных образов можно обратиться к репозиторию ADCM на Docker Hub.
> - Все данные ADCM сохраняются в папке */adcm/data* внутри Docker-контейнера. Во избежание потерь данных рекомендуется примонтировать к контейнеру том (Docker Volume) с последующей настройкой его бэкапов.

Для установки ADCM выполните следующие действия:

1. Скачайте Docker-образ из Arenadata Docker Registry:

```
$ sudo docker pull hub.arenadata.io/adcm/adcm:<version>
```

где `<version>` — это версия Docker-образа в одном из следующих форматов:

- `<major>.<minor>.<patch>` — если необходим конкретный патч ADCM. Пример: `2.0.0`.
- `<major>.<minor>` — если необходим последний патч в рамках выбранной версии ADCM. Пример: `2.0`.

> ВАЖНО
>
> - `latest` использовался для ранних версий ADCM и начиная с версии `2.0.0` больше не поддерживается.
> - До выпуска ADCM 2.0.0 для версионирования использовался следующий формат: `YYYY.MM.DD.HH`.

2. Создайте Docker-контейнер на основе загруженного образа:

> ПРИМЕЧАНИЕ
>
> Приведенная ниже команда создает контейнер с ADCM на порте `8000` и устанавливает подключение к внешней базе данных. Все данные контейнера будут храниться в директории */opt/adcm/*. Описание переменных окружения приведено в таблице ниже.

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

Если требуется запустить ADCM **с включенным SELinux**, при создании Docker-контейнера необходимо указать опцию `:Z` в параметре, содержащем имя каталога данных:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data:Z hub.arenadata.io/adcm/adcm:<version>
```

Опционально при создании контейнера можно указать переменные окружения, перечисленные ниже. Каждая переменная указывается в формате `-e <name>="<value>"`. Пример:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e LOG_LEVEL="INFO" hub.arenadata.io/adcm/adcm:<version>
```

> ПРИМЕЧАНИЕ
>
> Специальные переменные окружения, такие как `STATUS_LOG_LEVEL`, `ADCM_LOG_LEVEL`, `AUDIT_LOG_LEVEL`, `LDAP_LOG_LEVEL`, `BACKGROUND_TASKS_LOG_LEVEL` и `TASK_RUNNER_LOG_LEVEL`, являются более приоритетными, чем переменная окружения `LOG_LEVEL`.

Переменные окружения

Имя	Описание
DB_HOST	Имя или IP-адрес хоста, на котором развернута база данных
DB_PORT	Номер TCP-порта для подключения к базе данных
DB_USER	Имя пользователя для подключения к базе данных
DB_NAME	Имя базы данных
DB_PASS	Пароль пользователя для подключения к базе данных

DB_OPTIONS	Параметры для подключения к базе данных:
	- `sslmode` — флаг, используемый для установления SSL-соединения. Возможные значения:
	  - `disable` — установить только соединение без использования SSL.
	  - `allow` — попытаться установить соединение без использования SSL. Если не удастся — попытаться установить SSL-соединение.
	  - `prefer` (по умолчанию) — попытаться установить SSL-соединение. Если не удастся — попытаться установить соединение без использования SSL.
	  - `require` — попытаться установить только SSL-соединение. Если присутствует файл корневого сертификата удостоверяющего центра (Certificate Authority, CA) — проверить сертификат так же, как если бы было указано `verify-ca`.
	  - `verify-ca` — попытаться установить только SSL-соединение и проверить, что сертификат сервера выдан доверенным удостоверяющим центром.
	  - `verify-full` — попытаться установить только SSL-соединение и проверить, что сертификат сервера выдан доверенным удостоверяющим центром и запрашиваемое имя хоста сервера совпадает с указанным в сертификате.
	- `sslcert` — расположение файла SSL-сертификата клиента. Этот параметр игнорируется, если соединение устанавливается без SSL.
	- `sslkey` — расположение секретного ключа, используемого для клиентского сертификата. Этот параметр игнорируется, если соединение устанавливается без SSL.
	- `sslrootcert` — расположение файла, содержащего сертификат(ы) центра сертификации SSL. Если файл существует, сертификат сервера будет проверен одним из этих центров сертификации. Вы также можете указать специальное значение системы — в этом случае будут загружены доверенные корневые сертификаты системы. Точные местоположения этих корневых сертификатов зависят от реализации SSL и платформы.

DEFAULT_ADCM_URL	URL хоста, на котором разворачивается контейнер ADCM. Указывается в формате: `http://<IP>:8000/`. Если переменная не указана при создании контейнера, URL необходимо будет заполнить вручную в поле ***ADCM's URL*** на странице ***Settings***
LOG_LEVEL	Уровень логирования для всех типов логов. Возможные значения: - `DEBUG` - `INFO` - `WARNING` - `ERROR` (значение по умолчанию) - `CRITICAL`
STATUS_LOG_LEVEL	Уровень логирования для *status.log*. Возможные значения: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
ADCM_LOG_LEVEL	Уровень логирования для *adcm.log* и *adcm_debug.log*. Возможные значения: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
AUDIT_LOG_LEVEL	Уровень логирования для *audit.log*. Возможные значения: - `DEBUG` - `INFO` (значение по умолчанию) - `WARNING` - `ERROR` - `CRITICAL`
LDAP_LOG_LEVEL	Уровень логирования для *ldap.log*. Возможные значения: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
BACKGROUND_TASKS_LOG_LEVEL	Уровень логирования для *cron_task.log*. Возможные значения: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
TASK_RUNNER_LOG_LEVEL	Уровень логирования для *task_runner.err*. Возможные значения: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
MIGRATION_MODE	Флаг, указывающий на использование режима миграции — переноса базы данных ADCM с SQLite на PostgreSQL. Дополнительную информацию можно получить в статье Миграция. Возможные значения: - `0` — режим миграции выключен; - `1` — режим миграции включен.
CONSUL_URL	URL сервера HashiCorp Consul. Указывается в формате: `http://<IP>:8500/`
CONSUL_DATACENTER	Datacenter HashiCorp Consul. Если значение переменной не указано, то будет использовано значение, указанное в HashiCorp Consul по умолчанию
CONSUL_CACERT_FILE	Путь до файла сертификата удостоверяющего центра (Certificate Authority, CA) в случае включенного шифрования в HashiCorp Consul
CONSUL_CLIENT_CERT_FILE	Путь до файла сертификата клиента в случае включенного шифрования в HashiCorp Consul
CONSUL_CLIENT_KEY_FILE	Путь до файла с секретным ключом клиента в случае включенного шифрования в HashiCorp Consul

Шаг 3. Запуск ADCM

Поскольку ADCM представляет собой контейнер под управлением Docker, за его запуск и установку отвечают стандартные Docker-команды:

- **Запуск ADCM**.

```
$ sudo docker start adcm
```

- **Остановка ADCM**.

```
$ sudo docker stop adcm
```

- **Автозапуск ADCM**. Чтобы настроить автоматический запуск Docker-контейнера в случае непредвиденных ошибок, используйте следующую команду.

```
$ sudo docker update --restart=on-failure adcm
```

> ПРИМЕЧАНИЕ
>
> - Дополнительная информация по политикам перезапуска Docker-контейнеров (restart policies) может быть получена из документации Docker.
> - Если для текущего пользователя настроен доступ к Docker CLI без использования прав `root`, слово `sudo` в предыдущих командах необязательно.

Шаг 4. Проверка установки

Проверка статуса Docker-контейнера

Чтобы убедиться в успешной установке Docker-контейнера `adcm`, выполните следующую команду:

```
$ sudo docker container ls
```

Пример результата выполнения команды приведен ниже. Убедитесь, что в столбце `STATUS` выводится значение `Up`:

```
CONTAINER ID        IMAGE                             COMMAND             CREATED             STATUS              PORTS
NAMES
74b5628146e5        hub.arenadata.io/adcm/adcm:2.0   "/etc/startup.sh"   4 days ago          Up 21 minutes       0.0.0.0:8000-
>8000/tcp   adcm
```

Проверка доступности порта 8000

После установки и запуска ADCM его web-интерфейс должен стать доступным на `8000` порте развернутого контейнера.

Для операционной системы Ubuntu 22.04 предварительно установите пакет для запуска команды `netstat`:

```
$ sudo apt-get install net-tools
```

Для проверки доступности порта выполните команды:

- `netstat` — для вывода информации о состоянии порта.

```
$ sudo netstat -ntpl | grep 8000
```

Пример результата выполнения команды:

```
tcp6       0      0 :::8000                 :::*                    LISTEN
1514/docker-proxy-c
```

- `curl` — для проверки URL-соединения.

```
$ curl http://localhost:8000
```

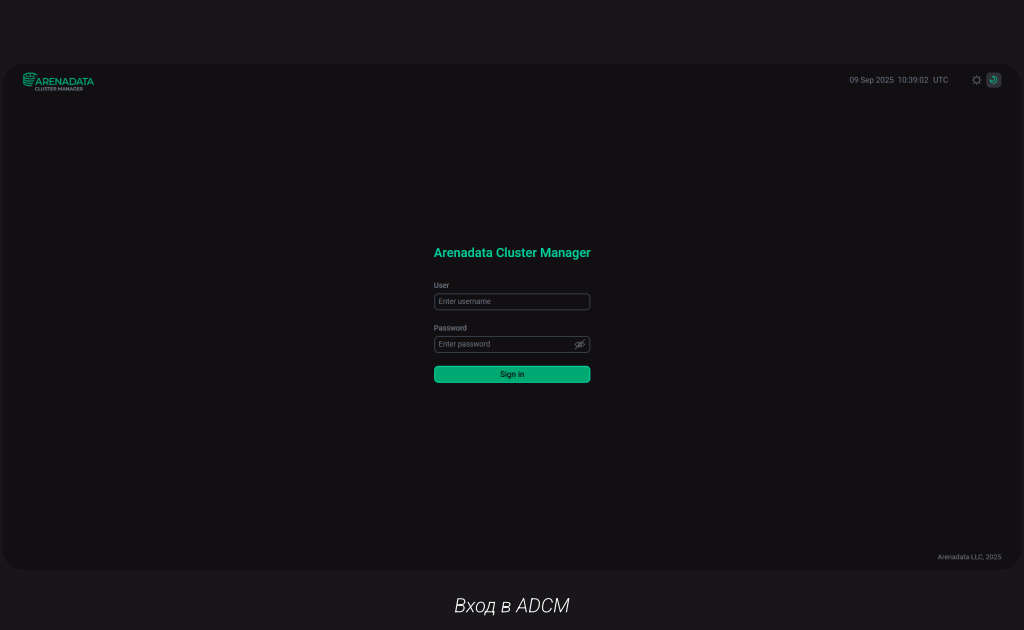
Пример результата выполнения команды:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Arenadata Cluster Manager</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/favicon.ico">
<link rel="stylesheet" href="styles.10db6328264e0907c52f.css"></head>
<body>
  <app-root></app-root>
<script src="runtime-es2015.7eb406ed18bf0258cd35.js" type="module"></script><script src="runtime-es5.7eb406ed18bf0258cd35.js" nomodule defer></script><script src="polyfills-es5.2e224d70daec4412d3c2.js" nomodule defer></script><script src="polyfills-es2015.37e0553ac06970d6a5b5.js" type="module"></script><script src="main-es2015.39851da0ebf9ed6fec45.js" type="module"></script><script src="main-es5.39851da0ebf9ed6fec45.js" nomodule defer></script></body>
</html>
```

Проверка web-интерфейса ADCM

При установке ADCM автоматически создается пользователь с административными правами. Чтобы зайти в ADCM, используйте следующие учетные данные:

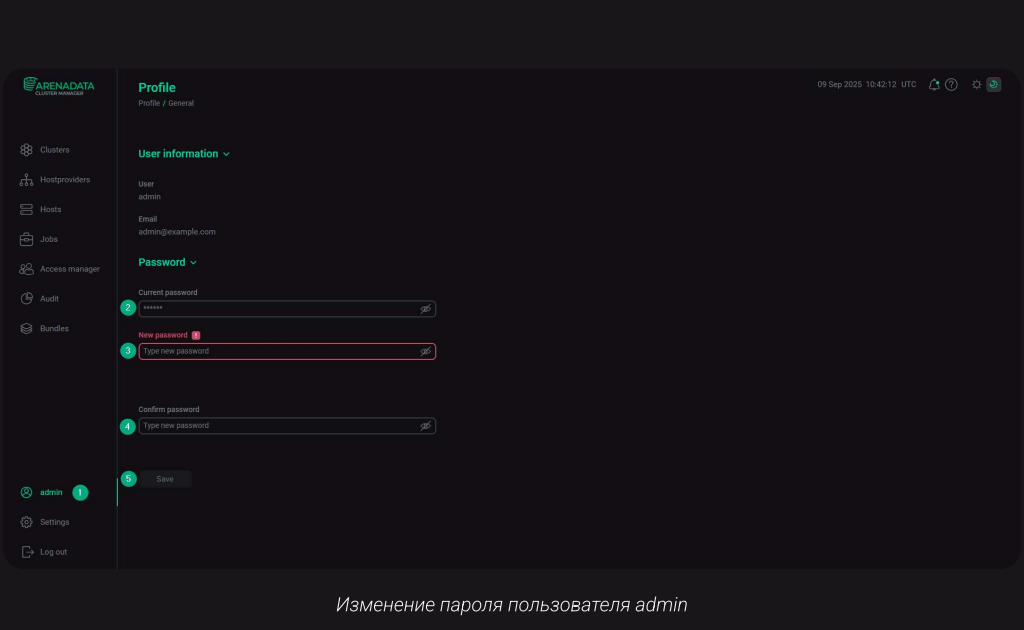
- ***User***: admin
- ***Password***: admin



Вход в ADCM

Для изменения пароля требуется выполнить следующие действия:

1. Откройте профиль текущего пользователя в левом навигационном меню.
2. Введите текущий пароль в поле ***Current password***.
3. Укажите новый пароль в поле ***New password***.
4. Подтвердите новый пароль в поле ***Confirm password***.
5. Нажмите ***Save***.

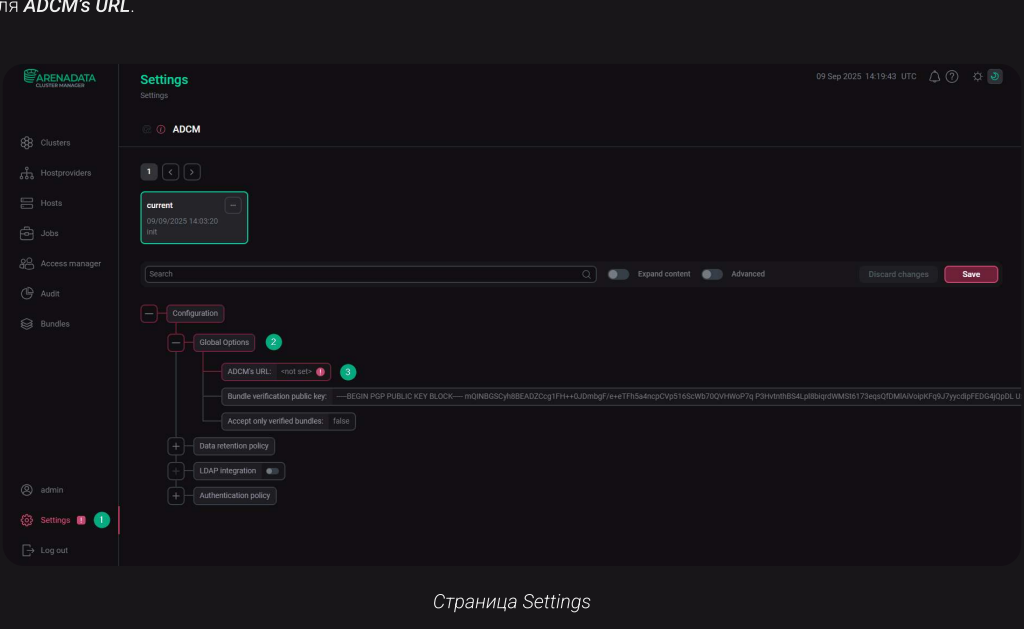


Изменение пароля пользователя admin

Установка URL ADCM

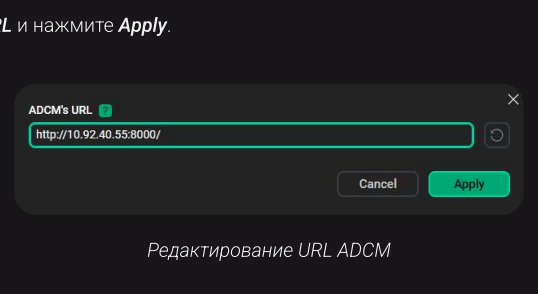
Для успешного обмена данными между ADCM и развернутыми с его помощью кластерами важно корректное определение внешнего адреса ADCM. В дальнейшем этот адрес будет использоваться для отправки в ADCM статусов кластерных компонентов. Для указания URL ADCM используется переменная окружения `DEFAULT_ADCM_URL` при создании контейнера ADCM. Если переменная не была указана, необходимо установить URL вручную в web-интерфейсе ADCM. Для этого выполните шаги:

1. Выберите пункт ***Settings*** в левом навигационном меню.
2. Раскройте узел ***Global Options*** в дереве конфигурационных настроек.
3. Нажмите на значение поля ***ADCM's URL***.



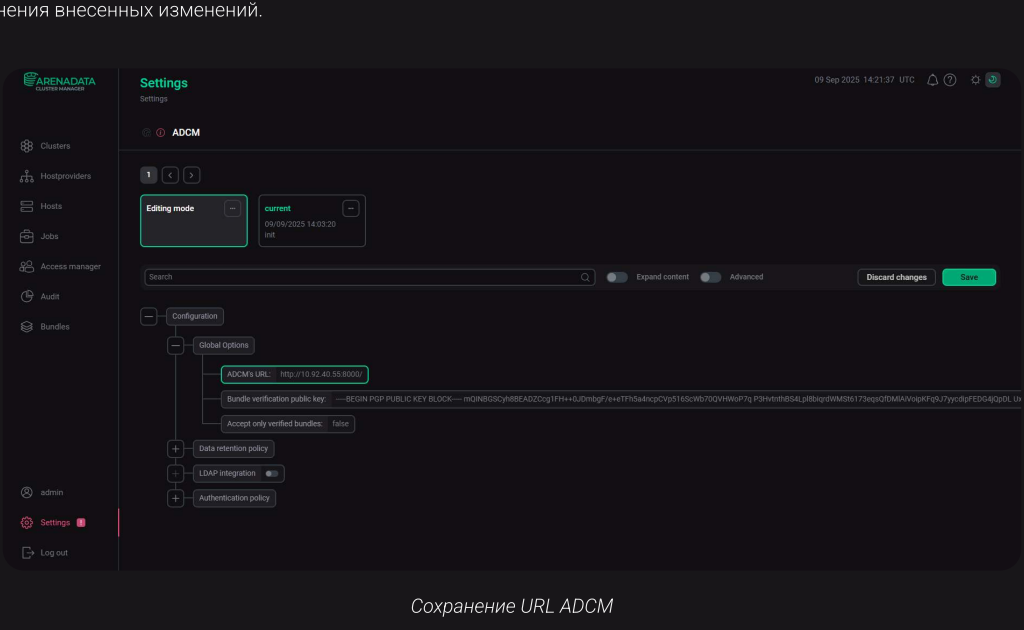
Страница Settings

4. В открывшемся окне измените значение ***ADCM's URL*** и нажмите ***Apply***.



Редактирование URL ADCM

5. Нажмите ***Save*** для сохранения внесенных изменений.



Сохранение URL ADCM

Шаг 5. Обновление ADCM

> ВНИМАНИЕ
>
> - Перед выполнением обновления рекомендуется сделать резервную копию данных ADCM. Дополнительную информацию можно получить в статье Резервное копирование и восстановление.
> - Понижение версии ADCM не поддерживается.
> - Не рекомендуется обновлять ADCM более, чем на 5 минорных версий (см. `<minor>` в описании формата версионирования ниже).
> - Перед обновлением ADCM до версии 2.7.0 выполните миграцию данных из SQLite в PostgreSQL. Миграцию можно осуществить в ADCM любой версии (до 2.6.0 включительно).

Для обновления ADCM до нужной версии выполните следующие шаги:

1. Остановите Docker-контейнер `adcm`:

```
$ sudo docker stop adcm
```

2. Удалите Docker-контейнер `adcm`:

```
$ sudo docker container rm adcm
```

3. Загрузите из Arenadata Docker Registry необходимый Docker-образ:

```
$ sudo docker pull hub.arenadata.io/adcm/adcm:<version>
```

где `<version>` — это версия Docker-образа в одном из следующих форматов:

- `<major>.<minor>.<patch>` — если необходим конкретный патч ADCM. Пример: `2.0.0`.
- `<major>.<minor>` — если необходим последний патч в рамках выбранной версии ADCM. Пример: `2.0`.

> ВАЖНО
>
> - `latest` использовался для ранних версий ADCM и начиная с версии `2.0.0` больше не поддерживается.
> - До выпуска ADCM 2.0.0 для версионирования использовался следующий формат: `YYYY.MM.DD.HH`.

4. Создайте новый Docker-контейнер на базе загруженного образа:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

> ПРИМЕЧАНИЕ
>
> Подробная информация о переменных окружения `DB_HOST`, `DB_PORT`, `DB_USER`, `DB_NAME`, `DB_PASS` и `DB_OPTIONS` приведена в статье Установка.

5. Запустите новый Docker-контейнер `adcm`:

```
$ sudo docker start adcm
```

Подготовка хостов

Назад к содержанию

Перед установкой нового кластера необходимо добавить информацию о его будущих хостах в ADCM. Это задача **хостпровайдеров** — специальных плагинов, отвечающих за взаимодействие между хостами и ADCM.

ADCM поддерживает несколько типов хостпровайдеров. Для получения подробной информации о том, как их настроить и использовать для создания хостов, обратитесь к документации:

- SSH
- Yandex Cloud
- VCloud (VMware)
- Cloud.ru Advanced
- CROC Cloud
- ADQM Cloud Bundle

ВНИМАНИЕ

Если в ADQM планируется использовать аутентификацию Kerberos, при создании хостов ADQM необходимо заполнять их полные доменные имена (FQDN).

Создание кластера

Дарья Барышева, Елена Дворядкина

Содержание

Назад к содержанию

Шаг 1. Загрузка бандла
Шаг 2. Загрузка бандла в ADCM
Шаг 3. Создание кластера на базе загруженного бандла

В ADCM **кластер** означает набор сервисов, запущенных на группе хостов.

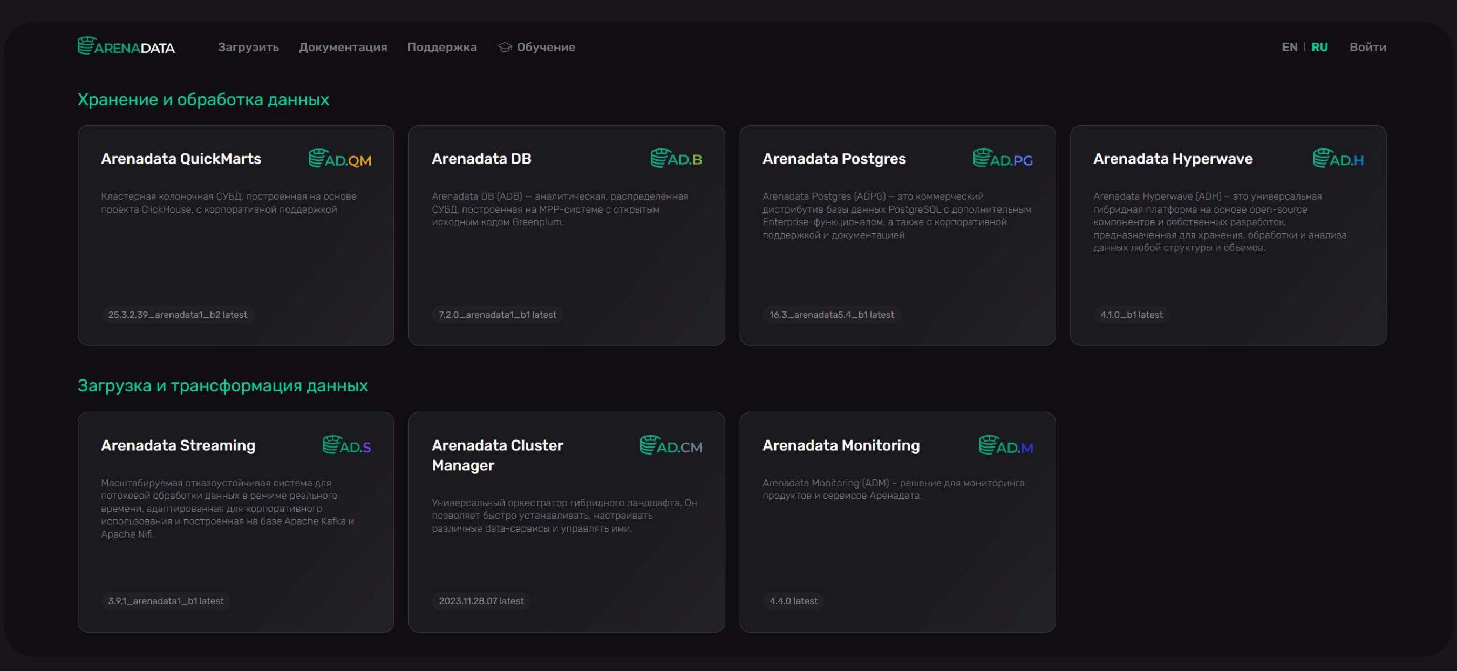
Дистрибутивы кластеров для ADCM поставляются в виде бандлов. В общем случае бандл представляет обычный архив, описывающий логику развертывания кластера.

Шаги для добавления нового кластера приведены ниже.

> **ПРИМЕЧАНИЕ**
>
> При использовании Enterprise-версии продукта следует пропустить шаг 1, получить бандл у команды поддержки Arenadata и перейти к шагу 2.

Шаг 1. Загрузка бандла

1. Перейдите на сайт https://network.arenadata.io/ и выберите продукт ***Arenadata QuickMarts***.


ARENADATA
Загрузить
Документация
Поддержка
Обучение
EN | RU
Войти
Хранение и обработка данных
Arenadata QuickMarts
25.3.2.39_arenadata1_b2 latest
Arenadata DB
7.2.0_arenadata1_b1 latest
Arenadata Postgres
16.3_arenadata5.4_b1 latest
Arenadata Hyperwave
4.1.0_b1 latest
Загрузка и трансформация данных
Arenadata Streaming
3.9.1_arenadata1_b1 latest
Arenadata Cluster Manager
2023.11.28.07 latest
Arenadata Monitoring
4.4.0 latest

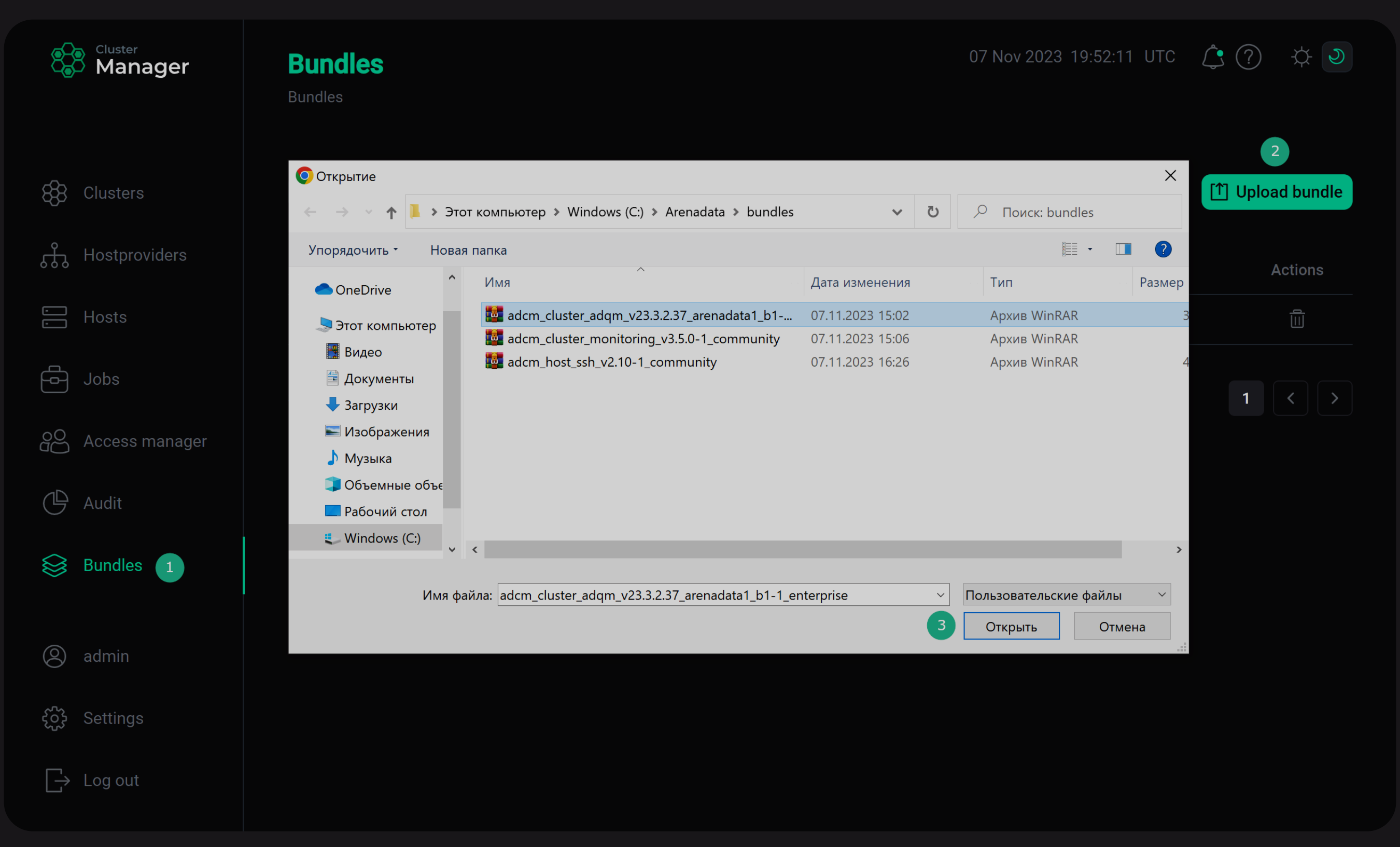
Переход к загрузке дистрибутивов

2. На следующей странице загрузите бандл для нужной версии ADQM.

Шаг 2. Загрузка бандла в ADCM

Чтобы загрузить бандл в ADCM:

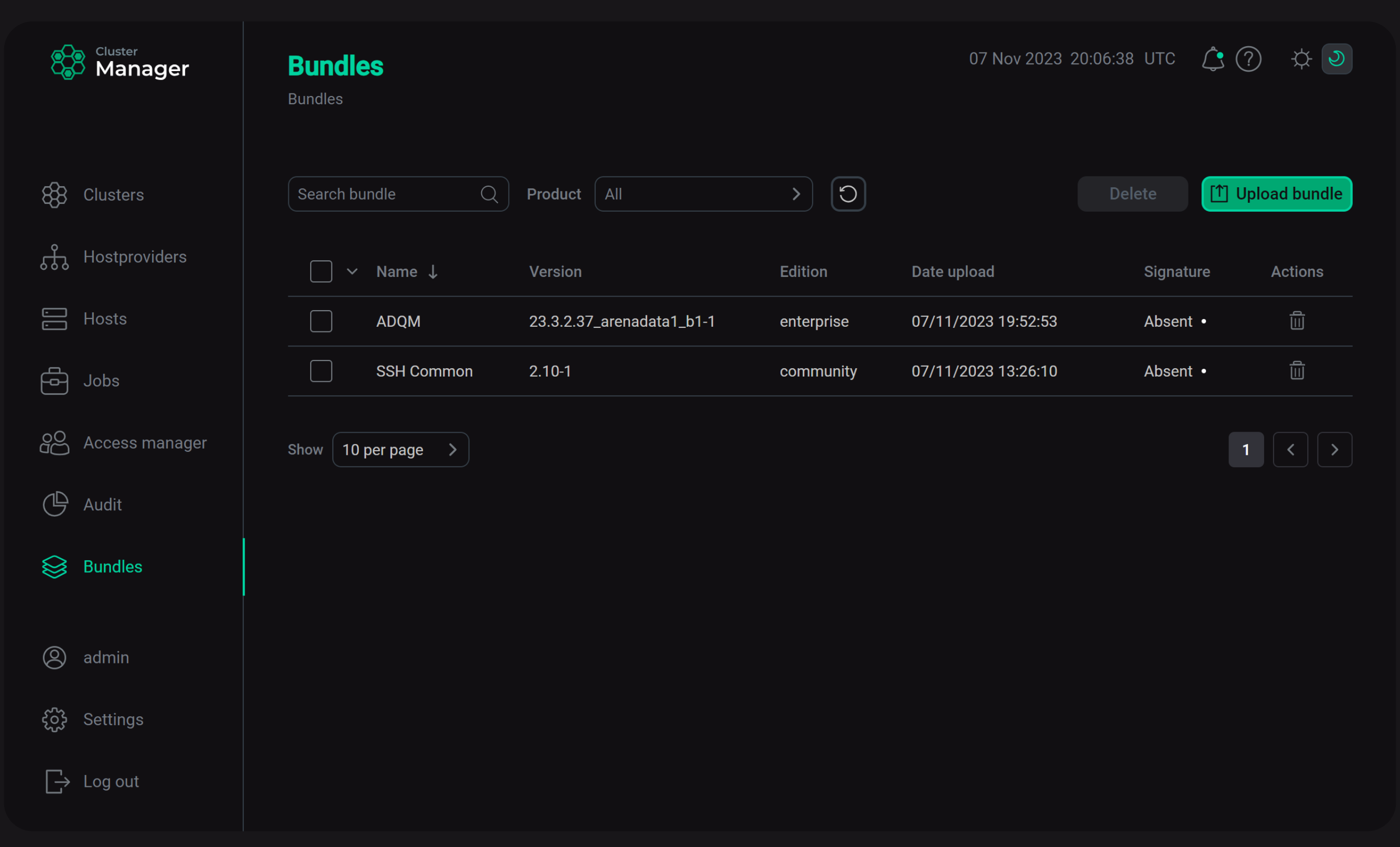
1. Выберите пункт ***Bundles*** в левом навигационном меню и нажмите ***Upload bundle***.
2. Выберите бандл в открывшемся диалоговом окне.


Cluster Manager
Bundles
07 Nov 2023 19:52:11 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Открытие
Этот компьютер > Windows (C:) > Arenadata > bundles
Поиск: bundles
Упорядочить
Новая папка
2
Upload bundle
Actions
3
Открыть
Отмена

Имя | Дата изменения | Тип
adcm_cluster_adqm_v23.3.2.37_arenadata1_b1-... | 07.11.2023 15:02 | Архив WinRAR
adcm_cluster_monitoring_v3.5.0-1_community | 07.11.2023 15:06 | Архив WinRAR
adcm_host_ssh_v2.10-1_community | 07.11.2023 16:26 | Архив WinRAR

Переход к загрузке бандла

3. После успешной загрузки бандл отображается на странице ***Bundles***.


Cluster Manager
Bundles
07 Nov 2023 20:06:38 UTC
Search bundle
Product
All
Delete
Upload bundle
Show
10 per page
1

Name	Version	Edition	Date upload	Signature	Actions
ADQM	23.3.2.37_arenadata1_b1-1	enterprise	07/11/2023 19:52:53	Absent	
SSH Common	2.10-1	community	07/11/2023 13:26:10	Absent	

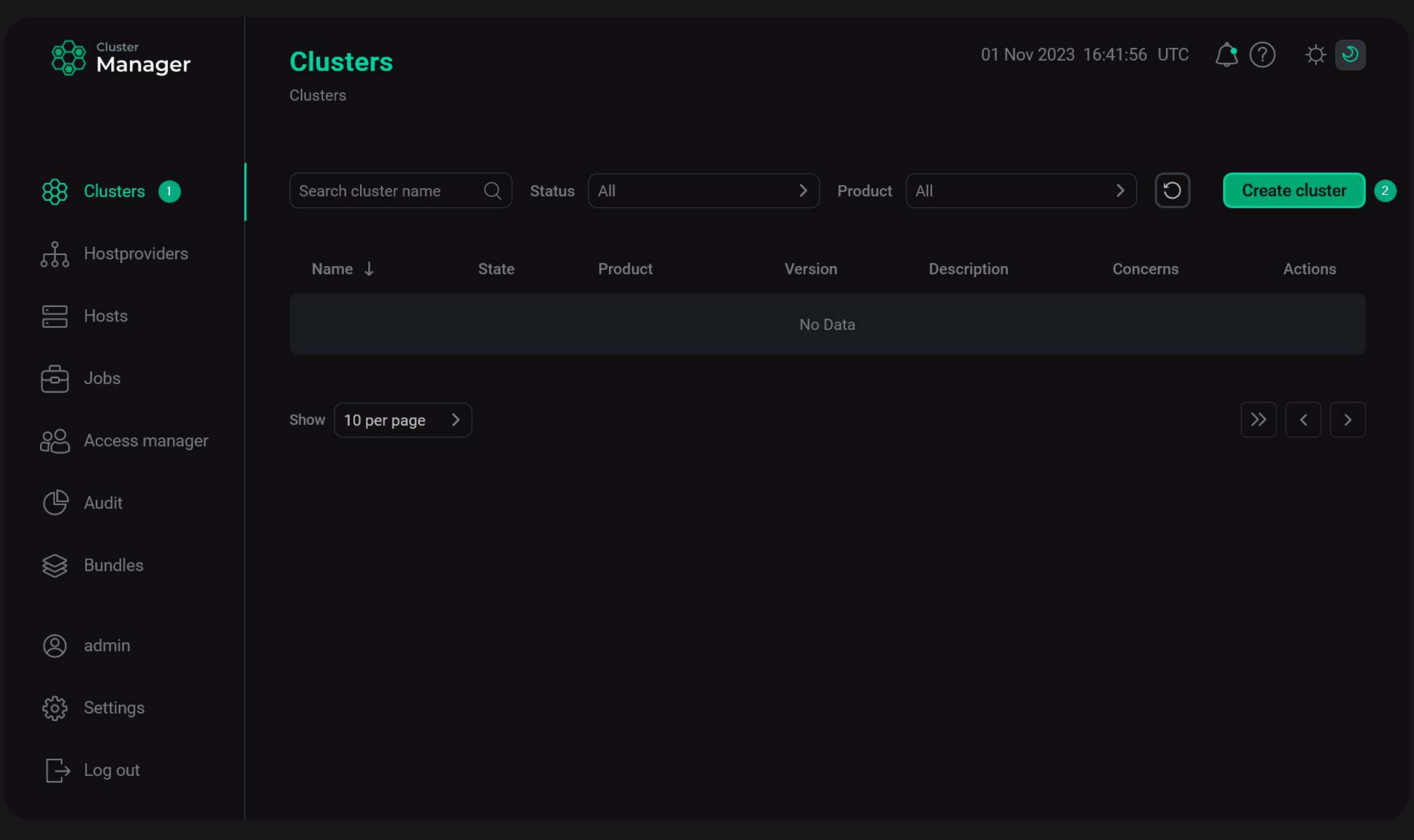
Результат успешной загрузки бандла

Шаг 3. Создание кластера на базе загруженного бандла

> **ВАЖНО**
>
> На этапе создания кластера через web-интерфейс ADCM генерируется только сам кластер. Это означает добавление записи о кластере в ADCM — сервисы на этом шаге еще не устанавливаются.

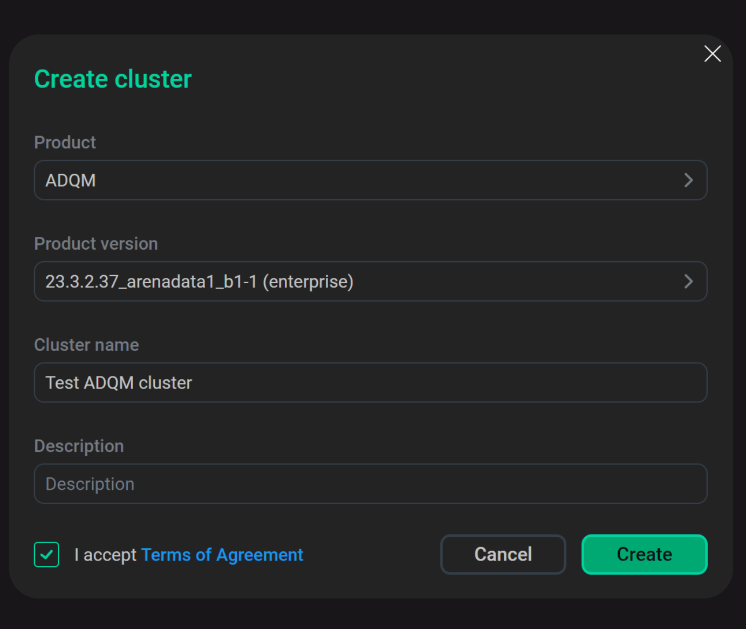
Для создания кластера необходимо выполнить шаги:

1. Выберите пункт ***Clusters*** в левом навигационном меню и нажмите ***Create cluster***.


Cluster Manager
Clusters
01 Nov 2023 16:41:56 UTC
Clusters
1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
2
Name
State
Product
Version
Description
Concerns
Actions
No Data
Show
10 per page

Переход к созданию кластера

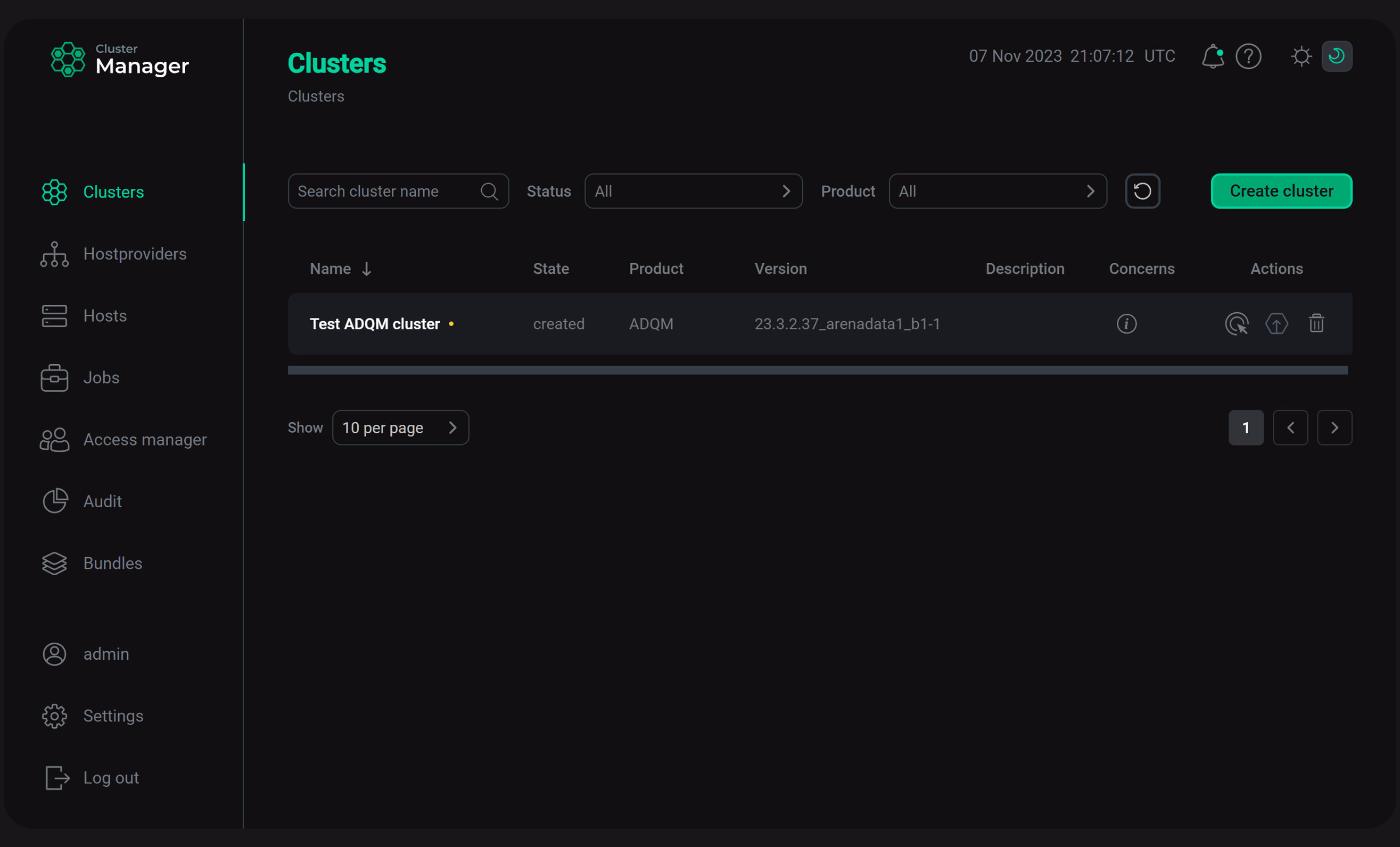
2. В открывшемся окне требуется описать новый кластер:
   - Выберите загруженный бандл в поле ***Product***.
   - Выберите версию бандла в поле ***Product version***. Несколько версий становятся доступны после загрузки нескольких бандлов для одного типа продуктов.
   - Введите имя кластера в поле ***Cluster name***.
   - При необходимости добавьте описание кластера в поле ***Description***.
   - Установите флаг ***I accept Terms of Agreement*** для подписания лицензионного соглашения. Флаг отображается только для бандлов версии Enterprise. Чтобы ознакомиться с текстом соглашения, перейдите по ссылке ***Terms of Agreement***. Подписать соглашение также возможно на странице ***Bundles***, нажав на соответствующий продуктовый бандл.
   - Нажмите ***Create***.


Create cluster
Cancel
Create

Product: ADQM
Product version: 23.3.2.37_arenadata1_b1-1 (enterprise)
Cluster name: Test ADQM cluster
Description: Description
☑ I accept Terms of Agreement

Создание кластера

3. После успешного добавления кластер отображается на странице ***Clusters***.


Cluster Manager
Clusters
07 Nov 2023 21:07:12 UTC
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

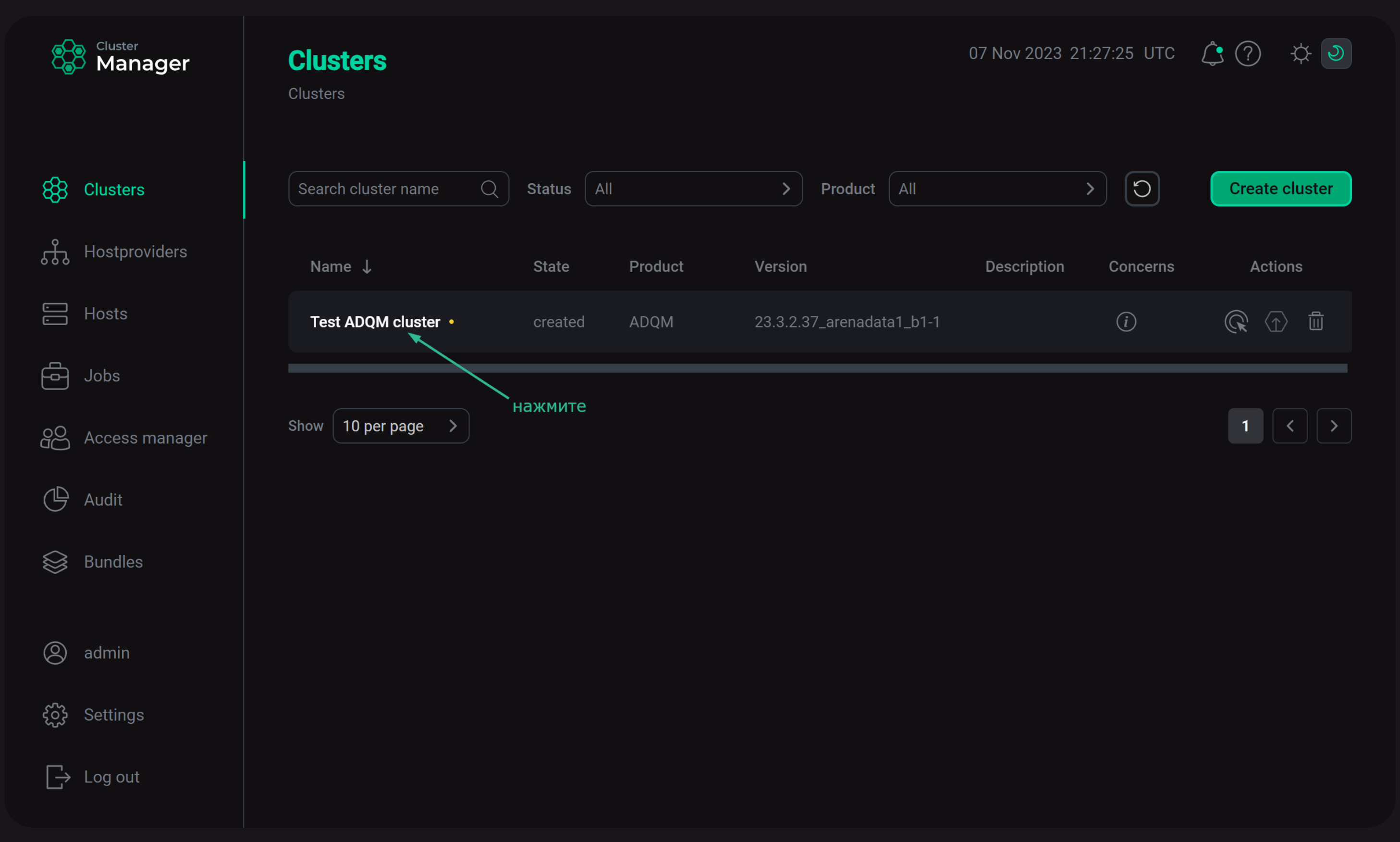
Результат успешного добавления кластера

Добавление сервисов

Дарья Барышева, Елена Дворядкина

В ADCM **сервис** означает программное обеспечение, выполняющее некоторую функцию. Примеры сервисов в кластерах ADQM: ADQMDB, Monitoring Clients, Zookeeper и так далее. Для добавления сервисов в кластер:

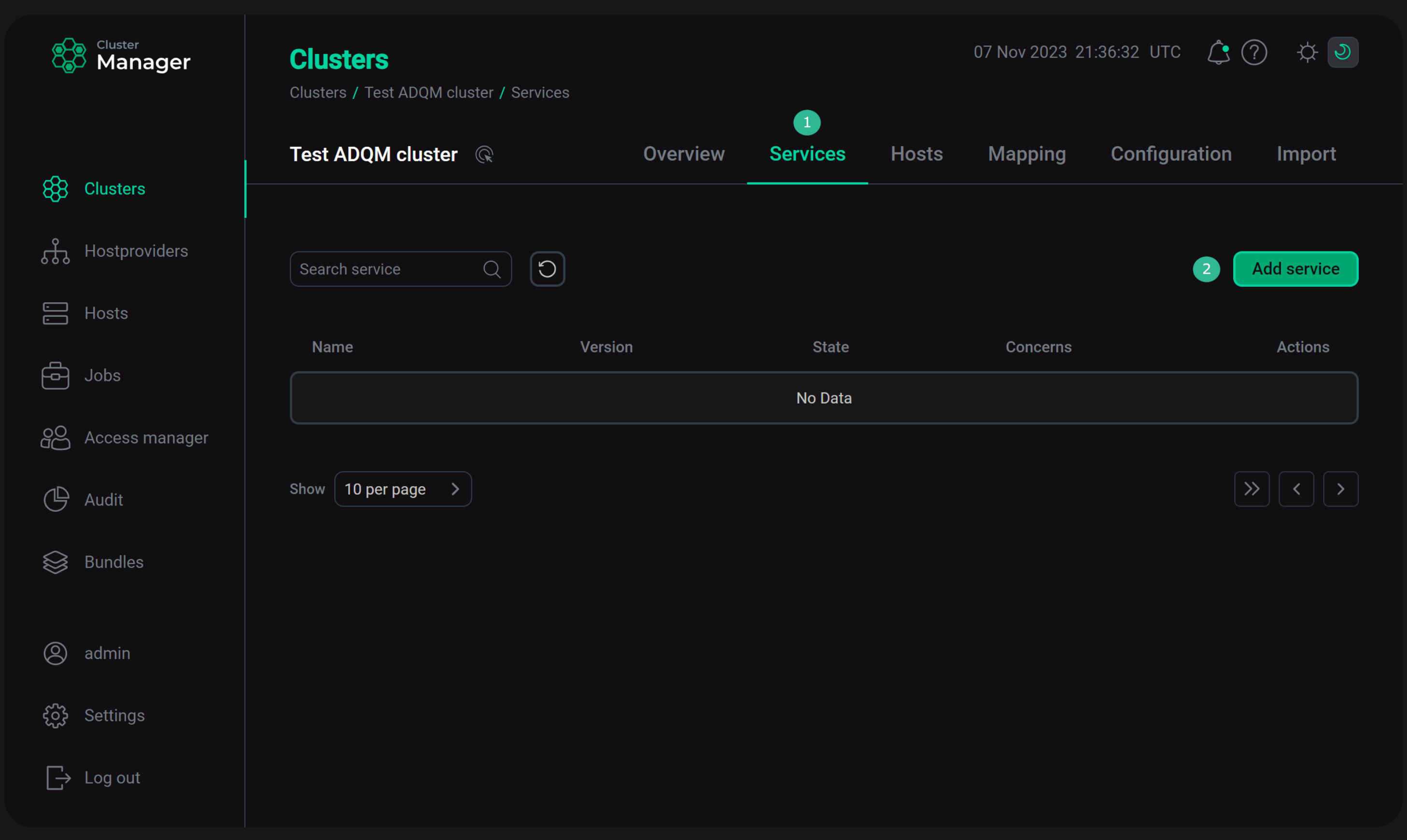
1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1
нажмите

Name ↓	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

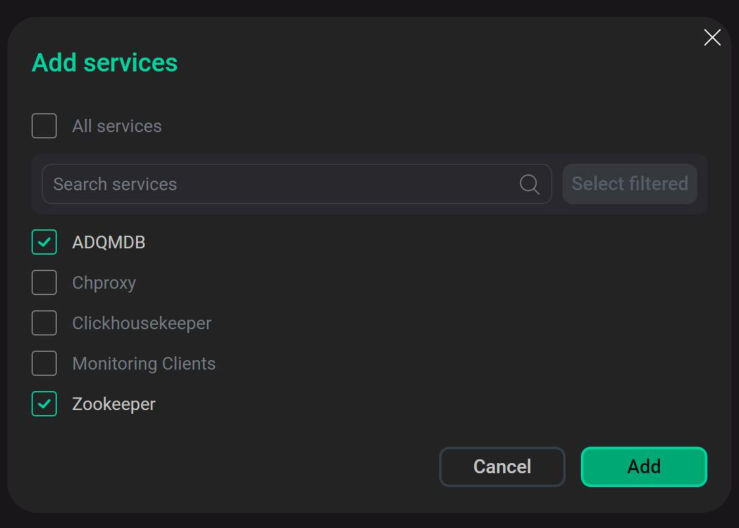
Выбор кластера

2. Откройте вкладку ***Services*** на странице кластера и нажмите ***Add service***.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
07 Nov 2023 21:36:32 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Test ADQM cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page

Переход к добавлению сервисов

3. В открывшемся окне выберите сервисы для добавления в кластер и нажмите ***Add***.


Add services
Search services
Select filtered
Cancel
Add

☐ All services
☑ ADQMDB
☐ Chproxy
☐ Clickhousekeeper
☐ Monitoring Clients
☑ Zookeeper

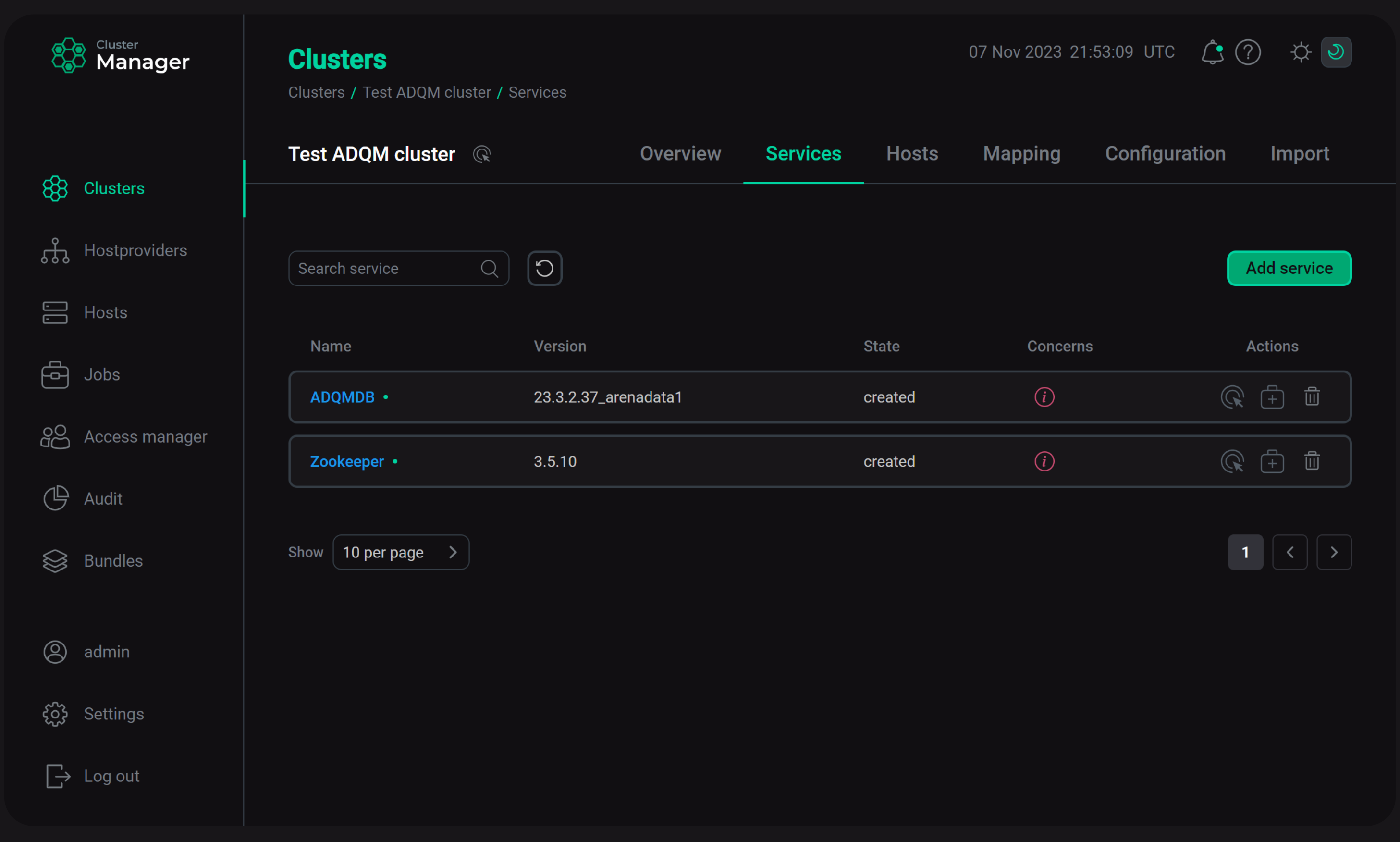
Выбор сервисов

Краткое описание доступных сервисов приведено ниже.

Сервисы, которые могут быть добавлены в кластер ADQM

Сервис	Обязательность	Назначение	Доступность
ADQMDB	Да	ClickHouse — это колоночно-ориентированная система управления базами данных (СУБД) для аналитической обработки запросов в режиме онлайн (OLAP)	Community/Enterprise
Zookeeper	Нет	Централизованный сервис для ведения информации о конфигурации, именования, обеспечения распределенной синхронизации и предоставления групповых услуг	Community/Enterprise
Clickhousekeeper	Нет	Сервис координации для репликации данных и выполнения распределенных DDL-запросов (альтернатива ZooKeeper)	Community/Enterprise
Monitoring Clients	Нет	Агенты, отправляющие информацию о хостах кластера в систему мониторинга, основанную на сервисах Diamond, Graphite и Grafana. Сервис мониторинга необходимо добавлять в кластер после создания и настройки кластера мониторинга	Community/Enterprise
Monitoring	Нет	Сервис для сбора и хранения метрик мониторинга кластера ADQM на базе системы мониторинга Prometheus и сервиса Grafana	Community/Enterprise
Chproxy	Нет	HTTP-прокси и балансировщик нагрузки для базы данных ClickHouse	Enterprise

4. В результате успешно добавленные сервисы отображаются на вкладке ***Services***.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
07 Nov 2023 21:53:09 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
Show
10 per page
1

Name	Version	State	Concerns	Actions
ADQMDB	23.3.2.37_arenadata1	created		
Zookeeper	3.5.10	created		

Результат успешного добавления сервисов в кластер

ПРИМЕЧАНИЕ

Опциональные сервисы могут быть добавлены в кластер позднее. Процесс добавления сервисов в уже развернутый кластер не отличается от описанного выше.

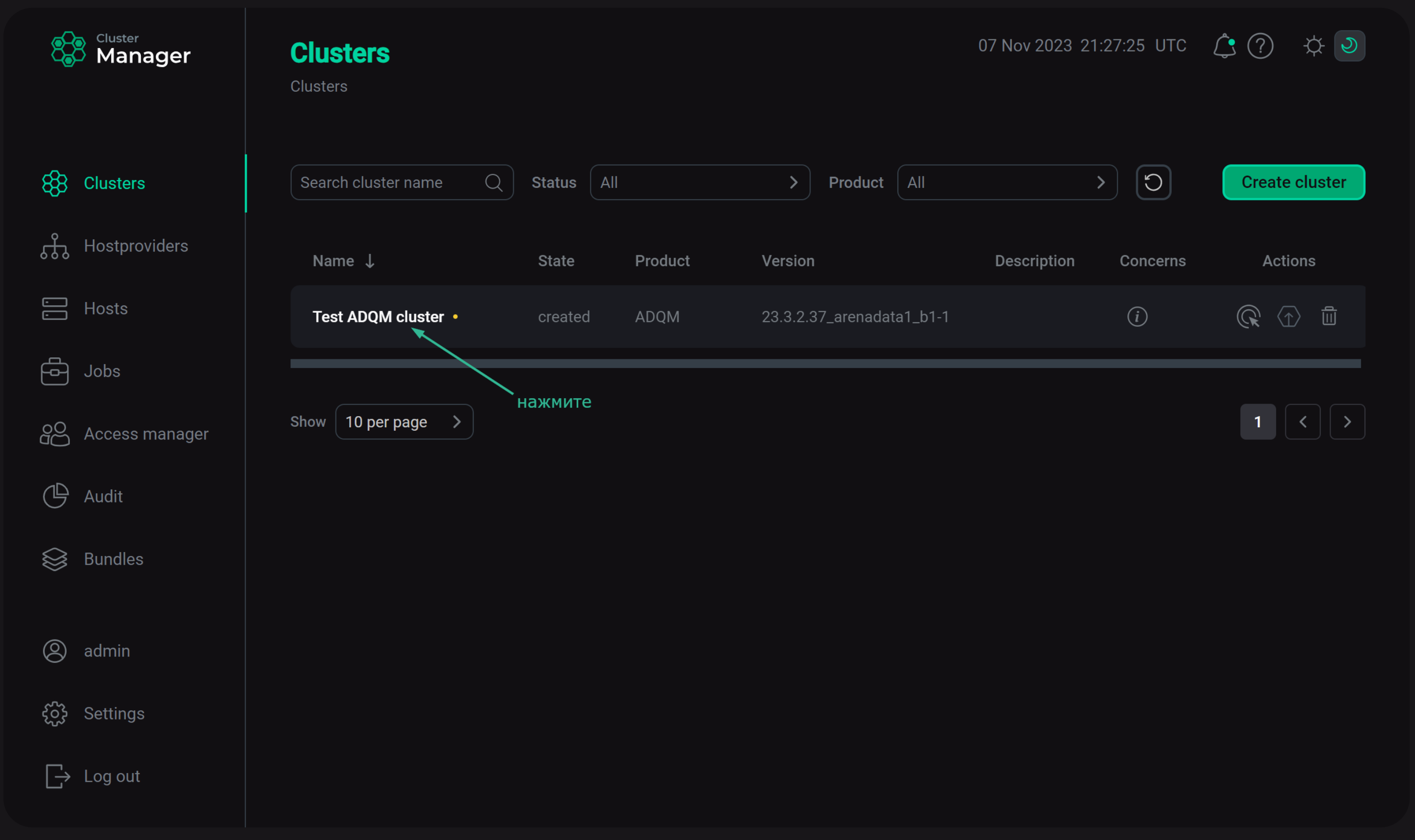
Добавление хостов в кластер

Дарья Барышева, Елена Дворядкина

Назад к содержанию

Для добавления хостов в кластер:

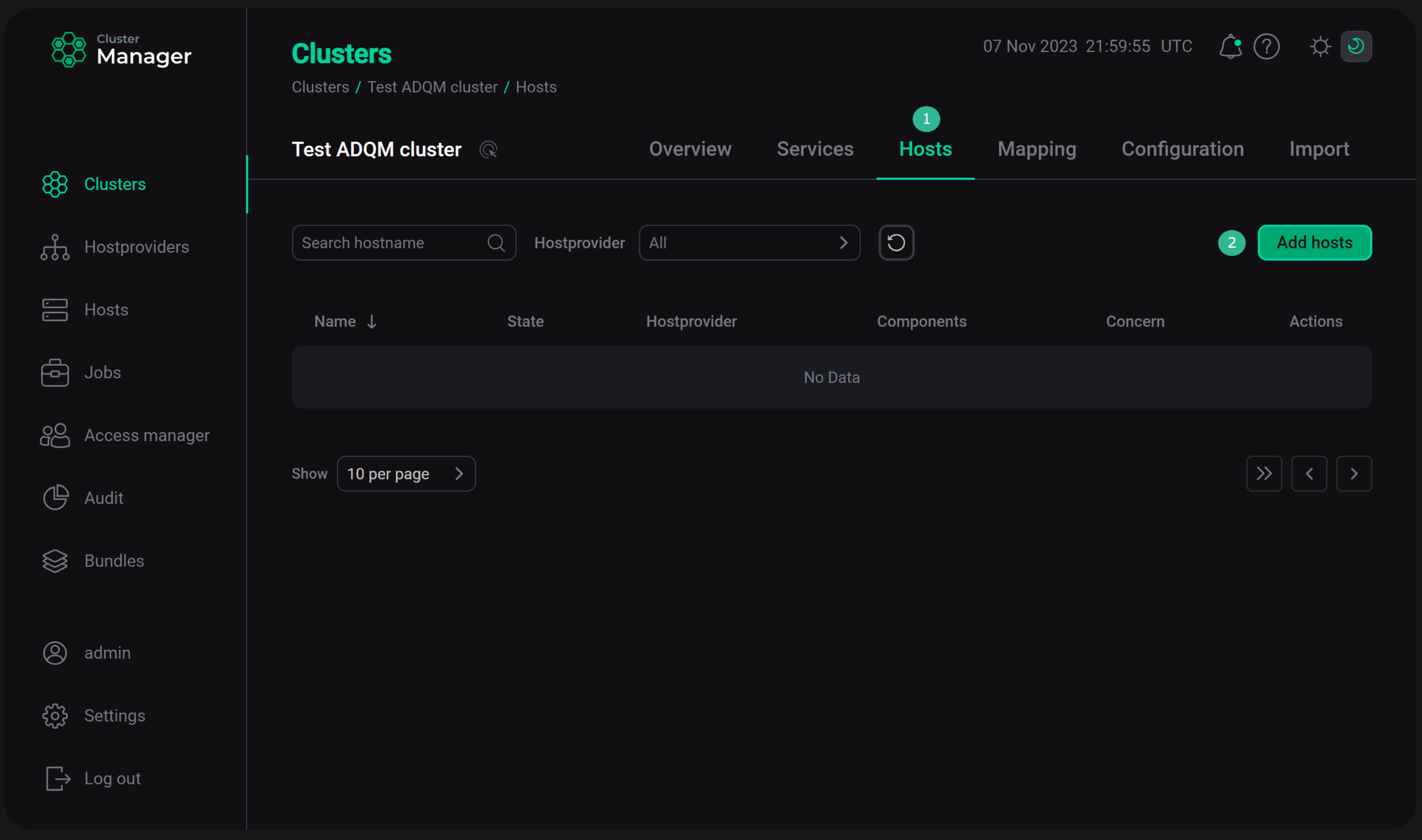
1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
нажмите
Show
10 per page
1

Name ↓	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

Выбор кластера

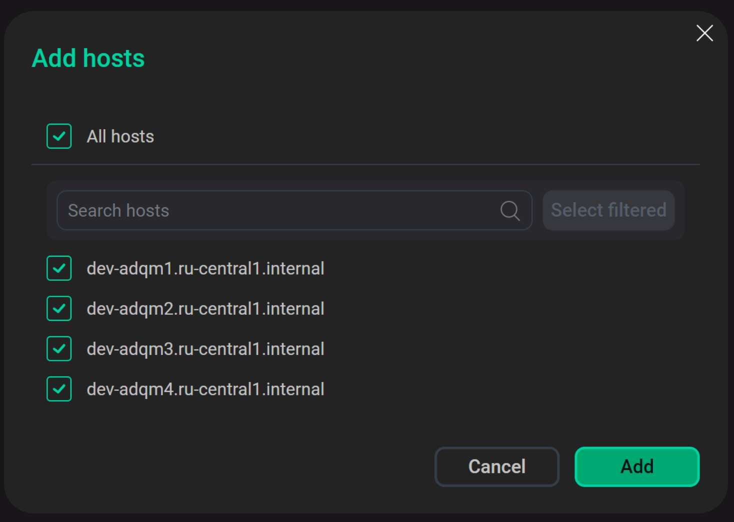
2. Откройте вкладку ***Hosts*** на странице кластера и нажмите ***Add hosts***.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Hosts
07 Nov 2023 21:59:55 UTC
Test ADQM cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
2
Add hosts
Show
10 per page

Name ↓	State	Hostprovider	Components	Concern	Actions
No Data					

Переход к добавлению хостов

3. В открывшемся окне выберите хосты для добавления в кластер и нажмите ***Add***.

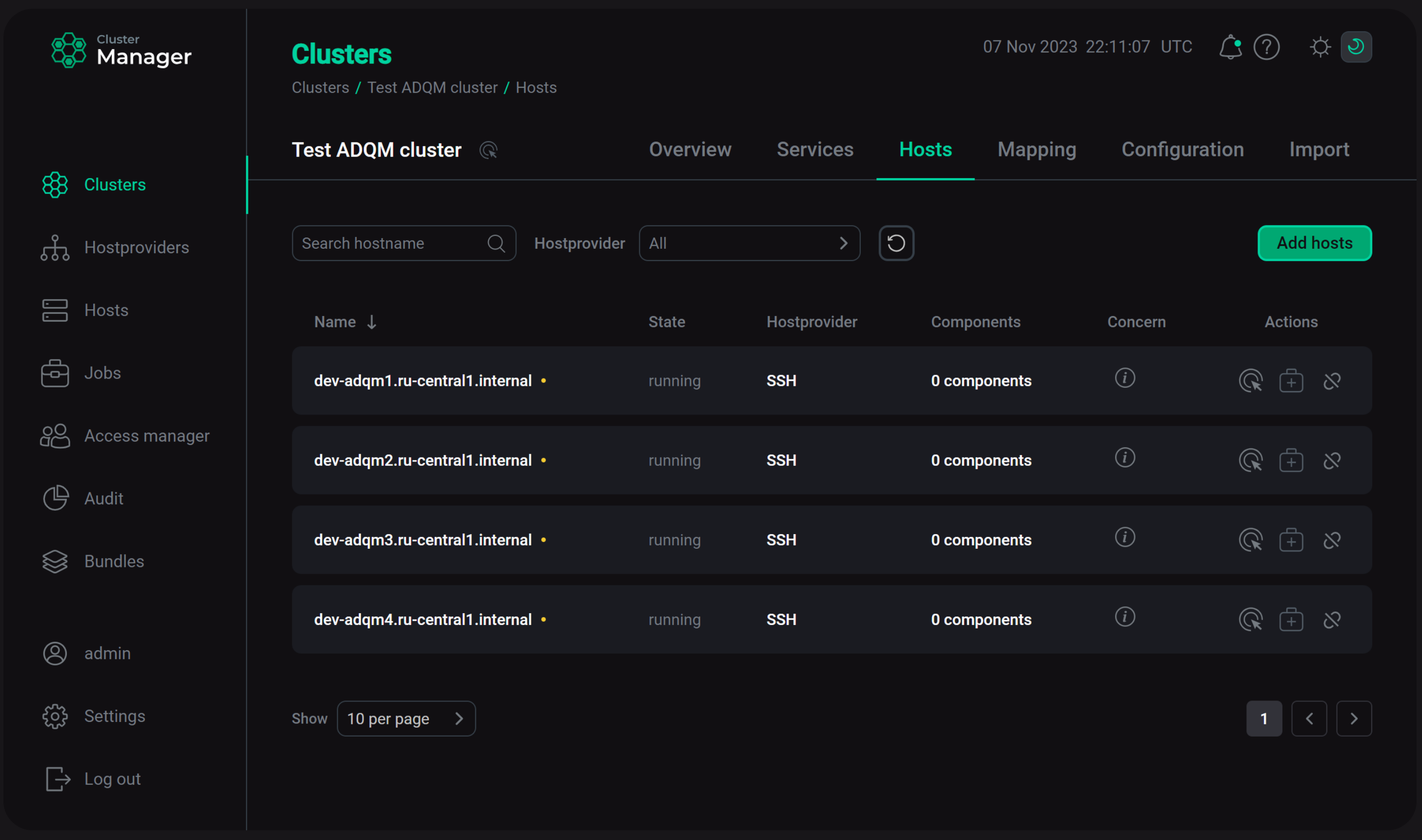

Add hosts
Search hosts
Select filtered
Cancel
Add

☑ All hosts

☑ dev-adqm1.ru-central1.internal
☑ dev-adqm2.ru-central1.internal
☑ dev-adqm3.ru-central1.internal
☑ dev-adqm4.ru-central1.internal

Выбор хостов

4. В результате успешно добавленные хосты отображаются на вкладке ***Hosts***.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Hosts
07 Nov 2023 22:11:07 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
Add hosts
Show
10 per page
1

Name ↓	State	Hostprovider	Components	Concern	Actions
dev-adqm1.ru-central1.internal	running	SSH	0 components		
dev-adqm2.ru-central1.internal	running	SSH	0 components		
dev-adqm3.ru-central1.internal	running	SSH	0 components		
dev-adqm4.ru-central1.internal	running	SSH	0 components		

Результат успешного добавления хостов в кластер

Добавление компонентов

Дарья Барышева, Елена Дворядкина

Назад к содержанию

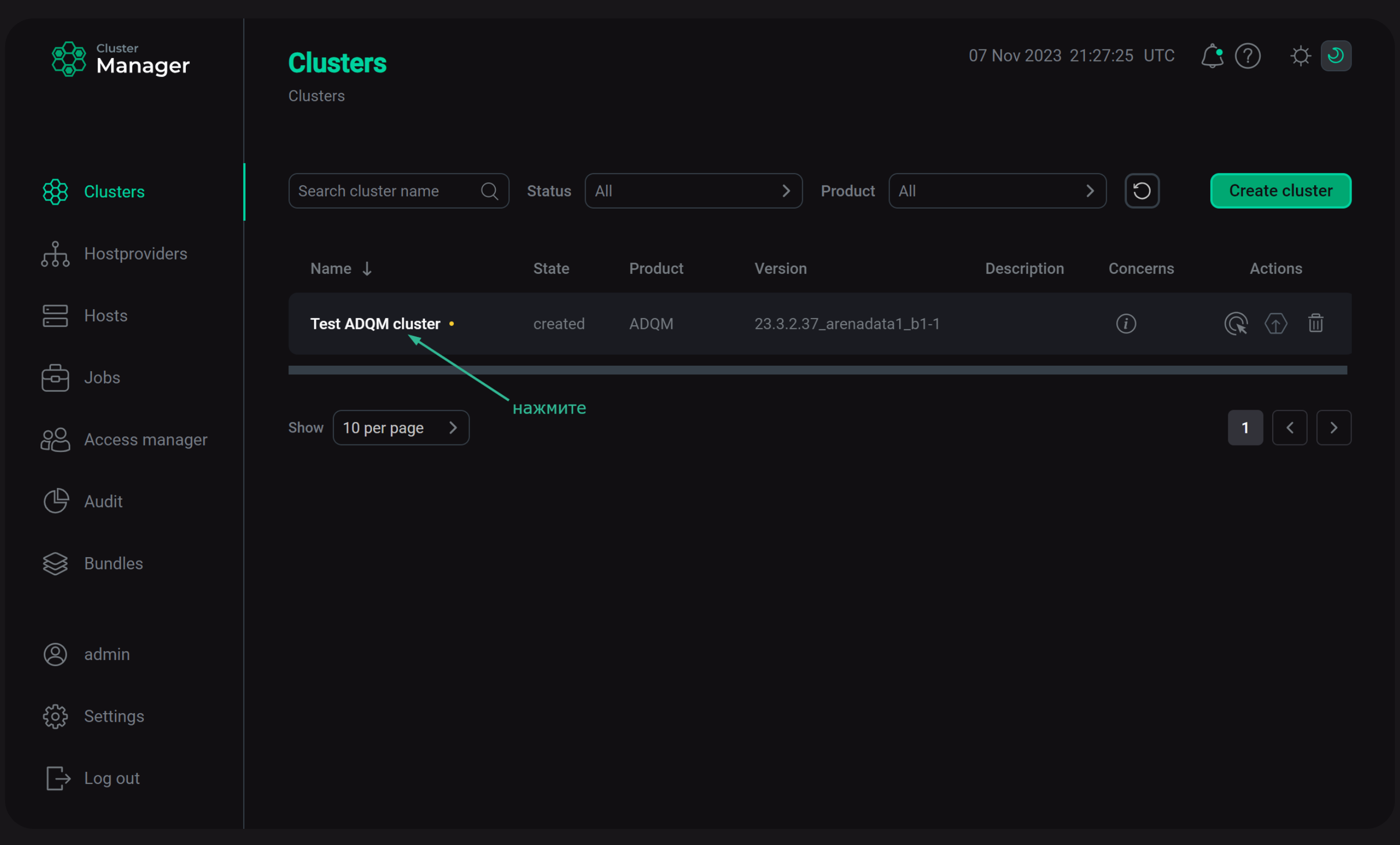
В ADCM **компонент** означает часть сервиса, которая должна быть развернута на одном или нескольких хостах кластера.

ВАЖНО

Изначально на хостах кластера нет ни одного компонента — распределение компонентов между хостами должно быть проведено вручную.

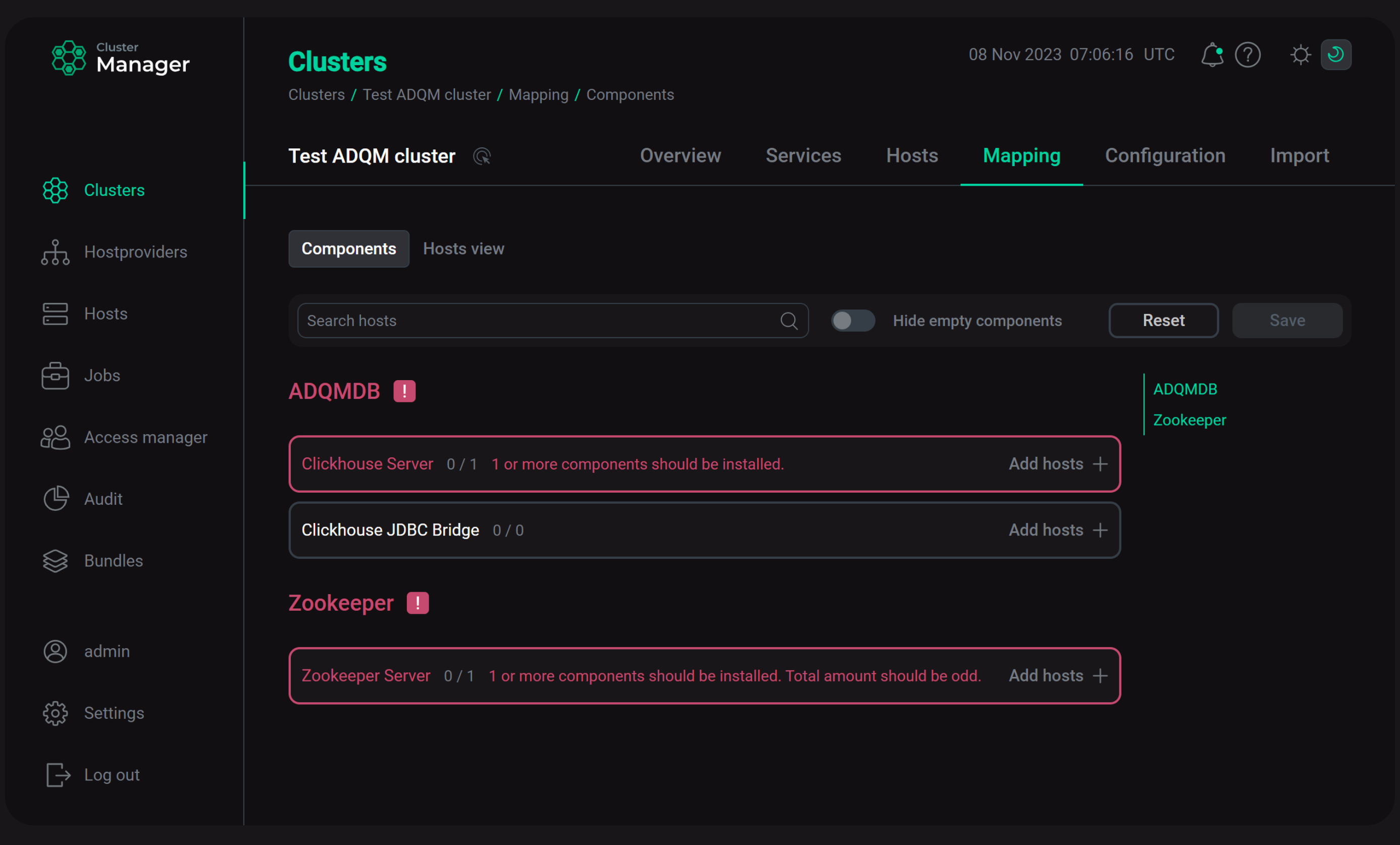
Для добавления компонентов на хосты кластера:

1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
07 Nov 2023 21:27:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test ADQM cluster
created
ADQM
23.3.2.37_arenadata1_b1-1
нажмите
Show
10 per page
1

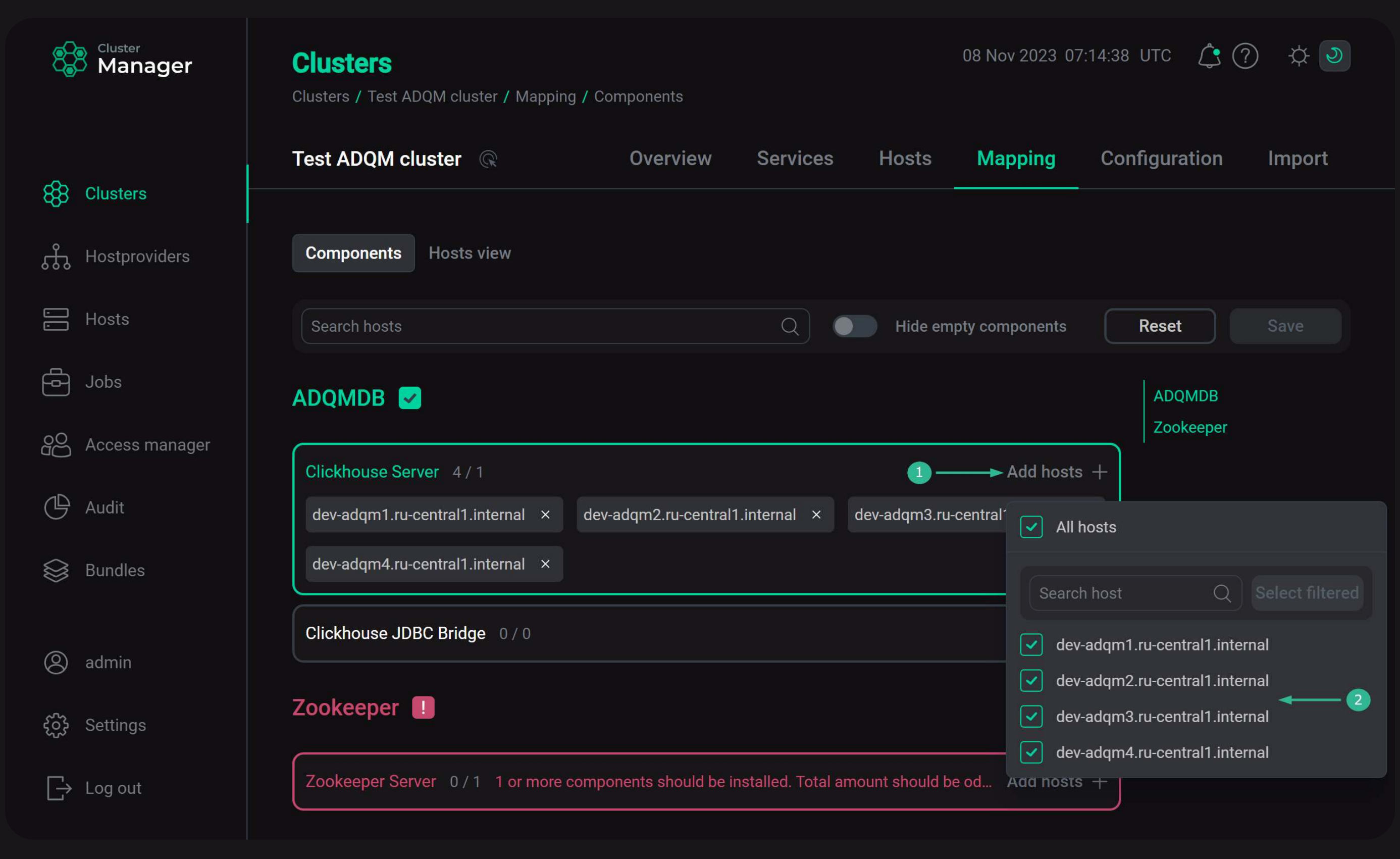
Выбор кластера

2. Откройте вкладку ***Mapping*** на странице кластера.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Mapping / Components
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
ADQMDB
Clickhouse Server 0 / 1 1 or more components should be installed.
Add hosts
Clickhouse JDBC Bridge 0 / 0
Add hosts
Zookeeper
Zookeeper Server 0 / 1 1 or more components should be installed. Total amount should be odd.
Add hosts
ADQMDB
Zookeeper

Переход к распределению сервисных компонентов

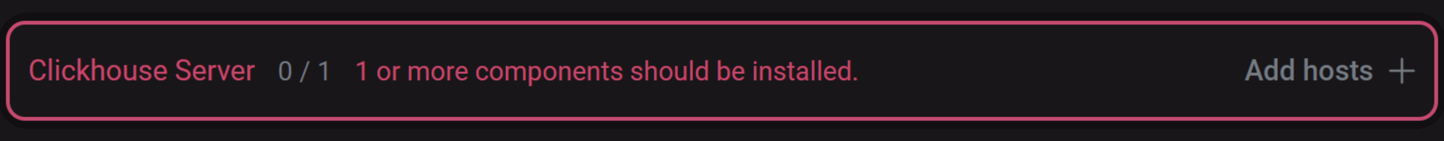
3. Распределите все компоненты между хостами кластера. Для назначения хостов компоненту нажмите ***Add hosts*** и выберите хосты в открывшемся окне.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Mapping / Components
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
ADQMDB
Clickhouse Server 4 / 1
1
Add hosts
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
Clickhouse JDBC Bridge 0 / 0
Zookeeper
Zookeeper Server 0 / 1 1 or more components should be installed. Total amount should be od...
Add hosts
All hosts
Search host
Select filtered
☑ dev-adqm1.ru-central1.internal
☑ dev-adqm2.ru-central1.internal
☑ dev-adqm3.ru-central1.internal
☑ dev-adqm4.ru-central1.internal
2

Выбор хостов для компонента

Обязательные компоненты

Каждый сервис может иметь обязательные и опциональные компоненты. Обязательные подсвечиваются красным цветом. Также в интерфейсе ADCM для них выводится информация о том, в каком количестве необходимо установить каждый компонент.


Clickhouse Server 0 / 1 1 or more components should be installed. Add hosts +

Обязательные компоненты

ВНИМАНИЕ

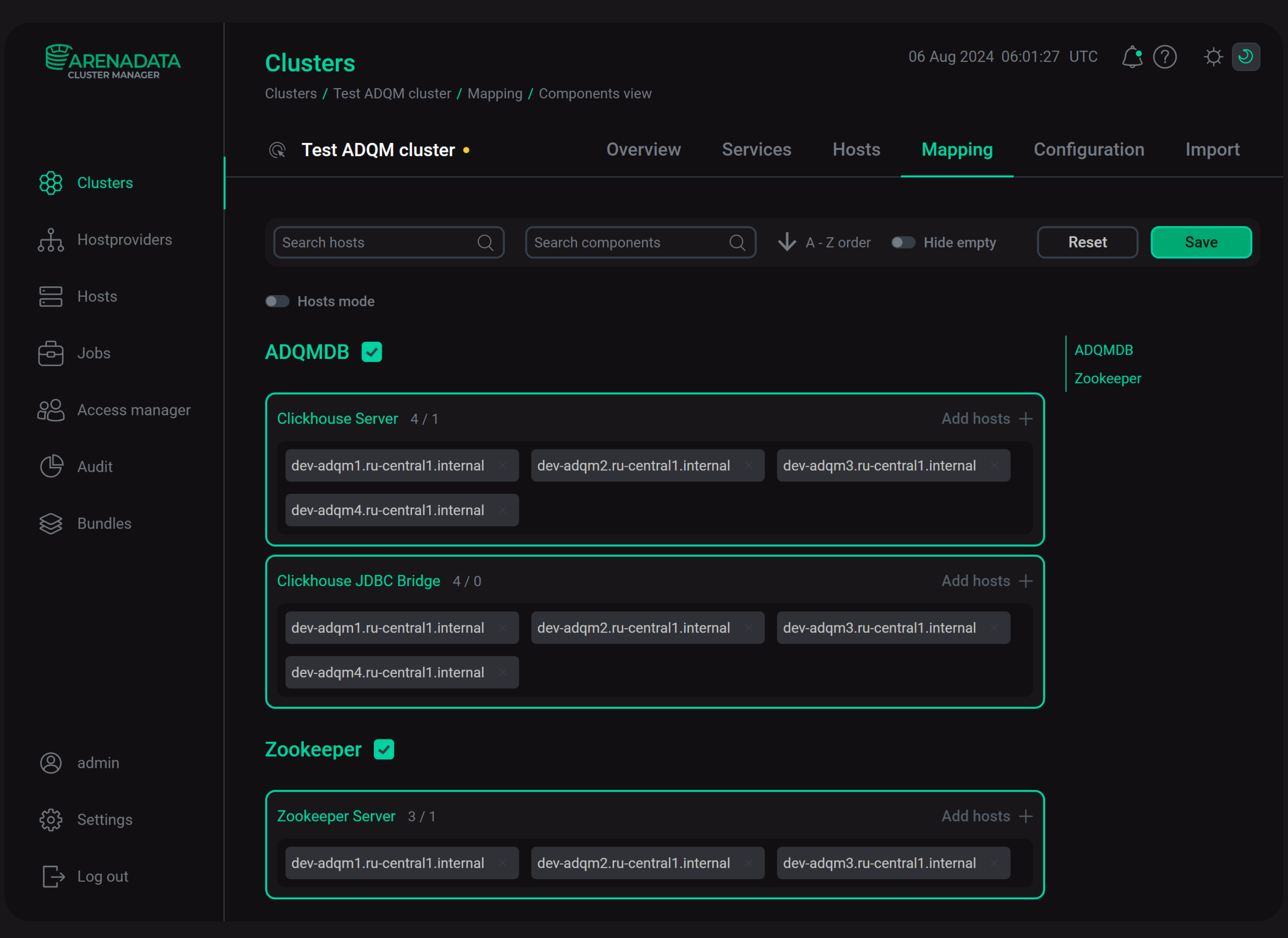
Без назначения нужного числа хостов обязательным компонентам сохранение общей схемы распределения невозможно.

Типовая схема распределения компонентов в кластере ADQM приведена в таблице ниже.

Распределение компонентов ADQM

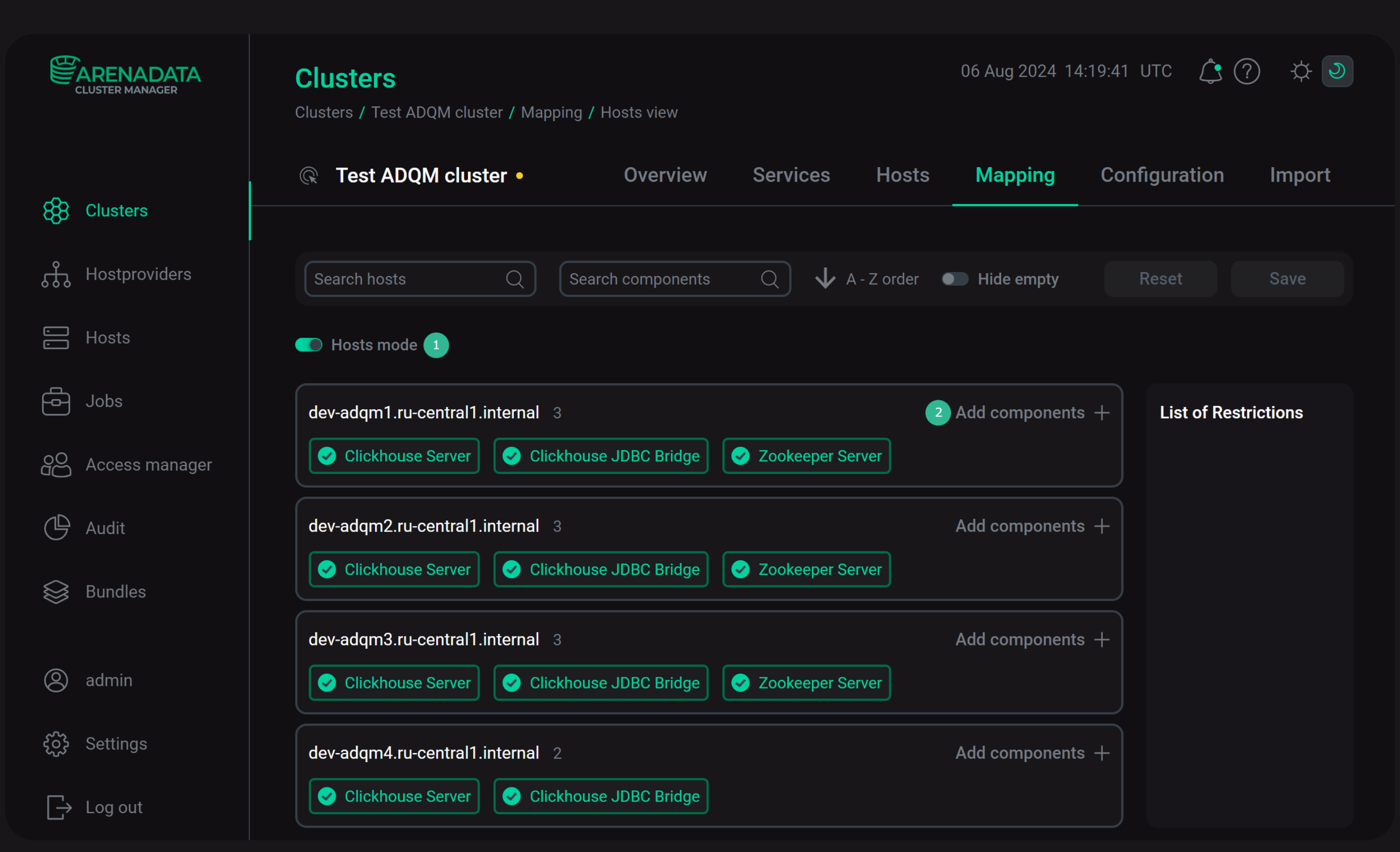
Хост	Сервис	Компонент	Примечание
Любой хост	ADQMDB	Clickhouse Server	Необходимо установить один или несколько компонентов
Все хосты с Clickhouse Server	ADQMDB	Clickhouse JDBC Bridge	При установке этого компонента (опционально) необходимо добавить его на все хосты кластера, где установлен компонент Clickhouse Server
Любой хост	Zookeeper	Zookeeper Server	Должен быть установлен один или несколько компонентов. Суммарное количество должно быть нечетным
Любой хост	Clickhousekeeper	Clickhouse Keeper Server	Должен быть установлен один или несколько компонентов. Суммарное количество должно быть нечетным
Любой хост	Chproxy	Chproxy Proxy Server	Необходимо установить один или несколько компонентов

4. После завершения распределения компонентов нажмите ***Save***.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Mapping / Components view
06 Aug 2024 06:01:27 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
ADQMDB
Clickhouse Server 4 / 1
Add hosts
Clickhouse JDBC Bridge 4 / 0
Add hosts
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
Zookeeper
Zookeeper Server 3 / 1
Add hosts
ADQMDB
Zookeeper

Сохранение распределения компонентов

Начиная с версии 2.2.0 ADCM поддерживает альтернативный способ сопоставления — по хостам. Для этого переведите в активное положение переключатель ***Hosts mode*** и распределите компоненты между хостами путем нажатия на кнопку ***Add components*** напротив каждого хоста.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Mapping / Hosts view
06 Aug 2024 14:19:41 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
1
dev-adqm1.ru-central1.internal 3
2
Add components
Clickhouse Server
Clickhouse JDBC Bridge
Zookeeper Server
dev-adqm2.ru-central1.internal 3
Add components
Clickhouse Server
Clickhouse JDBC Bridge
Zookeeper Server
dev-adqm3.ru-central1.internal 3
Add components
Clickhouse Server
Clickhouse JDBC Bridge
Zookeeper Server
dev-adqm4.ru-central1.internal 2
Add components
Clickhouse Server
Clickhouse JDBC Bridge
List of Restrictions

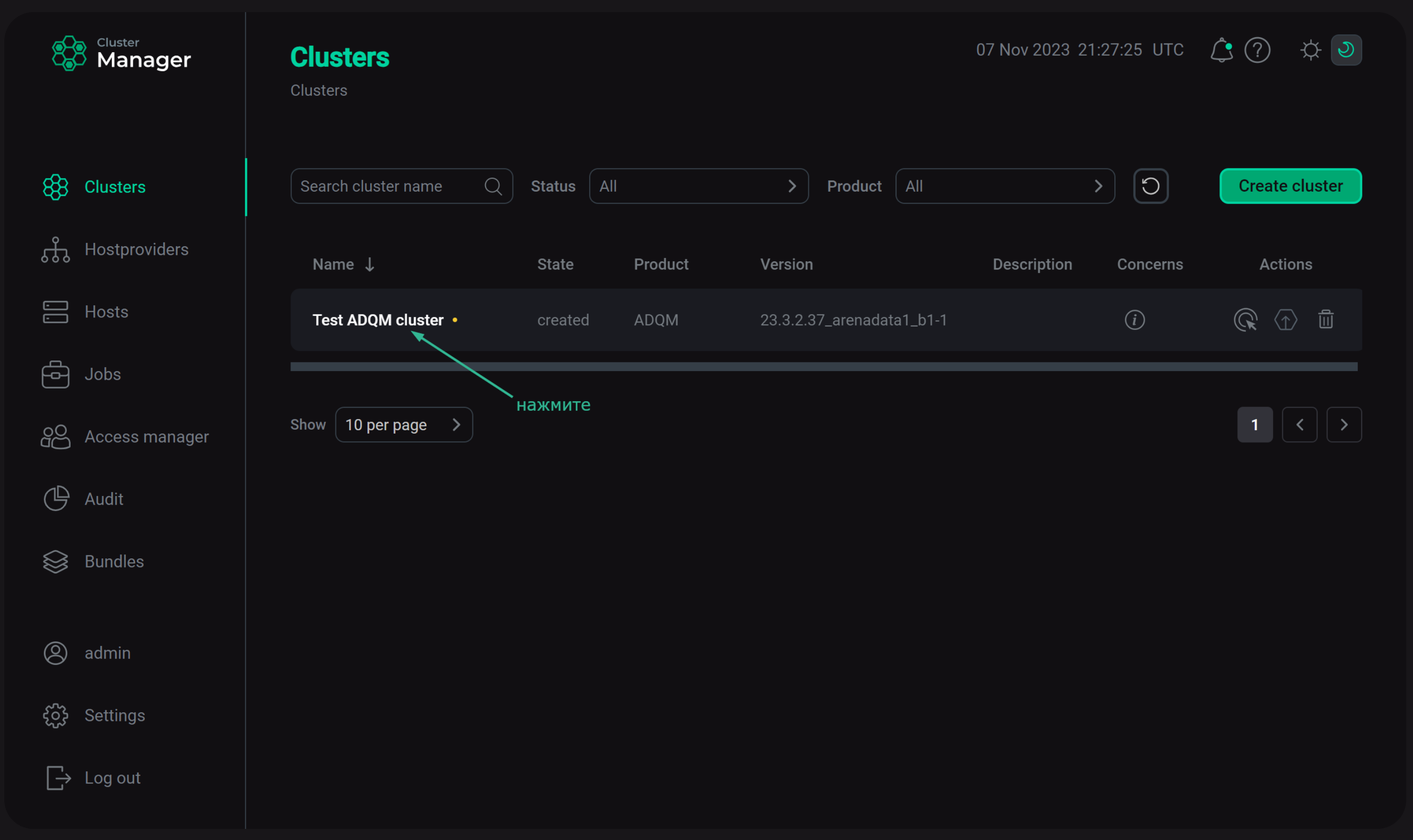
Распределение по хостам

Настройка сервисов

Дарья Барышева, Елена Дворядкина

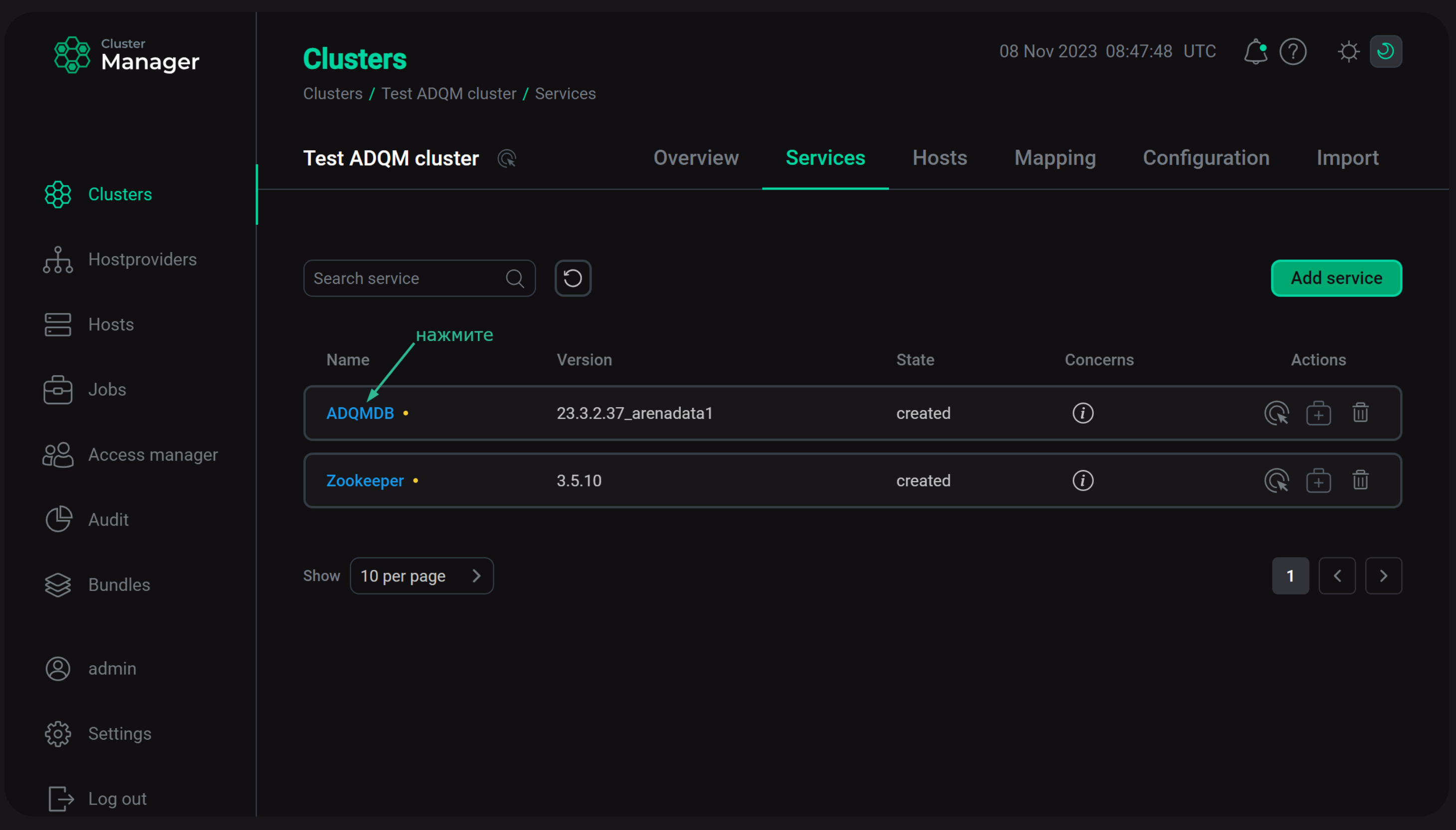
Назад к содержанию

1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test ADQM cluster
created
ADQM
23.3.2.37_arenadata1_b1-1
нажмите
Show
10 per page
1

Выбор кластера

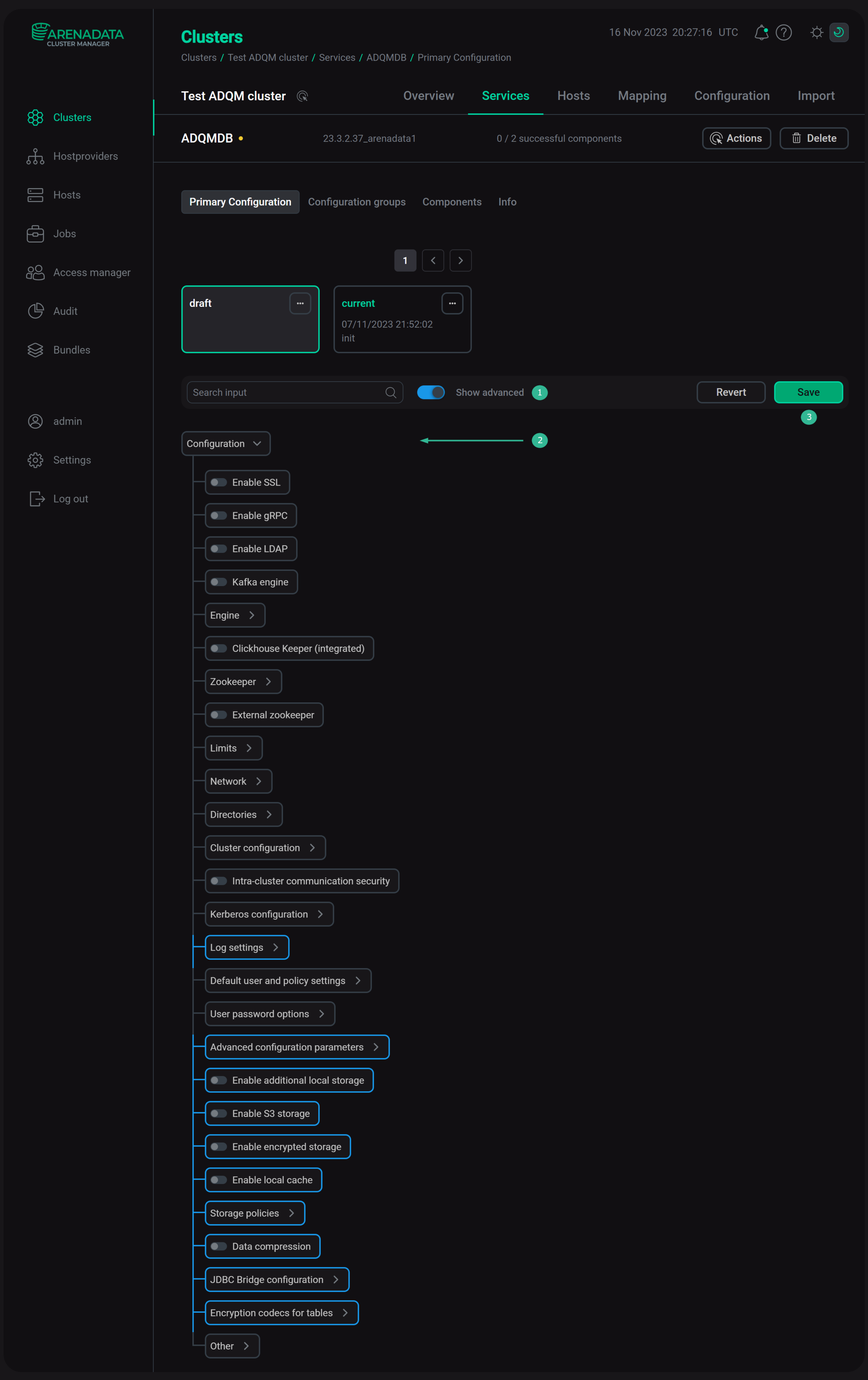
2. Откройте вкладку ***Services*** на странице кластера. Нажмите на имя сервиса в столбце ***Name***.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
08 Nov 2023 08:47:48 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
нажмите
Name
Version
State
Concerns
Actions
ADQMDB
23.3.2.37_arenadata1
created
Zookeeper
3.5.10
created
Show
10 per page
1

Переход к настройке сервиса

3. В открывшемся окне:

   - Откройте вкладку ***Primary Configuration***.
   - Переведите в активное состояние переключатель ***Show advanced***.
   - Заполните конфигурационные параметры выбранного сервиса. Поля, подсвеченные красным, обязательны для заполнения.
   - Нажмите ***Save***.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary Configuration
16 Nov 2023 20:27:16 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
23.3.2.37_arenadata1
0 / 2 successful components
Actions
Delete
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary Configuration
Configuration groups
Components
Info
1
draft
current
07/11/2023 21:52:02
init
Search input
Show advanced
1
Revert
Save
3
2
Configuration
Enable SSL
Enable gRPC
Enable LDAP
Kafka engine
Engine
Clickhouse Keeper (integrated)
Zookeeper
External zookeeper
Limits
Network
Directories
Cluster configuration
Intra-cluster communication security
Kerberos configuration
Log settings
Default user and policy settings
User password options
Advanced configuration parameters
Enable additional local storage
Enable S3 storage
Enable encrypted storage
Enable local cache
Storage policies
Data compression
JDBC Bridge configuration
Encryption codecs for tables
Other

Настройка сервиса

ПРИМЕЧАНИЕ

Более подробную информацию по всем конфигурационным параметрам можно получить в статье Конфигурационные параметры.

Настройка кластера

Дарья Барышева, Елена Дворядкина

Содержание

Назад к содержанию

Primary configuration

Configuration groups

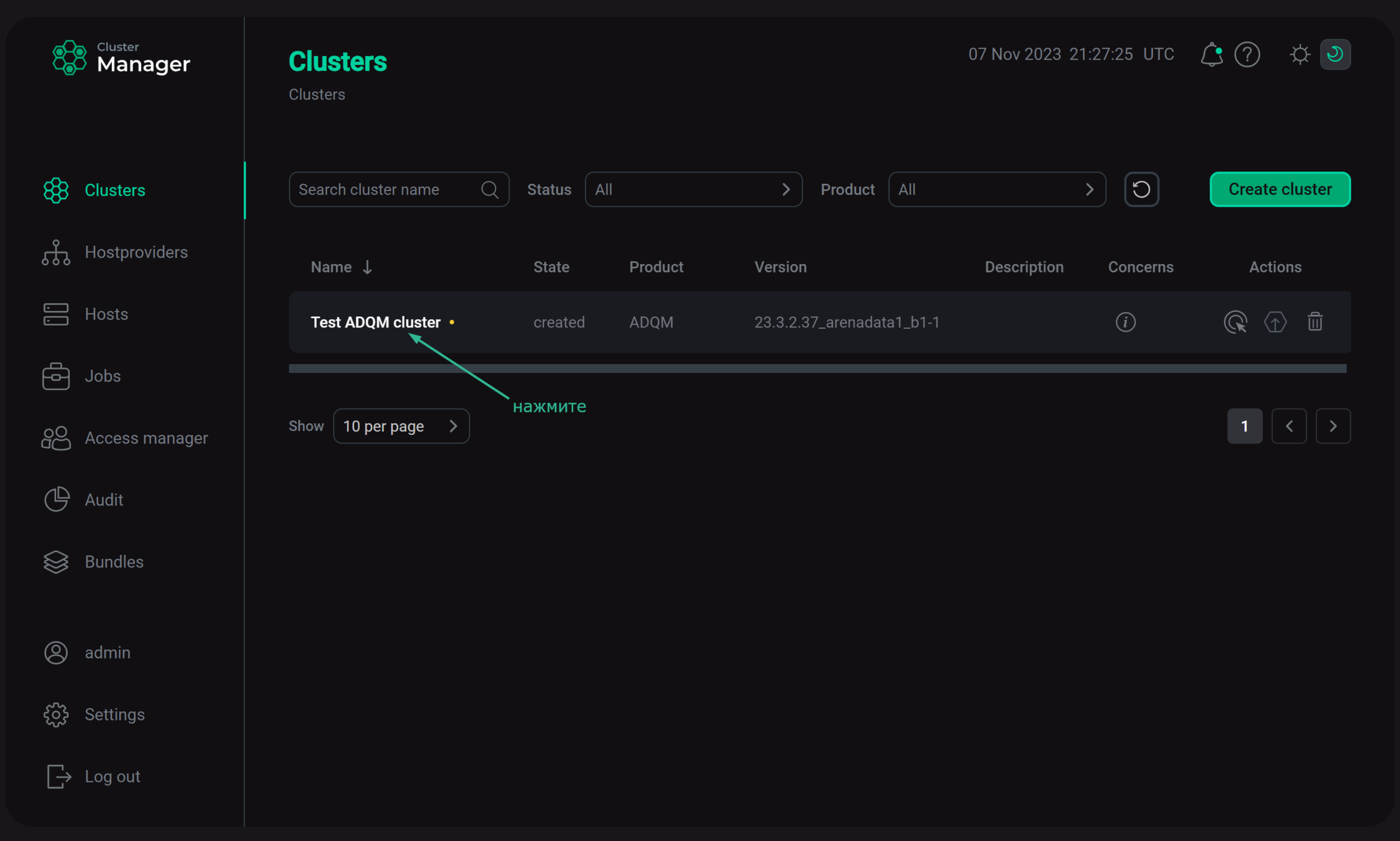
Ansible settings

ВАЖНО

В большинстве случаев кастомная настройка кластера не требуется — значения параметров можно оставить по умолчанию.

После добавления нового кластера можно выполнить его настройку. Для этого выполните шаги:

1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.



Cluster Manager
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Search cluster name
Status
All
Product
All
Create cluster
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
нажмите
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

Выбор кластера

2. Откройте вкладку ***Configuration*** на странице кластера. Вкладка ***Configuration*** состоит из трех подразделов: ***Primary configuration***, ***Configuration groups***, ***Ansible settings***.

3. Заполните необходимые конфигурационные параметры на нужной вкладке и нажмите ***Save***.

Primary configuration

Вкладка ***Configuration*** → ***Primary configuration*** содержит основные конфигурационные параметры кластера.

Доступные для редактирования конфигурационные параметры кластера ADQM перечислены ниже:

- ***Custom JRE*** — переключатель, определяющий необходимость использования кастомного пути к Java для сервисов кластера ADQM. При активации переключателя становится доступно для заполнения поле ***JAVA_HOME***, в котором требуется указать путь к работающей Java. Никакие системные пакеты JRE при этом устанавливаться не будут.

- ***Disable SElinux*** — отключает SElinux на хостах кластера. Параметр доступен начиная с версии ADQM 22.8.4.7.1.b1.

- ***Disable Firewalld*** — отключает firewalld на хостах кластера. Параметр доступен начиная с версии ADQM 22.8.4.7.1.b1.

- ***Reliability control*** — секция доступна при включенной опции ***Show advanced***. Содержит следующие настройки:

  - ***retries*** — количество попыток для операций в кластере (по умолчанию `3`);

  - ***delay*** — задержка между попытками в секундах (по умолчанию `10`);

  - ***timeout*** — тайм-аут проверки сетевых портов в секундах (по умолчанию `60`).

- Пути к репозиториям, которые будут использованы в процессе установки в зависимости от операционной системы кластера (Centos, Alt Linux, Red Hat, Ubuntu, Astra Linux или РЕД ОС):

  - ***monitoring*** — репозиторий для установки мониторинга;

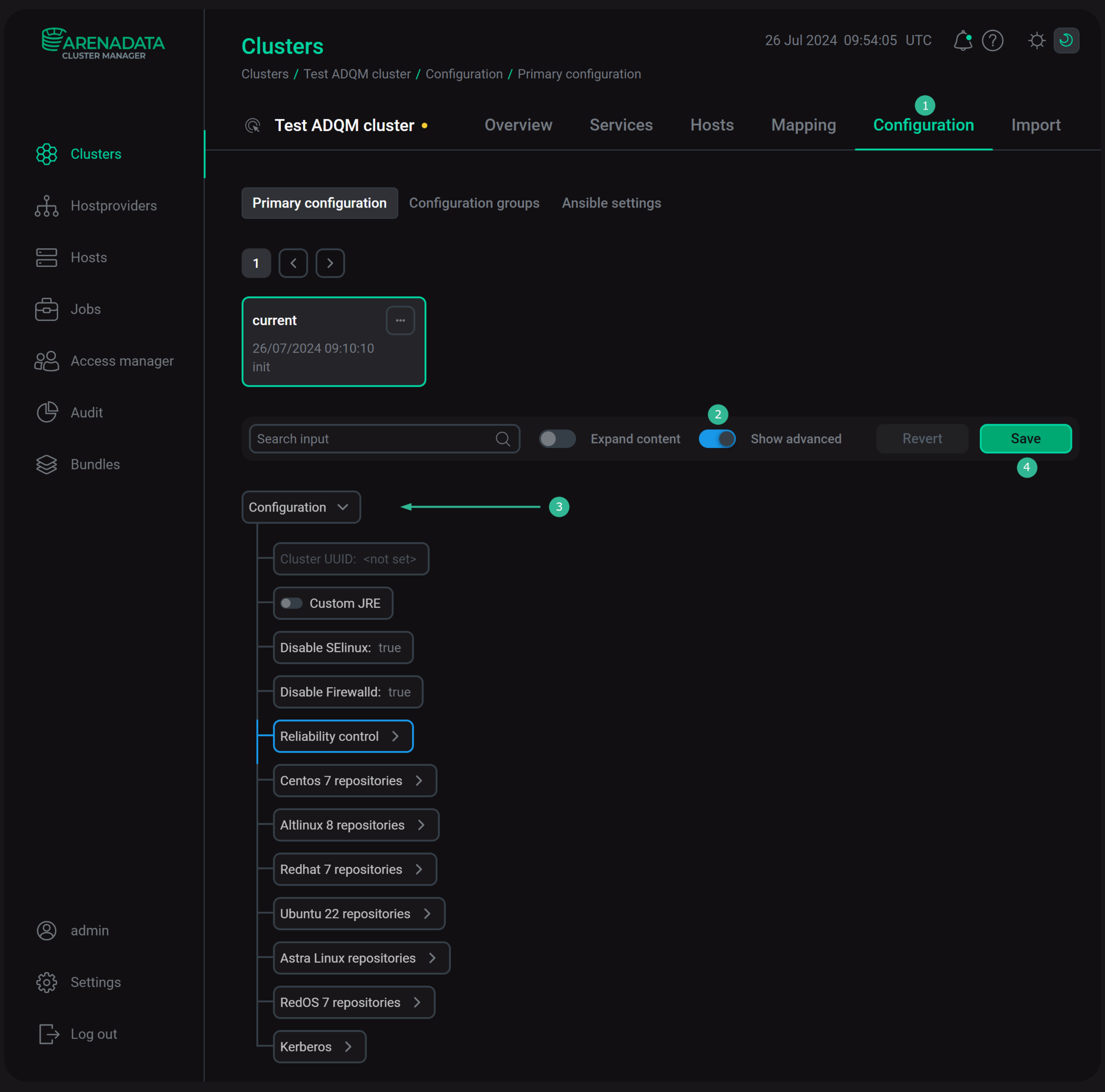
  - ***zookeeper_repo*** — репозиторий для установки Arenadata Zookeeper;

  - ***ADQM*** — репозиторий для установки ADQM;

  - ***prometheus_repo*** — репозиторий для установки Prometheus;

  - ***ADQM_arenadata_enterprise*** — репозиторий для установки Enterprise-версии ADQM.

- ***Kerberos*** — конфигурационные параметры для аутентификации по протоколу Kerberos (некоторые параметры становятся видимыми только при включенной опции ***Show advanced***).



ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Configuration / Primary configuration
26 Jul 2024 09:54:05 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
1
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
Ansible settings
1
Search input
Expand content
2
Show advanced
Revert
Save
4
Configuration
3
Reliability control
Centos 7 repositories
Altlinux 8 repositories
Redhat 7 repositories
Ubuntu 22 repositories
Astra Linux repositories
RedOS 7 repositories
Kerberos

current
26/07/2024 09:10:10
init

Cluster UUID: <not set>
Custom JRE
Disable SElinux: true
Disable Firewalld: true

Конфигурирование кластера

Configuration groups

Вкладка ***Configuration*** → ***Configuration groups*** предназначена для настройки конфиг-групп кластера.

Ansible settings

Вкладка ***Configuration*** → ***Ansible settings*** предназначена для ввода конфигурационных настроек Ansible на уровне кластера. Вкладка доступна начиная с версии ADCM 2.2.0.

Параметры Ansible

Имя	Описание	По умолчанию
forks	Число параллельных процессов, генерируемых при коммуникации с удаленными хостами	5

Установка кластера

Дарья Барышева, Елена Дворядкина

Содержание

Назад к содержанию

Шаг 1. Запуск установки
- Установка всего кластера
- Установка отдельных сервисов

Шаг 2. Просмотр результатов установки

Шаг 3. Проверка статуса кластера после установки

Шаг 1. Запуск установки

Существует два способа установки:

1. **Установка всего кластера**. В этом методе все сервисы кластера устанавливаются автоматически друг за другом после применения действия ***Install*** к кластеру.
2. **Установка отдельных сервисов**. В этом методе каждый сервис устанавливается вручную путем применения к нему одного или нескольких действий. Этот способ также можно использовать для добавления новых сервисов в уже развернутый кластер.

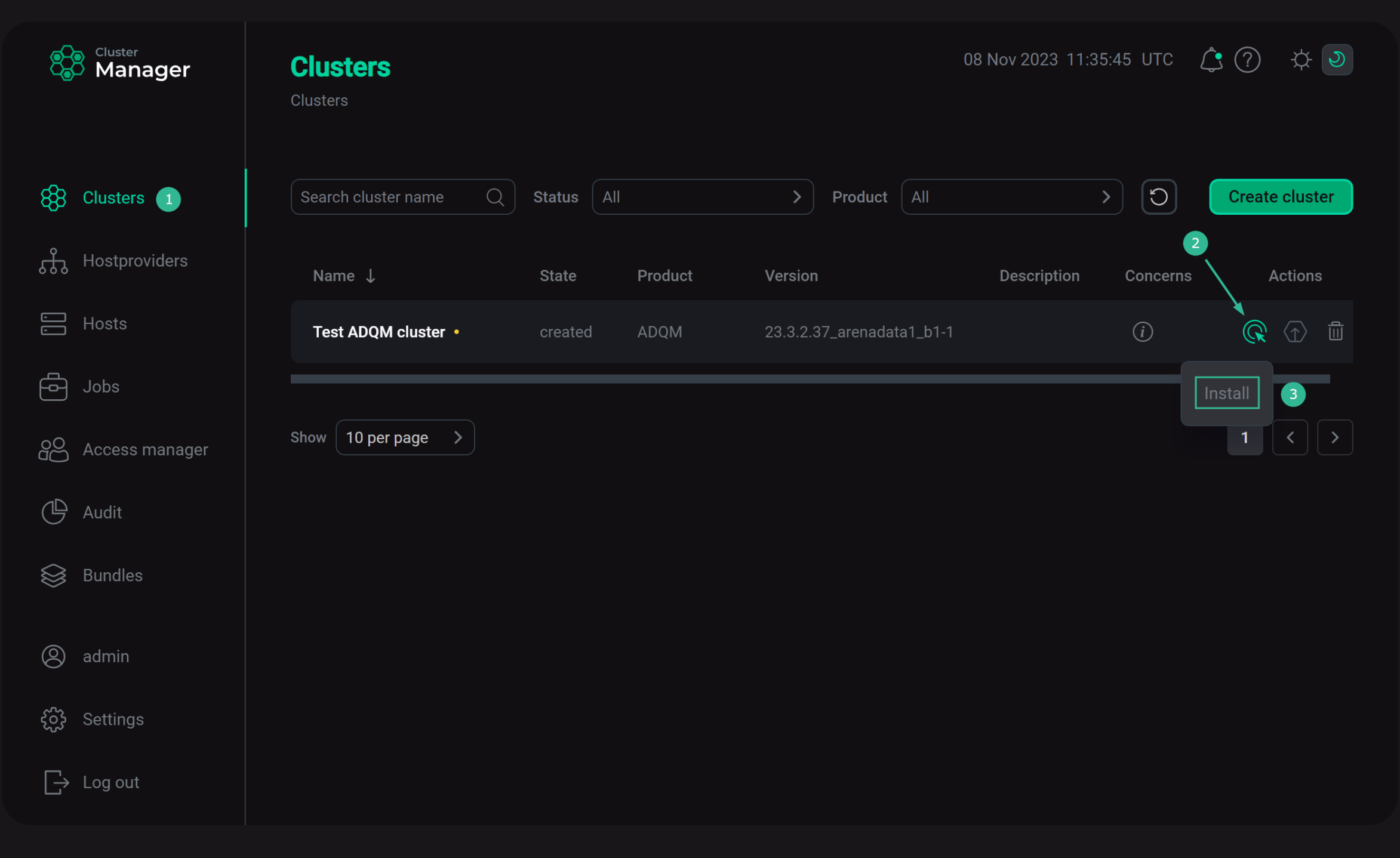
В обоих методах каждый сервис запускается автоматически после установки.

> **ПРИМЕЧАНИЕ**
>
> В случае ошибок логи можно найти на странице ***Jobs***.

Установка всего кластера

Чтобы установить все сервисы кластера за одну операцию, выполните шаги:

1. Примените действие ***Install*** к кластеру, выбранному на странице ***Clusters***, нажав на иконку в столбце ***Actions***.

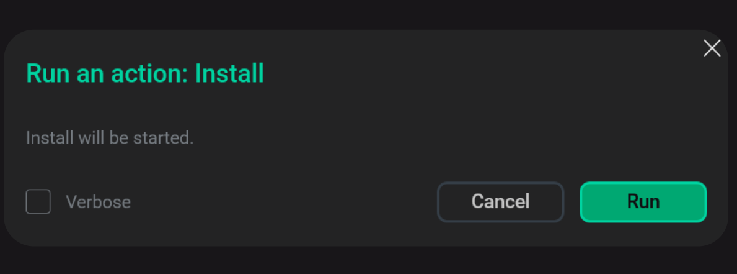


Cluster Manager
Clusters
Clusters
08 Nov 2023 11:35:45 UTC
Clusters 1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
2
Install
3
Show 10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

Переход к установке всего кластера

2. Подтвердите действие в открывшемся окне.



Run an action: Install

Install will be started.

☐ Verbose

Подтверждение действия

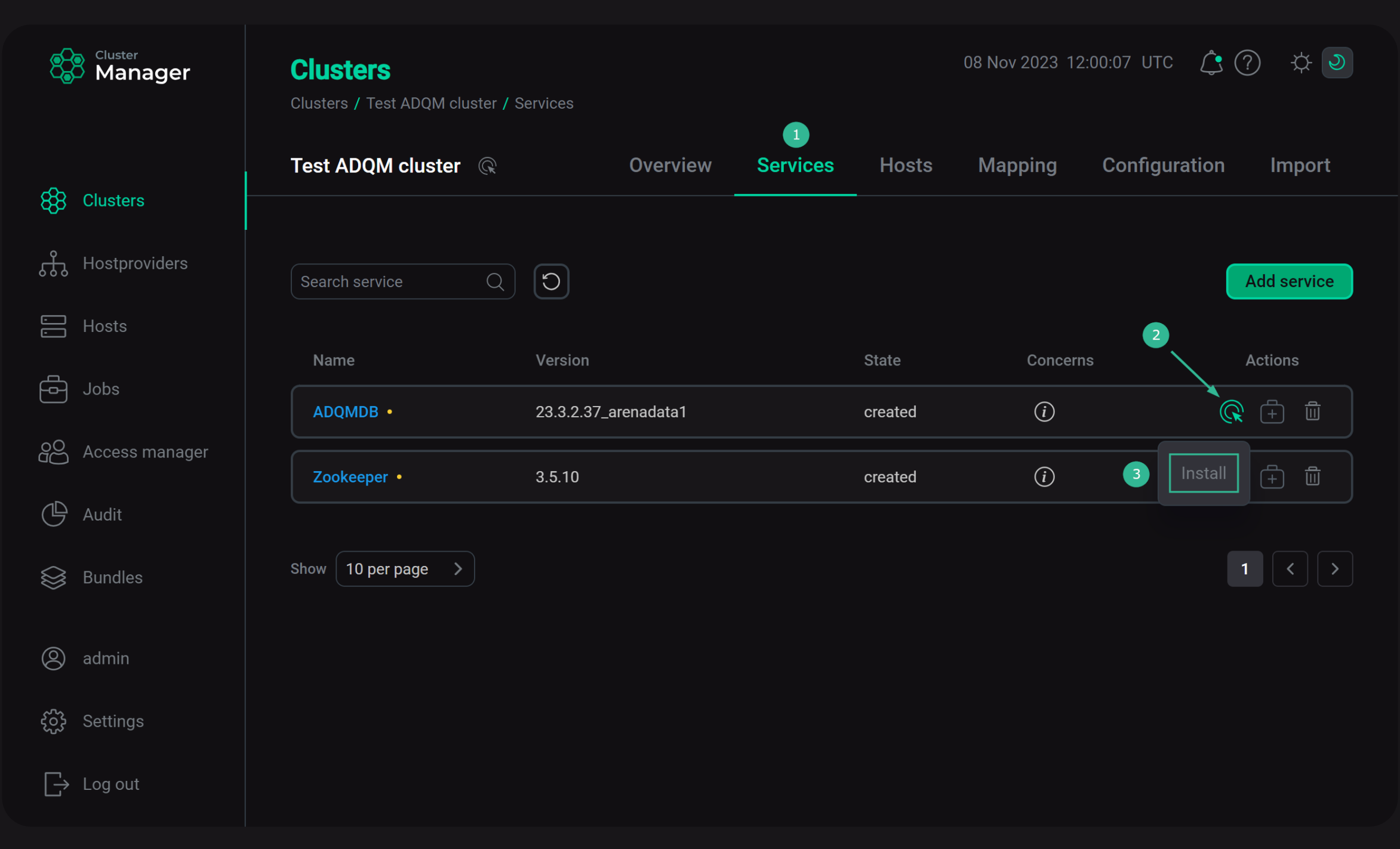
3. Дождитесь завершения установки.

Установка отдельных сервисов

Чтобы установить один или несколько сервисов вручную, необходимо предварительно добавить эти сервисы в кластер и затем применить к ним действия, описанные ниже.

> **ВНИМАНИЕ**
>
> Обратите внимание, что сервисы устанавливаются по очереди друг за другом. Не пытайтесь установить следующий сервис до окончания установки предыдущего.

1. Откройте вкладку ***Services*** на странице кластера. Для сервиса, который должен быть установлен, нажмите на иконку в столбце ***Actions*** и выберите действие ***Install*** из выпадающего списка.

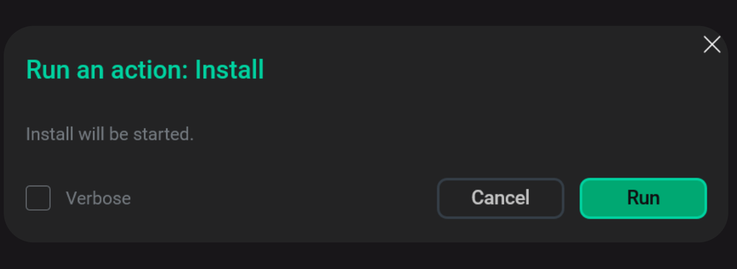


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
08 Nov 2023 12:00:07 UTC
Test ADQM cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
2
3
Install
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADQMDB	23.3.2.37_arenadata1	created		
Zookeeper	3.5.10	created		

Переход к установке отдельных сервисов

2. Подтвердите действие в открывшемся окне.



Run an action: Install

Install will be started.

☐ Verbose

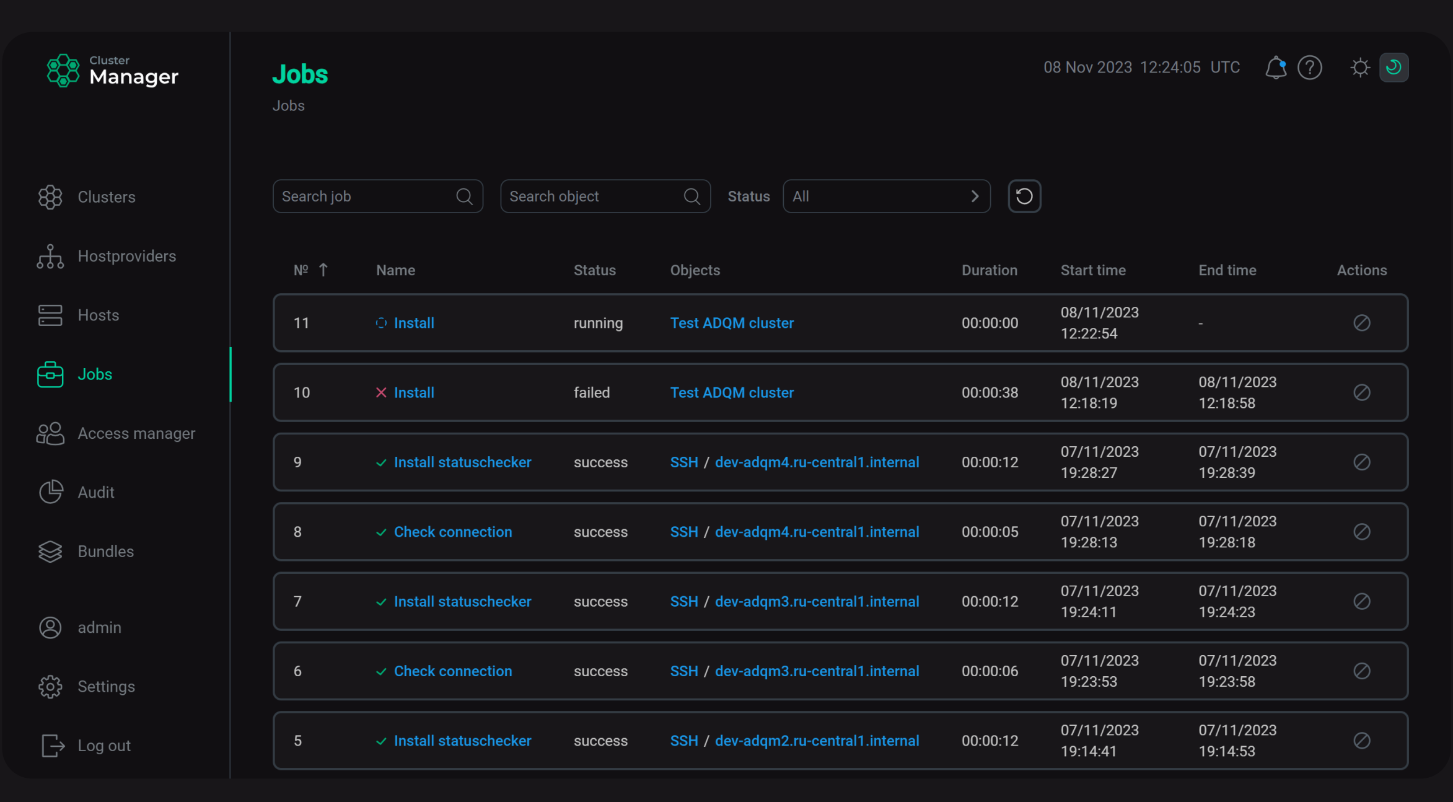
Подтверждение действия

3. Дождитесь завершения установки и убедитесь, что статус сервиса изменился с `created` на `installed`.
4. Повторите предыдущие шаги для всех сервисов, которые необходимо установить.

> **ВАЖНО**
>
> При установке сервисов вручную важно соблюдать следующий порядок:
>
> 1. Zookeeper / Clickhousekeeper
> 2. ADQMDB
> 3. Chproxy
> 4. Monitoring Clients / Monitoring

Шаг 2. Просмотр результатов установки

ADCM запускает отдельную задачу (job) для установки, как и для любого другого процесса. Для получения информации о статусе задач предназначена страница ***Jobs***.

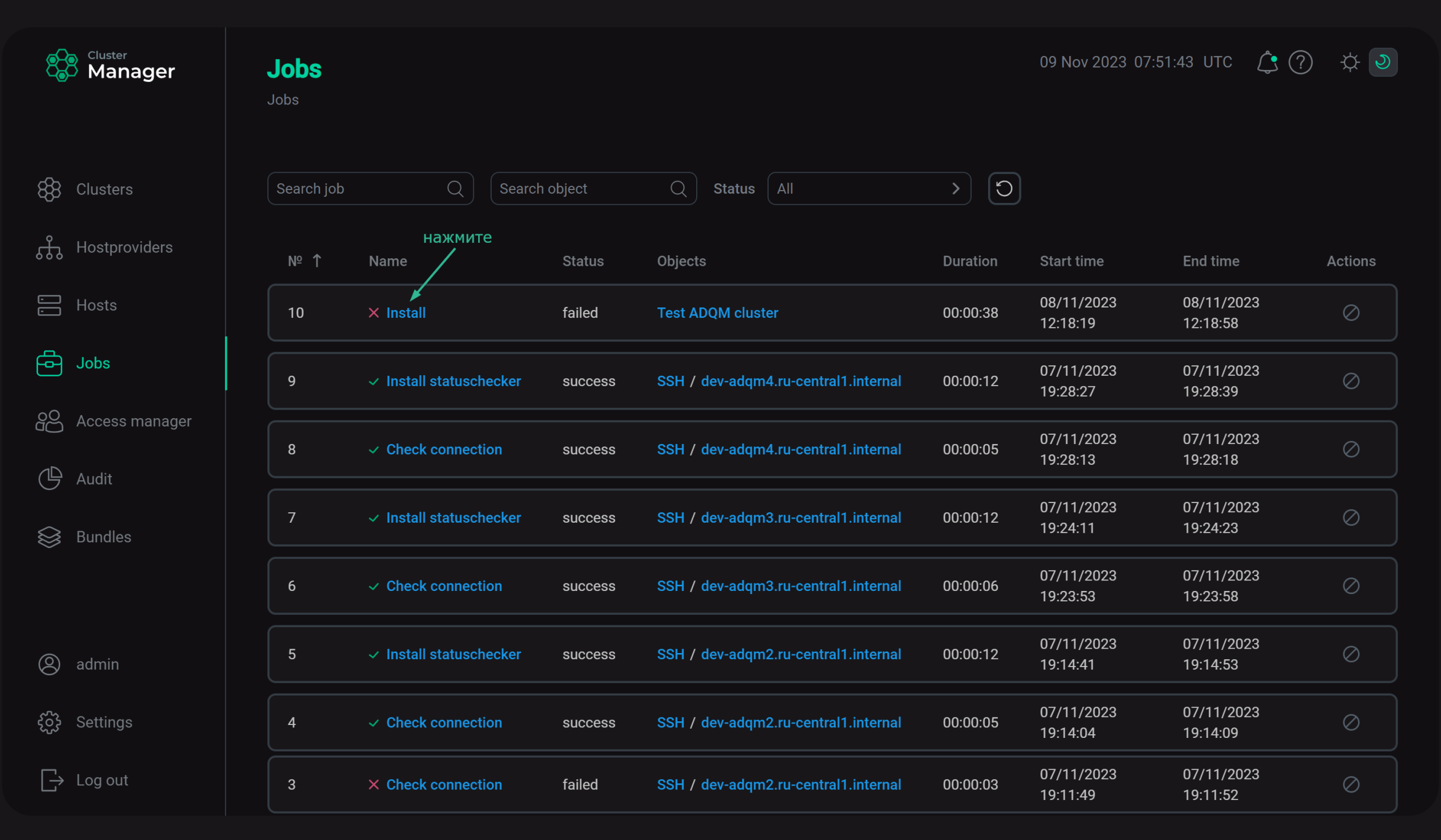


Cluster Manager
Jobs
Jobs
08 Nov 2023 12:24:05 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search job
Search object
Status All

№	Name	Status	Objects	Duration	Start time	End time	Actions
11	Install	running	Test ADQM cluster	00:00:00	08/11/2023 12:22:54	-	
10	Install	failed	Test ADQM cluster	00:00:38	08/11/2023 12:18:19	08/11/2023 12:18:58	
9	Install statuschecker	success	SSH / dev-adqm4.ru-central1.internal	00:00:12	07/11/2023 19:28:27	07/11/2023 19:28:39	
8	Check connection	success	SSH / dev-adqm4.ru-central1.internal	00:00:05	07/11/2023 19:28:13	07/11/2023 19:28:18	
7	Install statuschecker	success	SSH / dev-adqm3.ru-central1.internal	00:00:12	07/11/2023 19:24:11	07/11/2023 19:24:23	
6	Check connection	success	SSH / dev-adqm3.ru-central1.internal	00:00:06	07/11/2023 19:23:53	07/11/2023 19:23:58	
5	Install statuschecker	success	SSH / dev-adqm2.ru-central1.internal	00:00:12	07/11/2023 19:14:41	07/11/2023 19:14:53	

Страница Jobs

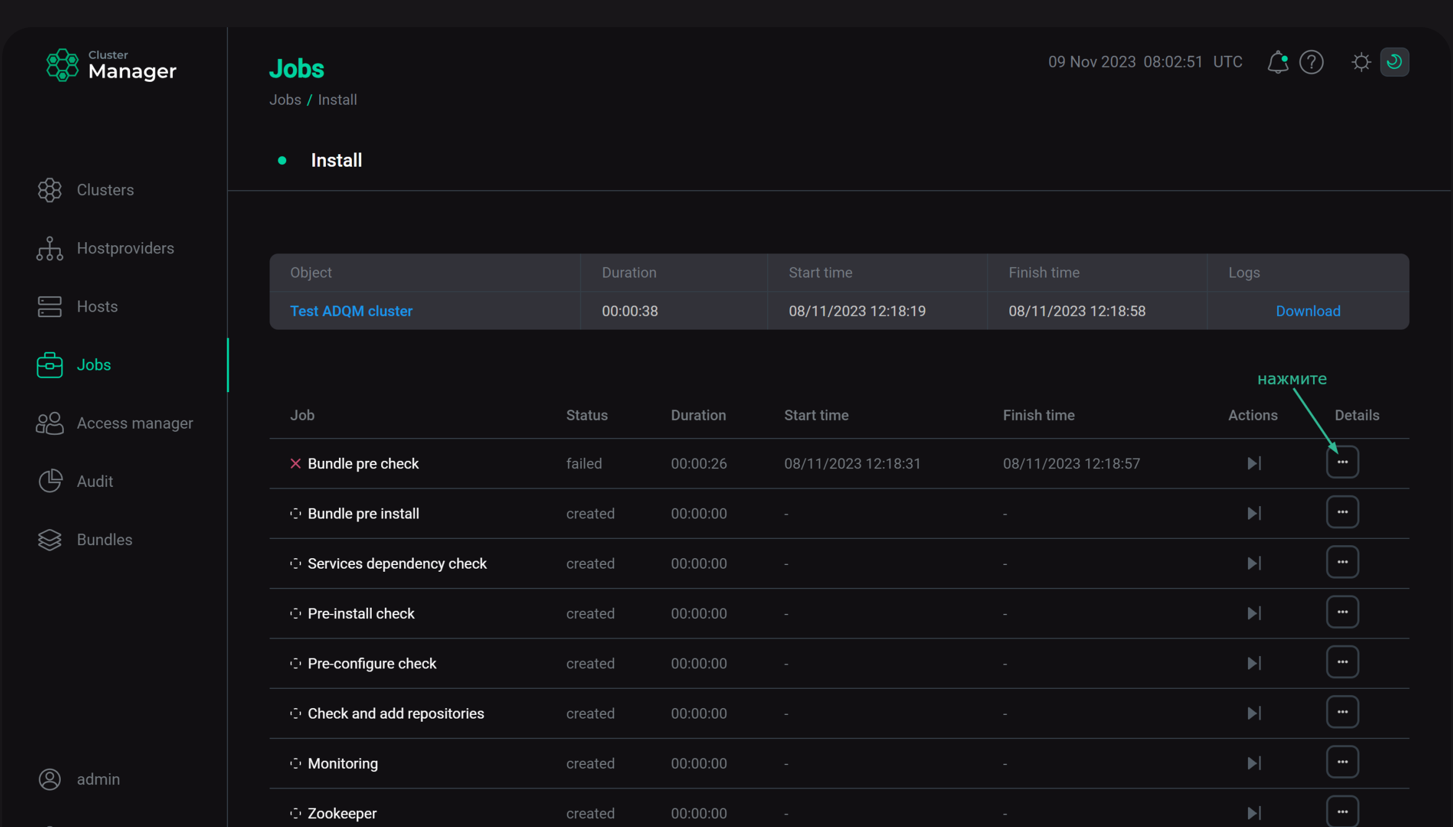
Если установка завершается успешно, соответствующая ей задача на странице ***Jobs*** переходит из статуса `running` в `success`; в противном случае — в статус `failed`. Чтобы получить более детальную информацию о возникших ошибках, можно перейти на страницу задачи, кликнув по ней в списке задач на странице ***Jobs***.



Cluster Manager
Jobs
Jobs
09 Nov 2023 07:51:43 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search job
Search object
Status All
нажмите

№	Name	Status	Objects	Duration	Start time	End time	Actions
10	Install	failed	Test ADQM cluster	00:00:38	08/11/2023 12:18:19	08/11/2023 12:18:58	
9	Install statuschecker	success	SSH / dev-adqm4.ru-central1.internal	00:00:12	07/11/2023 19:28:27	07/11/2023 19:28:39	
8	Check connection	success	SSH / dev-adqm4.ru-central1.internal	00:00:05	07/11/2023 19:28:13	07/11/2023 19:28:18	
7	Install statuschecker	success	SSH / dev-adqm3.ru-central1.internal	00:00:12	07/11/2023 19:24:11	07/11/2023 19:24:23	
6	Check connection	success	SSH / dev-adqm3.ru-central1.internal	00:00:06	07/11/2023 19:23:53	07/11/2023 19:23:58	
5	Install statuschecker	success	SSH / dev-adqm2.ru-central1.internal	00:00:12	07/11/2023 19:14:41	07/11/2023 19:14:53	
4	Check connection	success	SSH / dev-adqm2.ru-central1.internal	00:00:05	07/11/2023 19:14:04	07/11/2023 19:14:09	
3	Check connection	failed	SSH / dev-adqm2.ru-central1.internal	00:00:03	07/11/2023 19:11:49	07/11/2023 19:11:52	

Переход на страницу задачи (job)



Cluster Manager
Jobs
Jobs / Install
09 Nov 2023 08:02:51 UTC
Install
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
нажмите

Object	Duration	Start time	Finish time	Logs
Test ADQM cluster	00:00:38	08/11/2023 12:18:19	08/11/2023 12:18:58	Download

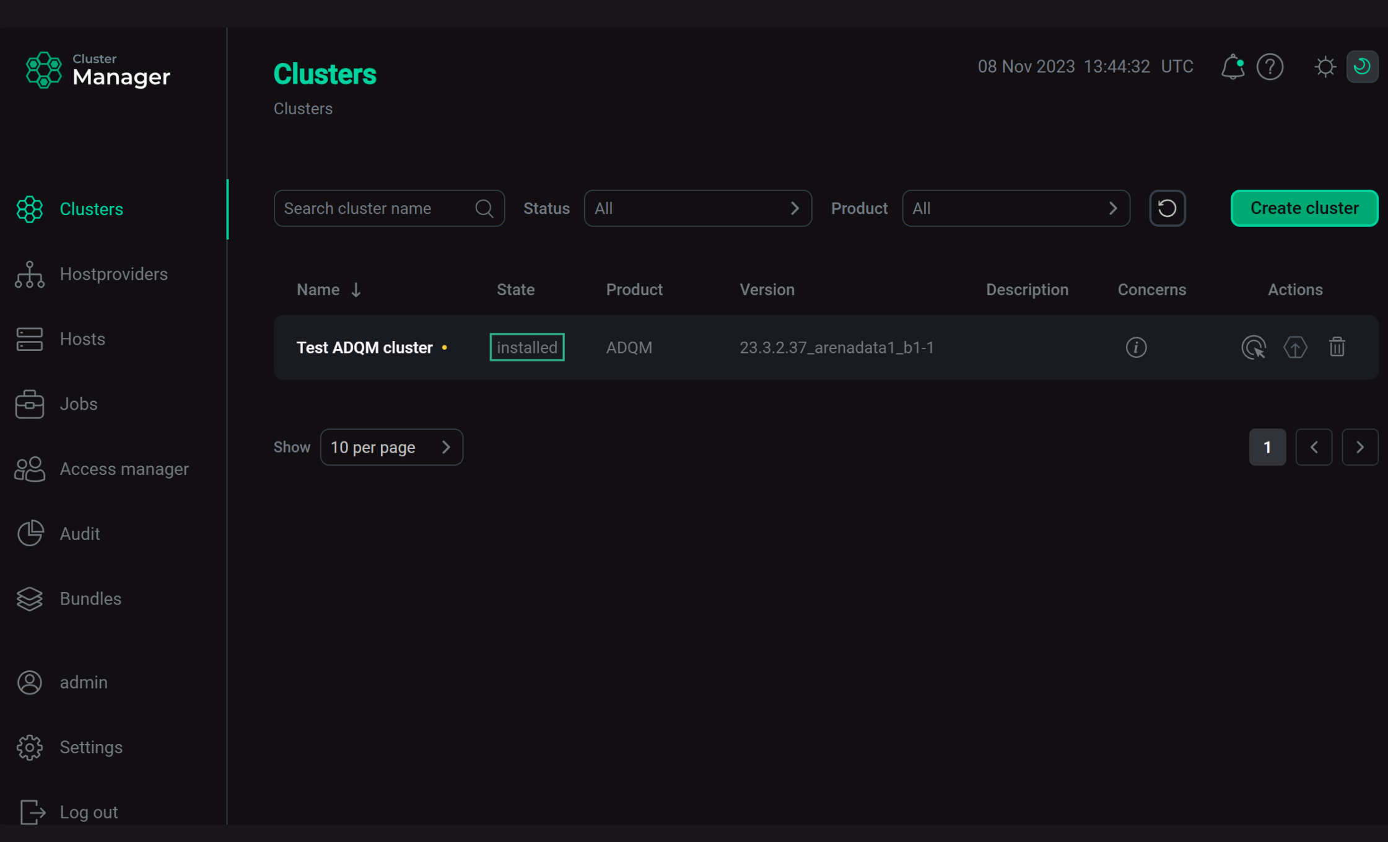
Job	Status	Duration	Start time	Finish time	Actions	Details
Bundle pre check	failed	00:00:26	08/11/2023 12:18:31	08/11/2023 12:18:57		
Bundle pre install	created	00:00:00	-	-		
Services dependency check	created	00:00:00	-	-		
Pre-install check	created	00:00:00	-	-		
Pre-configure check	created	00:00:00	-	-		
Check and add repositories	created	00:00:00	-	-		
Monitoring	created	00:00:00	-	-		
Zookeeper	created	00:00:00	-	-		
Clickhousekeeper	created	00:00:00	-	-		
ADQMDB	created	00:00:00	-	-		

Страница задачи установки кластера

Шаг 3. Проверка статуса кластера после установки

В результате успешной установки происходит обновление статуса кластера и его сервисов:

- Статус кластера, отображаемый в столбце ***State*** на странице ***Clusters***, меняет свое значение с `created` на `installed`.

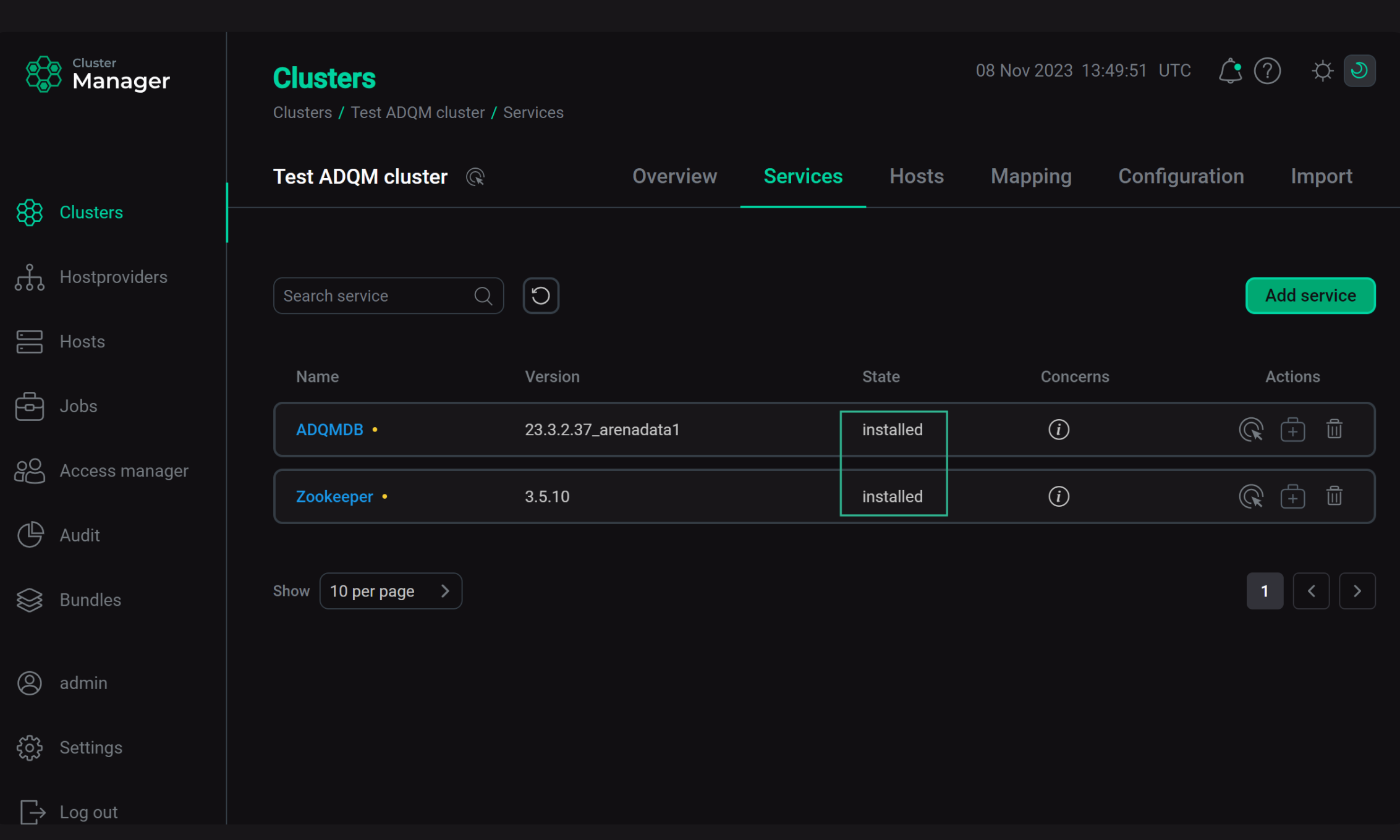


Cluster Manager
Clusters
Clusters
08 Nov 2023 13:44:32 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	installed	ADQM	23.3.2.37_arenadata1_b1-1			

Статус кластера после успешной установки

- Статус сервисов, отображаемый на вкладке ***Services*** страницы кластера, меняет свое значение с `created` на `installed`.



Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
08 Nov 2023 13:49:51 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADQMDB	23.3.2.37_arenadata1	installed		
Zookeeper	3.5.10	installed		

Статус сервисов кластера после успешной установки

Сервис мониторинга

Елена Дворядкина

Содержание

Назад к содержанию

Обзор
Шаг 1. Добавление сервиса
Шаг 2. Добавление компонентов
Шаг 3. Настройка сервиса
Шаг 4. Установка сервиса
Шаг 5. Проверка результатов
- Данные в формате Prometheus
- Веб-интерфейс Prometheus
- Веб-интерфейс Grafana

Сервис Monitoring (доступен начиная с версии ADQM 23.8.2.7) разворачивает в ADQM свой сервер Prometheus, который реализует сбор и хранение метрик мониторинга кластера ADQM, а также поддерживает возможность использования веб-приложения Grafana для визуализации и анализа информации. В данной статье описываются шаги, необходимые для подключения этого сервиса.

Обзор

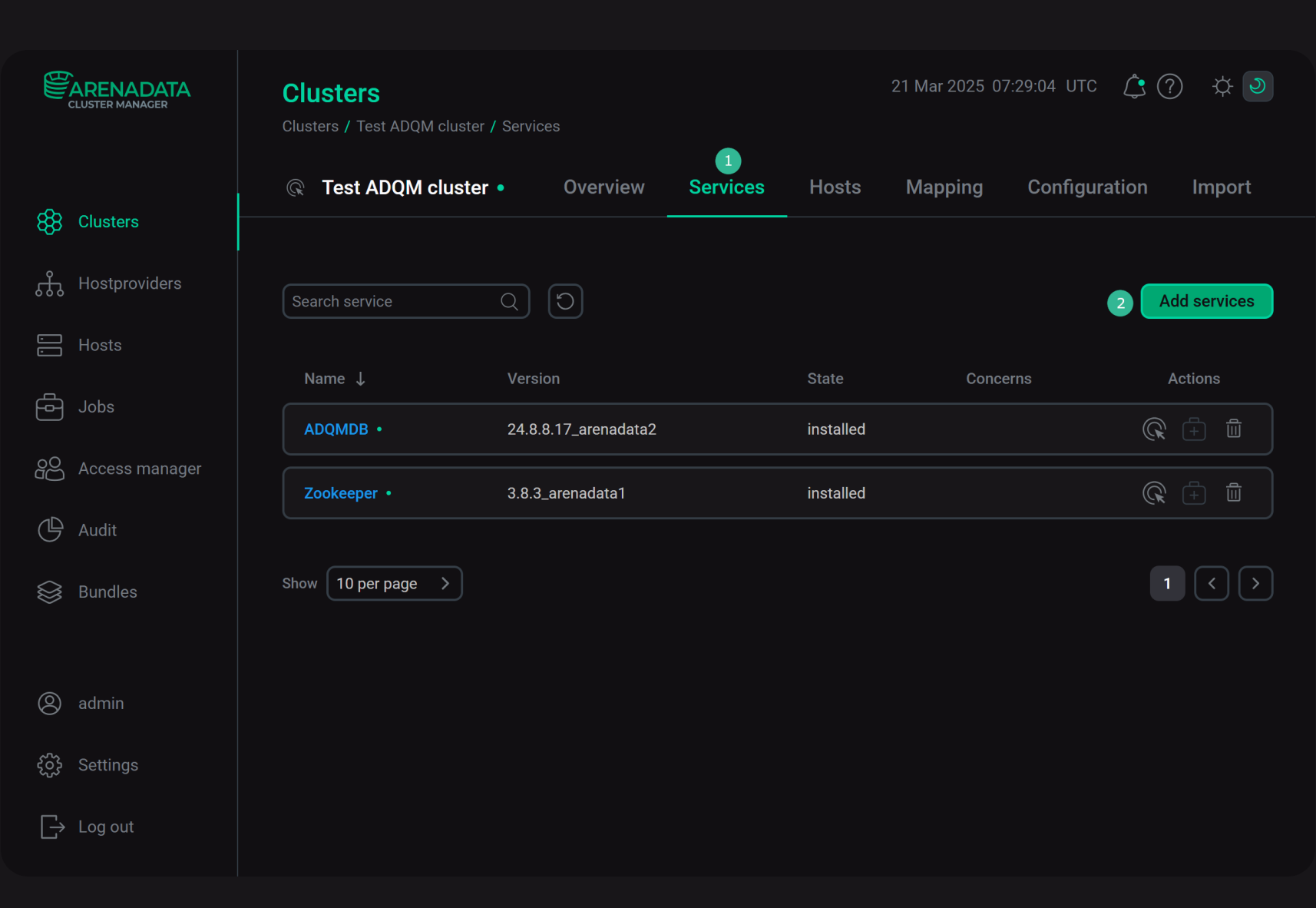
При добавлении сервиса Monitoring в ADQM на все или выборочные хосты кластера можно установить node exporter — агент мониторинга, считывающий с хоста системные метрики, которые будет собирать Prometheus. Prometheus также может собирать метрики сервисов ADQM (ClickHouse, ZooKeeper, ClickHouse Keeper, Chproxy). Эти метрики будут доступны в формате Prometheus на портах и конечных точках (endpoints), указанных в параметрах секции ***Metrics*** сервисных конфигураций. Просматривать и анализировать данные, которые собирает сервис мониторинга, можно также в веб-интерфейсе Prometheus или Grafana.

ПРИМЕЧАНИЕ

- Если у вас уже настроена совместимая с Prometheus система мониторинга (например, свой сервер Prometheus или VictoriaMetrics), можно использовать ее, чтобы собирать метрики ADQM. Для этого в своей системе мониторинга укажите настройки доступа к метрикам ADQM, используя параметры установленного в ADQM Prometheus, которые конфигурируются на странице сервиса Monitoring в интерфейсе ADCM.
- Вы также можете использовать механизм Federation, чтобы перенести все метрики из сервера Prometheus, развернутого в ADQM, в свой Prometheus.

Шаг 1. Добавление сервиса

1. В интерфейсе ADCM откройте страницу ***Clusters*** и кликните по имени кластера ADQM. Затем на открывшейся странице кластера перейдите на вкладку ***Services*** и нажмите ***Add services***.

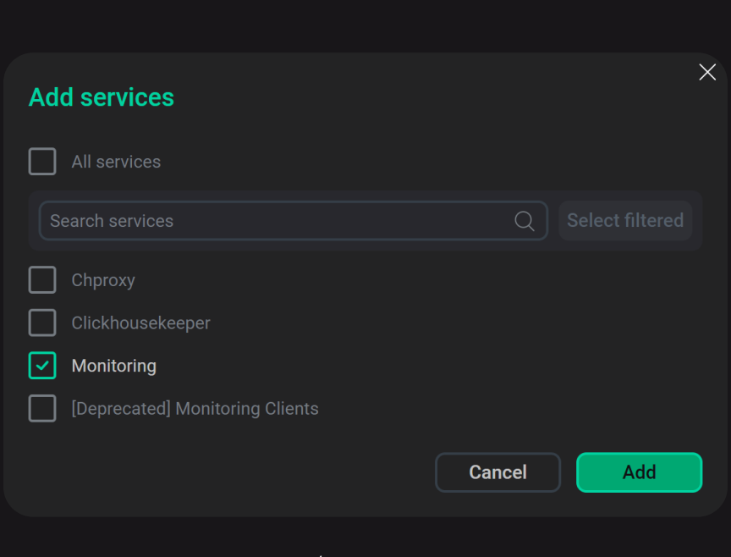


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 07:29:04 UTC
Test ADQM cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Hostproviders
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
2
Add services

Name	Version	State	Concerns	Actions
ADQMDB	24.8.8.17_arenadata2	installed		
Zookeeper	3.8.3_arenadata1	installed		

Переход к добавлению сервисов

2. В открывшемся окне выберите сервис ***Monitoring*** и нажмите ***Add***.

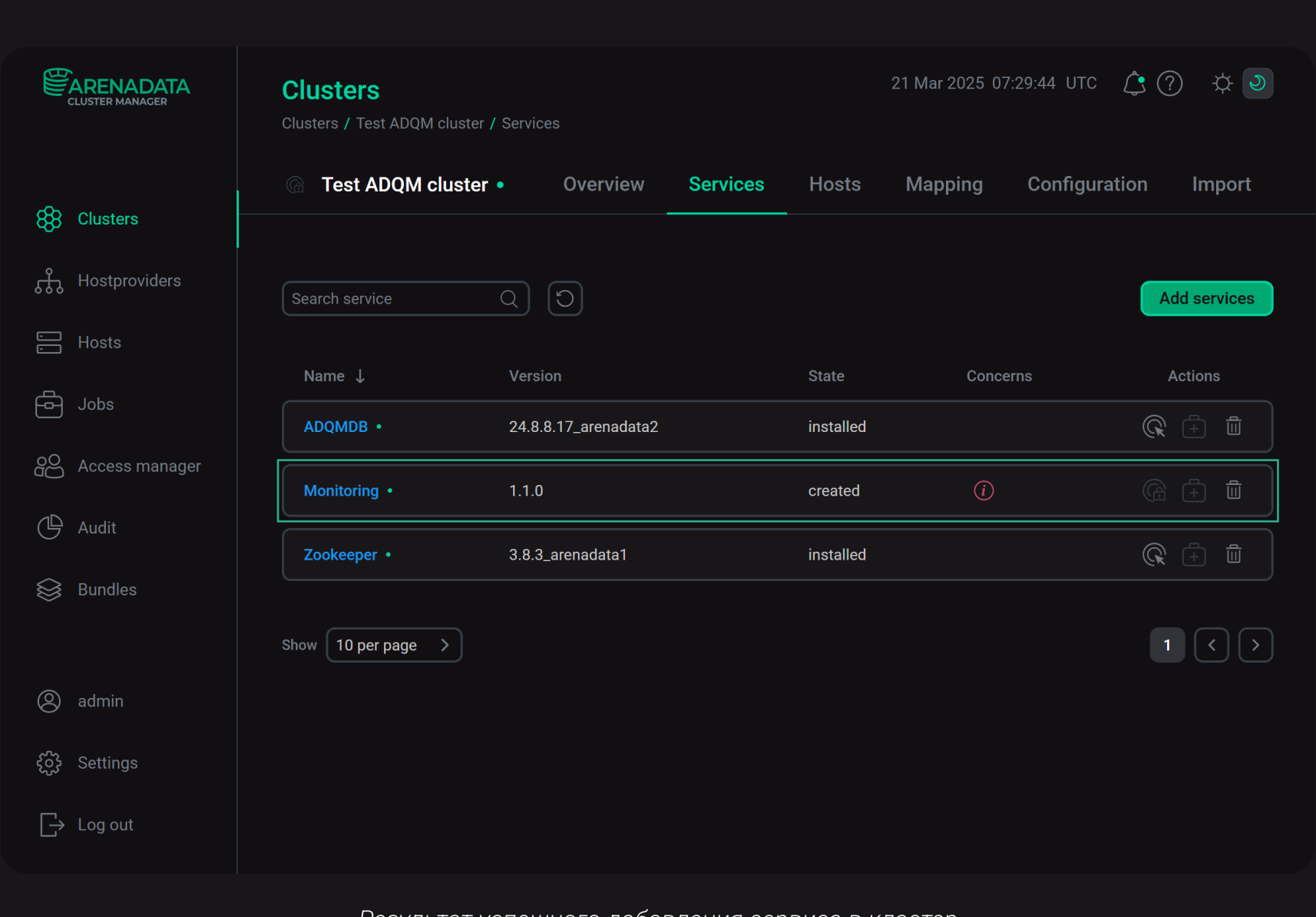


Add services
Search services
Select filtered
Cancel
Add

☐ All services
☐ Chproxy
☐ Clickhousekeeper
☑ Monitoring
☐ [Deprecated] Monitoring Clients

Выбор сервиса

В результате добавленный сервис отобразится на вкладке ***Services***.



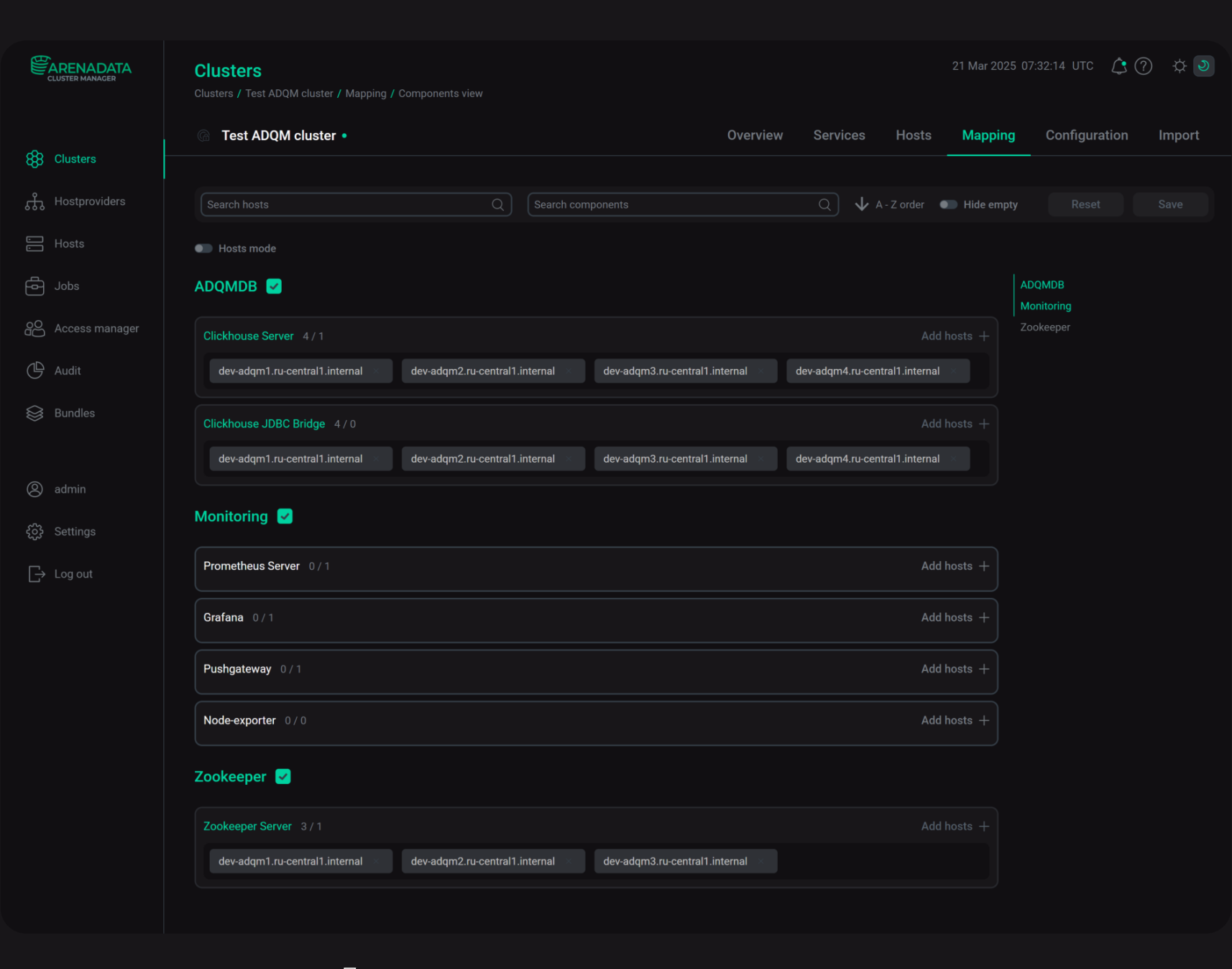
Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 07:29:44 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services

Name	Version	State	Concerns	Actions
ADQMDB	24.8.8.17_arenadata2	installed		
Monitoring	1.1.0	created		
Zookeeper	3.8.3_arenadata1	installed		

Результат успешного добавления сервиса в кластер

Шаг 2. Добавление компонентов

1. На странице кластера откройте вкладку ***Mapping***, чтобы перейти к распределению сервисных компонентов между хостами кластера.



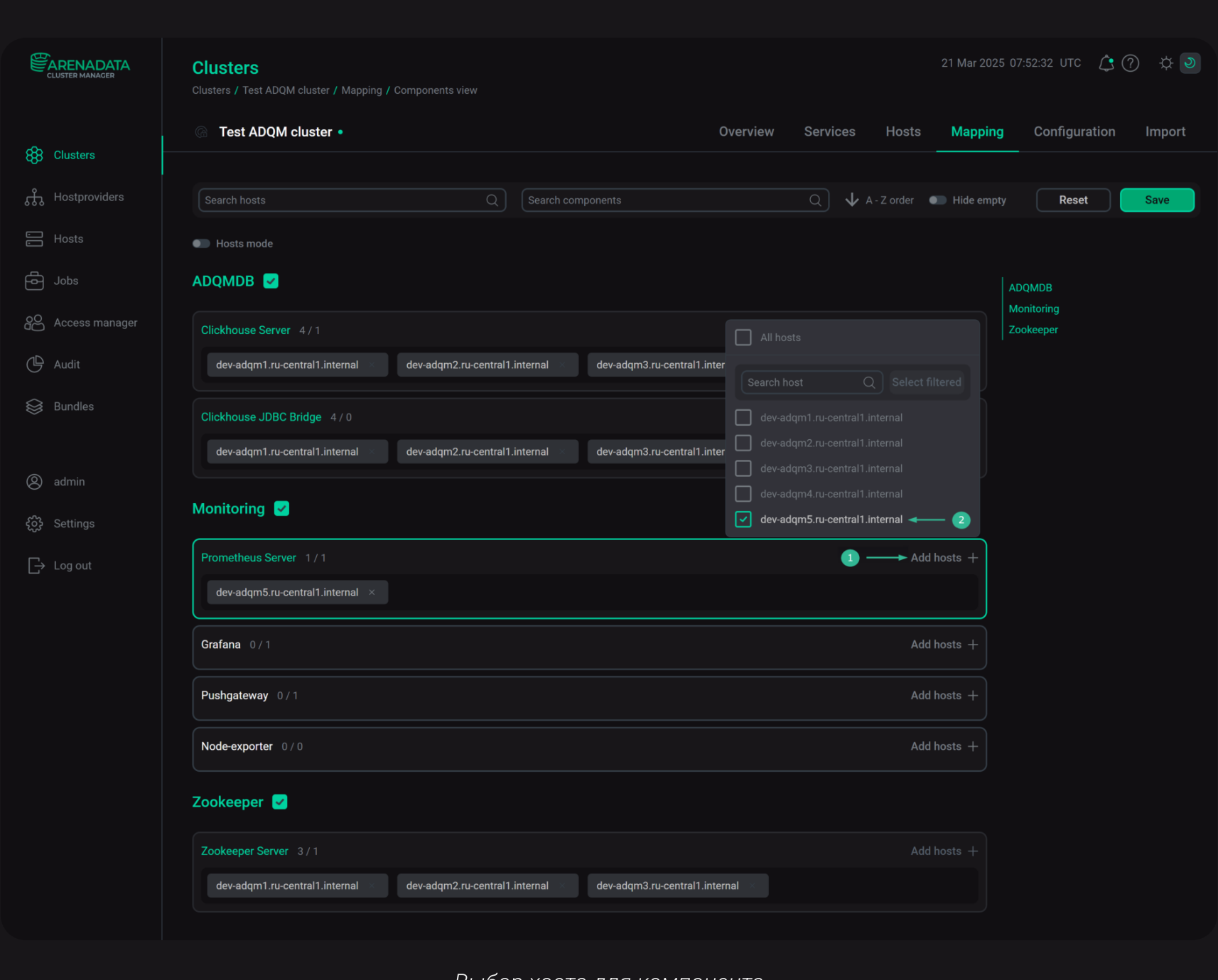
Clusters
Clusters / Test ADQM cluster / Mapping / Components view
Test ADQM cluster
Mapping
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
ADQMDB
Clickhouse Server 4 / 1
Clickhouse JDBC Bridge 4 / 0
Monitoring
Prometheus Server 0 / 1
Grafana 0 / 1
Pushgateway 0 / 1
Node-exporter 0 / 0
Zookeeper
Zookeeper Server 3 / 1
Add hosts

Переход к распределению сервисных компонентов

Компоненты сервиса Monitoring

Компонент	Описание
Prometheus Server	Хранит и позволяет запрашивать все метрики кластера ADQM
Grafana	Позволяет визуализировать метрики ADQM в виде графиков и диаграмм, организованных в дашборды
Pushgateway	Принимает и отдает в Prometheus статические метрики. Используется, чтобы передать в Prometheus структуру кластера ADQM
Node Exporter	Собирает системные метрики и отдает их в Prometheus

2. Назначьте хосты компонентам сервиса Monitoring — нажмите ***Add hosts*** и выберите нужные хосты в открывшемся окне.



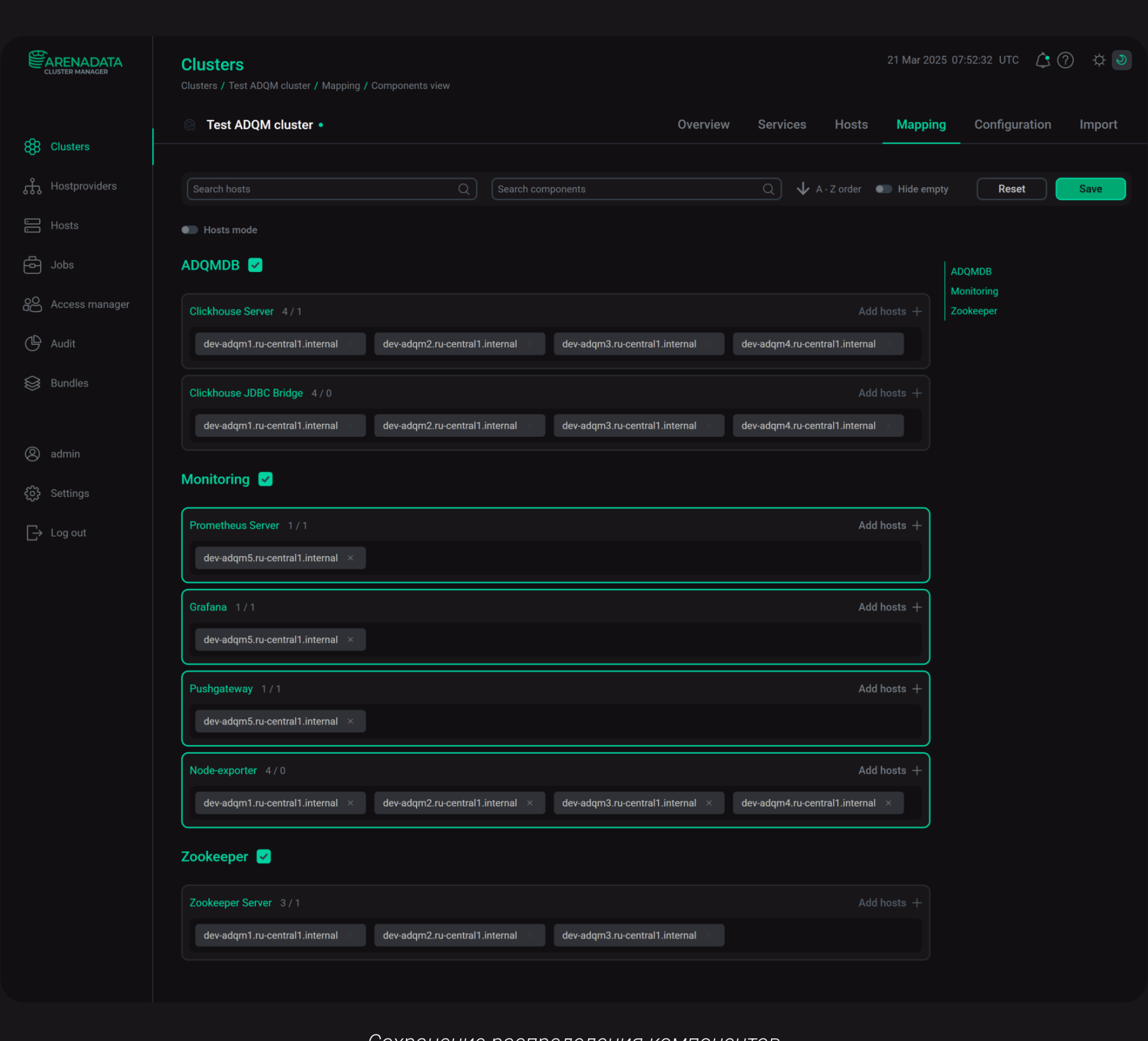
Monitoring
Prometheus Server 1 / 1
1
Add hosts
2
All hosts
Search host
Select filtered
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
dev-adqm5.ru-central1.internal

Выбор хоста для компонента

ВНИМАНИЕ

Компоненты Prometheus Server и Pushgateway не рекомендуется устанавливать на хосты с ADQM — используйте для них отдельные хосты. Иначе при выходе из строя хоста ADQM и/или критически большой нагрузке на него информация о соответствующих проблемах не будет сохранена.

3. После завершения распределения компонентов нажмите ***Save***, чтобы сохранить изменения.

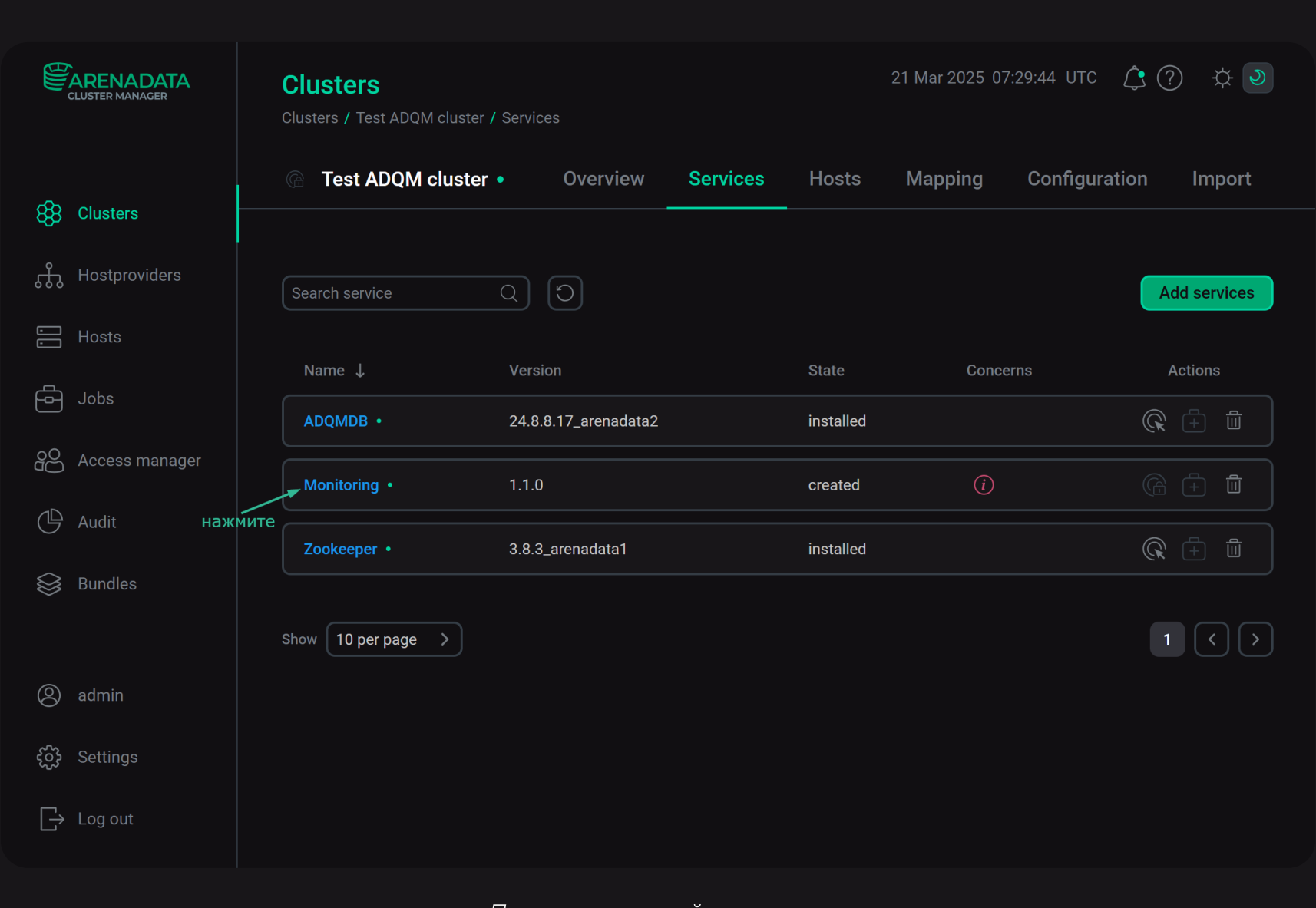


Save
Prometheus Server 1 / 1
Grafana 1 / 1
Pushgateway 1 / 1
Node-exporter 4 / 0
dev-adqm5.ru-central1.internal

Сохранение распределения компонентов

Шаг 3. Настройка сервиса

1. Откройте вкладку ***Services*** на странице кластера и кликните по имени сервиса ***Monitoring*** в столбце ***Name***.

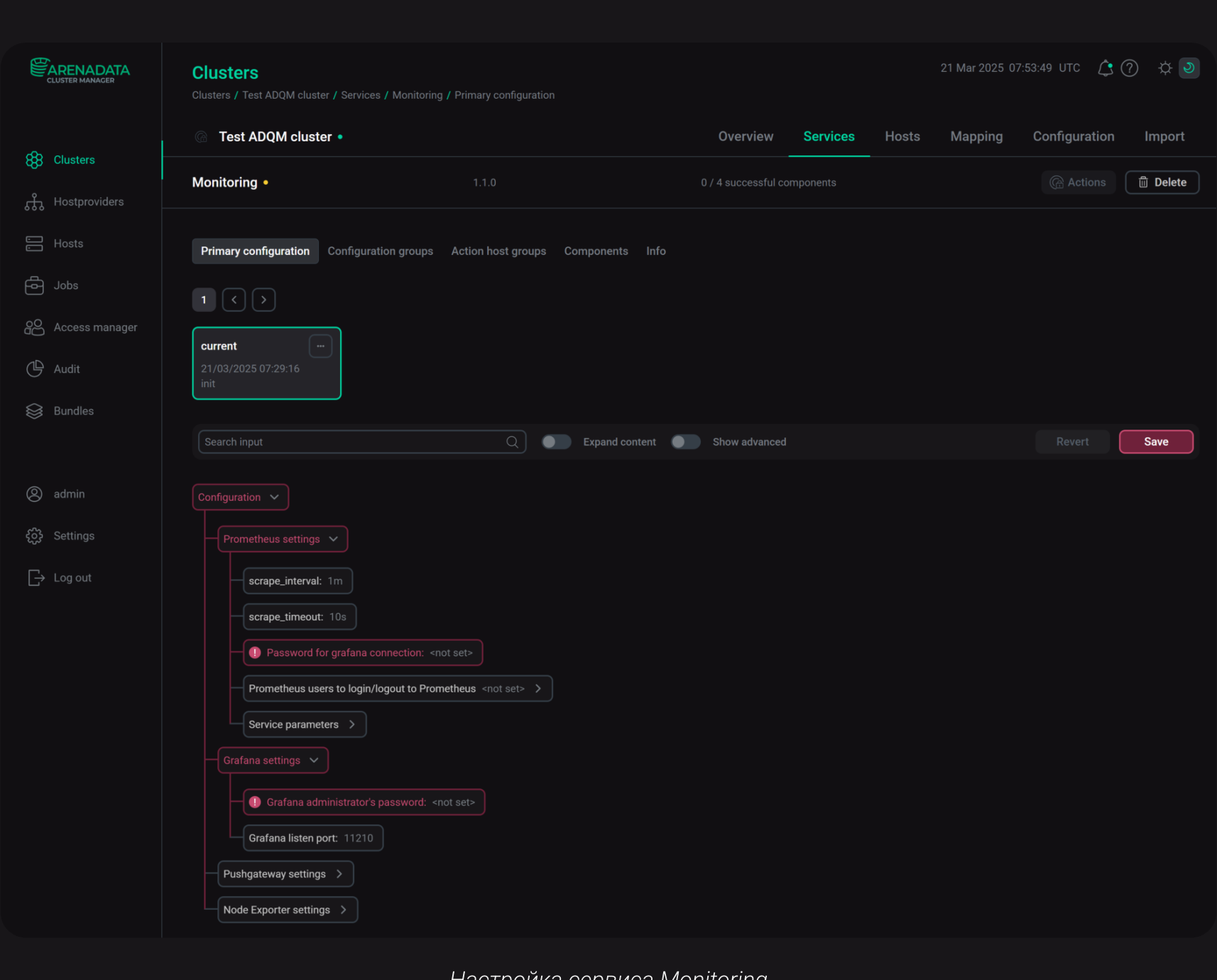


Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 07:29:44 UTC
Test ADQM cluster
Services
нажмите

Name	Version	State	Concerns	Actions
ADQMDB	24.8.8.17_arenadata2	installed		
Monitoring	1.1.0	created		
Zookeeper	3.8.3_arenadata1	installed		

Переход к настройке сервиса

2. В открывшемся окне заполните конфигурационные параметры сервиса — описания параметров приведены в разделе ***Monitoring*** статьи Конфигурационные параметры. Поля, подсвеченные красным, обязательны для заполнения.



Clusters / Test ADQM cluster / Services / Monitoring / Primary configuration
Monitoring
1.1.0
0 / 4 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
current
21/03/2025 07:29:16
init
Search input
Expand content
Show advanced
Revert
Save

Configuration
- Prometheus settings
  - scrape_interval: 1m
  - scrape_timeout: 10s
  - Password for grafana connection: <not set>
  - Prometheus users to login/logout to Prometheus <not set>
  - Service parameters
- Grafana settings
  - Grafana administrator's password: <not set>
  - Grafana listen port: 11210
- Pushgateway settings
- Node Exporter settings

Настройка сервиса Monitoring

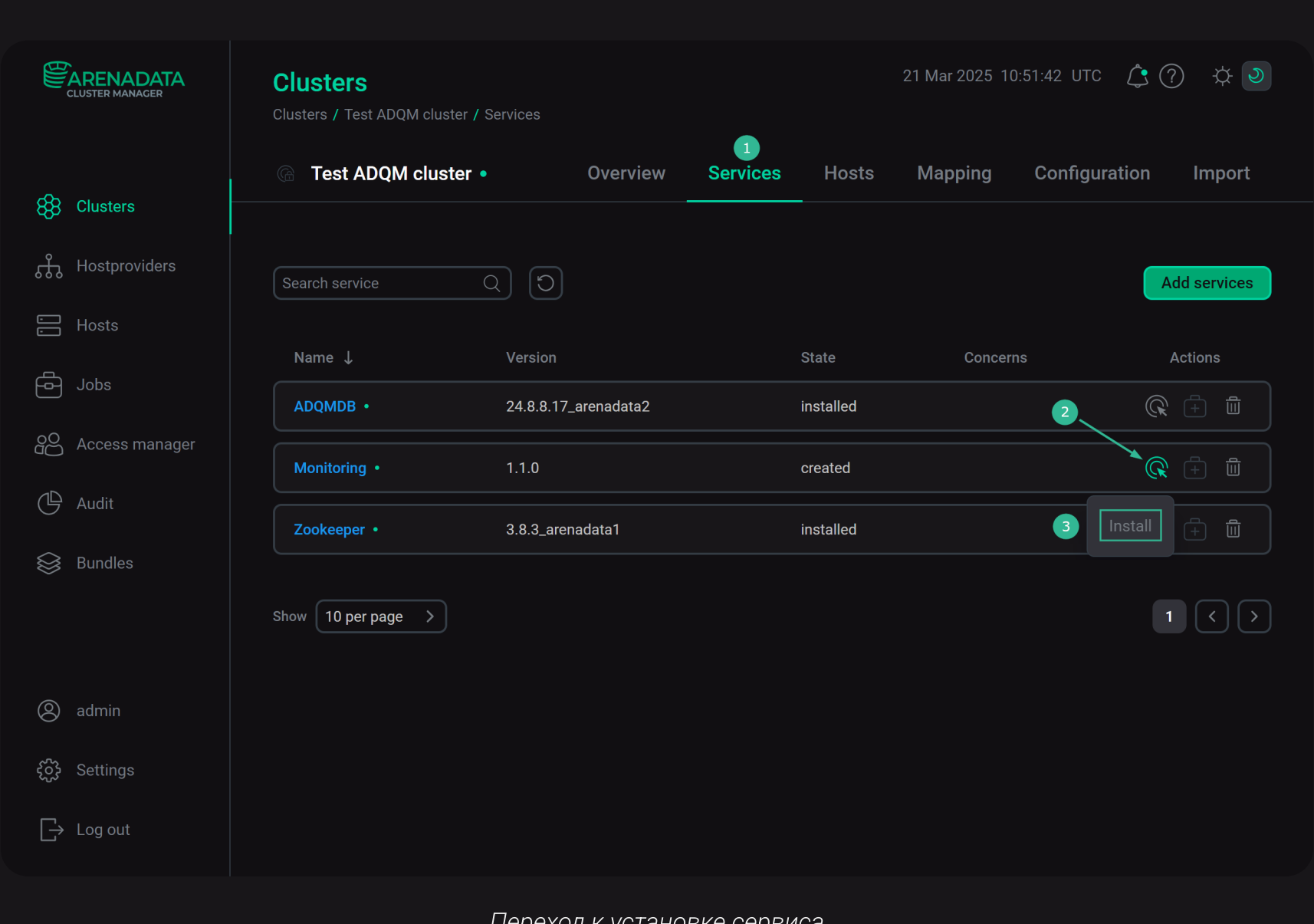
После указания всех необходимых параметров нажмите ***Save***.

ПРИМЕЧАНИЕ

Параметры доступа к метрикам сервисов ADQMDB, Zookeeper, Clickhousekeeper настраиваются в активируемой секции ***Metrics*** конфигурации каждого сервиса. Если не указать эти параметры вручную, сервис Monitoring при установке автоматически включит секцию ***Metrics*** для сервисов ADQM и будет использовать значения параметров по умолчанию.

Шаг 4. Установка сервиса

1. На вкладке ***Services*** в строке сервиса Monitoring кликните по иконке в столбце ***Actions*** и запустите действие ***Install***.

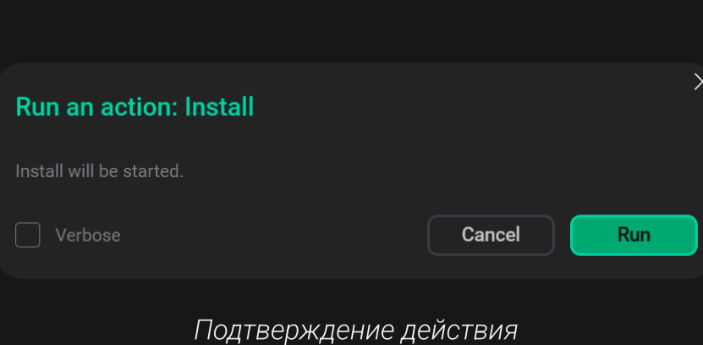


Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 10:51:42 UTC
1
Services
2
3
Install

Name	Version	State	Concerns	Actions
ADQMDB	24.8.8.17_arenadata2	installed		
Monitoring	1.1.0	created		
Zookeeper	3.8.3_arenadata1	installed		

Переход к установке сервиса

2. Подтвердите действие в открывшемся окне.



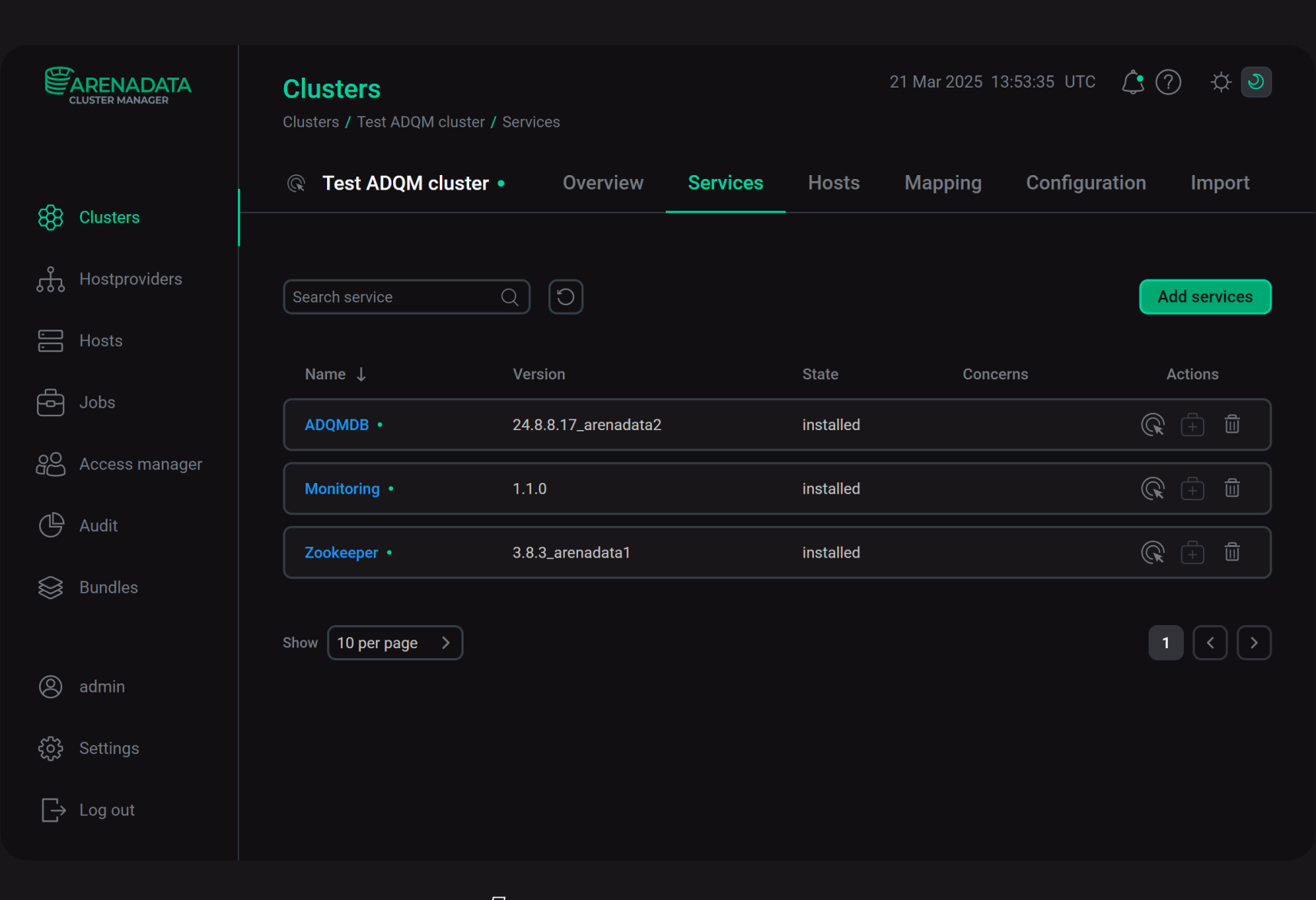
Run an action: Install
Cancel
Run

Install will be started.

☐ Verbose

Подтверждение действия

3. Дождитесь завершения установки и убедитесь, что статус сервиса изменился с `created` на `installed`.

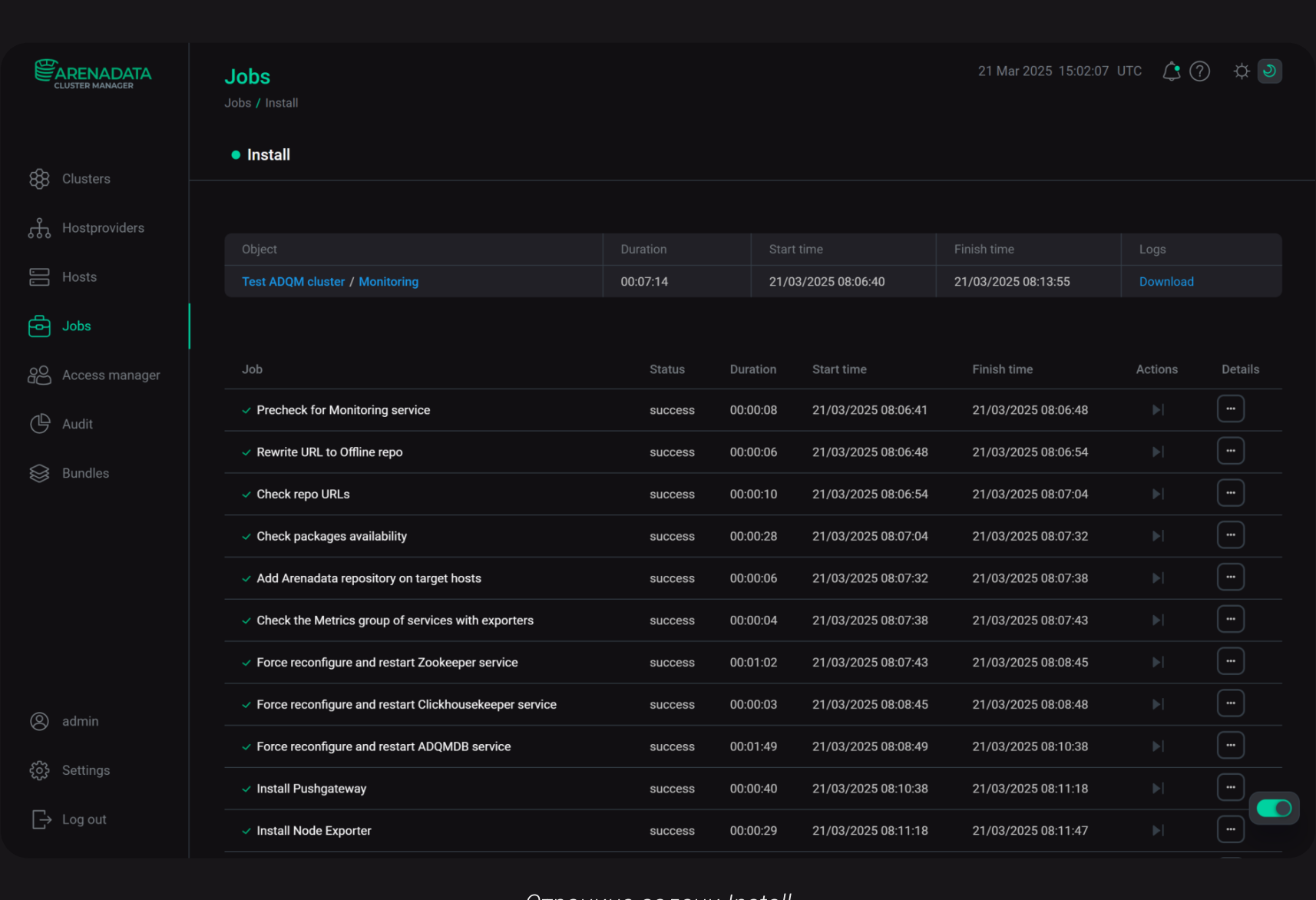


Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 13:53:35 UTC
Services

Name	Version	State	Concerns	Actions
ADQMDB	24.8.8.17_arenadata2	installed		
Monitoring	1.1.0	installed		
Zookeeper	3.8.3_arenadata1	installed		

Процесс установки завершен

Чтобы посмотреть процесс выполнения установки сервиса и проанализировать ошибки в случае их возникновения, выберите пункт ***Jobs*** в левом навигационном меню и кликните по имени задачи ***Install*** в списке задач ***Jobs***.



Jobs
Jobs / Install
21 Mar 2025 15:02:07 UTC
Install

Object	Duration	Start time	Finish time	Logs
Test ADQM cluster / Monitoring	00:07:14	21/03/2025 08:06:40	21/03/2025 08:13:55	Download

Job	Status	Duration	Start time	Finish time
Precheck for Monitoring service	success	00:00:08	21/03/2025 08:06:41	21/03/2025 08:06:48
Rewrite URL to Offline repo	success	00:00:06	21/03/2025 08:06:48	21/03/2025 08:06:54
Check repo URLs	success	00:00:10	21/03/2025 08:06:54	21/03/2025 08:07:04
Check packages availability	success	00:00:28	21/03/2025 08:07:04	21/03/2025 08:07:32
Add Arenadata repository on target hosts	success	00:00:06	21/03/2025 08:07:32	21/03/2025 08:07:38
Check the Metrics group of services with exporters	success	00:00:04	21/03/2025 08:07:38	21/03/2025 08:07:43
Force reconfigure and restart Zookeeper service	success	00:01:02	21/03/2025 08:07:43	21/03/2025 08:08:45
Force reconfigure and restart Clickhousekeeper service	success	00:00:03	21/03/2025 08:08:45	21/03/2025 08:08:48
Force reconfigure and restart ADQMDB service	success	00:01:49	21/03/2025 08:08:49	21/03/2025 08:10:38
Install Pushgateway	success	00:00:40	21/03/2025 08:10:38	21/03/2025 08:11:18
Install Node Exporter	success	00:00:29	21/03/2025 08:11:18	21/03/2025 08:11:47

Страница задачи Install

Шаг 5. Проверка результатов

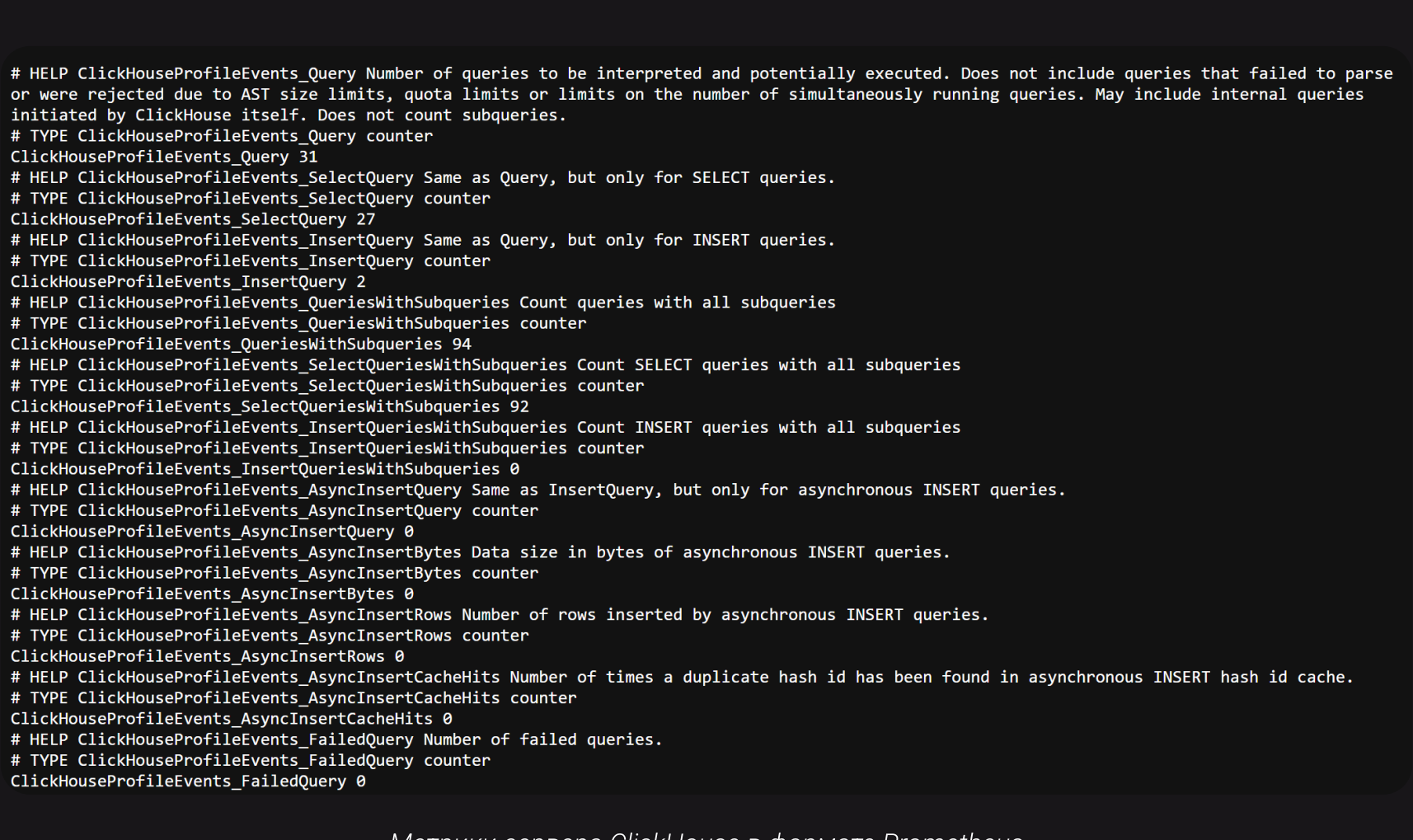
Сервис Monitoring запускается автоматически после установки. По завершении установки необходимо убедиться, что с хостов кластера поступают системные метрики и метрики сервисов ADQM — см. Метрики мониторинга кластера ADQM. Чтобы это проверить, можно в браузере посмотреть метрики в формате Prometheus, а также использовать веб-интерфейсы Prometheus и Grafana.

Данные в формате Prometheus

1. В адресную строку браузера введите адрес хоста кластера ADQM, а также укажите порт и конечную точку доступа (endpoint) к сервисным или системным метрикам, которые определяются параметрами:
   - в секции ***Metrics*** конфигурации сервисов ADQMDB, Zookeeper, Clickhousekeeper — настройки доступа к метрикам мониторинга этих сервисов;
   - в секции ***Node Exporter settings*** на странице сервиса Monitoring — настройки доступа к системным метрикам хоста кластера ADQM.

   Пример адреса для просмотра метрик сервера ClickHouse на хосте с IP `10.92.40.25`: *http://10.92.40.25:9363/metrics*.

2. На открывшейся странице будут показаны метрики мониторинга с указанного хоста кластера в формате Prometheus.

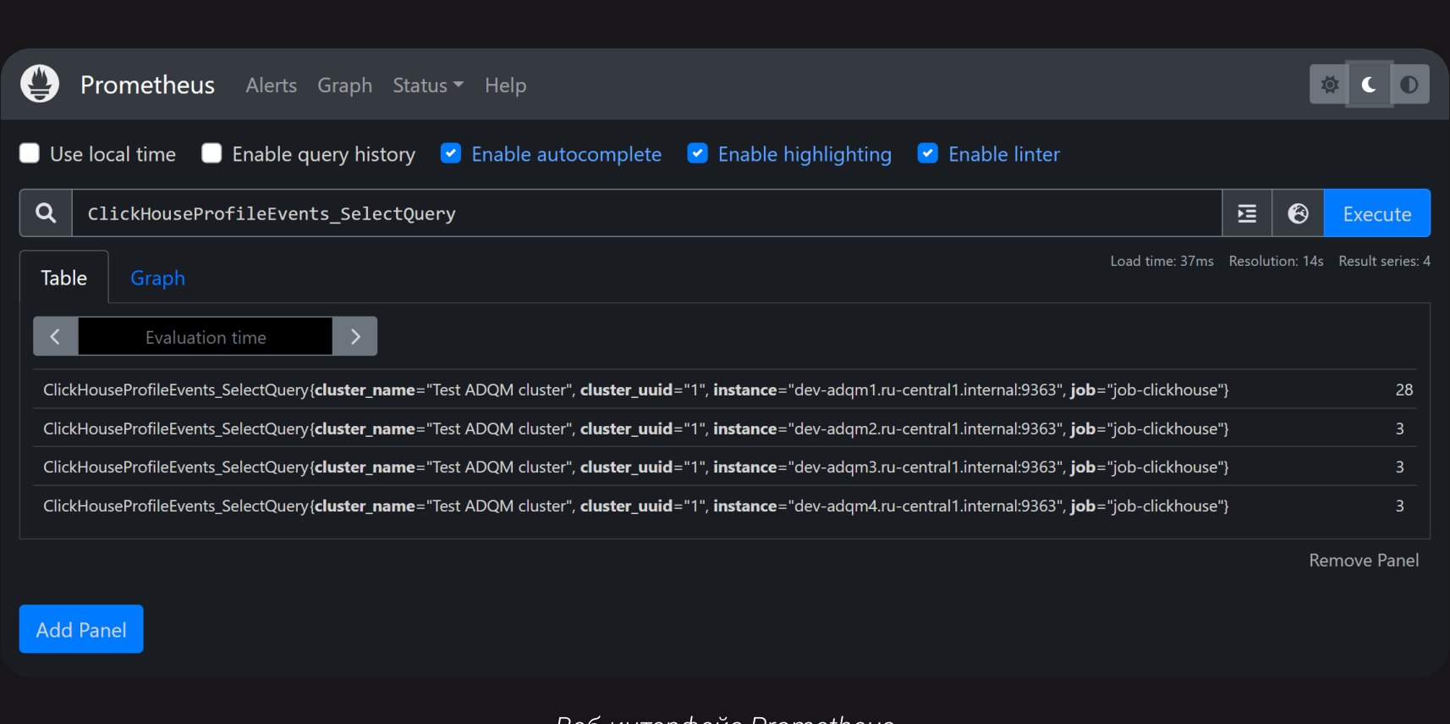


```
# HELP ClickHouseProfileEvents_Query Number of queries to be interpreted and potentially executed. Does not include queries that failed to parse
or were rejected due to AST size limits, quota limits or limits on the number of simultaneously running queries. May include internal queries
initiated by ClickHouse itself. Does not count subqueries.
# TYPE ClickHouseProfileEvents_Query counter
ClickHouseProfileEvents_Query 31
# HELP ClickHouseProfileEvents_SelectQuery Same as Query, but only for SELECT queries.
# TYPE ClickHouseProfileEvents_SelectQuery counter
ClickHouseProfileEvents_SelectQuery 27
# HELP ClickHouseProfileEvents_InsertQuery Same as Query, but only for INSERT queries.
# TYPE ClickHouseProfileEvents_InsertQuery counter
ClickHouseProfileEvents_InsertQuery 2
# HELP ClickHouseProfileEvents_QueriesWithSubqueries Count queries with all subqueries
# TYPE ClickHouseProfileEvents_QueriesWithSubqueries counter
ClickHouseProfileEvents_QueriesWithSubqueries 94
# HELP ClickHouseProfileEvents_SelectQueriesWithSubqueries Count SELECT queries with all subqueries
# TYPE ClickHouseProfileEvents_SelectQueriesWithSubqueries counter
ClickHouseProfileEvents_SelectQueriesWithSubqueries 92
# HELP ClickHouseProfileEvents_InsertQueriesWithSubqueries Count INSERT queries with all subqueries
# TYPE ClickHouseProfileEvents_InsertQueriesWithSubqueries counter
ClickHouseProfileEvents_InsertQueriesWithSubqueries 0
# HELP ClickHouseProfileEvents_AsyncInsertQuery Same as InsertQuery, but only for asynchronous INSERT queries.
# TYPE ClickHouseProfileEvents_AsyncInsertQuery counter
ClickHouseProfileEvents_AsyncInsertQuery 0
# HELP ClickHouseProfileEvents_AsyncInsertBytes Data size in bytes of asynchronous INSERT queries.
# TYPE ClickHouseProfileEvents_AsyncInsertBytes counter
ClickHouseProfileEvents_AsyncInsertBytes 0
# HELP ClickHouseProfileEvents_AsyncInsertRows Number of rows inserted by asynchronous INSERT queries.
# TYPE ClickHouseProfileEvents_AsyncInsertRows counter
ClickHouseProfileEvents_AsyncInsertRows 0
# HELP ClickHouseProfileEvents_AsyncInsertCacheHits Number of times a duplicate hash id has been found in asynchronous INSERT hash id cache.
# TYPE ClickHouseProfileEvents_AsyncInsertCacheHits counter
ClickHouseProfileEvents_AsyncInsertCacheHits 0
# HELP ClickHouseProfileEvents_FailedQuery Number of failed queries.
# TYPE ClickHouseProfileEvents_FailedQuery counter
ClickHouseProfileEvents_FailedQuery 0
```

Метрики сервера ClickHouse в формате Prometheus

Веб-интерфейс Prometheus

1. В адресной строке браузера введите IP-адрес хоста, где установлен компонент Prometheus Server сервиса Monitoring. Укажите номер порта, который определяется параметром ***web.listen-address*** в секции ***Prometheus settings*** в группе ***Service parameters*** конфигурации сервиса Monitoring (по умолчанию, `11200`). Пример адреса: *http://10.92.40.107:11200*. Для входа в интерфейс Prometheus используйте имя пользователя и пароль, которые также указываются в секции ***Prometheus settings*** — настройка ***Prometheus users to login/logout to Prometheus***.
2. В поле ***Expression*** можно ввести название метрики и нажать ***Execute*** — в интерфейсе будут показаны значения этой метрики на всех хостах кластера ADQM.



Prometheus
Alerts
Graph
Status
Help
Execute
Table
Graph
Evaluation time
Remove Panel
Add Panel

☐ Use local time ☐ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

`ClickHouseProfileEvents_SelectQuery`

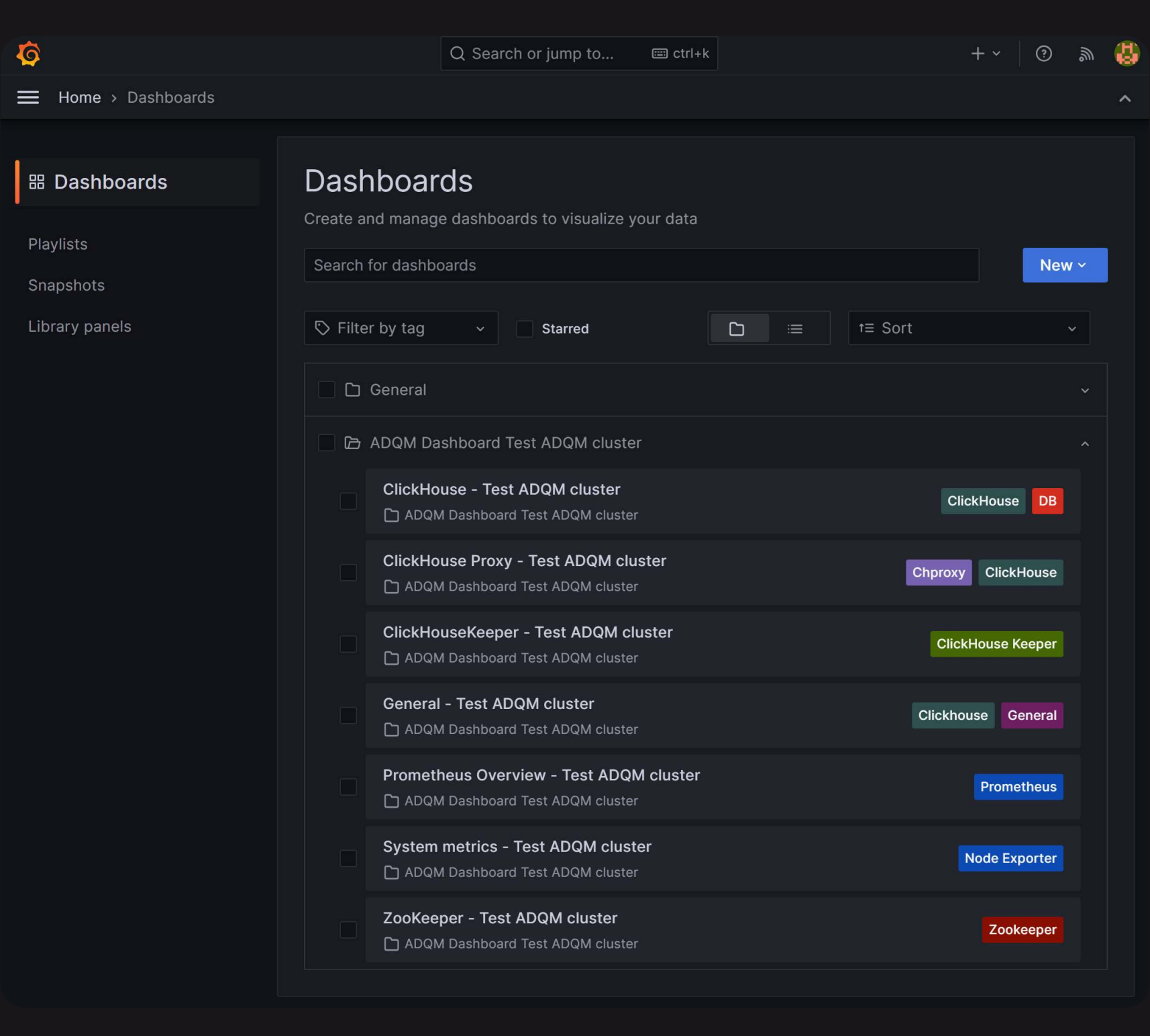
Load time: 37ms Resolution: 14s Result series: 4

```
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm1.ru-central1.internal:9363", job="job-clickhouse"}   28
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm2.ru-central1.internal:9363", job="job-clickhouse"}   3
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm3.ru-central1.internal:9363", job="job-clickhouse"}   3
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm4.ru-central1.internal:9363", job="job-clickhouse"}   3
```

Веб-интерфейс Prometheus

Веб-интерфейс Grafana

1. В адресной строке браузера введите адрес хоста, на котором развернут компонент Grafana сервиса Monitoring. В качестве номера порта укажите значение параметра ***Grafana listen port*** из раздела ***Grafana settings*** в настройках конфигурации сервиса мониторинга (по умолчанию `11210`). Например, *http://10.92.40.107:11210*. Для входа используйте `admin` в качестве имени пользователя и значение параметра ***Grafana administrator's password*** (находится также в разделе настроек ***Grafana settings***) в качестве пароля.
2. В открывшемся окне выберите пункт меню ***Home → Dashboards*** и разверните секцию ***ADQM Dashboard <ADQM_cluster_name>***. В этой секции можно выбрать один из доступных дашбордов для просмотра сервисных или системных метрик, поступающих из вашего кластера ADQM.



Search or jump to...
ctrl+k
Home › Dashboards
Dashboards
Playlists
Snapshots
Library panels
Create and manage dashboards to visualize your data
Search for dashboards
New
Filter by tag
Starred
Sort
General
ADQM Dashboard Test ADQM cluster
ClickHouse - Test ADQM cluster
ClickHouse
DB
ClickHouse Proxy - Test ADQM cluster
Chproxy
ClickHouse
ClickHouseKeeper - Test ADQM cluster
ClickHouse Keeper
General - Test ADQM cluster
Clickhouse
General
Prometheus Overview - Test ADQM cluster
Prometheus
System metrics - Test ADQM cluster
Node Exporter
ZooKeeper - Test ADQM cluster
Zookeeper

Переход к метрикам кластера ADQM в Grafana

Offline-установка

Назад к содержанию

Этот раздел посвящен **offline**-установке ADQM. Данный способ установки подходит для хостов без доступа к Интернету.

Для offline-установки необходимо установить и настроить Arenadata Enterprise Tools (ET). ET — это решение, предоставляющее инфраструктуру для развертывания продуктов Аренадата в среде с ограниченным доступом в интернет. Решение является частью платформы данных Аренадата и предоставляет репозитории, необходимые для установки и обновления компонентов платформы.

Процесс offline-установки через ADCM включает следующие шаги.

ПРИМЕЧАНИЕ

Эта функция доступна только в Enterprise-версии ADQM.

AD.CM

Шаг 1.

Установка ADCM

AD.CM

Шаг 2.

Подготовка хостов

AD.CM

Шаг 3.

Установка кластера ET

AD.CM

ARENADATA
QUICKMARTS

Шаг 4.

Установка кластера ADQM

AD.CM

Шаг 5.

Установка мониторинга

AD.CM

ARENADATA
ADQM CONTROL

Шаг 6.

Установка ADQM Control

Установка ADCM

Дарья Барышева

Содержание

Назад к содержанию

Шаг 1. Предварительные требования
  - Требования к software и hardware
  - Установка Docker
  - Отключение SELinux
  - Остановка firewalld
  - Настройка доступа к Docker CLI без прав root

Шаг 2. Установка ADCM

Шаг 3. Запуск ADCM

Шаг 4. Проверка установки
  - Проверка статуса Docker-контейнера
  - Проверка доступности порта 8000
  - Проверка web-интерфейса ADCM
  - Установка URL ADCM

Шаг 5. Обновление ADCM

Для развертывания ADCM требуется отдельный хост. Перед началом установки рекомендуем ознакомиться с документацией ADCM.

Этапы установки описаны ниже.

Шаг 1. Предварительные требования

Требования к software и hardware

ADCM поставляется в виде Docker-образа. Для его установки необходимы:

- Одна из приведенных ниже операционных систем:
  - Ubuntu 22.04
  - CentOS 7
  - RHEL 7
  - Alt Linux SP 8
  - Alt Linux SP 10.2
  - Astra Linux SE 1.7 Орел
  - Astra Linux 1.7.6 Воронеж
  - РЕД ОС 7.3 Сертифицированная редакция
- Пользователь с правами `sudo`.
- Доступ к официальным репозиториям CentOS Extras и CentOS Base. По умолчанию, эти репозитории доступны в операционной системе. Однако, если они были отключены, потребуется их предварительное включение (например, путем редактирования конфигурационного файла в директории */etc/yum.repos.d/*).
- Менеджер программных пакетов YUM/APT.
- Хост, отвечающий следующим требованиям:

ВАЖНО

Системные требования, приведенные ниже, являются **минимальными**. Целевой сайзинг необходимо рассчитывать исходя из требований вашей организации.

  - **CPU**: 4 ядра CPU;
  - **RAM**: 16 ГБ;
  - **HDD**: 50 ГБ.

ВАЖНО

Начиная с версии ADCM 2.7.0 для установки ADPG без подключения к Интернету и использования его в качестве базы данных ADCM требуется любой Container Registry Client с загруженными образами ADPG, ADCM и ADCM Installer. Установка выполняется с помощью ADCM Installer с указанием URL локального хранилища образов ADCM и ADPG.

Чтобы пользователь обладал необходимыми правами для установки ADCM, выполните следующее:

1. Предоставьте пользователю привилегии `root`:

CentOS 7/RHEL 7/РЕД ОС 7.3/Alt Linux SP 8

```
$ sudo usermod -aG wheel <user>
```

где `<user>` — имя пользователя.

Ubuntu 22.04/Astra Linux SE 1.7 Орел

```
$ sudo usermod -aG sudo <user>
```

где `<user>` — имя пользователя.

2. Убедитесь, что в файле */etc/sudoers* есть запись, указанная ниже в зависимости от используемой операционной системы, с помощью следующей команды:

```
$ sudo vi /etc/sudoers
```

Чтобы вернуться в терминал, нажмите `Esc`, введите `:q!` и нажмите `Enter`.

CentOS 7/RHEL 7/РЕД ОС 7.3

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Ubuntu 22.04/Astra Linux SE 1.7 Орел

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Alt Linux SP 8

```
## Uncomment to allow members of group wheel to execute any command
WHEEL_USERS ALL=(ALL:ALL) ALL
```

Установка Docker

ВНИМАНИЕ

Установка Docker с официального сайта не рекомендуется. Вместо этого используйте официальные репозитории дистрибутива Linux. Репозитории, предоставляемые разработчиками операционных систем, как правило, более стабильны и безопасны.

Ниже приведены шаги, необходимые для установки Docker, на примере нескольких поддерживаемых в ADCM операционных систем. Для других версий операционных систем, отличных от указанных во вкладках, шаги могут отличаться.

CentOS 7/RHEL 7

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo yum update -y
```

2. Установите пакеты, необходимые для Docker:

```
$ sudo yum install -y yum-utils docker device-mapper-persistent-data lvm2
```

где:

- `yum-utils` — пакет, который предоставляет утилиту `yum-config-manager`;
- `device-mapper-persistent-data` и `lvm2` — пакеты, необходимые драйверу хранилища devicemapper.

3. Запустите Docker:

```
$ sudo systemctl start docker
```

4. Включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker
```

Ubuntu 22.04

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo apt-get update
```

2. Установите пакеты, необходимые для Docker:

```
$ sudo apt-get install ca-certificates curl
```

где:

- `ca-certificates` — пакет корневых сертификатов;
- `curl` — пакет для работы с веб-ресурсами.

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

3. Добавьте GPG-ключ для Docker-репозитория:

```
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

4. Добавьте Docker-репозиторий:

```
$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "jammy") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Обновите все программные пакеты, в том числе пакеты для добавленного Docker-репозитория:

```
$ sudo apt-get update
```

6. Установите Docker Engine:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

7. Запустите Docker:

```
$ sudo systemctl start docker
```

8. Включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker
```

Alt Linux SP 8

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo apt-get update
```

2. Установите Docker Engine:

```
$ sudo apt-get install docker-engine
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

3. Запустите и включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable --now docker
```

Astra Linux SE 1.7 Орел

1. Обновите все программные пакеты, установленные в системе:

```
$ sudo apt-get update
```

2. Установите пакеты, необходимые для Docker:

```
$ sudo apt-get install ca-certificates curl net-tools
```

где:

- `ca-certificates` — пакет корневых сертификатов;
- `curl` — пакет для работы с веб-ресурсами;
- `net-tools` — пакет для управления сетевой подсистемой ядра Linux, в частности для запуска команды `netstat`.

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

3. Добавьте GPG-ключ для Docker-репозитория:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

4. Добавьте Docker-репозиторий:

```
$ echo "deb [arch=amd64] https://download.docker.com/linux/debian stretch stable" | sudo tee -a /etc/apt/sources.list
```

5. Обновите все программные пакеты, в том числе пакеты для добавленного Docker-репозитория:

```
$ sudo apt-get update
```

6. Установите Docker Engine:

```
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

7. Запустите Docker:

```
$ sudo systemctl start docker
```

8. Включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker
```

РЕД ОС 7.3

1. Установите Docker Engine:

```
$ sudo dnf install docker-ce docker-ce-cli
```

При выполнении команды подтвердите увеличение объема занятого дискового пространства.

2. Запустите и включите Docker в качестве системного сервиса:

```
$ sudo systemctl enable docker --now
```

Отключение SELinux

ПРИМЕЧАНИЕ

Отключение системы безопасности SELinux необходимо только в случае использования операционной системы CentOS 7 или RHEL 7.

Для долговременного отключения SELinux следует прописать `SELINUX=disabled` в конфигурационном файле */etc/selinux/config* и перезагрузить систему.

Файл может быть изменен с помощью редактора `vi` (или `vim`):

```
$ sudo vi /etc/selinux/config
```

Содержимое обновленного файла:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

Кроме этого, доступно временное отключение SELinux с помощью следующей команды:

```
$ sudo setenforce 0
```

Тем не менее рекомендуется отключить SELinux на постоянной основе, чтобы предотвратить ее перезапуск при каждой перезагрузке системы.

ВНИМАНИЕ

Не забудьте перезагрузить хост после отключения SELinux.

Остановка firewalld

Если на хосте используется сервис firewalld, требуется его остановка.

Перед остановкой firewalld убедитесь, что он установлен, запущен и включен, выполнив команду:

```
$ sudo systemctl status firewalld
```

Пример результата выполнения команды:

```
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-12-02 10:02:57 UTC; 3min 25s ago
     Docs: man:firewalld(1)
 Main PID: 3023 (firewalld)
    Tasks: 2
   Memory: 22.8M
   CGroup: /system.slice/firewalld.service
           └─3023 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid
```

Остановите firewalld с помощью команды:

```
$ sudo systemctl stop firewalld
```

Альтернативный вариант — отключить применение правил к сетевым пакетам при помощи команды `disable`:

```
$ sudo systemctl disable firewalld
```

Настройка доступа к Docker CLI без прав root

Для настройки доступа к Docker CLI без использования прав `root` необходимо создать группу пользователей `docker` (если она еще не существует) и добавить текущего пользователя в эту группу с определенными правами:

```
$ sudo groupadd docker
$ sudo usermod -a -G docker $USER
```

После запуска команд требуется перезайти в систему под текущим пользователем.

Шаг 2. Установка ADCM

Offline-установка ADCM включает следующие шаги:

1. Запросите у команды поддержки Arenadata архив с расширением *sh.xz*, который содержит специальный установочный shell-скрипт.
2. Создайте директорию на хосте, где должен быть развернут ADCM, и разместите в ней полученный архив.
3. Распакуйте архив в созданной директории любым из доступных способов.

ПРИМЕЧАНИЕ

Выполнение команды может занять некоторое время. Пожалуйста, дождитесь ее завершения.

4. Загрузите Docker-образ в локальное хранилище с помощью полученного скрипта:

```
$ sudo bash <script_name> unpack_adcm
```

где `<script_name>` — полное имя скрипта с расширением (например, *adcm_2.5.0.sh*).

Пример результата успешной обработки команды:

```
Loaded image: hub.arenadata.io/adcm/adcm:2.5.0
```

5. Создайте новый Docker-контейнер на базе загруженного Docker-образа:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

Шаг 3. Запуск ADCM

Поскольку ADCM представляет собой контейнер под управлением Docker, за его запуск и установку отвечают стандартные Docker-команды:

- **Запуск ADCM**.

```
$ sudo docker start adcm
```

- **Остановка ADCM**.

```
$ sudo docker stop adcm
```

- **Автозапуск ADCM**. Чтобы настроить автоматический запуск Docker-контейнера в случае непредвиденных ошибок, используйте следующую команду.

```
$ sudo docker update --restart=on-failure adcm
```

ПРИМЕЧАНИЕ

- Дополнительная информация по политикам перезапуска Docker-контейнеров (restart policies) может быть получена из документации Docker.
- Если для текущего пользователя настроен доступ к Docker CLI без использования прав `root`, слово `sudo` в предыдущих командах необязательно.

Шаг 4. Проверка установки

Проверка статуса Docker-контейнера

Чтобы убедиться в успешной установке Docker-контейнера `adcm`, выполните следующую команду:

```
$ sudo docker container ls
```

Пример результата выполнения команды приведен ниже. Убедитесь, что в столбце `STATUS` выводится значение `Up`:

```
CONTAINER ID        IMAGE                                COMMAND             CREATED             STATUS              PORTS
NAMES
74b5628146e5        hub.arenadata.io/adcm/adcm:2.0   "/etc/startup.sh"   4 days ago          Up 21 minutes       0.0.0.0:8000-
>8000/tcp   adcm
```

Проверка доступности порта 8000

После установки и запуска ADCM его web-интерфейс должен стать доступным на `8000` порте развернутого контейнера.

Для операционной системы Ubuntu 22.04 предварительно установите пакет для запуска команды `netstat`:

```
$ sudo apt-get install net-tools
```

Для проверки доступности порта выполните команды:

- `netstat` — для вывода информации о состоянии порта.

```
$ sudo netstat -ntpl | grep 8000
```

Пример результата выполнения команды:

```
tcp6       0      0 :::8000                 :::*                    LISTEN
1514/docker-proxy-c
```

- `curl` — для проверки URL-соединения.

```
$ curl http://localhost:8000
```

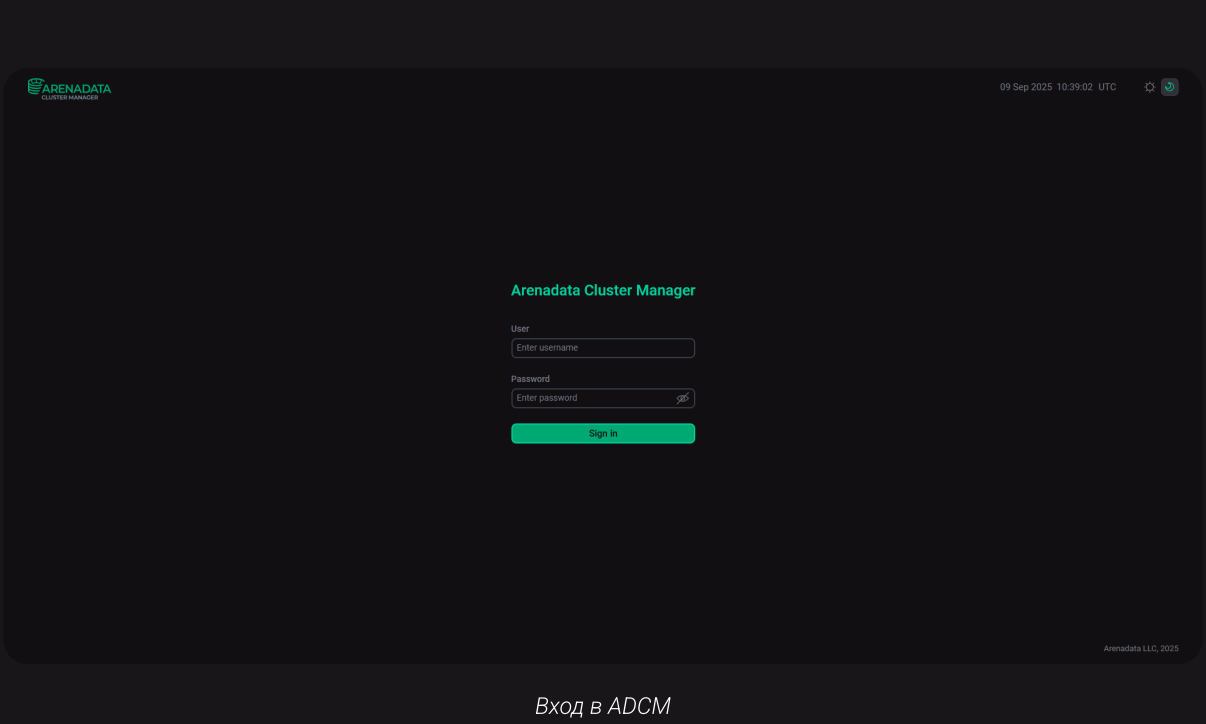
Пример результата выполнения команды:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Arenadata Cluster Manager</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/favicon.ico">
<link rel="stylesheet" href="styles.10db6328264e0907c52f.css"></head>
<body>
  <app-root></app-root>
<script src="runtime-es2015.7eb406ed18bf0258cd35.js" type="module"></script><script src="runtime-es5.7eb406ed18bf0258cd35.js"
nomodule defer></script><script src="polyfills-es5.2e224d70daec4412d3c2.js" nomodule defer></script><script src="polyfills-es2015.
37e0553ac06970d6a5b5.js" type="module"></script><script src="main-es2015.39851da0ebf9ed6fec45.js" type="module"></script><script
src="main-es5.39851da0ebf9ed6fec45.js" nomodule defer></script></body>
</html>
```

Проверка web-интерфейса ADCM

При установке ADCM автоматически создается пользователь с административными правами. Чтобы зайти в ADCM, используйте следующие учетные данные:

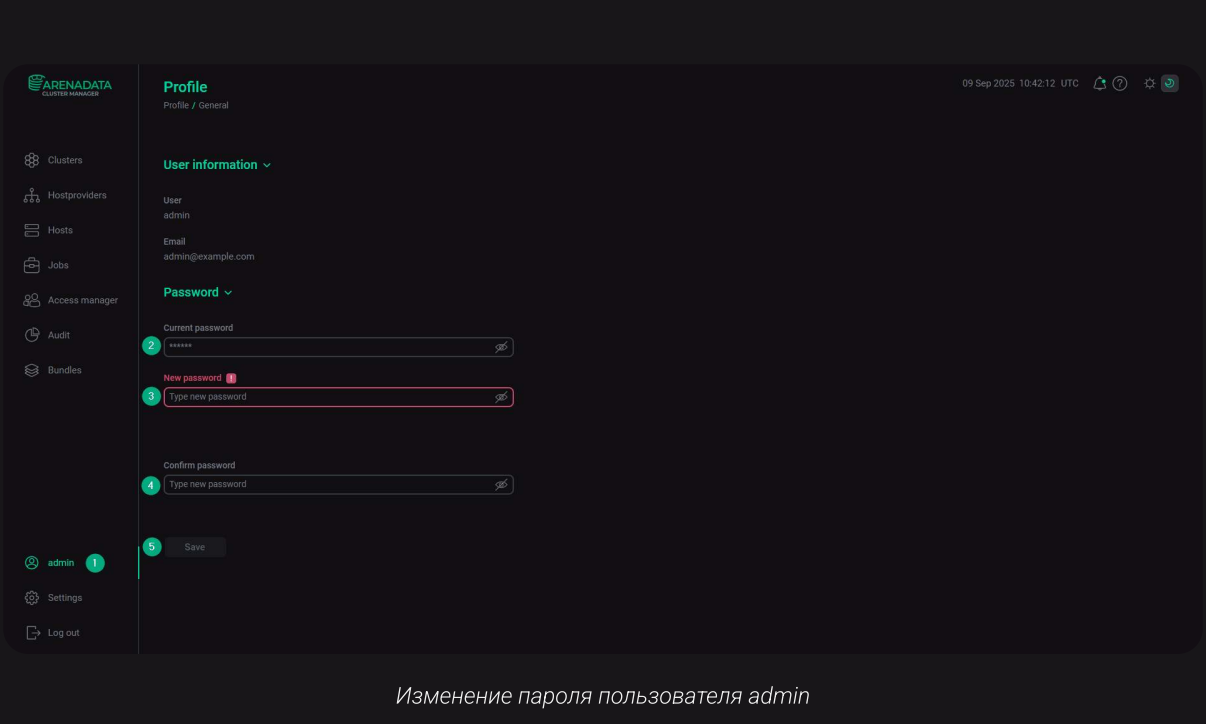
- ***User***: admin
- ***Password***: admin



Вход в ADCM

Для изменения пароля требуется выполнить следующие действия:

1. Откройте профиль текущего пользователя в левом навигационном меню.
2. Введите текущий пароль в поле ***Current password***.
3. Укажите новый пароль в поле ***New password***.
4. Подтвердите новый пароль в поле ***Confirm password***.
5. Нажмите ***Save***.

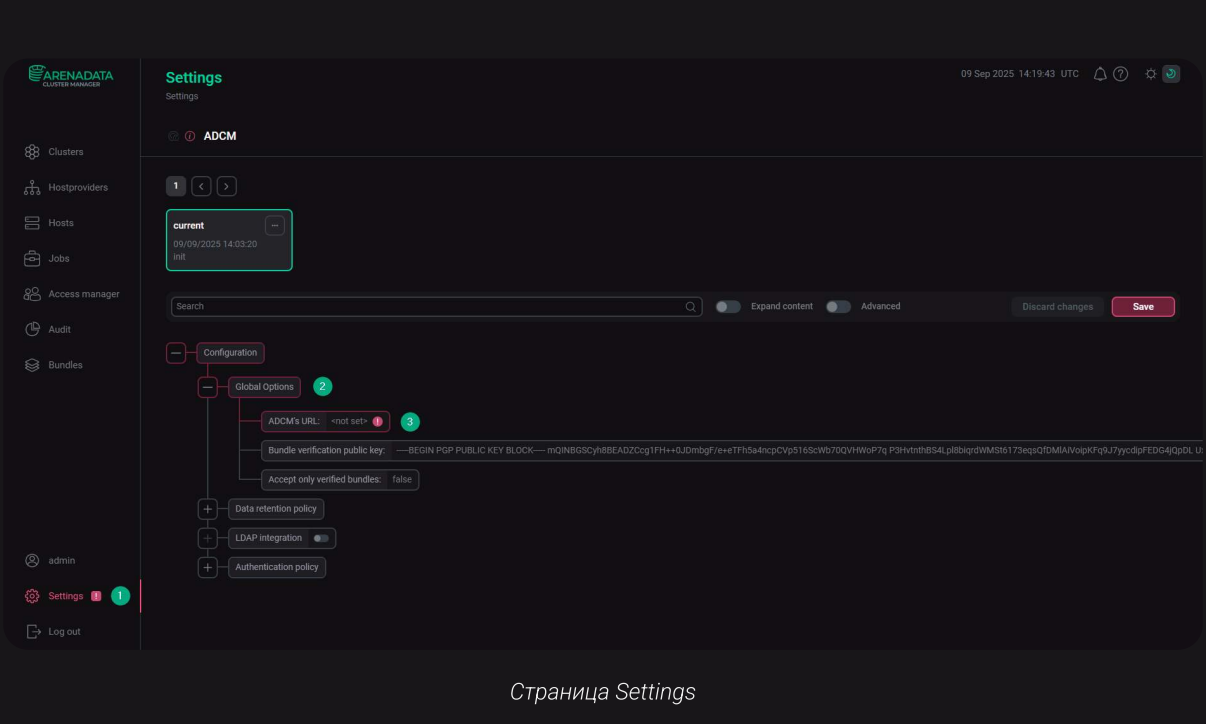


Изменение пароля пользователя admin

Установка URL ADCM

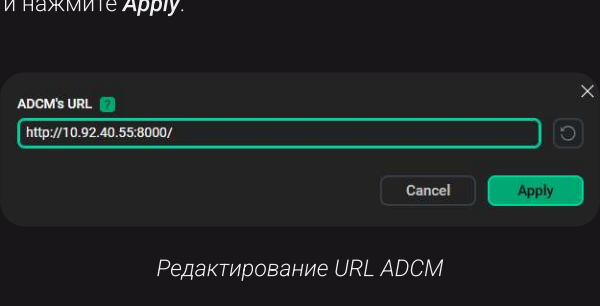
Для успешного обмена данными между ADCM и развернутыми с его помощью кластерами важно корректное определение внешнего адреса ADCM. В дальнейшем этот адрес будет использоваться для отправки в ADCM статусов кластерных компонентов. Для указания URL ADCM используется переменная окружения `DEFAULT_ADCM_URL` при создании контейнера ADCM. Если переменная не была указана, необходимо установить URL вручную в web-интерфейсе ADCM. Для этого выполните шаги:

1. Выберите пункт ***Settings*** в левом навигационном меню.
2. Раскройте узел ***Global Options*** в дереве конфигурационных настроек.
3. Нажмите на значение поля ***ADCM's URL***.



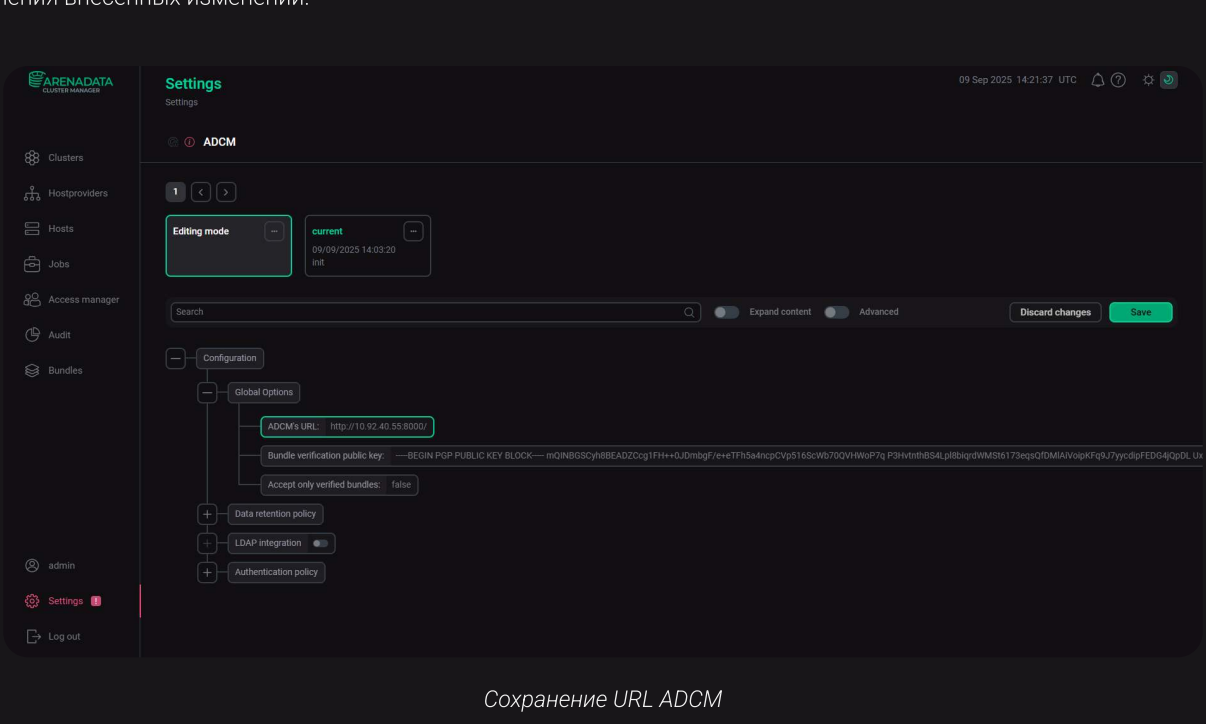
Страница Settings

4. В открывшемся окне измените значение ***ADCM's URL*** и нажмите ***Apply***.



Редактирование URL ADCM

5. Нажмите ***Save*** для сохранения внесенных изменений.



Сохранение URL ADCM

Шаг 5. Обновление ADCM

ВНИМАНИЕ

- Перед выполнением обновления рекомендуется сделать резервную копию данных ADCM. Дополнительную информацию можно получить в статье Резервное копирование и восстановление.
- Понижение версии ADCM не поддерживается.
- Не рекомендуется обновлять ADCM более, чем на 5 минорных версий (см. `<minor>` в описании формата версионирования ниже).
- Перед обновлением ADCM до версии 2.7.0 выполните миграцию данных из SQLite в PostgreSQL. Миграцию можно осуществить в ADCM любой версии (до 2.6.0 включительно).

Чтобы выполнить offline-обновление ADCM до нужной версии, сделайте следующее:

1. Остановите Docker-контейнер `adcm`:

```
$ sudo docker stop adcm
```

2. Удалите Docker-контейнер `adcm`:

```
$ sudo docker container rm adcm
```

ПРИМЕЧАНИЕ

Клиенты, использующие Enterprise-версии продуктов, могут обратиться в команду поддержки Arenadata для получения архива с расширением *sh.xz*, который содержит специальный скрипт для ADCM. После остановки Docker-контейнера `adcm` дальнейшее обновление ADCM включает установку согласно инструкции.

3. Загрузите Docker-образ из Arenadata Docker Registry на хост, который имеет доступ в Интернет.

```
$ sudo docker pull hub.arenadata.io/adcm/adcm:<version>
```

где `<version>` — это версия Docker-образа в одном из следующих форматов:

- `<major>.<minor>.<patch>` — если необходим конкретный патч ADCM. Пример: `2.0.0`.
- `<major>.<minor>` — если необходим последний патч в рамках выбранной версии ADCM. Пример: `2.0`.

ВАЖНО

- `latest` использовался для ранних версий ADCM и начиная с версии `2.0.0` больше не поддерживается.
- До выпуска ADCM 2.0.0 для версионирования использовался следующий формат: `YYYY.MM.DD.HH`.

4. Сохраните загруженный Docker-образ в файл с расширением *.tar*.

```
$ sudo docker save -o <file_name> hub.arenadata.io/adcm/adcm:<version>
```

где `<file_name>` — имя создаваемого файла с расширением (например, *adcm_2.5.0_arenadata.tar*).

ПРИМЕЧАНИЕ

Выполнение команды может занять некоторое время. Пожалуйста, дождитесь ее завершения.

5. Скопируйте сохраненный файл с текущего хоста на хост без доступа к Интернету любым из доступных способов.

ВАЖНО

Убедитесь в наличии настроенного доступа к хостам и необходимых прав доступа к файлу.

6. Загрузите Docker-образ из сохраненного файла.

```
$ sudo docker load -i <file_path>
```

где `<file_path>` — абсолютный путь к сохраненному файлу с Docker-образом.

Пример результата успешного выполнения команды:

```
Loaded image: hub.arenadata.io/adcm/adcm:2.5.0
```

7. Создайте новый Docker-контейнер на базе загруженного образа:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

ПРИМЕЧАНИЕ

Подробная информация о переменных окружения `DB_HOST`, `DB_PORT`, `DB_USER`, `DB_NAME`, `DB_PASS` и `DB_OPTIONS` приведена в статье Установка.

8. Запустите новый Docker-контейнер `adcm`:

```
$ sudo docker start adcm
```

Подготовка хостов

Назад к содержанию

Перед установкой нового кластера необходимо добавить информацию о его будущих хостах в ADCM. Это задача **хостпровайдеров** — специальных плагинов, отвечающих за взаимодействие между хостами и ADCM.

ADCM поддерживает несколько типов хостпровайдеров. Для получения подробной информации о том, как их настроить и использовать для создания хостов, обратитесь к документации:

- SSH
- Yandex Cloud
- VCloud (VMware)
- Cloud.ru Advanced
- CROC Cloud
- ADQM Cloud Bundle

Для offline-установки рекомендуется использование SSH-хостпровайдера.

ВНИМАНИЕ

Если в ADQM планируется использовать аутентификацию Kerberos, при создании хостов ADQM необходимо заполнять их полные доменные имена (FQDN).

Создание кластера

Дарья Барышева

Содержание

Назад к содержанию

Шаг 1. Получение Enterprise-версии бандла

Шаг 2. Загрузка бандла в ADCM

Шаг 3. Создание кластера на базе загруженного бандла

В ADCM **кластер** означает набор сервисов, запущенных на группе хостов.

Дистрибутивы кластеров для ADCM поставляются в виде бандлов. В общем случае бандл представляет обычный архив, описывающий логику развертывания кластера.

Шаги для добавления нового кластера приведены ниже.

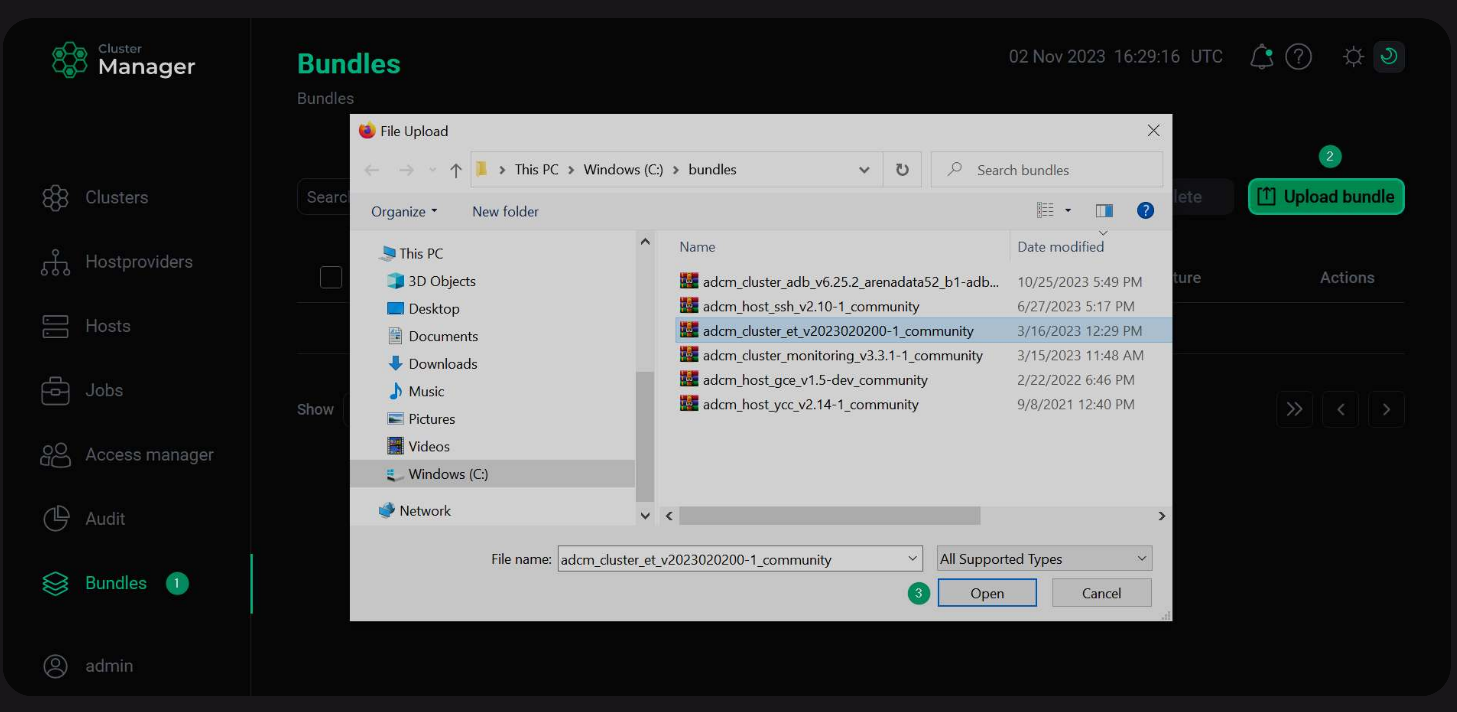
Шаг 1. Получение Enterprise-версии бандла

Пожалуйста, запросите бандл у команды поддержки Arenadata.

Шаг 2. Загрузка бандла в ADCM

Чтобы загрузить бандл в ADCM:

1. Выберите пункт ***Bundles*** в левом навигационном меню и нажмите ***Upload bundle***.
2. Выберите бандл в открывшемся диалоговом окне.


Cluster Manager
Bundles
Bundles
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
1
admin
02 Nov 2023 16:29:16 UTC
2
Upload bundle
File Upload
This PC > Windows (C:) > bundles
Search bundles
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Network
3
Open
Cancel

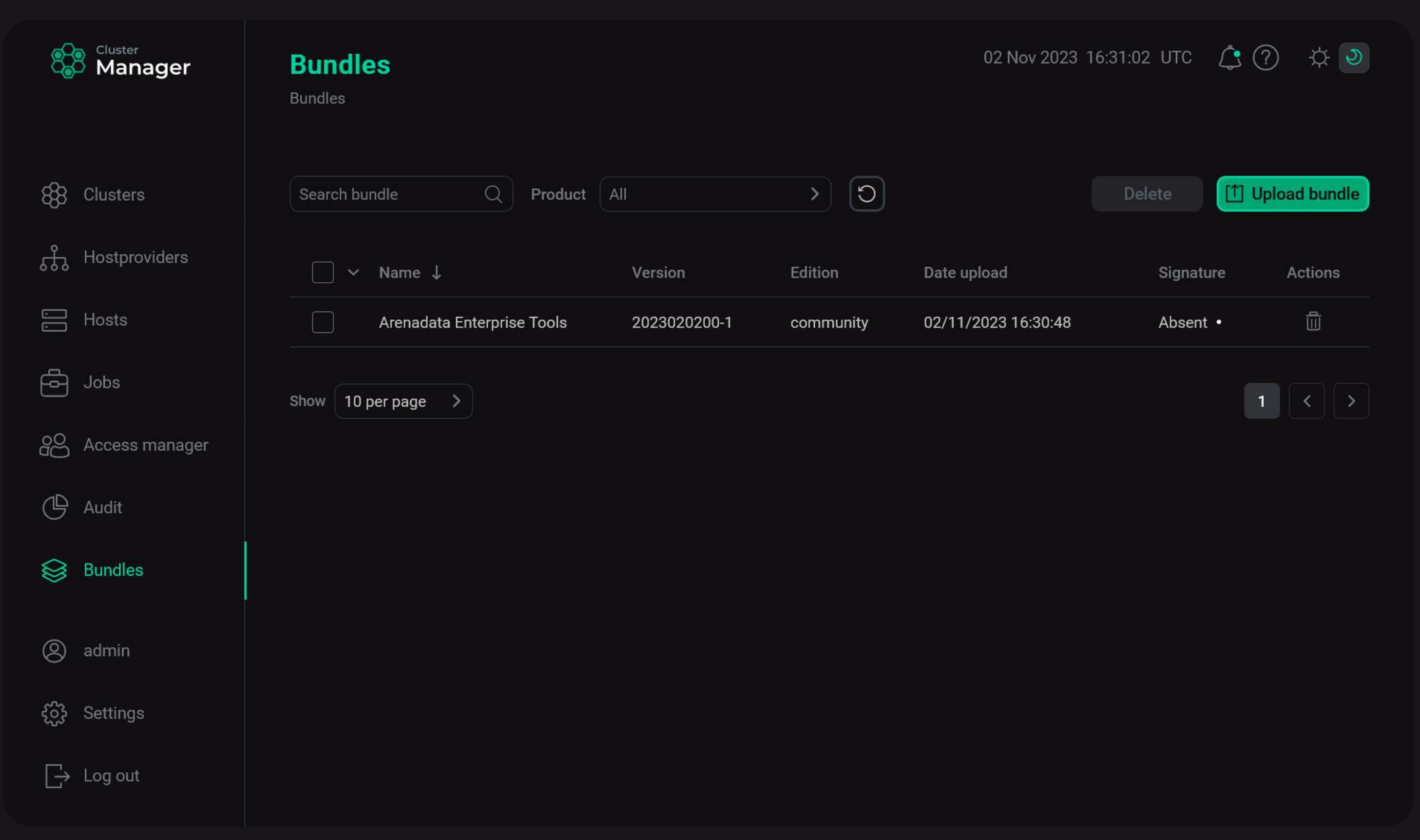
Name	Date modified
adcm_cluster_adb_v6.25.2_arenadata52_b1-adb...	10/25/2023 5:49 PM
adcm_host_ssh_v2.10-1_community	6/27/2023 5:17 PM
adcm_cluster_et_v2023020200-1_community	3/16/2023 12:29 PM
adcm_cluster_monitoring_v3.3.1-1_community	3/15/2023 11:48 AM
adcm_host_gce_v1.5-dev_community	2/22/2022 6:46 PM
adcm_host_ycc_v2.14-1_community	9/8/2021 12:40 PM

File name: adcm_cluster_et_v2023020200-1_community

All Supported Types

Переход к загрузке бандла

3. После успешной загрузки бандл отображается на странице ***Bundles***.


Cluster Manager
Bundles
Bundles
02 Nov 2023 16:31:02 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search bundle
Product
All
Delete
Upload bundle
Show
10 per page
1

Name	Version	Edition	Date upload	Signature	Actions
Arenadata Enterprise Tools	2023020200-1	community	02/11/2023 16:30:48	Absent •	

Результат успешной загрузки бандла

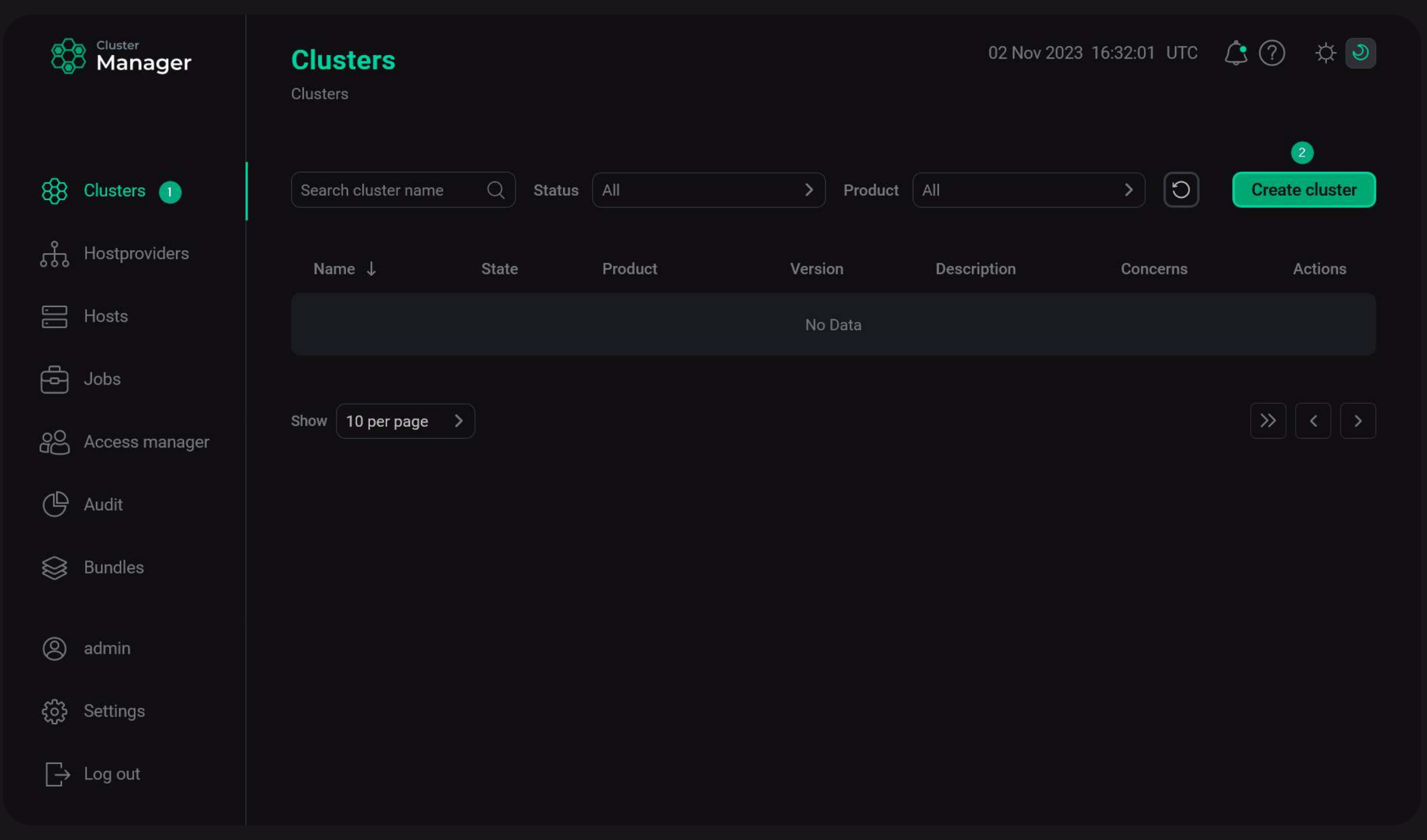
Шаг 3. Создание кластера на базе загруженного бандла

ВАЖНО

На этапе создания кластера через web-интерфейс ADCM генерируется только сам кластер. Это означает добавление записи о кластере в ADCM — сервисы на этом шаге еще не устанавливаются.

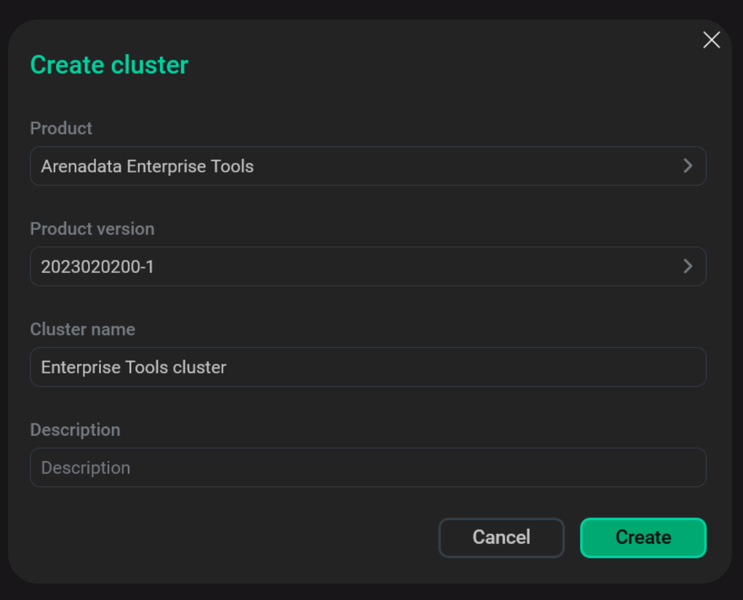
Для создания кластера необходимо выполнить шаги:

1. Выберите пункт ***Clusters*** в левом навигационном меню и нажмите ***Create cluster***.


Cluster Manager
Clusters
Clusters
02 Nov 2023 16:32:01 UTC
Clusters
1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
2
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
No Data
Show
10 per page

Переход к созданию кластера

2. В открывшемся окне требуется описать новый кластер:
   - Выберите загруженный бандл в поле ***Product***.
   - Выберите версию бандла в поле ***Product version***. Несколько версий становятся доступны после загрузки нескольких бандлов для одного типа продуктов.
   - Введите имя кластера в поле ***Cluster name***.
   - При необходимости добавьте описание кластера в поле ***Description***.
   - Установите флаг ***I accept Terms of Agreement*** для подписания лицензионного соглашения. Флаг отображается только для бандлов версии Enterprise. Чтобы ознакомиться с текстом соглашения, перейдите по ссылке ***Terms of Agreement***. Подписать соглашение также возможно на странице ***Bundles***, нажав на соответствующий продуктовый бандл.
   - Нажмите ***Create***.


Create cluster
Cancel
Create

Product: Arenadata Enterprise Tools

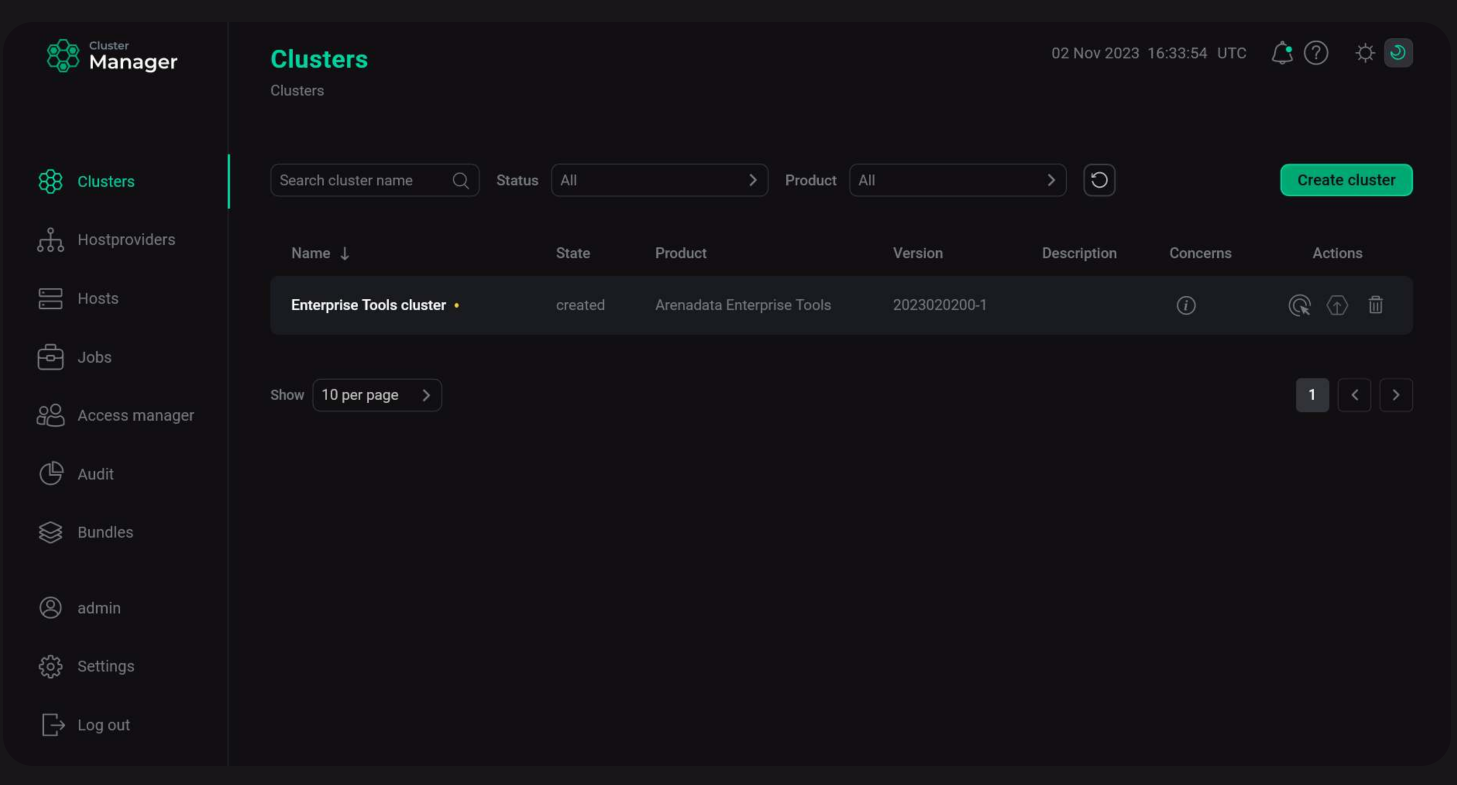
Product version: 2023020200-1

Cluster name: Enterprise Tools cluster

Description: Description

Создание кластера

3. После успешного добавления кластер отображается на странице ***Clusters***.


Cluster Manager
Clusters
Clusters
02 Nov 2023 16:33:54 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster •	created	Arenadata Enterprise Tools	2023020200-1			

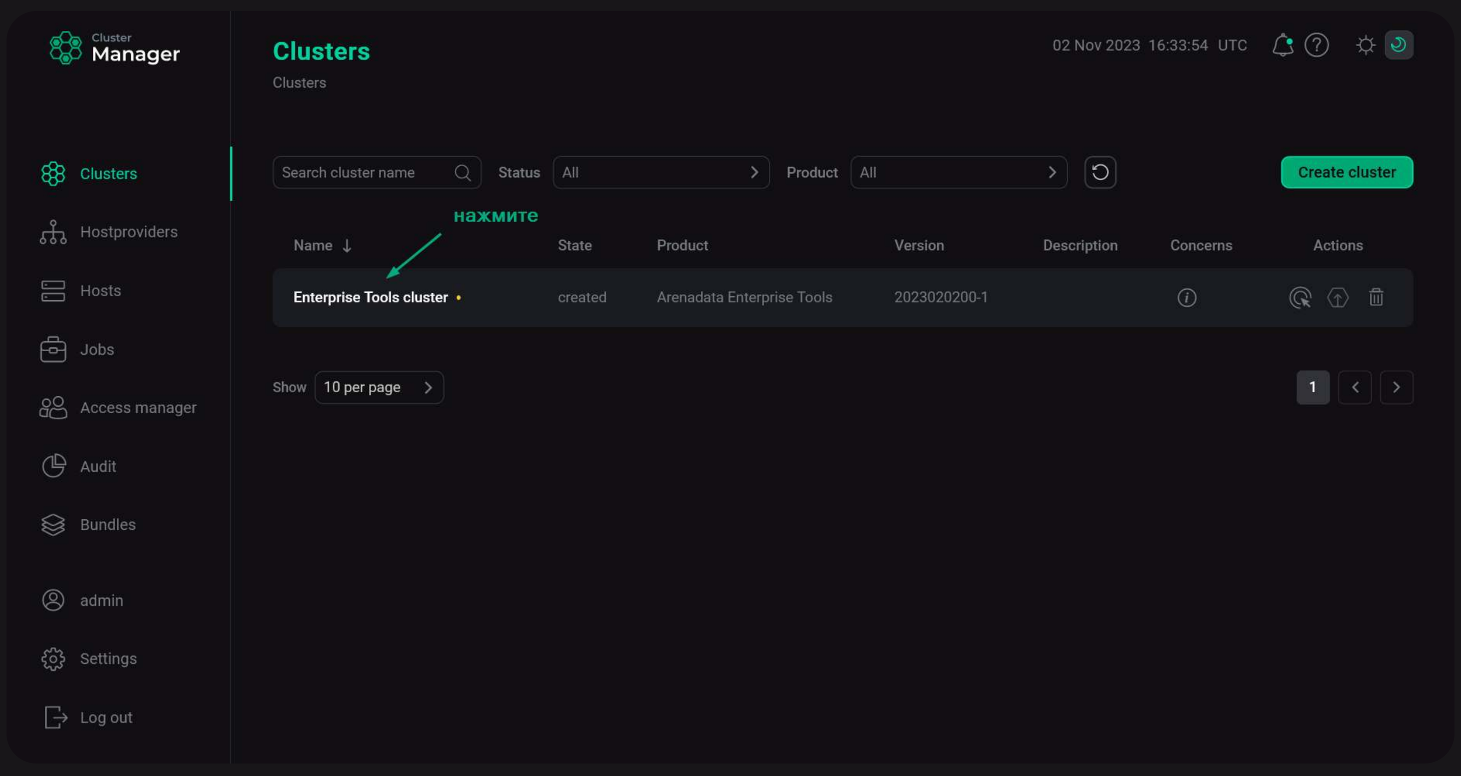
Результат успешного добавления кластера

Добавление сервисов

Дарья Барышева

В ADCM **сервис** означает программное обеспечение, выполняющее некоторую функцию. Для добавления сервисов в кластер:

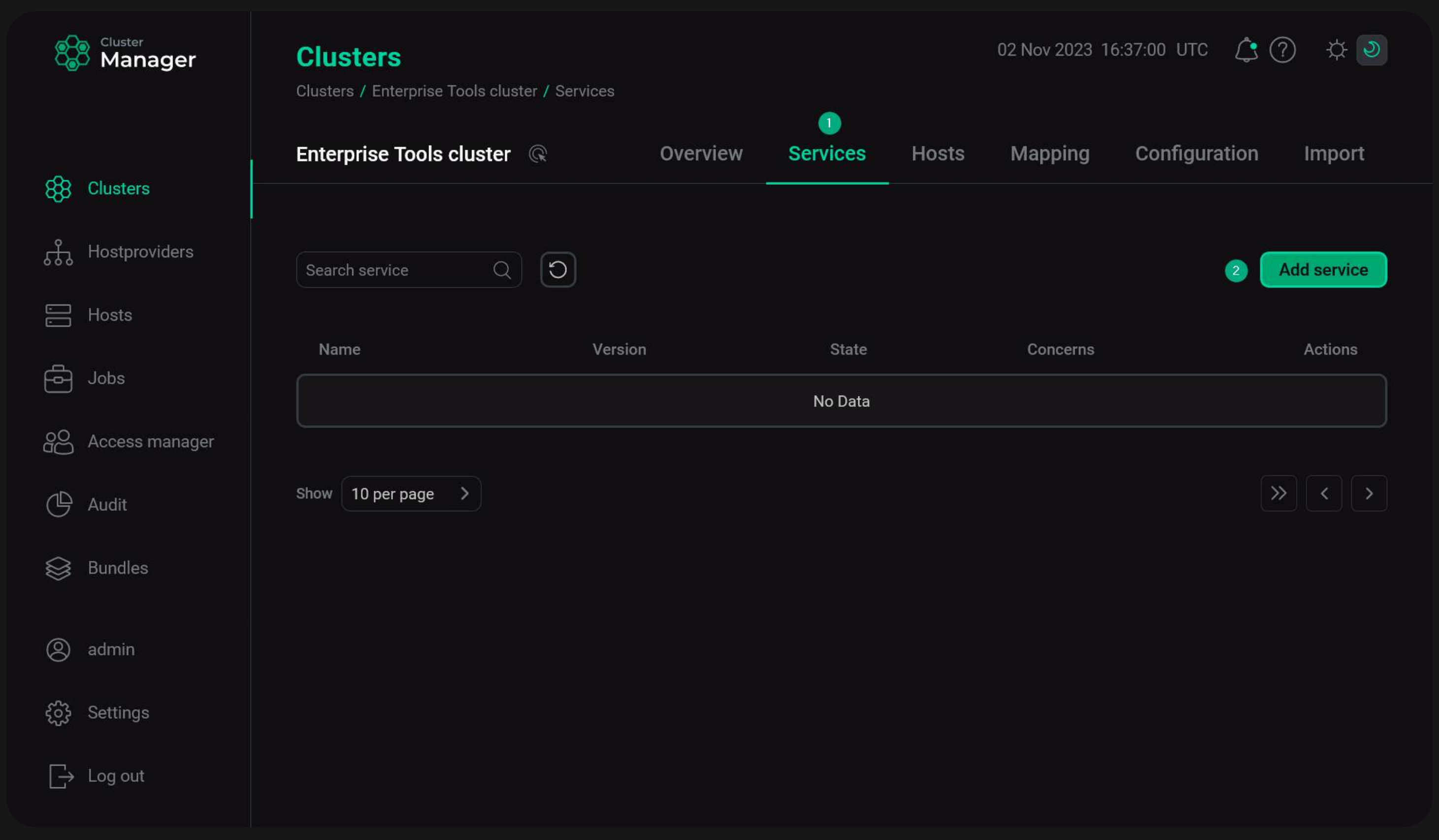
1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
02 Nov 2023 16:33:54 UTC
Search cluster name
Status
All
Product
All
Create cluster
нажмите
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	created	Arenadata Enterprise Tools	2023020200-1			

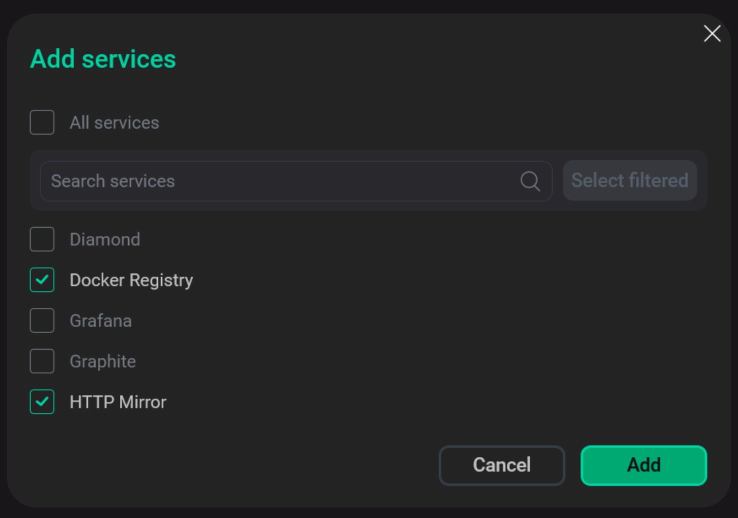
Выбор кластера

2. Откройте вкладку ***Services*** на странице кластера и нажмите ***Add service***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Enterprise Tools cluster / Services
02 Nov 2023 16:37:00 UTC
Enterprise Tools cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page

Переход к добавлению сервисов

3. В открывшемся окне выберите сервисы для добавления в кластер и нажмите ***Add***.


Add services
Search services
Select filtered
Cancel
Add

☐ All services

☐ Diamond
☑ Docker Registry
☐ Grafana
☐ Graphite
☑ HTTP Mirror

Выбор сервисов

Краткое описание доступных сервисов приведено ниже.

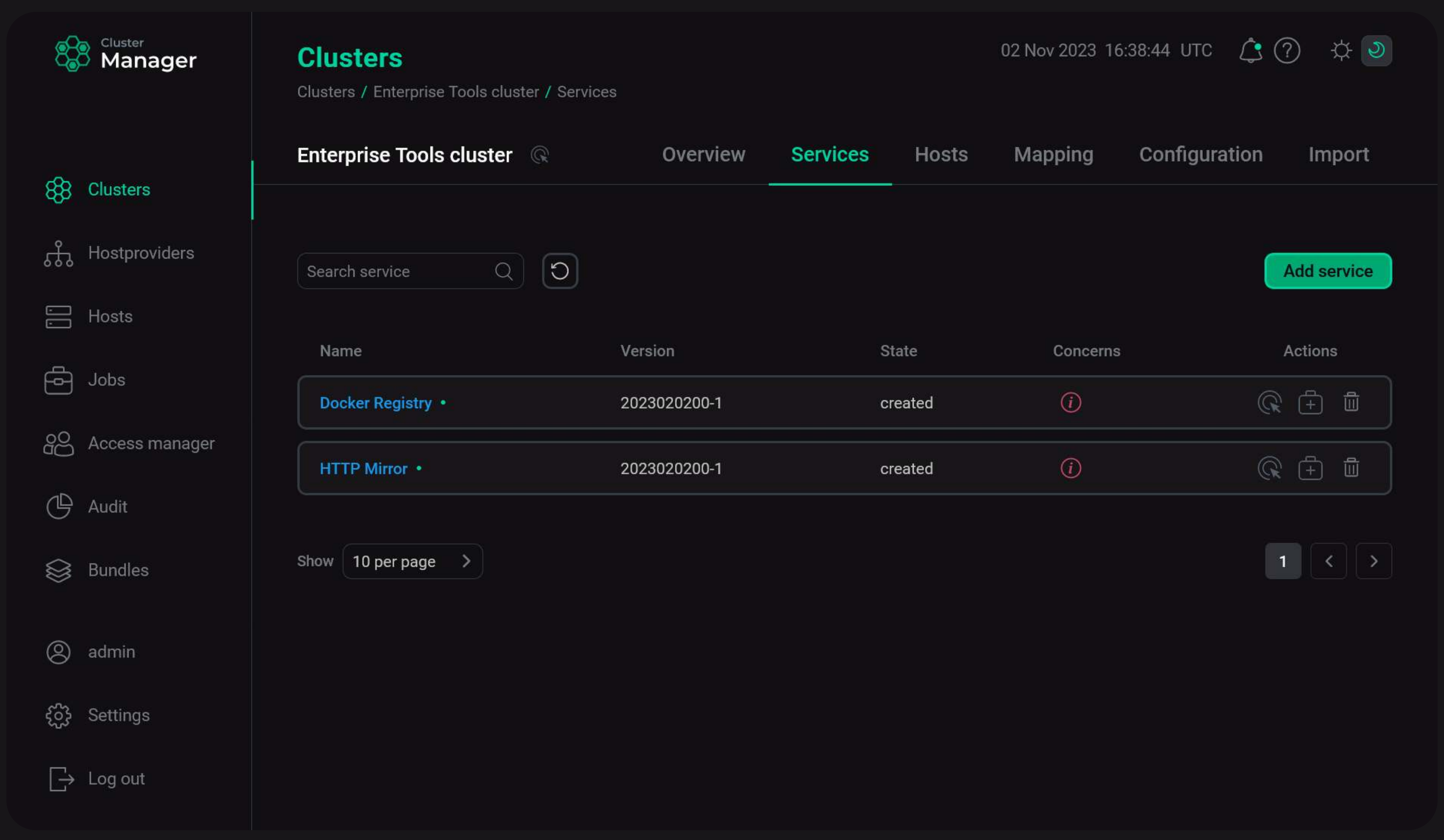
Сервисы, которые необходимо добавить в кластер Enterprise Tools

Сервис	Назначение
HTTP Mirror	Сервис, используемый для размещения пакетов репозиториев, необходимых для offline-установки продуктов
Docker Registry	Сервис, используемый для хранения образов Docker, необходимых для offline-установки продуктов

ПРИМЕЧАНИЕ

Сервис CoreDNS не используется (deprecated). Сервисы Diamond, Graphite и Grafana необходимы для настройки мониторинга — они будут описаны отдельно.

4. В результате успешно добавленные сервисы отображаются на вкладке ***Services***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Enterprise Tools cluster / Services
02 Nov 2023 16:38:44 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
Show
10 per page
1

Name	Version	State	Concerns	Actions
Docker Registry	2023020200-1	created		
HTTP Mirror	2023020200-1	created		

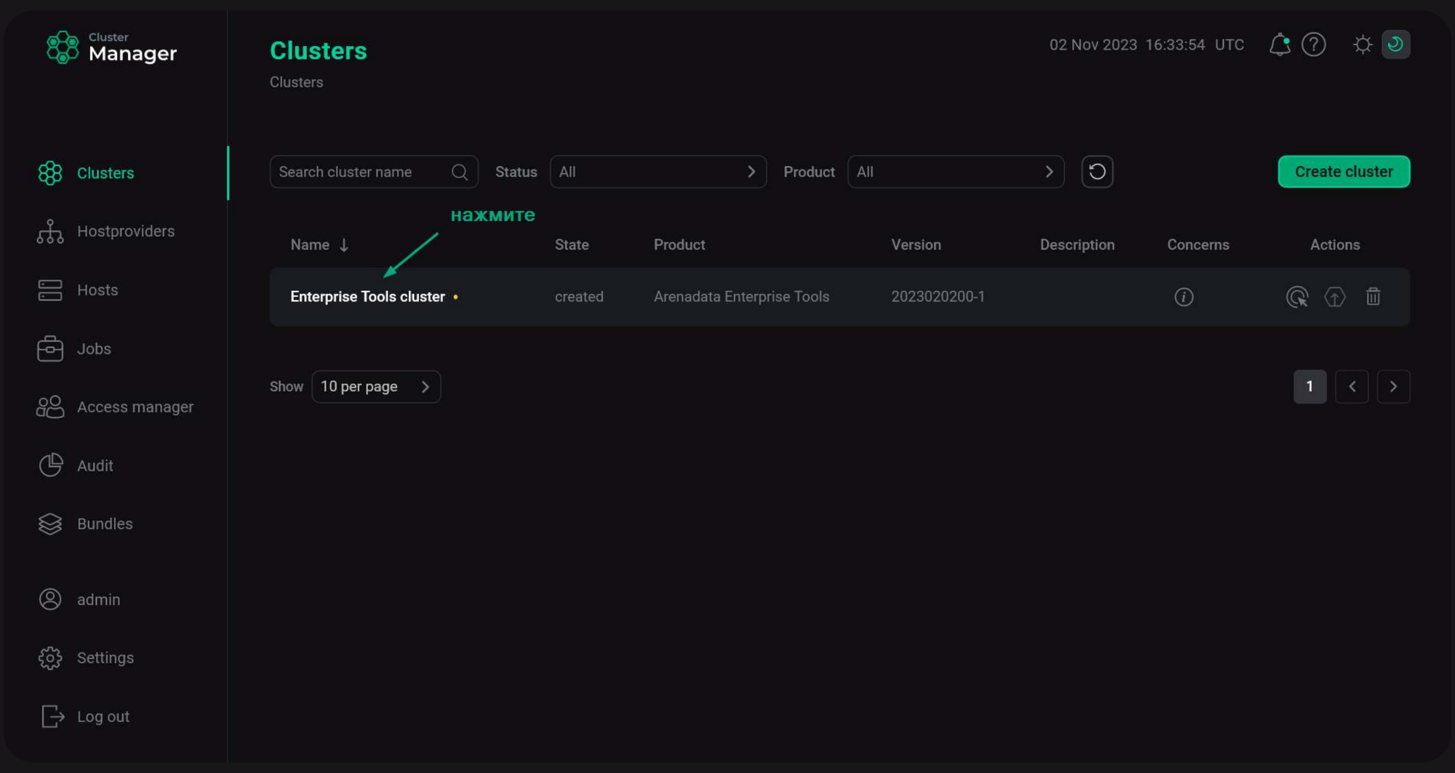
Результат успешного добавления сервисов в кластер

Добавление хостов в кластер

Дарья Барышева

Для добавления хостов в кластер:

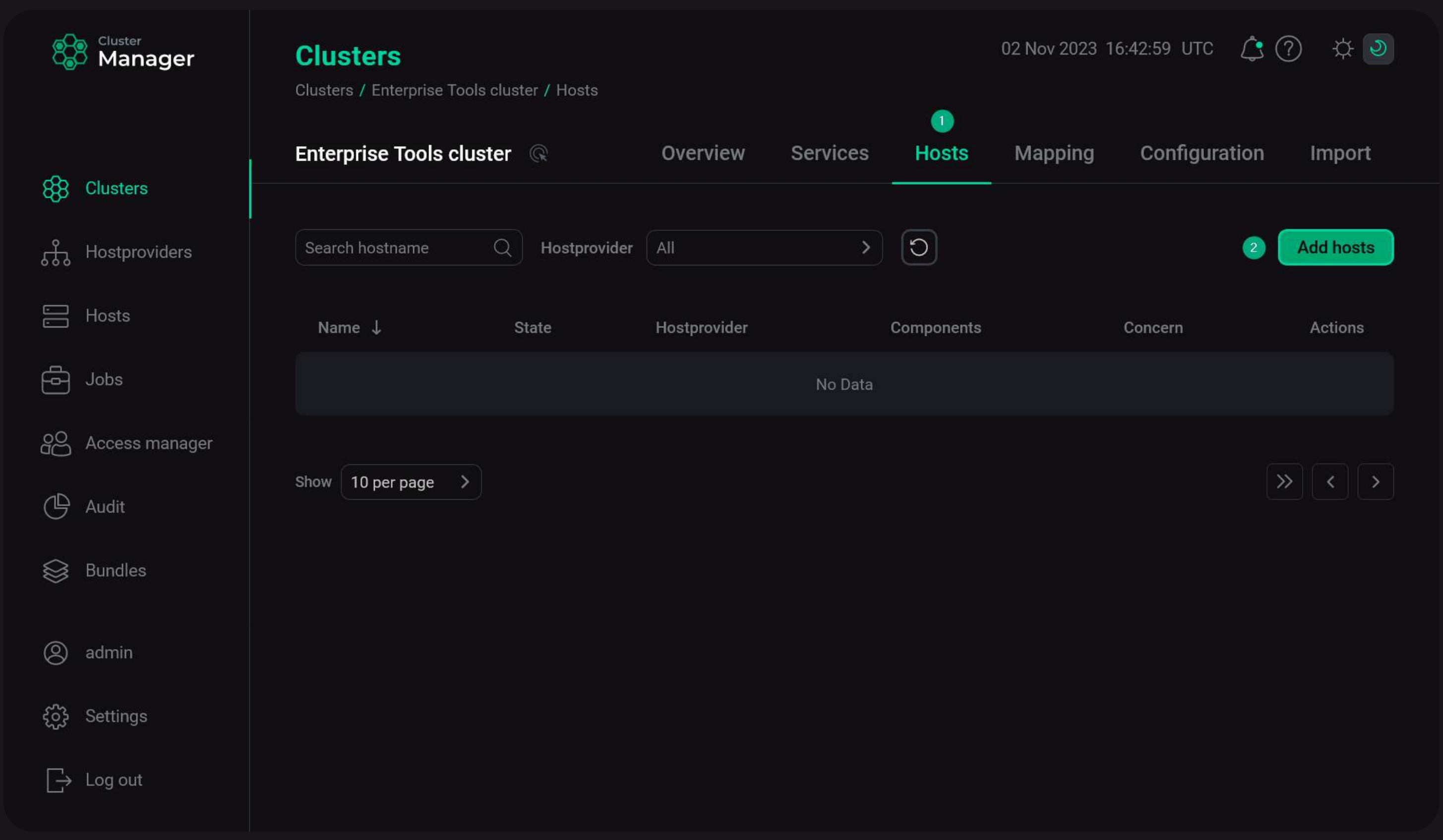
1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Clusters
02 Nov 2023 16:33:54 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
нажмите
Show 10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	created	Arenadata Enterprise Tools	2023020200-1			

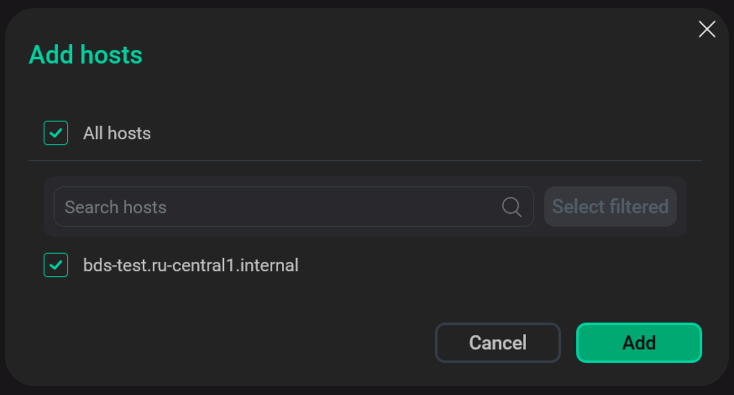
Выбор кластера

2. Откройте вкладку ***Hosts*** на странице кластера и нажмите ***Add hosts***.


Cluster Manager
Clusters
Clusters / Enterprise Tools cluster / Hosts
02 Nov 2023 16:42:59 UTC
Enterprise Tools cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider All
2
Add hosts
Name
State
Hostprovider
Components
Concern
Actions
No Data
Show 10 per page

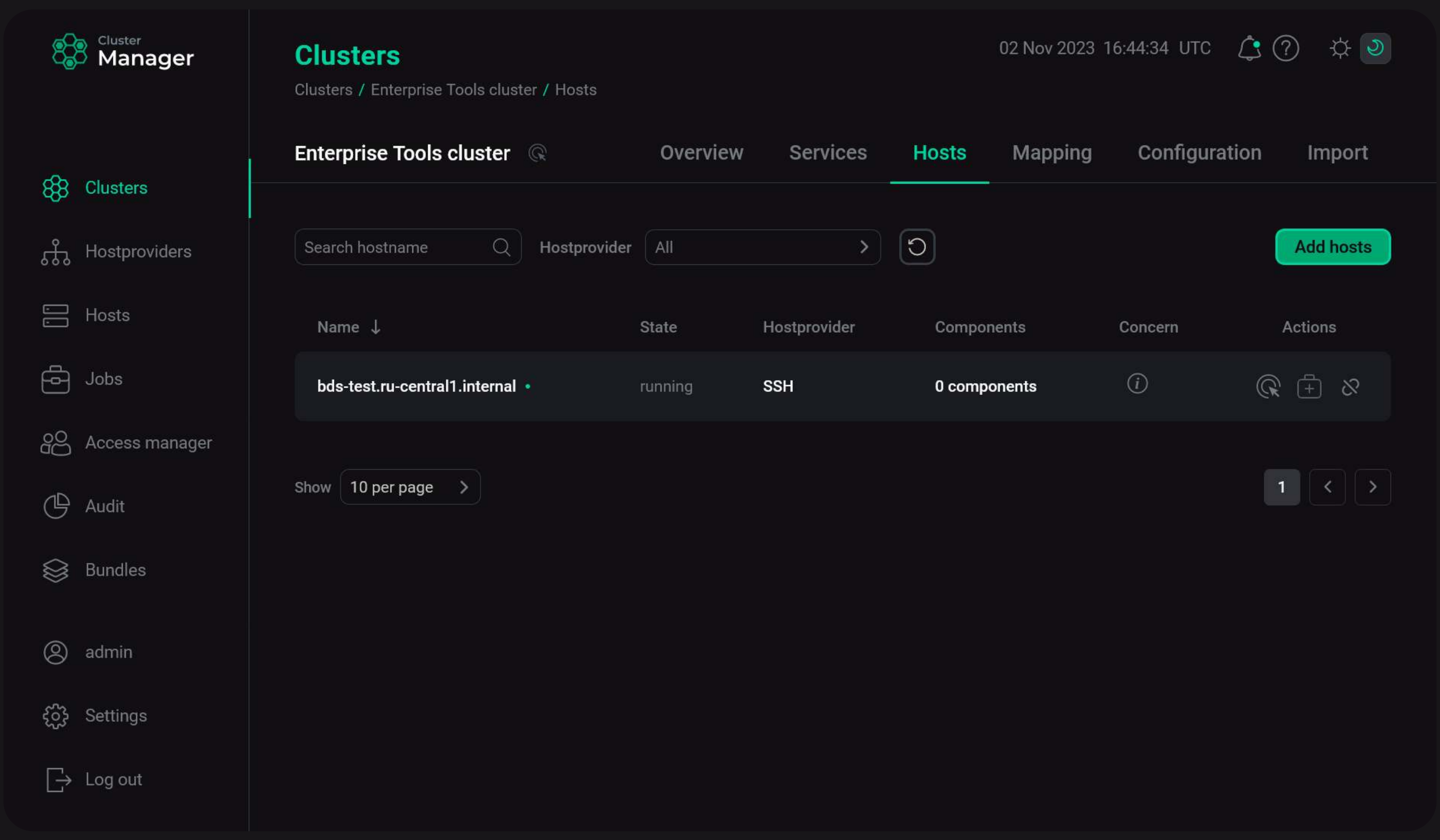
Переход к добавлению хостов

3. В открывшемся окне выберите хосты для добавления в кластер и нажмите ***Add***.


Add hosts
☑ All hosts
Search hosts
Select filtered
☑ bds-test.ru-central1.internal
Cancel
Add

Выбор хостов

4. В результате успешно добавленные хосты отображаются на вкладке ***Hosts***.


Cluster Manager
Clusters
Clusters / Enterprise Tools cluster / Hosts
02 Nov 2023 16:44:34 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider All
Add hosts
Show 10 per page
1

Name	State	Hostprovider	Components	Concern	Actions
bds-test.ru-central1.internal	running	SSH	0 components		

Результат успешного добавления хостов в кластер

Добавление компонентов

Дарья Барышева

Назад к содержанию

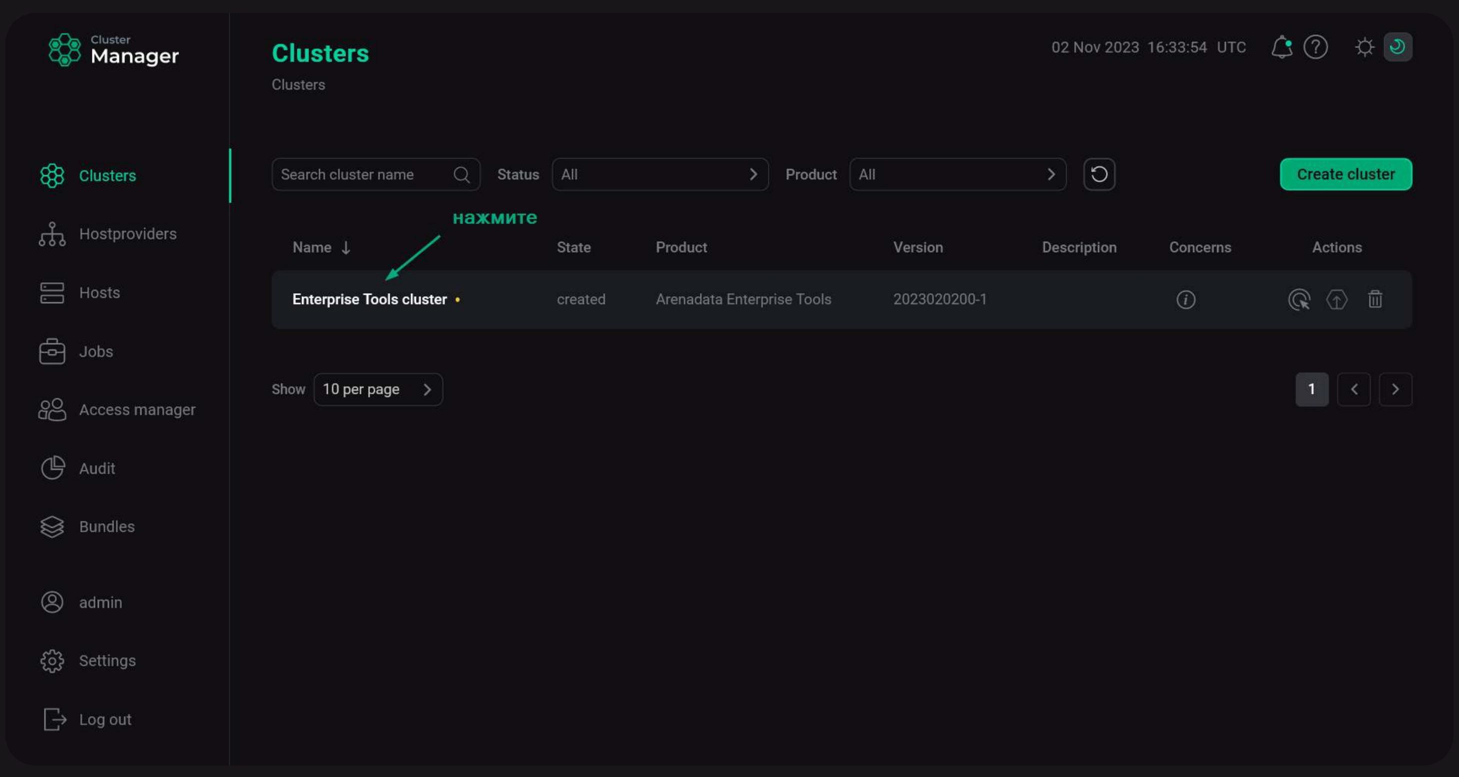
В ADCM **компонент** означает часть сервиса, которая должна быть развернута на одном или нескольких хостах кластера.

ВАЖНО

Изначально на хостах кластера нет ни одного компонента — распределение компонентов между хостами должно быть проведено вручную.

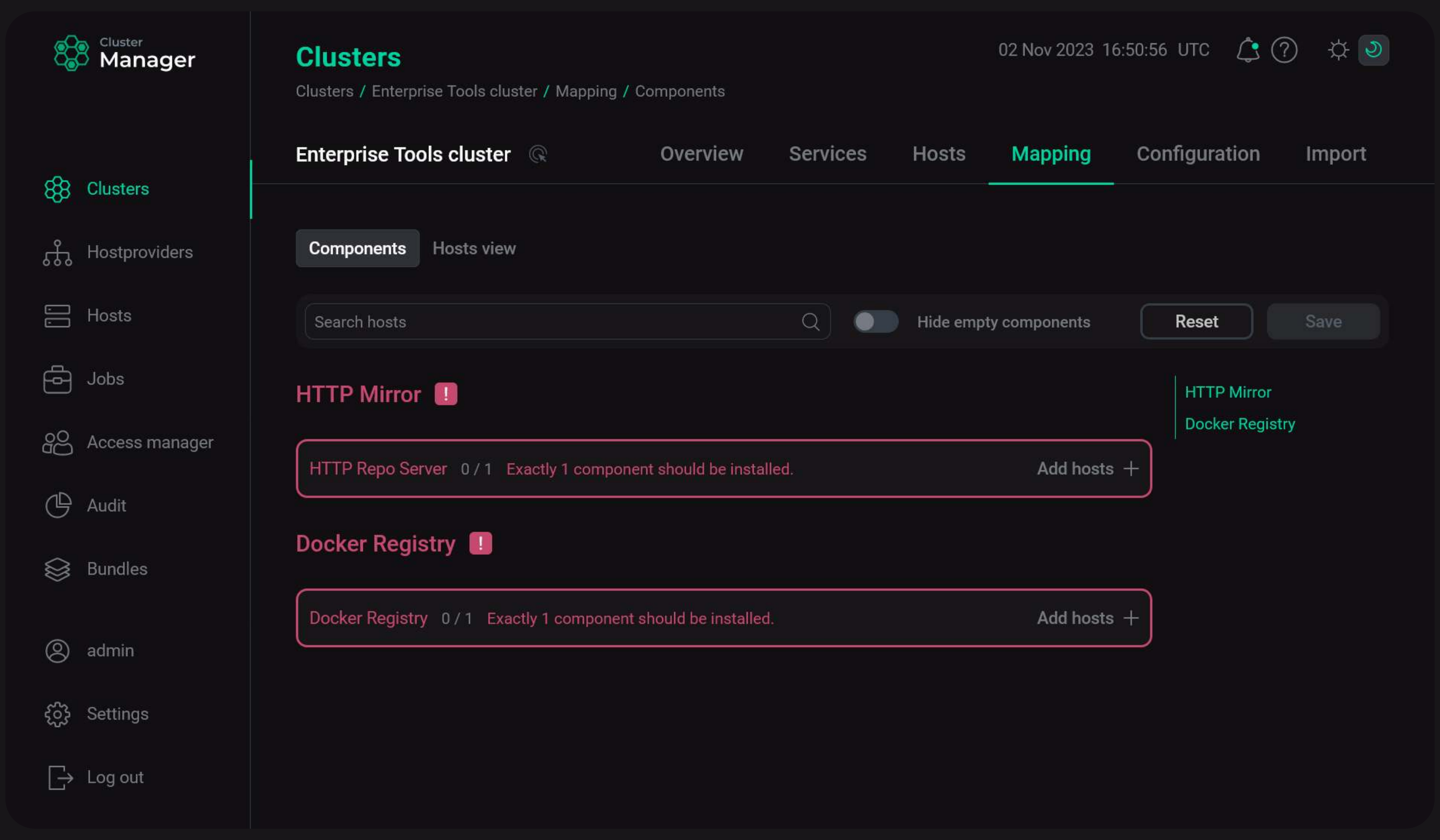
Для добавления компонентов на хосты кластера:

1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
02 Nov 2023 16:33:54 UTC
Search cluster name
Status
All
Product
All
Create cluster
нажмите
Name
State
Product
Version
Description
Concerns
Actions
Enterprise Tools cluster
created
Arenadata Enterprise Tools
2023020200-1
Show
10 per page
1

Выбор кластера

2. Откройте вкладку ***Mapping*** на странице кластера.

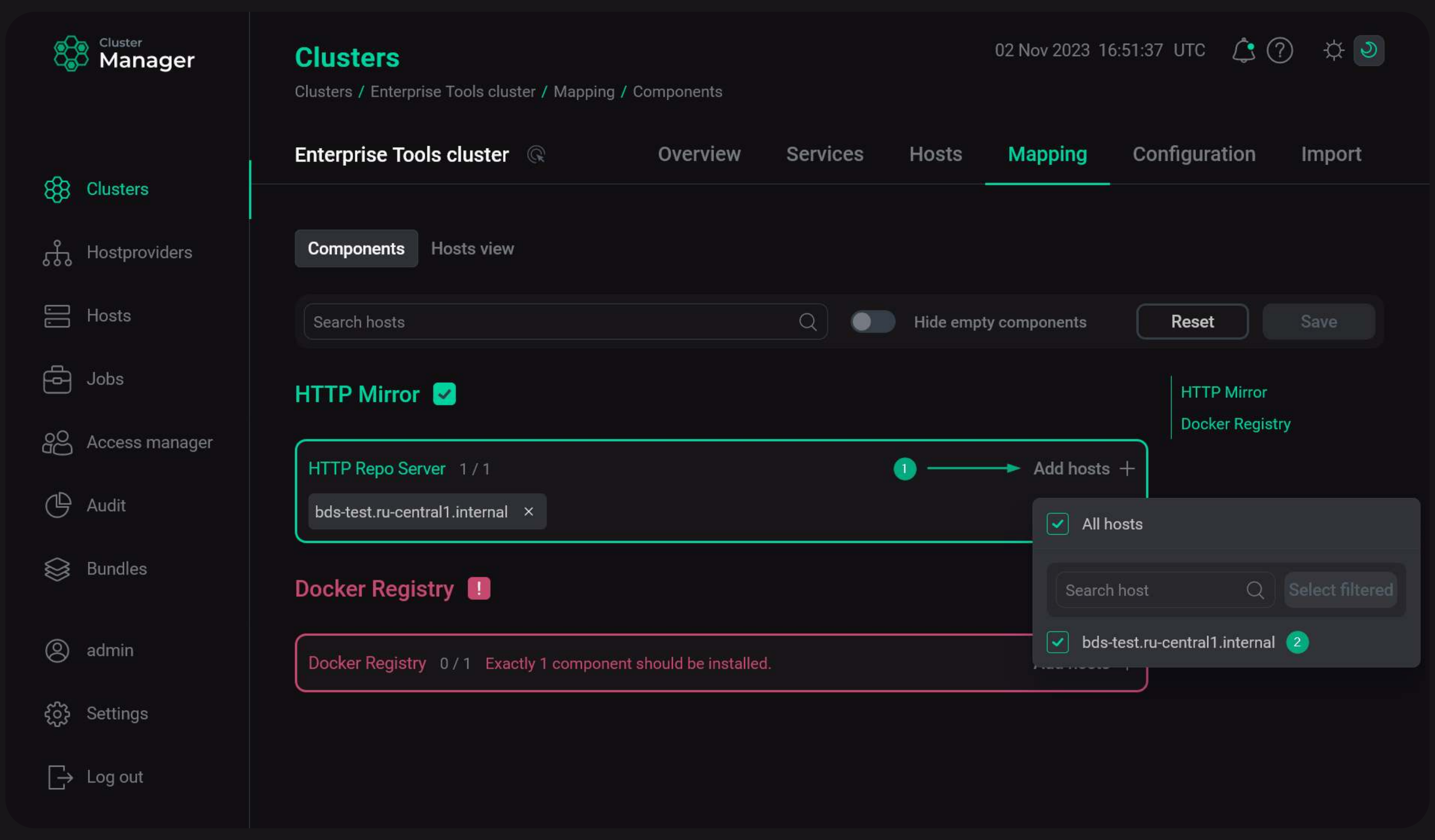

Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Enterprise Tools cluster / Mapping / Components
02 Nov 2023 16:50:56 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
HTTP Mirror
HTTP Mirror
Docker Registry
HTTP Repo Server 0 / 1
Add hosts
Docker Registry
Docker Registry 0 / 1
Add hosts

Exactly 1 component should be installed.

Exactly 1 component should be installed.

Переход к распределению сервисных компонентов

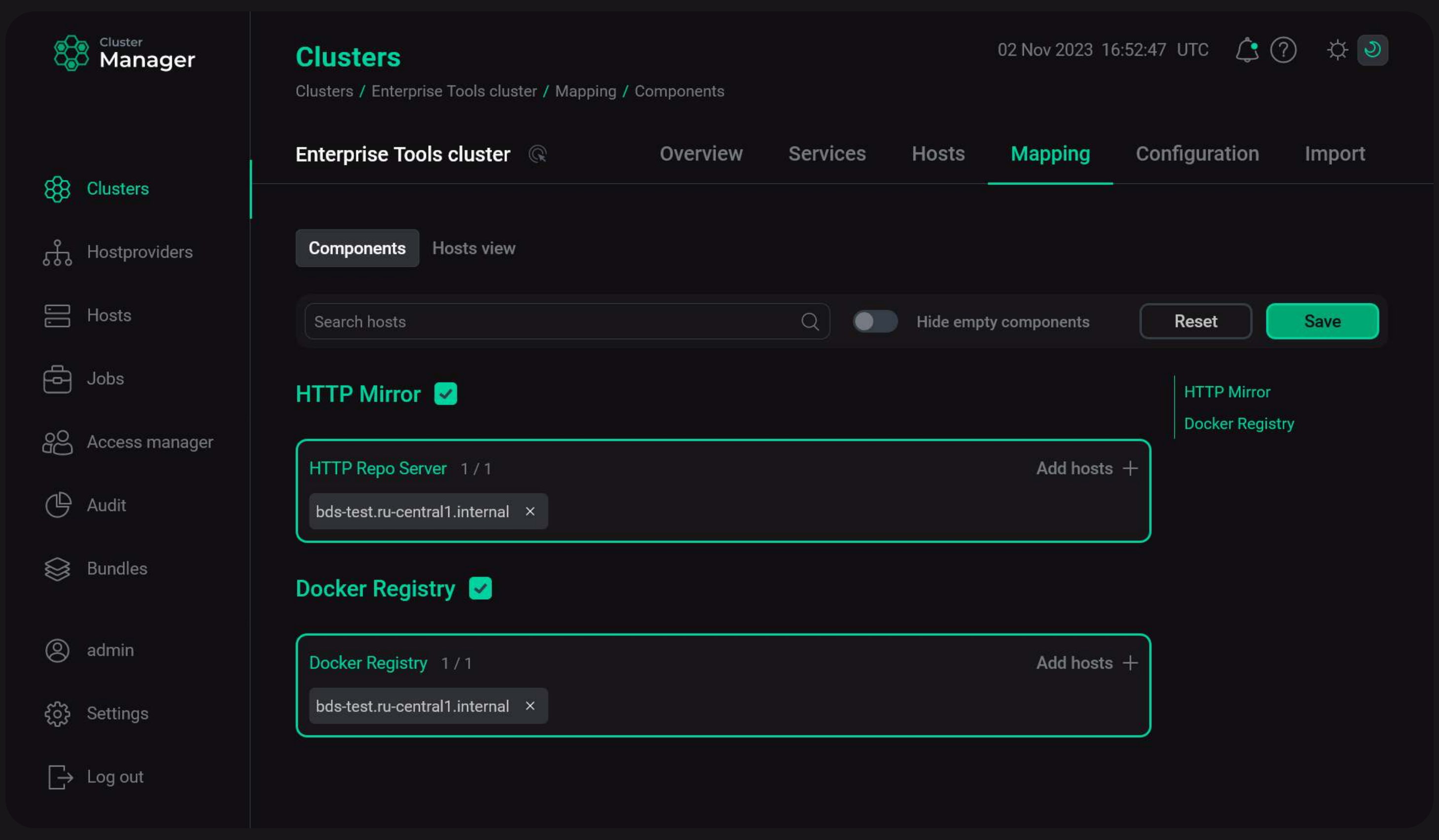
3. Распределите все компоненты между хостами кластера. Для назначения хостов компоненту нажмите ***Add hosts*** и выберите хосты в открывшемся окне.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Enterprise Tools cluster / Mapping / Components
02 Nov 2023 16:51:37 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
HTTP Mirror
HTTP Mirror
Docker Registry
HTTP Repo Server 1 / 1
1
Add hosts
bds-test.ru-central1.internal
All hosts
Search host
Select filtered
bds-test.ru-central1.internal
2
Docker Registry
Docker Registry 0 / 1

Exactly 1 component should be installed.

Выбор хостов для компонента

4. После завершения распределения компонентов нажмите ***Save***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Enterprise Tools cluster / Mapping / Components
02 Nov 2023 16:52:47 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
HTTP Mirror
HTTP Mirror
Docker Registry
HTTP Repo Server 1 / 1
Add hosts
bds-test.ru-central1.internal
Docker Registry
Docker Registry 1 / 1
Add hosts
bds-test.ru-central1.internal

Сохранение распределения компонентов

Установка кластера

Дарья Барышева

Содержание

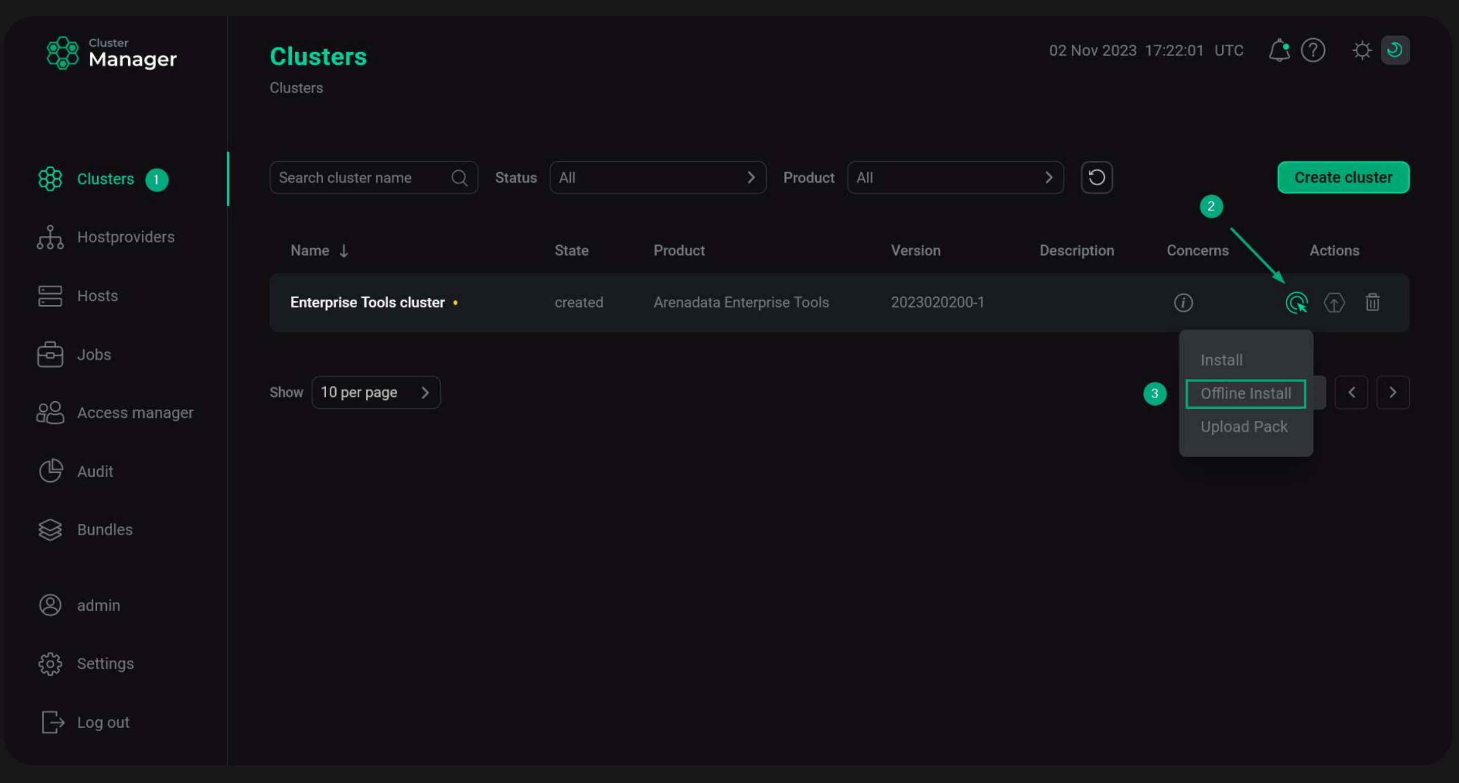
Назад к содержанию

Шаг 1. Запуск установки
Шаг 2. Загрузка пакетов для offline-установки
Шаг 3. Проверка результатов

Шаг 1. Запуск установки

Для установки кластера Enterprise Tools выполните следующие шаги:

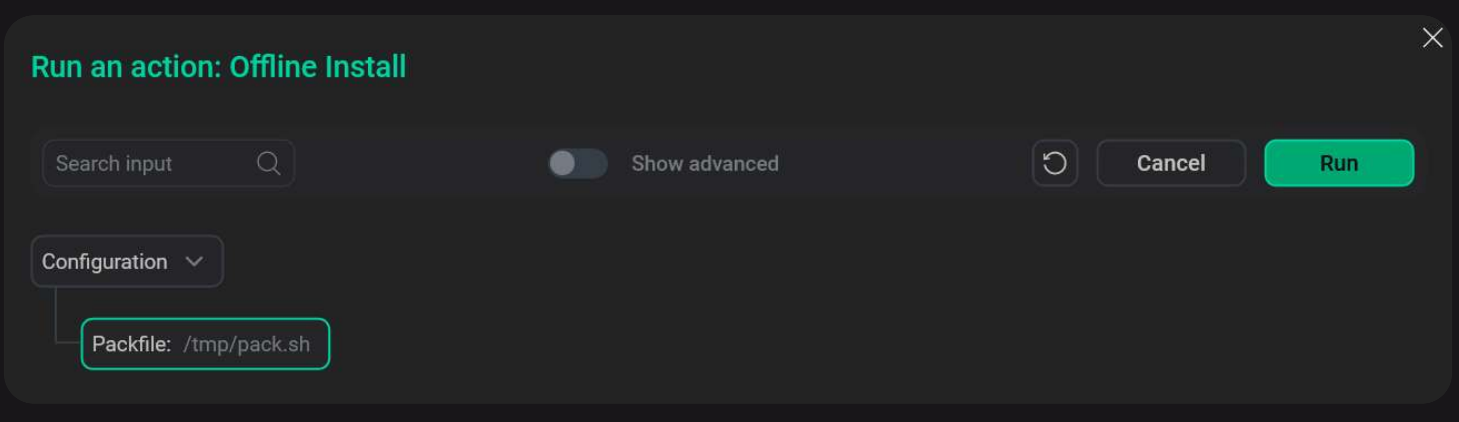
1. Запустите действие ***Offline Install*** для кластера, выбранного на странице ***Clusters***, нажав на иконку ◎ в столбце ***Actions***.


Cluster Manager
Clusters
Clusters
02 Nov 2023 17:22:01 UTC
Clusters
1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
2
Install
Offline Install
Upload Pack
3
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	created	Arenadata Enterprise Tools	2023020200-1			

Переход к установке кластера

2. В открывшемся диалоговом окне введите полный путь к shell-скрипту, полученному от команды поддержки Arenadata, и нажмите ***Run***.


Run an action: Offline Install
Search input
Show advanced
Cancel
Run
Configuration

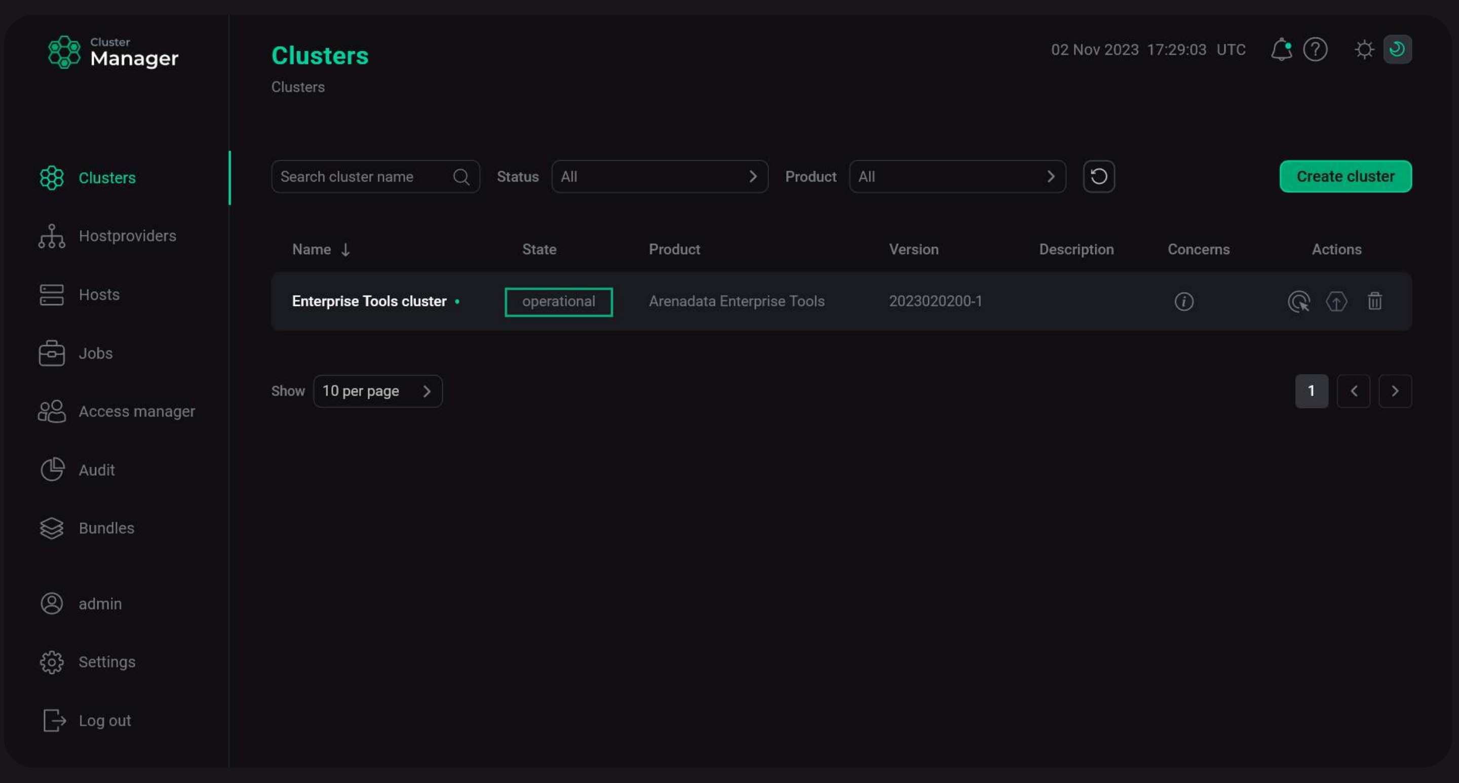
Packfile: /tmp/pack.sh

Подтверждение установки

3. Дождитесь завершения процесса установки.

ПРИМЕЧАНИЕ

В случае ошибок логи можно найти на странице ***Jobs***.


Cluster Manager
Clusters
Clusters
02 Nov 2023 17:29:03 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
1

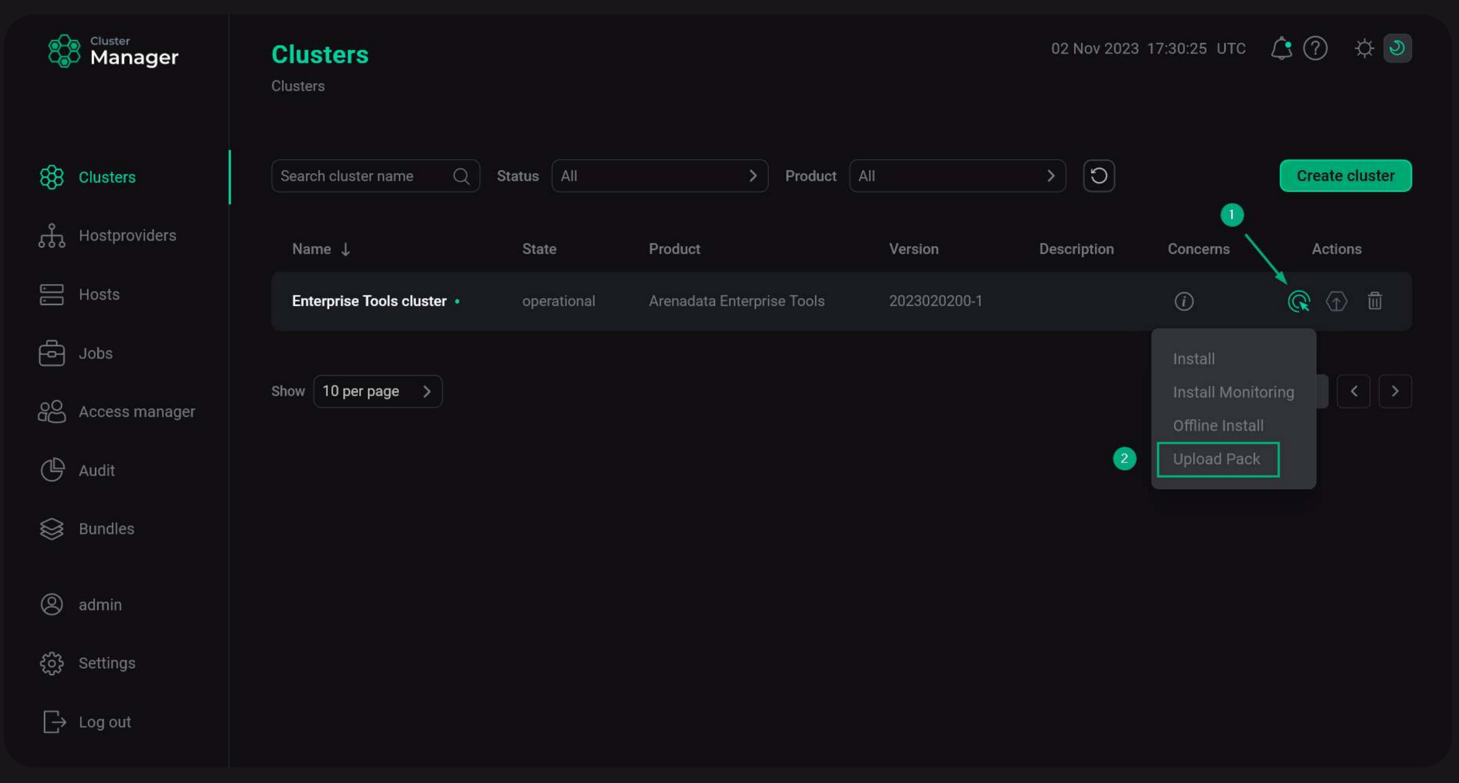
Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	operational	Arenadata Enterprise Tools	2023020200-1			

Процесс установки завершен

Шаг 2. Загрузка пакетов для offline-установки

В дополнение к установке сервисов ET требуется загрузка всех установочных файлов и образов в инициализированный http-репозиторий и Docker Registry на хосте, где развернут кластер ET. Это важно для последующего offline-развертывания продуктов Arenadata. Для загрузки выполните шаги:

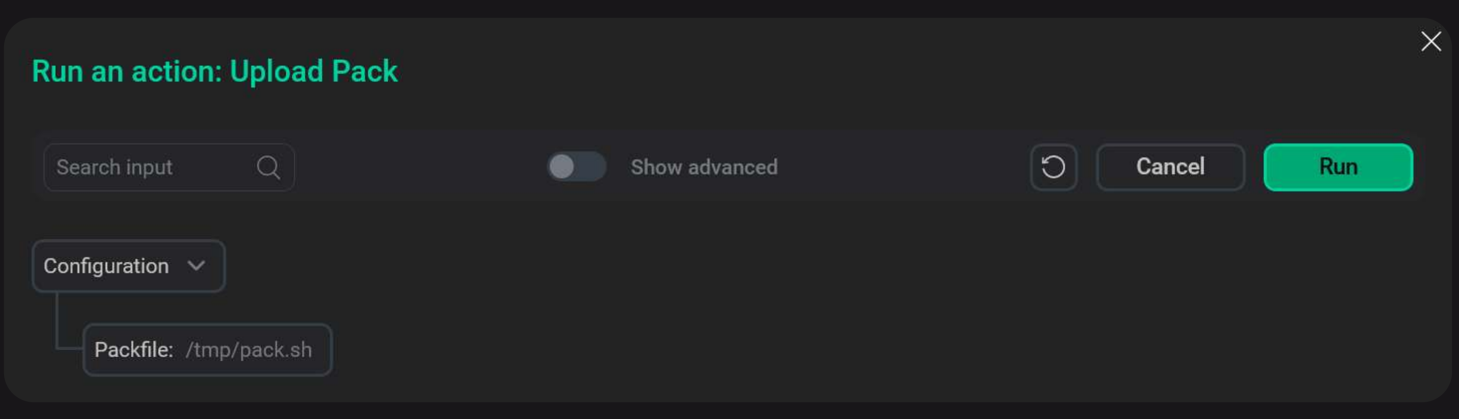
1. Запустите действие ***Upload Pack*** для кластера, выбранного на странице ***Clusters***, нажав на иконку ◎ в столбце ***Actions***.


Cluster Manager
Clusters
Clusters
02 Nov 2023 17:30:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
1
Install
Install Monitoring
Offline Install
Upload Pack
2
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	operational	Arenadata Enterprise Tools	2023020200-1			

Переход к загрузке пакетов

2. В открывшемся диалоговом окне введите полный путь к shell-скрипту, полученному от команды поддержки Arenadata, и нажмите ***Run***.


Run an action: Upload Pack
Search input
Show advanced
Cancel
Run
Configuration

Packfile: /tmp/pack.sh

Подтверждение загрузки

3. Дождитесь завершения процесса загрузки.

ПРИМЕЧАНИЕ

В случае ошибок логи можно найти на странице ***Jobs***.

Шаг 3. Проверка результатов

Для проверки установки на хосте, где развернут кластер, выполните шаги:

1. Убедитесь, что порты `443` и `81` находятся в статусе `LISTEN`. Первый порт используется сервисом Docker Registry, второй порт — сервисом HTTP Mirror.

```
$ sudo netstat -ntlp|grep L|grep -E "81|443"
```

Результат команды:

```
tcp6       0      0 :::443                  :::*                    LISTEN      1694/docker-proxy-c
tcp6       0      0 :::81                   :::*                    LISTEN      1708/docker-proxy-c
```

2. Проверьте наличие директории */opt/arenadata/etools/httprepo/arenadata-repo*. Она должна содержать файлы, необходимые для offline-установки продуктов, выбранных вами на этапе получения shell-скрипта от команды поддержки Arenadata.

```
$ ls -la /opt/arenadata/etools/httprepo/arenadata-repo
```

Следующий пример показывает данные, которые могут храниться в директории в случае установки ADQM.

```
total 0
drwxr-xr-x. 5 root root 46 june 15 09:00 .
drwxr-xr-x. 3 root root 28 june 15 08:59 ..
drwxr-xr-x. 3 root root 17 june 15 09:00 ADM
drwxr-xr-x. 3 root root 33 june 15 08:59 ADQM
drwxr-xr-x. 3 root root 19 june 15 09:00 zookeeper
```

Создание кластера

Дарья Барышева, Елена Дворядкина

Содержание

Назад к содержанию

Шаг 1. Получение Enterprise-версии бандла

Шаг 2. Загрузка бандла в ADCM

Шаг 3. Создание кластера на базе загруженного бандла

В ADCM **кластер** означает набор сервисов, запущенных на группе хостов.

Дистрибутивы кластеров для ADCM поставляются в виде бандлов. В общем случае бандл представляет обычный архив, описывающий логику развертывания кластера.

Шаги для добавления нового кластера приведены ниже.

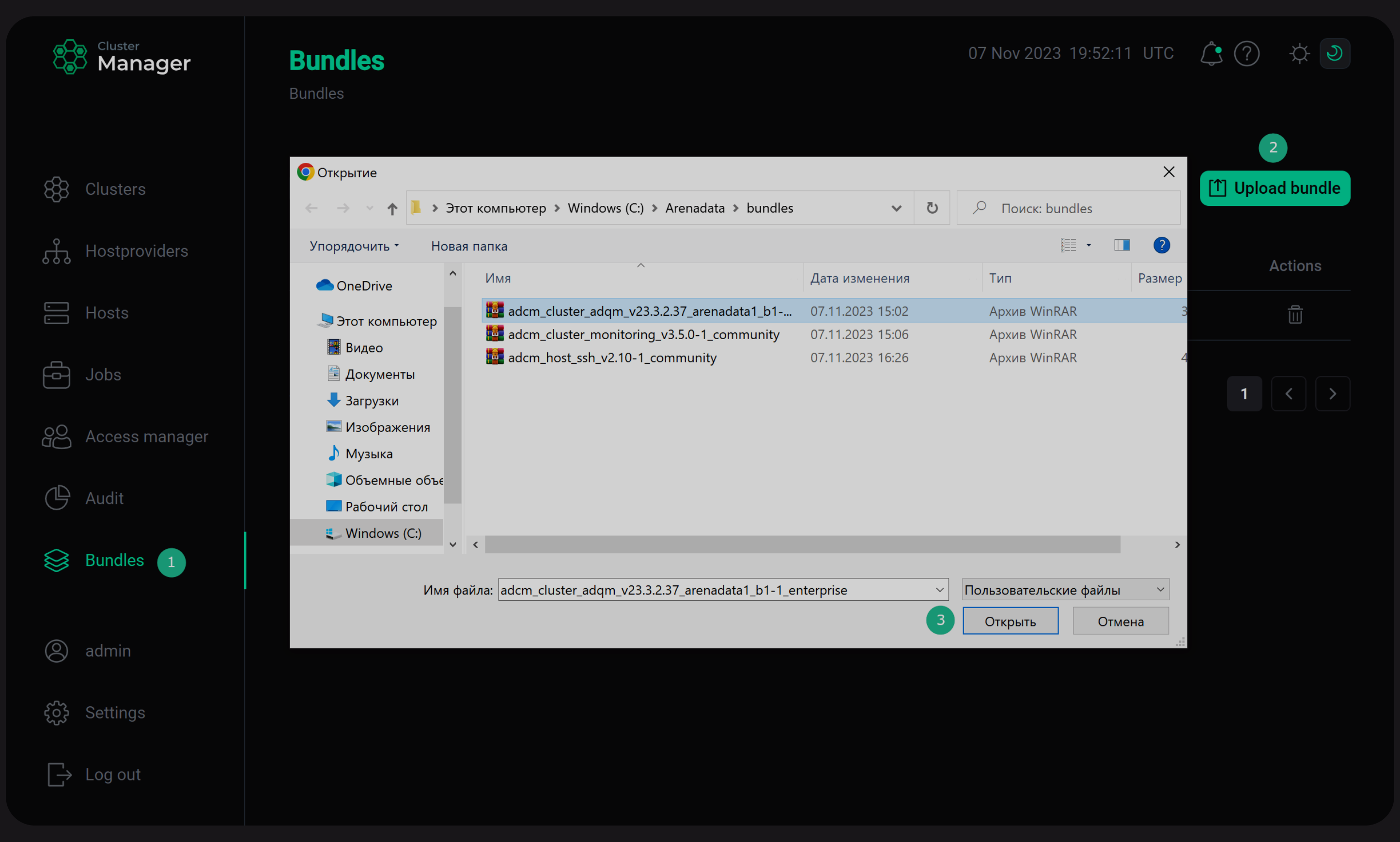
Шаг 1. Получение Enterprise-версии бандла

Пожалуйста, запросите бандл у команды поддержки Arenadata.

Шаг 2. Загрузка бандла в ADCM

Чтобы загрузить бандл в ADCM:

1. Выберите пункт ***Bundles*** в левом навигационном меню и нажмите ***Upload bundle***.
2. Выберите бандл в открывшемся диалоговом окне.



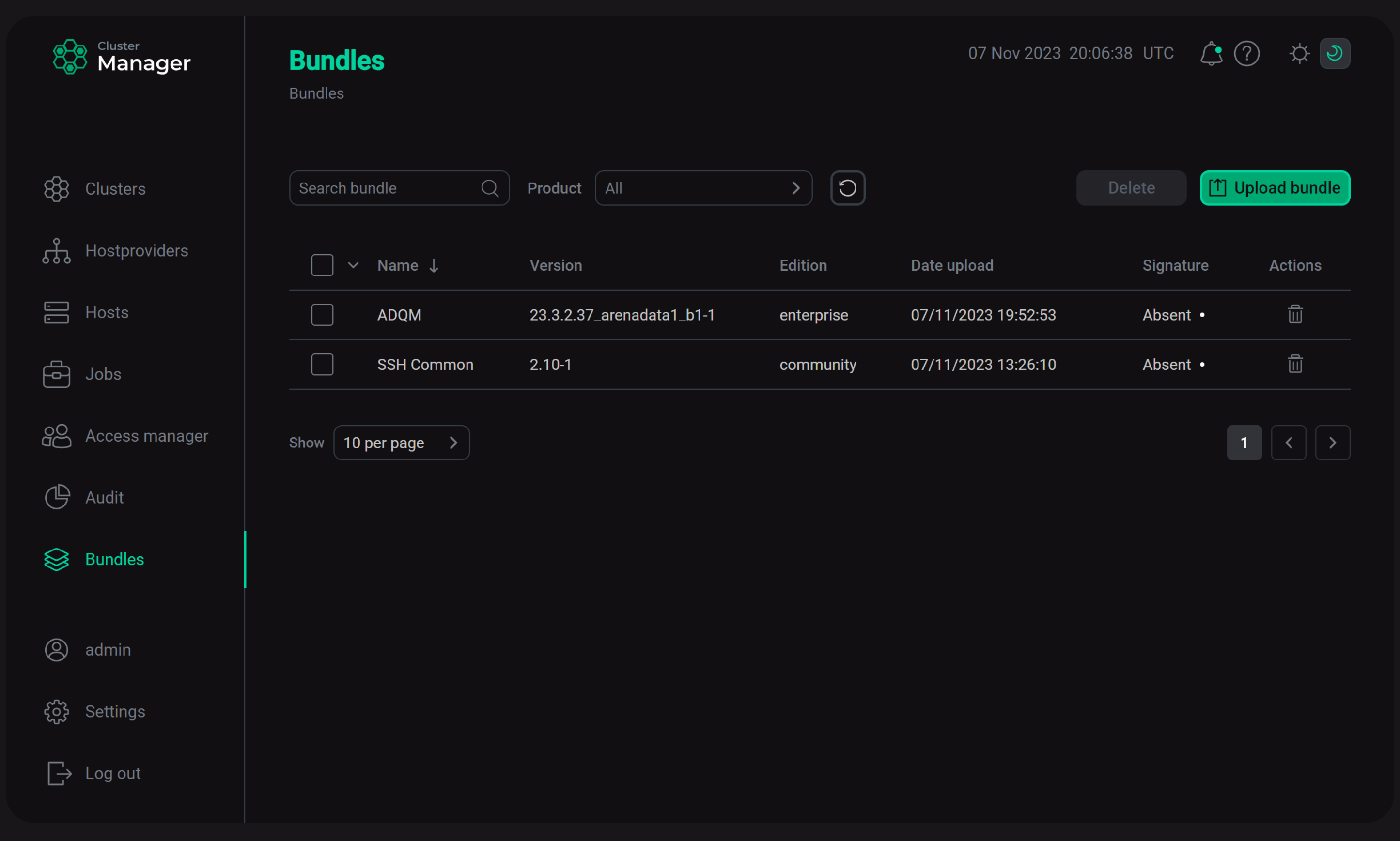
Cluster Manager
Bundles
Bundles
07 Nov 2023 19:52:11 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
1
admin
Settings
Log out
2
Upload bundle
Actions
Открытие
Этот компьютер > Windows (C:) > Arenadata > bundles
Поиск: bundles
Упорядочить
Новая папка
OneDrive
Этот компьютер
Видео
Документы
Загрузки
Изображения
Музыка
Объемные объе
Рабочий стол
Windows (C:)
Пользовательские файлы
3
Открыть
Отмена

Имя	Дата изменения	Тип
adcm_cluster_adqm_v23.3.2.37_arenadata1_b1-...	07.11.2023 15:02	Архив WinRAR
adcm_cluster_monitoring_v3.5.0-1_community	07.11.2023 15:06	Архив WinRAR
adcm_host_ssh_v2.10-1_community	07.11.2023 16:26	Архив WinRAR

Имя файла: adcm_cluster_adqm_v23.3.2.37_arenadata1_b1-1_enterprise

Переход к загрузке бандла

3. После успешной загрузки бандл отображается на странице ***Bundles***.



Cluster Manager
Bundles
Bundles
07 Nov 2023 20:06:38 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search bundle
Product
All
Delete
Upload bundle
Show
10 per page
1

Name	Version	Edition	Date upload	Signature	Actions
ADQM	23.3.2.37_arenadata1_b1-1	enterprise	07/11/2023 19:52:53	Absent •	
SSH Common	2.10-1	community	07/11/2023 13:26:10	Absent •	

Результат успешной загрузки бандла

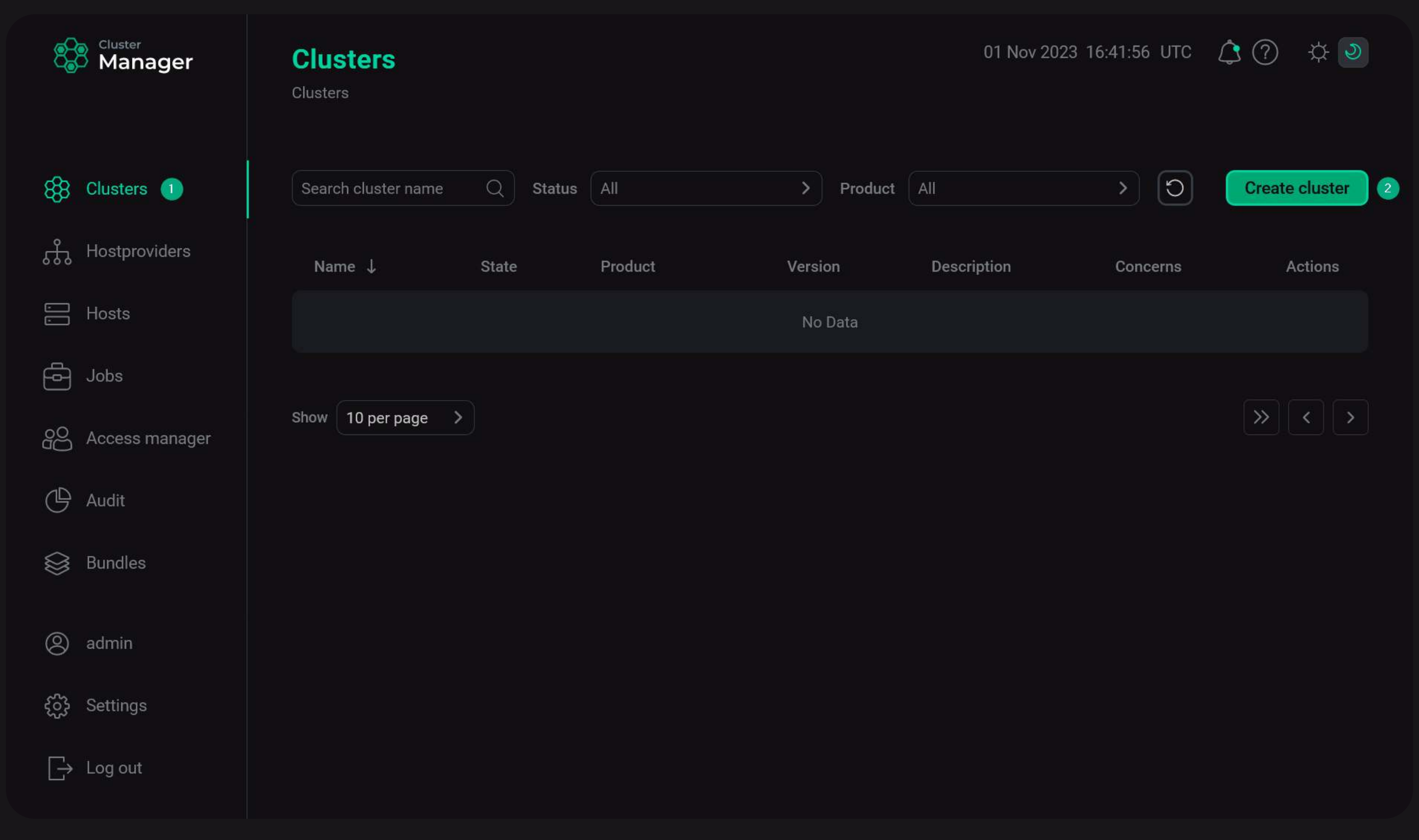
Шаг 3. Создание кластера на базе загруженного бандла

ВАЖНО

На этапе создания кластера через web-интерфейс ADCM генерируется только сам кластер. Это означает добавление записи о кластере в ADCM — сервисы на этом шаге не устанавливаются.

Для создания кластера необходимо выполнить шаги:

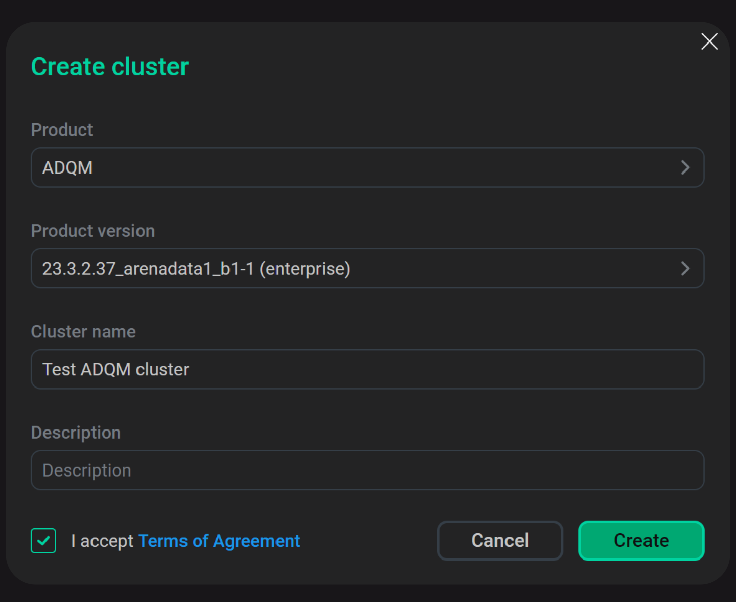
1. Выберите пункт ***Clusters*** в левом навигационном меню и нажмите ***Create cluster***.



Cluster Manager
Clusters
Clusters
01 Nov 2023 16:41:56 UTC
Clusters
1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
2
Name
State
Product
Version
Description
Concerns
Actions
No Data
Show
10 per page

Переход к созданию кластера

2. В открывшемся окне требуется описать новый кластер:
   - Выберите загруженный бандл в поле ***Product***.
   - Выберите версию бандла в поле ***Product version***. Несколько версий становятся доступны после загрузки нескольких бандлов для одного типа продуктов.
   - Введите имя кластера в поле ***Cluster name***.
   - При необходимости добавьте описание кластера в поле ***Description***.
   - Установите флаг ***I accept Terms of Agreement*** для подписания лицензионного соглашения. Флаг отображается только для бандлов версии Enterprise. Чтобы ознакомиться с текстом соглашения, перейдите по ссылке ***Terms of Agreement***. Подписать соглашение также возможно на странице ***Bundles***, нажав на соответствующий продуктовый бандл.
   - Нажмите ***Create***.



Create cluster
Cancel
Create

Product: ADQM

Product version: 23.3.2.37_arenadata1_b1-1 (enterprise)

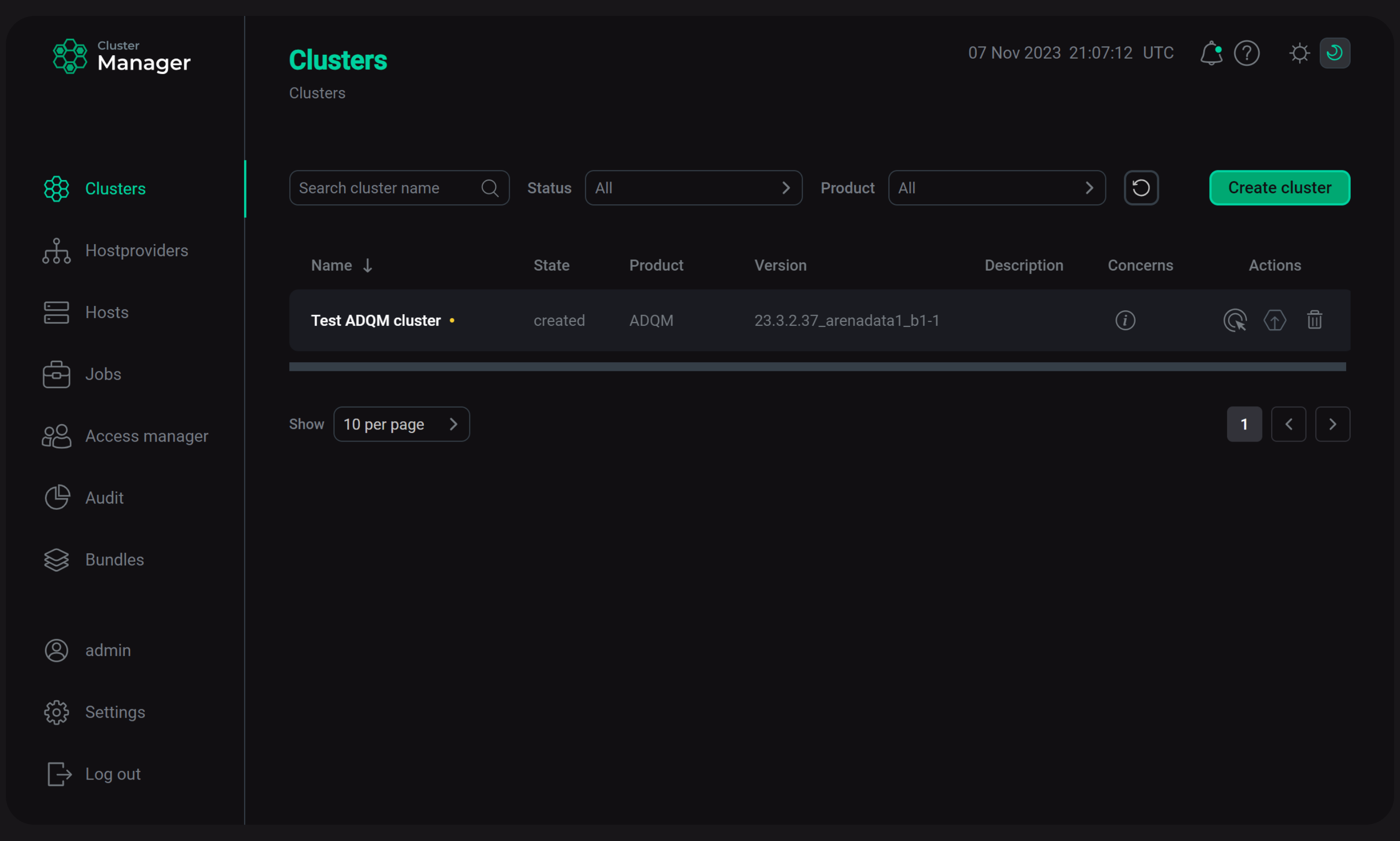
Cluster name: Test ADQM cluster

Description: Description

☑ I accept Terms of Agreement

Создание кластера

3. После успешного добавления кластер отображается на странице ***Clusters***.



Cluster Manager
Clusters
Clusters
07 Nov 2023 21:07:12 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster •	created	ADQM	23.3.2.37_arenadata1_b1-1			

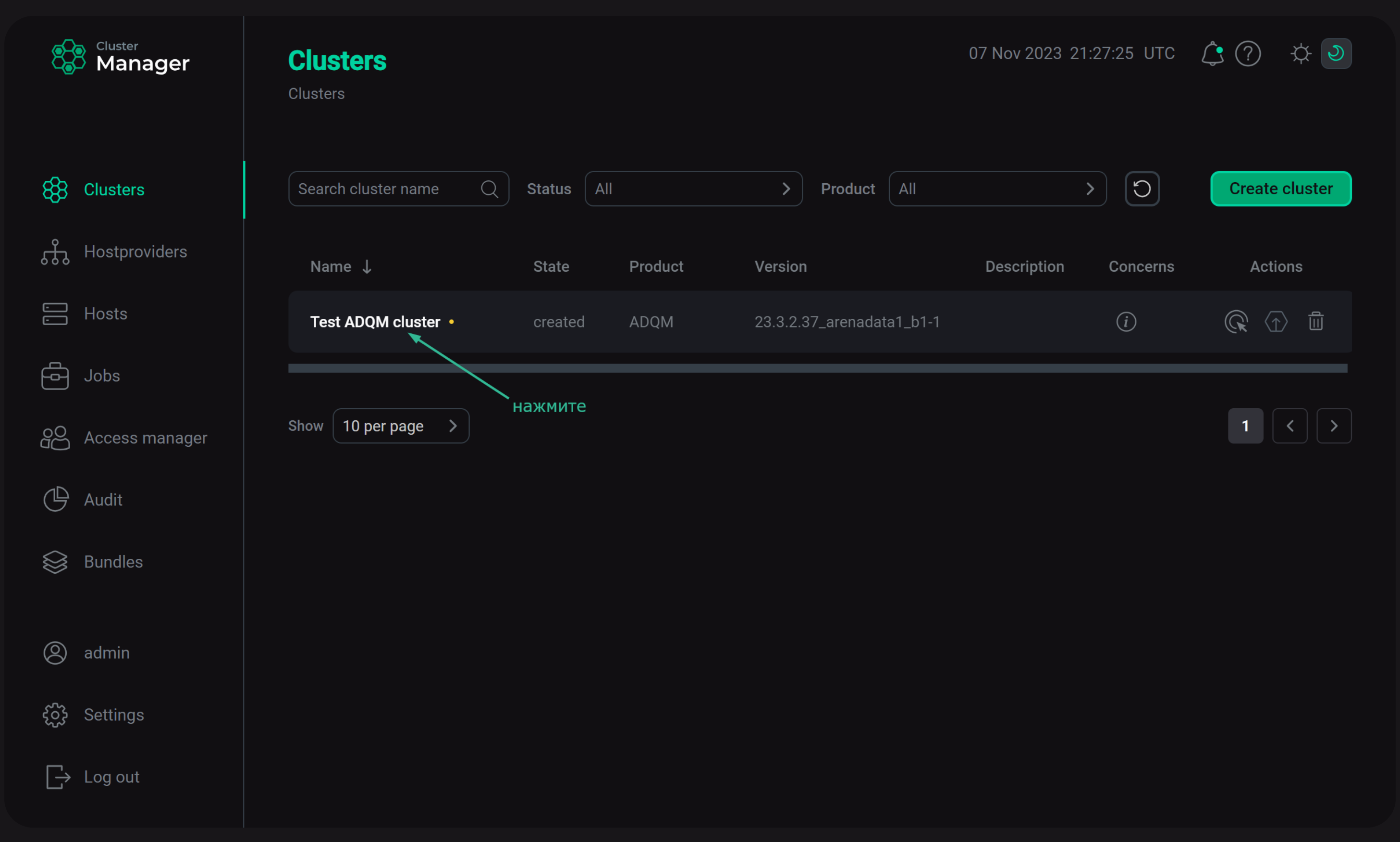
Результат успешного добавления кластера

Добавление сервисов

Дарья Барышева, Елена Дворядкина

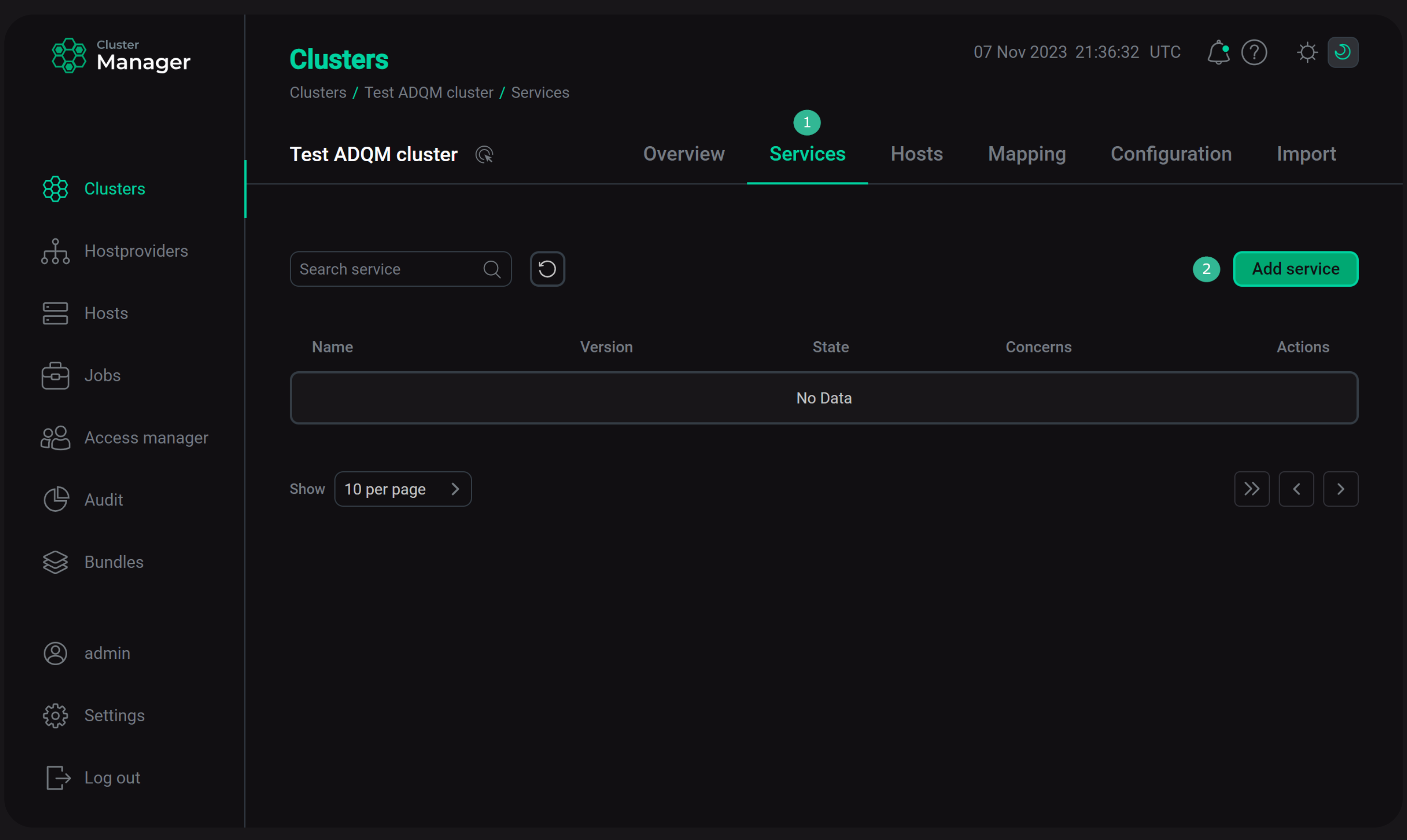
В ADCM **сервис** означает программное обеспечение, выполняющее некоторую функцию. Примеры сервисов в кластерах ADQM: ADQMDB, Monitoring Clients, Zookeeper и так далее. Для добавления сервисов в кластер:

1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test ADQM cluster
created
ADQM
23.3.2.37_arenadata1_b1-1
нажмите
Show
10 per page
1

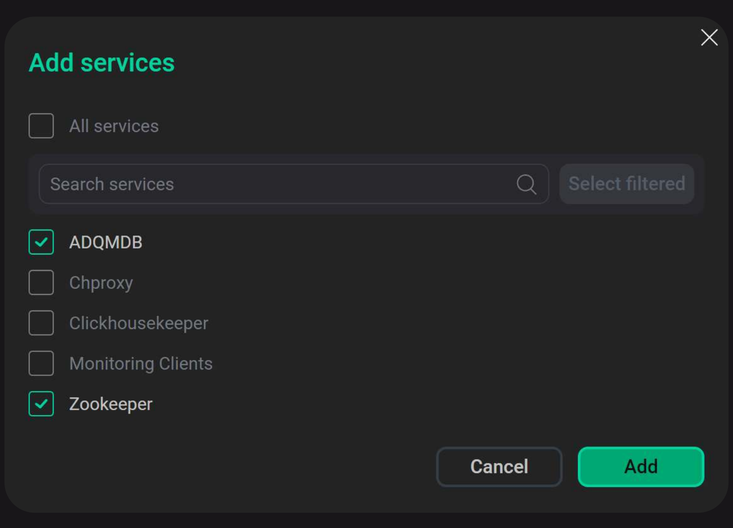
Выбор кластера

2. Откройте вкладку ***Services*** на странице кластера и нажмите ***Add service***.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
07 Nov 2023 21:36:32 UTC
Test ADQM cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page

Переход к добавлению сервисов

3. В открывшемся окне выберите сервисы для добавления в кластер и нажмите ***Add***.


Add services
All services
Search services
Select filtered
Cancel
Add

☐ All services
☑ ADQMDB
☐ Chproxy
☐ Clickhousekeeper
☐ Monitoring Clients
☑ Zookeeper

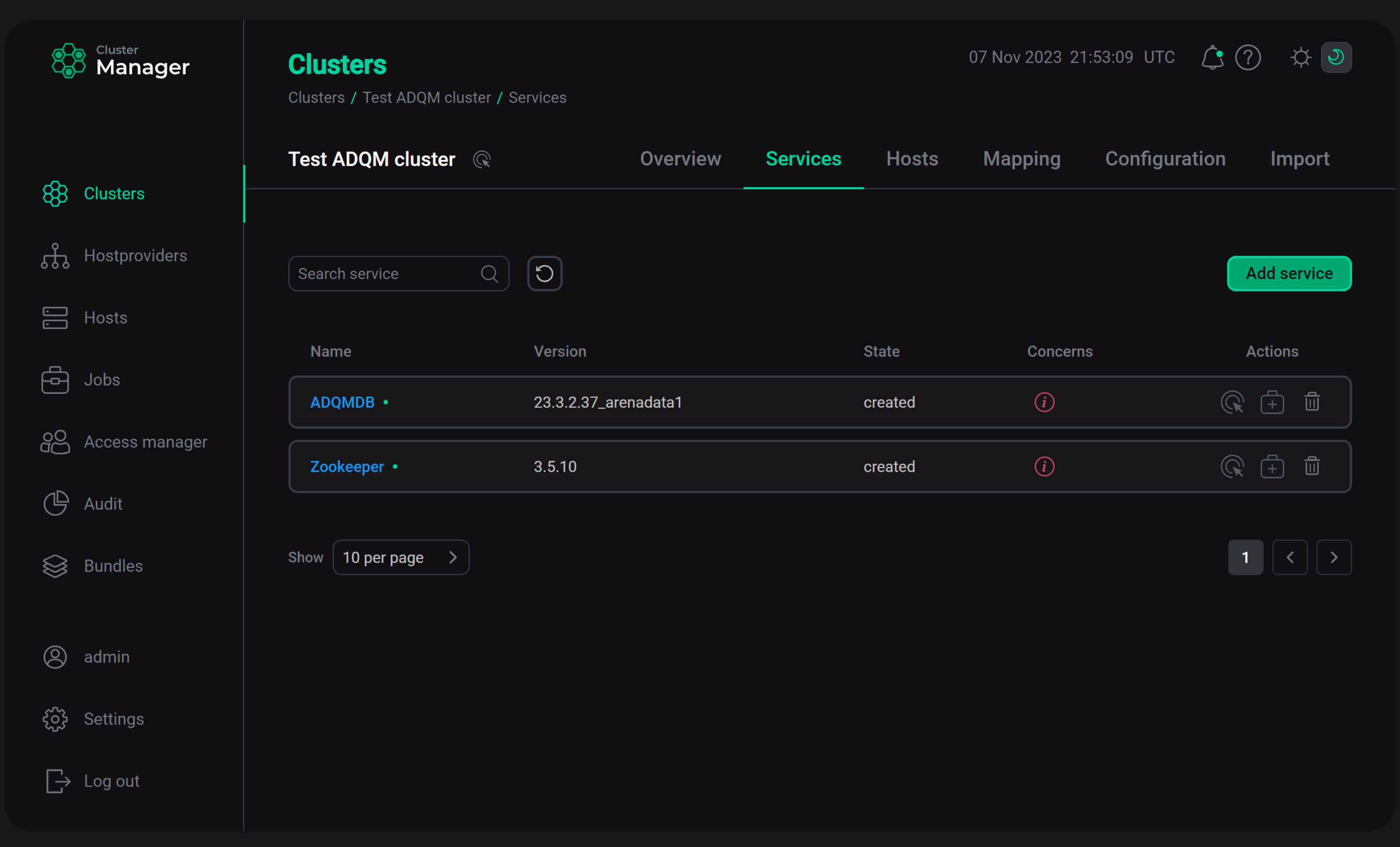
Выбор сервисов

Краткое описание доступных сервисов приведено ниже.

Сервисы, которые могут быть добавлены в кластер ADQM

Сервис	Обязательность	Назначение	Доступность
ADQMDB	Да	ClickHouse — это колоночно-ориентированная система управления базами данных (СУБД) для аналитической обработки запросов в режиме онлайн (OLAP)	Community/Enterprise
Zookeeper	Нет	Централизованный сервис для ведения информации о конфигурации, именования, обеспечения распределенной синхронизации и предоставления групповых услуг	Community/Enterprise
Clickhousekeeper	Нет	Сервис координации для репликации данных и выполнения распределенных DDL-запросов (альтернатива ZooKeeper)	Community/Enterprise
Monitoring Clients	Нет	Агенты, отправляющие информацию о хостах кластера в систему мониторинга, основанную на сервисах Diamond, Graphite и Grafana. Сервис мониторинга необходимо добавлять в кластер после создания и настройки кластера мониторинга	Community/Enterprise
Monitoring	Нет	Сервис для сбора и хранения метрик мониторинга кластера ADQM на базе системы мониторинга Prometheus и сервиса Grafana	Community/Enterprise
Chproxy	Нет	HTTP-прокси и балансировщик нагрузки для базы данных ClickHouse	Enterprise

4. В результате успешно добавленные сервисы отображаются на вкладке ***Services***.


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
07 Nov 2023 21:53:09 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Name
Version
State
Concerns
Actions
Show
10 per page
1

Name	Version	State
ADQMDB	23.3.2.37_arenadata1	created
Zookeeper	3.5.10	created

Результат успешного добавления сервисов в кластер

ПРИМЕЧАНИЕ

Опциональные сервисы могут быть добавлены в кластер позднее. Процесс добавления сервисов в уже развернутый кластер не отличается от описанного выше.

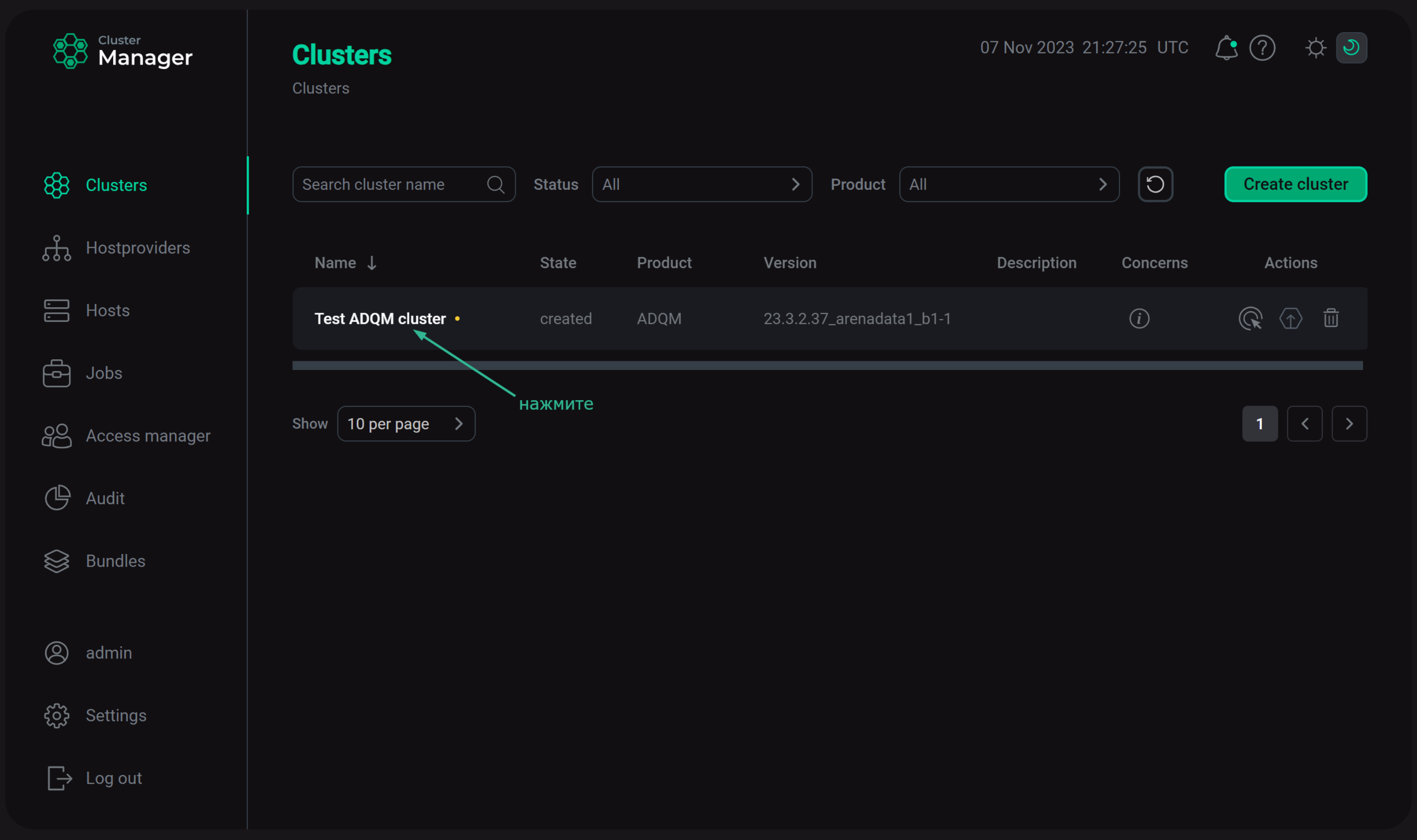
Добавление хостов в кластер

Дарья Барышева, Елена Дворядкина

Назад к содержанию

Для добавления хостов в кластер:

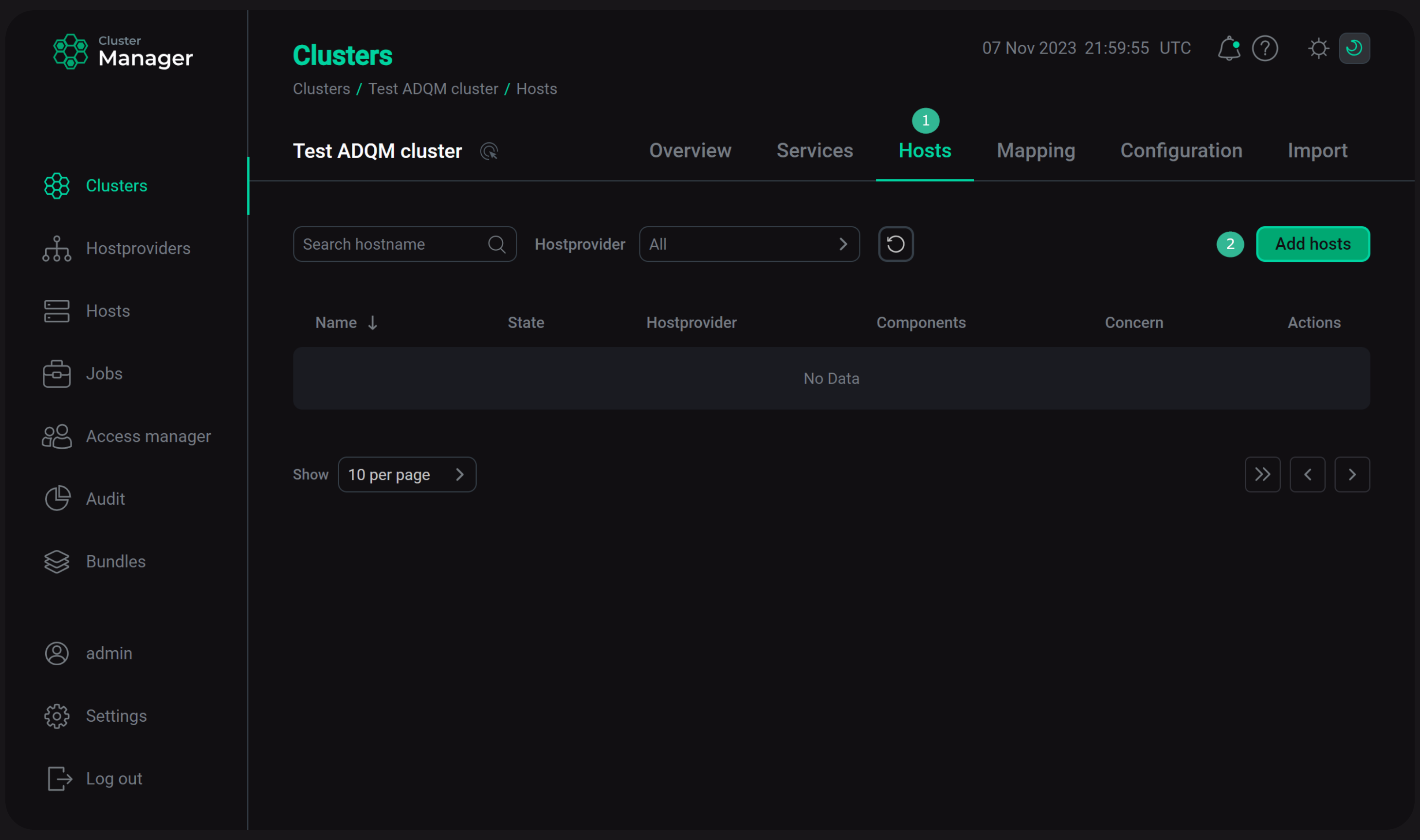
1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Search cluster name
Status
All
Product
All
Create cluster
нажмите
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

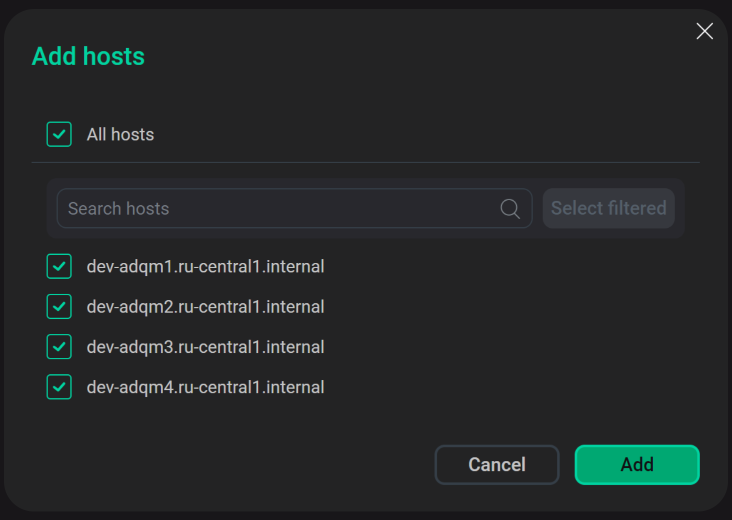
Выбор кластера

2. Откройте вкладку ***Hosts*** на странице кластера и нажмите ***Add hosts***.


Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Hosts
07 Nov 2023 21:59:55 UTC
Test ADQM cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
2
Add hosts
Name
State
Hostprovider
Components
Concern
Actions
No Data
Show
10 per page

Переход к добавлению хостов

3. В открывшемся окне выберите хосты для добавления в кластер и нажмите ***Add***.

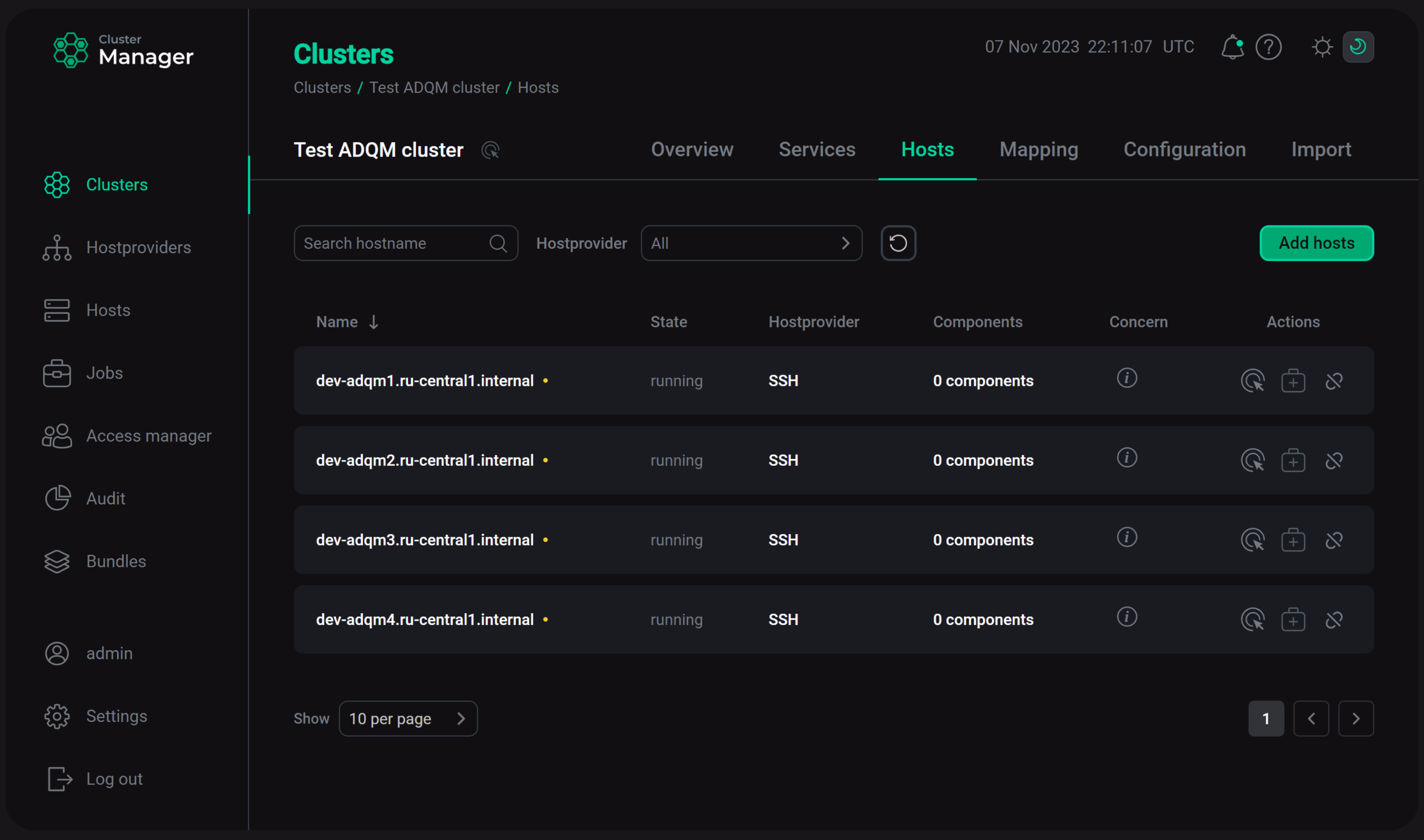

Add hosts
Search hosts
Select filtered
Cancel
Add

☑ All hosts

☑ dev-adqm1.ru-central1.internal
☑ dev-adqm2.ru-central1.internal
☑ dev-adqm3.ru-central1.internal
☑ dev-adqm4.ru-central1.internal

Выбор хостов

4. В результате успешно добавленные хосты отображаются на вкладке ***Hosts***.


Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Hosts
07 Nov 2023 22:11:07 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Add hosts
Show
10 per page
1

Name	State	Hostprovider	Components	Concern	Actions
dev-adqm1.ru-central1.internal	running	SSH	0 components		
dev-adqm2.ru-central1.internal	running	SSH	0 components		
dev-adqm3.ru-central1.internal	running	SSH	0 components		
dev-adqm4.ru-central1.internal	running	SSH	0 components		

Результат успешного добавления хостов в кластер

Добавление компонентов

Дарья Барышева, Елена Дворядкина

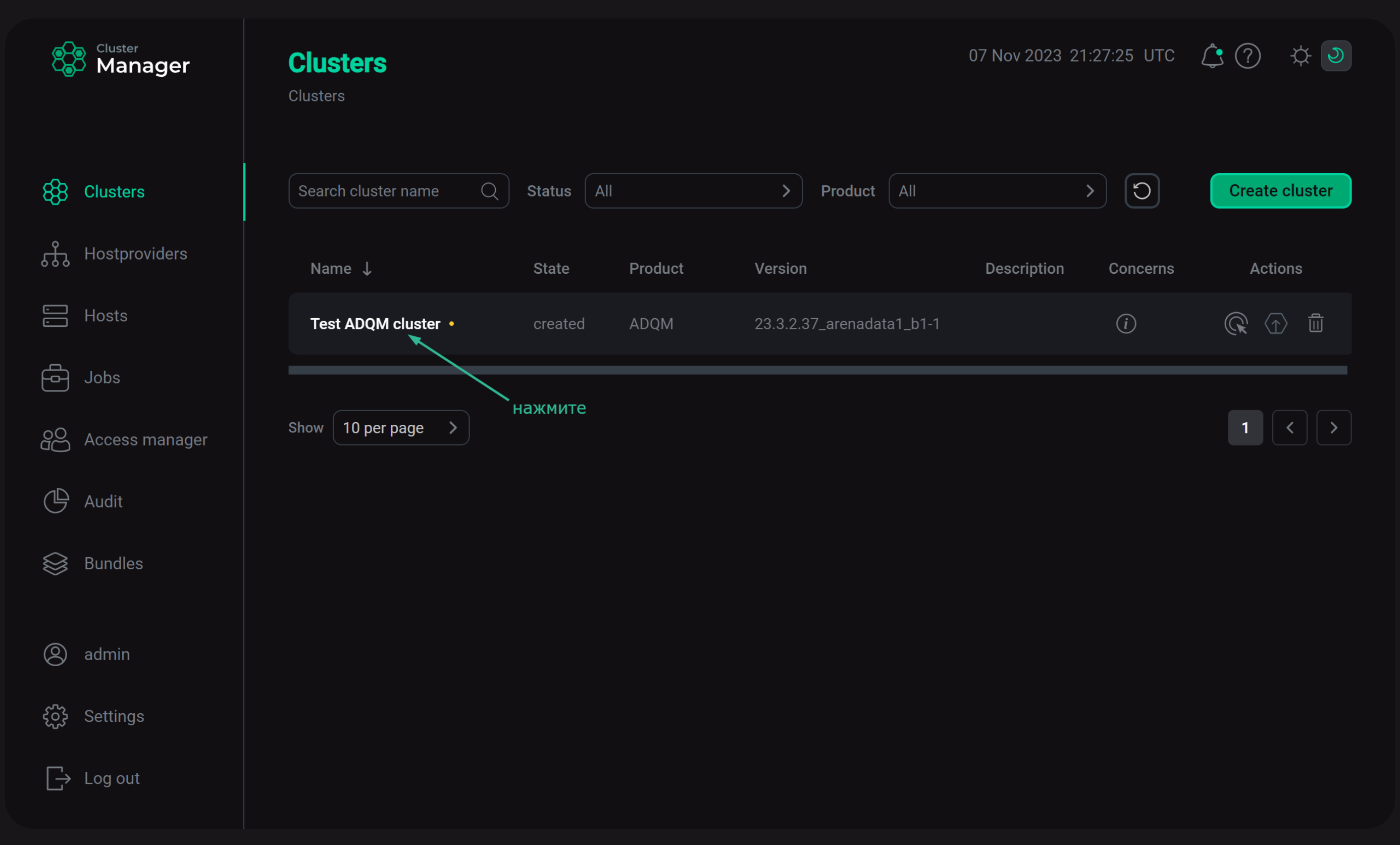
Назад к содержанию

В ADCM **компонент** означает часть сервиса, которая должна быть развернута на одном или нескольких хостах кластера.

> **ВАЖНО**
>
> Изначально на хостах кластера нет ни одного компонента — распределение компонентов между хостами должно быть проведено вручную.

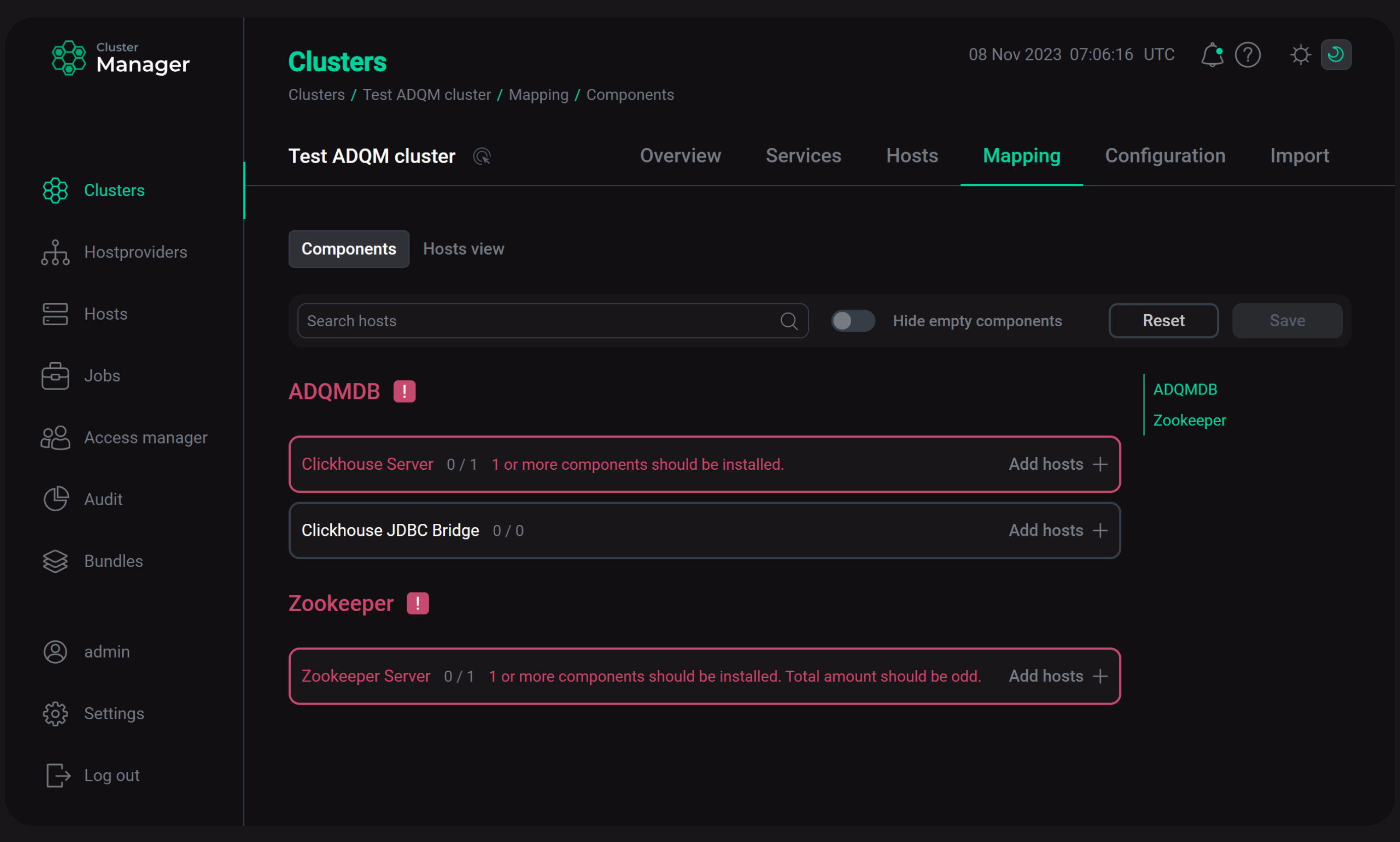
Для добавления компонентов на хосты кластера:

1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
07 Nov 2023 21:27:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test ADQM cluster
created
ADQM
23.3.2.37_arenadata1_b1-1
нажмите
Show 10 per page

Выбор кластера

2. Откройте вкладку ***Mapping*** на странице кластера.

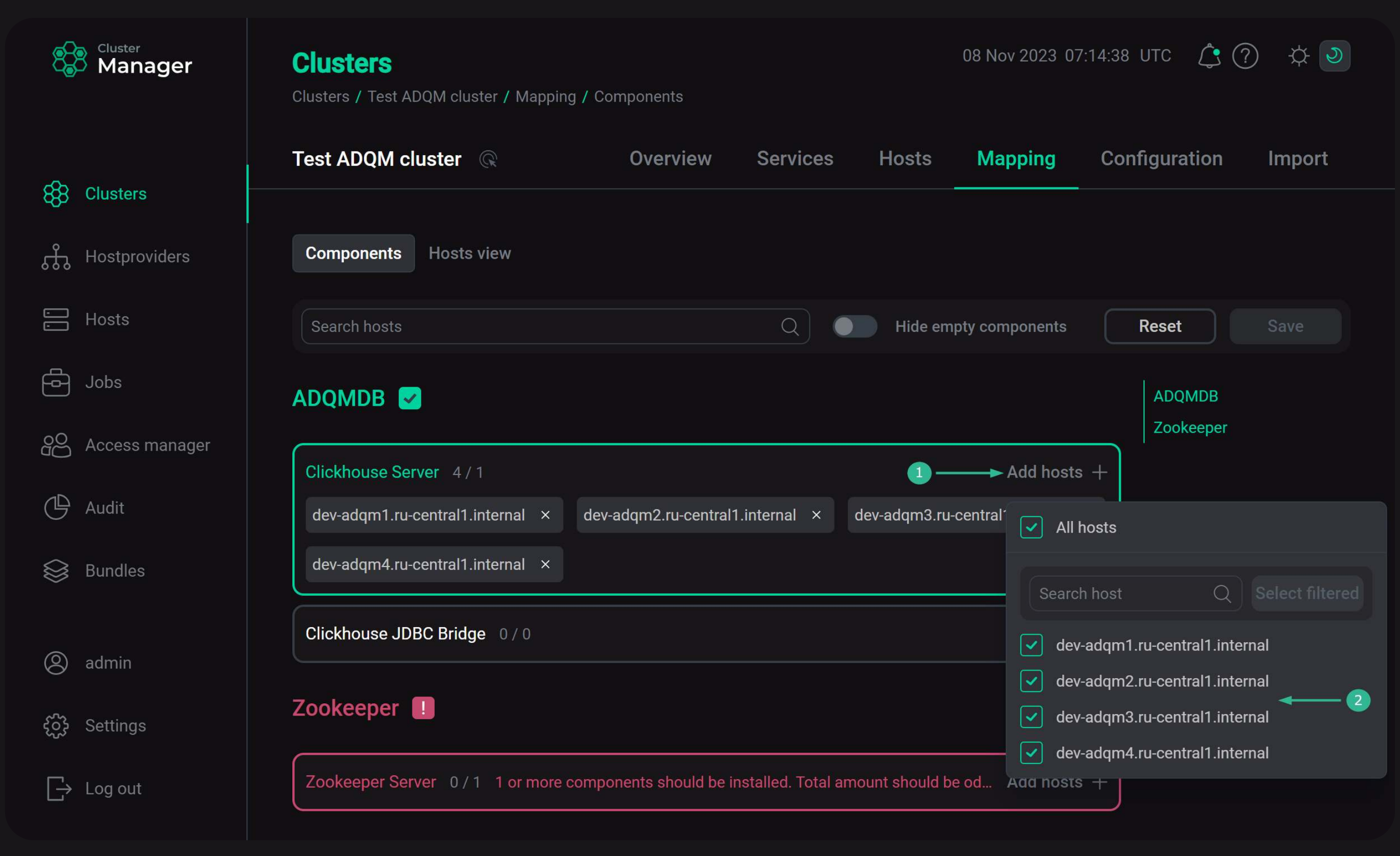

Cluster Manager
Clusters
Clusters / Test ADQM cluster / Mapping / Components
08 Nov 2023 07:06:16 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
ADQMDB
Clickhouse Server 0 / 1
Add hosts
Clickhouse JDBC Bridge 0 / 0
Zookeeper
Zookeeper Server 0 / 1

Clickhouse Server 0 / 1 1 or more components should be installed.

Zookeeper Server 0 / 1 1 or more components should be installed. Total amount should be odd.

Переход к распределению сервисных компонентов

3. Распределите все компоненты между хостами кластера. Для назначения хостов компоненту нажмите ***Add hosts*** и выберите хосты в открывшемся окне.

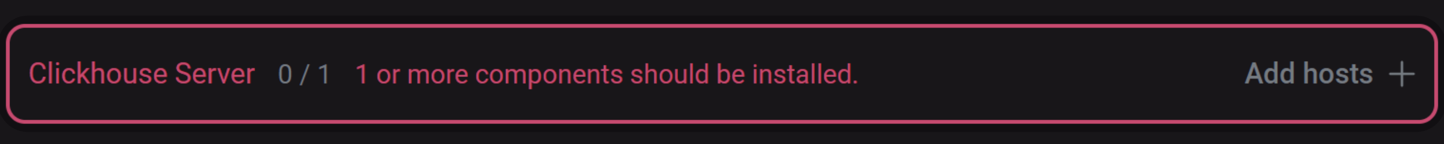

Cluster Manager
Clusters
Clusters / Test ADQM cluster / Mapping / Components
08 Nov 2023 07:14:38 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
ADQMDB
Clickhouse Server 4 / 1
1
Add hosts
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
Clickhouse JDBC Bridge 0 / 0
Zookeeper
All hosts
Search host
Select filtered
2

Zookeeper Server 0 / 1 1 or more components should be installed. Total amount should be od...

Выбор хостов для компонента

Обязательные компоненты

Каждый сервис может иметь обязательные и опциональные компоненты. Обязательные подсвечиваются красным цветом. Также в интерфейсе ADCM для них выводится информация о том, в каком количестве необходимо установить каждый компонент.


Clickhouse Server 0 / 1 1 or more components should be installed. Add hosts +

Обязательные компоненты

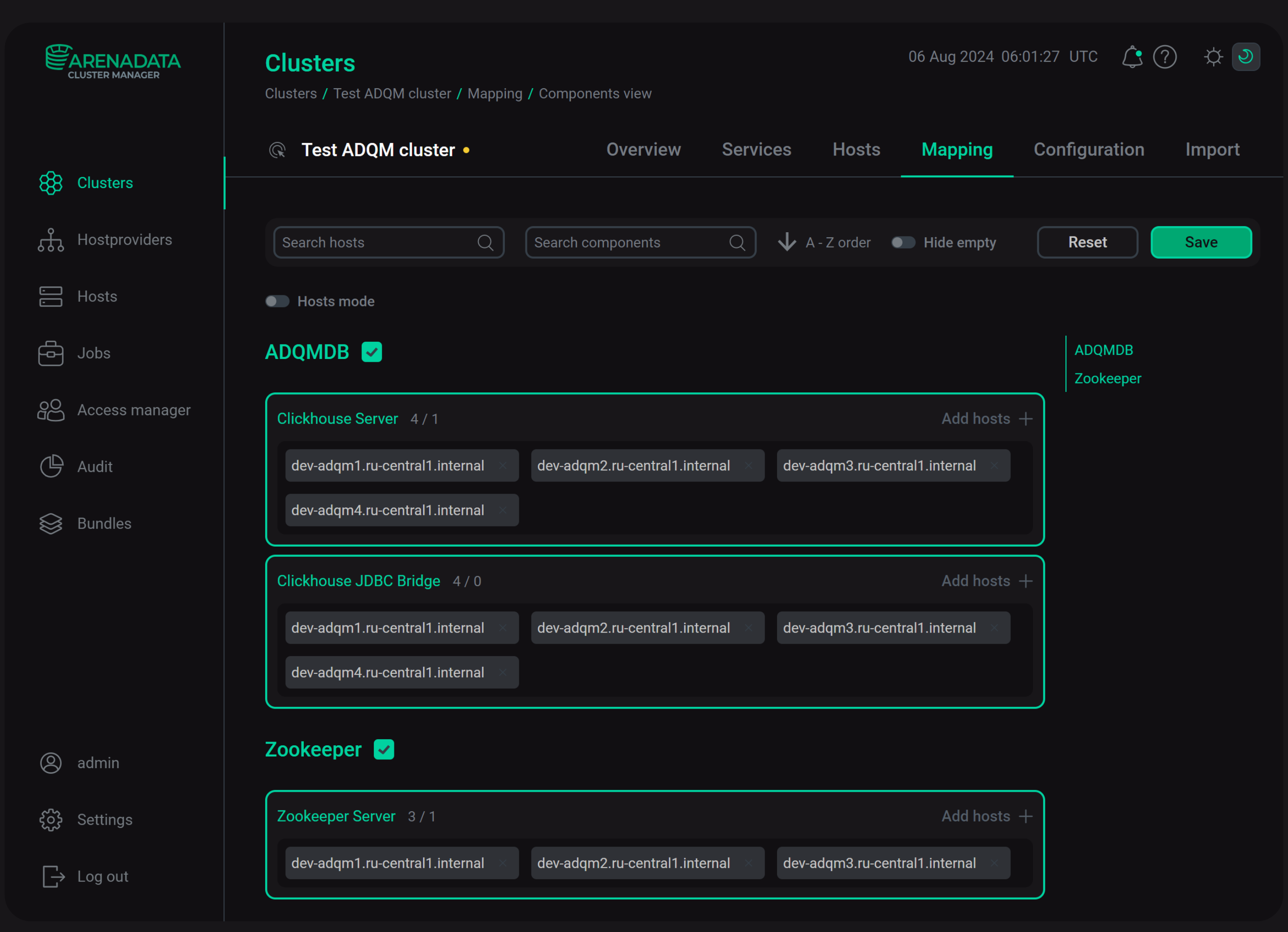
> **ВНИМАНИЕ**
>
> Без назначения нужного числа хостов обязательным компонентам сохранение общей схемы распределения невозможно.

Типовая схема распределения компонентов в кластере ADQM приведена в таблице ниже.

Распределение компонентов ADQM

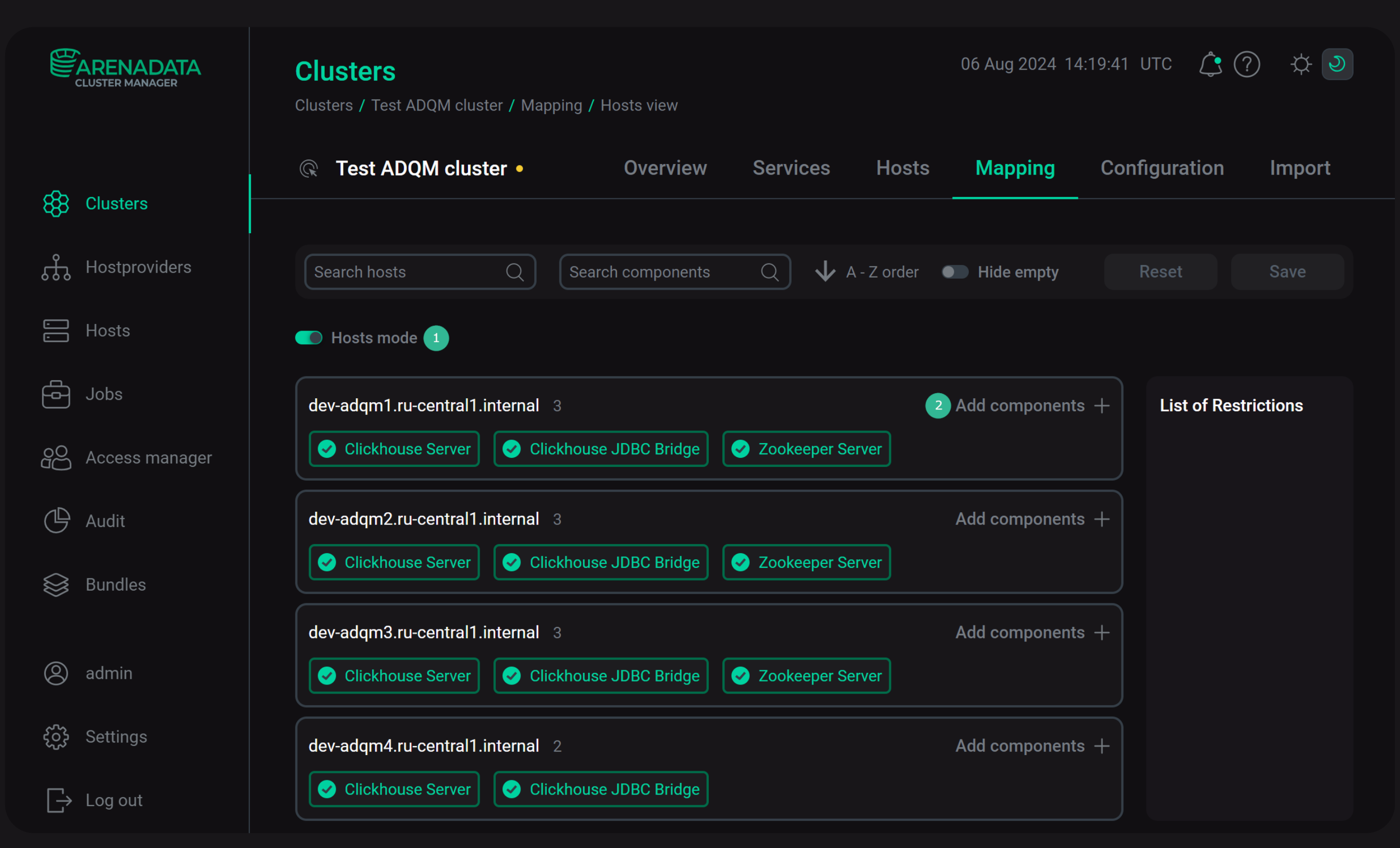
Хост	Сервис	Компонент	Примечание
Любой хост	ADQMDB	Clickhouse Server	Необходимо установить один или несколько компонентов
Все хосты с Clickhouse Server	ADQMDB	Clickhouse JDBC Bridge	При установке этого компонента (опционально) необходимо добавить его на все хосты кластера, где установлен компонент Clickhouse Server
Любой хост	Zookeeper	Zookeeper Server	Должен быть установлен один или несколько компонентов. Суммарное количество должно быть нечетным
Любой хост	Clickhousekeeper	Clickhouse Keeper Server	Должен быть установлен один или несколько компонентов. Суммарное количество должно быть нечетным
Любой хост	Chproxy	Chproxy Proxy Server	Необходимо установить один или несколько компонентов

4. После завершения распределения компонентов нажмите ***Save***.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Mapping / Components view
06 Aug 2024 06:01:27 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
ADQMDB
Clickhouse Server 4 / 1
Add hosts
Clickhouse JDBC Bridge 4 / 0
Zookeeper
Zookeeper Server 3 / 1
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal

Сохранение распределения компонентов

Начиная с версии 2.2.0 ADCM поддерживает альтернативный способ сопоставления — по хостам. Для этого переведите в активное положение переключатель ***Hosts mode*** и распределите компоненты между хостами путем нажатия на кнопку ***Add components*** напротив каждого хоста.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Mapping / Hosts view
06 Aug 2024 14:19:41 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
1
dev-adqm1.ru-central1.internal 3
2
Add components
List of Restrictions
dev-adqm2.ru-central1.internal 3
dev-adqm3.ru-central1.internal 3
dev-adqm4.ru-central1.internal 2
Clickhouse Server
Clickhouse JDBC Bridge
Zookeeper Server

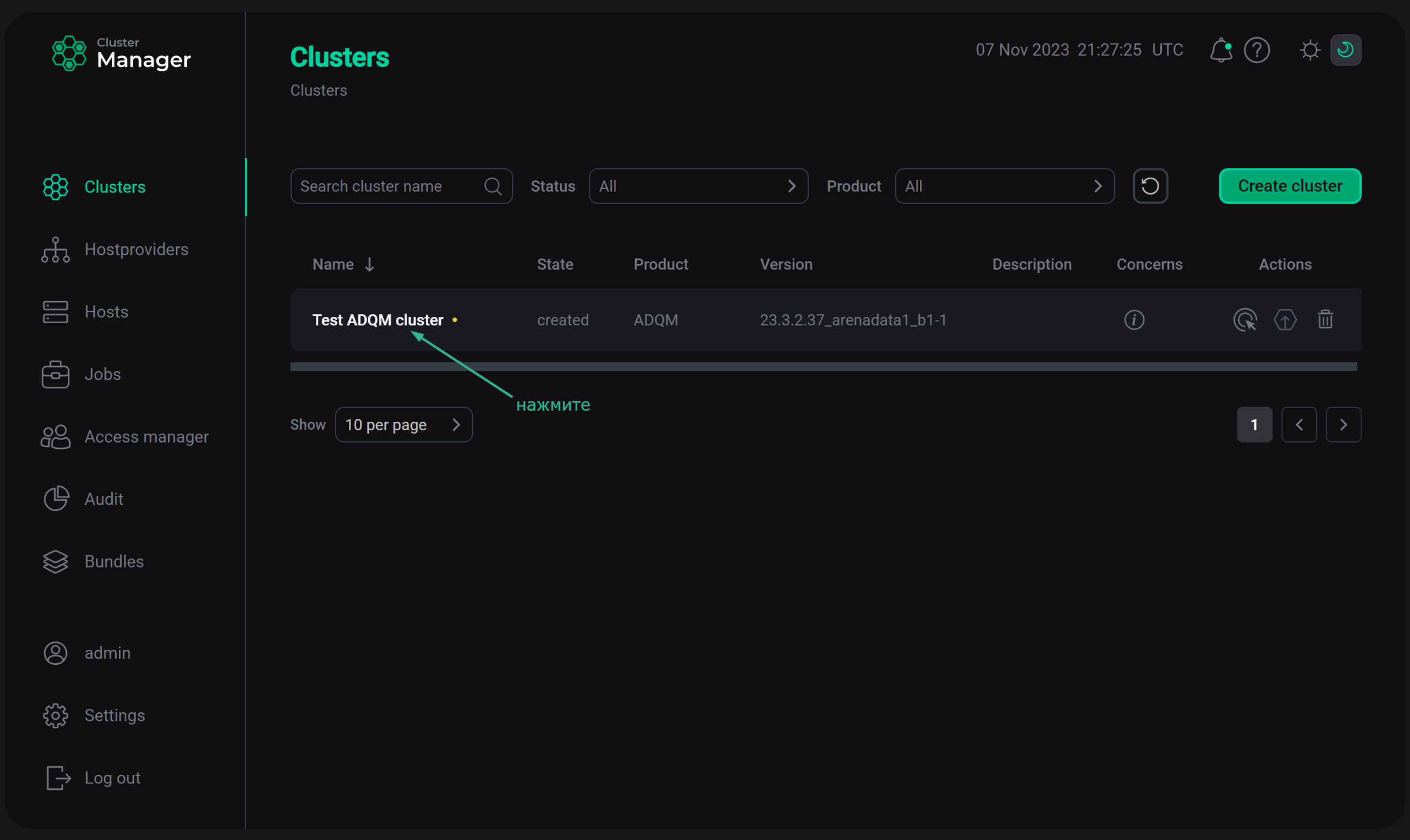
Распределение по хостам

Настройка сервисов

Дарья Барышева, Елена Дворядкина

Назад к содержанию

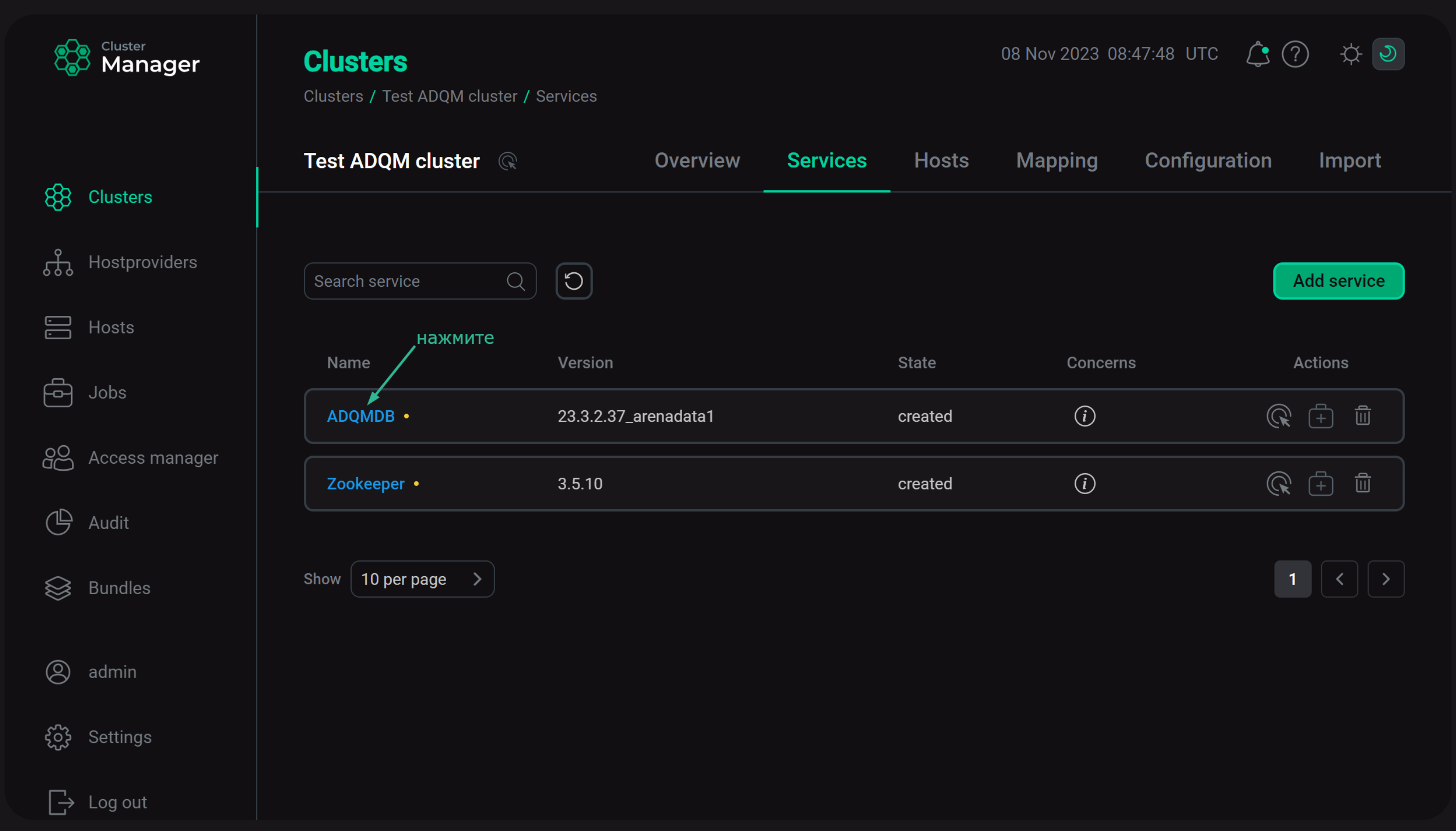
1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Search cluster name
Status
All
Product
All
Create cluster
нажмите
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

Выбор кластера

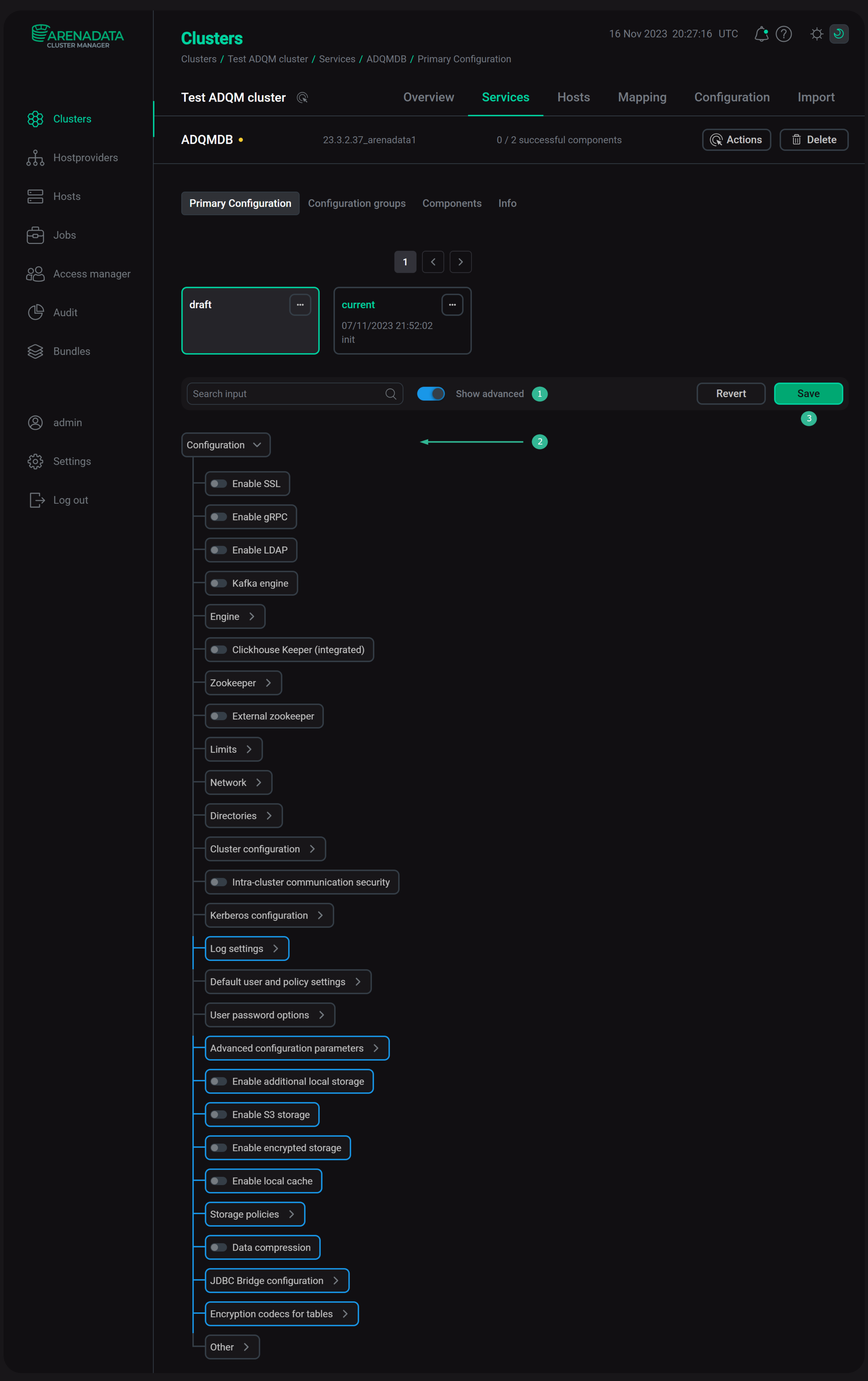
2. Откройте вкладку ***Services*** на странице кластера. Нажмите на имя сервиса в столбце ***Name***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Services
08 Nov 2023 08:47:48 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
нажмите
Show
10 per page
1

Name	Version	State	Concerns	Actions
ADQMDB	23.3.2.37_arenadata1	created		
Zookeeper	3.5.10	created		

Переход к настройке сервиса

3. В открывшемся окне:
   - Откройте вкладку ***Primary Configuration***.
   - Переведите в активное состояние переключатель ***Show advanced***.
   - Заполните конфигурационные параметры выбранного сервиса. Поля, подсвеченные красным, обязательны для заполнения.
   - Нажмите ***Save***.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary Configuration
16 Nov 2023 20:27:16 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
23.3.2.37_arenadata1
0 / 2 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1
draft
current
07/11/2023 21:52:02
init
Search input
Show advanced
1
Revert
Save
3
2
Configuration
Enable SSL
Enable gRPC
Enable LDAP
Kafka engine
Engine
Clickhouse Keeper (integrated)
Zookeeper
External zookeeper
Limits
Network
Directories
Cluster configuration
Intra-cluster communication security
Kerberos configuration
Log settings
Default user and policy settings
User password options
Advanced configuration parameters
Enable additional local storage
Enable S3 storage
Enable encrypted storage
Enable local cache
Storage policies
Data compression
JDBC Bridge configuration
Encryption codecs for tables
Other

Настройка сервиса

ПРИМЕЧАНИЕ

Более подробную информацию по всем конфигурационным параметрам можно получить в статье Конфигурационные параметры.

Настройка кластера

Дарья Барышева, Елена Дворядкина

Содержание

Назад к содержанию

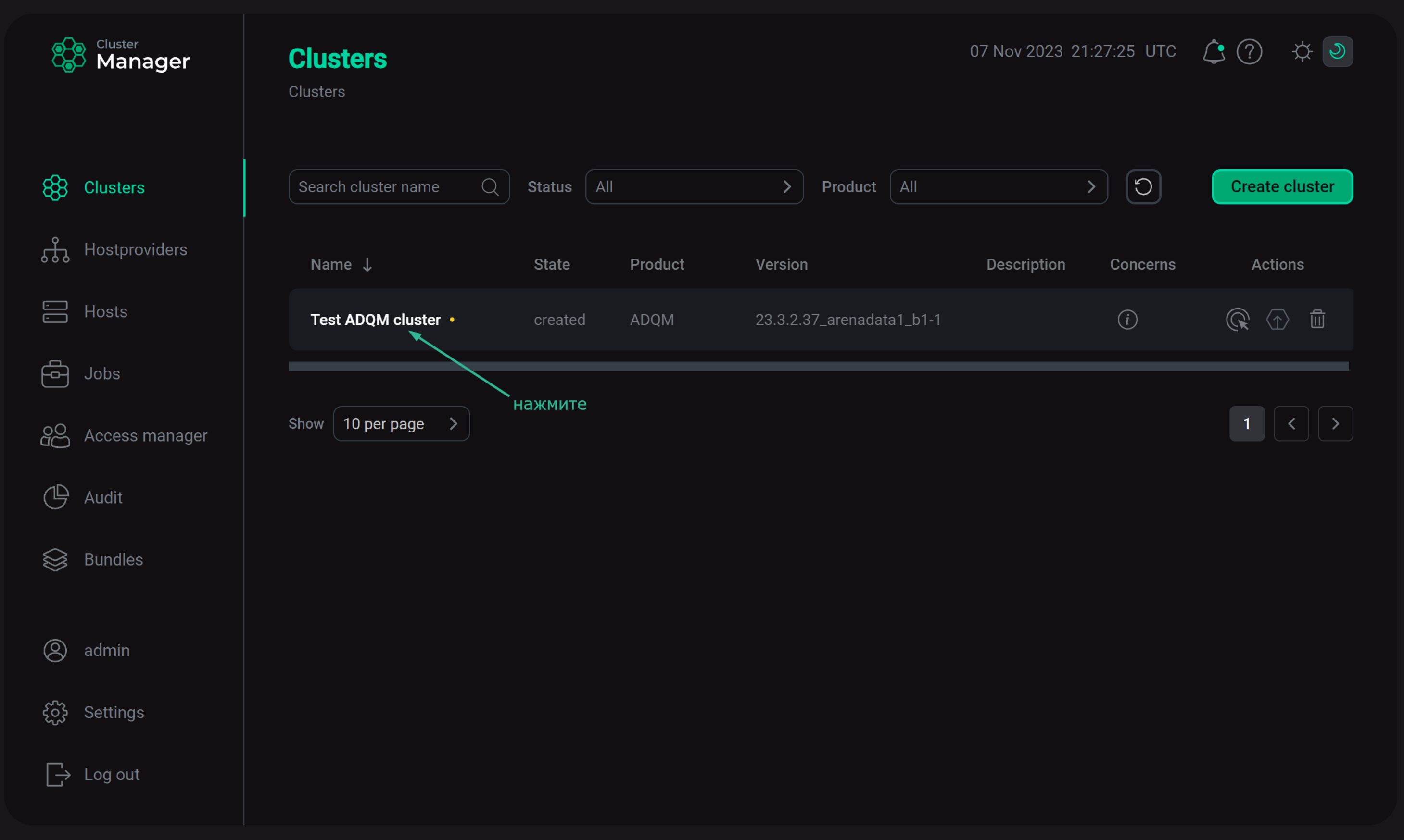
Primary configuration
Configuration groups
Ansible settings

ВАЖНО

В большинстве случаев кастомная настройка кластера не требуется — значения параметров можно оставить по умолчанию.

После добавления нового кластера можно выполнить его настройку. Для этого выполните шаги:

1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
07 Nov 2023 21:27:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test ADQM cluster
created
ADQM
23.3.2.37_arenadata1_b1-1
нажмите
Show
10 per page
1

Выбор кластера

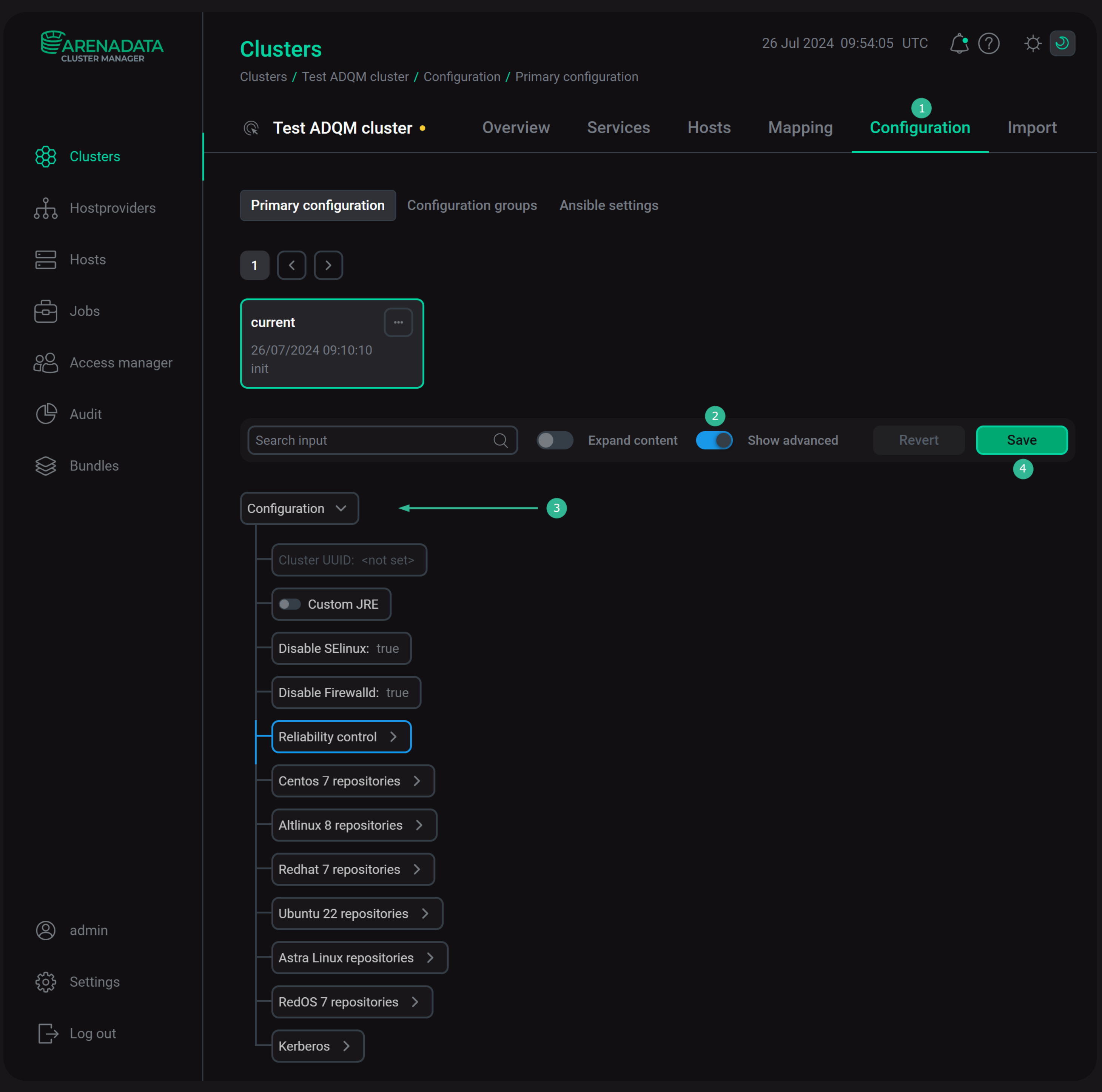
2. Откройте вкладку ***Configuration*** на странице кластера. Вкладка ***Configuration*** состоит из трех подразделов: ***Primary configuration***, ***Configuration groups***, ***Ansible settings***.
3. Заполните необходимые конфигурационные параметры на нужной вкладке и нажмите ***Save***.

Primary configuration

Вкладка ***Configuration*** → ***Primary configuration*** содержит основные конфигурационные параметры кластера.

Доступные для редактирования конфигурационные параметры кластера ADQM перечислены ниже:

- ***Custom JRE*** — переключатель, определяющий необходимость использования кастомного пути к Java для сервисов кластера ADQM. При активации переключателя становится доступно для заполнения поле ***JAVA_HOME***, в котором требуется указать путь к работающей Java. Никакие системные пакеты JRE при этом устанавливаться не будут.
- ***Disable SElinux*** — отключает SElinux на хостах кластера. Параметр доступен начиная с версии ADQM 22.8.4.7.1.b1.
- ***Disable Firewalld*** — отключает firewalld на хостах кластера. Параметр доступен начиная с версии ADQM 22.8.4.7.1.b1.
- ***Reliability control*** — секция доступна при включенной опции ***Show advanced***. Содержит следующие настройки:
  - ***retries*** — количество попыток для операций в кластере (по умолчанию `3`);
  - ***delay*** — задержка между попытками в секундах (по умолчанию `10`);
  - ***timeout*** — тайм-аут проверки сетевых портов в секундах (по умолчанию `60`).
- Пути к репозиториям, которые будут использованы в процессе установки в зависимости от операционной системы кластера (Centos, Alt Linux, Red Hat, Ubuntu, Astra Linux или РЕД ОС):
  - ***monitoring*** — репозиторий для установки мониторинга;
  - ***zookeeper_repo*** — репозиторий для установки Arenadata Zookeeper;
  - ***ADQM*** — репозиторий для установки ADQM;
  - ***prometheus_repo*** — репозиторий для установки Prometheus;
  - ***ADQM_arenadata_enterprise*** — репозиторий для установки Enterprise-версии ADQM.
- ***Kerberos*** — конфигурационные параметры для аутентификации по протоколу Kerberos (некоторые параметры становятся видимыми только при включенной опции ***Show advanced***).


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Configuration / Primary configuration
26 Jul 2024 09:54:05 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
1
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
Ansible settings
1
current
26/07/2024 09:10:10
init
Search input
Expand content
2
Show advanced
Revert
Save
4
Configuration
3
Cluster UUID: <not set>
Custom JRE
Disable SElinux: true
Disable Firewalld: true
Reliability control
Centos 7 repositories
Altlinux 8 repositories
Redhat 7 repositories
Ubuntu 22 repositories
Astra Linux repositories
RedOS 7 repositories
Kerberos

Конфигурирование кластера

Configuration groups

Вкладка ***Configuration*** → ***Configuration groups*** предназначена для настройки конфиг-групп кластера.

Ansible settings

Вкладка ***Configuration*** → ***Ansible settings*** предназначена для ввода конфигурационных настроек Ansible на уровне кластера. Вкладка доступна начиная с версии ADCM 2.2.0.

Параметры Ansible

Имя	Описание	По умолчанию
forks	Число параллельных процессов, генерируемых при коммуникации с удаленными хостами	5

ПРИМЕЧАНИЕ

На этапе offline-установки указанные параметры игнорируются. Вместо этих репозиториев используются данные, сохраненные локально в кластере Enterprise Tools.

Импорт настроек ET

Дарья Барышева, Елена Дворядкина

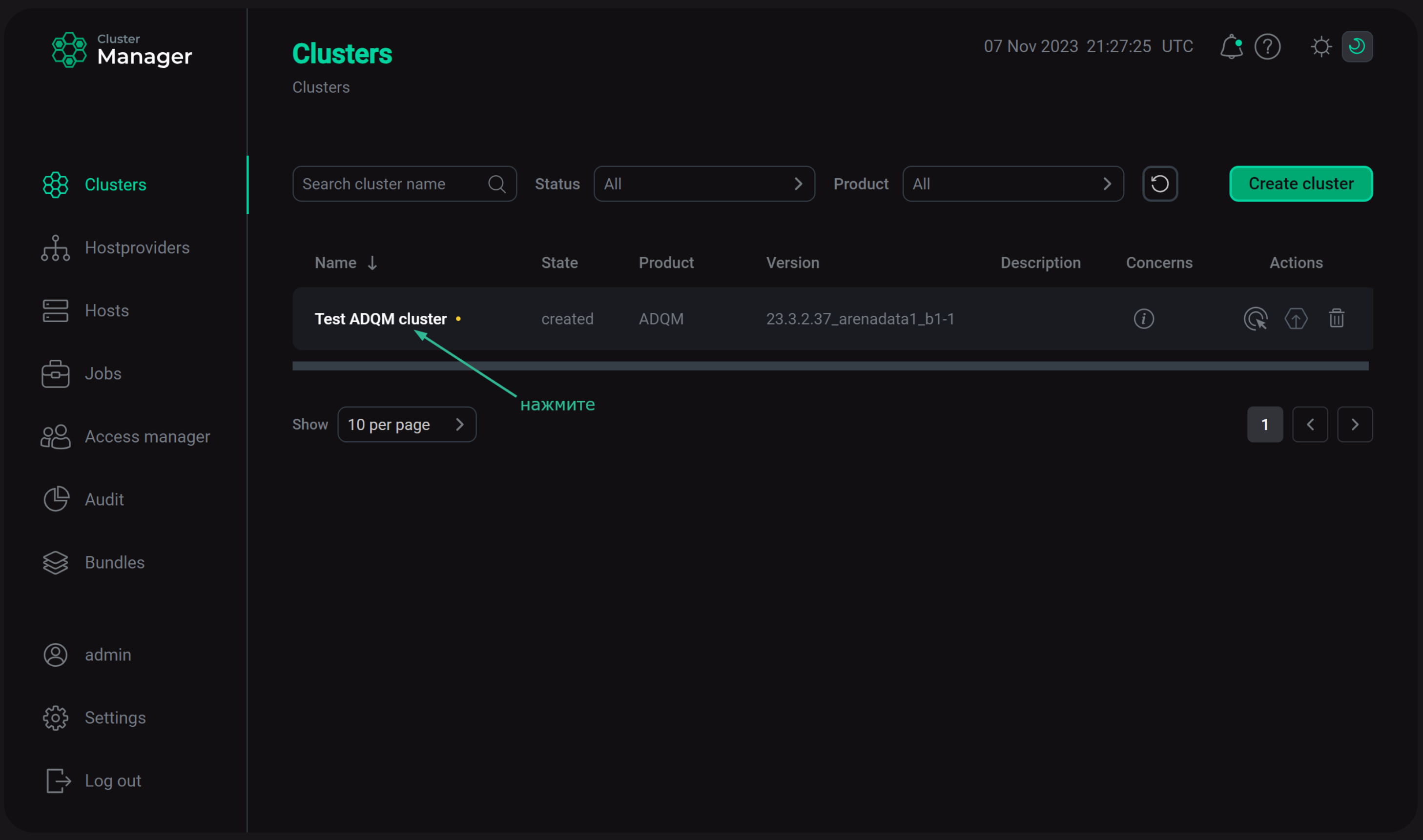
Назад к содержанию

ВАЖНО

Этот шаг необходим только в случае offline-установки.

Для установки кластера в режиме offline требуется интегрировать его с ранее созданным кластером Enterprise Tools (ET):

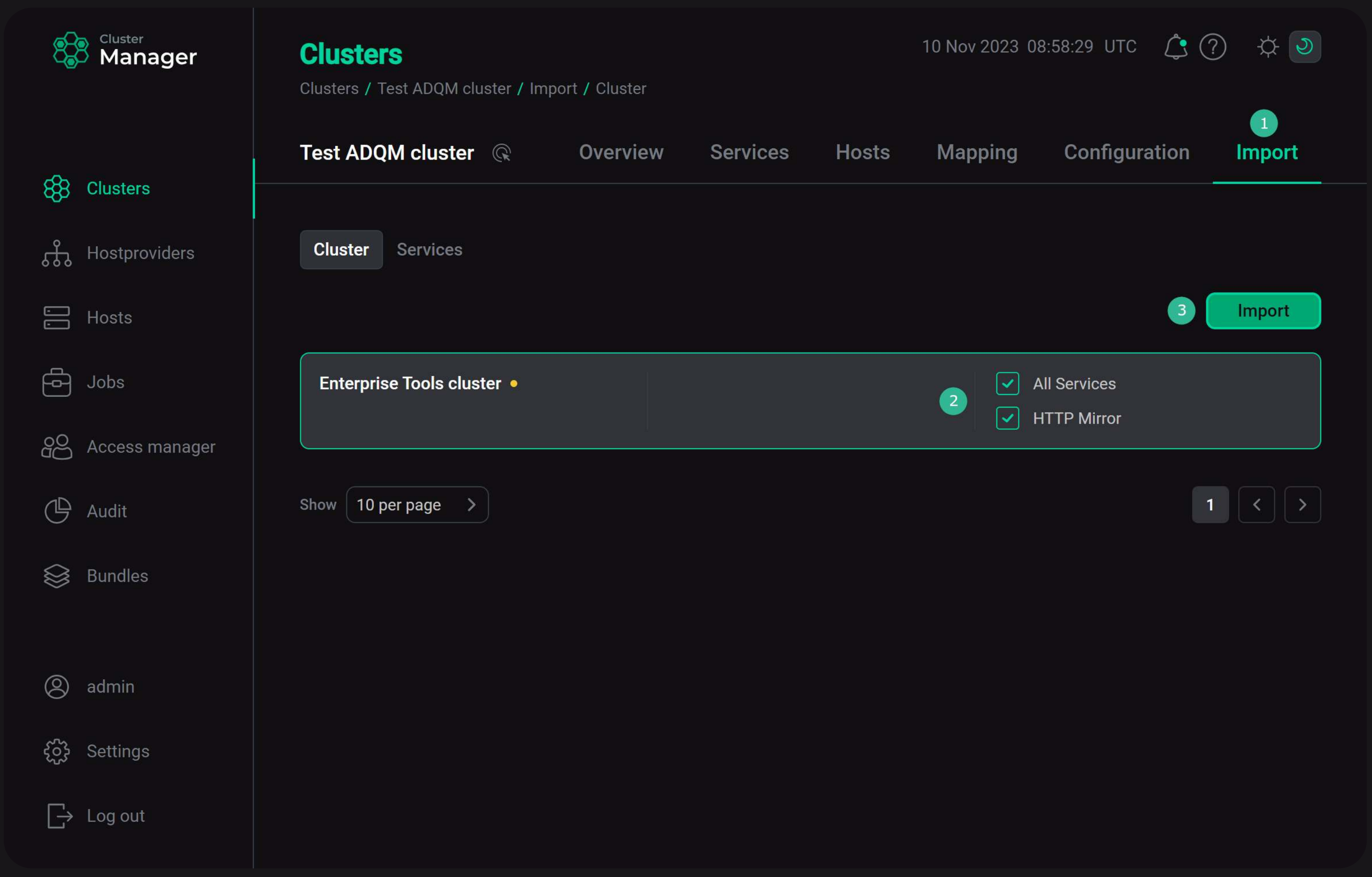
1. Выберите кластер на странице ***Clusters***. Для этого нажмите на имя кластера в столбце ***Name***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
07 Nov 2023 21:27:25 UTC
Search cluster name
Status
All
Product
All
Create cluster
нажмите
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

Выбор кластера

2. Откройте вкладку ***Import*** на странице кластера.
3. Выберите сервис ***HTTP Mirror*** предварительно созданного кластера ET.
4. Нажмите ***Import***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Import / Cluster
10 Nov 2023 08:58:29 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
1
Import
Cluster
Services
3
Import
Enterprise Tools cluster
2
☑ All Services
☑ HTTP Mirror
Show
10 per page
1

Интеграция с кластером ET

ПРИМЕЧАНИЕ

После успешного импорта данные, хранимые локально в кластере Enterprise Tools, могут быть использованы на этапе установки продуктового кластера. Таким образом, их загрузка через Интернет не требуется.

Установка кластера

Дарья Барышева, Елена Дворядкина

Содержание

Назад к содержанию

Шаг 1. Запуск установки
  - Установка всего кластера
  - Установка отдельных сервисов

Шаг 2. Просмотр результатов установки

Шаг 3. Проверка статуса кластера после установки

Шаг 1. Запуск установки

Существует два способа установки:

1. **Установка всего кластера**. В этом методе все сервисы кластера устанавливаются автоматически друг за другом после применения действия ***Install*** к кластеру.
2. **Установка отдельных сервисов**. В этом методе каждый сервис устанавливается вручную путем применения к нему одного или нескольких действий. Этот способ также можно использовать для добавления новых сервисов в уже развернутый кластер.

В обоих методах каждый сервис запускается автоматически после установки.

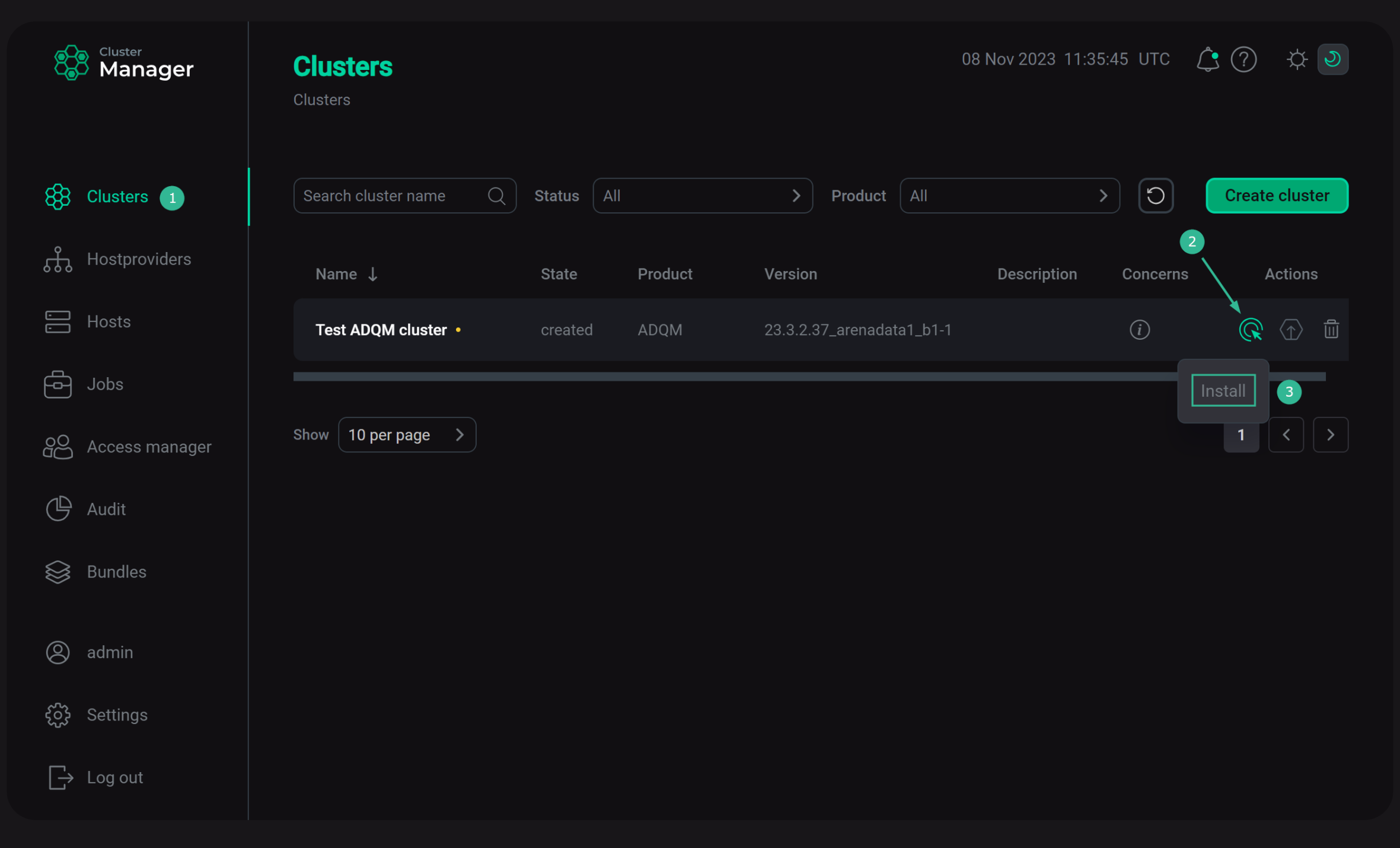
ПРИМЕЧАНИЕ

В случае ошибок логи можно найти на странице ***Jobs***.

Установка всего кластера

Чтобы установить все сервисы кластера за одну операцию, выполните шаги:

1. Примените действие ***Install*** к кластеру, выбранному на странице ***Clusters***, нажав на иконку в столбце ***Actions***.

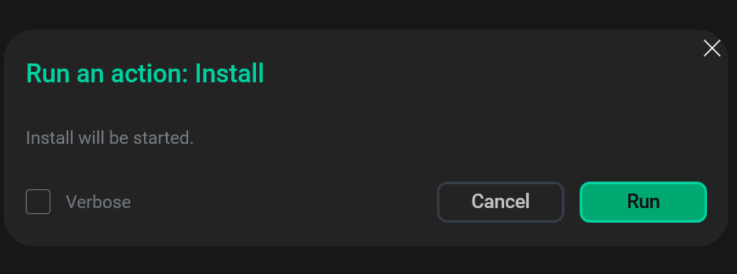


Cluster Manager
Clusters
Clusters
08 Nov 2023 11:35:45 UTC
Clusters 1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
2
Install
3
Show 10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	created	ADQM	23.3.2.37_arenadata1_b1-1			

Переход к установке всего кластера

2. Подтвердите действие в открывшемся окне.



Run an action: Install

Install will be started.

☐ Verbose
Cancel
Run

Подтверждение действия

3. Дождитесь завершения установки.

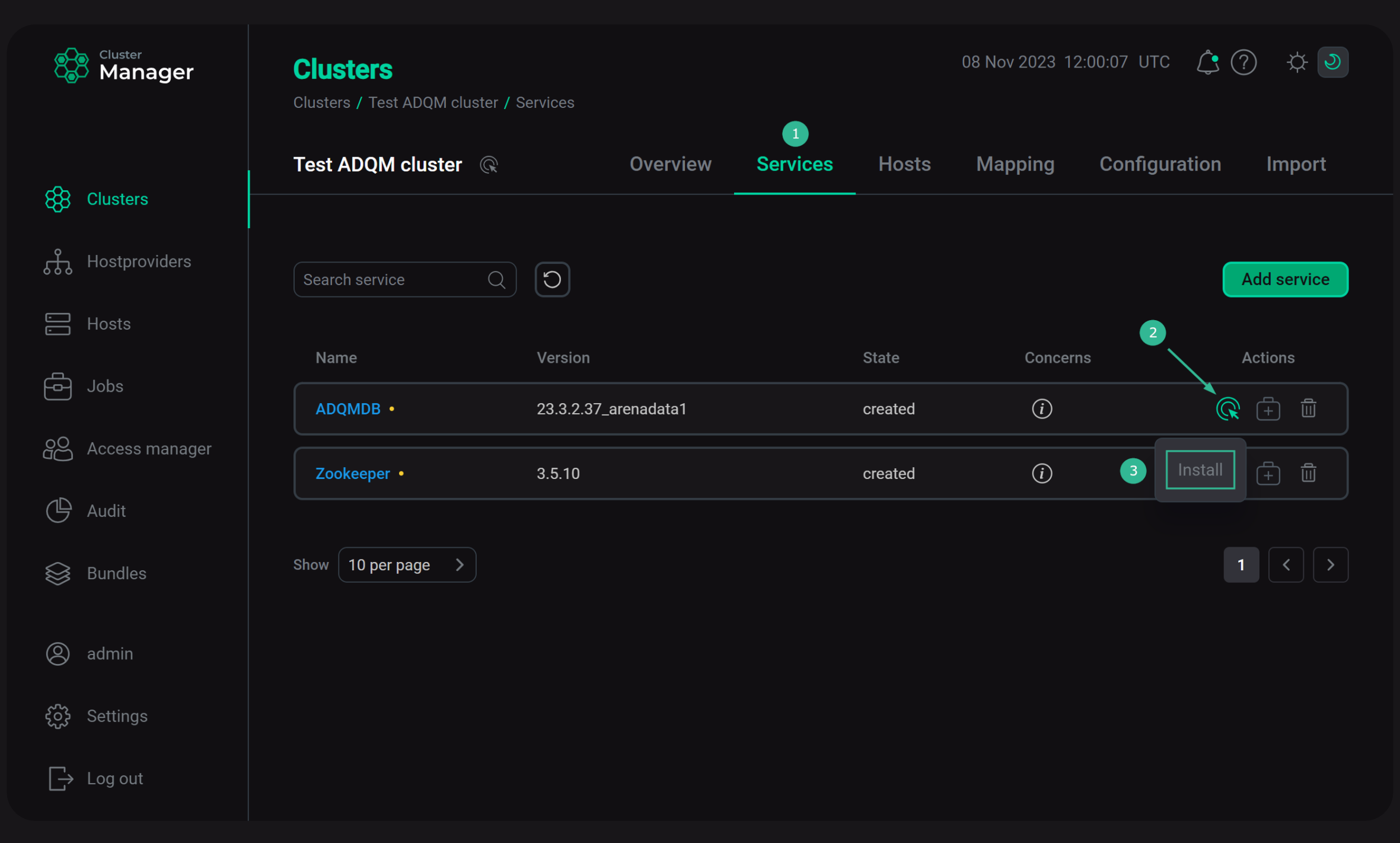
Установка отдельных сервисов

Чтобы установить один или несколько сервисов вручную, необходимо предварительно добавить эти сервисы в кластер и затем применить к ним действия, описанные ниже.

ВНИМАНИЕ

Обратите внимание, что сервисы устанавливаются по очереди друг за другом. Не пытайтесь установить следующий сервис до окончания установки предыдущего.

1. Откройте вкладку ***Services*** на странице кластера. Для сервиса, который должен быть установлен, нажмите на иконку в столбце ***Actions*** и выберите действие ***Install*** из выпадающего списка.

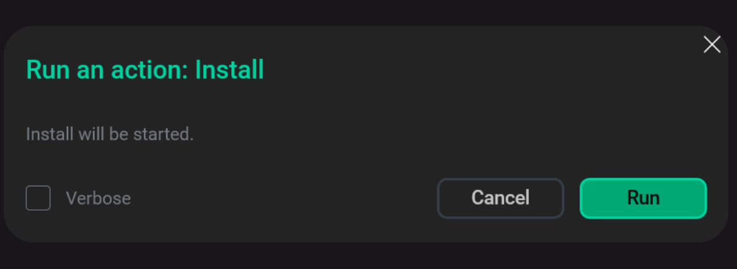


Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
08 Nov 2023 12:00:07 UTC
Test ADQM cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
2
3
Install
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADQMDB	23.3.2.37_arenadata1	created		
Zookeeper	3.5.10	created		

Переход к установке отдельных сервисов

2. Подтвердите действие в открывшемся окне.



Run an action: Install

Install will be started.

☐ Verbose
Cancel
Run

Подтверждение действия

3. Дождитесь завершения установки и убедитесь, что статус сервиса изменился с `created` на `installed`.
4. Повторите предыдущие шаги для всех сервисов, которые необходимо установить.

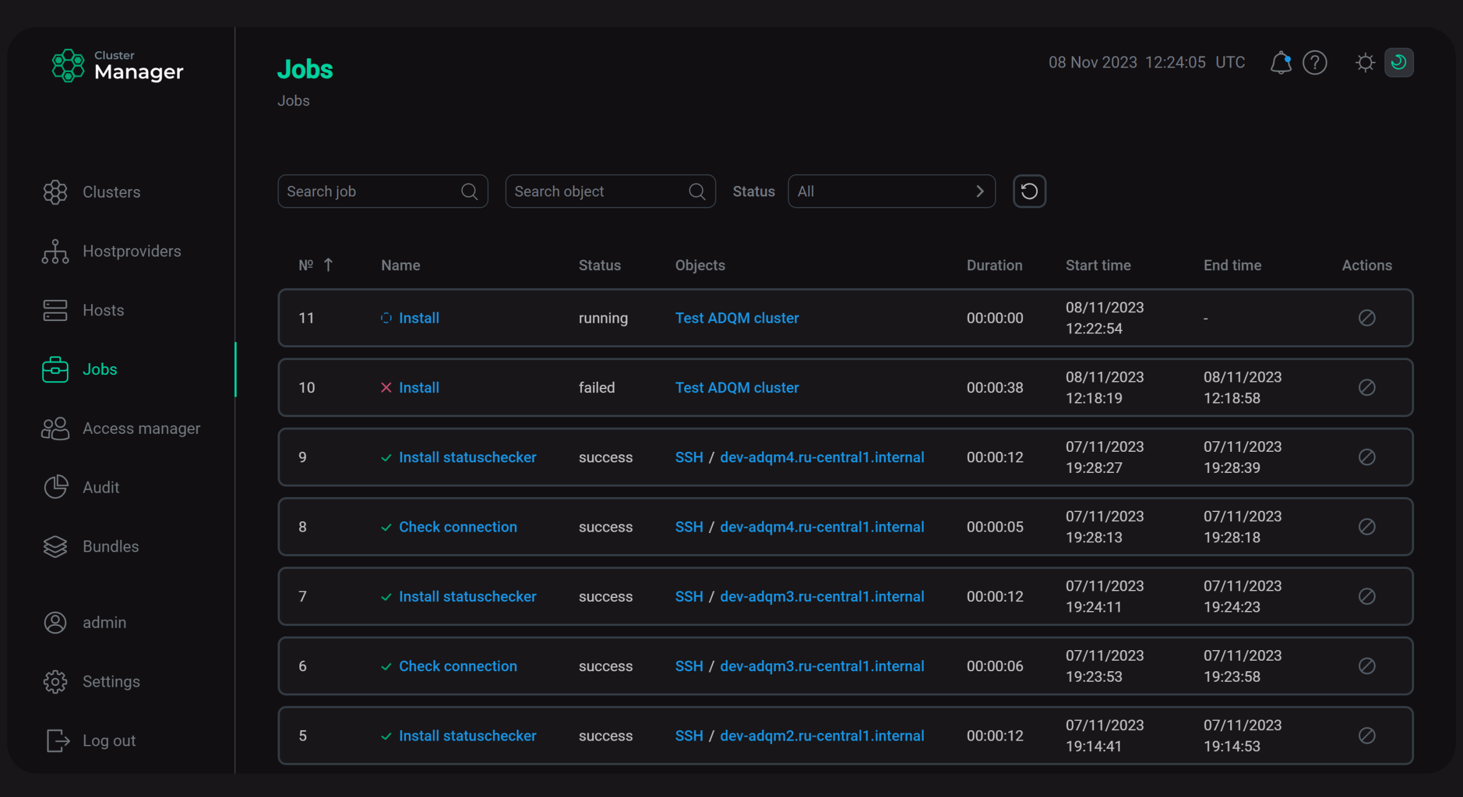
ВАЖНО

При установке сервисов вручную важно соблюдать следующий порядок:

1. Zookeeper / Clickhousekeeper
2. ADQMDB
3. Chproxy
4. Monitoring Clients / Monitoring

Шаг 2. Просмотр результатов установки

ADCM запускает отдельную задачу (job) для установки, как и для любого другого процесса. Для получения информации о статусе задач предназначена страница ***Jobs***.

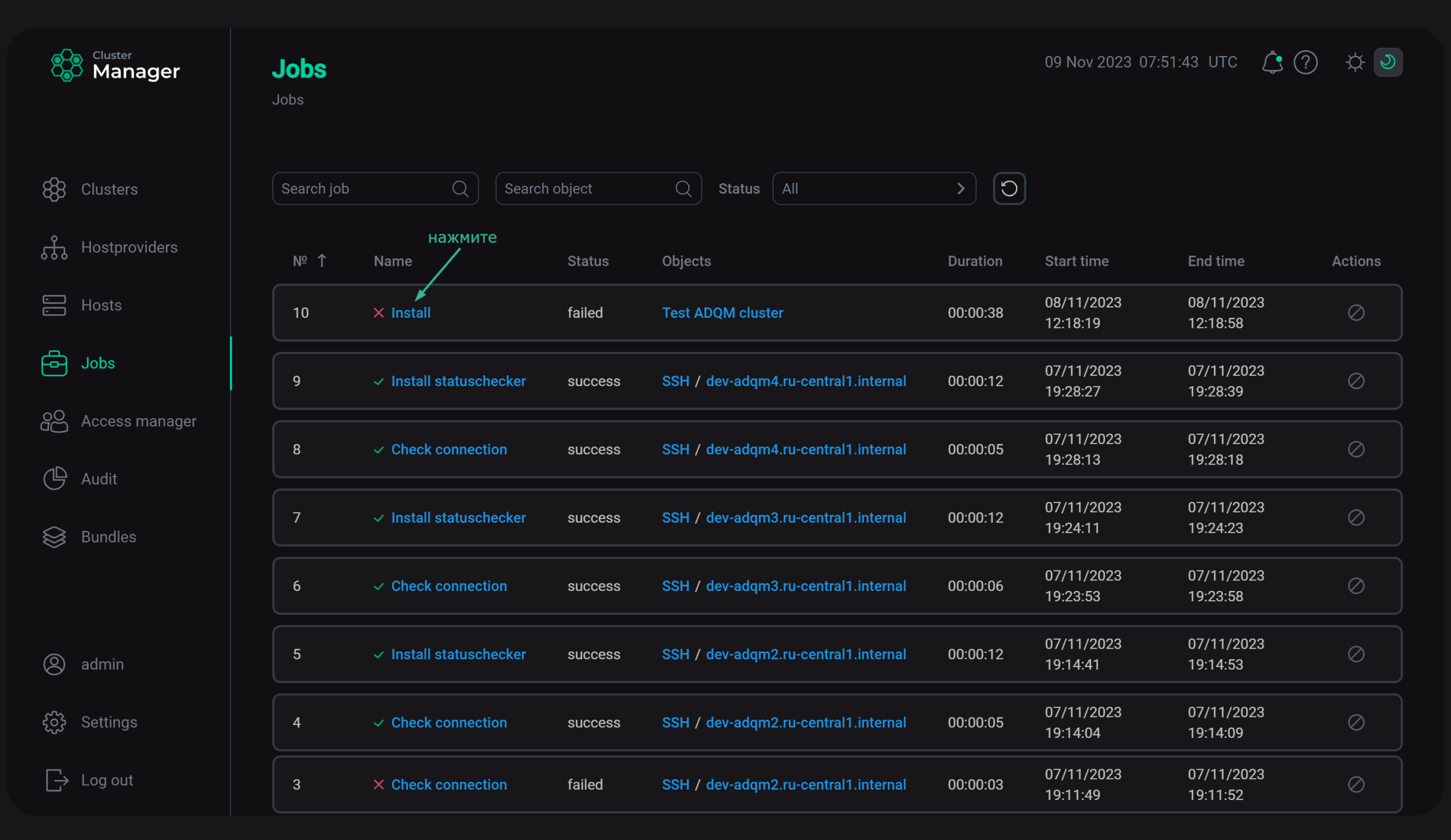


Cluster Manager
Jobs
Jobs
08 Nov 2023 12:24:05 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search job
Search object
Status All

№	Name	Status	Objects	Duration	Start time	End time	Actions
11	Install	running	Test ADQM cluster	00:00:00	08/11/2023 12:22:54	-	
10	Install	failed	Test ADQM cluster	00:00:38	08/11/2023 12:18:19	08/11/2023 12:18:58	
9	Install statuschecker	success	SSH / dev-adqm4.ru-central1.internal	00:00:12	07/11/2023 19:28:27	07/11/2023 19:28:39	
8	Check connection	success	SSH / dev-adqm4.ru-central1.internal	00:00:05	07/11/2023 19:28:13	07/11/2023 19:28:18	
7	Install statuschecker	success	SSH / dev-adqm3.ru-central1.internal	00:00:12	07/11/2023 19:24:11	07/11/2023 19:24:23	
6	Check connection	success	SSH / dev-adqm3.ru-central1.internal	00:00:06	07/11/2023 19:23:53	07/11/2023 19:23:58	
5	Install statuschecker	success	SSH / dev-adqm2.ru-central1.internal	00:00:12	07/11/2023 19:14:41	07/11/2023 19:14:53	

Страница Jobs

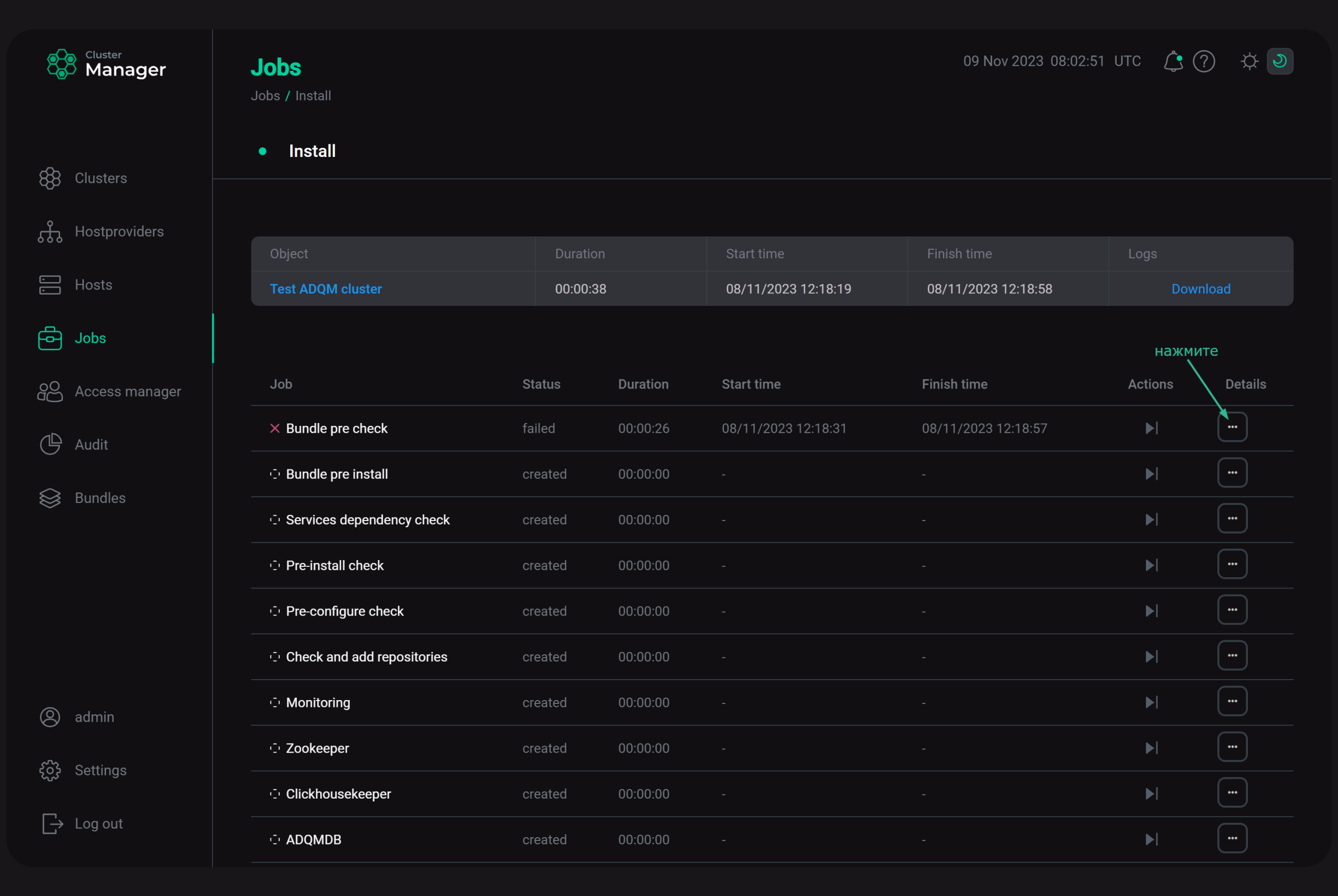
Если установка завершается успешно, соответствующая ей задача на странице ***Jobs*** переходит из статуса `running` в `success`; в противном случае — в статус `failed`. Чтобы получить более детальную информацию о возникших ошибках, можно перейти на страницу задачи, кликнув по ней в списке задач на странице ***Jobs***.



Cluster Manager
Jobs
Jobs
09 Nov 2023 07:51:43 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search job
Search object
Status All
нажмите

№	Name	Status	Objects	Duration	Start time	End time	Actions
10	Install	failed	Test ADQM cluster	00:00:38	08/11/2023 12:18:19	08/11/2023 12:18:58	
9	Install statuschecker	success	SSH / dev-adqm4.ru-central1.internal	00:00:12	07/11/2023 19:28:27	07/11/2023 19:28:39	
8	Check connection	success	SSH / dev-adqm4.ru-central1.internal	00:00:05	07/11/2023 19:28:13	07/11/2023 19:28:18	
7	Install statuschecker	success	SSH / dev-adqm3.ru-central1.internal	00:00:12	07/11/2023 19:24:11	07/11/2023 19:24:23	
6	Check connection	success	SSH / dev-adqm3.ru-central1.internal	00:00:06	07/11/2023 19:23:53	07/11/2023 19:23:58	
5	Install statuschecker	success	SSH / dev-adqm2.ru-central1.internal	00:00:12	07/11/2023 19:14:41	07/11/2023 19:14:53	
4	Check connection	success	SSH / dev-adqm2.ru-central1.internal	00:00:05	07/11/2023 19:14:04	07/11/2023 19:14:09	
3	Check connection	failed	SSH / dev-adqm2.ru-central1.internal	00:00:03	07/11/2023 19:11:49	07/11/2023 19:11:52	

Переход на страницу задачи (job)



Cluster Manager
Jobs
Jobs / Install
09 Nov 2023 08:02:51 UTC
Install
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
нажмите

Object	Duration	Start time	Finish time	Logs
Test ADQM cluster	00:00:38	08/11/2023 12:18:19	08/11/2023 12:18:58	Download

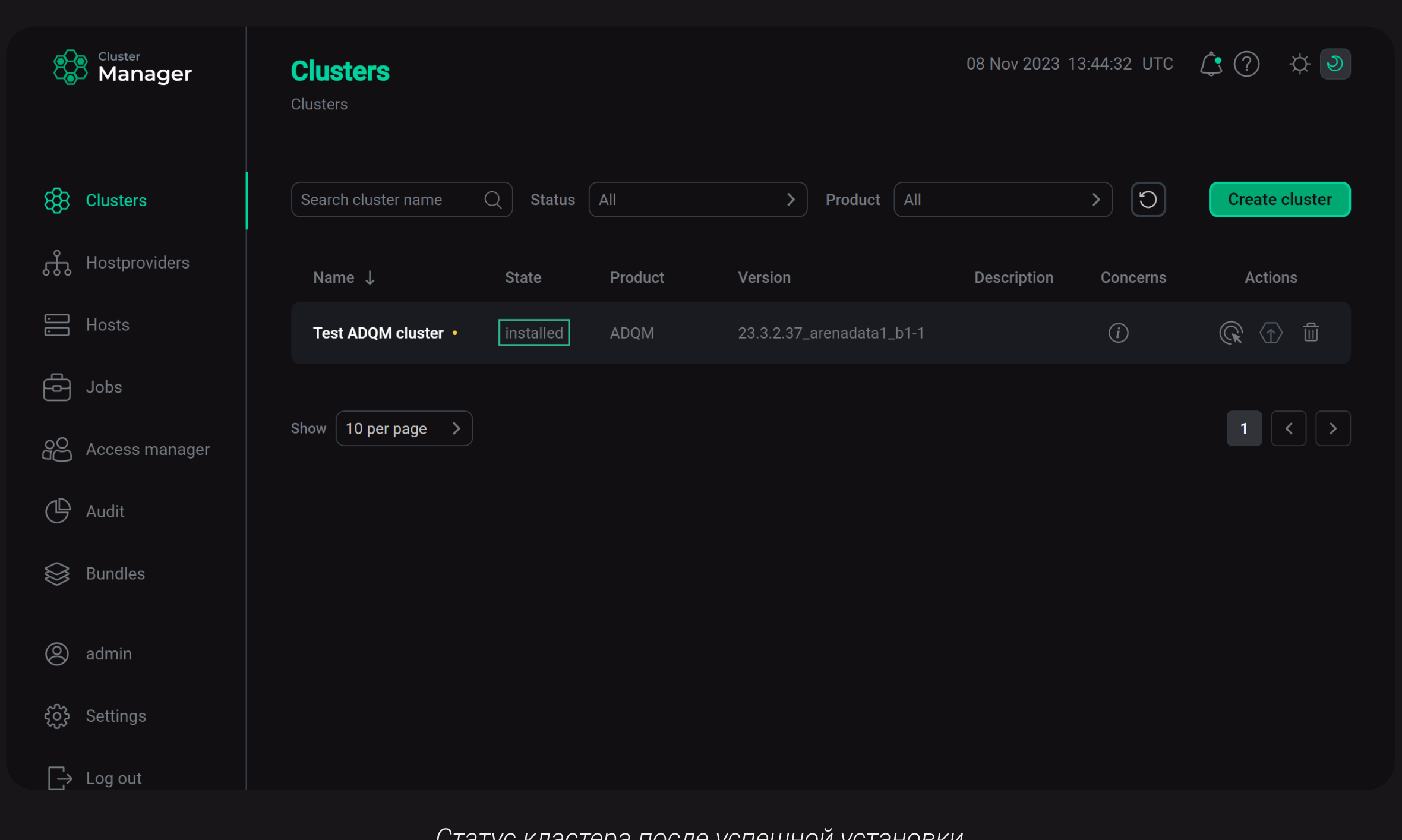
Job	Status	Duration	Start time	Finish time	Actions	Details
Bundle pre check	failed	00:00:26	08/11/2023 12:18:31	08/11/2023 12:18:57		
Bundle pre install	created	00:00:00	-	-		
Services dependency check	created	00:00:00	-	-		
Pre-install check	created	00:00:00	-	-		
Pre-configure check	created	00:00:00	-	-		
Check and add repositories	created	00:00:00	-	-		
Monitoring	created	00:00:00	-	-		
Zookeeper	created	00:00:00	-	-		
Clickhousekeeper	created	00:00:00	-	-		
ADQMDB	created	00:00:00	-	-		

Страница задачи установки кластера

Шаг 3. Проверка статуса кластера после установки

В результате успешной установки происходит обновление статуса кластера и его сервисов:

- Статус кластера, отображаемый в столбце ***State*** на странице ***Clusters***, меняет свое значение с `created` на `installed`.

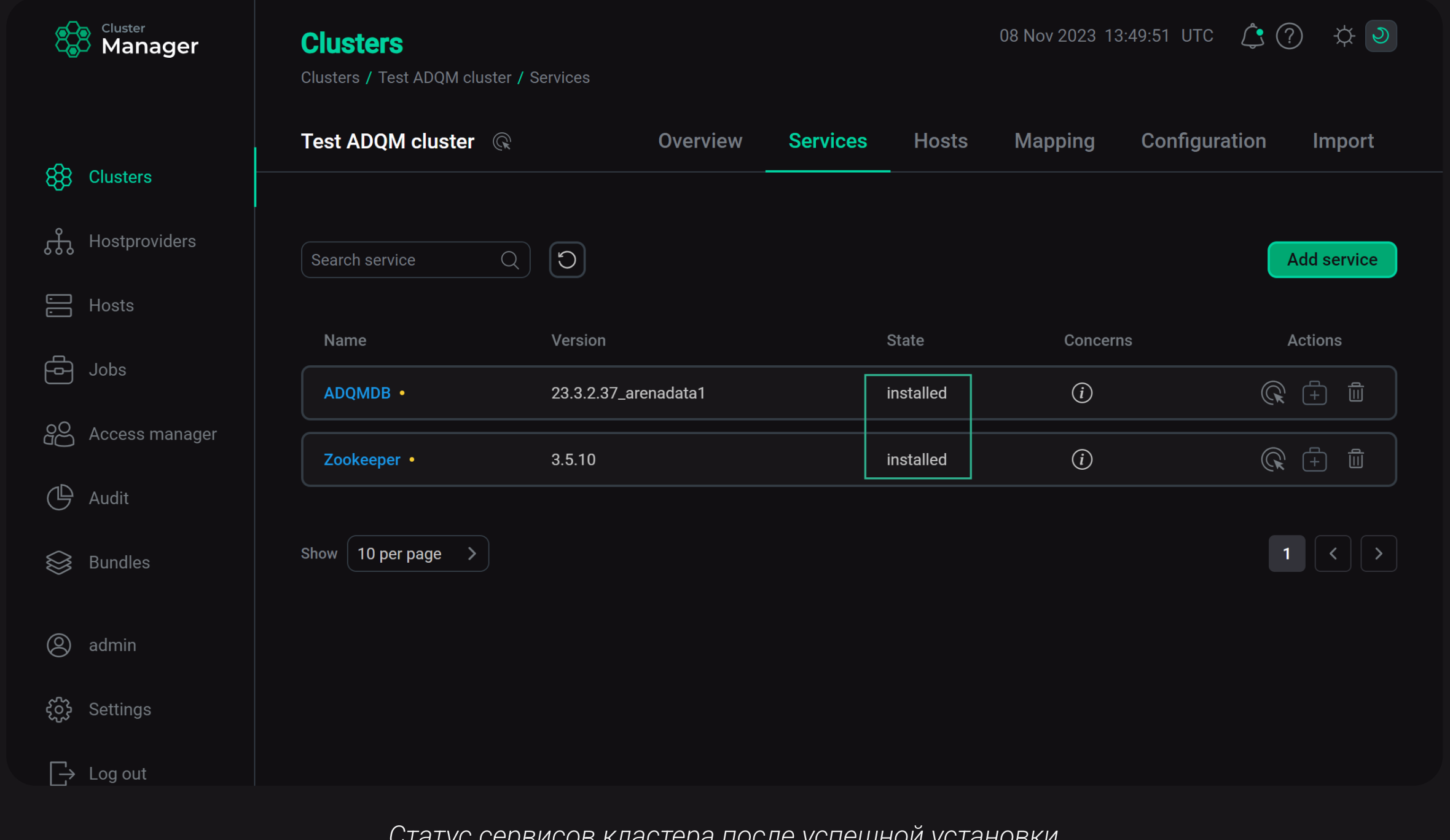


Cluster Manager
Clusters
Clusters
08 Nov 2023 13:44:32 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	installed	ADQM	23.3.2.37_arenadata1_b1-1			

Статус кластера после успешной установки

- Статус сервисов, отображаемый на вкладке ***Services*** страницы кластера, меняет свое значение с `created` на `installed`.



Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services
08 Nov 2023 13:49:51 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADQMDB	23.3.2.37_arenadata1	installed		
Zookeeper	3.5.10	installed		

Статус сервисов кластера после успешной установки

Сервис мониторинга

Елена Дворядкина

Содержание

Назад к содержанию

Обзор
Шаг 1. Добавление сервиса
Шаг 2. Добавление компонентов
Шаг 3. Настройка сервиса
Шаг 4. Установка сервиса
Шаг 5. Проверка результатов
  - Данные в формате Prometheus
  - Веб-интерфейс Prometheus
  - Веб-интерфейс Grafana

Сервис Monitoring (доступен начиная с версии ADQM 23.8.2.7) разворачивает в ADQM свой сервер Prometheus, который реализует сбор и хранение метрик мониторинга кластера ADQM, а также поддерживает возможность использования веб-приложения Grafana для визуализации и анализа информации. В данной статье описываются шаги, необходимые для подключения этого сервиса.

Обзор

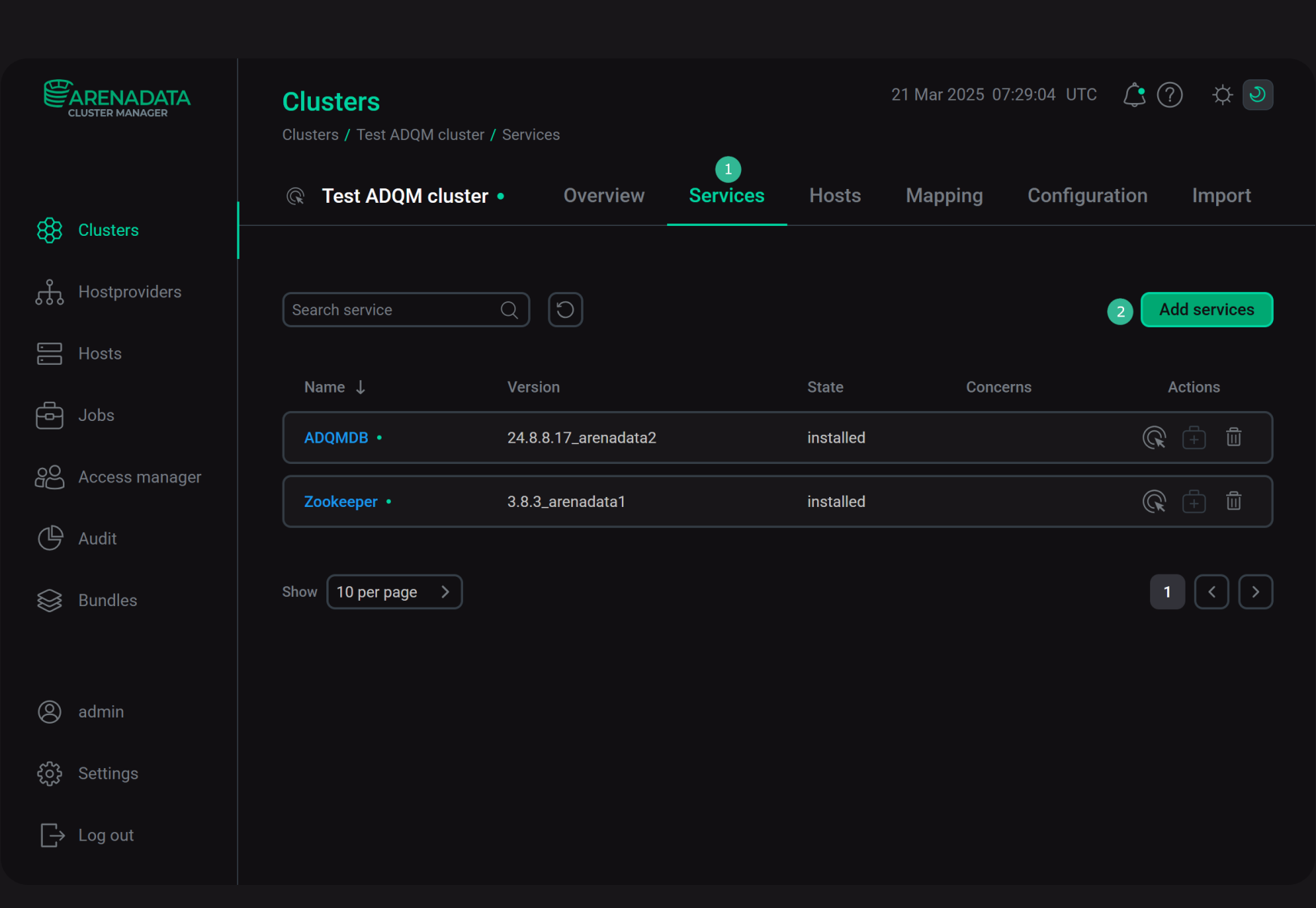
При добавлении сервиса Monitoring в ADQM на все или выборочные хосты кластера можно установить node exporter — агент мониторинга, считывающий с хоста системные метрики, которые будет собирать Prometheus. Prometheus также может собирать метрики сервисов ADQM (ClickHouse, ZooKeeper, ClickHouse Keeper, Chproxy). Эти метрики будут доступны в формате Prometheus на портах и конечных точках (endpoints), указанных в параметрах секции ***Metrics*** сервисных конфигураций. Просматривать и анализировать данные, которые собирает сервис мониторинга, можно также в веб-интерфейсе Prometheus или Grafana.

ПРИМЕЧАНИЕ

- Если у вас уже настроена совместимая с Prometheus система мониторинга (например, свой сервер Prometheus или VictoriaMetrics), можно использовать ее, чтобы собирать метрики ADQM. Для этого в своей системе мониторинга укажите настройки доступа к метрикам ADQM, используя параметры установленного в ADQM Prometheus, которые конфигурируются на странице сервиса Monitoring в интерфейсе ADCM.
- Вы также можете использовать механизм Federation, чтобы перенести все метрики из сервера Prometheus, развернутого в ADQM, в свой Prometheus.

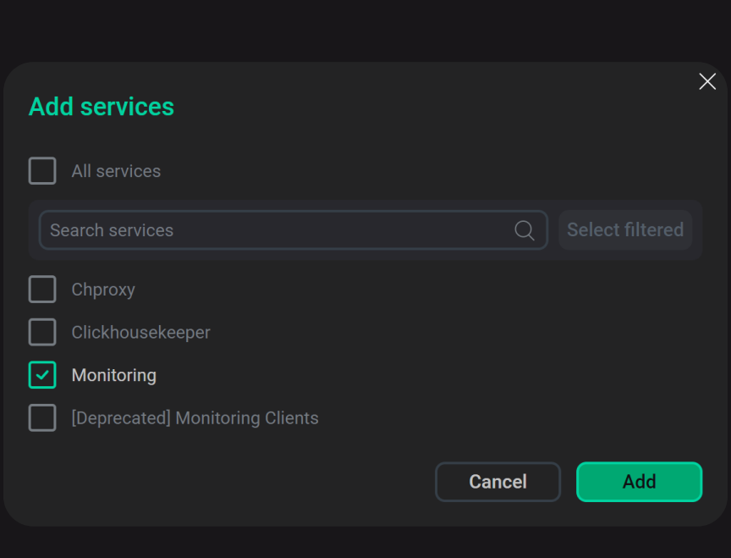
Шаг 1. Добавление сервиса

1. В интерфейсе ADCM откройте страницу ***Clusters*** и кликните по имени кластера ADQM. Затем на открывшейся странице кластера перейдите на вкладку ***Services*** и нажмите ***Add services***.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 07:29:04 UTC
Test ADQM cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
2
Add services
Name
Version
State
Concerns
Actions
ADQMDB
24.8.8.17_arenadata2
installed
Zookeeper
3.8.3_arenadata1
installed
Show
10 per page
1

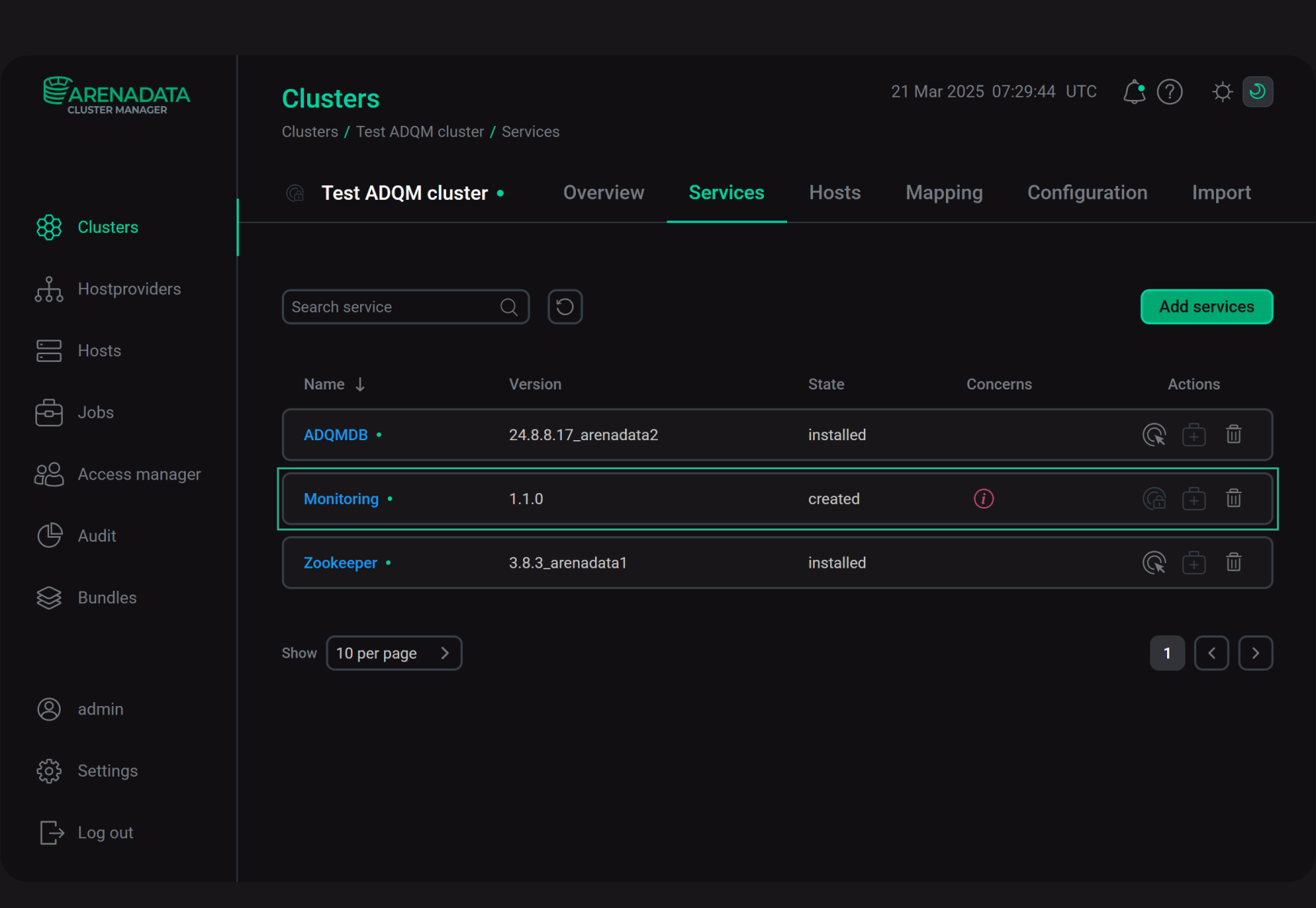
Переход к добавлению сервисов

2. В открывшемся окне выберите сервис ***Monitoring*** и нажмите ***Add***.


Add services
All services
Search services
Select filtered
Chproxy
Clickhousekeeper
Monitoring
[Deprecated] Monitoring Clients
Cancel
Add

Выбор сервиса

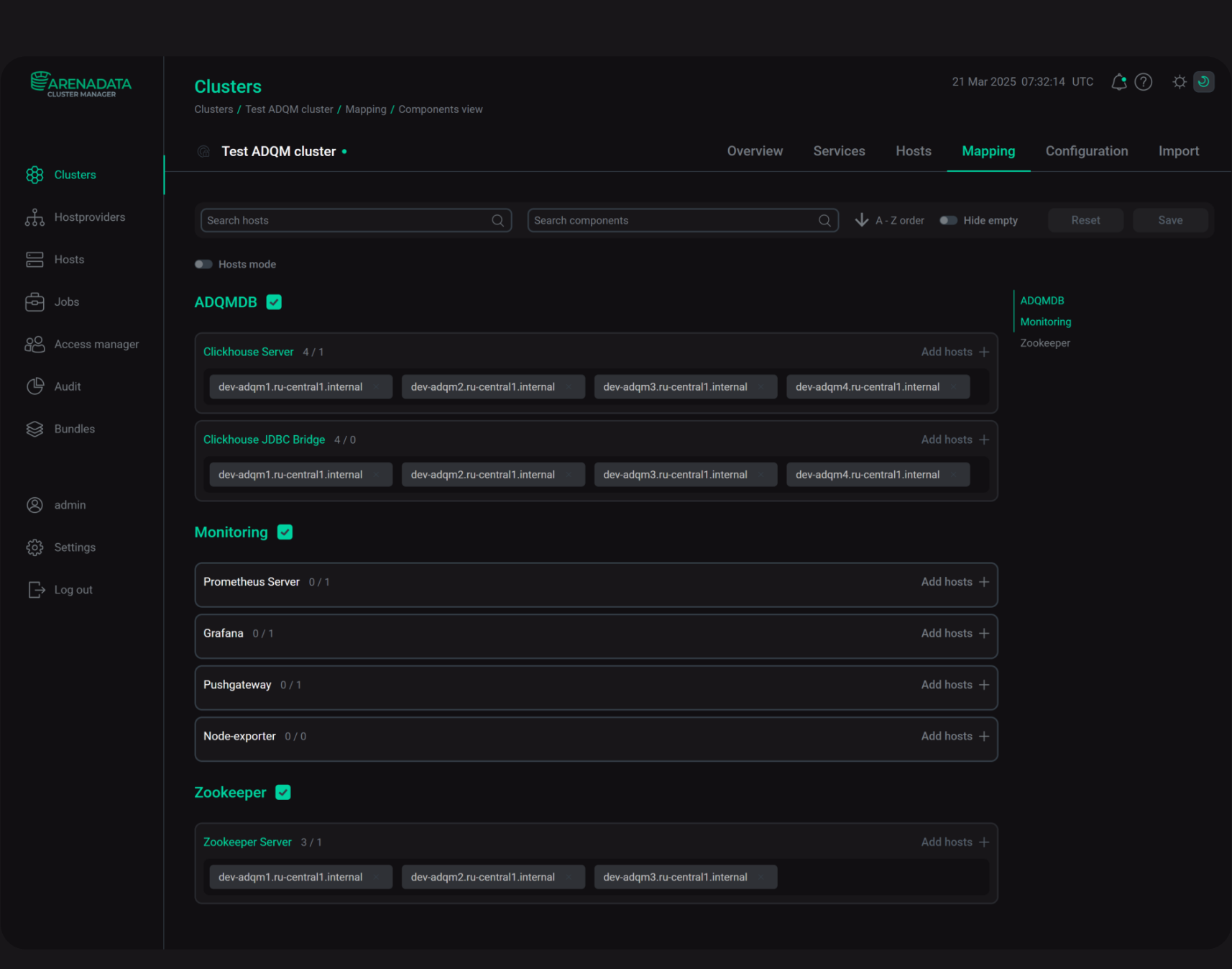
В результате добавленный сервис отобразится на вкладке ***Services***.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 07:29:44 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Name
Version
State
Concerns
Actions
ADQMDB
24.8.8.17_arenadata2
installed
Monitoring
1.1.0
created
Zookeeper
3.8.3_arenadata1
installed
Show
10 per page

Результат успешного добавления сервиса в кластер

Шаг 2. Добавление компонентов

1. На странице кластера откройте вкладку ***Mapping***, чтобы перейти к распределению сервисных компонентов между хостами кластера.

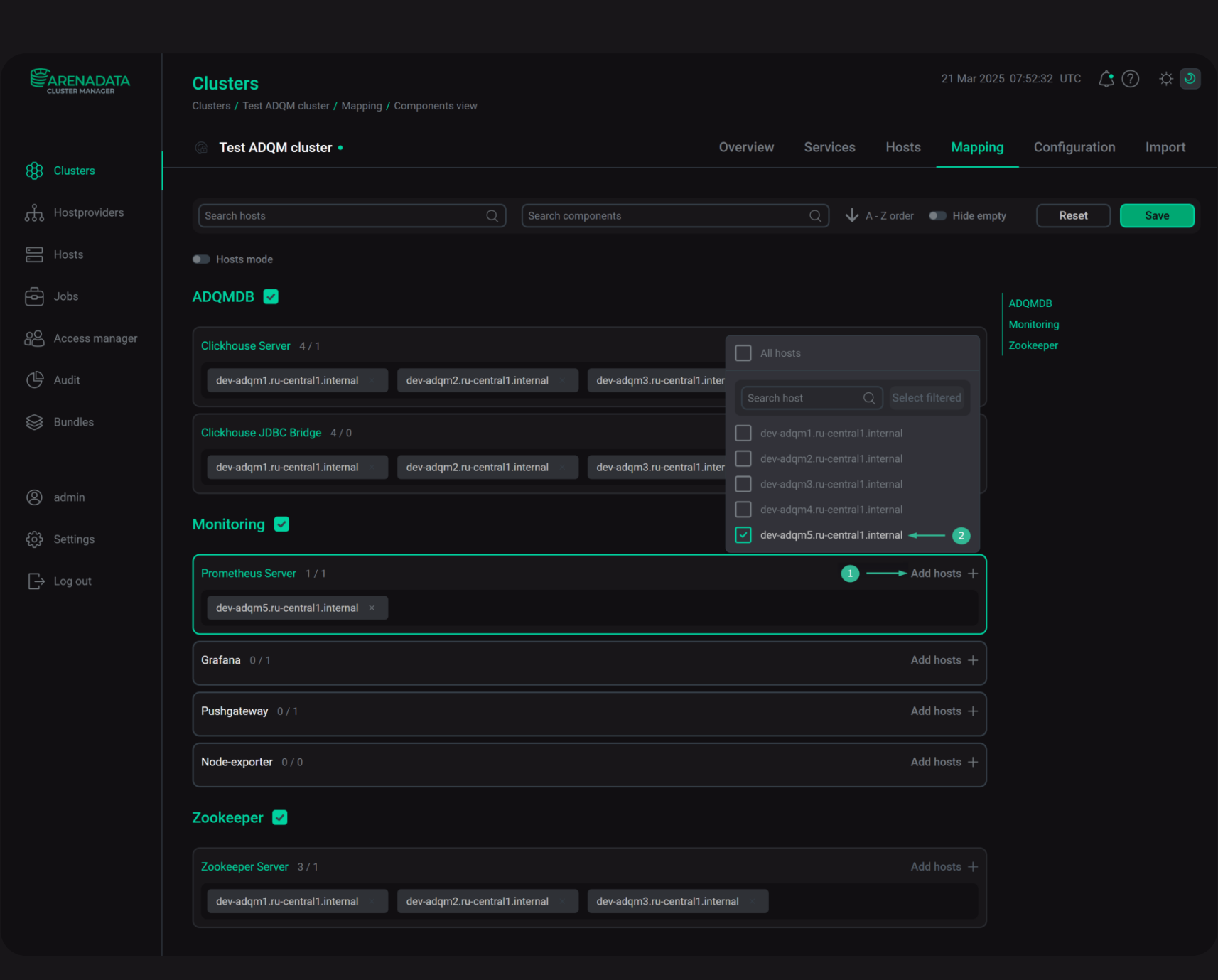

Clusters
Clusters / Test ADQM cluster / Mapping / Components view
21 Mar 2025 07:32:14 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
ADQMDB
Clickhouse Server 4 / 1
Add hosts
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
Clickhouse JDBC Bridge 4 / 0
Monitoring
Prometheus Server 0 / 1
Grafana 0 / 1
Pushgateway 0 / 1
Node-exporter 0 / 0
Zookeeper
Zookeeper Server 3 / 1

Переход к распределению сервисных компонентов

Компоненты сервиса Monitoring

Компонент	Описание
Prometheus Server	Хранит и позволяет запрашивать все метрики кластера ADQM
Grafana	Позволяет визуализировать метрики ADQM в виде графиков и диаграмм, организованных в дашборды
Pushgateway	Принимает и отдает в Prometheus статические метрики. Используется, чтобы передать в Prometheus структуру кластера ADQM
Node Exporter	Собирает системные метрики и отдает их в Prometheus

2. Назначьте хосты компонентам сервиса Monitoring — нажмите ***Add hosts*** и выберите нужные хосты в открывшемся окне.

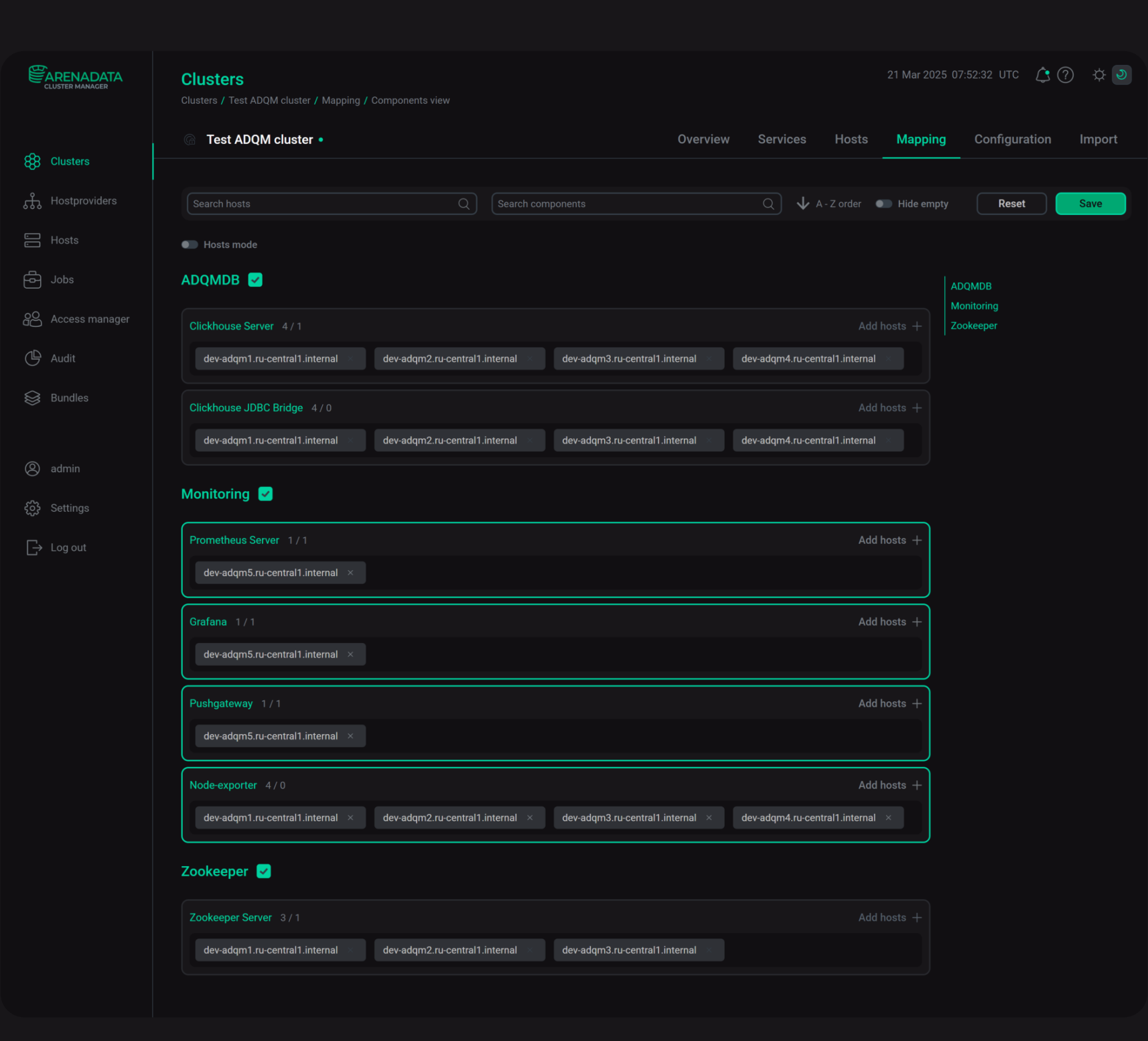

Clusters
Clusters / Test ADQM cluster / Mapping / Components view
21 Mar 2025 07:52:32 UTC
Test ADQM cluster
Mapping
Reset
Save
All hosts
Search host
Select filtered
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
dev-adqm5.ru-central1.internal
2
Prometheus Server 1 / 1
1
Add hosts
dev-adqm5.ru-central1.internal
Grafana 0 / 1
Pushgateway 0 / 1
Node-exporter 0 / 0
Zookeeper Server 3 / 1

Выбор хоста для компонента

ВНИМАНИЕ

Компоненты Prometheus Server и Pushgateway не рекомендуется устанавливать на хосты с ADQM — используйте для них отдельные хосты. Иначе при выходе из строя хоста ADQM и/или критически большой нагрузке на него информация о соответствующих проблемах не будет сохранена.

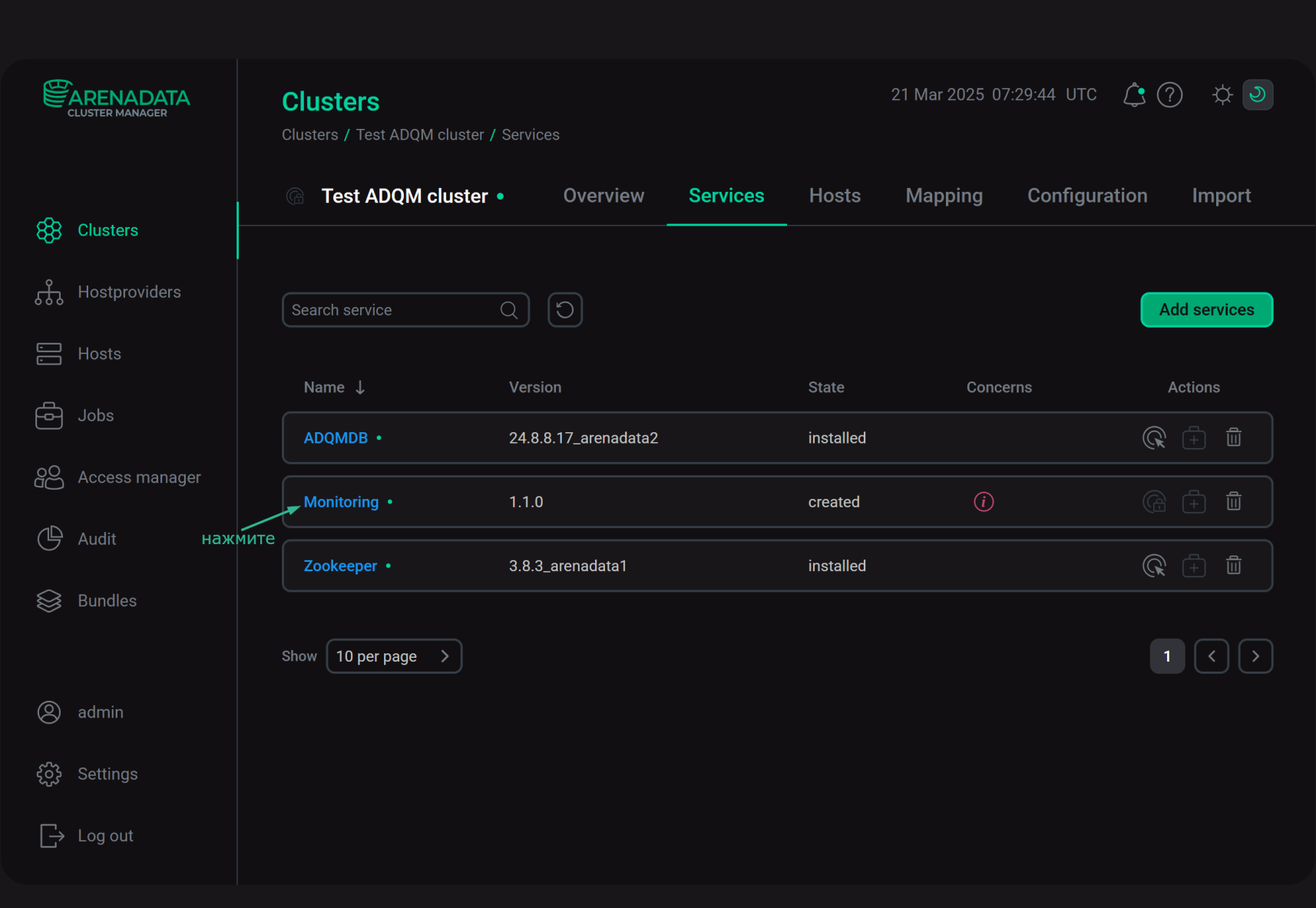
3. После завершения распределения компонентов нажмите ***Save***, чтобы сохранить изменения.


Clusters
Clusters / Test ADQM cluster / Mapping / Components view
21 Mar 2025 07:52:32 UTC
Test ADQM cluster
Mapping
Reset
Save
ADQMDB
Clickhouse Server 4 / 1
Clickhouse JDBC Bridge 4 / 0
Monitoring
Prometheus Server 1 / 1
dev-adqm5.ru-central1.internal
Grafana 1 / 1
dev-adqm5.ru-central1.internal
Pushgateway 1 / 1
dev-adqm5.ru-central1.internal
Node-exporter 4 / 0
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
Zookeeper
Zookeeper Server 3 / 1

Сохранение распределения компонентов

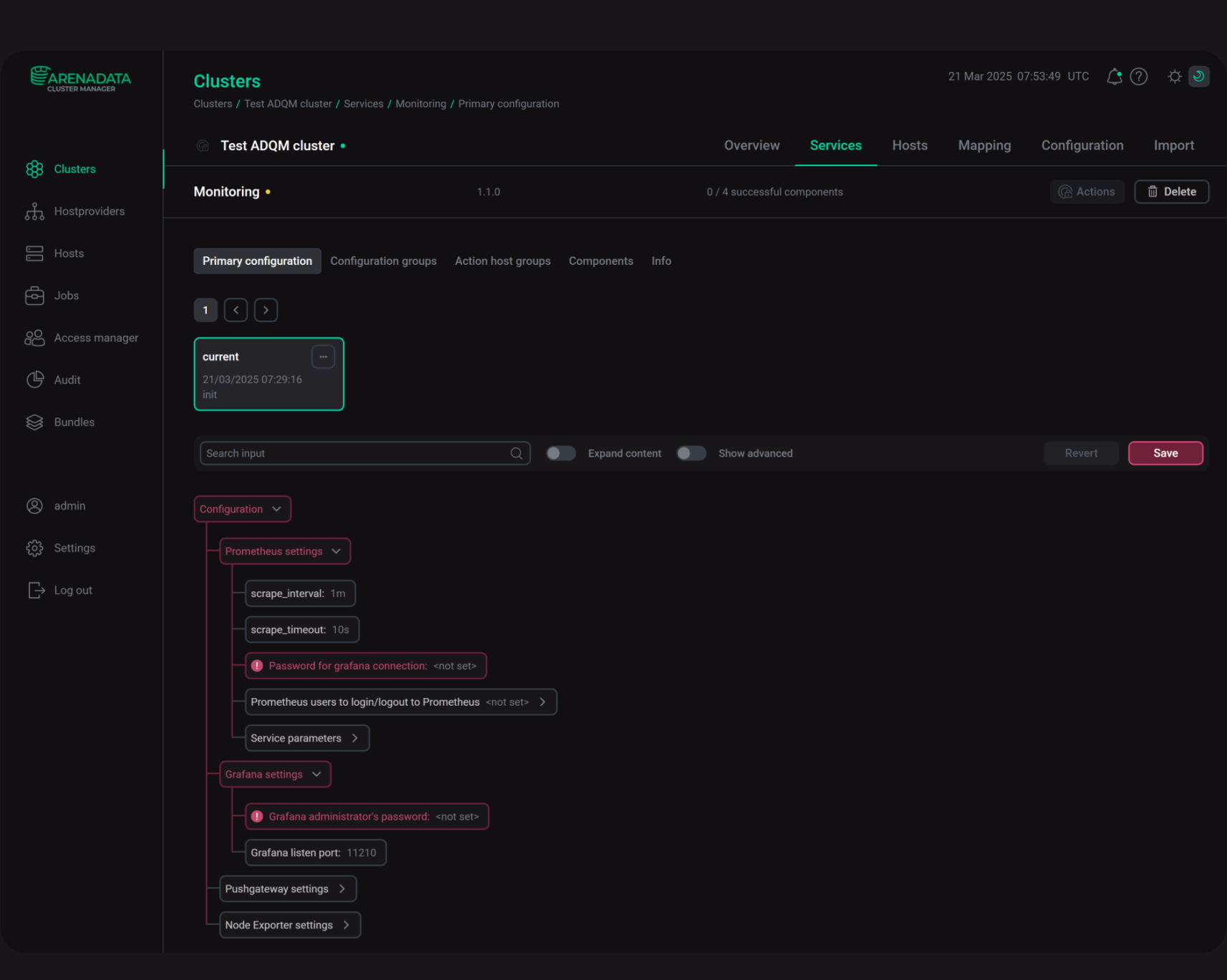
Шаг 3. Настройка сервиса

1. Откройте вкладку ***Services*** на странице кластера и кликните по имени сервиса ***Monitoring*** в столбце ***Name***.


Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 07:29:44 UTC
Test ADQM cluster
Services
Add services
ADQMDB
24.8.8.17_arenadata2
installed
Monitoring
1.1.0
created
нажмите
Zookeeper
3.8.3_arenadata1
installed

Переход к настройке сервиса

2. В открывшемся окне заполните конфигурационные параметры сервиса — описания параметров приведены в разделе ***Monitoring*** статьи Конфигурационные параметры. Поля, подсвеченные красным, обязательны для заполнения.


Clusters
Clusters / Test ADQM cluster / Services / Monitoring / Primary configuration
21 Mar 2025 07:53:49 UTC
Monitoring
1.1.0
0 / 4 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
current
21/03/2025 07:29:16
init
Search input
Expand content
Show advanced
Revert
Save
Configuration
Prometheus settings
scrape_interval: 1m
scrape_timeout: 10s
Password for grafana connection: <not set>
Prometheus users to login/logout to Prometheus <not set>
Service parameters
Grafana settings
Grafana administrator's password: <not set>
Grafana listen port: 11210
Pushgateway settings
Node Exporter settings

Настройка сервиса Monitoring

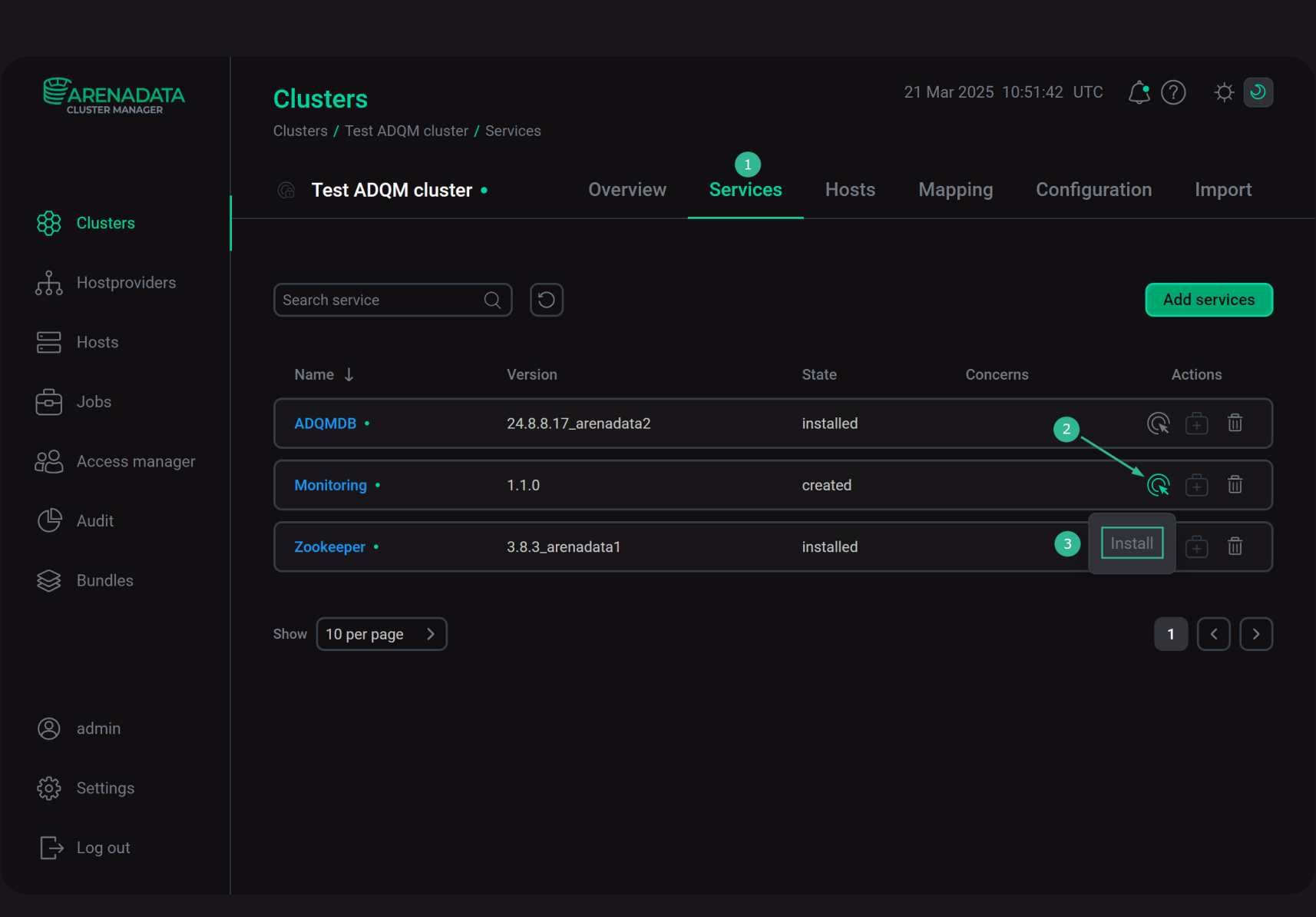
После указания всех необходимых параметров нажмите ***Save***.

ПРИМЕЧАНИЕ

Параметры доступа к метрикам сервисов ADQMDB, Zookeeper, Clickhousekeeper настраиваются в активируемой секции ***Metrics*** конфигурации каждого сервиса. Если не указать эти параметры вручную, сервис Monitoring при установке автоматически включит секцию ***Metrics*** для сервисов ADQM и будет использовать значения параметров по умолчанию.

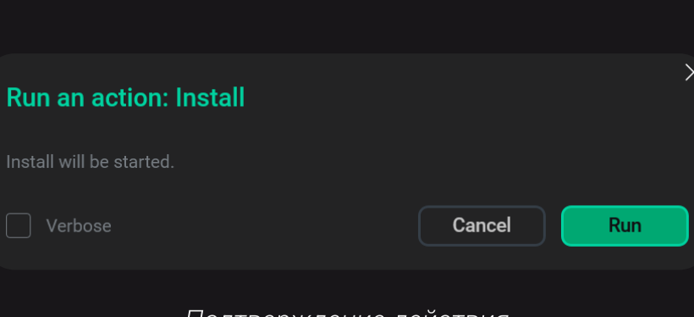
Шаг 4. Установка сервиса

1. На вкладке ***Services*** в строке сервиса Monitoring кликните по иконке в столбце ***Actions*** и запустите действие ***Install***.


Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 10:51:42 UTC
Test ADQM cluster
1
Services
Add services
ADQMDB
24.8.8.17_arenadata2
installed
2
Monitoring
1.1.0
created
3
Install
Zookeeper
3.8.3_arenadata1
installed

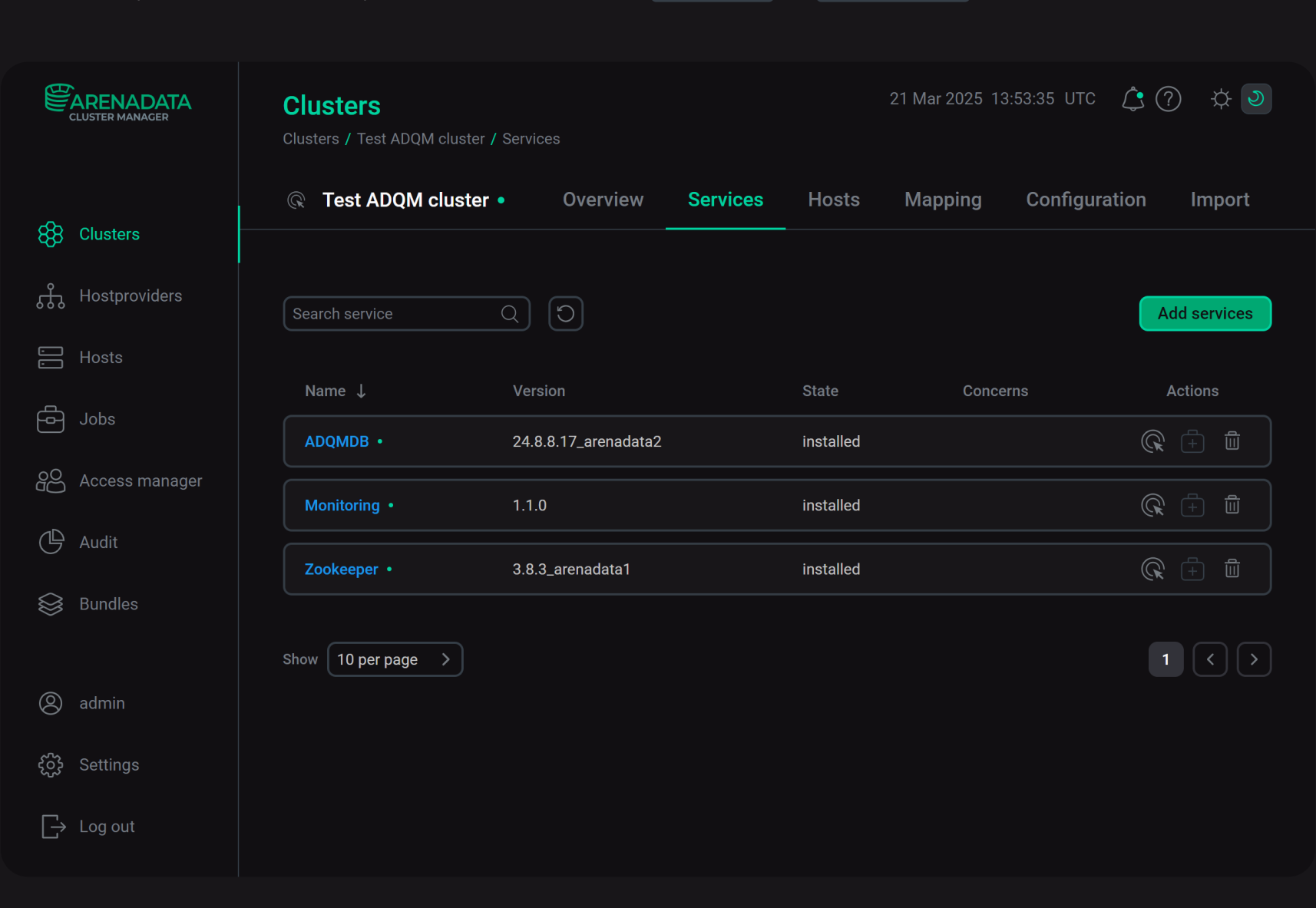
Переход к установке сервиса

2. Подтвердите действие в открывшемся окне.


Run an action: Install
Install will be started.
Verbose
Cancel
Run

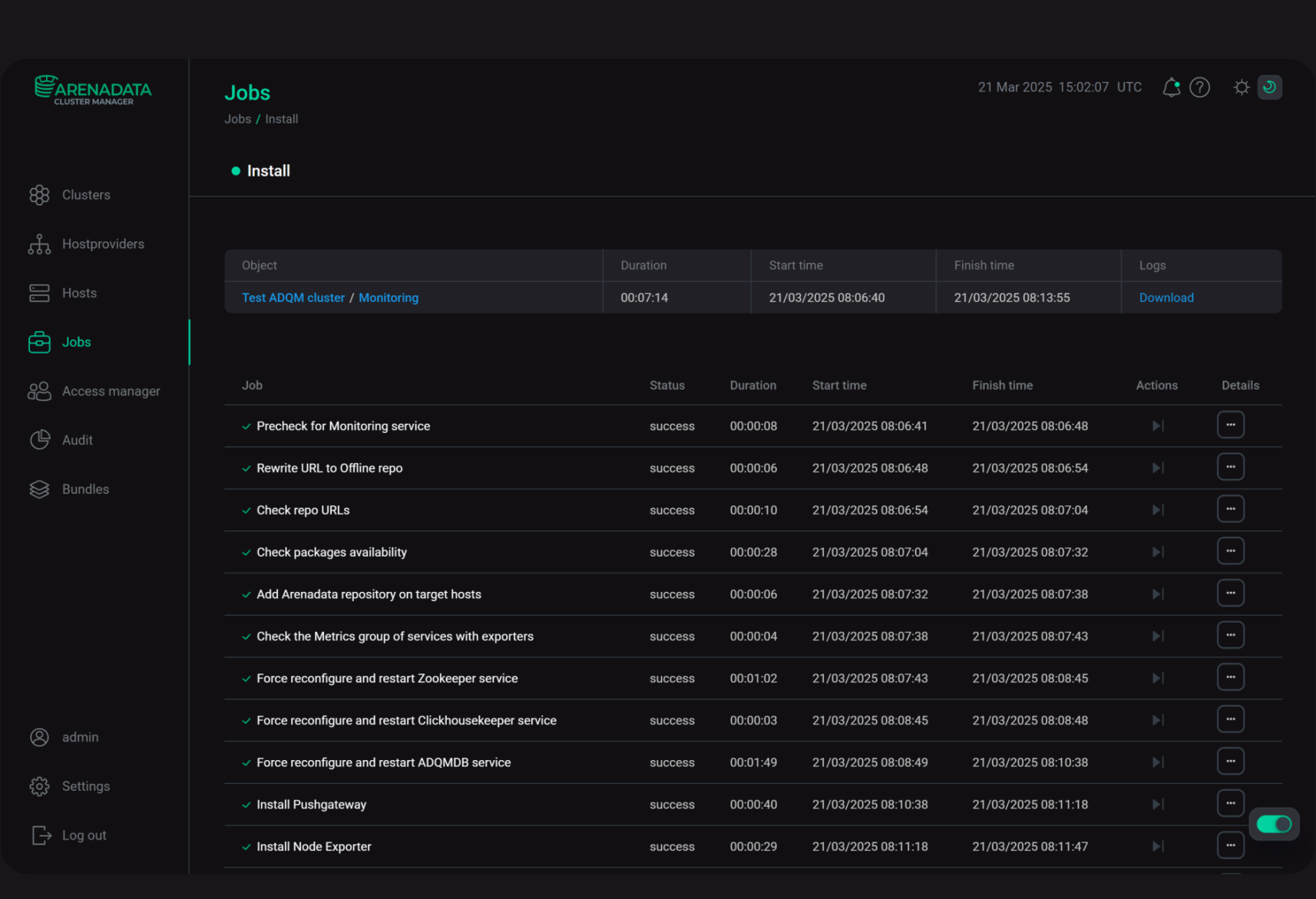
Подтверждение действия

3. Дождитесь завершения установки и убедитесь, что статус сервиса изменился с `created` на `installed`.


Clusters
Clusters / Test ADQM cluster / Services
21 Mar 2025 13:53:35 UTC
Test ADQM cluster
Services
Add services
ADQMDB
24.8.8.17_arenadata2
installed
Monitoring
1.1.0
installed
Zookeeper
3.8.3_arenadata1
installed

Процесс установки завершен

Чтобы посмотреть процесс выполнения установки сервиса и проанализировать ошибки в случае их возникновения, выберите пункт ***Jobs*** в левом навигационном меню и кликните по имени задачи ***Install*** в списке задач ***Jobs***.


Jobs
Jobs / Install
21 Mar 2025 15:02:07 UTC
Install
Download

Object	Duration	Start time	Finish time	Logs
Test ADQM cluster / Monitoring	00:07:14	21/03/2025 08:06:40	21/03/2025 08:13:55	Download

Job	Status	Duration	Start time	Finish time
Precheck for Monitoring service	success	00:00:08	21/03/2025 08:06:41	21/03/2025 08:06:48
Rewrite URL to Offline repo	success	00:00:06	21/03/2025 08:06:48	21/03/2025 08:06:54
Check repo URLs	success	00:00:10	21/03/2025 08:06:54	21/03/2025 08:07:04
Check packages availability	success	00:00:28	21/03/2025 08:07:04	21/03/2025 08:07:32
Add Arenadata repository on target hosts	success	00:00:06	21/03/2025 08:07:32	21/03/2025 08:07:38
Check the Metrics group of services with exporters	success	00:00:04	21/03/2025 08:07:38	21/03/2025 08:07:43
Force reconfigure and restart Zookeeper service	success	00:01:02	21/03/2025 08:07:43	21/03/2025 08:08:45
Force reconfigure and restart Clickhousekeeper service	success	00:00:03	21/03/2025 08:08:45	21/03/2025 08:08:48
Force reconfigure and restart ADQMDB service	success	00:01:49	21/03/2025 08:08:49	21/03/2025 08:10:38
Install Pushgateway	success	00:00:40	21/03/2025 08:10:38	21/03/2025 08:11:18
Install Node Exporter	success	00:00:29	21/03/2025 08:11:18	21/03/2025 08:11:47

Страница задачи Install

Шаг 5. Проверка результатов

Сервис Monitoring запускается автоматически после установки. По завершении установки необходимо убедиться, что с хостов кластера поступают системные метрики и метрики сервисов ADQM — см. Метрики мониторинга кластера ADQM. Чтобы это проверить, можно в браузере посмотреть метрики в формате Prometheus, а также использовать веб-интерфейсы Prometheus и Grafana.

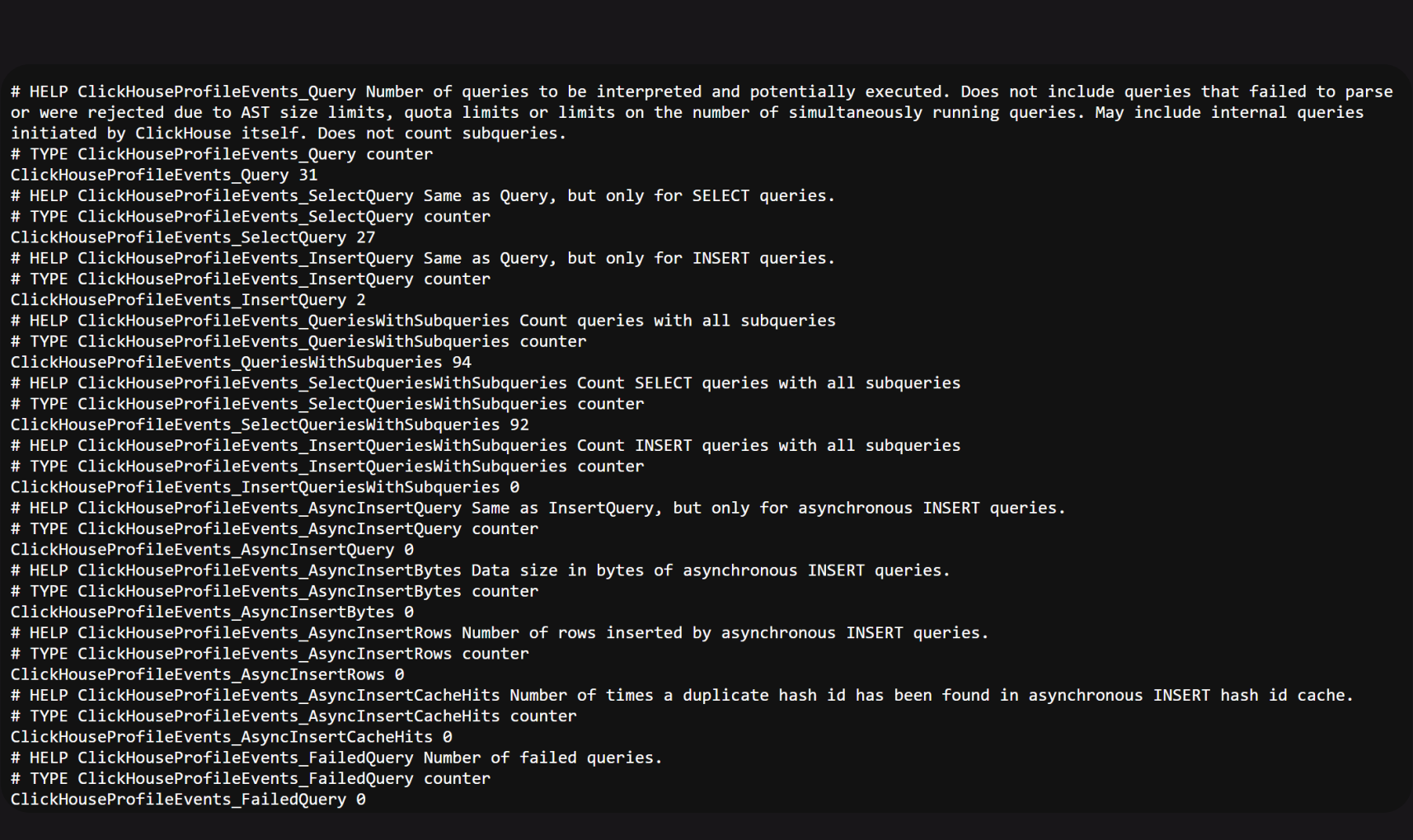
Данные в формате Prometheus

1. В адресную строку браузера введите адрес хоста кластера ADQM, а также укажите порт и конечную точку доступа (endpoint) к сервисным или системным метрикам, которые определяются параметрами:

   - в секции ***Metrics*** конфигурации сервисов ADQMDB, Zookeeper, Clickhousekeeper — настройки доступа к метрикам мониторинга этих сервисов;
   - в секции ***Node Exporter settings*** на странице сервиса Monitoring — настройки доступа к системным метрикам хоста кластера ADQM.

   Пример адреса для просмотра метрик сервера ClickHouse на хосте с IP `10.92.40.25`: *http://10.92.40.25:9363/metrics*.

2. На открывшейся странице будут показаны метрики мониторинга с указанного хоста кластера в формате Prometheus.



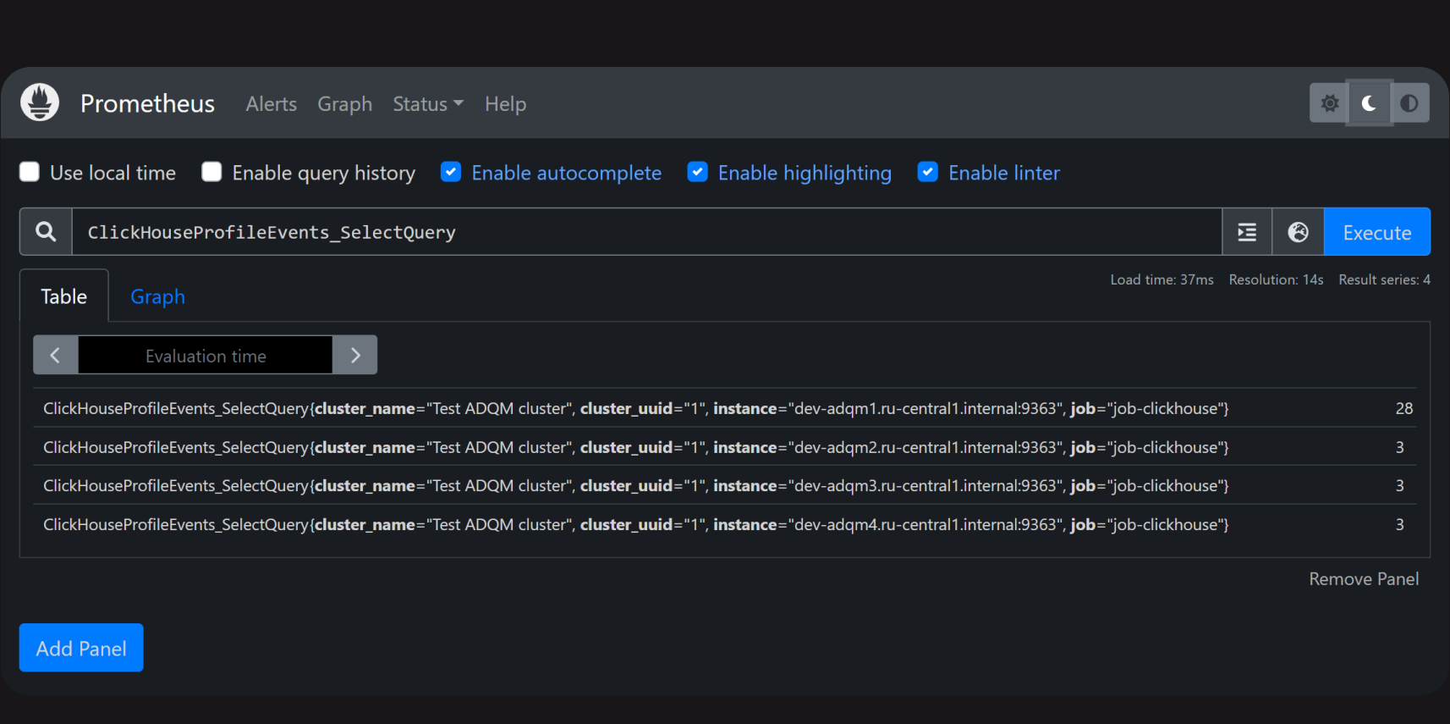
```
# HELP ClickHouseProfileEvents_Query Number of queries to be interpreted and potentially executed. Does not include queries that failed to parse
or were rejected due to AST size limits, quota limits or limits on the number of simultaneously running queries. May include internal queries
initiated by ClickHouse itself. Does not count subqueries.
# TYPE ClickHouseProfileEvents_Query counter
ClickHouseProfileEvents_Query 31
# HELP ClickHouseProfileEvents_SelectQuery Same as Query, but only for SELECT queries.
# TYPE ClickHouseProfileEvents_SelectQuery counter
ClickHouseProfileEvents_SelectQuery 27
# HELP ClickHouseProfileEvents_InsertQuery Same as Query, but only for INSERT queries.
# TYPE ClickHouseProfileEvents_InsertQuery counter
ClickHouseProfileEvents_InsertQuery 2
# HELP ClickHouseProfileEvents_QueriesWithSubqueries Count queries with all subqueries
# TYPE ClickHouseProfileEvents_QueriesWithSubqueries counter
ClickHouseProfileEvents_QueriesWithSubqueries 94
# HELP ClickHouseProfileEvents_SelectQueriesWithSubqueries Count SELECT queries with all subqueries
# TYPE ClickHouseProfileEvents_SelectQueriesWithSubqueries counter
ClickHouseProfileEvents_SelectQueriesWithSubqueries 92
# HELP ClickHouseProfileEvents_InsertQueriesWithSubqueries Count INSERT queries with all subqueries
# TYPE ClickHouseProfileEvents_InsertQueriesWithSubqueries counter
ClickHouseProfileEvents_InsertQueriesWithSubqueries 0
# HELP ClickHouseProfileEvents_AsyncInsertQuery Same as InsertQuery, but only for asynchronous INSERT queries.
# TYPE ClickHouseProfileEvents_AsyncInsertQuery counter
ClickHouseProfileEvents_AsyncInsertQuery 0
# HELP ClickHouseProfileEvents_AsyncInsertBytes Data size in bytes of asynchronous INSERT queries.
# TYPE ClickHouseProfileEvents_AsyncInsertBytes counter
ClickHouseProfileEvents_AsyncInsertBytes 0
# HELP ClickHouseProfileEvents_AsyncInsertRows Number of rows inserted by asynchronous INSERT queries.
# TYPE ClickHouseProfileEvents_AsyncInsertRows counter
ClickHouseProfileEvents_AsyncInsertRows 0
# HELP ClickHouseProfileEvents_AsyncInsertCacheHits Number of times a duplicate hash id has been found in asynchronous INSERT hash id cache.
# TYPE ClickHouseProfileEvents_AsyncInsertCacheHits counter
ClickHouseProfileEvents_AsyncInsertCacheHits 0
# HELP ClickHouseProfileEvents_FailedQuery Number of failed queries.
# TYPE ClickHouseProfileEvents_FailedQuery counter
ClickHouseProfileEvents_FailedQuery 0
```

Метрики сервера ClickHouse в формате Prometheus

Веб-интерфейс Prometheus

1. В адресной строке браузера введите IP-адрес хоста, где установлен компонент Prometheus Server сервиса Monitoring. Укажите номер порта, который определяется параметром ***web.listen-address*** в секции ***Prometheus settings*** в группе ***Service parameters*** конфигурации сервиса Monitoring (по умолчанию, `11200`). Пример адреса: *http://10.92.40.107:11200*. Для входа в интерфейс Prometheus используйте имя пользователя и пароль, которые также указываются в секции ***Prometheus settings*** — настройка ***Prometheus users to login/logout to Prometheus***.

2. В поле ***Expression*** можно ввести название метрики и нажать ***Execute*** — в интерфейсе будут показаны значения этой метрики на всех хостах кластера ADQM.


Prometheus
Alerts
Graph
Status
Help
Use local time
Enable query history
Enable autocomplete
Enable highlighting
Enable linter
Execute
Table
Graph
Load time: 37ms
Resolution: 14s
Result series: 4
Evaluation time
Remove Panel
Add Panel

ClickHouseProfileEvents_SelectQuery

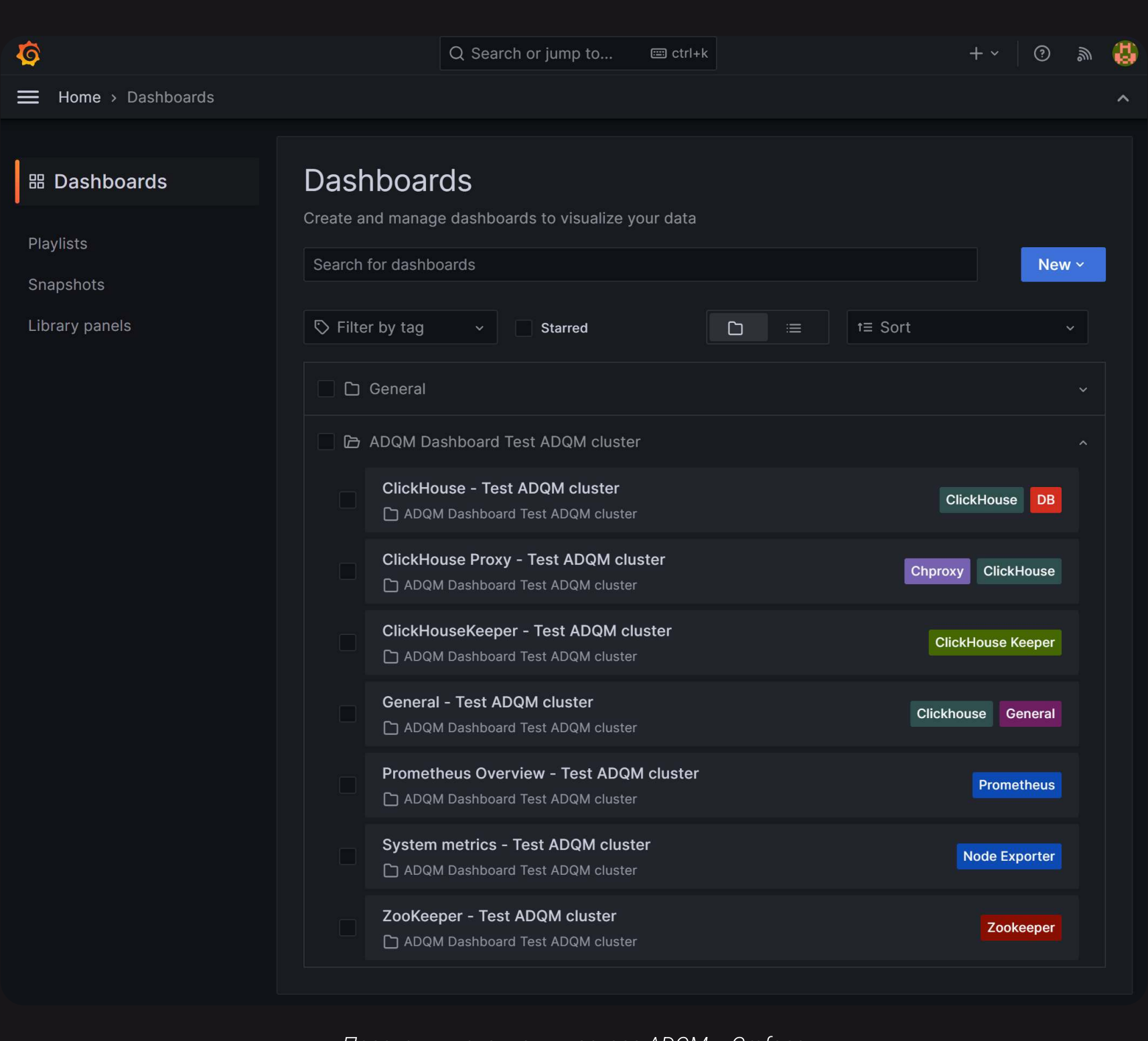
```
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm1.ru-central1.internal:9363", job="job-clickhouse"}   28
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm2.ru-central1.internal:9363", job="job-clickhouse"}   3
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm3.ru-central1.internal:9363", job="job-clickhouse"}   3
ClickHouseProfileEvents_SelectQuery{cluster_name="Test ADQM cluster", cluster_uuid="1", instance="dev-adqm4.ru-central1.internal:9363", job="job-clickhouse"}   3
```

Веб-интерфейс Prometheus

Веб-интерфейс Grafana

1. В адресной строке браузера введите адрес хоста, на котором развернут компонент Grafana сервиса Monitoring. В качестве номера порта укажите значение параметра ***Grafana listen port*** из раздела ***Grafana settings*** в настройках конфигурации сервиса мониторинга (по умолчанию `11210`). Например, *http://10.92.40.107:11210*. Для входа используйте `admin` в качестве имени пользователя и значение параметра ***Grafana administrator's password*** (находится также в разделе настроек ***Grafana settings***) в качестве пароля.

2. В открывшемся окне выберите пункт меню ***Home → Dashboards*** и разверните секцию ***ADQM Dashboard <ADQM_cluster_name>***. В этой секции можно выбрать один из доступных дашбордов для просмотра сервисных или системных метрик, поступающих из вашего кластера ADQM.


Search or jump to...
ctrl+k
Home › Dashboards
Dashboards
Playlists
Snapshots
Library panels
Dashboards
Create and manage dashboards to visualize your data
Search for dashboards
New
Filter by tag
Starred
Sort
General
ADQM Dashboard Test ADQM cluster
ClickHouse - Test ADQM cluster
ClickHouse
DB
ClickHouse Proxy - Test ADQM cluster
Chproxy
ClickHouse
ClickHouseKeeper - Test ADQM cluster
ClickHouse Keeper
General - Test ADQM cluster
Clickhouse
General
Prometheus Overview - Test ADQM cluster
Prometheus
System metrics - Test ADQM cluster
Node Exporter
ZooKeeper - Test ADQM cluster
Zookeeper
ADQM Dashboard Test ADQM cluster

Переход к метрикам кластера ADQM в Grafana

Подключение к ADQM

Елена Дворядкина

Содержание

Назад к содержанию

Обзор

Настройка доступа пользователей к серверам ADQM

Обзор

Существуют различные способы подключения к ADQM. Например, вы можете использовать следующие инструменты:

- Встроенный web-интерфейс ClickHouse
- Клиент командной строки ClickHouse — clickhouse-client
- JDBC-драйвер
- Сторонние клиентские инструменты (например, DBeaver)

ПРИМЕЧАНИЕ

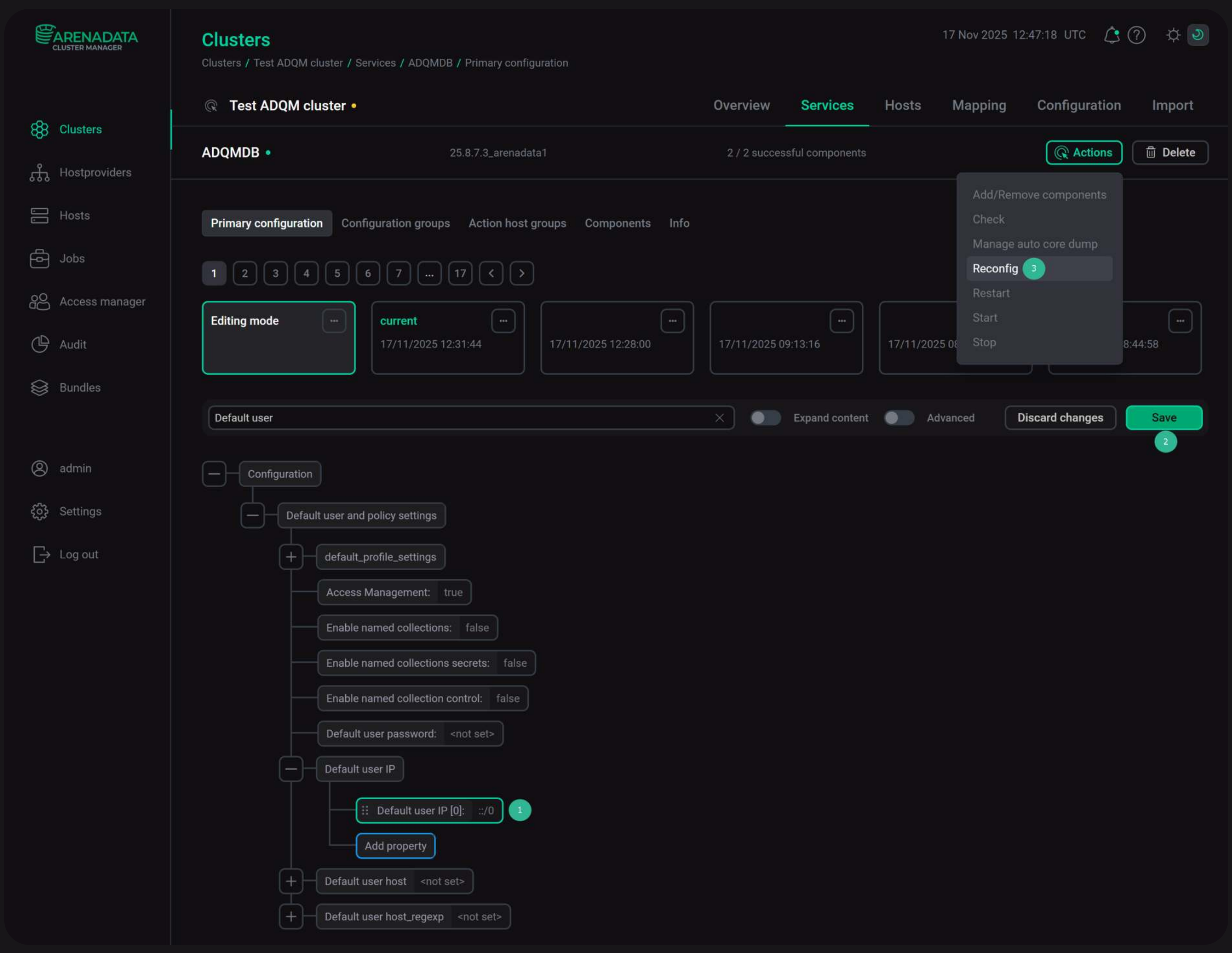
После установки ADQM вы можете сразу начать использовать web-интерфейс ClickHouse и `clickhouse-client`. Если необходимо задействовать сторонние инструменты, установите и настройте их в соответствии с вашими требованиями.

Настройка доступа пользователей к серверам ADQM

Для пользователей ADQM необходимо указать из каких сетей разрешено подключаться к серверам ADQM через web-интерфейс ClickHouse, JDBC-драйвер или сторонние клиенты (в том числе DBeaver).

Например, чтобы открыть пользователю `default` доступ к серверам ADQM с любого IP-адреса, нужно указать `::/0` (или `0.0.0.0/0` в случае отсутствия IPv6) в списке сетей, из которых пользователь может подключаться к ADQM. Это можно сделать одним из следующих способов:

- Добавьте значение `::/0` для параметра ***Default user IP*** в интерфейсе ADCM (***Clusters*** → ***ADQM cluster*** → ***Services*** → ***ADQMDB*** → ***Primary Configuration***) и сохраните изменение.



ARENADATA
CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
17 Nov 2025 12:47:18 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
25.8.7.3_arenadata1
2 / 2 successful components
Actions
Delete
Add/Remove components
Check
Manage auto core dump
Reconfig
3
Restart
Start
Stop
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
Action host groups
Components
Info
1
2
3
4
5
6
7
...
17
Editing mode
current
17/11/2025 12:31:44
17/11/2025 12:28:00
17/11/2025 09:13:16
17/11/2025 0
8:44:58
Default user
Expand content
Advanced
Discard changes
Save
2
Configuration
Default user and policy settings
default_profile_settings
Default user IP
1
Add property

Access Management: true

Enable named collections: false

Enable named collections secrets: false

Enable named collection control: false

Default user password: <not set>

Default user IP [0]: ::/0

Default user host <not set>

Default user host_regexp <not set>

Настройка IP-адресов с доступом к серверам ADQM

- Добавьте элемент `<ip>::/0</ip>` в секцию `networks` файла *users.xml*:

```
<users>
    <default>
        <password></password>
        <access_management>1</access_management>
        <networks incl="networks" replace="replace">
            <ip>::/0</ip>
            ...
        </networks>
        <profile>default</profile>
        <quota>default</quota>
    </default>
</users>
```

ВАЖНО

Независимо от выбранного способа — выполните действие ***Reconfig*** для сервиса ADQMDB в интерфейсе ADCM, чтобы применить изменения.

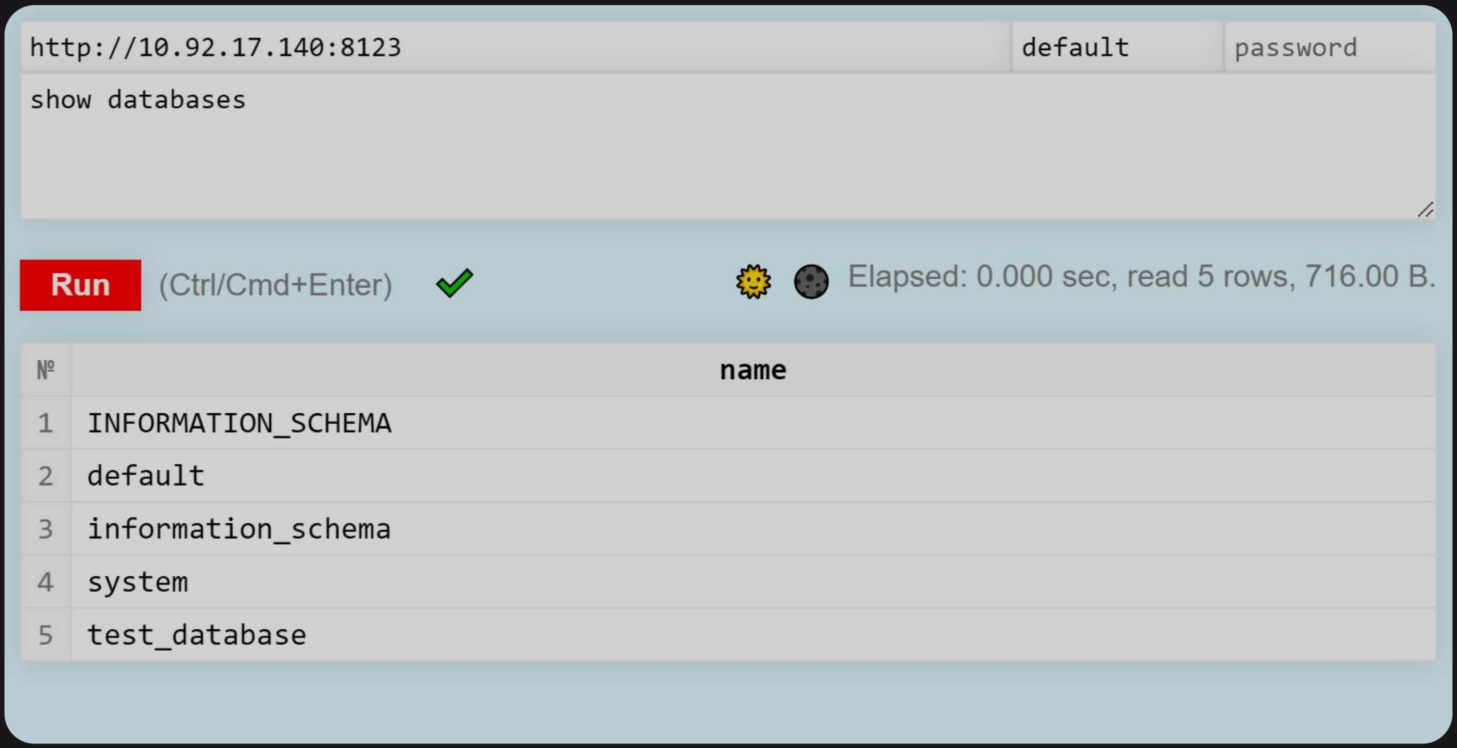
Подключение к ADQM через встроенный web-интерфейс ClickHouse

Елена Дворядкина

Назад к содержанию

Для работы с базами данных ADQM можно использовать встроенный web-интерфейс ClickHouse, доступный по адресу *http://<IP>:8123/play*, где `<IP>` — IP-адрес сервера ADQM. В полях ввода ***user*** и ***password***, расположенных справа в верхней части web-интерфейса, задается имя пользователя и пароль для подключения к серверу ADQM (пользователь должен быть определен в секции `users` конфигурационного файла сервера *users.xml*). По умолчанию используется имя пользователя `default` без пароля.

Введите запрос в основное текстовое поле интерфейса и нажмите кнопку ***Run*** для его выполнения.



http://10.92.17.140:8123 | default | password

```
show databases
```

Elapsed: 0.000 sec, read 5 rows, 716.00 B.

№	name
1	INFORMATION_SCHEMA
2	default
3	information_schema
4	system
5	test_database

Web-интерфейс ClickHouse

ПРИМЕЧАНИЕ

Для пользователей ADQM должно быть указано, из каких сетей разрешено подключение. В статье Подключение к ADQM приведен пример, как открыть пользователю `default` доступ к серверам ADQM через web-интерфейс ClickHouse с любого IP-адреса.

Использование клиента командной строки ClickHouse — clickhouse-client

Елена Дворядкина

Содержание

Назад к содержанию

Интерактивный режим

Пакетный режим

Формат вывода результатов запроса

Для подключения к ADQM можно использовать стандартный консольный клиент ClickHouse — `clickhouse-client`. После установки ADQM `clickhouse-client` доступен на каждом сервере кластера. Чтобы начать с ним работу, необходимо выполнить следующую команду:

```
$ clickhouse-client
```

Эта команда осуществляет соединение с локальным сервером ADQM (`localhost:9000`) от имени пользователя `default` без пароля.

```
$ clickhouse-client
ClickHouse client version 22.8.4.7.
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 22.8.4 revision 54460.

:)
```

Чтобы подключиться к удаленному серверу ADQM или задать имя пользователя, пароль и базу данных, отличные от значений по умолчанию, можно использовать следующие параметры команды `clickhouse-client`:

- `--host, -h` — имя хоста, на котором расположен сервер ADQM;
- `--port` — порт для подключения;
- `--user, -u` — имя пользователя, от имени которого выполняется подключение;
- `--password` — пароль;
- `--ask-password` — предложение пользователю ввести пароль;
- `--database, -d` — текущая база данных. Если значение не указано, текущей будет база данных по умолчанию, которая создается при установке ADQM с помощью конфигурационного параметра `Default database` (***ADQMDB*** *→* ***Other*** *→* ***Default database***).

Клиент может использоваться в интерактивном или пакетном (batch) режиме.

Создание таблиц для выполнения приведенных ниже примеров

Чтобы воспроизвести приведенные ниже примеры, создайте предварительно следующие таблицы:

- `test_table1`

```
CREATE TABLE test_table1 (a Int32, b String, c Int32) ENGINE = MergeTree ORDER BY a;
```

```
INSERT INTO test_table1 VALUES (1, 'aaa', 100), (2, 'bbb', 200), (3, 'ccc', 300), (4, 'ddd', 400), (5, 'eee', 500);
```

- `test_table2`

```
CREATE TABLE test_table2 (a Int32, b String, c Int32) ENGINE = MergeTree ORDER BY a;
```

```
INSERT INTO test_table2 VALUES (1, 'a', 0), (2, 'b', 0), (3, 'c', 0);
```

Интерактивный режим

После выполнения команды `clickhouse-client` терминал переключается в интерактивный режим работы с сервером ADQM и отображает приглашение для ввода запроса — `:)`. Выполнение запроса осуществляется нажатием `Enter` (точка с запятой в конце запроса необязательна).

При выполнении команды `clickhouse-client --multiline` активируется многострочный режим ввода запроса. В этом случае чтобы выполнить запрос, необходимо завершить его точкой с запятой и нажать `Enter`. Если в конце введенной строки нет точки с запятой, при нажатии `Enter` предлагается ввести следующую строку запроса.

Ниже приведен пример ввода и выполнения запроса в интерактивном режиме консольного клиента.

Запрос:

```
SELECT * FROM test_table1;
```

Вывод на экран:

```
SELECT *
FROM test_table1

Query id: 917d68af-6989-409e-a30f-89a657487e22

--a---b-------c--
│ 1 │ aaa │ 100 │
│ 2 │ bbb │ 200 │
│ 3 │ ccc │ 300 │
│ 4 │ ddd │ 400 │
│ 5 │ eee │ 500 │
-----------------

5 rows in set. Elapsed: 0.001 sec.
```

При выполнении запроса клиент выводит на экран:

- Прогресс выполнение запроса (для быстрых запросов может не отобразиться).
- Отформатированный запрос после его парсинга.
- ID запроса.
- Результат в заданном формате. По умолчанию `clickhouse-client` в интерактивном режиме использует формат `PrettyCompact`.
- Количество строк результата и время выполнения.

Чтобы выйти из интерактивного режима консольного клиента, можно нажать `Ctrl+D` или ввести вместо запроса `exit`, `quit`, `q` или `logout`.

Пакетный режим

Клиент также можно использовать в пакетном (не интерактивном) режиме. Для этого можно:

- Передать запрос в параметр `--query` команды `clickhouse-client`:

```
$ clickhouse-client --query "SELECT * FROM test_table2"
```

Пример вывода на экран:

```
1       a       0
2       b       0
3       c       0
```

- Отправить запрос в stdin:

```
$ echo "SELECT * FROM test_table2" | clickhouse-client
```

Пример вывода на экран:

```
1       a       0
2       b       0
3       c       0
```

- Одновременно использовать параметр `--query` и stdin. Например, для запросов `INSERT` можно начало запроса написать в параметре `query`, а вставляемые данные передать в stdin:

```
$ echo "(4,'d',4)" | clickhouse-client --query "INSERT INTO test_table2 VALUES"
$ clickhouse-client --query "SELECT * FROM test_table2"
```

Пример вывода на экран:

```
1       a       0
2       b       0
3       c       0
4       d       4
```

По умолчанию в пакетном режиме можно выполнить только один запрос. Чтобы разрешить выполнение нескольких запросов, используется параметр `--multiquery` (работает для всех запросов кроме `INSERT`). В этом случае в параметр `--query` можно передать строку, включающую несколько запросов, разделенных точкой с запятой. Например:

```
$ clickhouse-client --multiquery --query "select * from test_table1;select * from test_table2;"
```

В пакетном режиме для вывода результатов запроса по умолчанию используется формат TabSeparated. При выполнении нескольких запросов с использованием опции `--multiquery` результаты запросов выводятся друг за другом без дополнительных разделителей. В следующем разделе приведены способы изменения формата вывода данных.

Формат вывода результатов запроса

Формат вывода результатов запроса можно задать следующими способами:

- Указать нужный формат с помощью выражения `FORMAT` в запросе (указанный формат будет применяться для вывода результата данного запроса) или аргумента командной строки `--format` (этот формат будет использоваться по умолчанию для всех запросов).
- Указать `\G` в конце запроса или использовать параметр командной строки `--vertical`, чтобы использовать формат Vertical для отдельного запроса или всех запросов по умолчанию.
- Задать формат в конфигурационном файле клиента (*/etc/clickhouse-client/config.xml*).

Пример изменения формата вывода данных в пакетном режиме:

```
$ clickhouse-client --multiquery --query "select * from test_table1;select * from test_table2;" --format PrettyCompact
```

Вывод на экран:

```
--a---b-------c--
│ 1 │ aaa │ 100 │
│ 2 │ bbb │ 200 │
│ 3 │ ccc │ 300 │
│ 4 │ ddd │ 400 │
│ 5 │ eee │ 500 │
-----------------

--a---b---c--
│ 1 │ a │ 0 │
│ 2 │ b │ 0 │
│ 3 │ c │ 0 │
│ 4 │ d │ 4 │
-------------
```

Полную информацию об особенностях использования `clickhouse-client` можно получить в статье clickhouse-client документации ClickHouse.

Использование JDBC-драйвера для подключения JVM-based приложений к ADQM

Елена Дворядкина

Назад к содержанию

Для подключения JVM-based приложения к базе данных ADQM можно использовать JDBC-драйвер.

Ниже приведен пример приложения Java, которое собирается с помощью Maven и взаимодействует с сервером ADQM через JDBC-драйвер. Исходный код приложения находится в файле *<project_path>/src/main/java/io/arenadata/adqm/test/TestApp.java*, а файл *pom.xml* с описанием структуры проекта — в корневой папке проекта *<project_path>/pom.xml*.

1. В файле *pom.xml* добавьте следующую зависимость, чтобы подключить JDBC-драйвер из центрального репозитория Maven:

```
<dependencies>
    <dependency>
        <groupId>com.clickhouse</groupId>
        <artifactId>clickhouse-jdbc</artifactId>
        <version>0.3.2</version>
        <classifier>all</classifier>
        <exclusions>
            <exclusion>
                <groupId>*</groupId>
                <artifactId>*</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>
```

2. Создайте основной класс приложения, используя следующий код:

```
package io.arenadata.adqm.test;

import com.clickhouse.jdbc.*;
import java.sql.*;
import java.util.*;

public class TestApp {
    private static Connection getConnection(String url) throws SQLException {
        return DriverManager.getConnection(url);
    }

    public static void main(String[] args) {
        String url = "jdbc:clickhouse://<IP>:8123/default";
        try (
            Connection connection = getConnection(url);
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM <table_name> LIMIT 5")) {
                ResultSetMetaData rsMetaData = rs.getMetaData();
                int columns = rsMetaData.getColumnCount();
                while (rs.next()) {
                    for (int c = 1; c <= columns; c++) {
                        System.out.print(rsMetaData.getColumnName(c) + ":" + rs.getString(c) + (c < columns ? " | " : "\n"));
                    }
                }
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

   Этот код устанавливает соединение с сервером ADQM через DriverManager (от имени пользователя `default` без пароля) и выводит первые пять строк заданной таблицы базы данных `default` в выходной поток. Замените `<IP>` на IP-адрес вашего сервера ADQM и вместо `<table_name>` в строке запроса введите название таблицы, существующей в базе данных `default` на вашем сервере ADQM.

3. Для пользователя `default` укажите из каких сетей разрешено подключение к ADQM (см. пример в статье Подключение к ADQM).

4. Соберите и запустите проект. Пример вывода на экран:

```
a:1 | b:text1 | c:100
a:2 | b:text2 | c:200
a:3 | b:text3 | c:300
```

Использование DBeaver для подключения к ADQM

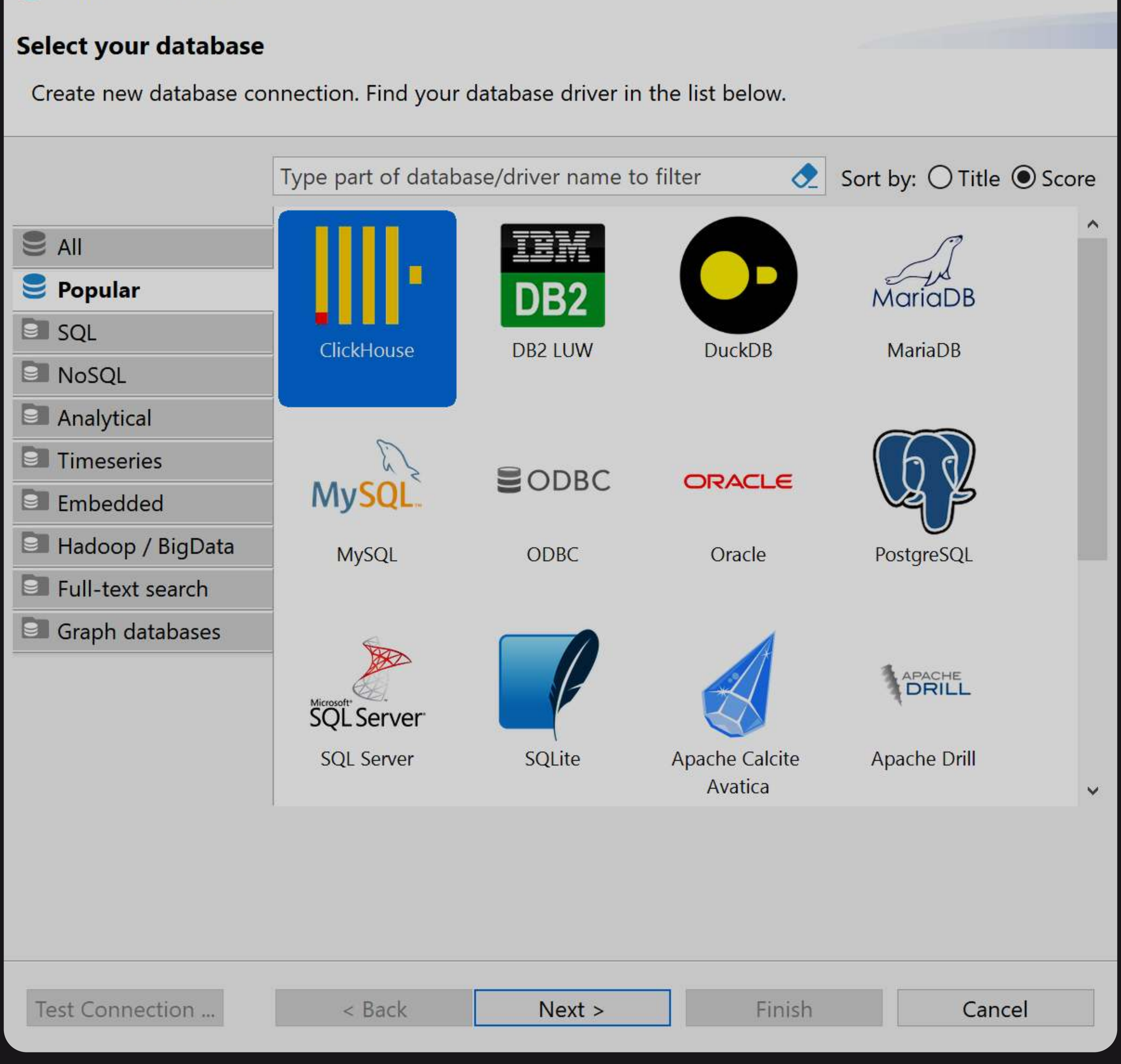
Елена Дворядкина

Назад к содержанию

Для подключения к ADQM можно использовать DBeaver — сторонний клиентский инструмент, который поддерживает различные типы баз данных. Информацию о том, как установить DBeaver, можно найти по ссылке Installation.

Чтобы подключиться к серверу ADQM, запустите DBeaver и выполните следующие шаги.

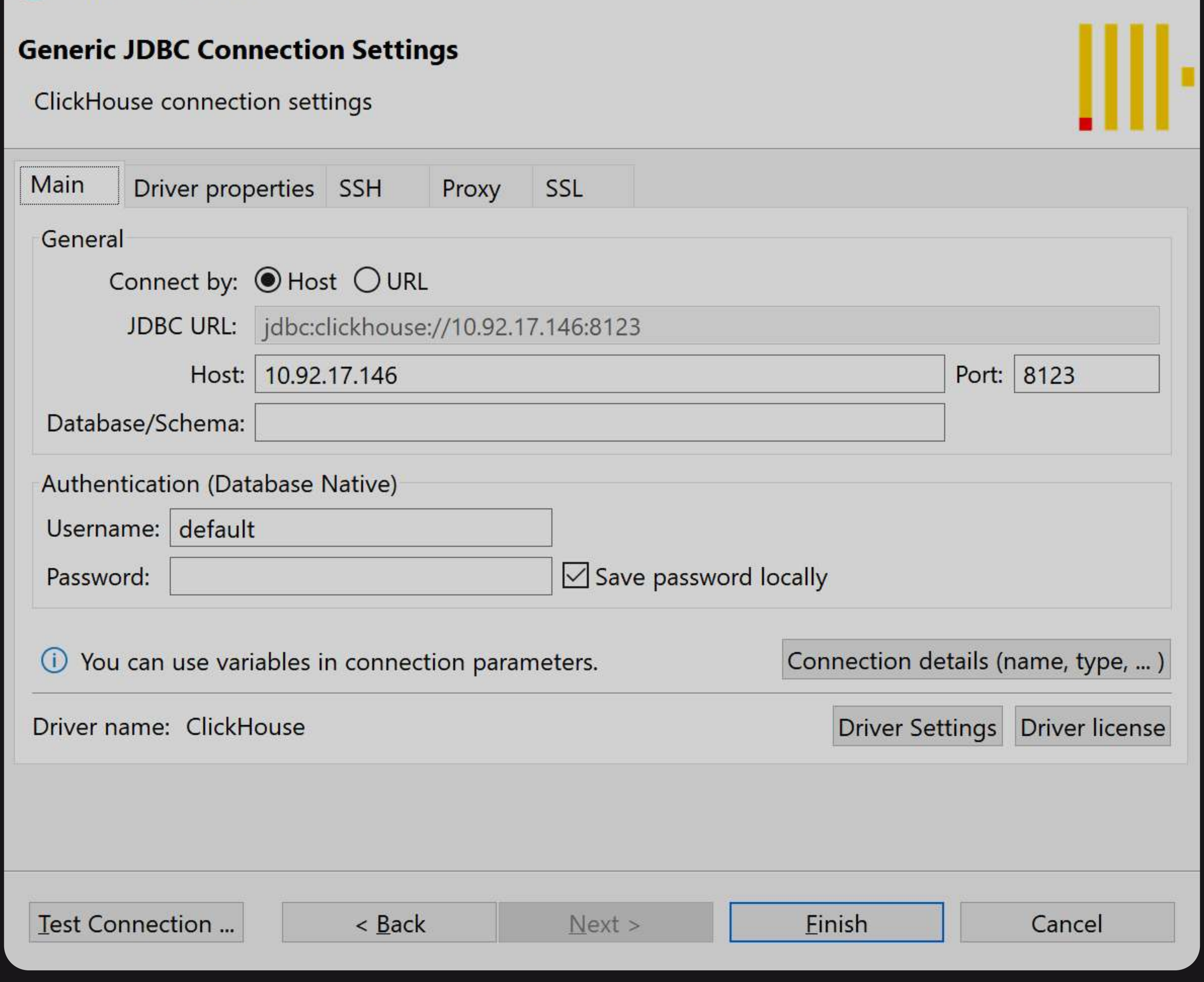
1. Выберите ***ClickHouse*** в окне ***Connect to a database*** (***Database*** → ***New Database Connection***).


Connect to a database
Select your database
Create new database connection. Find your database driver in the list below.
Type part of database/driver name to filter
Sort by: ○ Title ● Score
All
Popular
SQL
NoSQL
Analytical
Timeseries
Embedded
Hadoop / BigData
Full-text search
Graph databases
ClickHouse
DB2 LUW
DuckDB
MariaDB
MySQL
ODBC
Oracle
PostgreSQL
SQL Server
SQLite
Apache Calcite Avatica
Apache Drill
Test Connection ...
< Back
Next >
Finish
Cancel

Выбор СУБД

2. В настройках подключения укажите хост ADQM, имя пользователя и пароль (пользователь должен быть определен в секции `users` конфигурационного файла сервера *users.xml*). По умолчанию используется имя пользователя `default` без пароля. Убедитесь, что в конфигурации пользователя определены сети, из которых разрешено подключение к ADQM. В статье Подключение к ADQM описан пример, как открыть пользователю `default` доступ к серверам ADQM через DBeaver с любого IP-адреса.

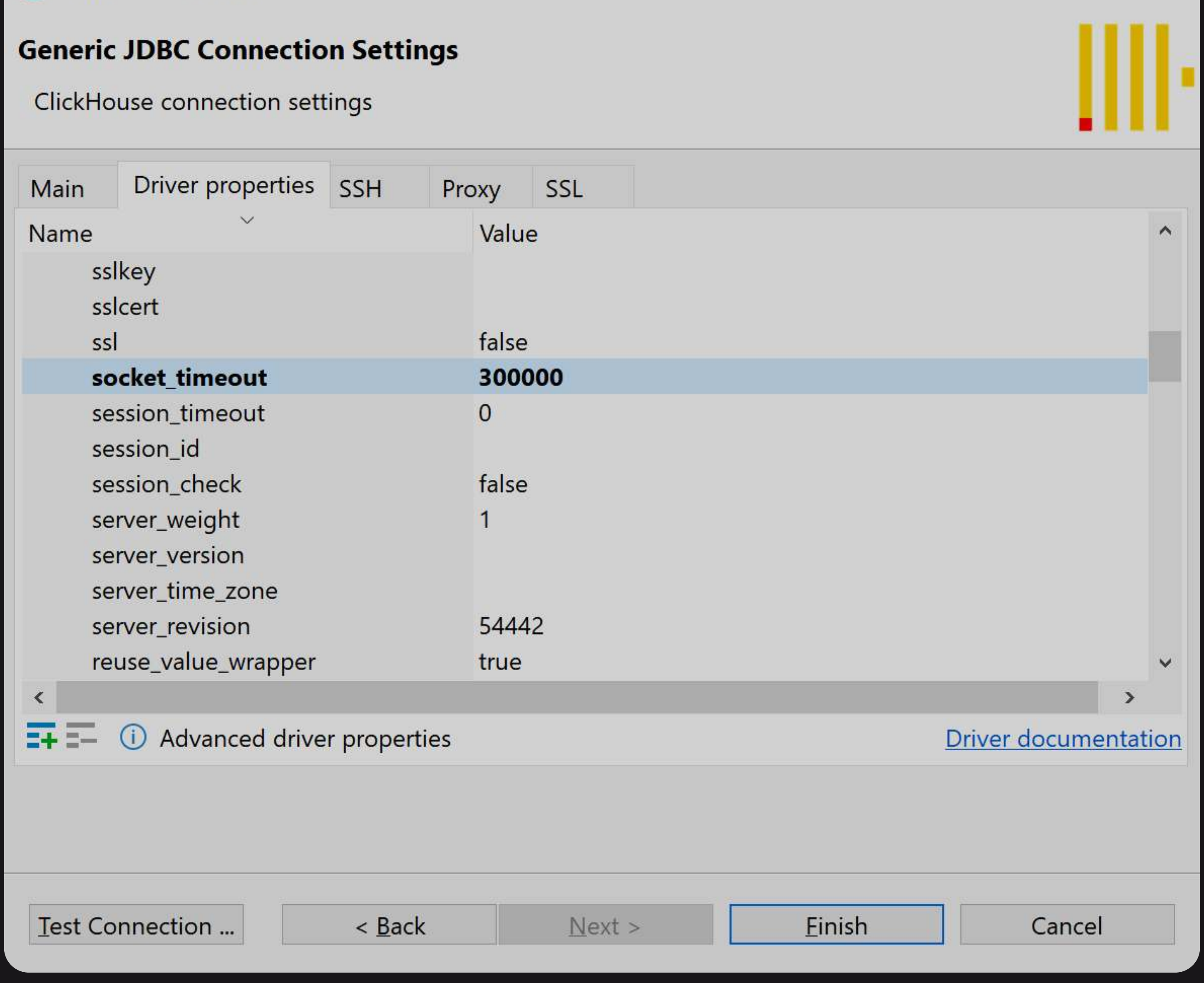
   В поле ***Database/Schema*** введите имя базы данных (если не указано, будет выполняться подключение к базе данных `default`).


Connect to a database
Generic JDBC Connection Settings
ClickHouse connection settings
Main
Driver properties
SSH
Proxy
SSL
General
Authentication (Database Native)
You can use variables in connection parameters.
Connection details (name, type, ...)
Driver Settings
Driver license
Test Connection ...
< Back
Next >
Finish
Cancel

Connect by: ● Host ○ URL
JDBC URL: jdbc:clickhouse://10.92.17.146:8123
Host: 10.92.17.146
Port: 8123
Database/Schema:
Username: default
Password:
☑ Save password locally
Driver name: ClickHouse

Настройки подключения

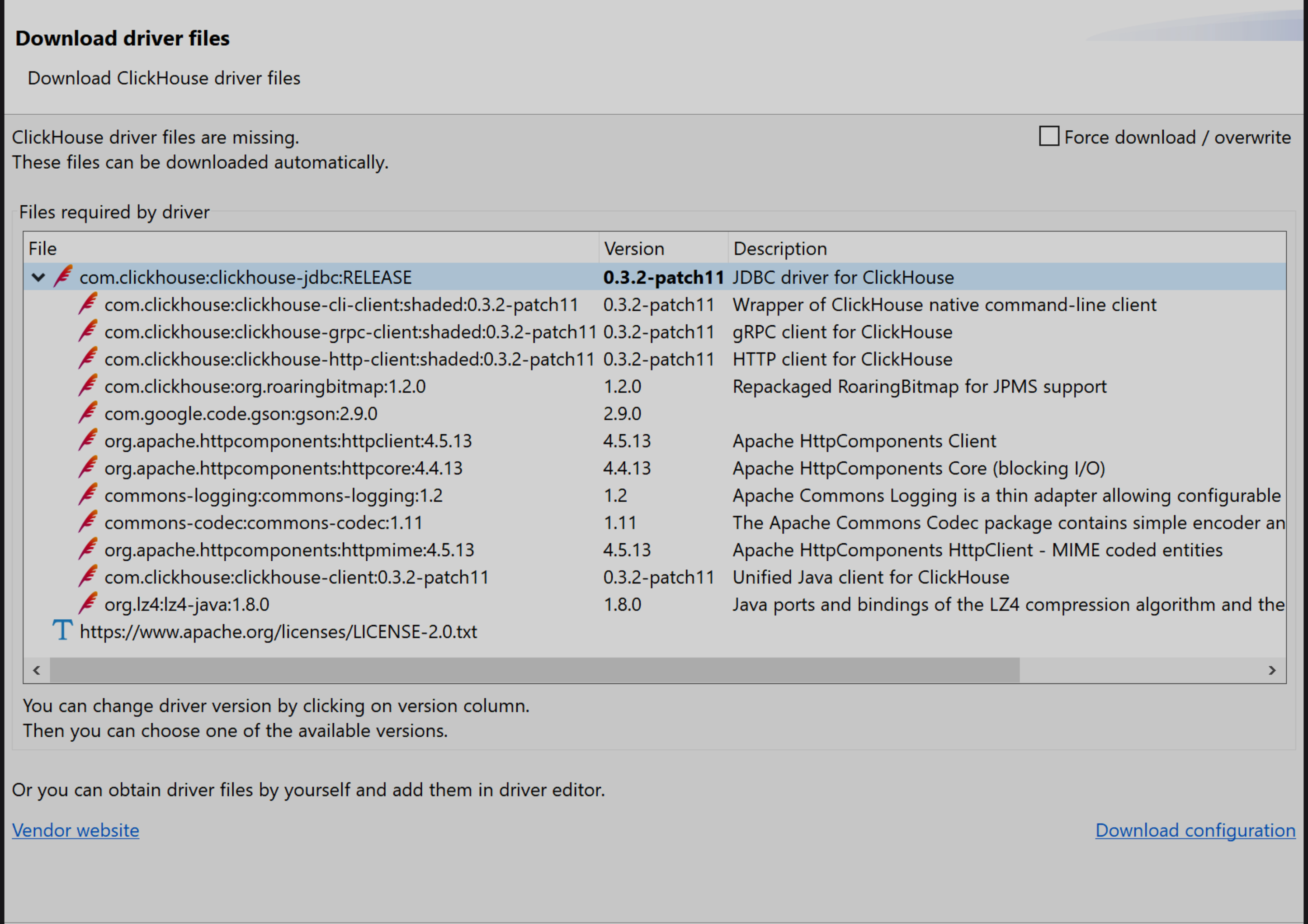
3. Если нужно будет работать с аналитическими запросами, которые обрабатываются дольше 30 секунд, рекомендуется увеличить значение параметра ***socket_timeout*** на вкладке ***Driver properties*** (например, до `300000` миллисекунд).


Connect to a database
Generic JDBC Connection Settings
ClickHouse connection settings
Main
Driver properties
SSH
Proxy
SSL
Advanced driver properties
Driver documentation
Test Connection ...
< Back
Next >
Finish
Cancel

Name	Value
sslkey	
sslcert	
ssl	false
socket_timeout	300000
session_timeout	0
session_id	
session_check	false
server_weight	1
server_version	
server_time_zone	
server_revision	54442
reuse_value_wrapper	true

Настройки драйвера

4. При первом подключении к серверу ADQM необходимо установить JDBC-драйвер.


Driver settings
Download driver files
Download ClickHouse driver files
Edit Driver
Download
Cancel
Vendor website
Download configuration

ClickHouse driver files are missing.
These files can be downloaded automatically.

☐ Force download / overwrite

Files required by driver

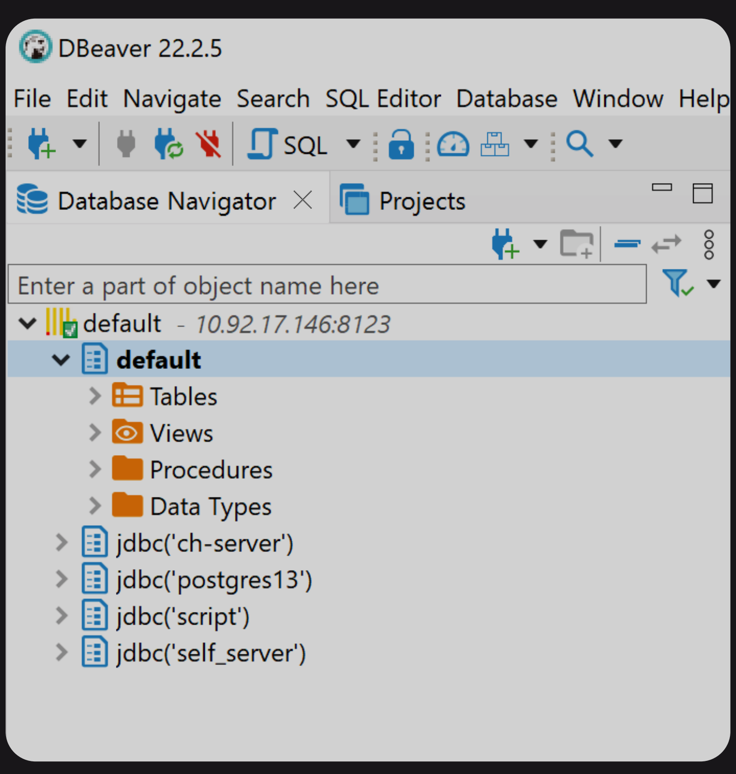
File	Version	Description
com.clickhouse:clickhouse-jdbc:RELEASE	0.3.2-patch11	JDBC driver for ClickHouse
com.clickhouse:clickhouse-cli-client:shaded:0.3.2-patch11	0.3.2-patch11	Wrapper of ClickHouse native command-line client
com.clickhouse:clickhouse-grpc-client:shaded:0.3.2-patch11	0.3.2-patch11	gRPC client for ClickHouse
com.clickhouse:clickhouse-http-client:shaded:0.3.2-patch11	0.3.2-patch11	HTTP client for ClickHouse
com.clickhouse:org.roaringbitmap:1.2.0	1.2.0	Repackaged RoaringBitmap for JPMS support
com.google.code.gson:gson:2.9.0	2.9.0	
org.apache.httpcomponents:httpclient:4.5.13	4.5.13	Apache HttpComponents Client
org.apache.httpcomponents:httpcore:4.4.13	4.4.13	Apache HttpComponents Core (blocking I/O)
commons-logging:commons-logging:1.2	1.2	Apache Commons Logging is a thin adapter allowing configurable
commons-codec:commons-codec:1.11	1.11	The Apache Commons Codec package contains simple encoder an
org.apache.httpcomponents:httpmime:4.5.13	4.5.13	Apache HttpComponents HttpClient - MIME coded entities
com.clickhouse:clickhouse-client:0.3.2-patch11	0.3.2-patch11	Unified Java client for ClickHouse
org.lz4:lz4-java:1.8.0	1.8.0	Java ports and bindings of the LZ4 compression algorithm and the
https://www.apache.org/licenses/LICENSE-2.0.txt		

You can change driver version by clicking on version column.
Then you can choose one of the available versions.

Or you can obtain driver files by yourself and add them in driver editor.

Установка драйвера ClickHouse

После установки соединения DBeaver отобразит базу данных на вкладке ***Database Navigator***.


DBeaver 22.2.5
File
Edit
Navigate
Search
SQL Editor
Database
Window
Help
SQL
Database Navigator
Projects
Enter a part of object name here

- default - 10.92.17.146:8123
  - default
    - Tables
    - Views
    - Procedures
    - Data Types
  - jdbc('ch-server')
  - jdbc('postgres13')
  - jdbc('script')
  - jdbc('self_server')

Вкладка Database Navigator

Аутентификация

Елена Дворядкина

Назад к содержанию

В процессе аутентификации ADQM идентифицирует пользователя и определяет, разрешено ли ему подключаться к серверу. Для управления правами доступа ADQM использует роли.

Поддерживаются следующие методы аутентификации:

- По паролю — пароли могут быть захешированы с помощью алгоритмов SHA-256 и double SHA-1 или храниться в незашифрованном виде.
- LDAP — сервер LDAP может использоваться как внешний аутентификатор для верификации паролей пользователей ADQM или как внешний каталог для хранения информации об учетных записях пользователей и их правах доступа.
- По сертификатам SSL — аутентификация пользователей может осуществляться по пользовательским сертификатам SSL, если для ADQM включена защита канала по протоколу SSL.
- Kerberos — пользователи, аутентифицируемые по протоколу Kerberos, могут работать с ADQM через интерфейс HTTP (используемый клиент должен поддерживать механизм SPNEGO).

Аутентификация по паролю

Елена Дворядкина

Содержание

Назад к содержанию

- Установка паролей пользователей
  - sha256_password
  - double_sha1_password
  - plaintext_password
  - no_password
- Изменение паролей пользователей
- Изменение пароля пользователя default

Пользователи ADQM могут быть аутентифицированы по паролю. Поддерживаются следующие методы аутентификации по паролю, которые отличаются тем, как пароль хранится на сервере ADQM и передается во время соединения:

- sha256_password — ADQM хранит хеш пароля, вычисленный по алгоритму SHA-256. Для дополнительной защиты можно использовать **соль** (SALT) — случайную строку, добавляемую в начало или конец пароля перед тем, как передать его в хеш-функцию.
- double_sha1_password — ADQM хранит пароль в виде хеша, вычисленного по алгоритму double SHA-1 (для совместимости с MySQL).
- plaintext_password — пароль хранится в незашифрованном виде. Поскольку этот метод является небезопасным, используйте его только если соединение с ADQM защищено шифрованием SSL.
- no_password — пользователь может подключаться к ADQM без пароля.

Пароль пользователя назначается при выполнении запроса `CREATE USER` / `ALTER USER` или через конфигурационный файл *users.xml*.

Установка паролей пользователей

sha256_password

Чтобы использовать алгоритм SHA-256 для хеширования пароля пользователя, можно указать его одним из следующих способов:

- назначить пароль в виде строки — ADQM автоматически вычислит хеш для введенной строки с помощью алгоритма SHA-256 и сохранит его;
- передать в ADQM хеш, предварительно получив его для строки пароля.

Указать строку пароля

Используйте один из следующих запросов, чтобы назначить пароль пользователю в виде строки:

```
CREATE USER <user_name> IDENTIFIED WITH sha256_password BY '<user_password>';
```

```
CREATE USER <user_name> IDENTIFIED BY '<user_password>'
```

При выполнении этих запросов ADQM добавит произвольное значение SALT к введенному паролю пользователя, захеширует получившееся значение и сохранит хеш и значение SALT.

Пример

Запрос:

```
CREATE USER andrew IDENTIFIED BY 'qwerty';
```

Вывод на экран:

```
CREATE USER andrew IDENTIFIED WITH sha256_hash BY 'E422256F5E13AB9C590BAFE5A91592BCF2417E8F992D1C74B9CF138C4D3646FF' SALT
'1EE6C0E7CA2F95253F6F0A5C2855AB4D7C00F55AB0BA83E5AB0ED505F168CC06'

Query id: e1de17ec-74fb-4b0a-ad25-e5ef2199d727

Ok.

0 rows in set. Elapsed: 0.001 sec.
```

Чтобы подключаться к ADQM от имени этого пользователя, необходимо указать заданный пароль:

```
$ clickhouse-client --user andrew --password qwerty
```

Указать хеш строки пароля

Чтобы назначить пароль пользователю в виде хеша, используйте следующий запрос:

```
CREATE USER <user_name> IDENTIFIED WITH sha256_hash BY '<hash>';
```

Если при вычислении хеша использовалась соль, хеш и соль передаются в запрос следующим образом:

```
CREATE USER <user_name> IDENTIFIED WITH sha256_hash BY '<hash>' SALT '<salt>';
```

Пример

Сгенерируйте хеш для строки `qwerty`:

```
$ PASSWORD="qwerty"; echo "$PASSWORD"; echo -n "$PASSWORD" | sha256sum | tr -d '-'
```

Вывод на экран (первая строка — пароль, вторая — хеш):

```
qwerty
65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5
```

При создании учетной записи пользователя передайте сгенерированный хеш:

```
CREATE USER alex IDENTIFIED WITH sha256_hash BY '65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5';
```

Убедитесь, что можно подключиться к ADQM от имени созданного пользователя, используя пароль `qwerty`:

```
$ clickhouse-client --user alex --password qwerty
```

Указать хеш SHA-256 можно также через конфигурационный файл *users.xml*, поместив его в элемент `password_sha256_hex`:

```
<users>
    <user_name>
        <password_sha256_hex>65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5</password_sha256_hex>
        <!-- Other settings -->
    </user_name>
</users>
```

double_sha1_password

Чтобы использовать алгоритм double SHA-1 для хеширования пароля пользователя, можно:

- указать пароль в виде строки — ADQM автоматически вычислит хеш для введенной строки с помощью алгоритма double SHA-1 и сохранит его;
- передать в ADQM хеш, предварительно получив его для строки пароля.

Указать строку пароля

Используйте следующий запрос, чтобы назначить пользователю пароль в виде строки:

```
CREATE USER <user_name> IDENTIFIED WITH double_sha1_password BY '<user_password>';
```

При выполнении этого запроса ADQM вычисляет хеш для введенной строки по алгоритму double SHA-1 и сохраняет этот хеш.

Пример

Запрос:

```
CREATE USER john IDENTIFIED WITH double_sha1_password BY 'qwerty';
```

Вывод на экран:

```
CREATE USER john IDENTIFIED WITH double_sha1_hash BY 'AA1420F182E88B9E5F874F6FBE7459291E8F4601'

Query id: 7441ff9b-d349-4c29-a687-61612da40323

Ok.

0 rows in set. Elapsed: 0.001 sec.
```

Теперь пользователь `john` может подключаться к ADQM, используя заданный пароль:

```
$ clickhouse-client --user john --password qwerty
```

Указать хеш строки пароля

Чтобы назначить пароль в виде хеша, используйте следующий запрос:

```
CREATE USER <user_name> IDENTIFIED WITH double_sha1_hash BY '<hash>';
```

Пример

Сгенерируйте хеш для строки `qwerty`:

```
$ PASSWORD="qwerty"; echo "$PASSWORD"; echo -n "$PASSWORD" | sha1sum | tr -d '-' | xxd -r -p | sha1sum | tr -d '-'
```

Вывод на экран (первая строка — пароль, вторая — хеш):

```
qwerty
AA1420F182E88B9E5F874F6FBE7459291E8F4601
```

При создании учетной записи пользователя передайте сгенерированный хеш следующим образом:

```
CREATE USER mary IDENTIFIED WITH double_sha1_hash BY 'AA1420F182E88B9E5F874F6FBE7459291E8F4601';
```

Убедитесь, что подключиться к ADQM от имени пользователя `mary` можно, используя пароль `qwerty`:

```
$ clickhouse-client --user mary --password qwerty
```

Двойной хеш SHA-1 можно также определить для пользователя в конфигурационном файле *users.xml*. Для этого нужно поместить хеш в элемент `password_double_sha1_hex`:

```
<users>
    <user_name>
        <password_double_sha1_hex>AA1420F182E88B9E5F874F6FBE7459291E8F4601</password_double_sha1_hex>
        <!-- Other settings -->
    </user_name>
</users>
```

plaintext_password

Пароль пользователя сохранится в ADQM в текстовом виде, если назначить его одним из следующих способов.

- Использовать выражение `IDENTIFIED WITH plaintext_password` в запросе `CREATE USER`:

```
CREATE USER <user_name> IDENTIFIED WITH plaintext_password BY '<user_password>';
```

- Указать пароль с помощью элемента `password` в конфигурационном файле *users.xml*:

```
<users>
    <user_name>
        <password>user_password</password>
        <!-- Other settings -->
    </user_name>
</users>
```

Если оставить элемент `password` пустым, пользователь сможет подключаться в ADQM без пароля:

```
<users>
    <user_name>
        <password></password>
        <!-- Other settings -->
    </user_name>
</users>
```

ВНИМАНИЕ

Хранить пароли пользователей в незашифрованном виде не рекомендуется.

no_password

Пользователя ADQM без пароля также можно создать одним из следующих запросов:

```
CREATE USER <user_name>;
```

```
CREATE USER <user_name> IDENTIFIED WITH no_password;
```

Поле `auth_type` таблицы system.users для пользователя будет содержать значение `no_password` (в отличие от пользователя, для которого задан пустой элемент `password` в файле *users.xml* — в этом случае `auth_type` возвращает `plaintext_password`).

Изменение паролей пользователей

Чтобы изменить пароль для существующего пользователя, выполните запрос `ALTER USER`:

```
ALTER USER <user_name>
    [NOT IDENTIFIED | IDENTIFIED {[WITH {no_password | plaintext_password | sha256_password | sha256_hash | double_sha1_password |
double_sha1_hash}]
    BY {'<password>' | '<hash>'}}];
```

Пример

Измените пароль пользователя `mary`, выполнив следующий запрос:

```
ALTER USER mary IDENTIFIED WITH sha256_password BY 'test';
```

ADQM сгенерирует хеш для нового пароля, добавив к нему произвольную строку SALT:

```
ALTER USER mary IDENTIFIED WITH sha256_hash BY 'FDBC16CDEBD27426D53E8EF5AC17A02894CB195C3F213009913E5827228A7049' SALT
'48F8DEDC2E1818BE4888553B8A55B2705F1E98494CB0387A36D6F89F06DF745B'

Query id: 6adb558b-3370-4829-9a60-f8264d366a00

Ok.

0 rows in set. Elapsed: 0.001 sec.
```

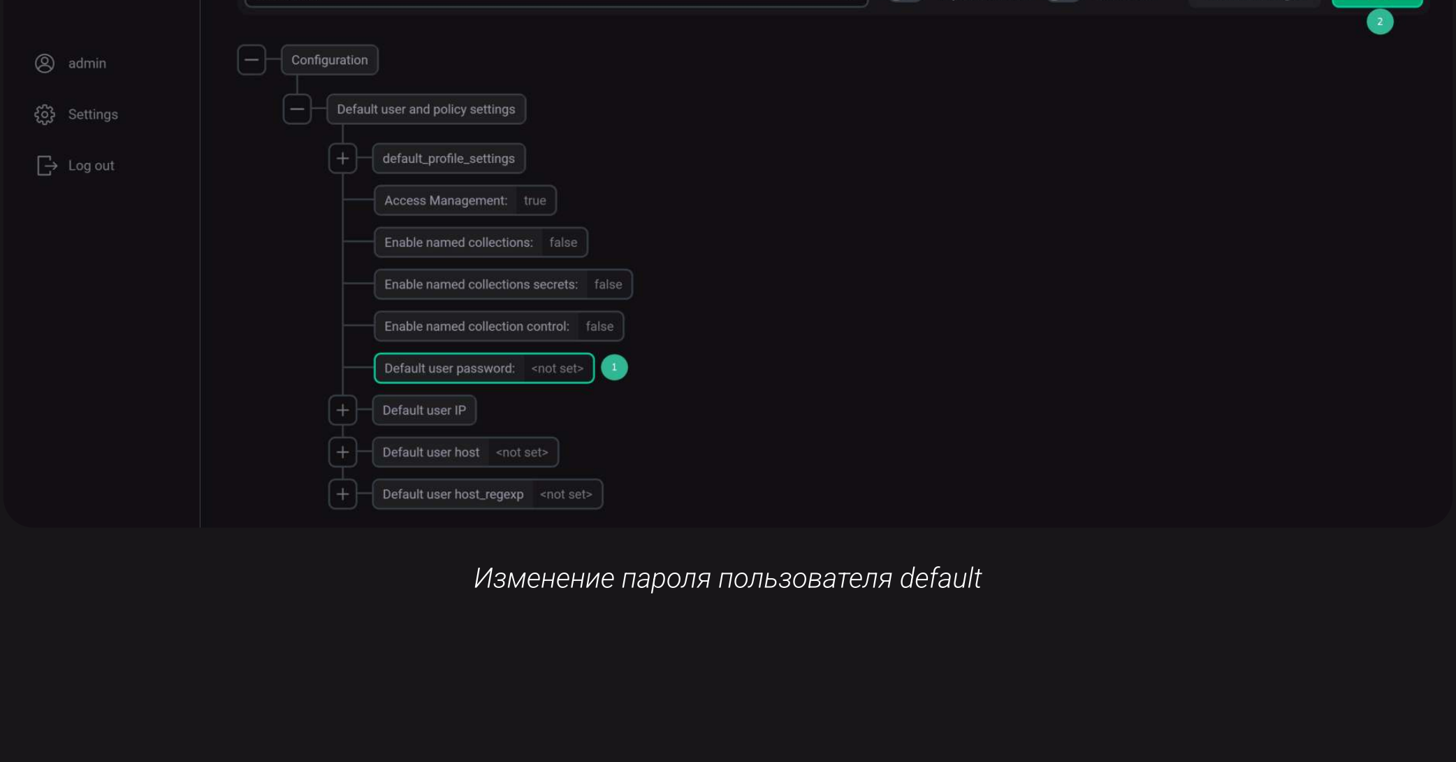
Теперь, чтобы подключаться к ADQM от имени пользователя `mary`, нужно использовать новый пароль `test`:

```
$ clickhouse-client --user mary --password test
```

Изменить пароль пользователя можно также поменяв значение соответствующего элемента (`password_sha256_hex`, `password_double_sha1_hex` или `password`) для пользователя в файле *users.xml*.

Изменение пароля пользователя default

По умолчанию ADQM создает аккаунт пользователя `default` без пароля. Назначить или изменить для него пароль можно через интерфейс ADCM. Для этого зайдите в конфигурационные настройки сервиса ADQMDB (***ADQM cluster*** → ***Services*** → ***ADQMDB*** → ***Primary Configuration***), введите и подтвердите пароль в секции ***Configuration*** → ***Default user and policy settings***, нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
17 Nov 2025 12:45:58 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
25.8.7.3_arenadata1
2 / 2 successful components
Actions
Delete
Add/Remove components
Check
Manage auto core dump
Reconfig
3
Restart
Start
Stop
Primary configuration
Configuration groups
Action host groups
Components
Info
1
2
3
4
5
6
7
...
17
current
17/11/2025 12:31:44
17/11/2025 12:28:00
17/11/2025 09:13:16
17/11/2025 08:50:19
Default user
Expand content
Advanced
Discard changes
Save
2
Configuration
Default user and policy settings
default_profile_settings
Access Management: true
Enable named collections: false
Enable named collections secrets: false
Enable named collection control: false
Default user password: <not set>
1
Default user IP
Default user host <not set>
Default user host_regexp <not set>

Изменение пароля пользователя default

Аутентификация с использованием LDAP

Елена Дворядкина

Содержание

Назад к содержанию

Настройка параметров подключения к серверу LDAP

Использование LDAP для аутентификации пользователей ADQM

Использование LDAP для хранения имен и прав пользователей

Для аутентификации пользователей ADQM можно использовать удаленный сервер LDAP как:

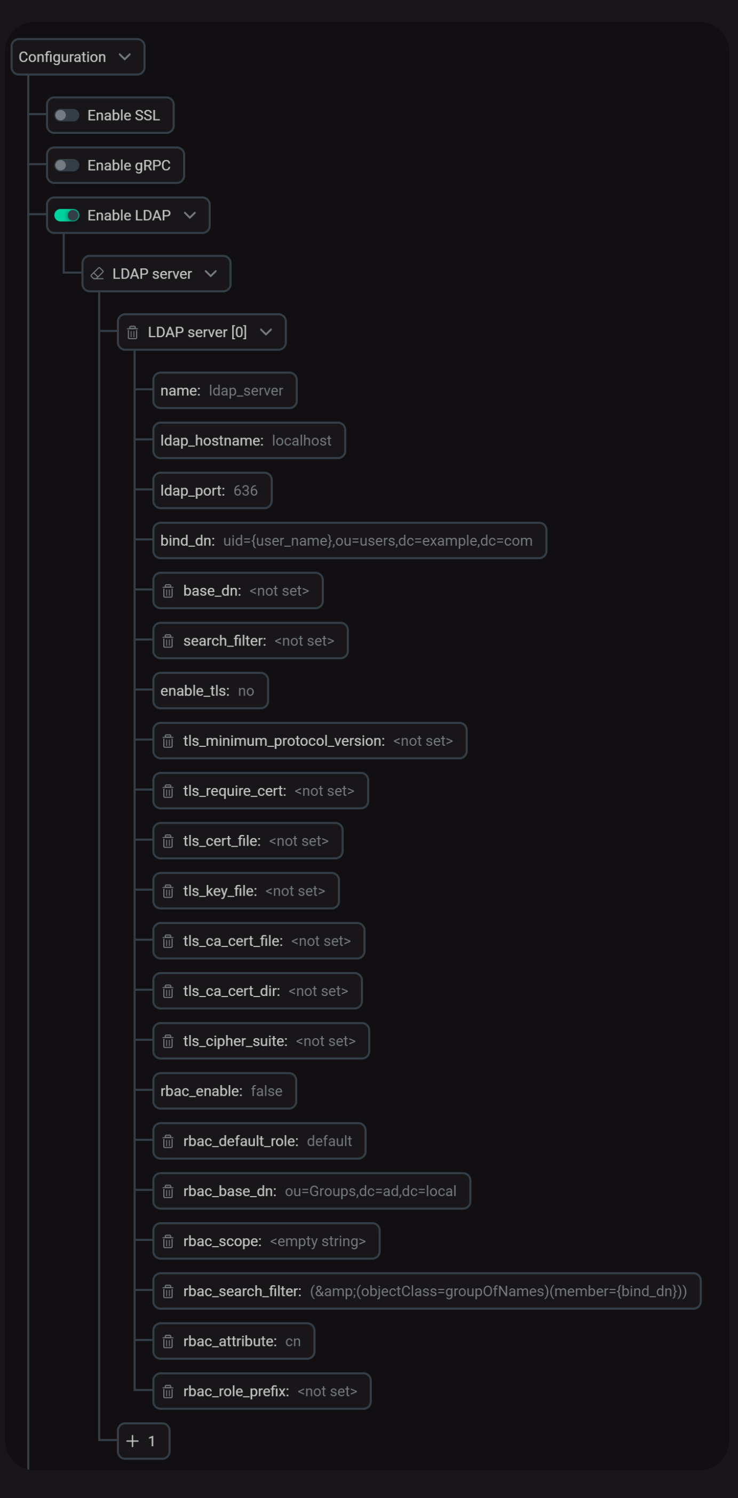
- внешний аутентификатор для существующих пользователей ADQM;
- внешний каталог, в котором хранится информация об учетных записях пользователей и их правах доступа. Этот подход позволяет аутентифицировать пользователей, не созданных локально в ADQM, но заданных на сервере LDAP.

Для обоих подходов необходимо определить параметры подключения ADQM к серверу LDAP.

Настройка параметров подключения к серверу LDAP

В интерфейсе ADCM перейдите на страницу конфигурирования сервиса ADQMDB (***ADQM cluster*** → ***Services*** → ***ADQMDB*** → ***Primary Configuration***) и активируйте опцию ***Configuration*** → ***Enable LDAP***. В развернувшейся секции настройте следующие параметры доступа к серверу LDAP:

- ***name*** — имя конфигурации подключения к серверу LDAP.
- ***ldap_hostname*** — имя хоста или IP сервера LDAP.
- ***ldap_port*** — порт сервера LDAP (если настройка ***enable_tls*** установлена в `yes`, по умолчанию используется порт `636`, иначе — `389`).
- ***bind_dn*** — шаблон для построения DN подключения, в котором подстрока `{user_name}` будет заменяться на фактическое имя пользователя при каждой попытке аутентификации.
- ***base_dn*** и ***search_filter*** — параметры поиска LDAP для определения фактического значения DN подключенного пользователя (используется в основном в фильтрах поиска для дальнейшего сопоставления ролей, когда сервер является Active Directory — полученный DN пользователя будет использоваться при замене подстрок `{user_dn}`).
- При необходимости использования безопасного соединения с сервером LDAP включите ***enable_tls*** и укажите пути к ключу и сертификатам SSL/TLS в соответствующих параметрах (подробнее в статье Конфигурационные параметры).



Configuration
Enable SSL
Enable gRPC
Enable LDAP
LDAP server
LDAP server [0]
name: ldap_server
ldap_hostname: localhost
ldap_port: 636
bind_dn: uid={user_name},ou=users,dc=example,dc=com
base_dn: <not set>
search_filter: <not set>
enable_tls: no
tls_minimum_protocol_version: <not set>
tls_require_cert: <not set>
tls_cert_file: <not set>
tls_key_file: <not set>
tls_ca_cert_file: <not set>
tls_ca_cert_dir: <not set>
tls_cipher_suite: <not set>
rbac_enable: false
rbac_default_role: default
rbac_base_dn: ou=Groups,dc=ad,dc=local
rbac_scope: <empty string>
rbac_search_filter: (&amp;(objectClass=groupOfNames)(member={bind_dn}))
rbac_attribute: cn
rbac_role_prefix: <not set>
+ 1

Параметры подключения к серверу LDAP

После определения всех параметров подключения к серверу LDAP, нажмите ***Save*** и выполните действие ***Reconfig*** с опцией ***Restart*** для сервиса ADQMDB, чтобы применить изменения. В секцию `<ldap_servers>` файла *config.xml* добавится настроенная конфигурация подключения к серверу LDAP с указанным именем (например, `ldap_server` для конфигурации на рисунке выше) — это имя нужно будет использовать при создании пользователей ADQM, аутентифицируемых через LDAP.

Использование LDAP для аутентификации пользователей ADQM

Удаленный сервер LDAP можно использовать для хранения и управления паролями пользователей, локально определенных в ADQM (созданных с помощью запроса `CREATE USER` или заданных в файле *users.xml*). То есть можно не указывать пароль при создании пользователя в ADQM, а использовать сервер LDAP для проверки пароля, введенного при аутентификации. Для этого при определении пользователя ADQM укажите конфигурацию подключения к серверу LDAP одним из следующих способов (в примерах ниже указывается конфигурация с именем `ldap_server`).

- Используйте выражение `IDENTIFIED WITH ldap SERVER` в запросе `CREATE USER`:

```
CREATE USER <ldap_user> IDENTIFIED WITH ldap SERVER 'ldap_server';
```

- Заполните элемент `ldap` для пользователя в конфигурационном файле *users.xml*:

```
<users>
    <ldap_user>
        <ldap>
            <server>ldap_server</server>
        </ldap>
        <!- ... -->
    </ldap_user>
</users>
```

где `<ldap_user>` — имя пользователя ADQM (должно совпадать с именем пользователя на сервере LDAP — значение между `auth_dn_prefix` и `auth_dn_suffix` в записи вида `dn: auth_dn_prefix{user_name}auth_dn_suffix`).

После этого пользователь `ldap_user` может подключаться к ADQM со своим паролем, заданным на сервере LDAP.

Использование LDAP для хранения имен и прав пользователей

Сервер LDAP может использоваться в качестве внешнего пользовательского каталога для ADQM. Иными словами, можно не создавать пользователей в ADQM локально, а разрешить аутентификацию пользователей только по данным из каталога LDAP.

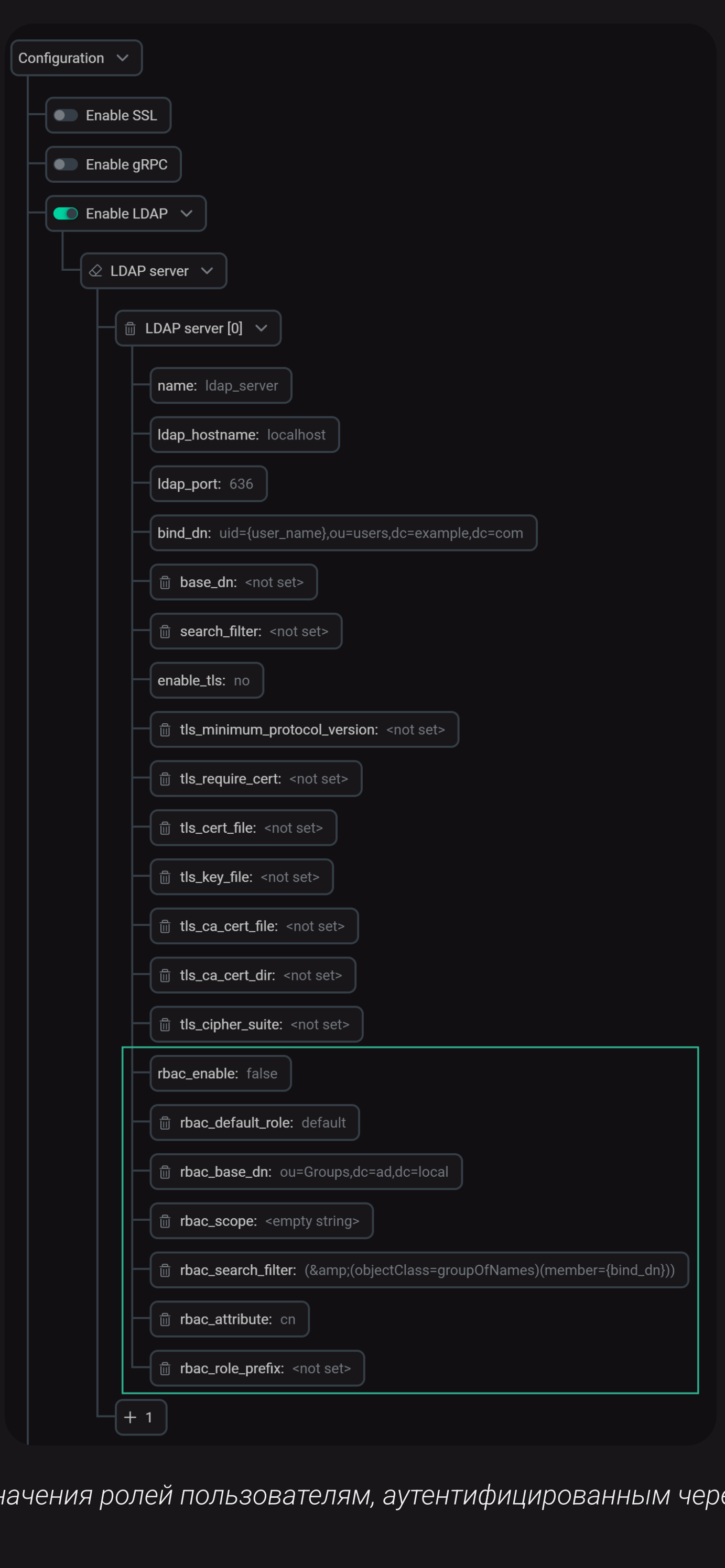
При каждой попытке аутентификации ADQM пытается найти локальное определение пользователя и аутентифицировать его заданным способом. Если аутентифицируемый пользователь в ADQM не существует, предполагается, что он определен во внешнем LDAP-каталоге. В этом случае, используя предоставленные реквизиты для входа, ADQM пытается "привязаться" (bind) к DN, указанному в параметрах подключения к серверу LDAP. Если попытка оказалась успешной, пользователь считается существующим и аутентифицированным.

Пользователям, аутентифицированным по данным из каталога LDAP, необходимо присвоить роли. Для этого выполните следующие действия.

1. Создайте в ADQM одну или несколько ролей, которые будут назначаться аутентифицированным пользователям, с помощью запроса:

```
CREATE ROLE <ldap_user_role>;
```

2. На странице конфигурации сервиса ADQMDB (секция ***Configuration*** → ***Enable LDAP***), в настройках ***LDAP server*** включите опцию ***rbac_enable*** (управление доступом на основе ролей) и установите параметры ниже в зависимости от того, как должны назначаться роли аутентифицированным через LDAP-каталог пользователям. Есть два способа назначения ролей.
    - **Задать роль по умолчанию для всех аутентифицированных пользователей.** Используя параметр ***rbac_default_role***, укажите роль, созданную ранее с помощью запроса `CREATE ROLE`.
    - **Сопоставить роли из каталога LDAP с ролями ADQM.** Задайте следующие параметры:
        - ***rbac_base_dn*** — базовый DN для поиска групп, включающих пользователя.
        - ***rbac_scope*** — область поиска LDAP (возможные значения: `base`, `one_level`, `children`, `subtree`).
        - ***rbac_search_filter*** — шаблон фильтра поиска LDAP для определения групп, которые необходимо выбрать для сопоставления пользователей. В результирующем фильтре подстроки шаблона `{user_name}`, `{bind_dn}`, `{user_dn}` и `{base_dn}` будут заменяться на фактическое имя пользователя, DN подключения, DN пользователя и базовый DN соответственно при каждом поиске LDAP.
        - ***rbac_attribute*** — имя атрибута, значение которого будет использоваться для сопоставления с ролью, заданной в ADQM.
        - ***rbac_role_prefix*** — префикс значения атрибута для поиска в дереве LDAP. Указанный префикс удаляется из найденного значения атрибута, а остальная часть значения рассматривается как имя заданной в ADQM роли — эта роль будет назначена аутентифицированному пользователю.



Configuration
Enable SSL
Enable gRPC
Enable LDAP
LDAP server
LDAP server [0]
name: ldap_server
ldap_hostname: localhost
ldap_port: 636
bind_dn: uid={user_name},ou=users,dc=example,dc=com
base_dn: <not set>
search_filter: <not set>
enable_tls: no
tls_minimum_protocol_version: <not set>
tls_require_cert: <not set>
tls_cert_file: <not set>
tls_key_file: <not set>
tls_ca_cert_file: <not set>
tls_ca_cert_dir: <not set>
tls_cipher_suite: <not set>
rbac_enable: false
rbac_default_role: default
rbac_base_dn: ou=Groups,dc=ad,dc=local
rbac_scope: <empty string>
rbac_search_filter: (&amp;(objectClass=groupOfNames)(member={bind_dn}))
rbac_attribute: cn
rbac_role_prefix: <not set>
+ 1

Параметры назначения ролей пользователям, аутентифицированным через LDAP-каталог

Чтобы завершить конфигурирование параметров и применить все изменения, нажмите ***Save*** и выполните ***Reconfig*** для сервиса ADQMDB. Указанные значения будут записаны в соответствующие секции внутри секции `user_directories/ldap` конфигурационного файла *config.xml*: `roles` (роль по умолчанию) и `role_mapping` (параметры поиска ролей в каталоге LDAP для сопоставления с ролями ADQM).

Описанные способы назначения ролей можно использовать по одному или оба одновременно. Аутентифицированному пользователю присваивается роль по умолчанию (`Default role`). Если дополнительно настроены параметры сопоставления ролей, результаты LDAP-поиска преобразуются в имена ролей и соответствующие роли ADQM тоже будут назначены пользователю. Если ни один из способов назначения ролей не сконфигурирован, пользователи не смогут выполнять никаких действий после аутентификации.

Аутентификация по SSL-сертификату

Елена Дворядкина

Содержание

Назад к содержанию

Создание SSL-сертификата пользователя

Настройка ADQM для аутентификации по SSL-сертификату

Создание пользователя

Пример

При подключении к ADQM по протоколу SSL выполняется обязательная проверка SSL-сертификатов входящих соединений — установлены могут быть только соединения, предоставившие действительный сертификат. Таким образом ADQM однозначно аутентифицирует входящее соединение и может идентифицировать пользователя по соответствующему пользовательскому SSL-сертификату. Для одного пользователя можно задать несколько сертификатов.

Ниже описывается последовательность действий, необходимых для включения аутентификации пользователей по SSL-сертификату. Для выполнения приведенных команд нужно установить `openssl` — инструмент командной строки для использования функций криптографической библиотеки OpenSSL.

ПРИМЕЧАНИЕ

- Возможность аутентифицировать пользователя по SSL-сертификату доступна только при подключении к ADQM через клиента, который поддерживает интерфейс HTTPS.
- Для демонстрационных целей в этой статье пользовательский SSL-сертификат подписывается самоподписанным (self-signed) корневым сертификатом центра сертификации (Certification Authority, CA). В реальных системах используйте доверенный центр сертификации для подписи сертификатов.

Создание SSL-сертификата пользователя

1. Сгенерируйте ключ и создайте файл с запросом на подпись сертификата (Certificate Signing Request, CSR):

```
$ openssl req \
  -newkey rsa:2048 \
  -nodes \
  -subj "/CN=<user_cert_identifier>" \
  -keyout <user_cert_name>.key \
  -out <user_cert_name>.csr
```

   Идентификатором сертификата CN (Common Name) может быть любое уникальное значение. Это значение будет использоваться при создании пользователя ADQM.

2. Создайте и подпишите сертификат пользователя, который будет использоваться для аутентификации:

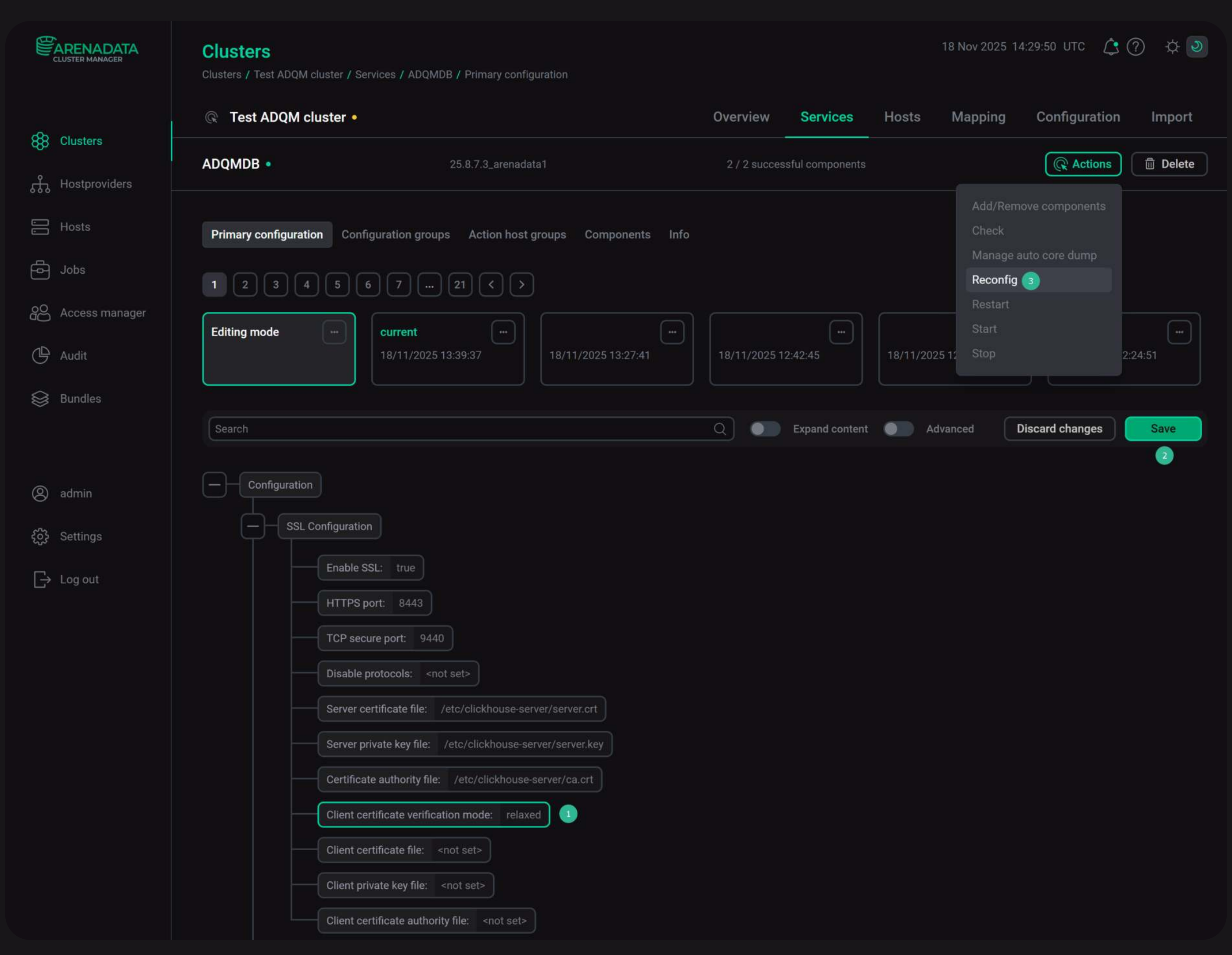
```
$ openssl x509 \
  -req \
  -in <user_cert_name>.csr \
  -out <user_cert_name>.crt \
  -CAcreateserial \
  -CA <ca>.crt \
  -CAkey <ca>.key \
  -days 365
```

   где `<ca>.crt` и `<ca>.key` — файлы, используемые при конфигурировании ADQM для защищенных соединений по протоколу SSL (см. Шифрование соединений с базой данных):

   - `<ca>.crt` — публичный корневой сертификат центра сертификации. Должен быть установлен на каждом сервере кластера ADQM и указан в настройках конфигурации сервиса ADQMDB (в интерфейсе ADCM: ***ADQM cluster → Services → ADQMDB → Primary configuration → SSL Configuration → Certificate authority file***).
   - `<ca>.key` — закрытый ключ центра сертификации. Должен храниться в секрете.

Настройка ADQM для аутентификации по SSL-сертификату

В интерфейсе ADCM перейдите к окну конфигурирования сервиса ADQMDB и в секции ***SSL Configuration*** с помощью опции ***Client certificate verification mode*** установите режим проверки пользовательских SSL-сертификатов (`relaxed`, `strict` или `once`). Нажмите ***Save*** и выполните действие ***Reconfig*** с опцией ***Restart*** для сервиса ADQMDB, чтобы сохранить и применить изменение значения параметра.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
18 Nov 2025 14:29:50 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
ADQMDB
25.8.7.3_arenadata1
2 / 2 successful components
Actions
Delete
Add/Remove components
Check
Manage auto core dump
Reconfig
3
Restart
Start
Stop
Primary configuration
Configuration groups
Action host groups
Components
Info
1 2 3 4 5 6 7 ... 21
Editing mode
current
18/11/2025 13:39:37
18/11/2025 13:27:41
18/11/2025 12:42:45
18/11/2025 1
2:24:51
Search
Expand content
Advanced
Discard changes
Save
2
Configuration
SSL Configuration
1

Enable SSL: true
HTTPS port: 8443
TCP secure port: 9440
Disable protocols: <not set>
Server certificate file: /etc/clickhouse-server/server.crt
Server private key file: /etc/clickhouse-server/server.key
Certificate authority file: /etc/clickhouse-server/ca.crt
Client certificate verification mode: relaxed
Client certificate file: <not set>
Client private key file: <not set>
Client certificate authority file: <not set>

Включение проверки пользовательских SSL-сертификатов

Создание пользователя

Чтобы создать пользователя ADQM, который будет аутентифицироваться по сертификату SSL, в запросе `CREATE USER` используйте выражение `IDENTIFIED WITH ssl_certificate` и укажите идентификатор сертификата (CN):

```
CREATE USER <user_name> IDENTIFIED WITH ssl_certificate CN '<user_cert_identifier>';
```

Если вы задаете пользователя через конфигурационный файл *users.xml*, в секции `certificates` для пользователя укажите элемент `common_name` с идентификатором пользовательского сертификата (в списке сертификатов может быть более одного элемента):

```
<users>
    <user_name>
        <certificates>
            <common_name>user_cert_identifier</common_name>
            <!-- More names -->
        </certificates>
        <!-- Other settings -->
    </user_name>
</users>
```

Пример

В этом примере показано, как создать и использовать SSL-сертификат для аутентификации пользователя при подключении к кластеру ADQM, для которого настроено шифрование соединений по протоколу SSL согласно инструкции в статье Шифрование соединений с базой данных.

1. Сгенерируйте ключ и создайте файл с запросом на подпись сертификата:

```
$ openssl req \
  -newkey rsa:2048 \
  -nodes -subj "/CN=arenadata:john" \
  -keyout arenadata_john.key \
  -out arenadata_john.csr
```

2. Создайте и подпишите пользовательский сертификат для аутентификации:

```
$ openssl x509 \
  -req \
  -in arenadata_john.csr \
  -out arenadata_john.crt \
  -CAcreateserial \
  -CA arenadata_ca.crt \
  -CAkey arenadata_ca.key \
  -days 365
```

   В этом запросе *arenadata_ca.crt* — корневой сертификат центра сертификации, *arenadata_ca.key* — закрытый ключ центра сертификации.

3. В настройках сервиса ADQMDB через параметр ***Client certificate verification mode*** включите режим проверки пользовательских сертификатов.

4. Создайте пользователя (например, на хосте *adqm-host-1.arenadata.local*), которого ADQM будет аутентифицировать по созданному сертификату:

```
CREATE USER john IDENTIFIED WITH ssl_certificate CN 'arenadata:john';
```

5. Назначьте пользователю привилегии (например, привилегии на чтение всех таблиц):

```
GRANT SELECT ON *.* TO john WITH GRANT OPTION;
```

6. Проверьте подключение к ADQM по HTTPS от имени этого пользователя. Например, можно выполнить следующий запрос из директории, где хранятся сертификаты:

```
$ echo 'SHOW TABLES' | \
  curl 'https://adqm-host-1.arenadata.local:8443' \
  --cert ./arenadata_john.crt \
  --key arenadata_john.key \
  --cacert arenadata_ca.crt \
  -H "X-ClickHouse-SSL-Certificate-Auth: on" \
  -H "X-ClickHouse-User: john" \
  --data-binary @-
```

   Если аутентификация прошла успешно, в результате выполнения запроса вернется список таблиц базы данных.

Аутентификация через Kerberos

Елена Дворядкина

Содержание

Назад к содержанию

Типы центров распределения ключей (KDC)

Общие требования для керберизации кластера ADQM

ADQM предоставляет возможность аутентифицировать существующих пользователей с использованием протокола Kerberos, который обеспечивает механизм взаимной аутентификации клиента и сервера перед установлением соединения между ними.

Основные понятия, используемые в Kerberos:

- **Принципал** (**principal**) — уникальное имя клиента (пользователя), который проходит аутентификацию через Kerberos.
- **Тикет** (**ticket**) — временные данные, выдаваемые клиенту для аутентификации на сервере, на котором располагается необходимый сервис.
- **Центр распределения ключей** (**Key Distribution Center**, **KDC**) — сервис, выдающий тикеты Kerberos и временные сеансовые ключи.
- **Область** (**realm**) — сеть Kerberos, которая включает KDC и одного или нескольких клиентов. Имя realm регистрозависимо, обычно пишется в верхнем регистре и совпадает с именем домена.

Типы центров распределения ключей (KDC)

Для керберизации кластера ADQM доступны следующие типы центров распределения ключей (KDC):

- MIT Kerberos — состоит из базы данных принципалов и хранилища ключей Kerberos.
- MS Active Directory — содержит базу данных принципалов и хранилище ключей Windows Server.
- FreeIPA — бесплатная система управления идентификацией пользователей для Linux/UNIX с открытым исходным кодом.

При использовании любого из этих типов KDC процесс идентификации и аутентификации пользователей практически одинаков. Единственное различие заключается в типе используемого KDC.

Общие требования для керберизации кластера ADQM

Чтобы внедрить Kerberos в существующий кластер ADQM, необходимо выполнение следующих условий:

- Кластер ADQM установлен, настроен и готов к работе.
- Хранилище пользователей и ключей MIT Kerberos, Active Directory или FreeIPA настроено и готово к работе.
- Существует учетная запись администратора MIT Kerberos, Active Directory или FreeIPA с правами на создание, удаление и управление учетными записями пользователей.

Пользователи, настроенные для аутентификации через Kerberos, могут работать с ADQM только через интерфейс HTTP. При этом используемый клиент должен поддерживать механизм SPNEGO. Например, можно использовать утилиту `curl`. Убедиться, что Kerberos и SPNEGO поддерживаются, можно с помощью следующей команды:

```
$ curl --version
```

Пример вывода на экран:

```
curl 7.81.0 (x86_64-pc-linux-gnu) libcurl/7.81.0 OpenSSL/1.0.2k-fips zlib/1.2.7
Release-Date: 2022-01-05
Protocols: dict file ftp ftps gopher gophers http https imap imaps mqtt pop3 pop3s rtsp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS GSS-API HSTS HTTPS-proxy IPv6 Kerberos Largefile libz NTLM NTLM_WB SPNEGO SSL UnixSockets
```

Использование сервера MIT Kerberos

Елена Дворядкина

Содержание

Назад к содержанию

Настройка сервера Kerberos

Включение MIT Kerberos KDC на кластере ADQM

Проверка аутентификации пользователя через Kerberos

Настройка сервера Kerberos

ПРИМЕЧАНИЕ

- Kerberos KDC может быть установлен на любом хосте, который имеет доступ к хосту с установленным ADCM (возможность подключения по FQDN).
- Допускается также установить Kerberos KDC на одном хосте с ADCM.

В приведенном в данной статье примере Kerberos KDC устанавливается на хост *dev-adcm-test.ru-central1.internal* с операционной системой CentOS. На этом же хосте установлен ADCM.

1. Установите Kerberos KDC на хост:

```
$ sudo -i
```

```
yum install krb5-libs krb5-server krb5-workstation
```

2. Внесите изменения в файлы конфигурации.

   - В файле */var/kerberos/krb5kdc/kdc.conf* замените realm на `RU-CENTRAL1.INTERNAL`:

```
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
 RU-CENTRAL1.INTERNAL = {
  #master_key_type = aes256-cts
  acl_file = /var/kerberos/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
  supported_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal camellia256-cts:normal
camellia128-cts:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal
 }
```

   - В файле */etc/krb5.conf* укажите значение параметра `default_realm`, а также заполните разделы `[realms]` и `[domain_realm]`:

```
# Configuration snippets may be placed in this directory as well
includedir /etc/krb5.conf.d/

[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 dns_lookup_realm = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
 rdns = false
 pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
 default_realm = RU-CENTRAL1.INTERNAL
 default_ccache_name = KEYRING:persistent:%{uid}

[realms]
 RU-CENTRAL1.INTERNAL = {
  kdc = dev-adcm-test.ru-central1.internal
  admin_server = dev-adcm-test.ru-central1.internal
 }

[domain_realm]
 .ru-central1.internal = RU-CENTRAL1.INTERNAL
 ru-central1.internal = RU-CENTRAL1.INTERNAL
```

   - Измените содержимое файла */var/kerberos/krb5kdc/kadm5.acl* следующим образом, чтобы любой принципал `/admin` в области `RU-CENTRAL1.INTERNAL` имел все административные привилегии, кроме извлечения ключей:

```
*/admin@RU-CENTRAL1.INTERNAL    *
```

   Полное описание значений тегов для разделов данных файлов приведено в статьях kdc.conf, krb5.conf и kadm5.acl документации MIT Kerberos.

3. Создайте базу данных принципалов Kerberos с помощью утилиты `kdb5_util`:

```
kdb5_util create -s -P P@ssw0rd
```

4. Запустите сервисы Kerberos и KDC, выполнив поочередно следующие команды:

```
systemctl start kadmin
systemctl start krb5kdc
```

   Проверить статус запущенных сервисов можно с помощью команд:

```
systemctl status kadmin
systemctl status krb5kdc
```

5. Создайте принципалы пользователей, используя консоль администратора Kerberos — `kadmin.local`.
   Имя принципала пользователя в базе данных Kerberos обычно имеет вид `primary/instance@REALM`, где:

   - `primary` — базовое имя пользователя (для успешной аутентификации в ADQM необходимо, чтобы это имя совпадало с именем пользователя ADQM, настроенного для использования Kerberos);
   - `instance` — необязательный компонент, квалифицирующий `primary`. Если `primary` — имя пользователя, то `instance` обычно не указывается или может использоваться, чтобы определить дополнительный принципал для пользователя (например, принципалы `john@RU-CENTRAL1.INTERNAL` и `john/admin@RU-CENTRAL1.INTERNAL` — отдельные принципалы с разными паролями и разрешениями).

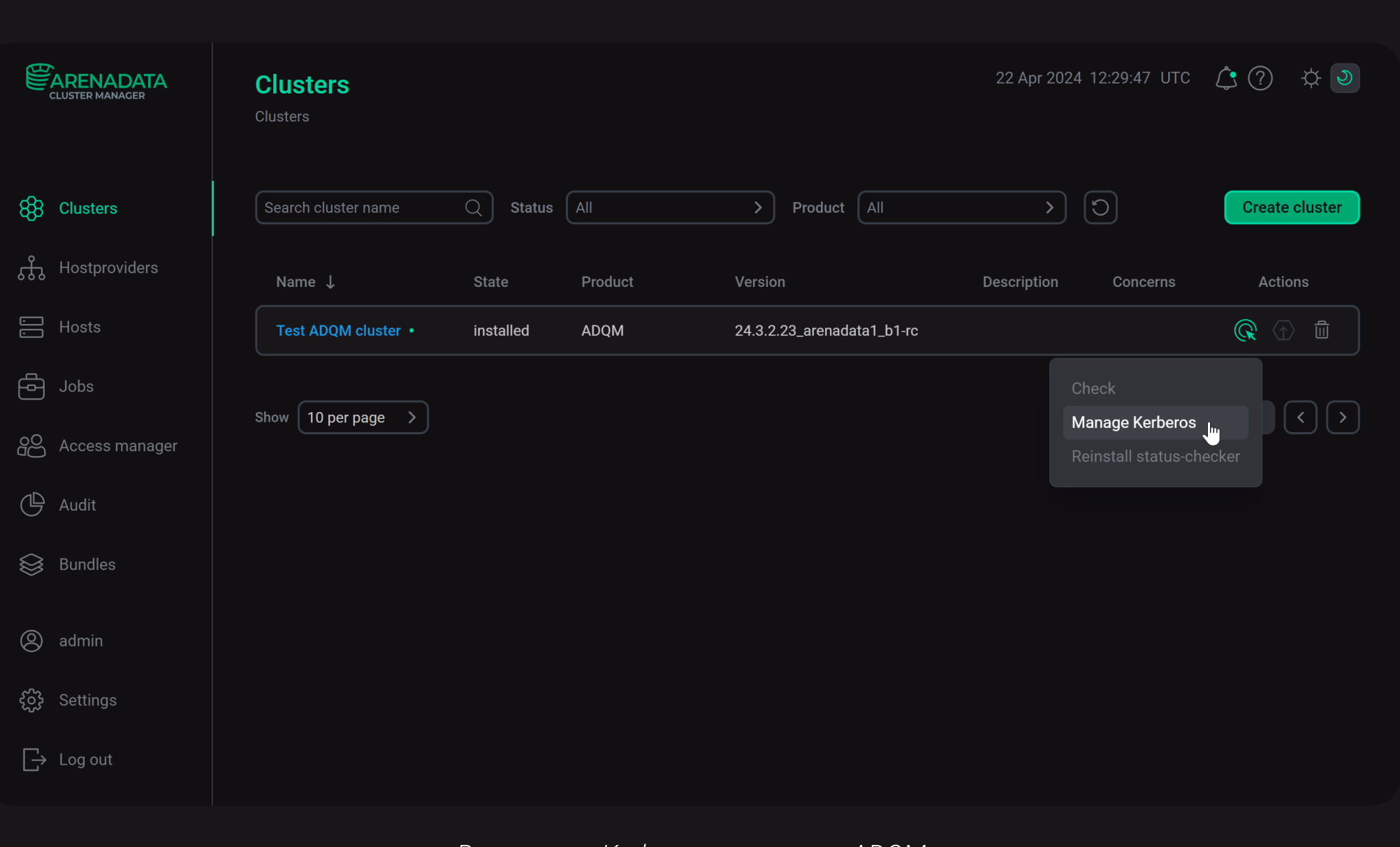
```
kadmin.local -q "add_principal -pw P@ssw0rd admin/admin"
```

```
kadmin.local -q "add_principal -pw P@ssw0rd adqm_kerb"
```

   Полное описание функций интерфейса командной строки `kadmin.local` и применяемых опций приведены в статье kadmin документации MIT Kerberos.

Включение MIT Kerberos KDC на кластере ADQM

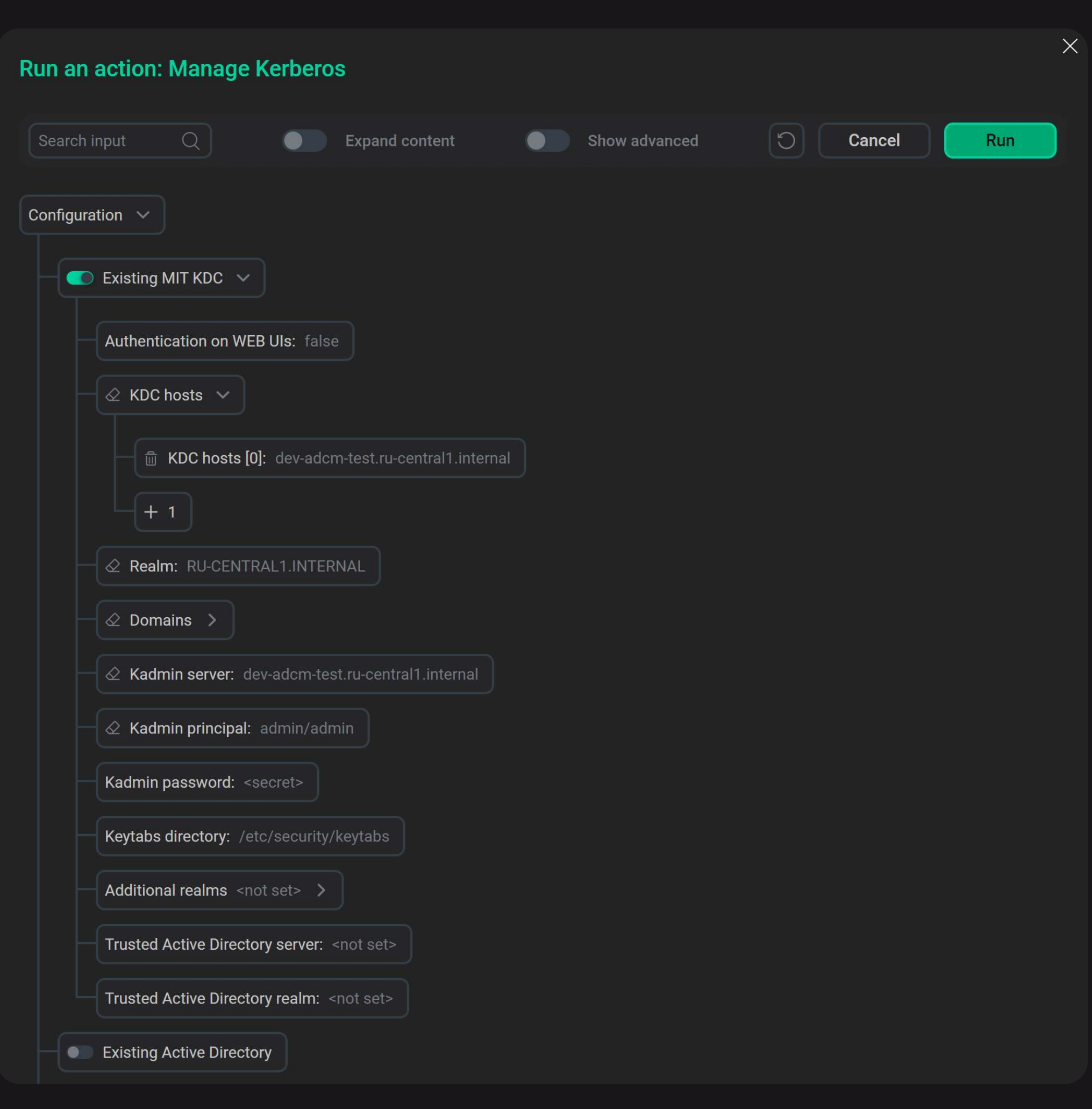
1. В интерфейсе ADCM на странице ***Clusters*** выберите установленный и подготовленный кластер ADQM.
2. Инициируйте включение Kerberos для кластера — нажмите на иконку ***Actions*** и выберите действие ***Manage Kerberos***.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
22 Apr 2024 12:29:47 UTC
Search cluster name
Status
Product
Create cluster
Check
Manage Kerberos
Reinstall status-checker
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	installed	ADQM	24.3.2.23_arenadata1_b1-rc			

Включение Kerberos в кластере ADQM

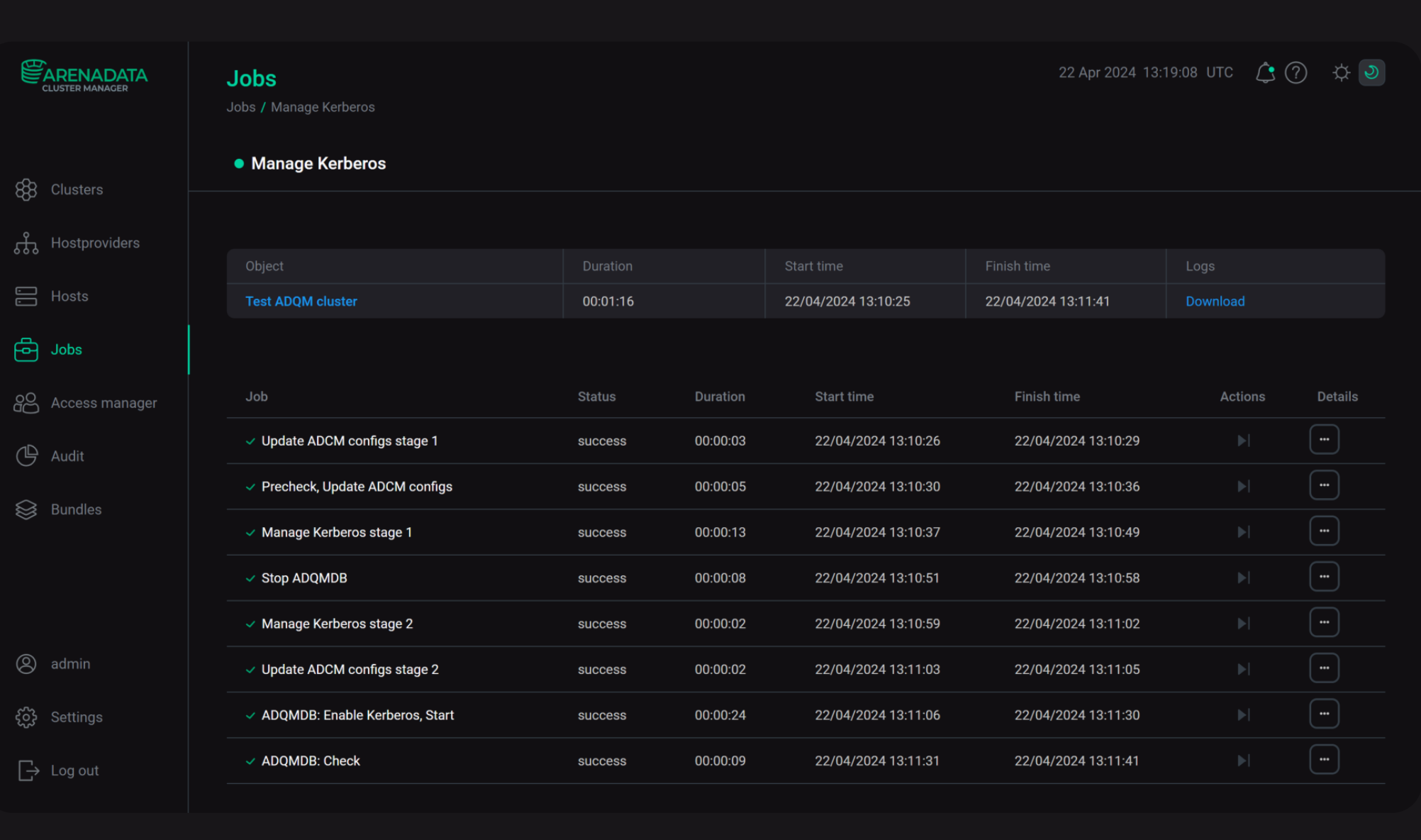
3. В открывшемся окне включите опцию ***Existing MIT KDC***, заполните конфигурационные параметры в соответствии с ранее установленными настройками сервера Kerberos и нажмите ***Run***.


Run an action: Manage Kerberos
Search input
Expand content
Show advanced
Cancel
Run
Configuration
+ 1

- Existing MIT KDC
  - Authentication on WEB UIs: false
  - KDC hosts
    - KDC hosts [0]: dev-adcm-test.ru-central1.internal
  - Realm: RU-CENTRAL1.INTERNAL
  - Domains
  - Kadmin server: dev-adcm-test.ru-central1.internal
  - Kadmin principal: admin/admin
  - Kadmin password: <secret>
  - Keytabs directory: /etc/security/keytabs
  - Additional realms <not set>
  - Trusted Active Directory server: <not set>
  - Trusted Active Directory realm: <not set>
- Existing Active Directory

Параметры MIT KDC

4. Дождитесь завершения активации Kerberos. Процесс и результат выполнения задачи ***Manage Kerberos*** отображается на странице ***Jobs***.

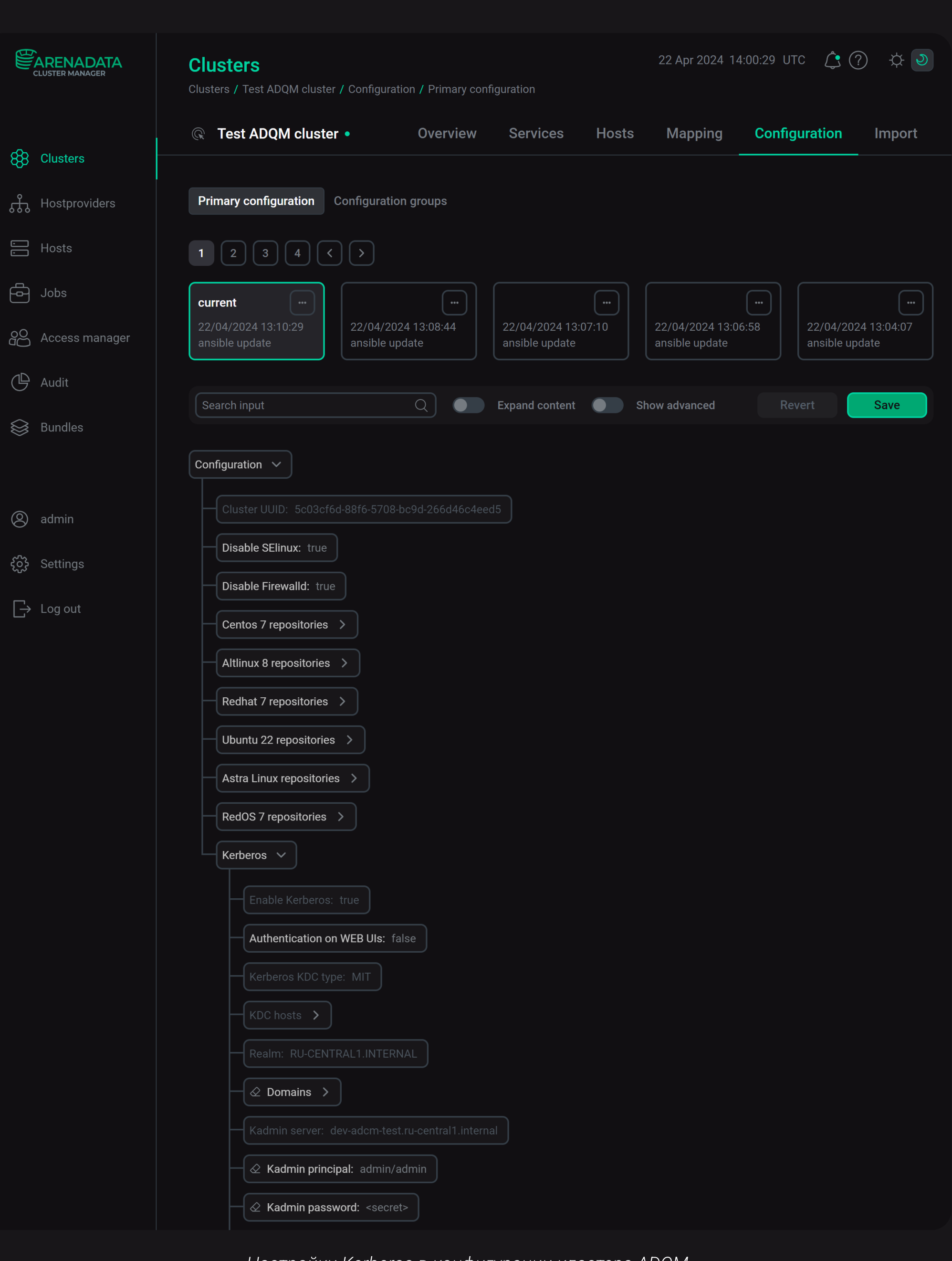

ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Jobs
Jobs / Manage Kerberos
22 Apr 2024 13:19:08 UTC
Manage Kerberos

Object	Duration	Start time	Finish time	Logs
Test ADQM cluster	00:01:16	22/04/2024 13:10:25	22/04/2024 13:11:41	Download

Job	Status	Duration	Start time	Finish time	Actions	Details
Update ADCM configs stage 1	success	00:00:03	22/04/2024 13:10:26	22/04/2024 13:10:29		
Precheck, Update ADCM configs	success	00:00:05	22/04/2024 13:10:30	22/04/2024 13:10:36		
Manage Kerberos stage 1	success	00:00:13	22/04/2024 13:10:37	22/04/2024 13:10:49		
Stop ADQMDB	success	00:00:08	22/04/2024 13:10:51	22/04/2024 13:10:58		
Manage Kerberos stage 2	success	00:00:02	22/04/2024 13:10:59	22/04/2024 13:11:02		
Update ADCM configs stage 2	success	00:00:02	22/04/2024 13:11:03	22/04/2024 13:11:05		
ADQMDB: Enable Kerberos, Start	success	00:00:24	22/04/2024 13:11:06	22/04/2024 13:11:30		
ADQMDB: Check	success	00:00:09	22/04/2024 13:11:31	22/04/2024 13:11:41		

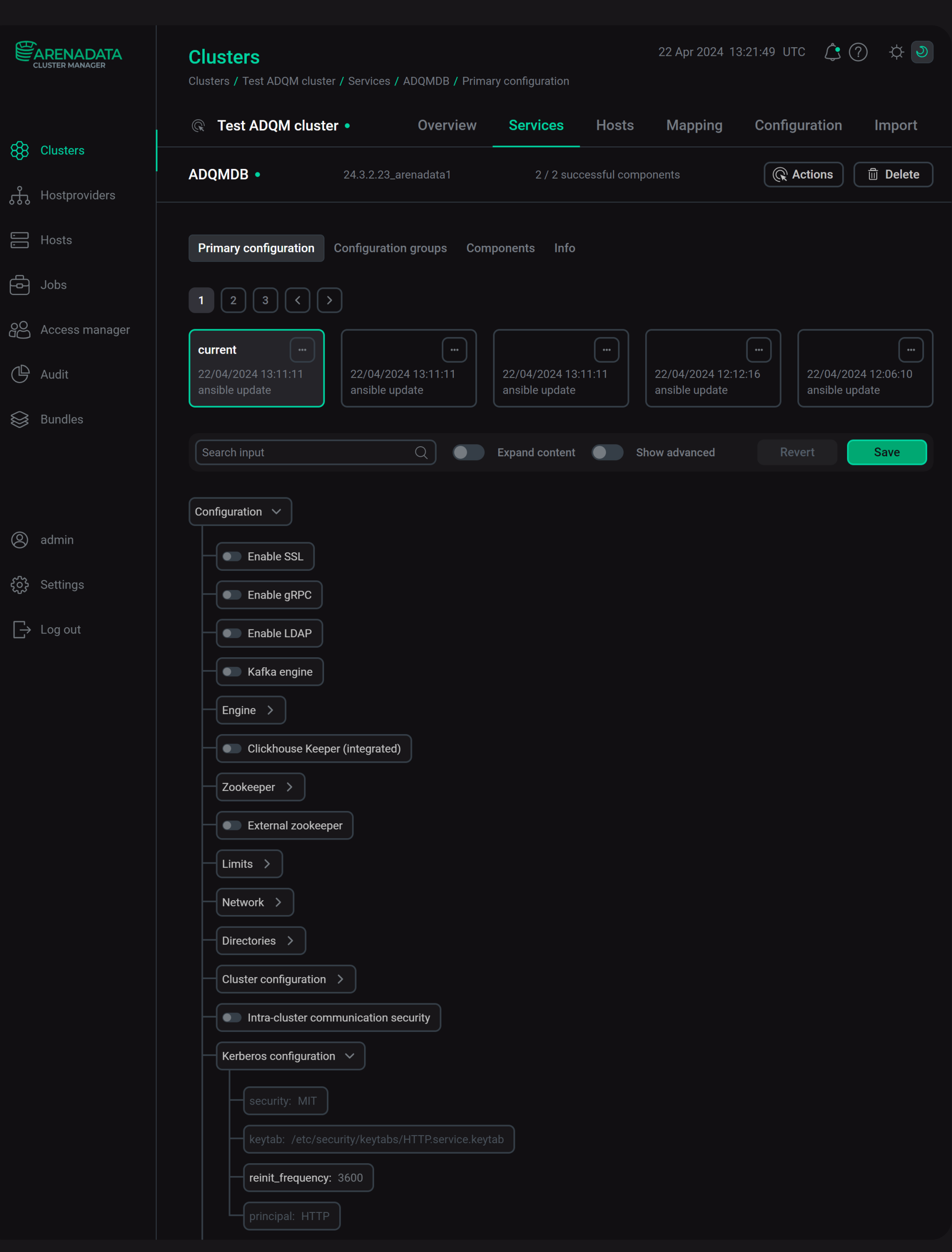
Процесс запуска Kerberos

После того, как Kerberos успешно установлен на кластере ADQM, соответствующие настройки появятся на странице конфигурации кластера (секция ***Kerberos***) и сервиса ADQMDB (секция ***Kerberos configuration***).


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Configuration / Primary configuration
22 Apr 2024 14:00:29 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Primary configuration
Configuration groups
1 2 3 4
current 22/04/2024 13:10:29 ansible update
22/04/2024 13:08:44 ansible update
22/04/2024 13:07:10 ansible update
22/04/2024 13:06:58 ansible update
22/04/2024 13:04:07 ansible update
Search input
Expand content
Show advanced
Revert
Save

- Configuration
  - Cluster UUID: 5c03cf6d-88f6-5708-bc9d-266d46c4eed5
  - Disable SElinux: true
  - Disable Firewalld: true
  - Centos 7 repositories
  - Altlinux 8 repositories
  - Redhat 7 repositories
  - Ubuntu 22 repositories
  - Astra Linux repositories
  - RedOS 7 repositories
  - Kerberos
    - Enable Kerberos: true
    - Authentication on WEB UIs: false
    - Kerberos KDC type: MIT
    - KDC hosts
    - Realm: RU-CENTRAL1.INTERNAL
    - Domains
    - Kadmin server: dev-adcm-test.ru-central1.internal
    - Kadmin principal: admin/admin
    - Kadmin password: <secret>

Настройки Kerberos в конфигурации кластера ADQM


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
22 Apr 2024 13:21:49 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
24.3.2.23_arenadata1
2 / 2 successful components
Actions
Delete
Primary configuration
Configuration groups
Components
Info
1 2 3
current 22/04/2024 13:11:11 ansible update
22/04/2024 13:11:11 ansible update
22/04/2024 13:11:11 ansible update
22/04/2024 12:12:16 ansible update
22/04/2024 12:06:10 ansible update
Search input
Expand content
Show advanced
Revert
Save

- Configuration
  - Enable SSL
  - Enable gRPC
  - Enable LDAP
  - Kafka engine
  - Engine
  - Clickhouse Keeper (integrated)
  - Zookeeper
  - External zookeeper
  - Limits
  - Network
  - Directories
  - Cluster configuration
  - Intra-cluster communication security
  - Kerberos configuration
    - security: MIT
    - keytab: /etc/security/keytabs/HTTP.service.keytab
    - reinit_frequency: 3600
    - principal: HTTP

Настройки Kerberos в конфигурации сервиса ADQMDB

Проверка аутентификации пользователя через Kerberos

После успешной керберизации выполнение ADQM-команд на узлах кластера возможно только после получения соответствующего тикета Kerberos. Ниже приведен пример доступа к керберизованному кластеру:

1. Создайте в ADQM пользователя, для которого существует принципал в базе данных Kerberos. Для этого используйте выражение `IDENTIFIED WITH kerberos` в запросе `CREATE USER`:

```
CREATE USER adqm_kerb IDENTIFIED WITH kerberos;
```

2. Создайте для принципала пользователя тикет Kerberos:

```
$ kinit -V adqm_kerb@RU-CENTRAL1.INTERNAL
```

   Введите пароль:

```
Password for adqm_kerb@RU-CENTRAL1.INTERNAL:
```

   Опции команды `kinit` описаны в статье kinit документации MIT Kerberos.

3. Проверьте тикет:

```
$ klist
```

```
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: adqm_kerb@RU-CENTRAL1.INTERNAL

Valid starting       Expires              Service principal
05/04/2023 07:13:58  05/05/2023 07:13:58  krbtgt/RU-CENTRAL1.INTERNAL@RU-CENTRAL1.INTERNAL
```

4. Для проверки аутентификации через Kerberos отправьте в ADQM тестовую команду, используя утилиту `curl` с поддержкой механизма SPNEGO:

```
$ echo "select currentUser()" | curl --negotiate -u : http://<host_name>:8123/ --data-binary @-
```

   Вывод на экран при успешной аутентификации:

```
adqm_kerb
```

Использование Active Directory для Kerberos

Елена Дворядкина

Содержание

Назад к содержанию

Подготовка к запуску Active Directory на кластере ADQM

Включение Active Directory на кластере ADQM

Kerberos-параметры Active Directory

Проверка аутентификации пользователя через Kerberos

Основные понятия, используемые в Kerberos LDAP:

- **Active Directory** — база данных и набор служб, которые соединяют пользователей с необходимыми сетевыми ресурсами.
- **LDAP-сервер** — иерархическая база данных, служба каталогов, основанная на Active Directory, применяется для централизованного хранения учетных записей. Учетные данные включают в себя принципалы, по которым происходит проверка валидности пользователя в Kerberos.
- **LDAP** — протокол прикладного уровня для доступа к службе каталогов Active Directory, используется при керберизации кластера для заведения принципалов пользователей.
- **LDAPS** — активация LDAP с применением протокола SSL/TLS для создания безопасного трафика. Создание паролей для пользователей ADQM с учетными записями в Active Directory при керберизации кластера возможно только с применением LDAPS.
- **Key Distribution Center** (**KDC**) — компонент системы управления доступом, отвечающий за обслуживание запросов пользователей на доступ к ресурсам путем предоставления тикетов доступа и сеансовых ключей. Использует Active Directory в качестве базы данных учетных записей.
- **DN (Distinguished Name)** — учетная запись в Active Directory. DN должно быть уникальным в пределах дерева. В DN описывается содержимое атрибутов в дереве (путь навигации) для доступа к конкретной записи.
  DN состоит из серии **RDN (Relative Distinguished Names)**, определяемых путем перемещения вверх по дереву в направлении его корневой записи. RDN записываются слева направо.

> Пример DN для пользователя, используемый в данной статье: *cn=ad_admin,ou=users,dc=example,dc=com*.
>
> В этой записи назначены следующие RDN:
>
> - cn — общее имя пользователя (common name).
> - ou — организационная единица (organizational unit), контейнер в домене Active Directory, который может содержать пользователей, группы и компьютеры. Организационная единица может иметь несколько ou внутри себя.
> - dc — части имени домена (domain component), представляют вершину дерева LDAP, которое использует DNS для определения своего пространства имен. Обозначение для домена Active Directory с DNS-именем `EXAMPLE.COM` имеет вид `dc=example,dc=com`. Имя `EXAMPLE.COM` также присваивается для области (realm), для которой проводится аутентификация.

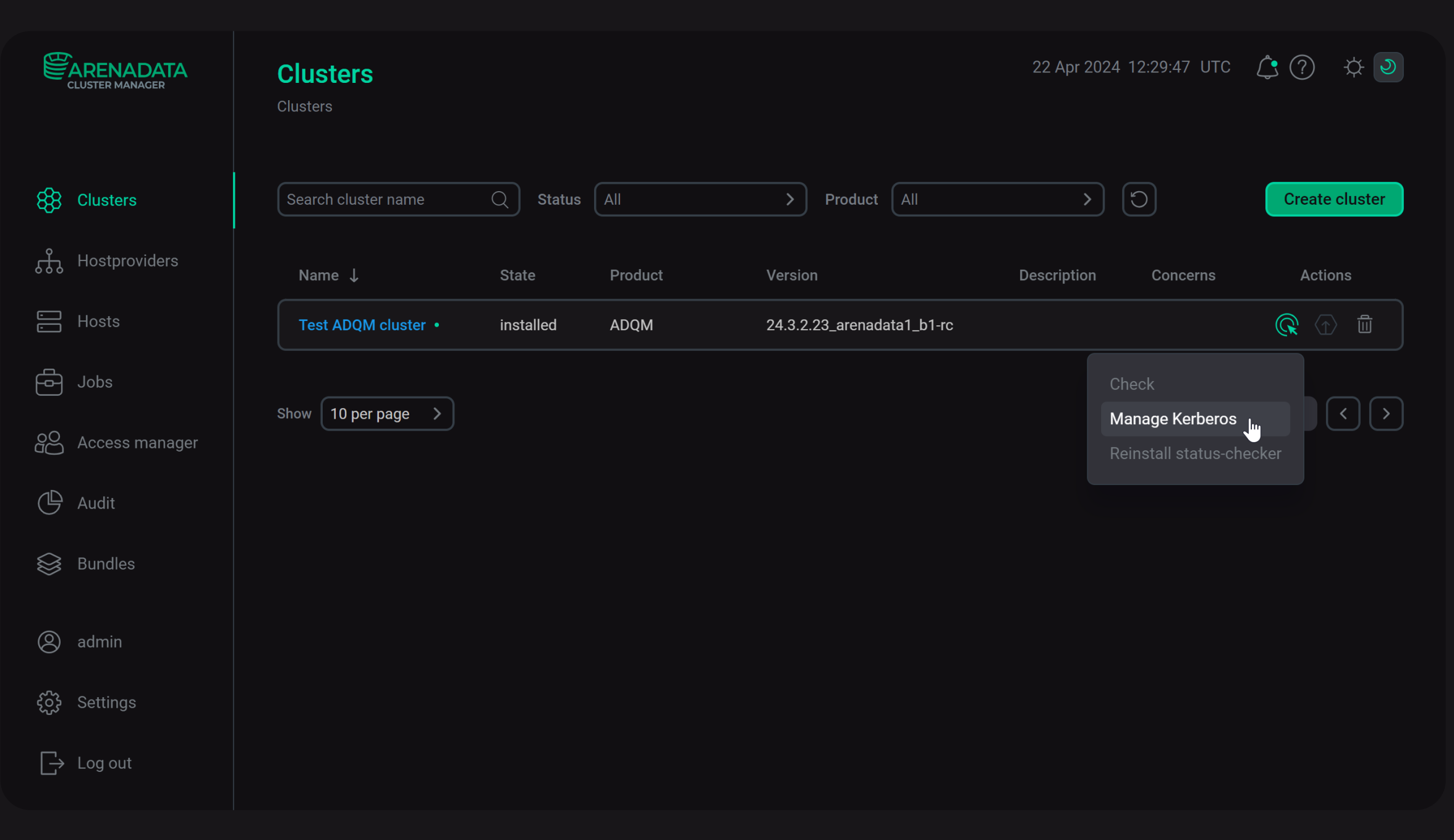
Подготовка к запуску Active Directory на кластере ADQM

1. Настройте сервер Active Directory (LDAP-сервер).
2. Создайте учетную запись пользователя для подключения кластера к LDAP-серверу по протоколу LDAP.
3. Создайте сертификат для пользователя, активировав протокол LDAP через протокол SSL с использованием стороннего центра сертификации.
4. Установите пароль для учетной записи.
5. Привяжите к учетной записи имя принципала пользователя (например, `ad_admin@EXAMPLE.COM`) и сформируйте файл *keytab* с этими принципалом и ключами.

Включение Active Directory на кластере ADQM

Для керберизации кластера с помощью Active Directory выполните описанные ниже шаги:

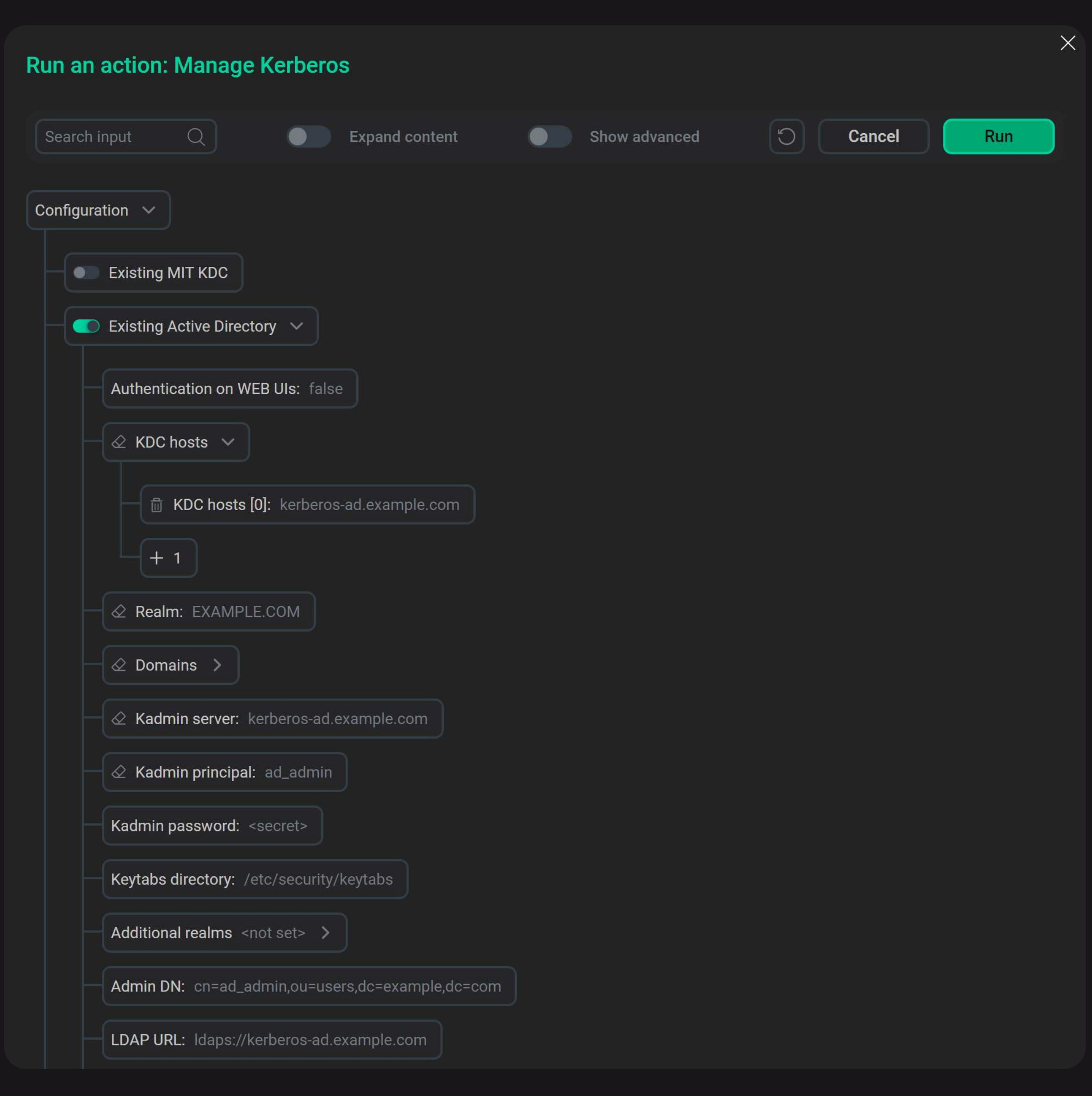
1. В интерфейсе ADCM на странице ***Clusters*** выберите установленный и подготовленный кластер ADQM.
2. Инициируйте включение Kerberos для кластера — нажмите на иконку ***Actions*** и выберите действие ***Manage Kerberos***.



ARENADATA CLUSTER MANAGER
Clusters
22 Apr 2024 12:29:47 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test ADQM cluster
installed
ADQM
24.3.2.23_arenadata1_b1-rc
Show 10 per page
Check
Manage Kerberos
Reinstall status-checker

Включение Kerberos в кластере ADQM

3. В открывшемся окне включите опцию ***Existing Active Directory*** и укажите параметры Active Directory в соответствии со значениями, установленными при настройке LDAP-сервера.



Run an action: Manage Kerberos

Existing MIT KDC

Existing Active Directory

Authentication on WEB UIs: false

KDC hosts [0]: kerberos-ad.example.com

Realm: EXAMPLE.COM

Kadmin server: kerberos-ad.example.com

Kadmin principal: ad_admin

Kadmin password: <secret>

Keytabs directory: /etc/security/keytabs

Additional realms <not set>

Admin DN: cn=ad_admin,ou=users,dc=example,dc=com

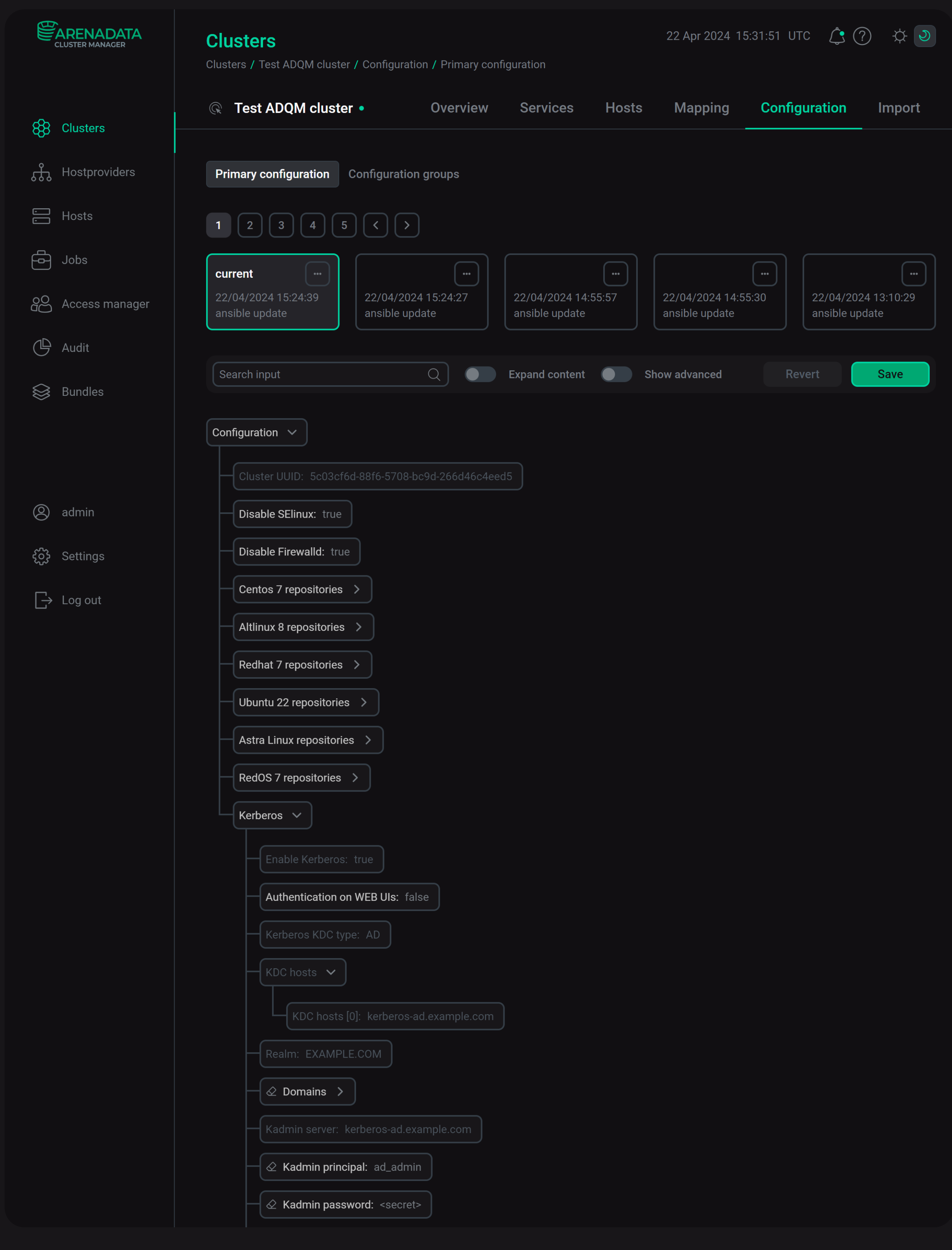
LDAP URL: ldaps://kerberos-ad.example.com

Search input
Expand content
Show advanced
Cancel
Run
Configuration
KDC hosts
+ 1
Domains

Параметры Active Directory

4. Нажмите ***Run*** и дождитесь завершения керберизации. Процесс и результат выполнения задачи ***Manage Kerberos*** отображается на странице ***Jobs***.

После того, как Kerberos успешно установлен на кластере ADQM, соответствующие настройки Kerberos появятся на странице конфигурации кластера (секция ***Kerberos***) и сервиса ADQMDB (секция ***Kerberos configuration***).



Cluster UUID: 5c03cf6d-88f6-5708-bc9d-266d46c4eed5

Disable SElinux: true

Disable Firewalld: true

Enable Kerberos: true

Authentication on WEB UIs: false

Kerberos KDC type: AD

KDC hosts [0]: kerberos-ad.example.com

Realm: EXAMPLE.COM

Kadmin server: kerberos-ad.example.com

Kadmin principal: ad_admin

Kadmin password: <secret>

ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Configuration / Primary configuration
22 Apr 2024 15:31:51 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
1 2 3 4 5
current
22/04/2024 15:24:39
ansible update
22/04/2024 15:24:27
ansible update
22/04/2024 14:55:57
ansible update
22/04/2024 14:55:30
ansible update
22/04/2024 13:10:29
ansible update
Search input
Expand content
Show advanced
Revert
Save
Configuration
Centos 7 repositories
Altlinux 8 repositories
Redhat 7 repositories
Ubuntu 22 repositories
Astra Linux repositories
RedOS 7 repositories
Kerberos
KDC hosts
Domains

Настройки Kerberos в конфигурации кластера ADQM

Kerberos-параметры Active Directory

Следующая таблица содержит описания параметров, необходимых для керберизации кластера ADQM с помощью Active Directory.

Параметр	Описание
KDC hosts	Адрес сервера KDC (при керберизации с применением LDAP этот адрес совпадает с адресом LDAP-сервера)
Realm	Область для аутентификации в Kerberos
Kadmin server	Адрес сервера, на котором установлена система администрирования Kerberos (часто этот адрес совпадает с адресом сервера KDC)
Kadmin principal	Имя принципала пользователя, созданного для подключения кластера к Active Directory с правами создания пользователей и групп в контейнере ***Container DN***
Kadmin password	Пароль принципала пользователя ***Kadmin principal***
Admin DN	Учетная запись в Active Directory для пользователя, используемого для подключения кластера к Active Directory
LDAP URL	Адрес LDAP-сервера (`ldaps://<hostname or IP address>:<port>`)
Container DN	DN контейнера, где хранятся сервисные принципалы
TLS CA certificate Path (опционально)	Локальный путь, по которому на всех хостах кластера хранится сертификат CA для пользователя ***Admin DN*** (если сертификат актуальный, параметр ***TLS CA certificate*** указывать не нужно)
TLS CA certificate (опционально)	Содержимое сертификата CA для пользователя ***Admin DN***, которое запишется на хост в файл по пути, указанному в ***TLS CA certificate Path***

Проверка аутентификации пользователя через Kerberos

После успешной керберизации выполнение ADQM-команд на узлах кластера возможно только после получения соответствующего тикета Kerberos. Ниже приведен пример доступа к кластеру, керберизованному с помощью Active Directory:

1. Создайте пользователя ADQM, который имеет запись в базе данных службы каталогов Active Directory и принципал для заданной области (realm). Для этого используйте выражение `IDENTIFIED WITH kerberos` в запросе `CREATE USER`:

```
CREATE USER adqm_kerb_ad IDENTIFIED WITH kerberos;
```

2. Создайте для принципала пользователя тикет Kerberos:

```
$ kinit -V adqm_kerb_ad@EXAMPLE.COM
```

   Введите пароль:

```
Password for adqm_kerb_ad@EXAMPLE.COM:
```

   Опции команды `kinit` описаны в статье kinit документации MIT Kerberos.

3. Проверьте тикет:

```
$ klist
```

```
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: adqm_kerb_ad@EXAMPLE.COM

Valid starting       Expires              Service principal
05/05/2023 09:05:36  05/06/2023 09:05:36  krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

4. Для проверки аутентификации через Kerberos отправьте в ADQM тестовую команду, используя утилиту `curl` с поддержкой механизма SPNEGO:

```
$ echo "select currentUser()" | curl --negotiate -u : http://<host_name>:8123/ --data-binary @-
```

   Вывод на экран при успешной аутентификации:

```
adqm_kerb_ad
```

Проблема с типом шифрования при керберизации в РЕД ОС

Елена Дворядкина

Содержание

Назад к содержанию

- Причина
- Решение — изменение криптографической политики
  - LEGACY
  - CUSTOM

При установке ADQM на хост с операционной системой РЕД ОС и попытке керберизации кластера с помощью Active Directory керберизация может быть прервана на этапе выполнения команды `kinit` со следующей ошибкой:

```
kinit: KDC has no support for encryption type while getting initial credentials
```

Причина

Проблема возникает из-за разных наборов типов шифрования, разрешенных в Active Directory и РЕД ОС. В РЕД ОС этот набор определяет криптографическая политика (cryptographic policy).

По умолчанию в РЕД ОС установлена криптографическая политика `DEFAULT`:

```
$ update-crypto-policies --show
```

```
DEFAULT
```

Эта политика определяет следующие типы шифрования, которые разрешено использовать протоколу Kerberos:

```
$ cat /usr/share/crypto-policies/DEFAULT/krb5.txt
```

```
[libdefaults]
permitted_enctypes = aes256-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 camellia256-cts-cmac aes128-cts-hmac-sha1-96 aes128-cts-
hmac-sha256-128 camellia128-cts-cmac
```

В списке отсутствует тип шифрования `arcfour` (также известный как `rc4`), который используется при запросе тикетов Kerberos с сервера Active Directory. Этот тип шифрования был удален из политики `DEAFAULT`, так как считается ненадежным — см. Ensuring support for common encryption types in AD and RHEL.

Решение — изменение криптографической политики

Чтобы добавить в РЕД ОС поддержку типа шифрования `arcfour` и решить указанную выше проблему, рекомендуется переключить криптографическую политику РЕД ОС по умолчанию на одну из следующих.

LEGACY

Криптографическая политика `LEGACY` — одна из предопределенных криптографических политик РЕД ОС, которая поддерживает более широкий список типов шифрования, чем `DEFAULT`, и обеспечивает совместимость со старыми версиями и системами.

Типы шифрования для Kerberos, которые поддерживает политика `LEGACY`:

```
$ cat /usr/share/crypto-policies/LEGACY/krb5.txt
```

```
[libdefaults]
permitted_enctypes = aes256-cts-hmac-sha1-96 aes256-cts-hmac-sha384-192 camellia256-cts-cmac aes128-cts-hmac-sha1-96 aes128-cts-
hmac-sha256-128 camellia128-cts-cmac des3-cbc-sha1 arcfour-hmac-md5
```

Чтобы переключить криптографическую политику системы на `LEGACY`, выполните следующую команду как пользователь `root`:

```
$ update-crypto-policies --set LEGACY
```

После переключения криптографической политики необходимо выполнить перезагрузку системы:

```
Setting system policy to LEGACY
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies to fully take place.
```

Учитывайте также, что набор разрешенных типов шифрования будет изменен и для других сервисов, таких как:

```
$ ls /usr/share/crypto-policies/LEGACY/
```

```
bind.txt gnutls.txt java.txt krb5.txt libreswan.txt nss.txt opensshserver.txt openssh.txt opensslcnf.txt openssl.txt
```

CUSTOM

Вы можете создать новую криптографическую политику (например, `CUSTOM`) на основе политики `DEFAULT` и модифицировать список поддерживаемых типов шифрования.

1. Скопируйте каталог политики `DEFAULT` в каталог `CUSTOM`:

```
$ cp -r /usr/share/crypto-policies/DEFAULT/ /usr/share/crypto-policies/CUSTOM
```

2. Добавьте тип `arcfour-hmac-md5` в конец файла */usr/share/crypto-policies/CUSTOM/krb5.txt*.

3. Установите новую криптографическую политику:

```
$ update-crypto-policies --set CUSTOM
```

4. После переключения политики на `CUSTOM` также нужно выполнить перезапуск системы:

```
Setting system policy to CUSTOM
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies to fully take place.
```

Использование FreeIPA для Kerberos

Елена Дворядкина

Содержание

Назад к содержанию

Включение FreeIPA на кластере ADQM

Kerberos-параметры FreeIPA

Проверка интеграции с FreeIPA

FreeIPA — бесплатная система управления идентификацией пользователей для Linux/UNIX с открытым исходным кодом. FreeIPA включает в себя такие компоненты, как Linux (Fedora), 389 Directory Server, MIT Kerberos, NTP, DNS, Dogtag (система сертификатов). Основной задачей FreeIPA является централизованное управление учетными данными и аутентификация.

Для обеспечения безопасности ADQM использует Kerberos, LDAP, SSL-сертификаты/ключи и другие смежные технологии. Для такого набора инструментов FreeIPA отлично подходит в качестве системы управления идентификацией.

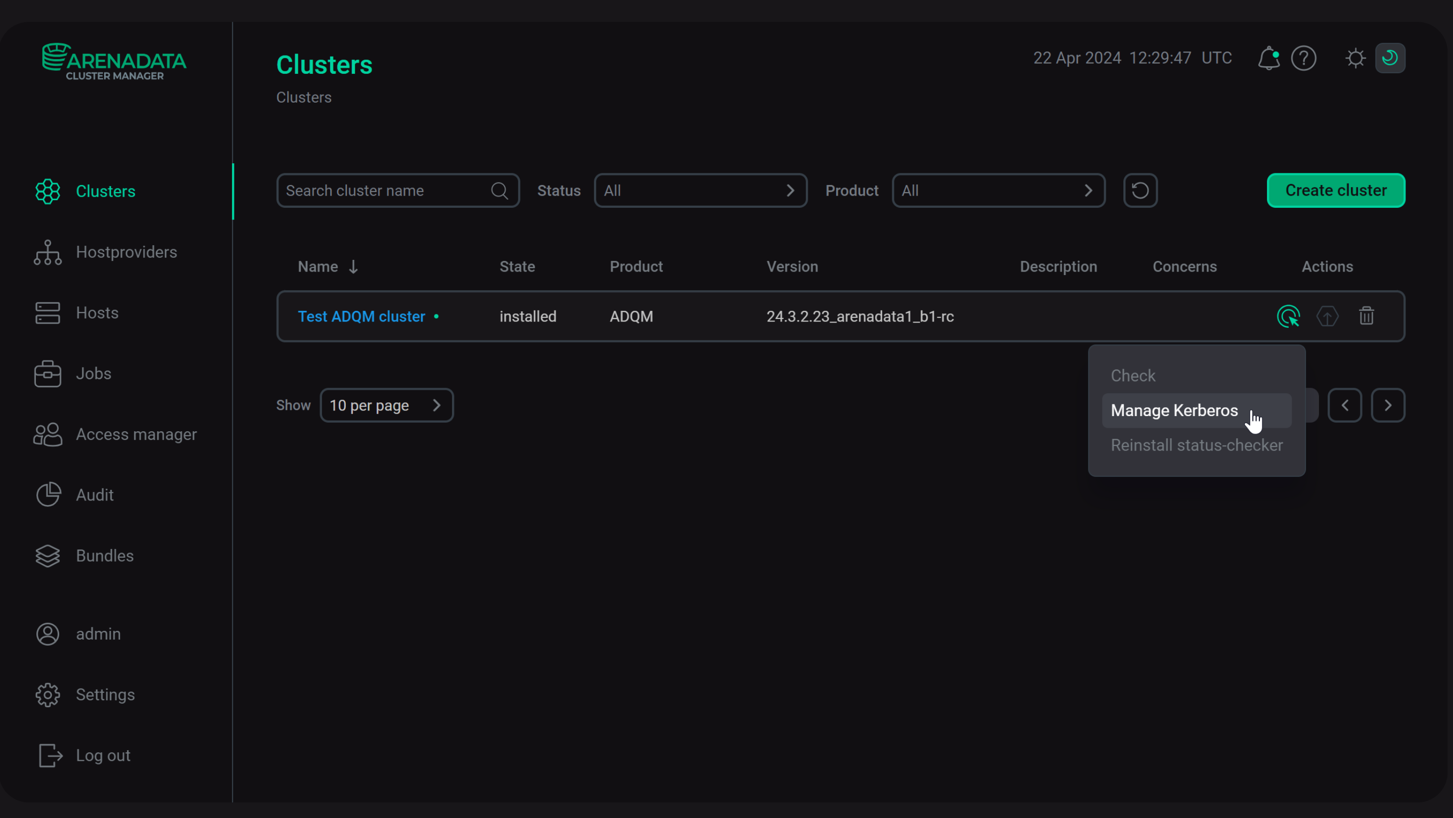
Для керберизации кластера с помощью FreeIPA используется ADCM. В процессе керберизации ADCM устанавливает IPA-клиент на каждый хост кластера и устанавливает соединение с сервером FreeIPA, используя заданные параметры Kerberos для FreeIPA.

В данной статье для примера используется тестовый FreeIPA-сервер, развернутый на хосте *freeipa-server.ru-central1.internal* с операционной системой CentOS. Этот сервер представляет собой базовую конфигурацию `yum install ipa-server` с параметрами установки по умолчанию.

Включение FreeIPA на кластере ADQM

Для керберизации кластера с помощью FreeIPA выполните описанные ниже шаги:

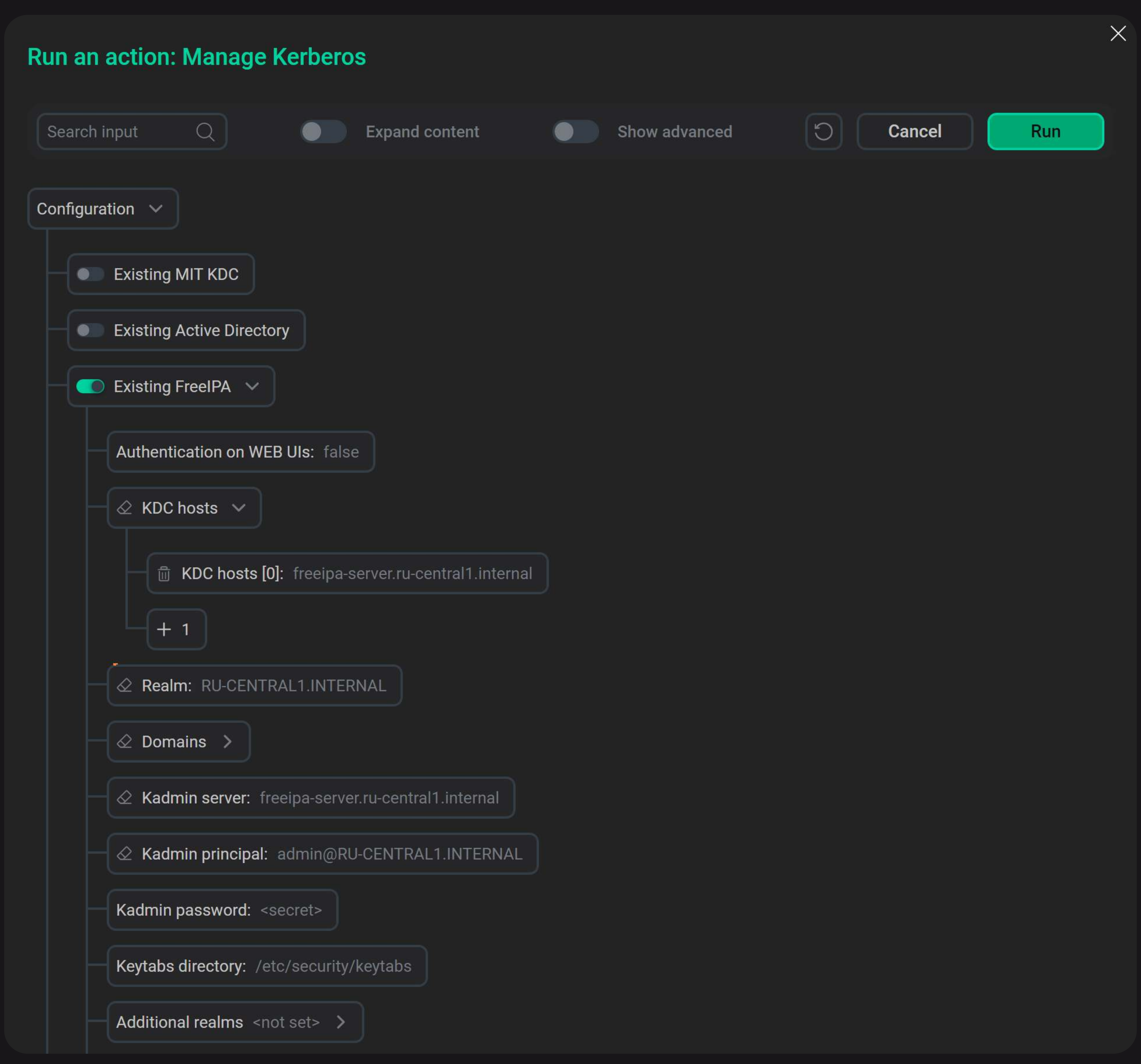
1. В интерфейсе ADCM на странице ***Clusters*** инициируйте включение Kerberos для кластера — нажмите на иконку ***Actions*** и выберите действие ***Manage Kerberos***.



Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	installed	ADQM	24.3.2.23_arenadata1_b1-rc			

Включение Kerberos в кластере ADQM

2. В открывшемся окне включите ***Existing FreeIPA*** и укажите Kerberos-параметры FreeIPA.



Authentication on WEB UIs: false

KDC hosts [0]: freeipa-server.ru-central1.internal

Realm: RU-CENTRAL1.INTERNAL

Kadmin server: freeipa-server.ru-central1.internal

Kadmin principal: admin@RU-CENTRAL1.INTERNAL

Kadmin password: <secret>

Keytabs directory: /etc/security/keytabs

Additional realms <not set>

Параметры FreeIPA

3. Нажмите ***Run*** и дождитесь завершения керберизации. Процесс и результат выполнения задачи ***Manage Kerberos*** отображается на странице ***Jobs***.

После того, как Kerberos успешно установлен на кластере ADQM, соответствующие настройки появятся на странице конфигурации кластера в секции ***Kerberos*** и на странице конфигурации сервиса ADQMDB в секции ***Kerberos configuration***.

Kerberos-параметры FreeIPA

Следующая таблица содержит описания параметров, необходимых для керберизации кластера ADQM с помощью FreeIPA. Актуальные значения для этих параметров можно получить, выполнив команду `ipa user-find <ipa_admin>` на FreeIPA-сервере, где `<ipa_admin>` — имя пользователя IPA Admin.

Параметр	Описание
KDC hosts	Хост (один или более), на котором запущен FreeIPA-сервер
Realm	Kerberos realm для подключения к FreeIPA-серверу
Domains	Домен (один или более), привязанный к FreeIPA
Kadmin server	Адрес хоста, на котором запущен `kadmin`
Kadmin principal	Имя принципала для подключения через `kadmin`, например `admin@RU-CENTRAL1.INTERNAL`
Kadmin password	Пароль пользователя IPA Admin
IpaClient No NTP Autoconf	Отключает настройку NTP при установке IPA-клиентов
IpaClient No DNS Lookup	Отключает поиск сервера FreeIPA по DNS при установке IPA-клиентов

Проверка интеграции с FreeIPA

После успешной керберизации выполнение ADQM-команд на узлах кластера возможно только после получения соответствующего тикета Kerberos. Ниже приведен пример доступа к кластеру, керберизованному с помощью FreeIPA:

1. Создайте пользователя ADQM, который соответствует учетной записи в домене FreeIPA. Для этого используйте выражение `IDENTIFIED WITH kerberos` в запросе `CREATE USER`:

```
CREATE USER adqm_kerb_freeipa IDENTIFIED WITH kerberos;
```

2. Запросите тикет Kerberos для пользователя:

```
$ kinit adqm_kerb_freeipa@RU-CENTRAL1.INTERNAL
```

Введите пароль:

```
Password for adqm_kerb_freeipa@RU-CENTRAL1.INTERNAL:
```

Опции команды `kinit` описаны в статье kinit документации MIT Kerberos.

3. Проверьте тикет:

```
$ klist
```

```
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: adqm_kerb_freeipa@RU-CENTRAL1.INTERNAL

Valid starting       Expires              Service principal
05/10/2023 10:20:30  05/11/2023 10:20:30  krbtgt/RU-CENTRAL1.INTERNAL@RU-CENTRAL1.INTERNAL
```

4. Для проверки аутентификации через Kerberos отправьте в ADQM тестовую команду, используя утилиту `curl` с поддержкой механизма SPNEGO:

```
$ echo "select currentUser()" | curl --negotiate -u : http://<host_name>:8123/ --data-binary @-
```

Вывод на экран при успешной аутентификации:

```
adqm_kerb_freeipa
```

Авторизация

Елена Дворядкина

Содержание

Назад к содержанию

- Настройка авторизации для нового кластера
- Аккаунт пользователя
  - Создание пользователя
  - Изменение существующего пользователя
  - Просмотр настроек пользователя
  - Привилегии пользователя
  - Удаление пользователя
- Роль
  - Управление ролями
  - Привилегии ролей
  - Назначение роли пользователю
  - Применение роли пользователя
- Политика доступа к строкам
  - Создание политики доступа к строкам
  - Изменение политики доступа к строкам
  - Просмотр настроек политики доступа к строкам
  - Удаление политики доступа к строкам
- Профиль настроек
  - Создание профиля настроек
  - Изменение профиля настроек
  - Просмотр параметров профиля настроек
  - Удаление профиля настроек
- Квота
  - Создание квоты
  - Изменение квоты
  - Просмотр информации о квотах
  - Удаление квоты

Система управления доступом к базе данных ADQM, основанная на концепции Role Based Access Control (RBAC), включает следующие объекты:

- Аккаунт пользователя
- Роль
- Политика доступа к строкам
- Профиль настроек
- Квота

Создавать и настраивать эти объекты можно через SQL-запросы (рекомендуется) или конфигурационные файлы *users.xml* и *config.xml*. В данной статье описывается SQL-ориентированное управление доступом — приводится базовый синтаксис и примеры запросов для создания пользователей ADQM и ролей с определенными разрешениями и ограничениями на работу с базами данных, таблицами, столбцами и строками.

Создание таблицы и базы данных для выполнения приведенных ниже примеров

Чтобы воспроизвести приведенные ниже примеры, создайте таблицу `test_table` в текущей базе данных и новую базу данных `test_database`:

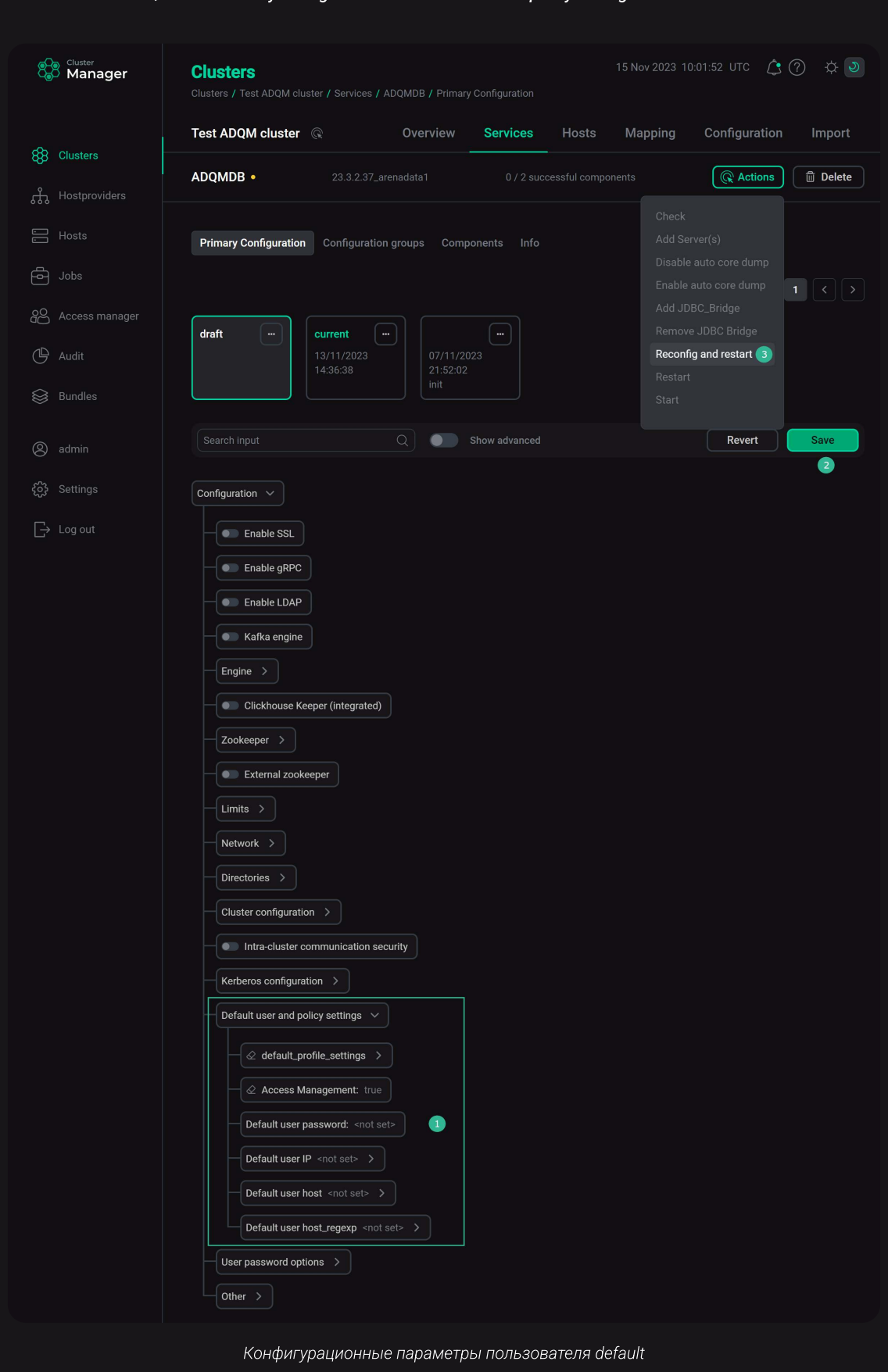
```
CREATE TABLE test_table (x Int32, y String, z Int32) ENGINE = MergeTree ORDER BY x;
```

```
CREATE DATABASE test_database;
```

Настройка авторизации для нового кластера

Чтобы режим управления доступом через SQL-запросы был включен, значение параметра access_management для пользователя должно быть установлено в `1` в файле *users.xml*.

При создании нового кластера ADQM по умолчанию предоставляется аккаунт пользователя `default` без пароля, у которого активирована опция `access_management` и есть все права на управление доступом. Этот аккаунт используется во всех случаях, когда имя пользователя не определено (например, при подключении к ADQM через клиента или выполнении распределенных запросов). Изменить настройки пользователя `default` (в том числе назначить пароль для входа в ADQM) можно через интерфейс ADCM: ***ADQM cluster → Services → ADQMDB → Primary Configuration → Default user and policy settings***.



Конфигурационные параметры пользователя default

Используйте аккаунт пользователя `default`, чтобы настроить безопасный доступ к базе данных ADQM сразу после создания кластера. Для этого выполните следующие действия:

1. Подключитесь к ADQM как пользователь `default` (например, через клиент командной строки clickhouse-client).
2. Создайте аккаунты пользователей (включая аккаунт администратора), необходимые для дальнейшей работы в ADQM.
3. Ограничьте разрешения для пользователя `default` и отключите для него SQL-ориентированное управление доступом (опция ***Access Management*** в интерфейсе ADCM).

Аккаунт пользователя

Аккаунт пользователя (user) — объект системы доступа, который позволяет авторизовать пользователя при его подключении к ADQM. Аккаунт пользователя содержит следующую информацию:

- идентификационную информацию;
- привилегии, определяющие типы запросов, которые пользователю разрешено выполнять;
- хосты, с которых пользователь может подключаться к серверу ADQM;
- назначенные пользователю роли и роли по умолчанию;
- настройки, которые применяются при входе пользователя в систему.

Создание пользователя

Чтобы создать новый аккаунт пользователя в ADQM, используйте запрос `CREATE USER`:

```
CREATE USER [IF NOT EXISTS | OR REPLACE] <user_name1> [ON CLUSTER <cluster_name1>]
    [, <user_name2> [ON CLUSTER <cluster_name2>] ...]
    [NOT IDENTIFIED | IDENTIFIED {[WITH {no_password | plaintext_password | sha256_password | sha256_hash | double_sha1_password | double_sha1_hash}] BY {'password' | 'hash'}} | {WITH ldap SERVER 'ldap_server'} | {WITH ssl_certificate CN '<certificate_CN>'} | {WITH kerberos [REALM 'realm']}]
    [HOST {LOCAL | NAME '<fqdn>' | REGEXP '<name_regexp>' | IP '<ip_address>' | LIKE '<pattern>'} [,...] | ANY | NONE]
    [DEFAULT ROLE <role_name> [,...] | ALL | ALL EXCEPT <role_name> [,...] ]
    [DEFAULT DATABASE <database_name> | NONE]
    [GRANTEES {<user_name> | <role_name> | ANY | NONE} [,...] [EXCEPT {<user_name> | <role_name>} [,...]]]
    [SETTINGS <setting_name> [= <value>] [MIN [=] <min_value>] [MAX [=] <max_value>] [READONLY | WRITABLE] | PROFILE '<profile_name>'] [,...];
```

Основные выражения

ON CLUSTER	По умолчанию создается пользователь, который может подключаться только к тому серверу, на котором был выполнен запрос `CREATE USER`. Чтобы создать пользователя для подключения ко всем серверам кластера, нужно использовать выражение `ON CLUSTER` с указанием имени кластера
NOT IDENTIFIED | IDENTIFIED	Метод аутентификации пользователя: по паролю, через LDAP-сервер, по SSL-сертификату или по протоколу Kerberos
HOST	Хост, с которого можно устанавливать соединение с сервером ADQM
DEFAULT ROLE	Роль по умолчанию (активируются автоматически при входе пользователя в ADQM)
DEFAULT DATABASE	База данных, в которую будет входить пользователь при подключении к ADQM
GRANTEES	Пользователи или роли, которым разрешено получать привилегии от создаваемого пользователя при условии, что этому пользователю предоставлено соответствующее разрешение с использованием WITH GRANT OPTION
SETTINGS	Значения настроек и их ограничения, которые применяются при входе пользователя в систему. Помимо настроек, явно определенных через выражение `SETTINGS`, при входе пользователя также применяются настройки назначенных ему ролей и настройки из профилей настроек (`CREATE USER <user_name> SETTINGS PROFILE <profile_name>`)

Пример

Создайте пользователя `john`, который может подключаться к любому серверу кластера `default_cluster`, используя пароль `qwerty`:

```
CREATE USER john ON CLUSTER default_cluster IDENTIFIED WITH sha256_password BY 'qwerty';
```

Изменение существующего пользователя

Изменить настройки существующего пользователя можно через запрос `ALTER USER`:

```
ALTER USER [IF EXISTS] <user_name1> [ON CLUSTER <cluster_name1>] [RENAME TO <new_user_name1>]
    [, <user_name2> [ON CLUSTER <cluster_name2>] [RENAME TO <new_user_name2>] ...]
    [NOT IDENTIFIED | IDENTIFIED {[WITH {no_password | plaintext_password | sha256_password | sha256_hash | double_sha1_password | double_sha1_hash}] BY {'password' | 'hash'}} | {WITH ldap SERVER 'ldap_server'} | {WITH ssl_certificate CN '<certificate_CN>'} | {WITH kerberos [REALM 'realm']}]
    [[ADD | DROP] HOST {LOCAL | NAME '<fqdn>' | REGEXP '<name_regexp>' | IP '<ip_address>' | LIKE '<pattern>'} [,...] | ANY | NONE]
    [DEFAULT ROLE <role_name> [,...] | ALL | ALL EXCEPT <role_name> [,...] ]
    [GRANTEES {<user_name> | <role_name> | ANY | NONE} [,...] [EXCEPT {<user_name> | <role_name>} [,...]]]
    [SETTINGS <setting_name> [= <value>] [MIN [=] <min_value>] [MAX [=] <max_value>] [READONLY | WRITABLE] | PROFILE '<profile_name>'] [,...]
```

Пример

Следующий запрос дополнительно настраивает ранее созданного пользователя `john` так, что теперь он может подключаться к серверам ADQM только с указанного IP-адреса:

```
ALTER USER john HOST IP '10.92.17.140';
```

Просмотр настроек пользователя

Посмотреть текущие настройки пользователя (параметры, указанные при создании пользователя, с учетом изменений, выполненных через `ALTER USER`), можно с помощью запроса `SHOW CREATE USER`:

```
SHOW CREATE USER [<user_name1> [, <user_name2> ...] | CURRENT_USER];
```

Пример

Выполните следующий запрос, чтобы посмотреть настройки пользователя `john`:

```
SHOW CREATE USER john;
```

Вывод на экран:

```
--CREATE USER john---------------------------------------------------------
│ CREATE USER john IDENTIFIED WITH sha256_password HOST IP '10.92.17.140' │
---------------------------------------------------------------------------
```

Привилегии пользователя

Чтобы определить права доступа пользователя к объектам ADQM, нужно присвоить ему **привилегии** — разрешения на выполнение определенных типов запросов.

Предоставление привилегий пользователю

Назначить пользователю привилегии можно двумя способами:

- присвоить роль — совокупность привилегий;
- назначить привилегии напрямую через запрос GRANT:

```
GRANT [ON CLUSTER <cluster_name>] <privilege>[(<column_name> [,...])] [,...]
    ON {<db_name>.<table_name> | <db_name>.* | *.* | <table_name> | *} TO {<user_name> | CURRENT_USER} [,...] [WITH GRANT OPTION] [WITH REPLACE OPTION];
```

Аргументы и выражения

<privilege>	Тип привилегии
<column_name>	Имя столбца, для которого разрешено выполнение запросов, определяемых привилегией
<db_name>.<table_name>	Имя базы данных и имя таблицы, определяющие область действия привилегии
<user_name>	Имя пользователя, которому назначается привилегия
WITH GRANT OPTION	Пользователю разрешается выполнять запрос `GRANT`, то есть он может передавать свои привилегии той же или меньшей области действия другим пользователям и ролям, указанным как `GRANTEES` при выполнении запроса CREATE USER / ALTER USER
WITH REPLACE OPTION	Все старые привилегии пользователя заменяются новыми привилегиями (если выражение не используется в запросе, добавляются новые привилегии)

Примеры:

- Выдайте пользователю `john` привилегию на выполнение запросов `SELECT` для столбцов `x` и `y` таблицы `test_table` в текущей базе данных:

```
GRANT SELECT(x,y) ON test_table TO john;
```

- Создайте новый аккаунт пользователя `admin` и предоставьте ему полные права администратора:

```
CREATE USER admin IDENTIFIED BY 'admin';
```

```
GRANT ALL ON *.* TO admin WITH GRANT OPTION;
```

Просмотр привилегий пользователя

Набор привилегий пользователя складывается из привилегий, назначенных ему напрямую, и привилегий, присвоенных через роли. Используйте запрос SHOW GRANTS, чтобы посмотреть список привилегий пользователя (если в запросе пользователь не указан, возвращается список привилегий текущего пользователя):

```
SHOW GRANTS [FOR <user_name>];
```

Пример запроса:

```
SHOW GRANTS FOR john;
```

Вывод на экран:

```
--GRANTS FOR john--------------------------------
│ GRANT SELECT(x,y) ON default.test_table TO john │
-------------------------------------------------
```

Отзыв привилегий пользователя

Чтобы отозвать привилегию, используйте запрос `REVOKE`:

```
REVOKE [ON CLUSTER <cluster_name>] <privilege>[(<column_name> [,...])] [,...]
    ON {<db_name>.<table_name> | <db_name>.* | *.* | <table_name> | *}
    FROM {<user_name> | CURRENT_USER} [,...] | ALL | ALL EXCEPT {<user_name> | CURRENT_USER} [,...];
```

Пример запроса, который отзывает у пользователя `john` привилегию на чтение данных из столбца `y` таблицы `test_table`:

```
REVOKE SELECT(y) ON test_table FROM john;
```

Удаление пользователя

Удалить аккаунт пользователя можно выполнив запрос `DROP USER`:

```
DROP USER [IF EXISTS] <user_name> [,...] [ON CLUSTER <cluster_name>];
```

Например, следующий запрос удаляет аккаунт пользователя `john` на кластере `default_cluster`:

```
DROP USER john ON CLUSTER default_cluster;
```

Роль

Роль (role) — набор привилегий, которые получает пользователь, если ему назначена эта роль.

Управление ролями

В следующей таблице перечислены запросы, с помощью которых можно управлять ролями ADQM.

Запрос	Описание
CREATE ROLE	Создает новую роль в ADQM
ALTER ROLE	Изменяет настройки существующей роли
SHOW CREATE ROLE	Выводит на экран параметры, использованные при создании роли (с учетом изменений, если выполнялся запрос `ALTER ROLE`)
DROP ROLE	Удаляет роль. При удалении роль отзывается у всех объектов системы доступа, которым она присвоена

Синтаксис

CREATE ROLE

```
CREATE ROLE [IF NOT EXISTS | OR REPLACE] <role_name1> [ON CLUSTER <cluster_name1>]
    [, <role_name2> [ON CLUSTER <cluster_name2>] ...]
    [SETTINGS <setting_name> [= <value>] [MIN [=] <min_value>] [MAX [=] <max_value>]
    [CONST|READONLY|WRITABLE|CHANGEABLE_IN_READONLY] | PROFILE '<profile_name>'] [,...];
```

ALTER ROLE

```
ALTER ROLE [IF EXISTS] <role_name1> [ON CLUSTER <cluster_name1>] [RENAME TO <new_role_name1>]
    [, <role_name2> [ON CLUSTER <cluster_name2>] [RENAME TO <new_role_name2>] ...]
    [SETTINGS <setting_name> [= <value>] [MIN [=] <min_value>] [MAX [=] <max_value>]
    [CONST|READONLY|WRITABLE|CHANGEABLE_IN_READONLY] | PROFILE '<profile_name>'] [,...];
```

SHOW CREATE ROLE

```
SHOW CREATE ROLE <role_name> [,...];
```

DROP ROLE

```
DROP ROLE [IF EXISTS] <role_name> [,...] [ON CLUSTER <cluster_name>];
```

Привилегии ролей

Чтобы определить привилегии, которые будет предоставлять роль при назначении пользователю, используйте запрос `GRANT`:

```
GRANT [ON CLUSTER <cluster_name>] <privilege>[(<column_name> [,...])] [,...] ON {<db_name>.<table_name> | <db_name>.* | *.* | <table_name> | *}
    TO <role_name> [,...] [WITH GRANT OPTION] [WITH REPLACE OPTION];
```

Аргументы и выражения

<privilege>	Тип привилегии
<column_name>	Имя столбца, для которого разрешено выполнение запросов, определяемых привилегией
<db_name>.<table_name>	Имя базы данных и имя таблицы, определяющие область действия привилегии
<role_name>	Имя роли, которой назначается привилегия
WITH GRANT OPTION	Пользователю, которому будет назначена роль, разрешается выполнять запрос `GRANT` (то есть он сможет передавать свои привилегии той же или меньшей области действия другим пользователям и ролям)
WITH REPLACE OPTION	Все старые привилегии роли заменяются новыми привилегиями (если выражение не используется в запросе, добавляются новые привилегии)

Пример

Создайте роль и назначьте ей привилегию на чтение всех таблиц базы данных `test_database`:

```
CREATE ROLE reader;
```

```
GRANT SELECT ON test_database.* TO reader;
```

Назначение роли пользователю

Одному пользователю можно назначить несколько ролей. Роль также может быть присвоена другой роли. Назначать и отзывать роли можно с помощью следующих запросов:

Запрос	Описание
GRANT	Назначает роль пользователю или другой роли. ■ `WITH ADMIN OPTION` — пользователю/роли присваивается привилегия `ADMIN OPTION`, которая разрешает пользователю назначать свои роли другому пользователю. ■ `WITH REPLACE OPTION` — все старые роли пользователя/роли заменяются новыми ролями (если опция не указана, добавляются новые роли).
REVOKE	Отзывает роль у пользователя или другой роли

Синтаксис

GRANT

```
GRANT [ON CLUSTER <cluster_name>] <role_name> [,...]
    TO {<user_name> | <another_role_name> | CURRENT_USER} [,...]
    [WITH ADMIN OPTION] [WITH REPLACE OPTION];
```

REVOKE

```
REVOKE [ON CLUSTER <cluster_name>] [ADMIN OPTION FOR] <role_name> [,...]
    FROM {<user_name> | <another_role_name> | CURRENT_USER} [,...]
        | ALL | ALL EXCEPT {<user_name> | <another_role_name> | CURRENT_USER} [,...];
```

Пример

Назначьте роль `reader` пользователю `john`:

```
GRANT reader TO john;
```

Убедитесь, что к ранее назначенной привилегии пользователя добавилась роль `reader`:

```
SHOW GRANTS FOR john;
```

Результат выполнения запроса:

```
--GRANTS FOR john--------------------------------
│ GRANT SELECT(x,y) ON default.test_table TO john │
│ GRANT reader TO john                            │
-------------------------------------------------
```

Применение роли пользователя

Пользователи могут применять (активировать) назначенные роли в произвольных комбинациях с помощью запросов `SET ROLE` и `SET DEFAULT ROLE`. Конечный объем доступных пользователю действий определяется комбинацией привилегий, назначенных аккаунту пользователя напрямую, и привилегий всех примененных ролей.

Запрос	Описание
SET ROLE	Активирует роль для текущего пользователя
SET DEFAULT ROLE	Устанавливает роль по умолчанию для пользователя. Роли по умолчанию активируются автоматически при входе пользователя в ADQM. Ролями по умолчанию могут быть установлены только ранее назначенные пользователю роли. Установить роль по умолчанию можно также с помощью выражения `DEFAULT ROLE` запроса CREATE USER или ALTER USER

Синтаксис

SET ROLE

```
SET ROLE {DEFAULT | NONE | <role_name> [,...] | ALL | ALL EXCEPT <role_name> [,...]};
```

SET DEFAULT ROLE

```
SET DEFAULT ROLE {NONE | <role_name> [,...] | ALL | ALL EXCEPT <role_name> [,...]}
    TO {<user_name>
```

Пример

После того, как пользователю `john` была назначена роль `reader` (см. пример выше), он может применить эту роль и выполнять разрешенные ей запросы:

```
SET ROLE reader;
```

```
SELECT * FROM test_database.*;
```

Политика доступа к строкам

Политика доступа к строкам — фильтр для конкретной таблицы, определяющий, какие строки доступны пользователю или роли.

Создание политики доступа к строкам

Чтобы создать политику доступа к строкам, используйте запрос `CREATE ROW POLICY`:

```
CREATE [ROW] POLICY [IF NOT EXISTS | OR REPLACE] <policy_name1> [ON CLUSTER <cluster_name1>] ON [<db_name1>.]<table_name1>
        [, <policy_name2> [ON CLUSTER <cluster_name2>] ON [<db_name2>.]<table_name2> ...]
    [AS {PERMISSIVE | RESTRICTIVE}]
    [FOR SELECT] USING <filter_condition>
    [TO {<user_name> | <role_name> [,...] | ALL | ALL EXCEPT <user_name> | <role_name> [,...]}];
```

Основные выражения

AS	Выражение, которое определяет, как комбинировать политики, если одновременно активно более одной политики для одной и той же таблицы и одного и того же пользователя. Возможные значения: ■ `PERMISSIVE` (по умолчанию) — политики комбинируются с использованием логического оператора `OR`; ■ `RESTRICTIVE` — политики комбинируются с использованием логического оператора `AND`. Иными словами, строка таблицы будет доступна для пользователя, если выполняется условие `any_of(PERMISSIVE_filters) AND all_of(RESTRICTIVE_filters)`. Например, если заданы следующие политики, пользователь `john` будет видеть только те строки, для которых одновременно выполняются условия `x=1` и `y=2`:
USING	Условие для фильтрации строк. Пользователь может видеть строку, если указанное условие, вычисленное для строки, возвращает ненулевой результат
TO	Пользователи и роли, для которых должна действовать политика

AS

```
CREATE ROW POLICY policy_1 ON test_table USING x=1 TO mary, john;
CREATE ROW POLICY policy_2 ON test_table USING y=2 AS RESTRICTIVE TO john, andrew;
```

Если для таблицы не указано ни одной политики доступа к строкам, любой пользователь может выполнить команду `SELECT` и получить все строки таблицы. Если создать хотя бы одну политику для таблицы, то доступ к строкам будет управляться этой политикой для всех пользователей (в том числе тех, для которых политика явно не указана).

Например, если определена политика:

```
CREATE ROW POLICY policy_1 ON test_table USING x=1 TO mary, john;
```

то:

- пользователи `mary` и `john` смогут видеть только строки, для которых выполняется условие `x=1`;
- все остальные пользователи вообще не будут иметь доступ к строкам таблицы `test_table`.

Чтобы скорректировать это поведение и предоставить другим пользователям доступ ко всем строкам, необходимо настроить дополнительную политику для этих пользователей. Например, создайте следующую политику для пользователя `admin`:

```
CREATE ROW POLICY allow_admin_filter ON test_table USING 1 TO admin;
```

Изменение политики доступа к строкам

Запрос `ALTER ROW POLICY` позволяет изменить существующую политику доступа к строкам таблицы:

```
ALTER [ROW] POLICY [IF EXISTS] <policy_name1> [ON CLUSTER <cluster_name1>] ON [<db_name1>.]<table_name1> [RENAME TO <new_policy_name1>]
        [, <policy_name2> [ON CLUSTER <cluster_name2>] ON [<db_name2>.]<table_name2> [RENAME TO <new_policy_name2>] ...]
    [AS {PERMISSIVE | RESTRICTIVE}]
    [FOR SELECT]
    [USING {<filter_condition> | NONE}][,...]
    [TO {<user_name> | <role_name> [,...] | ALL | ALL EXCEPT <user_name> | <role_name> [,...]}];
```

Просмотр настроек политики доступа к строкам

Посмотреть текущие настройки политики доступа к строкам определенной таблицы можно с помощью запроса `SHOW CREATE ROW POLICY`:

```
SHOW CREATE [ROW] POLICY <policy_name> ON [<db_name1>.]<table_name1> [, [<db_name2>.]<table_name2> ...];
```

Удаление политики доступа к строкам

Чтобы удалить политику доступа к строкам таблицы, выполните запрос `DROP ROW POLICY`:

```
DROP [ROW] POLICY [IF EXISTS] <policy_name> [,...] ON [<db_name>.]<table_name> [,...] [ON CLUSTER <cluster_name>];
```

Профиль настроек

Профиль настроек — набор настроек с указанными значениями и/или ограничениями, который назначается пользователю (или роли) и применяется при входе пользователя в систему.

Создание профиля настроек

Выполните запрос `CREATE SETTINGS PROFILE`, чтобы создать профиль настроек. Синтаксис запроса:

```
CREATE SETTINGS PROFILE [IF NOT EXISTS | OR REPLACE] <profile_name1> [ON CLUSTER <cluster_name1>]
    [, <profile_name2> [ON CLUSTER <cluster_name2>] ...]
    [SETTINGS <setting_name> [= <value>] [MIN [=] <min_value>] [MAX [=] <max_value>]
[CONST|READONLY|WRITABLE|CHANGEABLE_IN_READONLY] | INHERIT '<profile_name>'] [,...]
    [TO {<role_name> | <user_name> | CURRENT_USER} [,...] | ALL | ALL EXCEPT {<role_name> | <user_name> | CURRENT_USER} [,...]];
```

Пример

Создайте профиль настроек `my_profile`, который содержит значение настройки `max_memory_usage` и запрещает менять эту переменную, и присвойте его пользователю `john`. Есть несколько способов это сделать.

- Выполните запрос `CREATE SETTINGS PROFILE`, используя выражение `TO`, чтобы указать пользователя:

```
CREATE SETTINGS PROFILE my_profile SETTINGS max_memory_usage = 100000000 READONLY TO john;
```

- Создайте профиль настроек, а затем назначьте его пользователю с помощью выражения `SETTINGS PROFILE` в запросе `CREATE USER` или `ALTER USER`:

```
CREATE SETTINGS PROFILE my_profile SETTINGS max_memory_usage = 100000000 READONLY;
```

```
ALTER USER john SETTINGS PROFILE my_profile;
```

- Профиль настроек можно также назначить роли, которая в свою очередь присваивается пользователю:

```
ALTER ROLE reader SETTINGS PROFILE my_profile;
```

Изменение профиля настроек

При необходимости изменить существующий профиль настроек используйте запрос `ALTER SETTINGS PROFILE`:

```
ALTER SETTINGS PROFILE [IF EXISTS] TO <profile_name1> [ON CLUSTER <cluster_name1>] [RENAME TO <new_profile_name1>]
    [, <profile_name2> [ON CLUSTER <cluster_name2>] [RENAME TO <new_profile_name2>] ...]
    [SETTINGS <setting_name> [= <value>] [MIN [=] <min_value>] [MAX [=] <max_value>]
[CONST|READONLY|WRITABLE|CHANGEABLE_IN_READONLY] | INHERIT '<profile_name>'] [,...]
    [TO {<role_name> | <user_name> | CURRENT_USER} [,...] | ALL | ALL EXCEPT {<role_name> | <user_name> | CURRENT_USER} [,...]];
```

Просмотр параметров профиля настроек

Посмотреть параметры, использованные при создании профиля настроек, можно с помощью запроса `SHOW CREATE SETTINGS PROFILE`:

```
SHOW CREATE [SETTINGS] PROFILE <profile_name1> [, <profile_name2> ...];
```

Удаление профиля настроек

Запрос `DROP SETTINGS PROFILE` удаляет профиль настроек:

```
DROP [SETTINGS] PROFILE [IF EXISTS] <profile_name> [,...] [ON CLUSTER <cluster_name>];
```

При удалении профиль настроек отзывается у всех объектов системы доступа, которым он присвоен.

Квота

Квота — набор параметров, для которых ведется подсчет значений или устанавливаются ограничивающие значения, которые не должны быть превышены за некоторый интервал времени.

Квота присваивается пользователю или роли и позволяет учитывать или ограничивать потребление ресурсов (например, количество обработанных запросов, строк и прочитанных байт) за определенный интервал времени для этого пользователя или роли:

- **Подсчет значений параметров за интервал времени.** В течение указанного интервала времени соответствующие данные считаются после выполнения каждого запроса и выводятся в лог сервера. Когда интервал заканчивается, все накопленные значения сбрасываются, и для следующего интервала расчет квоты начинается заново.
- **Ограничение значений параметров за интервал времени.** Если в течение указанного интервала времени значение параметра превысило заданное ограничение, выводится сообщение о том, значение какого параметра и на сколько превышено, и когда начнется новый интервал (когда можно будет снова выполнять запросы). В рамках одной квоты можно указать несколько интервалов с разными ограничениями.

При распределенной обработке запроса учитываются ресурсы, потраченные на всех удаленных серверах. Накопленные величины хранятся на сервере-инициаторе запроса (если пользователь зайдет на другой сервер — там квота будет действовать "с нуля").

Создание квоты

Используйте запрос `CREATE QUOTA`, чтобы создать квоту и присвоить ее пользователю или роли:

```
CREATE QUOTA [IF NOT EXISTS | OR REPLACE] <quota_name> [ON CLUSTER <cluster_name>]
    [KEYED BY {user_name | ip_address | client_key | client_key, user_name | client_key, ip_address} | NOT KEYED]
    [FOR [RANDOMIZED] INTERVAL <number> {second | minute | hour | day | week | month | quarter | year}
        {MAX { {queries | query_selects | query_inserts | errors | result_rows | result_bytes | read_rows | read_bytes | execution_time} = <max_value> } [,...] |
        NO LIMITS | TRACKING ONLY} [,...]]
    [TO {<role_name> | <user_name> | CURRENT_USER} [,...] | ALL | ALL EXCEPT {<role_name> | <user_name> | CURRENT_USER} [,...]];
```

Основные выражения

KEYED BY	Ключ квоты, по которому определяется совместное использование квоты. Учет ресурсов для каждого значения ключа квоты ведется независимо. Иными словами, соединения с одинаковым ключом квоты совместно используют один и тот же объем ресурсов, для которого выделена одна и та же квота. Ключи `user_name`, `ip_address`, `client_key`, `client_key, user_name` и `client_key, ip_address` соответствуют полям таблицы system.quotas
FOR INTERVAL	Интервал времени, для которого подсчитывается или ограничивается использование ресурсов. `queries`, `query_selects`, `query_inserts`, `errors`, `result_rows`, `result_bytes`, `read_rows`, `read_bytes`, `execution_time` — параметры (соответствуют полям таблицы system.quotas_usage), значения которых можно ограничить для интервала времени
TO	Пользователи и роли, для которых выделяется квота

Пример

Создайте квоту, ограничивающую максимальное количество запросов для пользователя `john` — не более 10 запросов каждые 15 минут:

```
CREATE QUOTA my_quota FOR INTERVAL 15 minute MAX queries = 10 TO john;
```

Если пользователь выполнил 10 запросов до того, как истекли 15 минут с начала действия квоты, при попытке выполнить одиннадцатый запрос появится сообщение:

```
Code: 201. DB::Exception: Received from localhost:9000.
DB::Exception: Quota for user `john` for 900s has been exceeded: queries = 11/10.
Interval will end at 2023-02-08 07:15:00. Name of quota template: `my_quota`. (QUOTA_EXCEEDED)
```

Изменение квоты

Изменить настройки существующей квоты можно, выполнив запрос `ALTER QUOTA`:

```
ALTER QUOTA [IF EXISTS] <quota_name> [ON CLUSTER <cluster_name>]
    [RENAME TO <new_quota_name>]
    [KEYED BY {user_name | ip_address | client_key | client_key, user_name | client_key, ip_address} | NOT KEYED]
    [FOR [RANDOMIZED] INTERVAL <number> {second | minute | hour | day | week | month | quarter | year}
        {MAX { {queries | query_selects | query_inserts | errors | result_rows | result_bytes | read_rows | read_bytes | execution_time} = <max_value> } [,...] |
        NO LIMITS | TRACKING ONLY} [,...]]
    [TO {<role_name> | <user_name> | CURRENT_USER} [,...] | ALL | ALL EXCEPT {<role_name> | <user_name> | CURRENT_USER} [,...]];
```

Просмотр информации о квотах

- **Список имен всех квот.** Выполните запрос `SHOW QUOTAS`, чтобы получить имена всех существующих квот:

```
SHOW QUOTAS;
```

- **Текущие настройки квоты.** Параметры квоты (с учетом всех изменений, если для квоты выполнялся запрос `ALTER QUOTA`) можно получить одним из следующих способов:
  - с помощью запроса `SHOW CREATE QUOTA`:

```
SHOW CREATE QUOTA [<quota_name> [, ...] | CURRENT];
```

  - из системной таблицы system.quotas:

```
SELECT * FROM system.quotas [WHERE name = '<quota_name>'] [FORMAT <format_name>];
```

- **Данные об использовании квоты.** Узнать сколько ресурсов использовано и сколько осталось можно:
  - с помощью запроса `SHOW QUOTA` (использование квоты для всех пользователей или только для текущего пользователя):

```
SHOW [CURRENT] QUOTA;
```

  - из системной таблицы system.quotas_usage (использование квоты всеми пользователями):

```
SELECT * FROM system.quotas_usage [WHERE quota_name = '<quota_name>'] [FORMAT <format_name>];
```

  - из системной таблицы system.quota_usage (использование квоты текущим пользователем):

```
SELECT * FROM system.quota_usage [WHERE quota_name = '<quota_name>'] [FORMAT <format_name>];
```

Удаление квоты

Запрос `DROP QUOTA` удаляет квоту:

```
DROP QUOTA [IF EXISTS] <quota_name> [,...] [ON CLUSTER <cluster_name>];
```

Шифрование соединений с базой данных

Елена Дворядкина

Содержание

Назад к содержанию

- Создание SSL-сертификатов
- Конфигурирование ADQM для включения SSL
  - Действие Manage SSL
  - Параметры SSL в сервисных конфигурациях
  - Пример
- Проверка соединения
  - Подключение через clickhouse-client
  - Проверка межсерверных подключений
  - Подключение по HTTPS
- Отключение SSL

Канал обмена данными между сервером ADQM и клиентом, а также между серверами кластера можно защитить шифрованием SSL. Эта статья описывает как настроить ADQM, чтобы обеспечить проверку SSL-сертификатов для входящих соединений и использовать безопасные протоколы для межсерверного взаимодействия. Для примера используется кластер ADQM, включающий следующие хосты:

- 10.92.17.146, adqm-host-1.arenadata.local;
- 10.92.17.106, adqm-host-2.arenadata.local;
- 10.92.17.140, adqm-host-3.arenadata.local;
- 10.92.16.245, adqm-host-4.arenadata.local.

ПРИМЕЧАНИЕ

Для демонстрационных целей в этой статье серверные SSL-сертификаты подписываются самоподписанным (self-signed) корневым сертификатом центра сертификации (Certification Authority, CA). В реальных системах используйте доверенный центр сертификации для подписи сертификатов.

Создание SSL-сертификатов

Для выполнения приведенных команд необходимо установить `openssl` (минимальная версия 1.1.1) — инструмент командной строки для использования функций криптографической библиотеки OpenSSL.

1. Сгенерируйте закрытый ключ:

```
$ openssl genrsa -out arenadata_ca.key 2048
```

Этот ключ будет использоваться для создания корневого сертификата центра сертификации, а также для подписи других сертификатов (серверных сертификатов и сертификатов для аутентификации пользователей), поэтому храните его в безопасном месте.

2. Создайте самоподписанный корневой сертификат CA:

```
$ openssl req \
  -x509 \
  -subj "/CN=arenadata.local CA" \
  -nodes \
  -key arenadata_ca.key \
  -days 1095 \
  -out arenadata_ca.crt
```

3. Для каждого сервера кластера сгенерируйте ключ и создайте запрос на подпись сертификата (Certificate Signing Request, CSR):

```
$ openssl req -newkey rsa:2048 -nodes -subj "/CN=adqm-host-1.arenadata.local" -addext "subjectAltName = DNS:adqm-host-1.arenadata.local,IP:10.92.17.146" -keyout adqm-host-1.key -out adqm-host-1.csr
$ openssl req -newkey rsa:2048 -nodes -subj "/CN=adqm-host-2.arenadata.local" -addext "subjectAltName = DNS:adqm-host-2.arenadata.local,IP:10.92.17.106" -keyout adqm-host-2.key -out adqm-host-2.csr
$ openssl req -newkey rsa:2048 -nodes -subj "/CN=adqm-host-3.arenadata.local" -addext "subjectAltName = DNS:adqm-host-3.arenadata.local,IP:10.92.17.140" -keyout adqm-host-3.key -out adqm-host-3.csr
$ openssl req -newkey rsa:2048 -nodes -subj "/CN=adqm-host-4.arenadata.local" -addext "subjectAltName = DNS:adqm-host-4.arenadata.local,IP:10.92.16.245" -keyout adqm-host-4.key -out adqm-host-4.csr
```

В качестве идентификатора сертификата CN (Common Name) нужно указать полное доменное имя сервера (Fully Qualified Domain Name, FQDN), для которого создается сертификат. Иначе сертификат нельзя будет проверить.

4. Для каждого сервера кластера создайте и подпишите сертификат, используя закрытый ключ и корневой сертификат CA:

```
$ openssl x509 -req -in adqm-host-1.csr -out adqm-host-1.crt -CAcreateserial -CA arenadata_ca.crt -CAkey arenadata_ca.key -days 365
$ openssl x509 -req -in adqm-host-2.csr -out adqm-host-2.crt -CAcreateserial -CA arenadata_ca.crt -CAkey arenadata_ca.key -days 365
$ openssl x509 -req -in adqm-host-3.csr -out adqm-host-3.crt -CAcreateserial -CA arenadata_ca.crt -CAkey arenadata_ca.key -days 365
$ openssl x509 -req -in adqm-host-4.csr -out adqm-host-4.crt -CAcreateserial -CA arenadata_ca.crt -CAkey arenadata_ca.key -days 365
```

5. На каждый хост кластера ADQM скопируйте корневой сертификат, а также сертификат и ключ соответствующего сервера. Эти файлы должны храниться по одинаковому пути и называться одинаково на каждом хосте кластера — пути и названия файлов корневого сертификата, сертификата и ключа сервера указываются в настройках кластера (см. секцию Параметры SSL в сервисных конфигурациях).

Например, на хост *adqm-host-1.arenadata.local* скопируйте:

- *arenadata_ca.crt* в */etc/clickhouse-server/ca.crt*;
- *adqm-host-1.crt* в */etc/clickhouse-server/server.crt*;
- *adqm-host-1.key* в */etc/clickhouse-server/server.key*.

Файлы */etc/clickhouse-server/server.crt* и */etc/clickhouse-server/server.key* по умолчанию указаны в параметрах ***Server certificate file*** и ***Server private key file*** сервиса ADQMDB соответственно. Если на хостах ADQM файлы сертификата и ключа имеют другие названия и/или расположены в другой директории, соответствующие пути нужно указать в этих параметрах.

Самоподписанный корневой сертификат необходимо указать одним из следующих способов:

- Передать путь и название файла корневого сертификата в параметр ***Certificate authority file*** конфигурации сервиса ADQMDB (см. пример ниже), который по умолчанию имеет пустое значение.
- Добавить корневой сертификат в доверенные сертификаты. Например, на CentOS это выполняется следующими командами:

```
$ sudo cp ./arenadata_ca.crt /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust
```

ПРИМЕЧАНИЕ

После включения SSL для сервиса ADQMDB (см. следующий раздел) директориям, где хранятся используемые SSL-сертификаты и ключи, а также файлам сертификатов и ключей автоматически назначаются владелец и группа `clickhouse` и следующие права доступа:

- `0770` — для директорий, где хранятся серверные и клиентские сертификаты и ключи;
- `0771` — для директорий, где хранятся корневые сертификаты;
- `0600` — для файлов серверных и клиентских сертификатов и ключей;
- `0664` — для файлов корневых сертификатов.

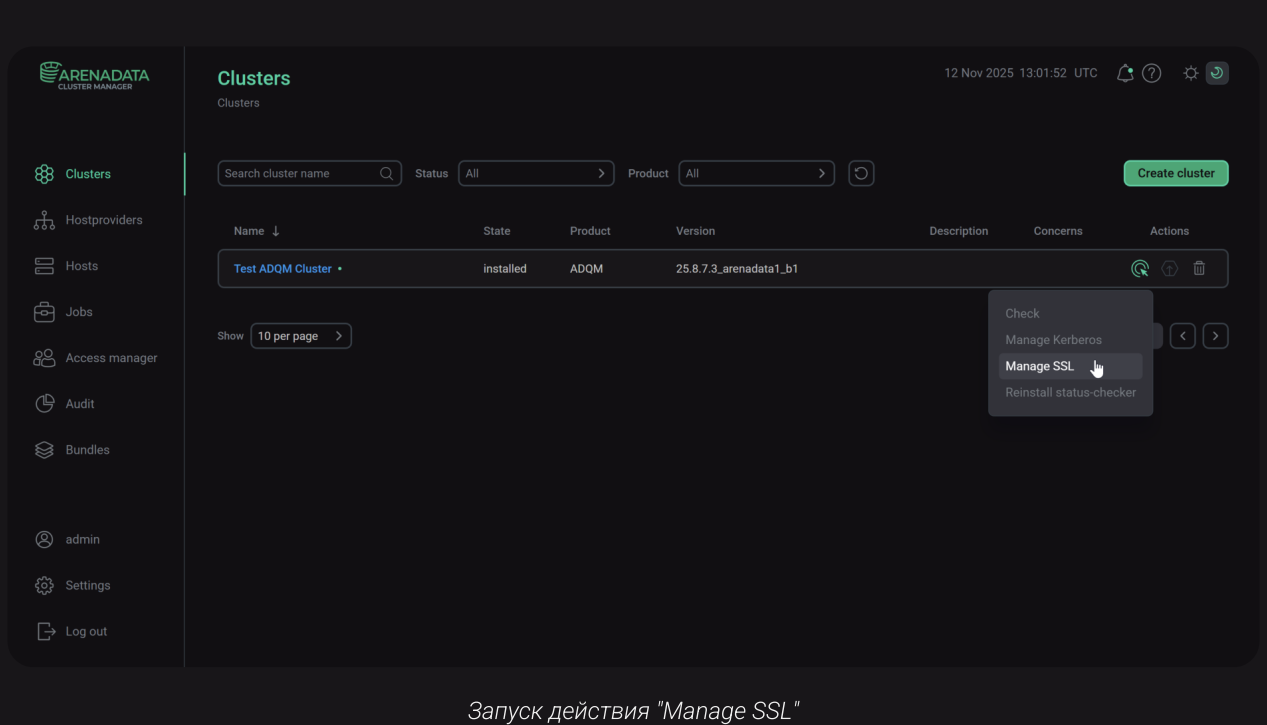
Конфигурирование ADQM для включения SSL

Включить SSL-шифрование соединений в ADQM можно через интерфейс ADCM, который предоставляет для этого следующие функциональные возможности:

- Действие Manage SSL на уровне кластера — можно включить SSL сразу для нескольких сервисов ADQM, а также отключить порты для незащищенных соединений.
- Параметры SSL в конфигурациях сервисов ADQM — можно настроить SSL для каждого сервиса отдельно (например, изменить пути к файлам сертификатов и ключей по умолчанию, включить проверку клиентских сертификатов, указать корневой сертификат центра сертификации и др.).

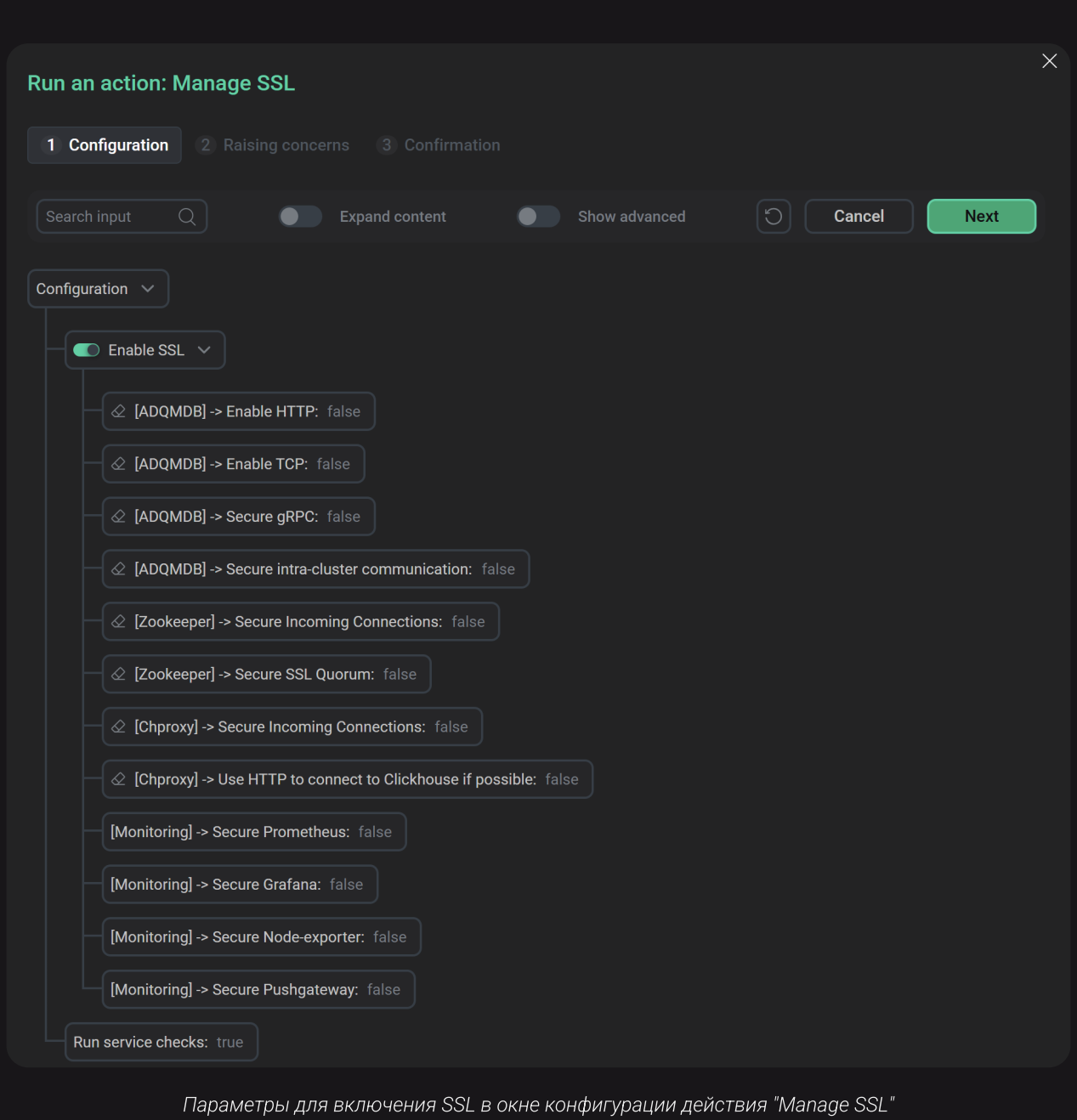
Действие Manage SSL

1. На странице ***Clusters*** в строке кластера ADQM нажмите иконку ⚙, чтобы открыть список действий для работы с кластером, и выберите ***Manage SSL***.



Запуск действия "Manage SSL"

Откроется диалоговое окно, где при активированном переключателе ***Enable SSL*** становятся доступны параметры, с помощью которых можно указать какие сервисы ADQM должны использовать безопасные соединения.


Run an action: Manage SSL

Configuration
- Enable SSL
  - [ADQMDB] -> Enable HTTP: false
  - [ADQMDB] -> Enable TCP: false
  - [ADQMDB] -> Secure gRPC: false
  - [ADQMDB] -> Secure intra-cluster communication: false
  - [Zookeeper] -> Secure Incoming Connections: false
  - [Zookeeper] -> Secure SSL Quorum: false
  - [Chproxy] -> Secure Incoming Connections: false
  - [Chproxy] -> Use HTTP to connect to Clickhouse if possible: false
  - [Monitoring] -> Secure Prometheus: false
  - [Monitoring] -> Secure Grafana: false
  - [Monitoring] -> Secure Node-exporter: false
  - [Monitoring] -> Secure Pushgateway: false
- Run service checks: true

Параметры для включения SSL в окне конфигурации действия "Manage SSL"

2. Заполните необходимые параметры в окне конфигурации действия ***Manage SSL***:

- Параметры в секции ***Enable SSL*** — чтобы включить/отключить SSL-шифрование входящих и внутренних соединений для различных сервисов ADQM.

Параметры включения SSL для сервисов

При выполнении действия ***Manage SSL*** указанные в его конфигурации значения параметров перезаписывают значения соответствующих параметров в конфигурациях сервисов — см. столбец ***Параметр сервиса ADQM*** в приведенной ниже таблице.

Параметр Manage SSL	Описание параметра	Параметр сервиса ADQM
[ADQMDB] → Enable HTTP	Позволяет отключить порт для соединения с ADQM по протоколу HTTP, чтобы использовался только HTTPS	Сервис ***ADQMDB***: параметр ***Network → Enable http***
[ADQMDB] → Enable TCP	Позволяет отключить TCP-порт для незащищенного соединения, чтобы использовался только TCP-порт для SSL-защищенного соединения (***SSL Configuration → TCP secure port***)	Сервис ***ADQMDB***: параметр ***Network → Enable tcp***
[ADQMDB] → Secure gRPC	Включает SSL для защищенного взаимодействия с клиентами через gRPC	Сервис ***ADQMDB***: параметр ***Enable gRPC → Enable SSL***
[ADQMDB] → Secure intra-cluster communication	Если параметр активирован, только безопасные протоколы (HTTPS, защищенный TCP) будут использоваться для обмена данными между серверами ClickHouse	Сервис ***ADQMDB***: параметр ***Intra-cluster communication security → Enable secure intra-cluster communication***
[Zookeeper] → Secure Incoming Connections	Включает использование безопасных входящих соединений в ZooKeeper	Сервис ***Zookeeper***: параметр ***SSL configuration → Enable SSL***
[Zookeeper] → Secure SSL Quorum	Активирует зашифрованную кворумную коммуникацию в ZooKeeper	Сервис ***Zookeeper***: параметр ***SSL configuration → sslQuorum***
[Chproxy] → Secure Incoming Connections	Включает использование безопасных входящих соединений в Chproxy	Сервис ***Chproxy***: параметр ***Network configuration → Enable HTTPS***
[Chproxy] → Use HTTP to connect to Clickhouse if possible	Указывает, какой протокол (HTTP или HTTPS) Chproxy будет использовать для подключения к ClickHouse	Сервис ***Chproxy***: параметр ***Network configuration → Use HTTP to connect to ADQMDB***
[Monitoring] → Secure Prometheus	Включает SSL-шифрование соединений с компонентом Prometheus Server сервиса Monitoring	Сервис ***Monitoring***: параметр ***[Prometheus] → Enable SSL***
[Monitoring] → Secure Grafana	Включает SSL-шифрование соединений с компонентом Grafana сервиса Monitoring	Сервис ***Monitoring***: параметр ***[Grafana] → Enable SSL***
[Monitoring] → Secure Node-exporter	Включает SSL-шифрование соединений с компонентом Node Exporter сервиса Monitoring	Сервис ***Monitoring***: параметр ***[Node-exporter] → Enable SSL***
[Monitoring] → Secure Pushgateway	Включает SSL-шифрование соединений с компонентом Pushgateway сервиса Monitoring	Сервис ***Monitoring***: параметр ***[Pushgateway] → Enable SSL***

- Параметр ***Run service checks*** — чтобы указать, нужно ли запустить проверку работоспособности и доступности кластера после применения настроек.

3. Нажмите ***Next***, а затем — ***Run*** в окне подтверждения запуска действия.

Во время выполнения действия ***Manage SSL*** ADCM последовательно останавливает все сервисы, перенастраивает их для включения или выключения SSL-шифрования, затем запускает сервисы и производит их проверку.

Результат действия и процесс его выполнения можно посмотреть на странице Jobs, где отображаются детали выполнения всех внутренних этапов действий, включая подробный stdout/stderr вывод Ansible.

Параметры SSL в сервисных конфигурациях

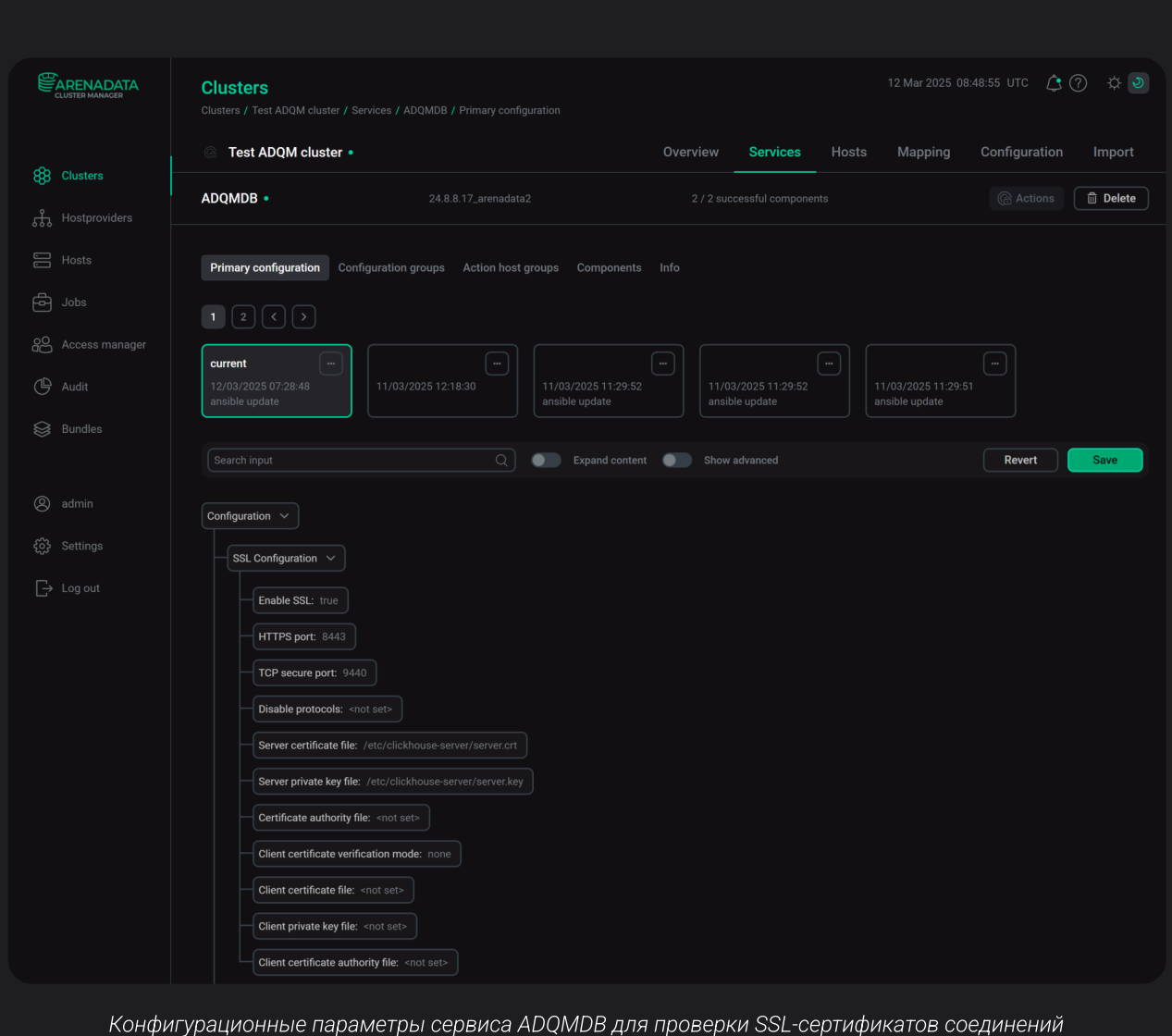
Альтернативный способ настроить защищенные соединения в ADQM — сконфигурировать параметры SSL для каждого сервиса отдельно.

1. Перейдите на вкладку ***Services*** кластера ADQM и выберите сервис, для которого нужно включить SSL-шифрование — ADQMDB, Zookeeper или Chproxy.
2. На вкладке ***Primary configuration*** настройте параметры SSL — название секции, где находятся эти параметры, зависит от сервиса.

ADQMDB: SSL Configuration

В разделе ***SSL Configuration*** конфигурации сервиса ADQMDB доступны параметры:

- ***Enable SSL*** — включает прослушивание портов для SSL-защищенных соединений по протоколам HTTPS и TCP.
- ***HTTPS port*** — порт для безопасного соединения по HTTPS (по умолчанию `8443`).
- ***TCP secure port*** — TCP-порт для безопасного взаимодействия с клиентами и межсерверной коммуникации по бинарному протоколу (по умолчанию `9440`).
- ***Disable protocols*** — протоколы, которые не разрешено использовать (например, `sslv2`, `sslv3`).
- ***Server certificate file*** — путь к файлу серверного SSL-сертификата в формате PEM.
- ***Server private key file*** — путь к файлу закрытого ключа серверного SSL-сертификата.
- ***Certificate authority file*** — путь к файлу корневого сертификата центра сертификации (Certification Authority, CA) для проверки клиентских сертификатов. Указывается, если необходимо включить TLS-аутентификацию.
- ***Client certificate verification mode*** — режим проверки пользовательских SSL-сертификатов, подписанных корневым сертификатом (должен быть указан в ***Certificate authority file***).
- ***Client certificate file***, ***Client private key file***, ***Client certificate authority file*** — параметры для настройки защищенного соединения между ClickHouse и ZooKeeper.

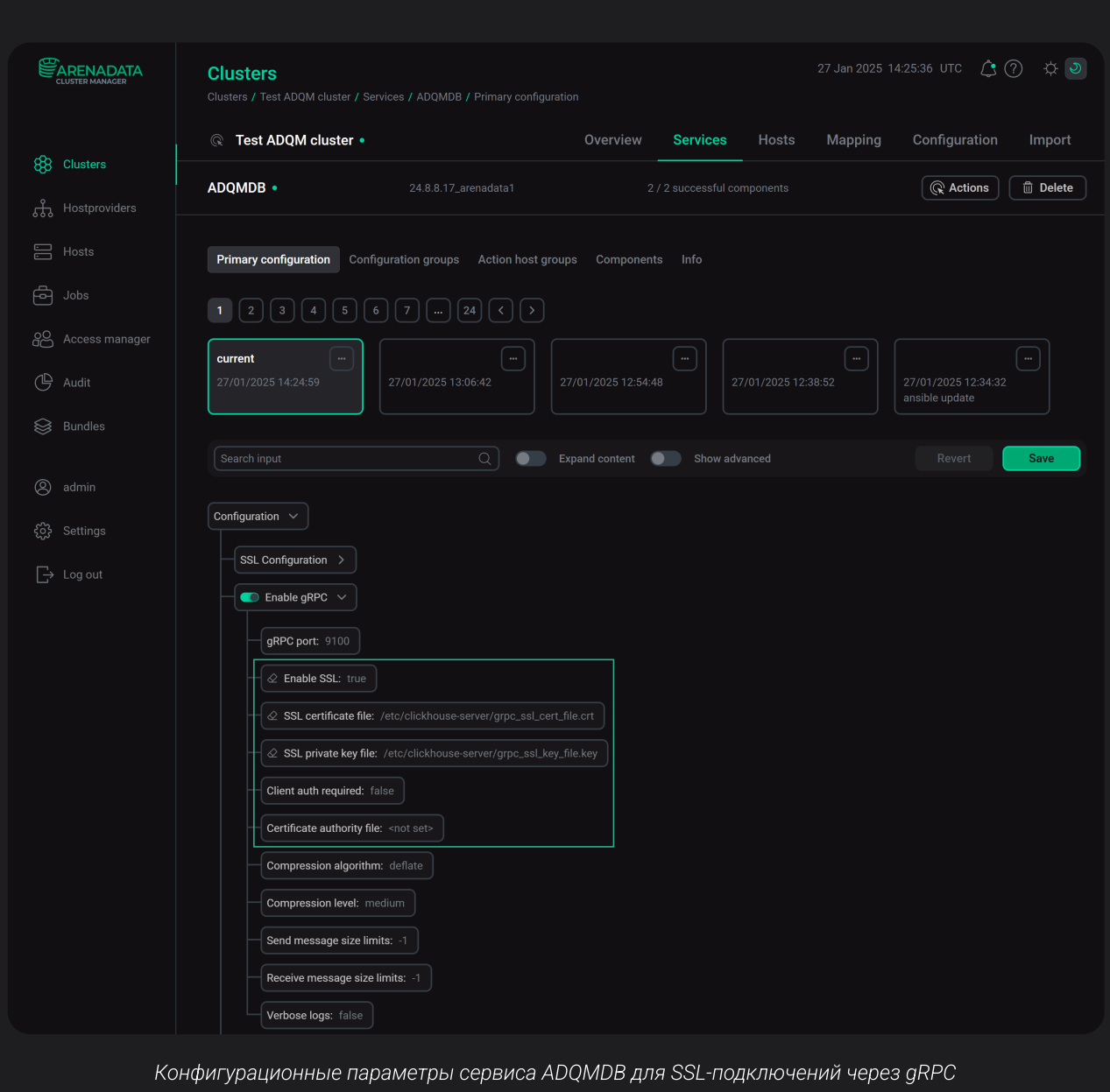


Конфигурационные параметры сервиса ADQMDB для проверки SSL-сертификатов соединений

ADQMDB: Enable gRPC

Если для сервиса ADQMDB включены опции ***Enable SSL*** (в разделе ***SSL Configuration***) и ***Enable gRPC***, в разделе ***Enable gRPC*** можно также настроить следующие параметры SSL для gRPC-подключений:

- ***Enable SSL*** — включает SSL для защищенного взаимодействия с клиентами через gRPC.
- ***SSL certificate file*** — путь к файлу SSL-сертификата в формате PEM для gRPC-подключений.
- ***SSL private key file*** — путь к файлу закрытого ключа SSL-сертификата для gRPC-подключений.
- ***Client auth required*** — указывает, будет ли сервер запрашивать сертификат клиента.
- ***Certificate authority file*** — путь к файлу корневого сертификата центра сертификации (Certification Authority, CA) в формате PEM, который будет использоваться для проверки клиентских сертификатов. Указывается при включенной опции ***Client auth required***.



Конфигурационные параметры сервиса ADQMDB для SSL-подключений через gRPC

ADQMDB: Network

В разделе ***Network*** конфигурации сервиса ADQMDB доступны параметры для отключения незащищенных портов:

- ***Enable http*** — позволяет отключить порт для соединений по протоколу HTTP (в пользу соединений через HTTPS-порт, который настраивается параметром ***SSL Configuration → HTTPS port***).
- ***Enable tcp*** — позволяет отключить порт для незащищенных соединений по протоколу ClickHouse TCP (в пользу безопасных соединений по TCP через порт ***SSL Configuration → TCP secure port***).

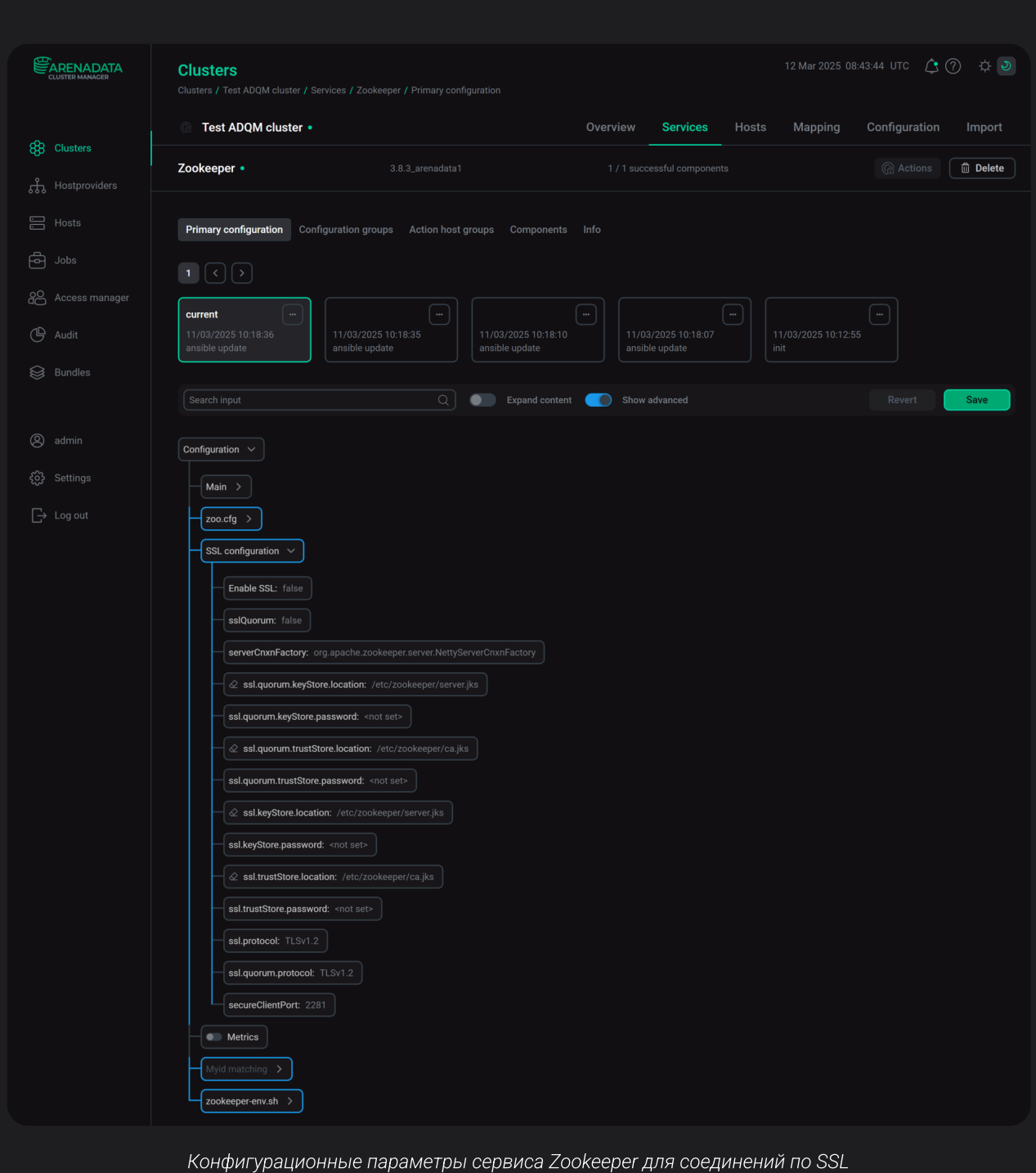
ADQMDB: Secure intra-cluster communication

В разделе ***Secure intra-cluster communication*** конфигурации сервиса ADQMDB доступен параметр ***Enable secure intra-cluster communication***, который позволяет включить использование только безопасных протоколов (HTTPS, защищенный TCP) для обмена данными между серверами ClickHouse. Чтобы активировать параметр, необходимо также включить ***SSL Configuration → Enable SSL***.

Zookeeper: SSL configuration

Раздел ***SSL configuration*** конфигурации сервиса Zookeeper (доступен, если включена опция ***Show advanced***) содержит параметры для настройки SSL-соединений между ClickHouse и ZooKeeper:

- ***Enable SSL*** — включает использование безопасных входящих соединений в ZooKeeper;
- ***sslQuorum*** — включает шифрование внутренних соединений в кворуме;
- ***serverCnxnFactory*** — имя класса, реализующего `ServerCnxnFactory` (чтобы использовать серверную коммуникацию на основе TLS, установите значение параметра `NettyServerCnxnFactory`);
- ***ssl.quorum.keyStore.location*** — путь к файлу хранилища ключей сервера;
- ***ssl.quorum.keyStore.password*** — пароль, используемый при создании хранилища ключей, указанного в ***ssl.quorum.keyStore.location***;
- ***ssl.quorum.trustStore.location*** — путь к файлу хранилища доверенных сертификатов сервера;
- ***ssl.quorum.trustStore.password*** — пароль, используемый при создании хранилища доверенных сертификатов, указанного в ***ssl.quorum.trustStore.location***;
- ***ssl.keyStore.location*** — путь к файлу хранилища ключей сервера, который будет использоваться для клиентских TLS-подключений;
- ***ssl.keyStore.password*** — пароль, используемый при создании хранилища ключей, указанного в ***ssl.keyStore.location***;
- ***ssl.trustStore.location*** — путь к файлу серверного хранилища доверенных сертификатов, который будет использоваться для клиентских TLS-подключений;
- ***ssl.trustStore.password*** — пароль, используемый при создании хранилища доверенных сертификатов, указанного в ***ssl.trustStore.location***;
- ***ssl.protocol*** — протокол TLS для общения с клиентом;
- ***ssl.quorum.protocol*** — протокол TLS для общения в кворуме;
- ***secureClientPort*** — порт для безопасных клиентских соединений.

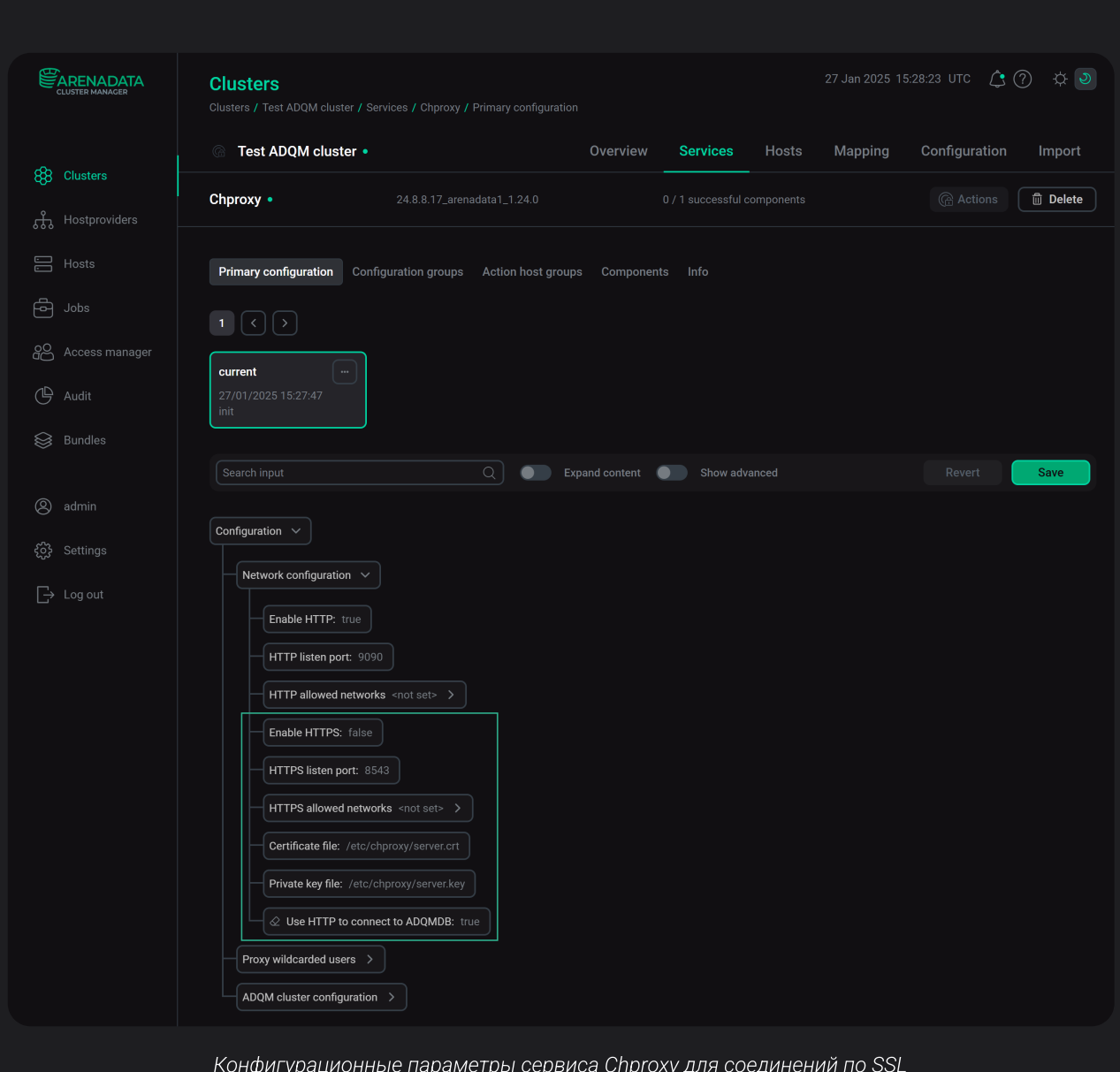


Конфигурационные параметры сервиса Zookeeper для соединений по SSL

Chproxy: Network configuration

В разделе ***Network configuration*** конфигурации сервиса Chproxy доступны следующие параметры для настройки безопасных соединений:

- ***Enable HTTPS*** — включает прослушивание запросов по HTTPS;
- ***HTTPS listen port*** — порт, на котором прослушиваются HTTPS-запросы;
- ***HTTPS allowed networks*** — список сетей, из которых разрешен доступ по HTTPS;
- ***Certificate file*** — путь к файлу серверного SSL-сертификата;
- ***Private key file*** — путь к файлу закрытого ключа серверного SSL-сертификата;
- ***Use HTTP to connect to ADQMDB*** — указывает, какой протокол (HTTP или HTTPS) Chproxy будет использовать для подключения к ClickHouse.

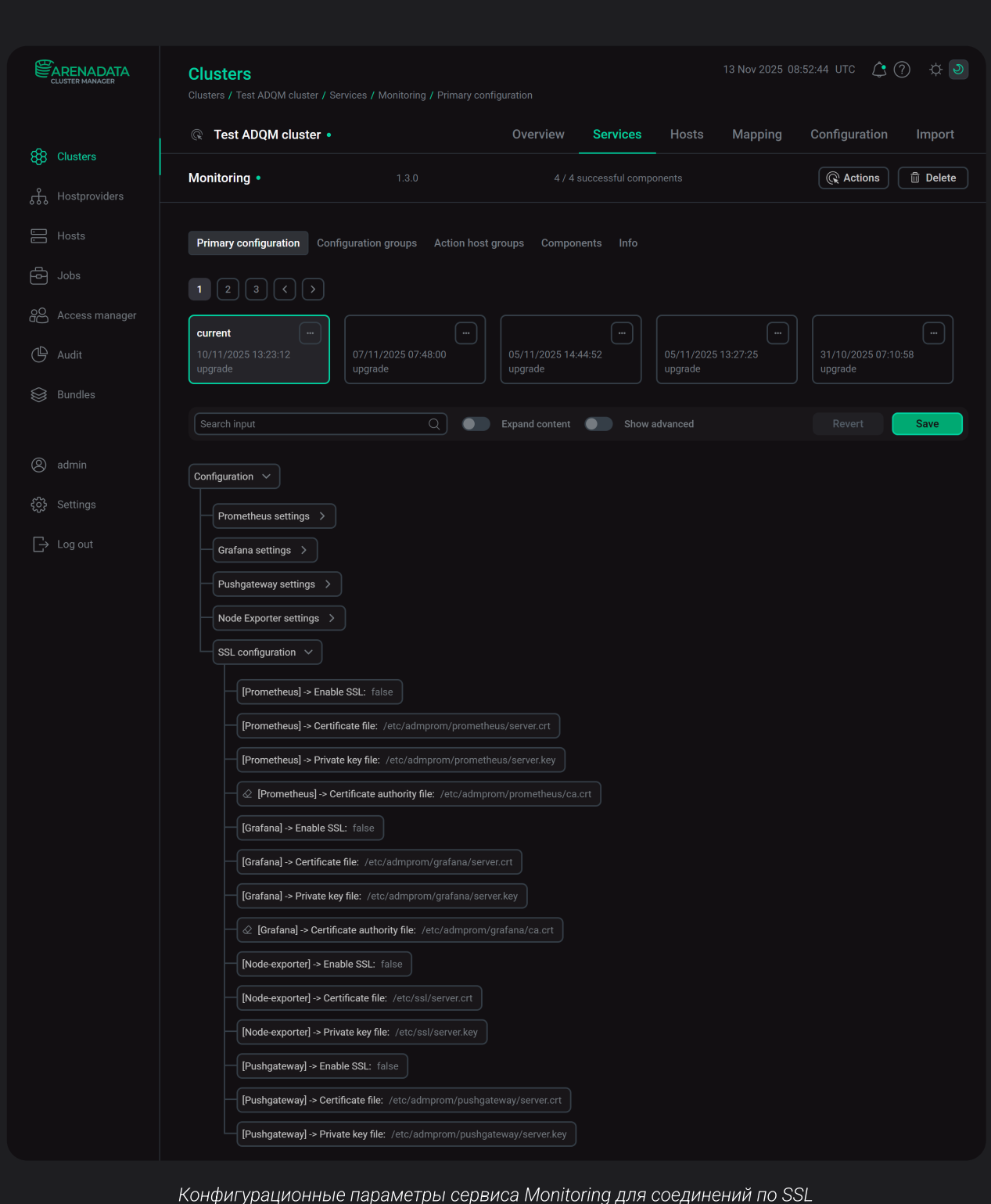


Конфигурационные параметры сервиса Chproxy для соединений по SSL

Monitoring: SSL configuration

В разделе ***SSL configuration*** конфигурации сервиса Monitoring доступны следующие параметры для настройки SSL-защищенных подключений к компонентам сервиса (файлы сертификатов/ключей должны храниться по указанным в параметрах путям на хостах кластера, где установлены соответствующие компоненты мониторинга):

- ***[Prometheus] → Enable SSL*** — включает SSL-шифрование входящих соединений для компонента мониторинга Prometheus Server;
- ***[Prometheus] → Certificate file*** — путь к файлу серверного SSL-сертификата в формате CRT (файл **.crt*) для компонента Prometheus Server;
- ***[Prometheus] → Private key file*** — путь к файлу закрытого ключа серверного SSL-сертификата (файл **.key*) для компонента Prometheus Server;
- ***[Prometheus] → Certificate authority file*** — путь к файлу корневого сертификата центра сертификации в формате CRT (файл *.crt) для проверки источников, предоставляющих данные в Prometheus (target);
- ***[Grafana] → Enable SSL*** — включает SSL-шифрование входящих соединений для компонента мониторинга Grafana;
- ***[Grafana] → Certificate file*** — путь к файлу серверного SSL-сертификата в формате CRT (файл **.crt*) для компонента Grafana;
- ***[Grafana] → Private key file*** — путь к файлу закрытого ключа серверного SSL-сертификата (файл **.key*) для компонента Grafana;
- ***[Grafana] → Certificate authority file*** — путь к файлу корневого сертификата центра сертификации в формате CRT (файл **.crt*) для проверки сертификата Prometheus;
- ***[Node-exporter] → Enable SSL*** — включает SSL-шифрование входящих соединений для компонента мониторинга Node Exporter;
- ***[Node-exporter] → Certificate file*** — путь к файлу серверного SSL-сертификата в формате CRT (файл **.crt*) для компонента Node Exporter;
- ***[Node-exporter] → Private key file*** — путь к файлу закрытого ключа серверного SSL-сертификата (файл **.key*) для компонента Node Exporter;
- ***[Pushgateway] → Enable SSL*** — включает SSL-шифрование входящих соединений для компонента Pushgateway;
- ***[Pushgateway] → Certificate file*** — путь к файлу серверного SSL-сертификата в формате CRT (файл **.crt*) для компонента Pushgateway;
- ***[Pushgateway] → Private key file*** — путь к файлу закрытого ключа серверного SSL-сертификата (файл **.key*) для компонента Pushgateway.

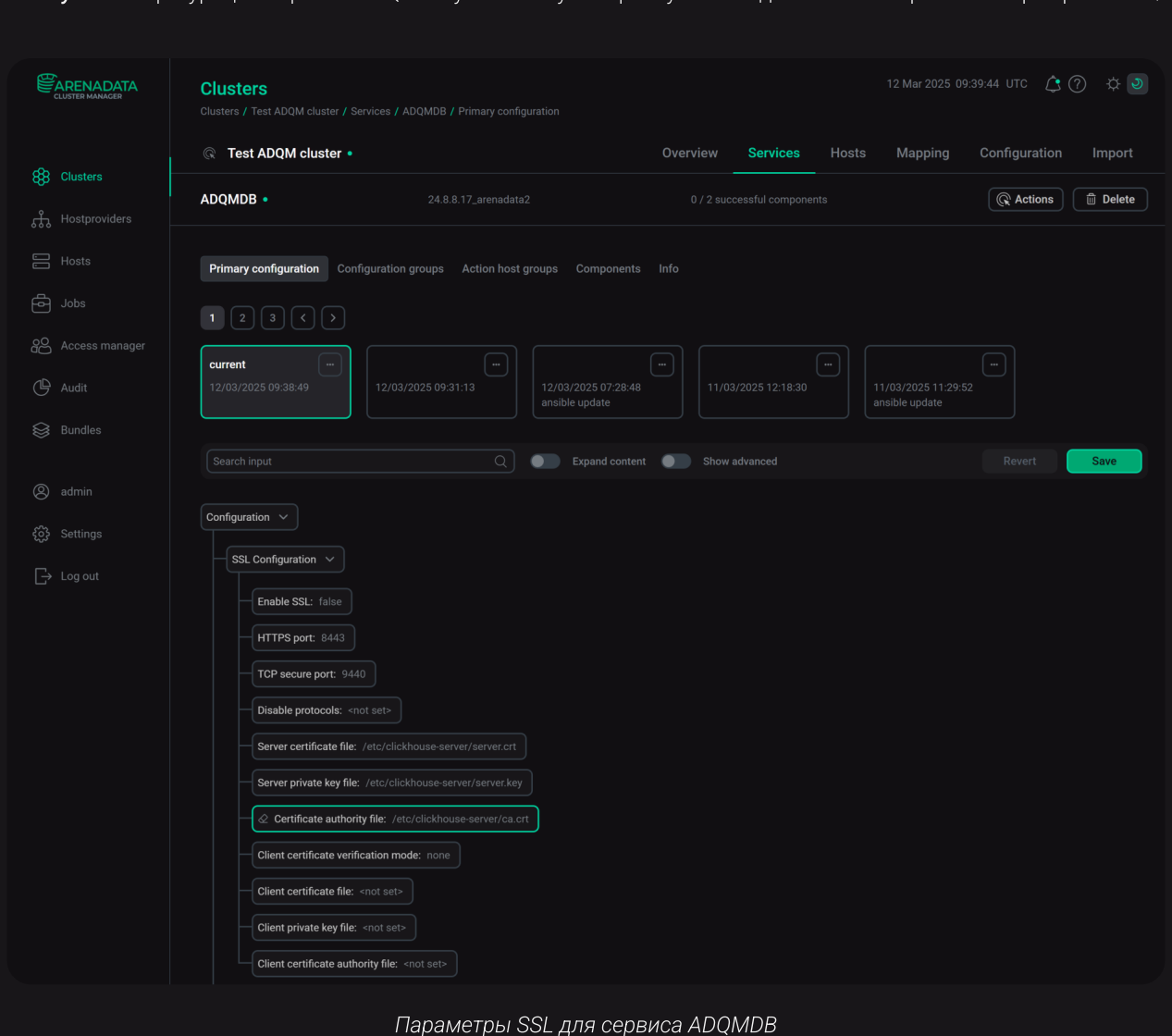


Конфигурационные параметры сервиса Monitoring для соединений по SSL

3. После указания всех необходимых параметров настройки SSL-соединений нажмите ***Save*** и выполните действие для применения изменений в конфигурации сервиса.

Пример

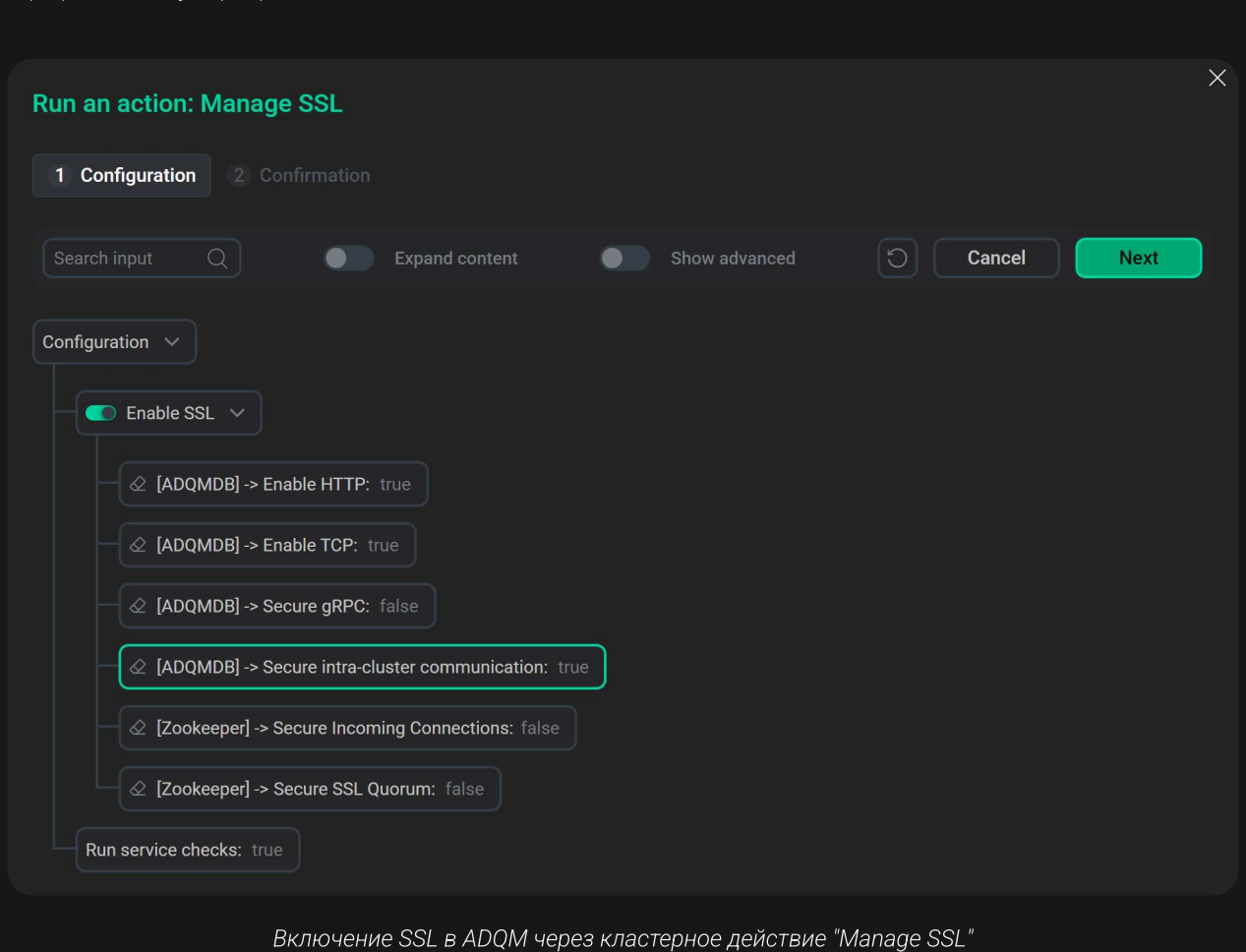
1. В параметре ***Certificate authority file*** конфигурации сервиса ADQMDB укажите путь к файлу самоподписанного корневого сертификата — */etc/clickhouse-server/ca.crt*.



Параметры SSL для сервиса ADQMDB

Нажмите ***Save***.

2. Запустите кластерное действие ***Manage SSL***. В окне с настройками активируйте ***Enable SSL***, а также включите опцию ***Secure intra-cluster communication*** для шифрования внутреннего трафика между серверами ClickHouse.


Run an action: Manage SSL

Configuration
- Enable SSL
  - [ADQMDB] -> Enable HTTP: true
  - [ADQMDB] -> Enable TCP: true
  - [ADQMDB] -> Secure gRPC: false
  - [ADQMDB] -> Secure intra-cluster communication: true
  - [Zookeeper] -> Secure Incoming Connections: false
  - [Zookeeper] -> Secure SSL Quorum: false
- Run service checks: true

Включение SSL в ADQM через кластерное действие "Manage SSL"

Нажмите ***Next*** и далее — ***Run***, чтобы начать выполнение действия.

Проверка соединения

Подключение через clickhouse-client

Чтобы убедиться, что соединение с ADQM защищено шифрованием SSL, можно подключиться к одному из серверов ADQM через clickhouse-client с флагом `--secure`, указав TCP-порт для безопасного соединения по SSL (`9440`). Например:

```
$ clickhouse-client --host adqm-host-1.arenadata.local --secure --port 9440 --user default
```

Вывод на экран после выполнения этой команды подтверждает подключение через защищенный порт `9440`:

```
ClickHouse client version 24.8.8.17.
Connecting to adqm-host-1.arenadata.local:9440 as user default.
Connected to ClickHouse server version 24.8.8.
```

Обратите внимание, опция ***Client certificate verification mode*** в секции ***SSL Configuration*** конфигурации сервиса ADQMDB должна быть выключена (значение `none`), так как в этом примере для подключения к ADQM не используется аутентификация пользователя по SSL-сертификату.

Проверка межсерверных подключений

Выполните распределенный запрос, например, который создает нового пользователя ADQM на каждом хосте:

```
CREATE USER test_user ON CLUSTER default_cluster IDENTIFIED WITH sha256_password BY 'qwerty';
```

В выводе результата выполнения запроса видно, что взаимодействие между хостами ADQM также осуществляется через порт `9440`:

```
   ┌─host────────────────────────┬─port─┬─status─┬─error─┬─num_hosts_remaining─┬─num_hosts_active─┐
1. │ adqm-host-4.arenadata.local │ 9440 │      0 │       │                   3 │                0 │
2. │ adqm-host-1.arenadata.local │ 9440 │      0 │       │                   2 │                0 │
3. │ adqm-host-2.arenadata.local │ 9440 │      0 │       │                   1 │                0 │
4. │ adqm-host-3.arenadata.local │ 9440 │      0 │       │                   0 │                0 │
   └─────────────────────────────┴──────┴────────┴───────┴─────────────────────┴──────────────────┘
```

Подключение по HTTPS

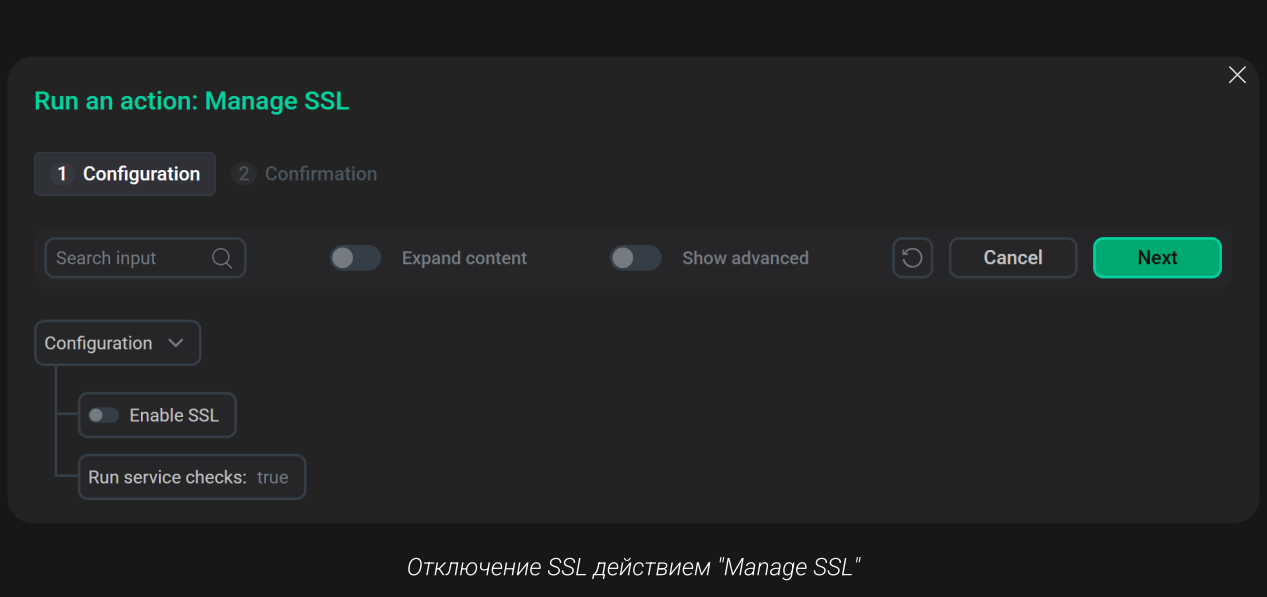
Для тестирования соединения с ADQM по протоколу HTTPS можно выполнить следующую команду на хосте кластера:

```
$ echo 'SHOW TABLES' | curl 'https://adqm-host-1.arenadata.local:8443/?query=' --data-binary @- --cacert arenadata_ca.crt
```

Если соединение успешно установлено, в результате выполнения запроса вернется список таблиц базы данных `default` на хосте *adqm-host-1.arenadata.local*.

Отключение SSL

Для отключения SSL запустите действие ***Manage SSL*** с деактивированной группой ***Enable SSL*** или выключите опцию ***Enable SSL*** в конфигурации нужного сервиса.


Run an action: Manage SSL

Configuration
- Enable SSL
- Run service checks: true

Отключение SSL действием "Manage SSL"

Кластерные действия

Елена Дворядкина

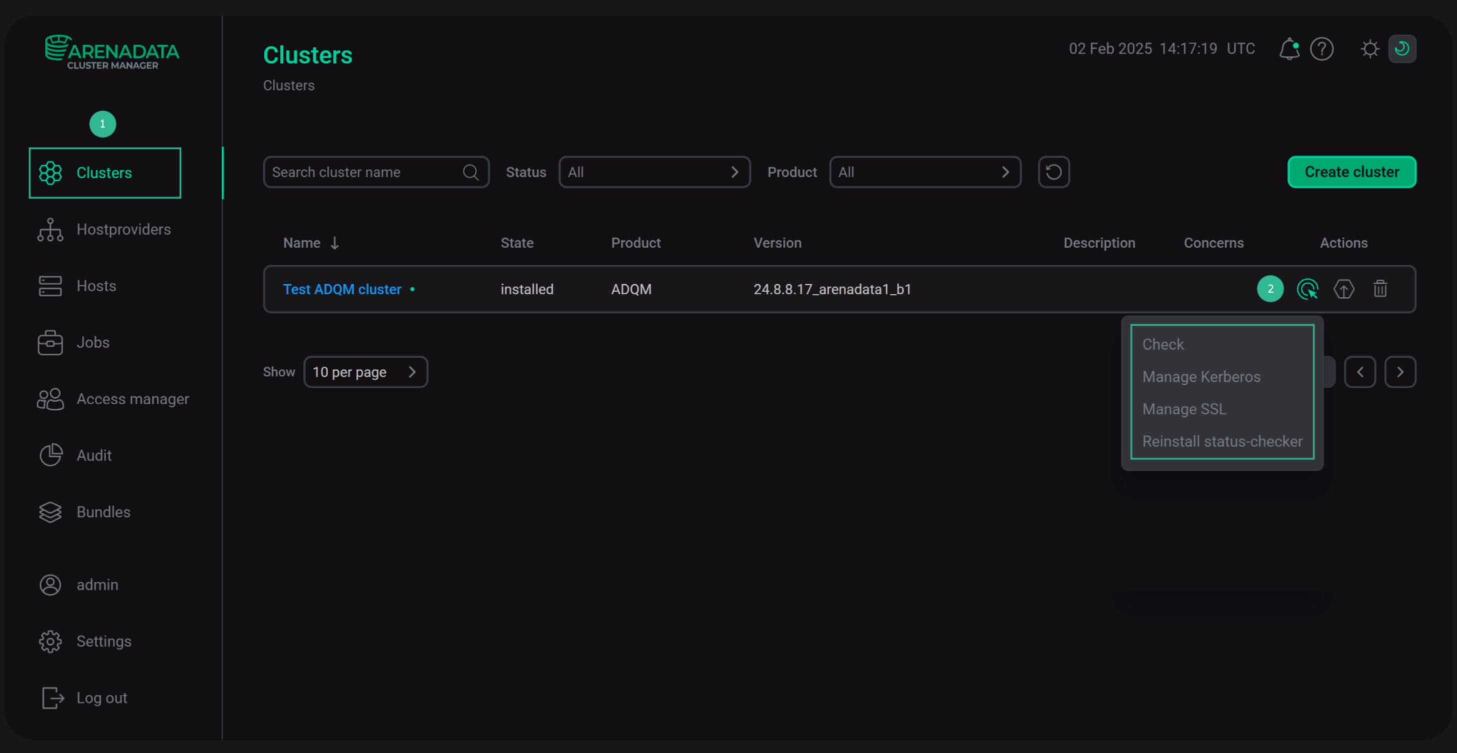
Содержание

Назад к содержанию

Управление кластером

Обновление кластера

Управление кластером ADQM осуществляется на странице ***Clusters*** в пользовательском интерфейсе ADCM.



ARENADATA
CLUSTER MANAGER
1
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
02 Feb 2025 14:17:19 UTC
Search cluster name
Status
All
Product
All
Create cluster
2
Check
Manage Kerberos
Manage SSL
Reinstall status-checker
Show
10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	installed	ADQM	24.8.8.17_arenadata1_b1			

Кластер ADQM в web-интерфейсе ADCM

Описание страницы ***Clusters*** и общих действий на этой странице приведено в документации ADCM.

Управление кластером

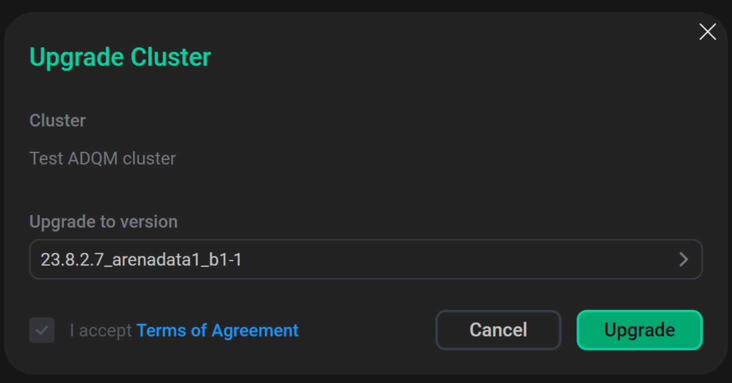
Нажмите иконку , чтобы открыть список доступных действий, и выберите действие для выполнения. Набор доступных действий зависит от состояния кластера:

- Для нового кластера (состояние кластера `created`) доступно действие:
  - ***Install*** — устанавливает все сервисы кластера.
- После установки кластера (состояние кластера `installed`) доступны действия:
  - ***Check*** — проверяет, работают ли установленные компоненты кластера правильно.
  - ***Manage Kerberos*** — позволяет включать, реконфигурировать и отключать аутентификацию по протоколу Kerberos.
  - ***Manage SSL*** — позволяет настраивать и включать/отключать SSL-шифрование для сервисов кластера ADQM.
  - ***Reinstall status-checker*** — перенастраивает и перезапускает службу проверки состояния (statuschecker) для всех сервисов кластера. Используется в случае миграции кластера под управление нового сервера ADCM.
- Для подготовленного к обновлению версии кластера (состояние кластера `upgradable`) доступно действие:
  - ***Upgrade*** — выполняет обновление версии кластера.

Обновление кластера

Чтобы обновить версию кластера ADQM, сначала необходимо загрузить новый бандл — см. шаг ***Загрузка бандла в ADCM*** в статье Создание кластера.

Когда новый бандл загружен, откройте страницу ***Clusters***. Иконка для кластера ADQM в столбце ***Actions*** станет активной. Кликните иконку — в открывшемся окне ***Upgrade Cluster*** выберите версию, доступную для обновления, и нажмите ***Upgrade***.



Upgrade Cluster
Cancel
Upgrade

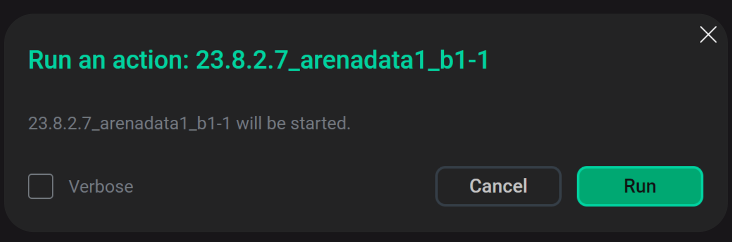
Cluster
Test ADQM cluster

Upgrade to version
23.8.2.7_arenadata1_b1-1

☑ I accept Terms of Agreement

Запуск обновления

После этого ADCM покажет диалоговое окно подтверждения подготовки к обновлению. Кликните ***Run***.



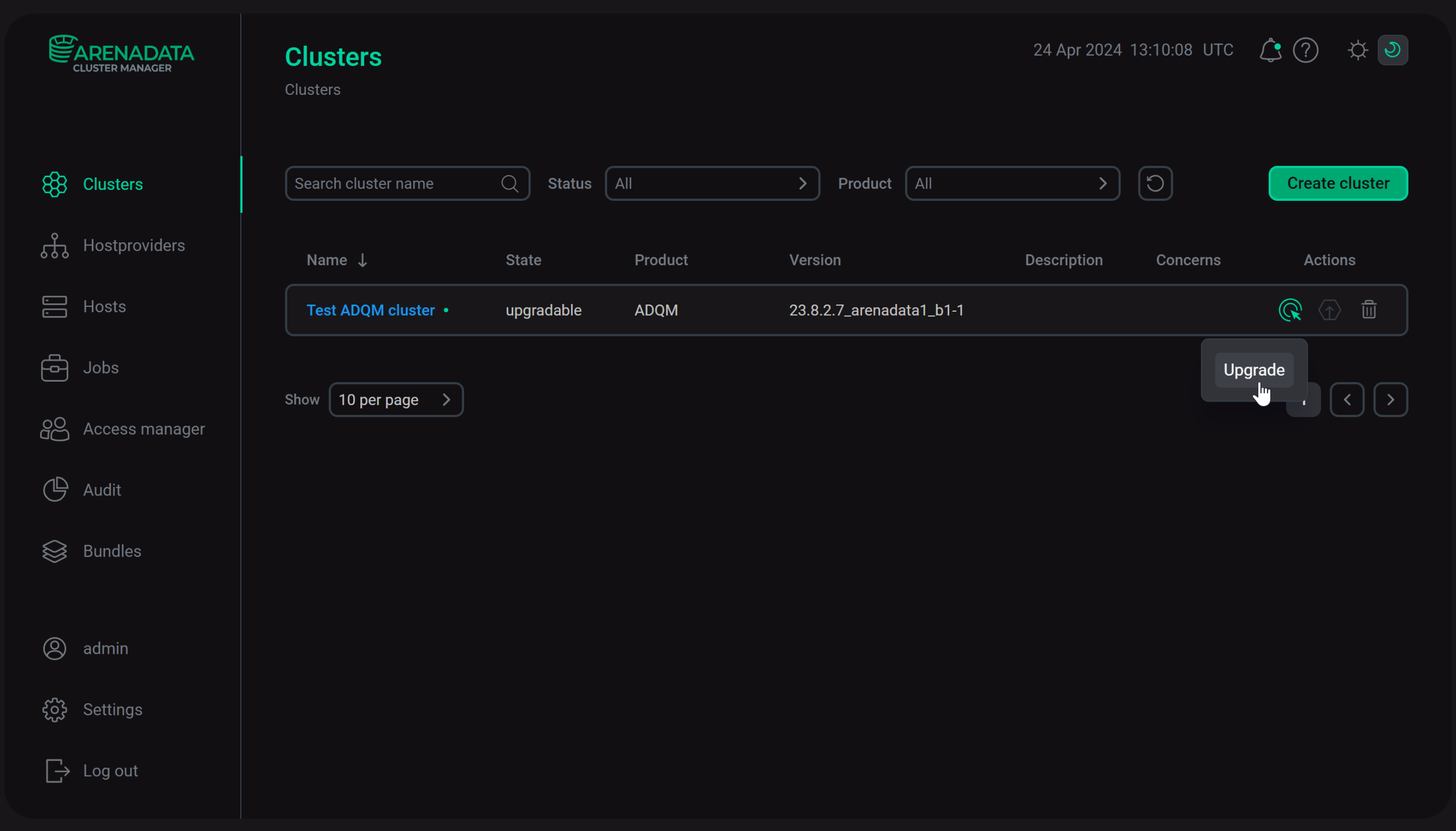
Run an action: 23.8.2.7_arenadata1_b1-1

23.8.2.7_arenadata1_b1-1 will be started.

☐ Verbose

Подтверждение подготовки к обновлению

В результате ADCM изменит статус кластера на `upgradable` — кластер готов к обновлению. Кликните иконку в столбце ***Actions*** и запустите действие ***Upgrade***.



ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
24 Apr 2024 13:10:08 UTC
Search cluster name
Status
All
Product
All
Create cluster
Upgrade
Show
10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	upgradable	ADQM	23.8.2.7_arenadata1_b1-1			

Выполнение обновления

В открывшемся диалоговом окне нажмите ***Run***, чтобы подтвердить выполнение обновления. Как только процесс обновления будет завершен, ADCM изменит состояние кластера на `installed`.

ПРИМЕЧАНИЕ

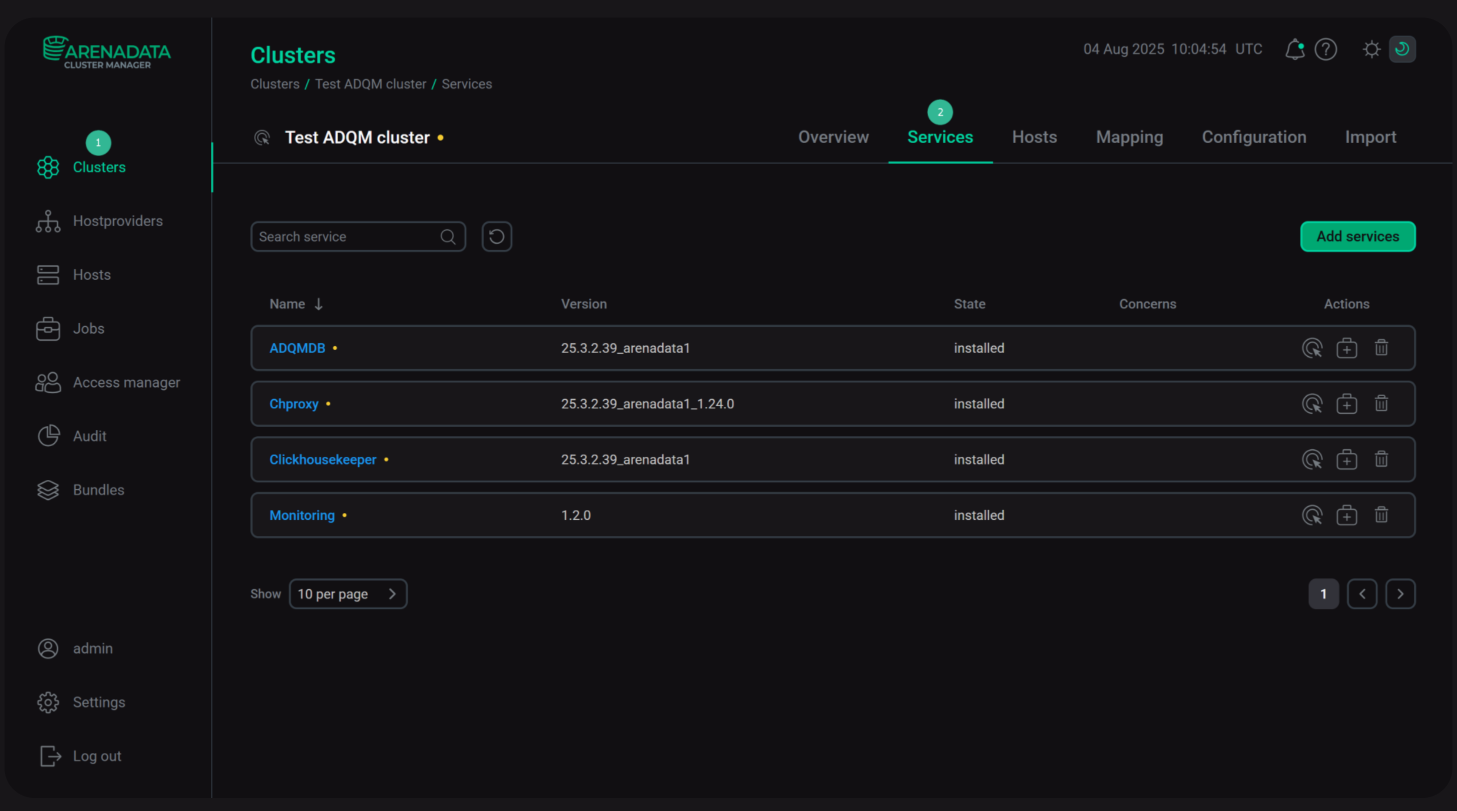
В ADCM начиная с версии 2.3 при обновлении ADQM с версии 23.3.2.37.1.b1 (минимально) на версию 24.3.2.23.2.b1 и выше подготовленный к обновлению кластер имеет статус `upgrading`, в котором доступно два действия — ***Complete the upgrade*** и ***Abort the upgrade***. Такой кластер можно обновить без его полной остановки, последовательно обновляя сервис ADQMDB на группах хостов и проверяя корректность работы новой версии сервиса на промежуточных этапах обновления кластера. В случае обнаружения проблем с новой версией ClickHouse есть возможность вернуться к предыдущей версии (до завершения обновления кластера). Детали использования данной функциональности описаны в статье Rolling upgrade.

Сервисные действия

Елена Дворядкина

Назад к содержанию

Пользовательский интерфейс ADCM позволяет выполнять действия для управления сервисами кластера ADQM. Действия доступны на вкладке ***Services***. Чтобы перейти на эту вкладку, кликните по имени кластера на странице ***Clusters*** и откройте вкладку ***Services***.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADQM cluster / Services
04 Aug 2025 10:04:54 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Show 10 per page
1
2

Name	Version	State	Concerns	Actions
ADQMDB	25.3.2.39_arenadata1	installed		
Chproxy	25.3.2.39_arenadata1_1.24.0	installed		
Clickhousekeeper	25.3.2.39_arenadata1	installed		
Monitoring	1.2.0	installed		

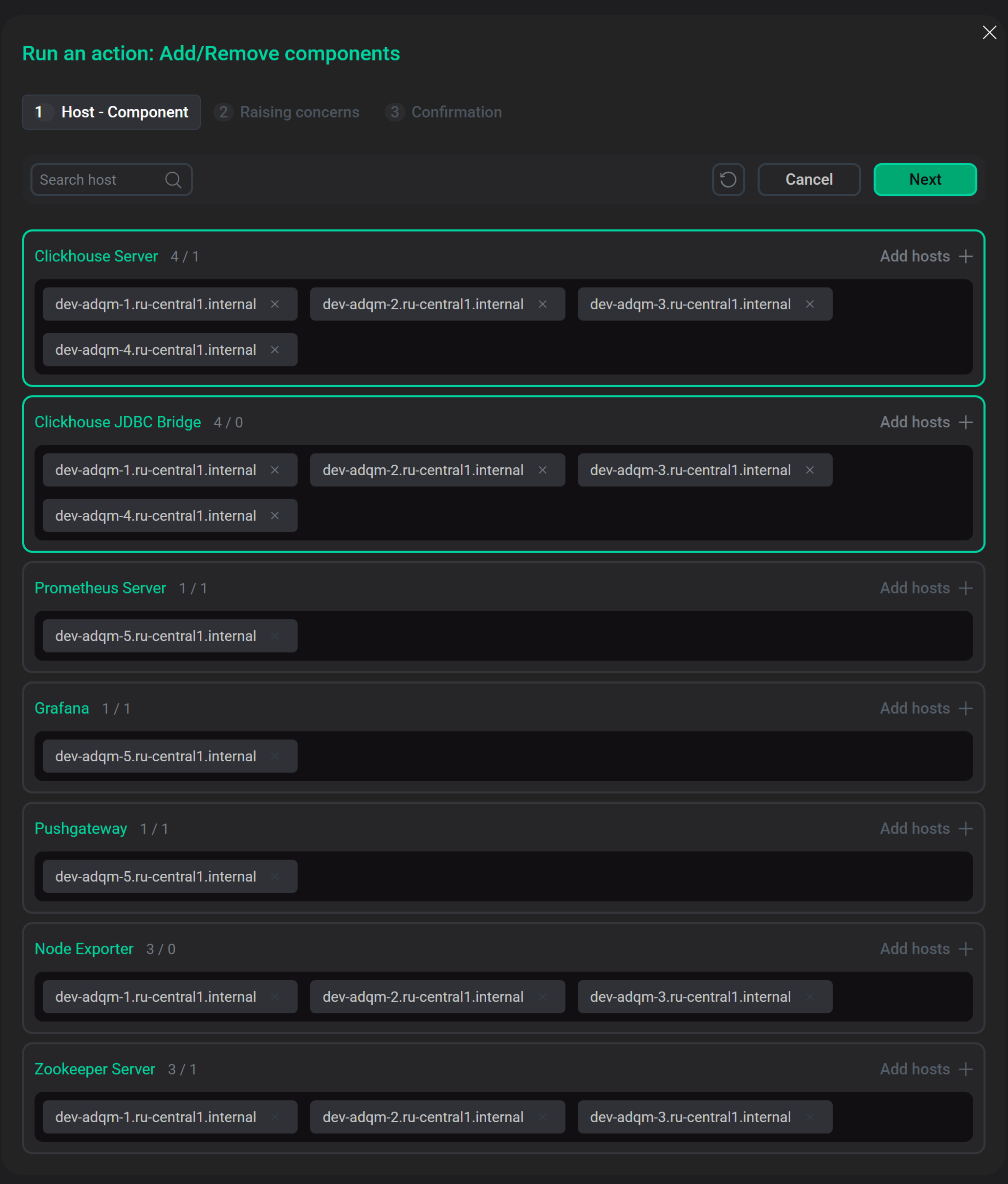
Сервисы кластера ADQM в интерфейсе ADCM

Описание вкладки ***Services*** и общих действий на этой вкладке приведено в документации ADCM. Набор действий, доступных для работы с сервисом кластера ADQM, зависит от типа сервиса.

Действия, доступные для всех сервисов ADQM:

- ***Install*** (если сервис в состоянии `created`) — устанавливает сервис.
- ***Add/Remove components*** — открывает интерфейс сопоставления компонентов и хостов, в котором можно добавить, удалить или перераспределить компоненты сервиса.

ADQMDB


Run an action: Add/Remove components
1 Host - Component
2 Raising concerns
3 Confirmation
Search host
Cancel
Next
Add hosts

Clickhouse Server 4 / 1: dev-adqm-1.ru-central1.internal, dev-adqm-2.ru-central1.internal, dev-adqm-3.ru-central1.internal, dev-adqm-4.ru-central1.internal

Clickhouse JDBC Bridge 4 / 0: dev-adqm-1.ru-central1.internal, dev-adqm-2.ru-central1.internal, dev-adqm-3.ru-central1.internal, dev-adqm-4.ru-central1.internal

Prometheus Server 1 / 1: dev-adqm-5.ru-central1.internal

Grafana 1 / 1: dev-adqm-5.ru-central1.internal

Pushgateway 1 / 1: dev-adqm-5.ru-central1.internal

Node Exporter 3 / 0: dev-adqm-1.ru-central1.internal, dev-adqm-2.ru-central1.internal, dev-adqm-3.ru-central1.internal

Zookeeper Server 3 / 1: dev-adqm-1.ru-central1.internal, dev-adqm-2.ru-central1.internal, dev-adqm-3.ru-central1.internal

Интерфейс сопоставления компонентов и хостов для сервиса ADQMDB

- ***Check*** — проверяет соответствие настроек хостов, компонентов и сопутствующих сервисов требованиям сервиса, для которого оно вызвано, а также правильность работы самого сервиса.
- ***Start*** — запускает сервис.
- ***Stop*** — останавливает сервис.

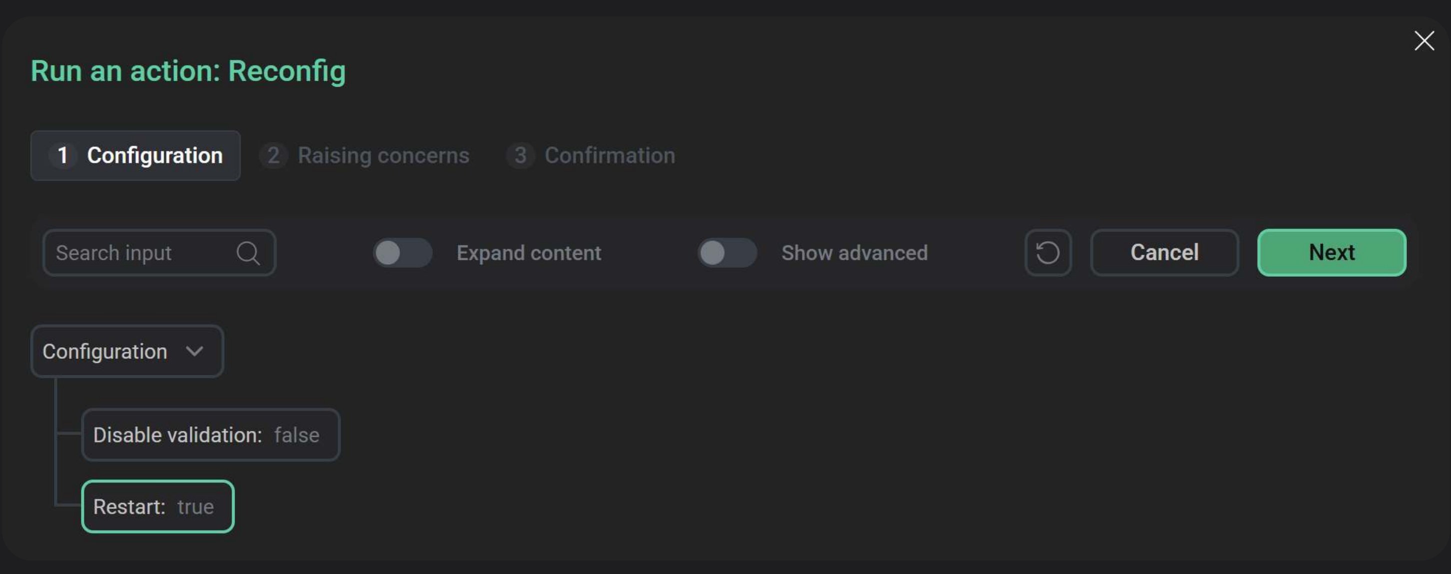
Действия, специфичные для отдельных сервисов:

- ***Reconfig*** (ADQMDB, Clickhousekeeper, Chproxy) — обновляет конфигурацию сервиса в соответствии с параметрами, указанными на странице конфигурации сервиса.

ADQMDB

В окне ***Run an action***, которое открывается при запуске действия ***Reconfig*** для сервиса ADQMDB, на вкладке ***Configuration*** можно активировать опции:

- ***Disable validation*** — отключить валидацию конфигурации (в некоторых случаях конфигурация может считаться некорректной — например, если были удалены используемые политики хранения);
- ***Restart*** — перезапустить сервис (для применения изменений некоторых параметров перезапуск сервиса не требуется — в этом случае достаточно перезагрузить конфигурацию).


Run an action: Reconfig
1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Disable validation: false

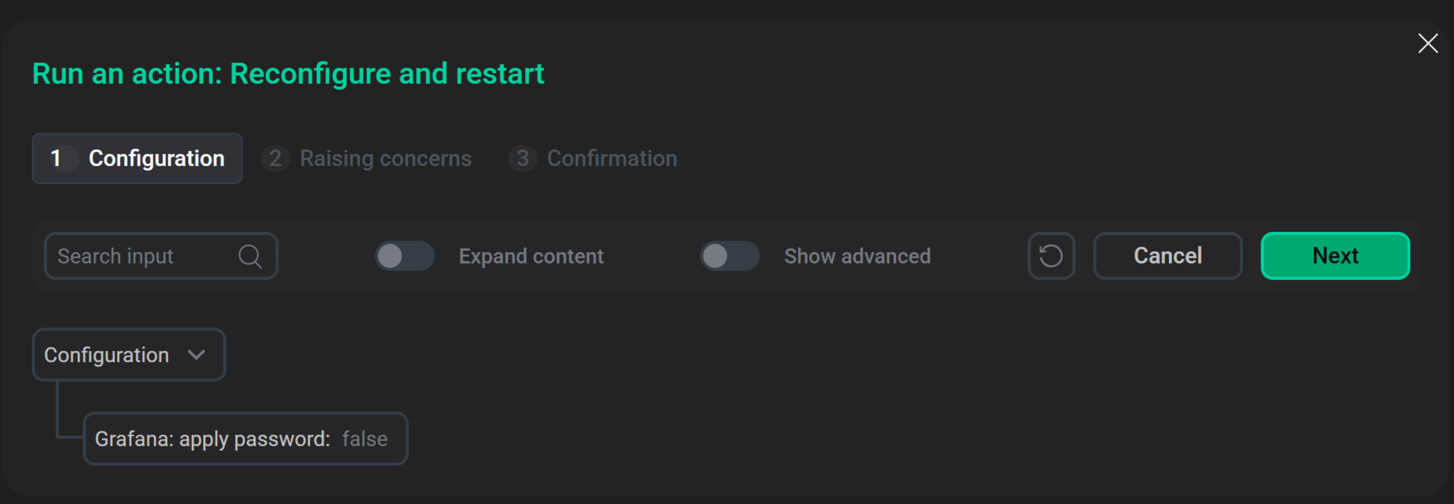
Restart: true

Настройка действия "Reconfig" сервиса ADQMDB

- ***Restart*** (ADQMDB, Clickhousekeeper, Chproxy) — останавливает, затем запускает сервис.
- ***Reconfig and restart*** (Zookeeper, Monitoring) — обновляет конфигурацию сервиса в соответствии с параметрами, указанными на странице конфигурации сервиса, и перезапускает сервис.

Monitoring

В окне ***Run an action***, которое открывается при запуске действия для сервиса Monitoring, с помощью параметра ***Grafana: apply password*** можно указать следует ли переустановить пароль администратора Grafana, используя значение параметра ***Grafana settings*** → ***Grafana administrator's password***, расположенного на вкладке ***Primary configuration*** страницы конфигурации сервиса Monitoring.


Run an action: Reconfigure and restart
1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Grafana: apply password: false

Настройка действия "Reconfigure and restart" сервиса Monitoring

- ***Manage auto core dump*** (ADQMDB) — включает/отключает автоматический дамп памяти.
- ***Reinstall*** (Monitoring) — переустанавливает сервис.

Конфигурирование логических кластеров в интерфейсе ADCM

Елена Дворядкина

Содержание

Назад к содержанию

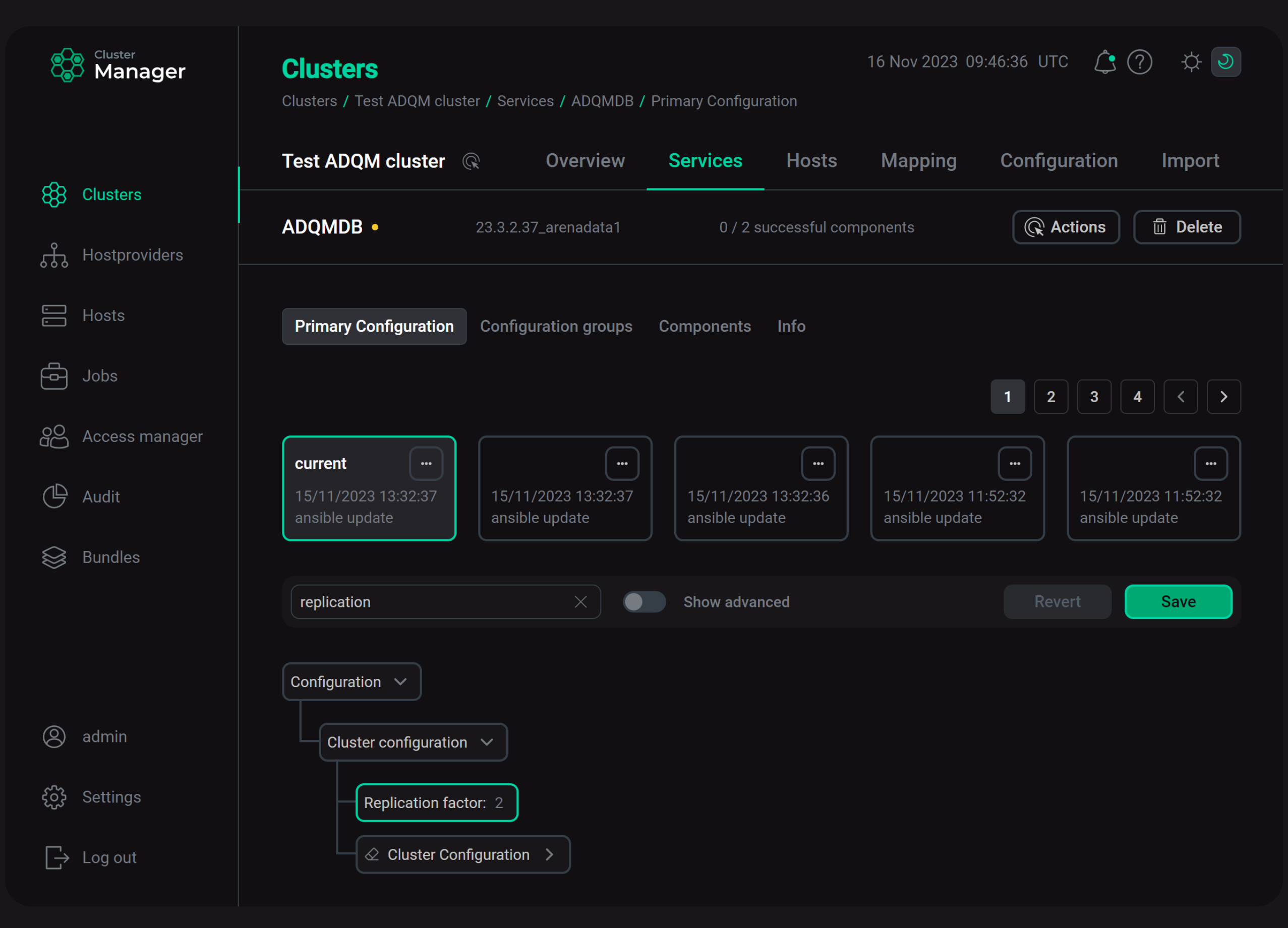
Автоматическое создание логического кластера при установке ADQM

Конфигурирование дополнительных логических кластеров

Чтобы организовать распределенное хранение и репликацию данных, а также выполнять распределенные запросы (`ON CLUSTER`) на нескольких хостах ADQM, необходимо объединить хосты в логический кластер. Сконфигурировать логический кластер в рамках кластера ADQM можно через интерфейс ADCM — с помощью параметров в секции ***Cluster configuration*** на странице конфигурации сервиса ADQMDB.

Автоматическое создание логического кластера при установке ADQM

На этапе предварительной настройки ADQM (перед запуском установки кластера), укажите параметр ***Replication factor***, чтобы автоматически сгруппировать хосты в логический кластер.



Cluster Manager
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary Configuration
16 Nov 2023 09:46:36 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
23.3.2.37_arenadata1
0 / 2 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1
2
3
4
current
15/11/2023 13:32:37
ansible update
15/11/2023 13:32:37
ansible update
15/11/2023 13:32:36
ansible update
15/11/2023 11:52:32
ansible update
15/11/2023 11:52:32
ansible update
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
replication
Show advanced
Revert
Save
Configuration
Cluster configuration
Cluster Configuration

Replication factor: 2

Установка фактора репликации

Топология кластера сформируется по следующей схеме:

- Все хосты ADQM разбиваются на группы — шарды.
- Каждый хост в шарде — это реплика. Количество реплик в шарде определяется значением параметра ***Replication factor***.
- Если общее количество хостов не делится ровно на указанное значение фактора репликации, то количество хостов-реплик в последнем шарде будет определяться как остаток от деления.

После установки кластера ADQM в секцию `remote_servers` конфигурационного файла *config.xml* будет добавлен логический кластер `default_cluster` — эта топология создается один раз и не изменяется при дальнейшей реконфигурации кластера.

Например, если в кластере ADQM 4 хоста (*host-1*, *host-2*, *host-3*, *host-4*) и параметр ***Replication factor*** установлен в `2`, на основе указанного фактора репликации будет сгенерирован логический кластер, включающий два шарда, в каждом из которых по два хоста-реплики:

```
<remote_servers>
    <default_cluster>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host-2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-3</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host-4</host>
                <port>9000</port>
            </replica>
        </shard>
    </default_cluster>
</remote_servers>
```

Информация о кластере `default_cluster` будет добавлена в системную таблицу system.clusters:

```
SELECT cluster, shard_num, replica_num, host_name FROM system.clusters;
```

```
┌─cluster─────────┬─shard_num─┬─replica_num─┬─host_name─┐
│ default_cluster │         1 │           1 │ host-1    │
│ default_cluster │         1 │           2 │ host-2    │
│ default_cluster │         2 │           1 │ host-3    │
│ default_cluster │         2 │           2 │ host-4    │
└─────────────────┴───────────┴─────────────┴───────────┘
```

При создании логического кластера в конфигурационный файл *config.xml* на каждом хосте также добавляется секция `macros` с идентификаторами шарда и реплики для хоста. Например, макросы для *host-4*:

```
<macros>
    <replica>2</replica>
    <shard>2</shard>
</macros>
```

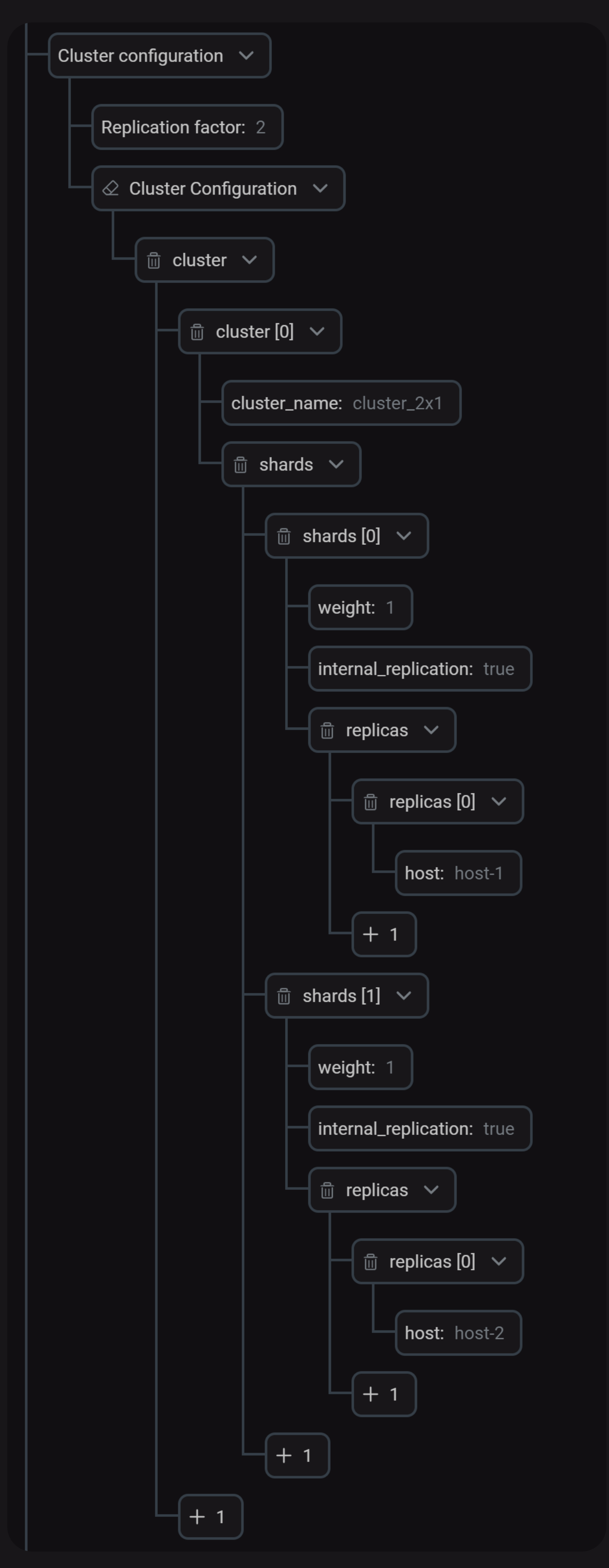
Макросы используются для автоматической подстановки идентификаторов шарда и реплики, соответствующих хосту, при создании реплицируемых таблиц на кластере (см. пример в статье Типовой кластер).

Конфигурирование дополнительных логических кластеров

Хосты ADQM можно комбинировать в несколько логических кластеров различных топологий (один хост может участвовать в разных логических кластерах).

Чтобы сконфигурировать логический кластер, используйте параметр ***Cluster Configuration***, добавляя необходимые элементы с помощью иконки `+ 1`:

1. Добавьте новый кластер и укажите его имя.
2. Определите, сколько шардов должно быть в кластере — добавьте элементы в список ***shards***.
   Для каждого шарда укажите значения параметров ***weight*** и ***internal_replication*** (подробнее об этих параметрах в статье concept:architecture/sharding.adoc#configuration), а также добавьте хосты-реплики в список ***replicas***.



Cluster configuration
Cluster Configuration
cluster
cluster [0]
shards
shards [0]
replicas
replicas [0]
+ 1
shards [1]
replicas
replicas [0]
+ 1
+ 1
+ 1

Replication factor: 2
cluster_name: cluster_2x1
weight: 1
internal_replication: true
host: host-1
weight: 1
internal_replication: true
host: host-2

Конфигурирование логического кластера

3. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы сохранить настройки.
   Описание кластера появится в секции `remote_servers` файла *config.xml*:

```
<remote_servers>
    <!-- default_cluster configuration is here -->

    <cluster_2x1>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-1</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-2</host>
                <port>9000</port>
            </replica>
        </shard>
    </cluster_2x1>
</remote_servers>
```

   Информация о кластере также будет добавлена в таблицу system.clusters:

```
SELECT cluster, shard_num, replica_num, host_name FROM system.clusters;
```

```
┌─cluster─────────┬─shard_num─┬─replica_num─┬─host_name─┐
│ cluster_2x1     │         1 │           1 │ host-1    │
│ cluster_2x1     │         2 │           1 │ host-2    │
│ default_cluster │         1 │           1 │ host-1    │
│ default_cluster │         1 │           2 │ host-2    │
│ default_cluster │         2 │           1 │ host-3    │
│ default_cluster │         2 │           2 │ host-4    │
└─────────────────┴───────────┴─────────────┴───────────┘
```

Макросы для всех кластеров кроме `default_cluster` формируются с префиксом `<cluster_name>_`. Например, для приведенного выше кластера `cluster_2x1` в конфигурацию *config.xml* на хосте *host-1* добавятся следующие макросы:

```
<macros>
    <!-- default_cluster macros are here -->

    <cluster_2x1_replica>1</cluster_2x1_replica>
    <cluster_2x1_shard>1</cluster_2x1_shard>
</macros>
```

Rolling upgrade

Елена Дворядкина

Содержание

Назад к содержанию

Шаг 1. Подготовка к обновлению

Шаг 2. Обновление ADQMDB по группам хостов

Шаг 3. Завершение обновления

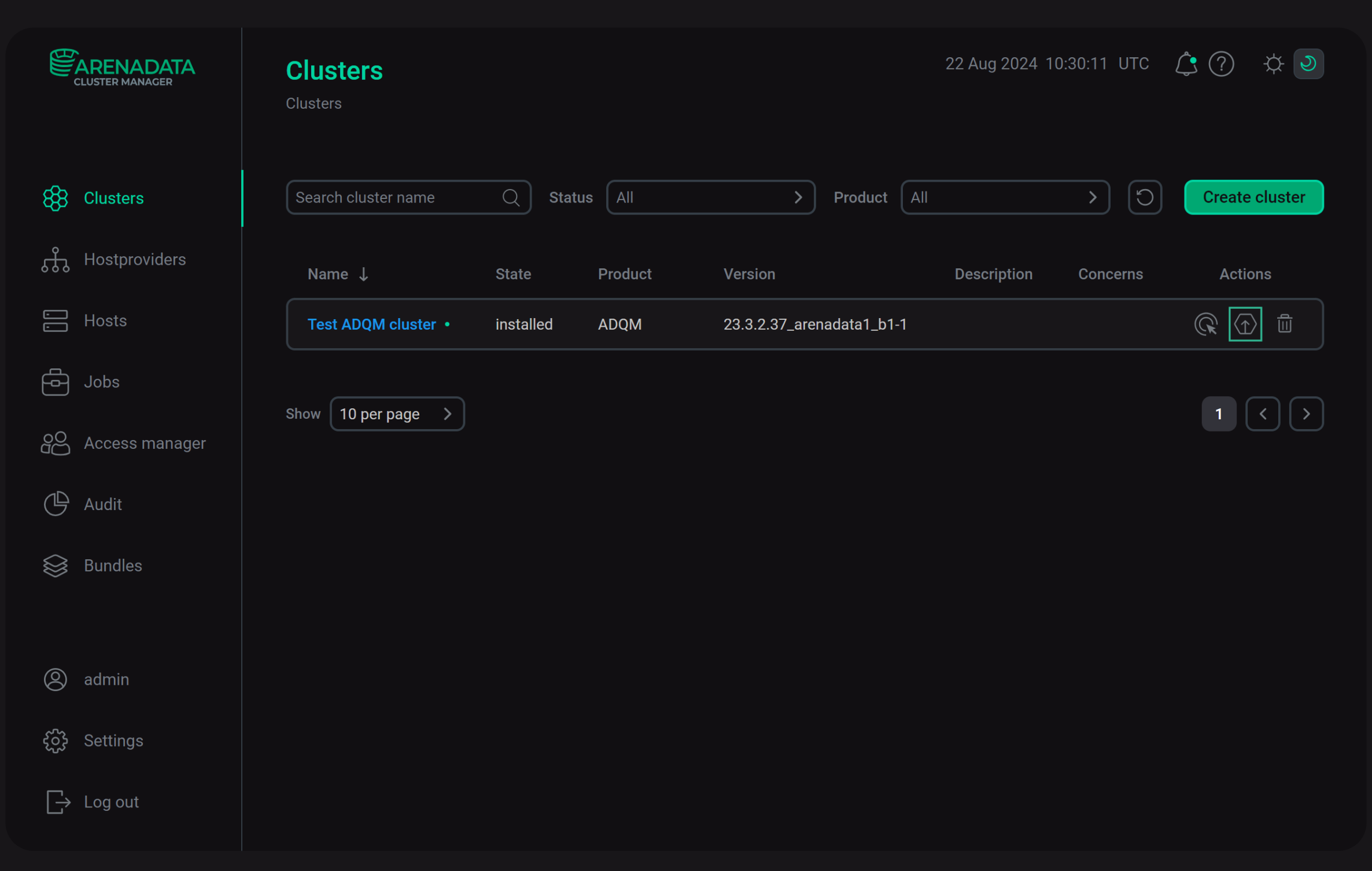
Rolling upgrade — возможность последовательного обновления сервиса ADQMDB на отдельных хостах ADQM без долговременной остановки кластера.

> **ВАЖНО**
>
> - Функциональность rolling upgrade доступна при обновлении ADQM минимальной версии 23.3.2.37.1.b1 на версию 24.3.2.23.2.b1 и выше в ADCM начиная с версии 2.3.
> - Rolling upgrade можно применить только к кластерам ADQM в статусе `installed`.

Ниже описан процесс последовательного обновления кластера ADQM через интерфейс ADCM.

Шаг 1. Подготовка к обновлению

1. На странице ***Bundles*** загрузите новый бандл ADQM — см. шаг ***Загрузка бандла в ADCM*** в статье Создание кластера.
2. Откройте страницу ***Clusters***, где после загрузки нового бандла станет активной иконка в столбце ***Actions*** для кластера ADQM — кликните по иконке.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Create cluster

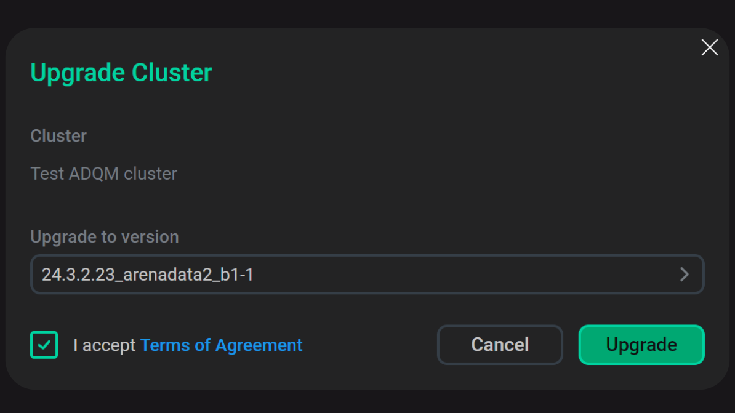
Clusters

22 Aug 2024 10:30:11 UTC

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	installed	ADQM	23.3.2.37_arenadata1_b1-1			

Подготовка кластера к обновлению

В открывшемся окне ***Upgrade Cluster*** выберите доступную для обновления версию и нажмите ***Upgrade***.



Upgrade Cluster

Cluster: Test ADQM cluster

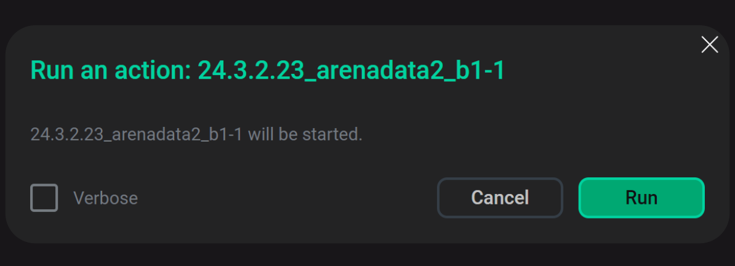
Upgrade to version: 24.3.2.23_arenadata2_b1-1

☑ I accept Terms of Agreement

Cancel
Upgrade

Выбор версии для обновления

После этого ADCM покажет диалоговое окно подтверждения подготовки к обновлению. Кликните ***Run***.



Run an action: 24.3.2.23_arenadata2_b1-1

24.3.2.23_arenadata2_b1-1 will be started.

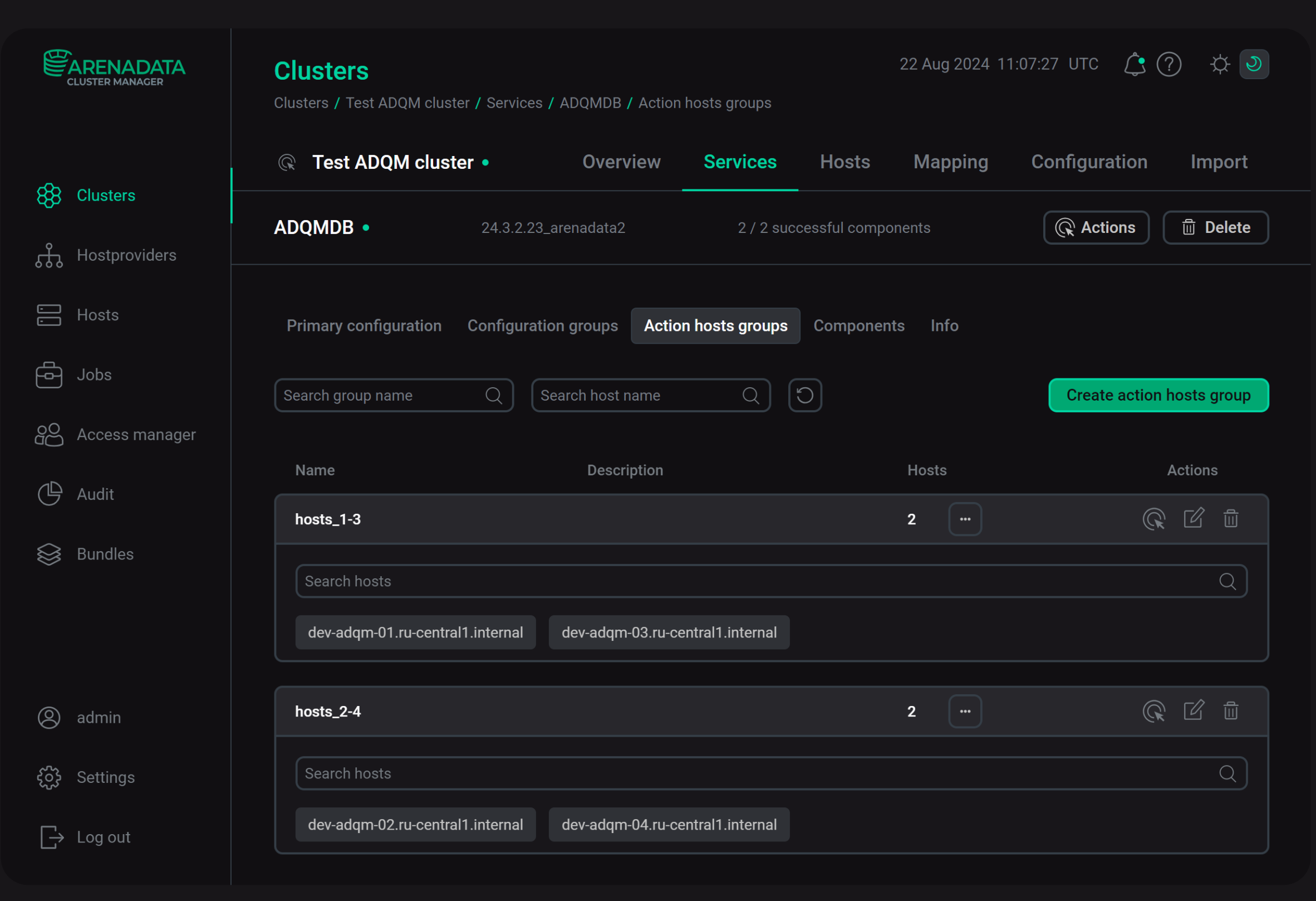
☐ Verbose

Подтверждение выполнения подготовки к обновлению

Когда действие будет выполнено, ADCM изменит статус кластера на `upgrading` — кластер готов к обновлению.

Шаг 2. Обновление ADQMDB по группам хостов

1. На странице конфигурации сервиса ADQMDB (***Clusters → <ADQM_cluster> → Services → ADQMDB***) откройте вкладку ***Action hosts groups*** и объедините хосты кластера в группы, чтобы затем обновить версию ADQMDB поочередно для каждой группы хостов. Например, можно создать группы хостов так, чтобы каждая из них включала по одной реплике каждого шарда. На рисунке ниже показан пример разделения хостов на группы по этому принципу для следующей топологии кластера:
   - шард 1 с двумя репликами — *dev-adqm-01.ru-central1.internal* и *dev-adqm-02.ru-central1.internal*;
   - шард 2 с двумя репликами — *dev-adqm-03.ru-central1.internal* и *dev-adqm-04.ru-central1.internal*.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Overview
Services
Hosts
Mapping
Configuration
Import
Actions
Delete
Primary configuration
Configuration groups
Action hosts groups
Components
Info
Create action hosts group

Clusters / Test ADQM cluster / Services / ADQMDB / Action hosts groups

22 Aug 2024 11:07:27 UTC

Test ADQM cluster

ADQMDB 24.3.2.23_arenadata2 2 / 2 successful components

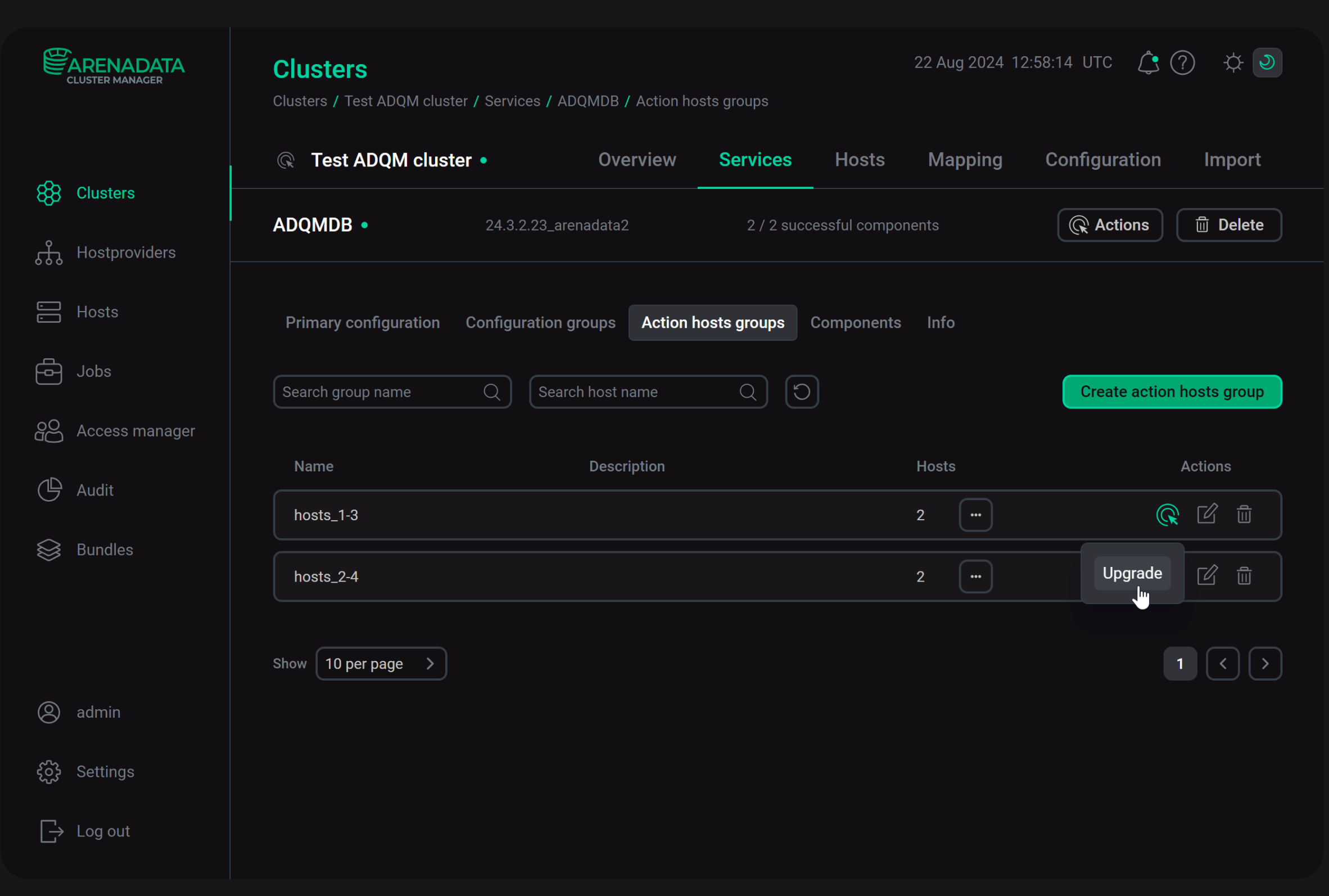
Name	Description	Hosts	Actions
hosts_1-3		2	
hosts_2-4		2	

hosts_1-3: dev-adqm-01.ru-central1.internal, dev-adqm-03.ru-central1.internal

hosts_2-4: dev-adqm-02.ru-central1.internal, dev-adqm-04.ru-central1.internal

Распределение хостов по группам для последовательного обновления сервиса ADQMDB

2. Выполните обновление сервиса ADQMDB для одной группы хостов. Для этого в строке группы хостов кликните по иконке в столбце ***Actions*** и запустите действие ***Upgrade***.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Overview
Services
Hosts
Mapping
Configuration
Import
Actions
Delete
Primary configuration
Configuration groups
Action hosts groups
Components
Info
Create action hosts group
Upgrade

Clusters / Test ADQM cluster / Services / ADQMDB / Action hosts groups

22 Aug 2024 12:58:14 UTC

Test ADQM cluster

ADQMDB 24.3.2.23_arenadata2 2 / 2 successful components

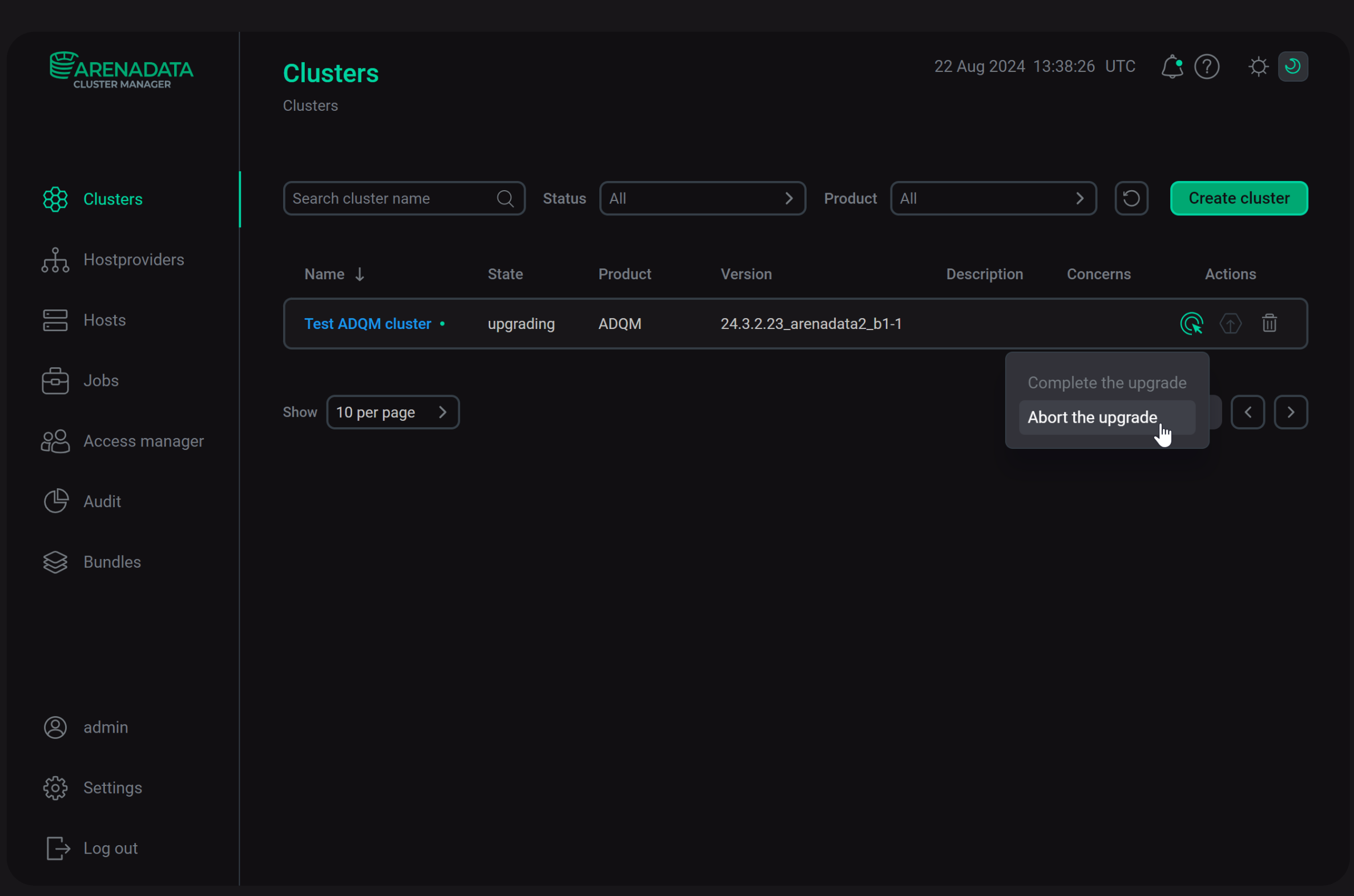
Name	Description	Hosts	Actions
hosts_1-3		2	
hosts_2-4		2	

Запуск обновления ADQMDB для группы хостов

   В открывшемся диалоговом окне нажмите ***Run***, чтобы подтвердить выполнение обновления. После завершения обновления сервиса ADQMDB для группы хостов, кластер останется в состоянии `upgrading`.

   Действие ***Upgrade*** можно запустить для группы хостов повторно, если, например, не удалось выполнить обновление на некоторых хостах или в группу были добавлены новые хосты.

3. После успешного выполнения действия ***Upgrade*** убедитесь, что на всех хостах группы установлена новая версия ClickHouse (например, можно выполнить команду `select version()` в консольном клиенте clickhouse-client) и базы данных работают корректно (ClickHouse со своей стороны гарантирует техническую совместимость двух версий в пределах года, но бизнес-поведение может меняться, поэтому прикладной результат следует проверять для каждой версии отдельно). Дальнейшие действия зависят от результата проверки:
   - Если проверка прошла успешно, можно продолжить обновление ADQMDB на других хостах кластера (запустить действие ***Upgrade*** для следующей группы хостов) или перейти к завершению обновления кластера.
   - В случае возникновения каких-либо проблем при тестировании новой версии ClickHouse можно прервать обновление кластера и вернуть предыдущую версию сервиса ADQMDB на обновленных ранее хостах. Чтобы это сделать, перейдите на страницу ***Clusters*** и запустите для кластера действие ***Abort the upgrade***.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Create cluster
Complete the upgrade
Abort the upgrade

Clusters

22 Aug 2024 13:38:26 UTC

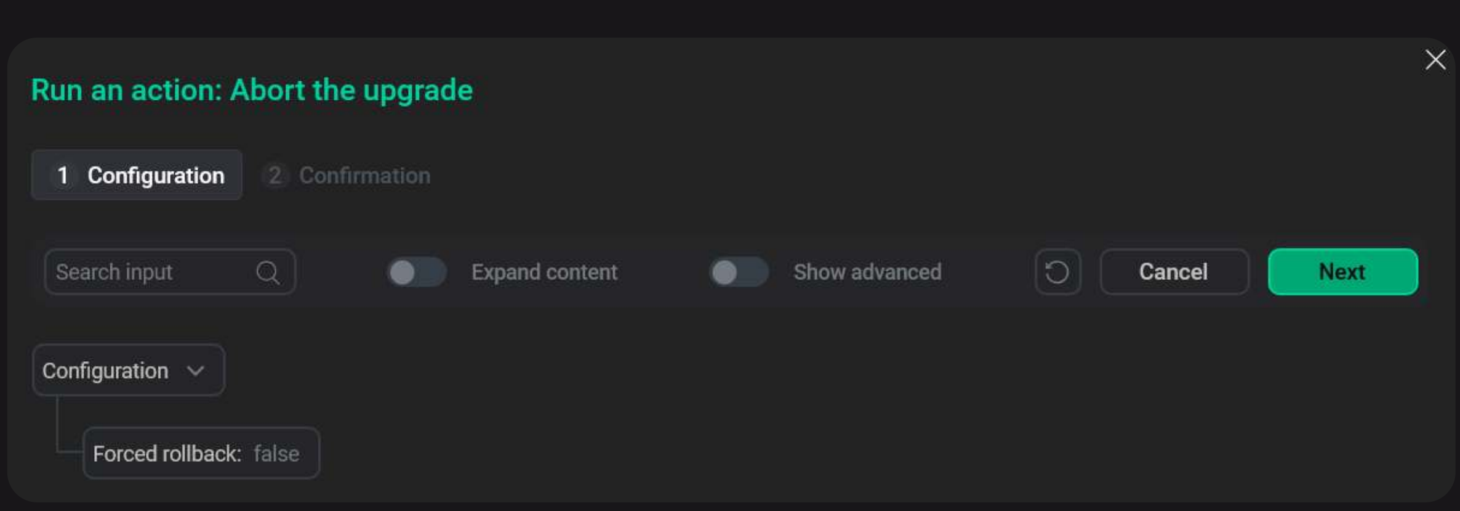
Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	upgrading	ADQM	24.3.2.23_arenadata2_b1-1			

Отмена обновления сервиса ADQMDB

   После отмены обновления бандл кластера вернется к предыдущей версии (и ClickHouse на всех хостах кластера соответственно тоже), а состояние кластера установится в `installed`.

   Если отменить обновление не удалось (например, из-за того, что поврежден или отсутствует файл с информацией, необходимой для возврата к предыдущей версии, или архив с файлами конфигурации старой версии сервиса ADQMDB), можно запустить действие ***Abort the upgrade*** с включенной опцией ***Forced rollback*** — в этом случае осуществится принудительный откат к предыдущей версии бандла ADQM, после которого необходимо будет запустить действие ***Reconfig & restart*** для сервиса ADQMDB.

   Начиная с версии ADQM 24.3.11.7.1.b1 установка опции ***Forced rollback*** в действии ***Abort the upgrade*** приводит к восстановлению репозиториев ADQM и удалению обновленных пакетов ADQM.



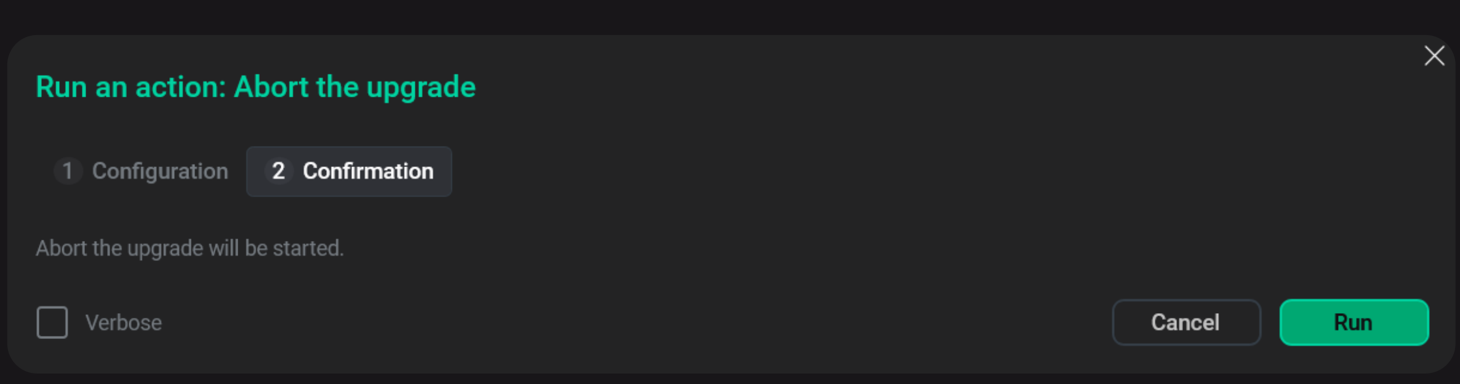
Run an action: Abort the upgrade

1 Configuration
2 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Forced rollback: false

Опция Forced rollback для отмены обновления

   После ввода параметров действия на вкладке ***Configuration*** необходимо нажать ***Next*** и подтвердить действие на следующей вкладке ***Confirmation***, нажав ***Run***.



Run an action: Abort the upgrade

1 Configuration
2 Confirmation
Cancel
Run

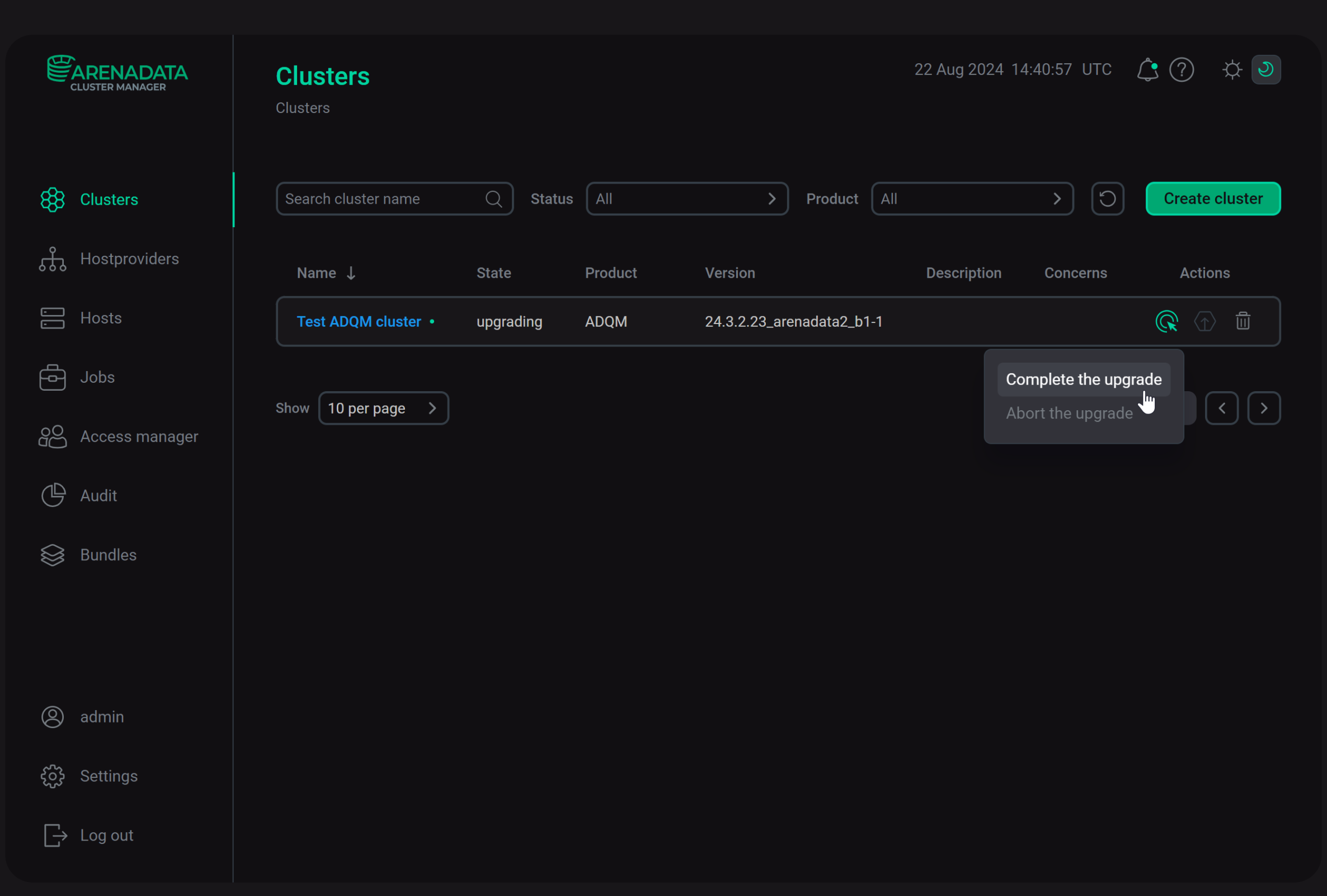
Abort the upgrade will be started.

☐ Verbose

Подтверждение отмены обновления

Шаг 3. Завершение обновления

После того, как сервис ADQMDB переведен на новую версию на хостах, где было необходимо провести предварительное обновление, завершите обновление кластера действием ***Complete the upgrade*** (на странице ***Clusters***), которое обновит ADQMDB на остальных хостах, а также все другие сервисы ADQM в кластере. После выполнения этого действия уже не будет возможности вернуться к предыдущей версии ADQM.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Create cluster
Complete the upgrade
Abort the upgrade

Clusters

22 Aug 2024 14:40:57 UTC

Name	State	Product	Version	Description	Concerns	Actions
Test ADQM cluster	upgrading	ADQM	24.3.2.23_arenadata2_b1-1			

Завершение обновления кластера ADQM

Когда процесс обновления будет завершен, ADCM изменит состояние кластера на `installed`.

> **ПРИМЕЧАНИЕ**
>
> Запуск действия ***Complete the upgrade*** начинает обычное обновление кластера, то есть обновление всех сервисов ADQM на всех хостах (как при запуске действия Upgrade для кластера, обновляемого на версию ADQM до 24.3.2.23.1.b1 включительно).

Режим обслуживания

Елена Дворядкина

Содержание

Назад к содержанию

Обзор

Примеры

- Перевод хоста в режим обслуживания
- Удаление хоста из кластера
- Замена хоста ADQMDB

Обзор

Для хостов ADQM в пользовательском интерфейсе ADCM поддерживается режим обслуживания (maintenance mode). Временно недоступные или работающие некорректно хосты можно перевести в режим обслуживания, чтобы исключить их участие в действиях на уровне кластера или сервисов и выполнять действия без возможных ошибок, которые могут быть связаны с этими хостами. Состояние хостов в режиме обслуживания не учитывается при агрегировании информации по состоянию кластера/сервиса.

Эту функциональность можно использовать для обслуживания оборудования или программного обеспечения, изменения параметров конфигурации, устранения неполадок, вывода из эксплуатации или удаления узлов кластера.

Особенности и ограничения использования режима обслуживания для ADQM:

- На хост в режиме обслуживания запрещается добавлять какие-либо компоненты сервисов ADQM, но можно на логическом уровне удалить компоненты с хоста в режиме обслуживания, если он недоступен.
- Если хотя бы один хост кластера переведен в режим обслуживания, кластерные/сервисные действия ***Install*** и ***Upgrade*** недоступны.
- Для сервисов Zookeeper и Clickhousekeeper выполнение действия ***Reconfig and restart***/***Reconfig*** может завершаться ошибкой, если лидер кластера ZooKeeper/ClickHouse Keeper находится на хосте в режиме обслуживания.
- Выполнение действий сервиса ADQMDB (кроме ***Install*** и Upgrade) при наличии хостов в режиме обслуживания поддерживается в том числе если используется интегрированный ClickHouse Keeper.

ПРИМЕЧАНИЕ

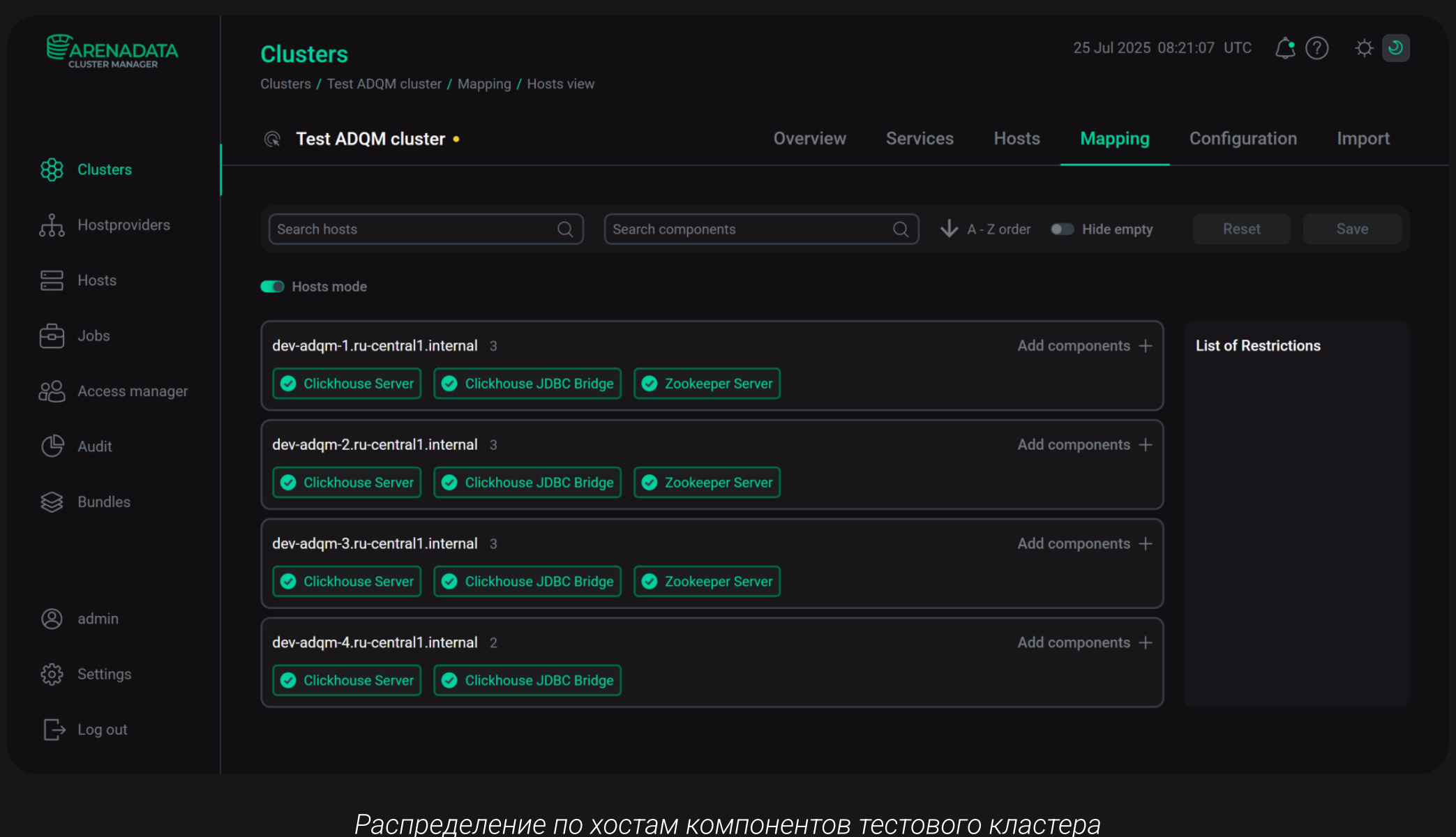
Режим обслуживания в ADQM на данный момент поддерживается только для хостов.

Примеры

В приведенных ниже примерах используется развернутый на четырех хостах кластер ADQM, который имеет следующую топологию:

- шард 1 с двумя репликами — *dev-adqm-1.ru-central1.internal* и *dev-adqm-2.ru-central1.internal*;
- шард 2 с двумя репликами — *dev-adqm-3.ru-central1.internal* и *dev-adqm-4.ru-central1.internal*.

В кластере установлены сервисы ADQMDB и Zookeeper, компоненты которых распределены по хостам кластера как показано на следующем рисунке.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Mapping / Hosts view
25 Jul 2025 08:21:07 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
List of Restrictions
Add components

dev-adqm-1.ru-central1.internal 3: Clickhouse Server, Clickhouse JDBC Bridge, Zookeeper Server
dev-adqm-2.ru-central1.internal 3: Clickhouse Server, Clickhouse JDBC Bridge, Zookeeper Server
dev-adqm-3.ru-central1.internal 3: Clickhouse Server, Clickhouse JDBC Bridge, Zookeeper Server
dev-adqm-4.ru-central1.internal 2: Clickhouse Server, Clickhouse JDBC Bridge

Распределение по хостам компонентов тестового кластера

В кластер добавлена одна реплицируемая таблица для тестирования:

```sql
CREATE TABLE test_repl_table on cluster 'default_cluster' (id UInt64)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/test_repl_table', '{replica}')
ORDER BY id;
```

В таблицу шарда 2 добавлена строка данных:

```sql
INSERT INTO test_repl_table VALUES ('1');
```

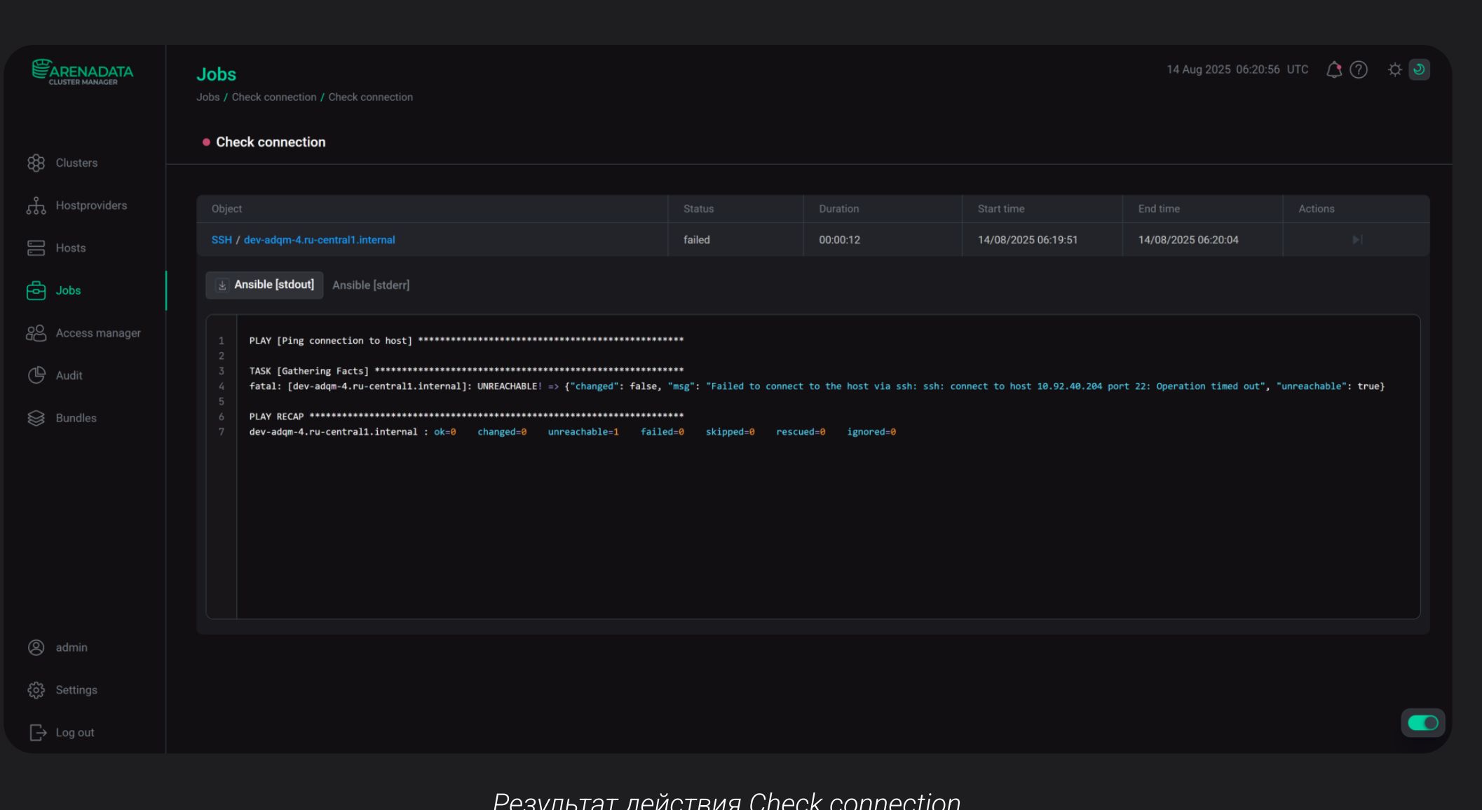
Перевод хоста в режим обслуживания

Следующий пример показывает, как включить режим обслуживания для хоста ADQMDB, чтобы временно исключить его из обработки.

1. Выключите хост, на котором установлены компоненты сервиса ADQMDB — например, хост *dev-adqm-4.ru-central1.internal*.

Проверка соединения с хостом по SSH

В Ansible [stdout] действия ***Check connection*** выводится сообщение о недоступности хоста.


Jobs
Jobs / Check connection / Check connection
14 Aug 2025 06:20:56 UTC
Check connection
Ansible [stdout]
Ansible [stderr]

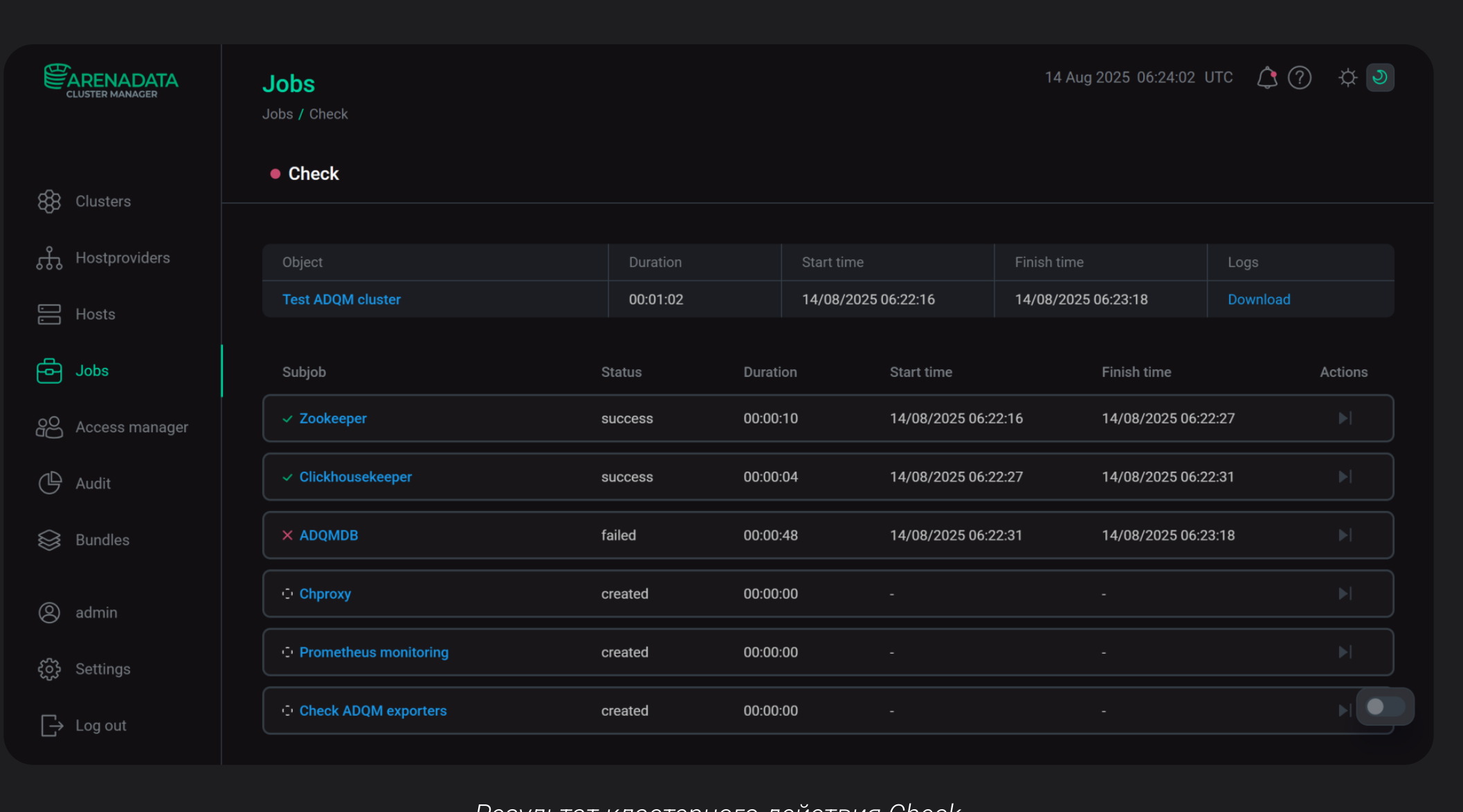
Object	Status	Duration	Start time	End time	Actions
SSH / dev-adqm-4.ru-central1.internal	failed	00:00:12	14/08/2025 06:19:51	14/08/2025 06:20:04	

```
PLAY [Ping connection to host] ************************************************

TASK [Gathering Facts] ********************************************************
fatal: [dev-adqm-4.ru-central1.internal]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: connect to host 10.92.40.204 port 22: Operation timed out", "unreachable": true}

PLAY RECAP ********************************************************************
dev-adqm-4.ru-central1.internal : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
```

Результат действия Check connection

Проверка состояния кластера

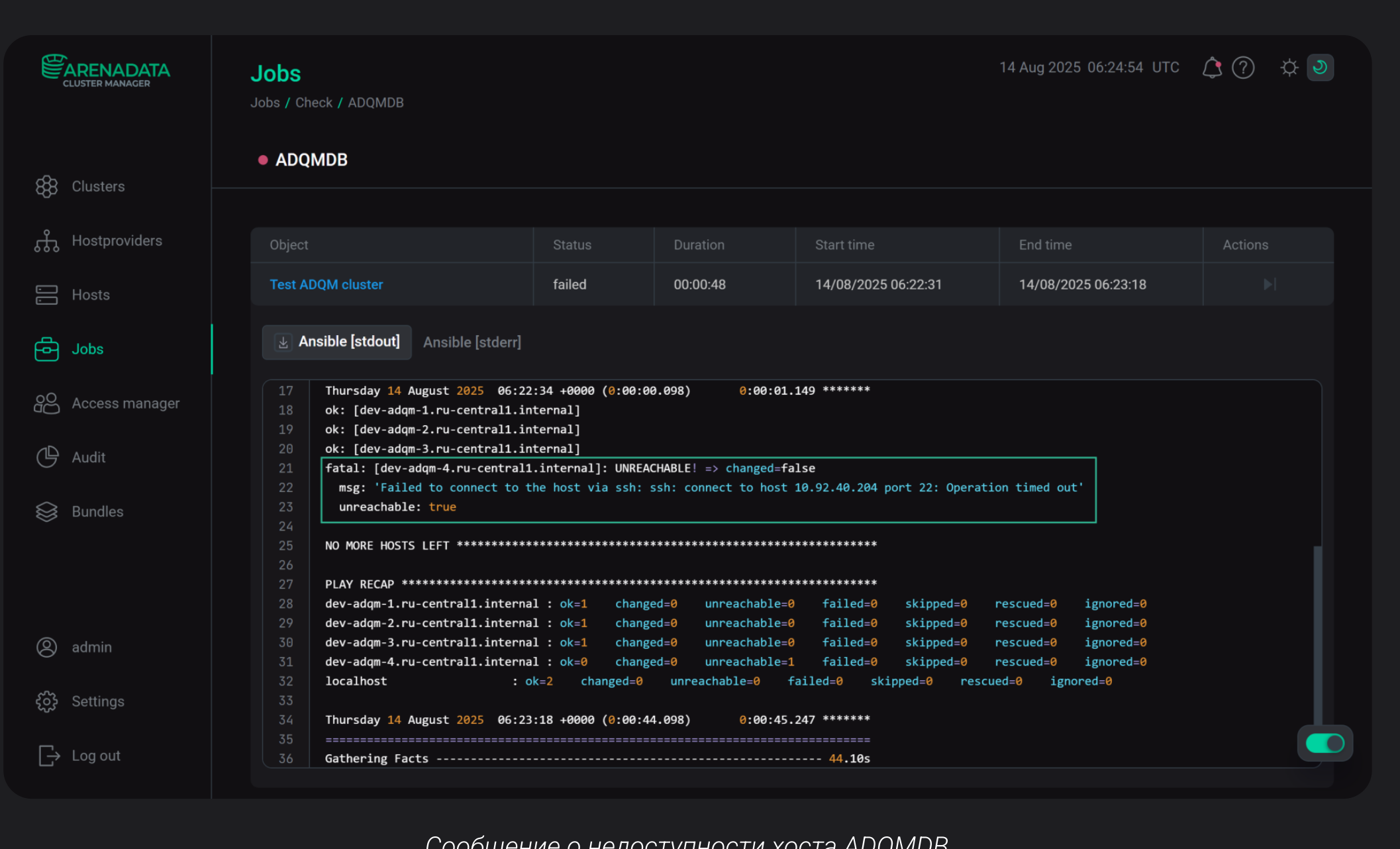
Выполнение кластерного действия ***Check*** завершается ошибкой, так как при проверке сервиса ADQMDB обнаружено, что один из хостов недоступен.


Jobs
Jobs / Check
14 Aug 2025 06:24:02 UTC
Check

Object	Duration	Start time	Finish time	Logs
Test ADQM cluster	00:01:02	14/08/2025 06:22:16	14/08/2025 06:23:18	Download

Subjob	Status	Duration	Start time	Finish time	Actions
Zookeeper	success	00:00:10	14/08/2025 06:22:16	14/08/2025 06:22:27	
Clickhousekeeper	success	00:00:04	14/08/2025 06:22:27	14/08/2025 06:22:31	
ADQMDB	failed	00:00:48	14/08/2025 06:22:31	14/08/2025 06:23:18	
Chproxy	created	00:00:00	-	-	
Prometheus monitoring	created	00:00:00	-	-	
Check ADQM exporters	created	00:00:00	-	-	

Результат кластерного действия Check

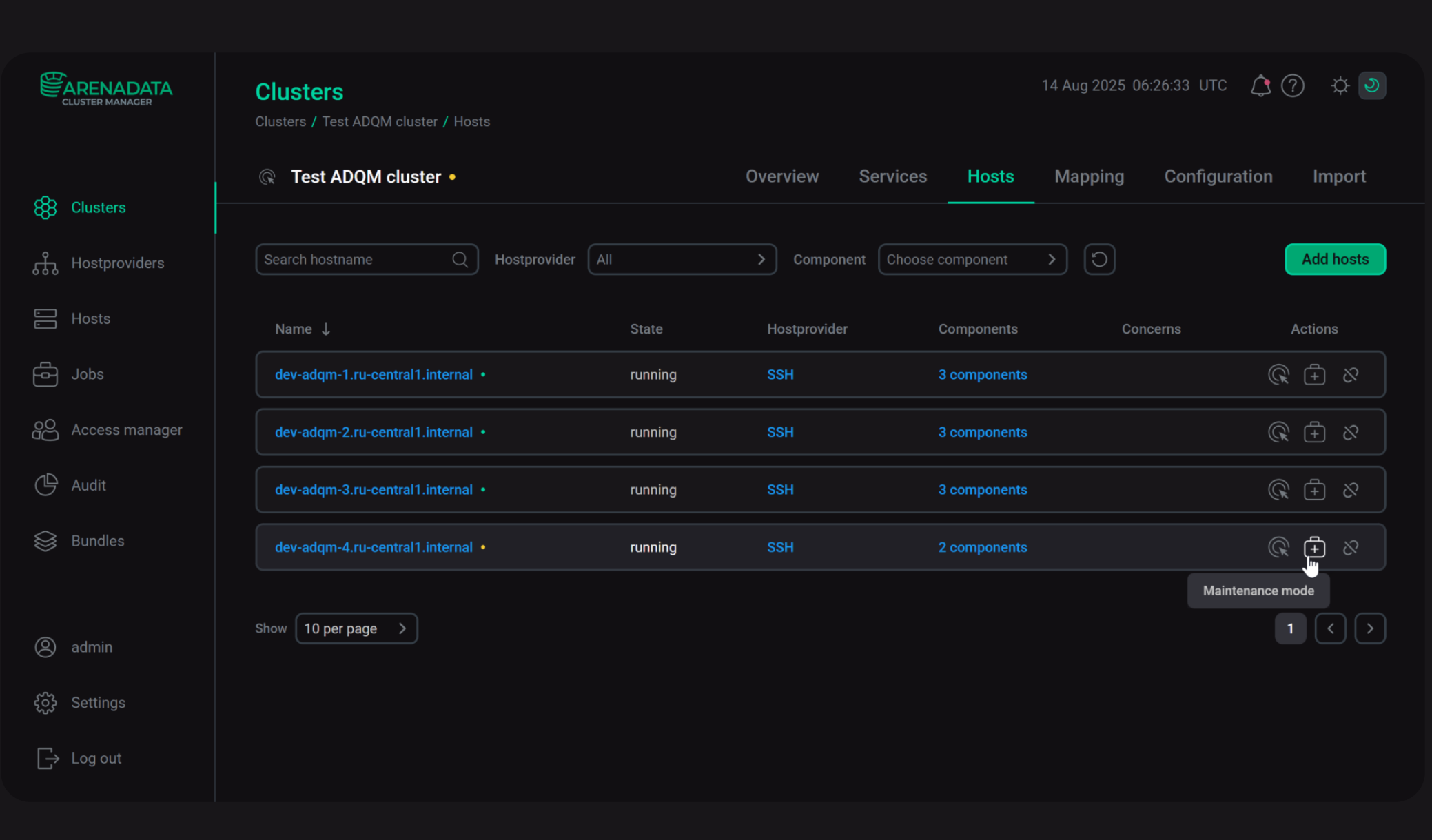

Jobs
Jobs / Check / ADQMDB
14 Aug 2025 06:24:54 UTC
ADQMDB
Ansible [stdout]
Ansible [stderr]

Object	Status	Duration	Start time	End time	Actions
Test ADQM cluster	failed	00:00:48	14/08/2025 06:22:31	14/08/2025 06:23:18	

```
Thursday 14 August 2025  06:22:34 +0000 (0:00:00.098)       0:00:01.149 *******
ok: [dev-adqm-1.ru-central1.internal]
ok: [dev-adqm-2.ru-central1.internal]
ok: [dev-adqm-3.ru-central1.internal]
fatal: [dev-adqm-4.ru-central1.internal]: UNREACHABLE! => changed=false
  msg: 'Failed to connect to the host via ssh: ssh: connect to host 10.92.40.204 port 22: Operation timed out'
  unreachable: true

NO MORE HOSTS LEFT ************************************************************

PLAY RECAP ********************************************************************
dev-adqm-1.ru-central1.internal : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
dev-adqm-2.ru-central1.internal : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
dev-adqm-3.ru-central1.internal : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
dev-adqm-4.ru-central1.internal : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

Thursday 14 August 2025  06:23:18 +0000 (0:00:44.098)       0:00:45.247 *******
===============================================================================
Gathering Facts --------------------------------------------------------- 44.10s
```

Сообщение о недоступности хоста ADQMDB

2. Переведите хост *dev-adqm-4.ru-central1.internal* в режим обслуживания. Для этого в строке хоста на вкладке ***Hosts*** страницы кластера нажмите на иконку ⊞.

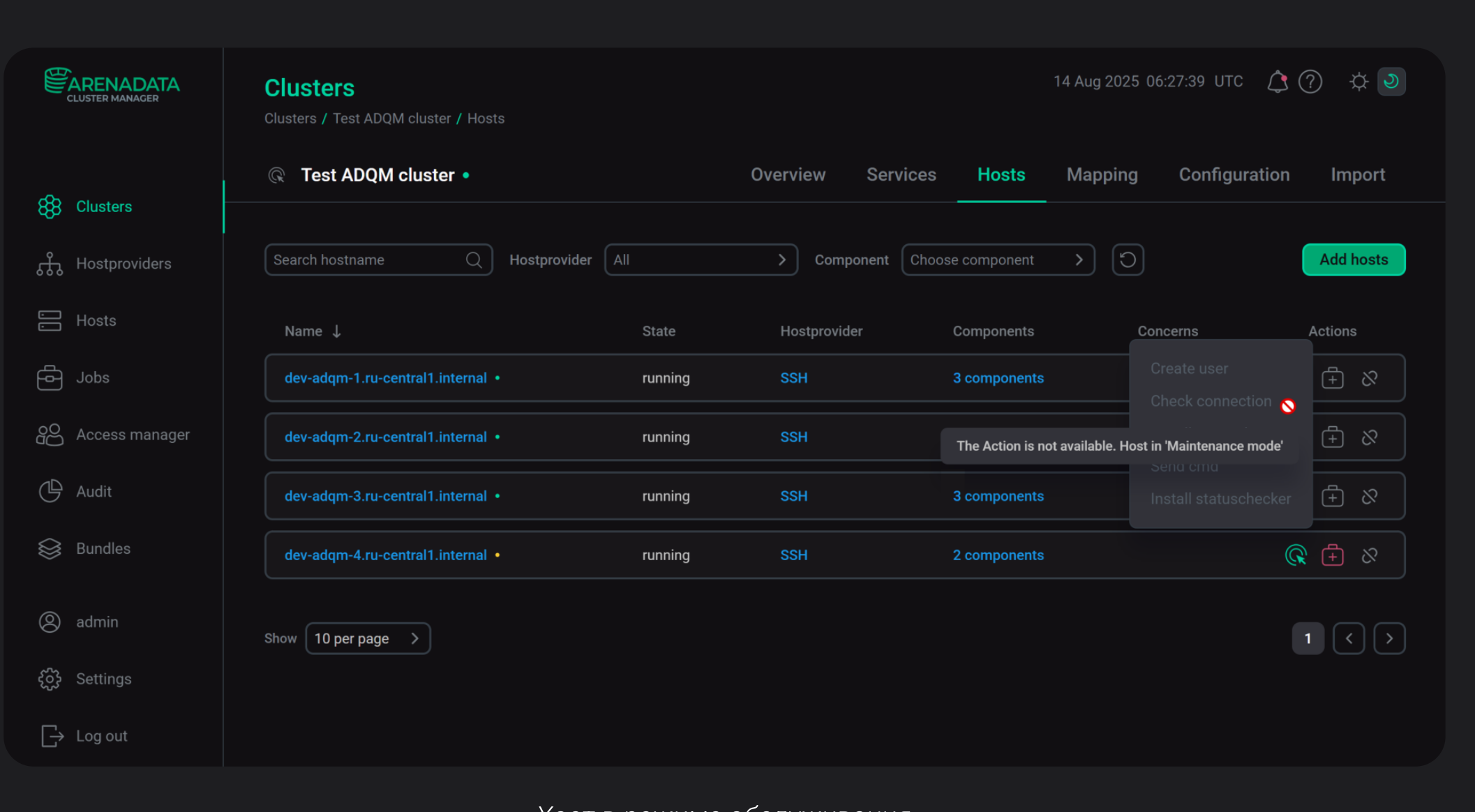

Clusters
Clusters / Test ADQM cluster / Hosts
14 Aug 2025 06:26:33 UTC
Test ADQM cluster
Search hostname
Hostprovider
All
Component
Choose component
Add hosts
Maintenance mode
Show
10 per page

Name	State	Hostprovider	Components	Concerns	Actions
dev-adqm-1.ru-central1.internal	running	SSH	3 components		
dev-adqm-2.ru-central1.internal	running	SSH	3 components		
dev-adqm-3.ru-central1.internal	running	SSH	3 components		
dev-adqm-4.ru-central1.internal	running	SSH	2 components		

Включение режима обслуживания для хоста

Хост в режиме обслуживания

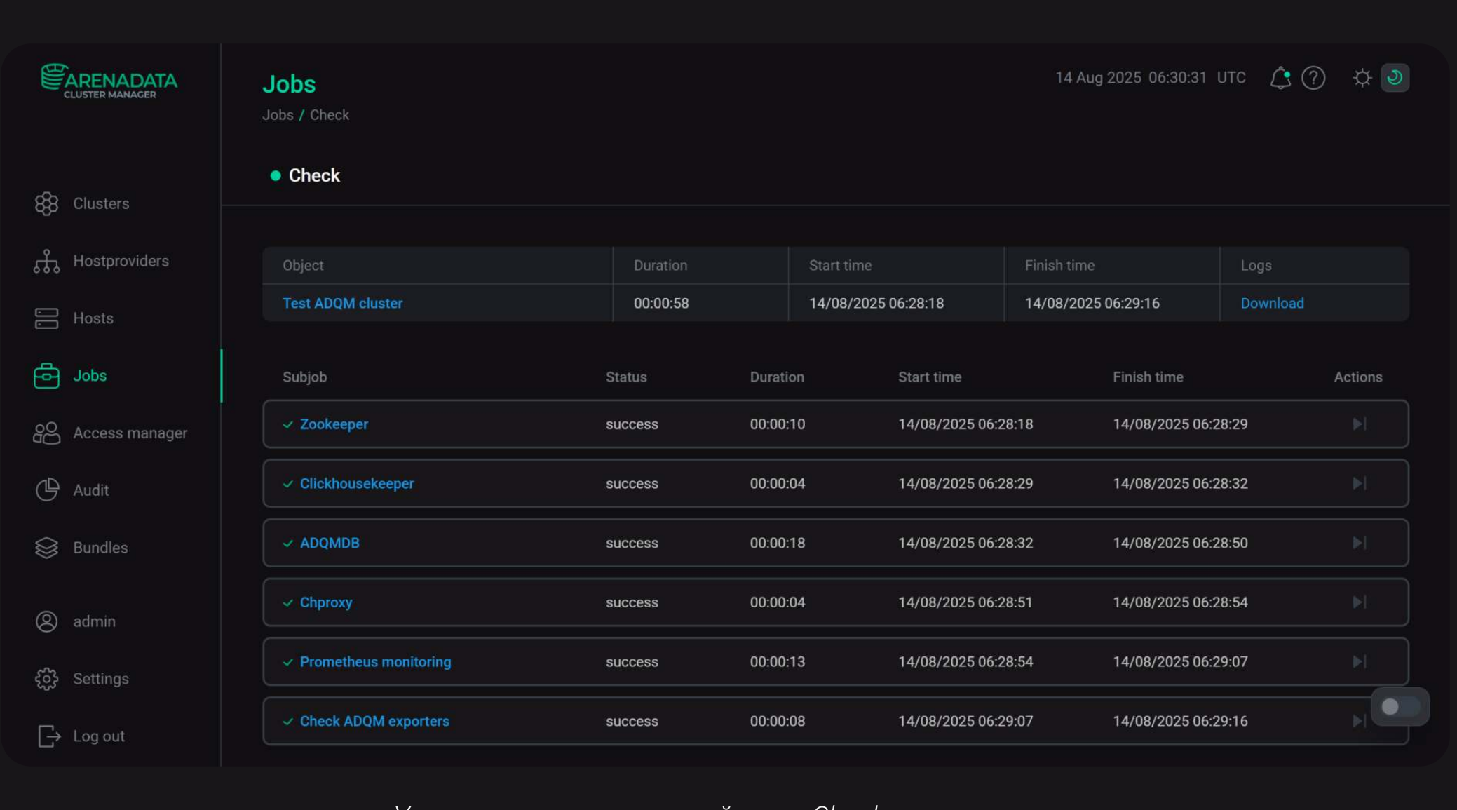
Для хоста в режиме обслуживания становятся недоступны действия в интерфейсе ADCM.


Clusters
Clusters / Test ADQM cluster / Hosts
14 Aug 2025 06:27:39 UTC
Test ADQM cluster
Create user
Check connection
The Action is not available. Host in 'Maintenance mode'
Send cmd
Install statuschecker
Add hosts

Name	State	Hostprovider	Components	Concerns	Actions
dev-adqm-1.ru-central1.internal	running	SSH	3 components		
dev-adqm-2.ru-central1.internal	running	SSH			
dev-adqm-3.ru-central1.internal	running	SSH	3 components		
dev-adqm-4.ru-central1.internal	running	SSH	2 components		

Хост в режиме обслуживания

Теперь полная недоступность хоста по сети не приводит к ошибке выполнения кластерных действий (например, ***Check***) и действий сервиса ADQMDB (***Check***, ***Start***, ***Stop***, ***Restart***, ***Reconfig***, ***Manage auto core dump***).

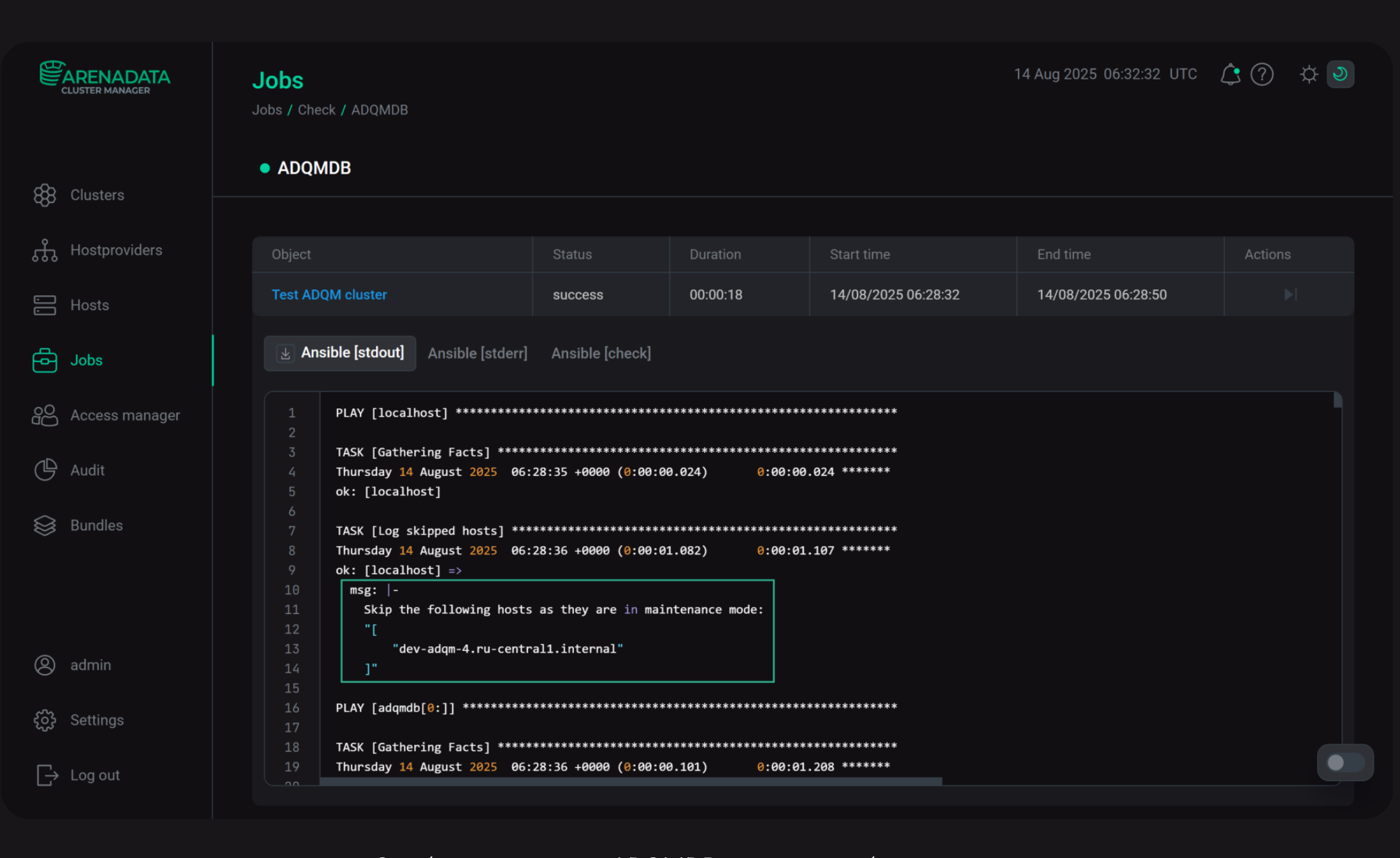

Jobs
Jobs / Check
14 Aug 2025 06:30:31 UTC
Check

Object	Duration	Start time	Finish time	Logs
Test ADQM cluster	00:00:58	14/08/2025 06:28:18	14/08/2025 06:29:16	Download

Subjob	Status	Duration	Start time	Finish time	Actions
Zookeeper	success	00:00:10	14/08/2025 06:28:18	14/08/2025 06:28:29	
Clickhousekeeper	success	00:00:04	14/08/2025 06:28:29	14/08/2025 06:28:32	
ADQMDB	success	00:00:18	14/08/2025 06:28:32	14/08/2025 06:28:50	
Chproxy	success	00:00:04	14/08/2025 06:28:51	14/08/2025 06:28:54	
Prometheus monitoring	success	00:00:13	14/08/2025 06:28:54	14/08/2025 06:29:07	
Check ADQM exporters	success	00:00:08	14/08/2025 06:29:07	14/08/2025 06:29:16	

Успешное выполнение действия Check для кластера

При этом в начале Ansible [stdout] шага действия, соответствующего проверке сервиса ADQMDB, выводится сообщение о том, что хост не участвует в обработке, так как находится в режиме обслуживания.


Jobs
Jobs / Check / ADQMDB
14 Aug 2025 06:32:32 UTC
ADQMDB
Ansible [stdout]
Ansible [stderr]
Ansible [check]

Object	Status	Duration	Start time	End time	Actions
Test ADQM cluster	success	00:00:18	14/08/2025 06:28:32	14/08/2025 06:28:50	

```
PLAY [localhost] **************************************************************

TASK [Gathering Facts] ********************************************************
Thursday 14 August 2025  06:28:35 +0000 (0:00:00.024)       0:00:00.024 *******
ok: [localhost]

TASK [Log skipped hosts] ******************************************************
Thursday 14 August 2025  06:28:36 +0000 (0:00:01.082)       0:00:01.107 *******
ok: [localhost] =>
  msg: |-
    Skip the following hosts as they are in maintenance mode:
    "[
       "dev-adqm-4.ru-central1.internal"
    ]"

PLAY [adqmdb[0:]] *************************************************************

TASK [Gathering Facts] ********************************************************
Thursday 14 August 2025  06:28:36 +0000 (0:00:00.101)       0:00:01.208 *******
```

Сообщение о хосте ADQMDB в режиме обслуживания

После выполнения действия ***Reconfig*** для сервиса ADQMDB введенный в режим обслуживания хост будет удален из описания топологии кластера в секции `remote_servers` конфигурационного файла */etc/clickhouse-server/config.xml* на всех хостах ADQMDB, кроме хоста в режиме обслуживания. При этом кластер без этого хоста остается в работоспособном состоянии — на других хостах данные доступны, с ними можно продолжать работать.

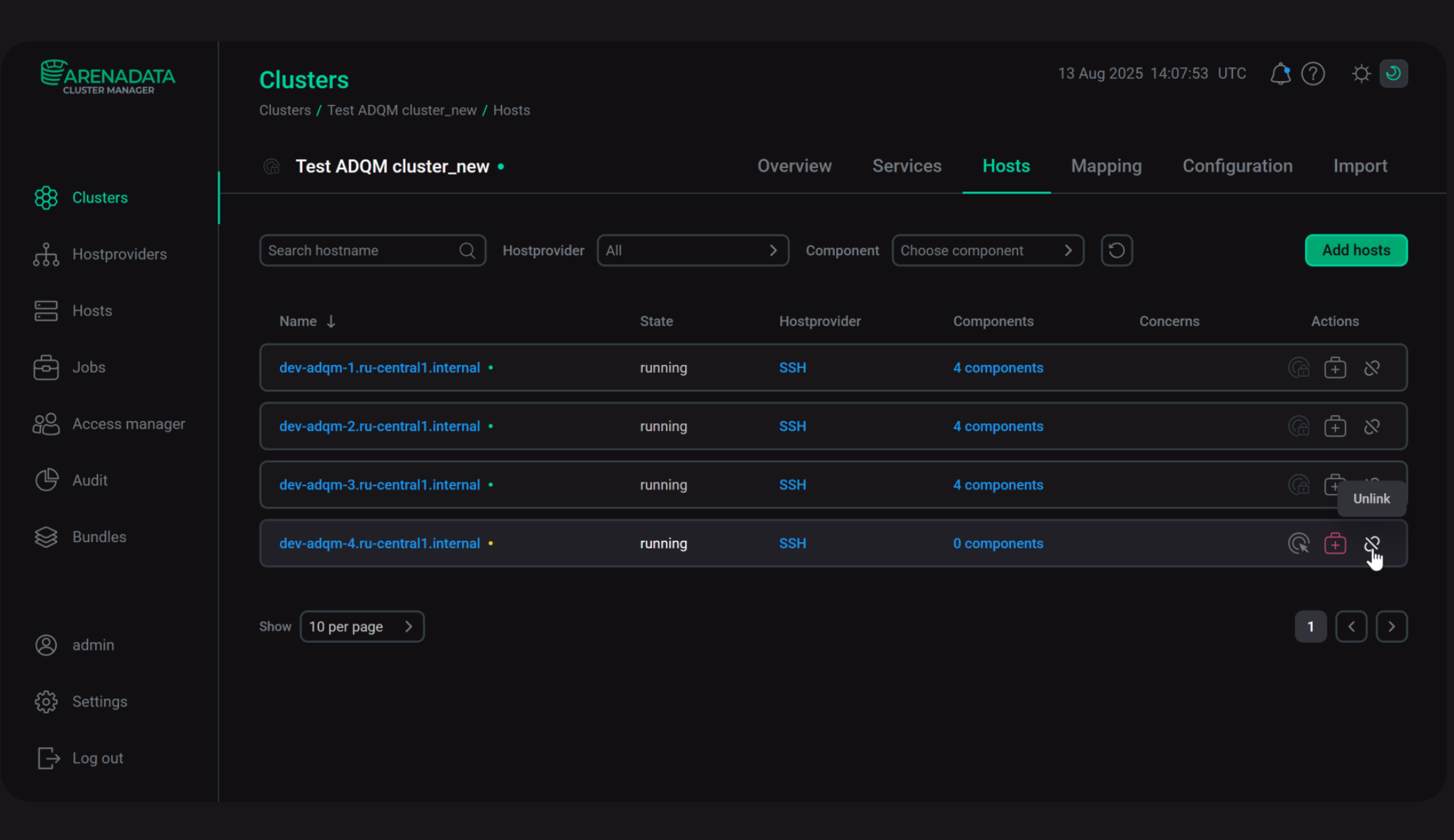
Чтобы вернуть хост в обработку действиями в ADCM, необходимо вывести его из режима обслуживания, затем выполнить действие ***Reconfig*** для сервиса ADQMDB.

Удаление хоста из кластера

Если один из хостов ADQM вышел из строя, он может быть удален из кластера (например, такой сценарий допускается, если в кластере останутся реплики данных и наличие этого хоста не критично, или взамен удаленного хоста будет добавлен новый хост).

В следующем примере из кластера удаляется хост *dev-adqm-4.ru-central1.internal*, на котором установлены компоненты сервиса ADQMDB.

1. Переведите хост *dev-adqm-4.ru-central1.internal* в режим обслуживания как описано выше.
2. Удалите реплику, соответствующую хосту *dev-adqm-4.ru-central1.internal*, из топологии логического кластера, используя настройку Cluster Configuration сервиса ADQMDB. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы применить новую топологию кластера.
3. Запустите действие ***Add/Remove components*** для сервиса ADQMDB и удалите компоненты Clickhouse Server и Clickhouse JDBC Bridge с хоста *dev-adqm-4.ru-central1.internal*.
4. После освобождения хоста от компонентов его можно удалить из кластера — на странице ***Hosts*** нажмите на иконку ⦸.


Clusters
Clusters / Test ADQM cluster_new / Hosts
13 Aug 2025 14:07:53 UTC
Test ADQM cluster_new
Search hostname
Hostprovider
All
Component
Choose component
Add hosts
Unlink
Show
10 per page

Name	State	Hostprovider	Components	Concerns	Actions
dev-adqm-1.ru-central1.internal	running	SSH	4 components		
dev-adqm-2.ru-central1.internal	running	SSH	4 components		
dev-adqm-3.ru-central1.internal	running	SSH	4 components		
dev-adqm-4.ru-central1.internal	running	SSH	0 components		

Удаление хоста из кластера

Подтвердите действие, нажав ***Unlink*** в появившемся окне.

ВНИМАНИЕ

При удалении хоста ZooKeeper или ClickHouse Keeper необходимо подготовить хост-замену и поменять вышедший из строя хост на хост-замену в одно действие ***Add/Remove components*** для сервиса Zookeeper или Clickhousekeeper (так как число хостов ZooKeeper/ClickHouse Keeper не должно быть четным).

Замена хоста ADQMDB

Следующий пример показывает, как вместо удаленного хоста ADQMDB добавить в кластер новый хост (с тем же именем — *dev-adqm-4.ru-central1.internal*) и восстановить репликацию данных между новым хостом и *dev-adqm-3.ru-central1.internal*.

1. Создайте в ADCM новый хост *dev-adqm-4.ru-central1.internal* и добавьте его в кластер ADQM.
2. Добавьте этот хост в топологию кластера через параметр ***Cluster Configuration*** сервиса ADQMDB, указав его как реплику хоста *dev-adqm-3.ru-central1.internal* в шарде 2.
3. Вызовите действие ***Add/Remove components*** для сервиса ADQMDB. Добавьте компоненты Clickhouse Server и Clickhouse JDBC Bridge на новый хост.
4. На новом хосте создайте таблицу:

```sql
CREATE TABLE test_repl_table on cluster 'default_cluster' (id UInt64)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/test_repl_table', '{replica}')
ORDER BY id;
```

При возникновении ошибки `REPLICA_ALREADY_EXISTS` перед созданием таблицы на новом хосте выполните следующую команду на хосте *dev-adqm-3.ru-central1.internal*, который является репликой *dev-adqm-4.ru-central1.internal*:

```sql
SYSTEM DROP REPLICA 'dev-adqm-4.ru-central1.internal';
```

Убедитесь, что в таблицу на новом хосте вставились данные с реплики *dev-adqm-3.ru-central1.internal*:

```sql
SELECT * FROM test_repl_table;
```

```
   ┌─id─┐
1. │  1 │
   └────┘
```

Сервисы координации

Елена Дворядкина

Чтобы организовать репликацию данных и выполнение распределенных DDL-запросов (`ON CLUSTER`) в ADQM, при создании кластера необходимо настроить и подключить один из сервисов координации:

- Apache ZooKeeper — известный сервис с открытым исходным кодом для синхронизации и координации распределенных систем.
  ZooKeeper реализован на языке Java, имеет простую и эффективную модель данных. Используемый в ZooKeeper алгоритм координации ZAB (ZooKeeper Atomic Broadcast) обеспечивает линеаризуемость операций записи, но не гарантирует линеаризуемость операций чтения, так как каждый узел ZooKeeper выполняет чтение локально.

- ClickHouse Keeper — альтернативный сервис координации, предоставляющий совместимый с ZooKeeper клиент-серверный протокол.
  ClickHouse Keeper реализован на языке C++ и имеет ту же модель данных, что и ZooKeeper. Используемый алгоритм RAFT предоставляет лучшие гарантии согласованности по сравнению с ZooKeeper — опционально позволяет достичь линеаризуемости чтения. ClickHouse Keeper можно использовать как равноценную замену ZooKeeper, установив его на отдельные серверы, или как внутреннюю часть сервера ClickHouse.

Общие рекомендации по выбору сервиса координации в зависимости от требований к системе:

- Для тестовых или небольших кластеров можно использовать внутренний ClickHouse Keeper, установленный на тех же серверах, что и ClickHouse.

- Для реальных больших систем рекомендуется устанавливать ClickHouse Keeper или ZooKeeper на отдельные серверы.

- Если у вас уже есть настроенный кластер ZooKeeper, можно использовать его для кластера ADQM.

ВАЖНО

- Так как снепшоты и логи ClickHouse Keeper имеют несовместимый с ZooKeeper формат, при миграции кластера ADQM с ZooKeeper на ClickHouse Keeper необходимо сконвертировать данные и перенести их вручную.

- Межсерверный протокол ClickHouse Keeper также несовместим с ZooKeeper, поэтому нельзя создать кластер, который одновременно использует ZooKeeper и ClickHouse Keeper.

ZooKeeper

Елена Дворядкина

Содержание

Назад к содержанию

Внешний ZooKeeper

ZooKeeper как сервис ADQM

Пример тестирования реплицируемых и распределенных таблиц

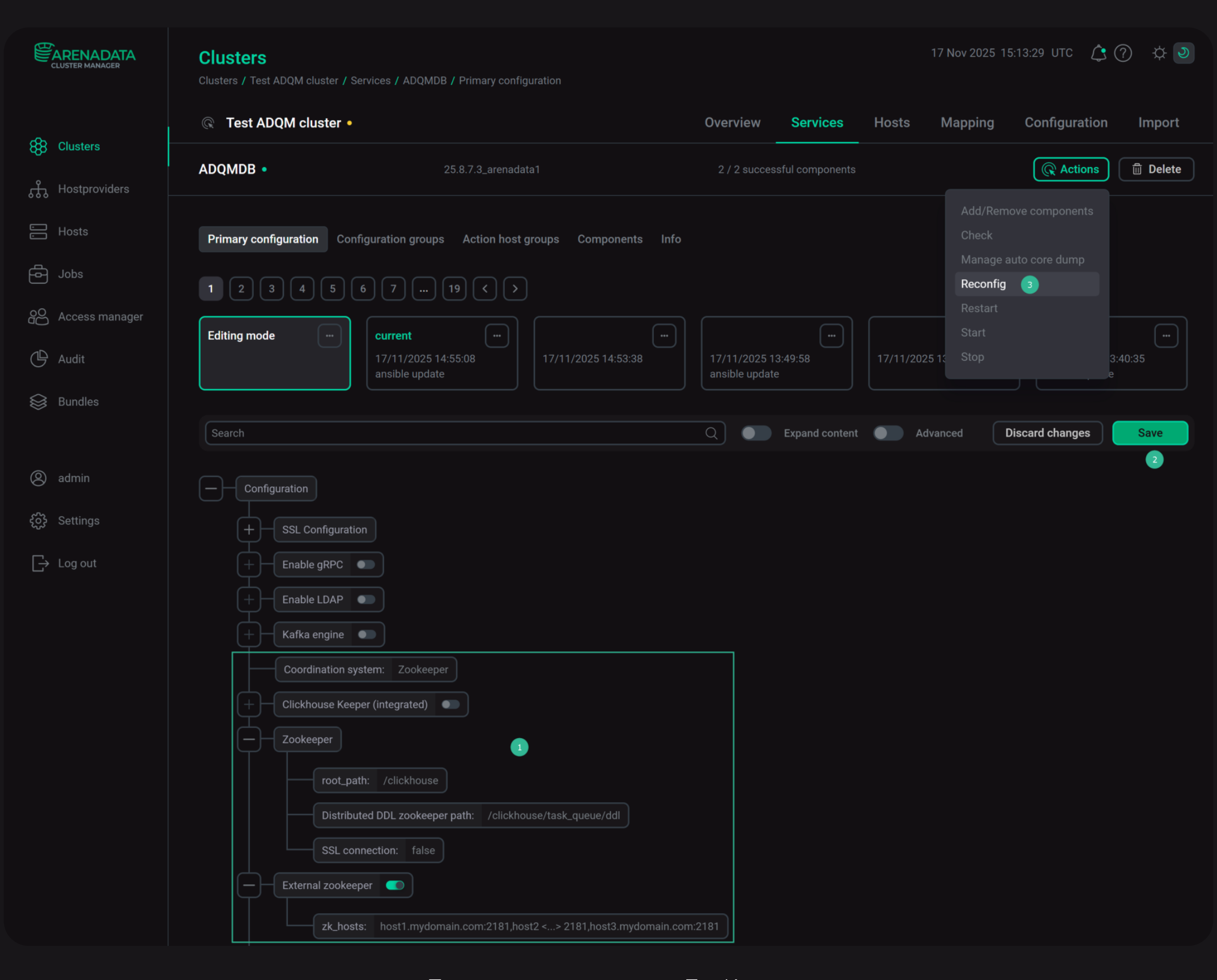
- Конфигурирование кластера ADQM с ZooKeeper
- Реплицируемые таблицы
- Распределенные таблицы

Чтобы репликация данных и выполнение распределенных запросов в ADQM осуществлялись через ZooKeeper, можно:

- использовать внешний кластер ZooKeeper;
- установить сервис ZooKeeper в кластер ADQM.

Внешний ZooKeeper

1. На странице конфигурации сервиса ADQMDB в интерфейсе ADCM выполните следующие действия:
   - активируйте ***External zookeeper*** и в поле ***zk_hosts*** перечислите через запятую все хосты в ансамбле ZooKeeper с указанием портов, через которые они взаимодействуют;
   - укажите пути к узлам ZooKeeper в секции ***Zookeeper***;
   - установите значение `Zookeeper` для параметра ***Coordination system***.
2. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы включить использование внешнего ZooKeeper в качестве сервиса координации для кластера ADQM.



Coordination system: Zookeeper
root_path: /clickhouse
Distributed DDL zookeeper path: /clickhouse/task_queue/ddl
SSL connection: false
External zookeeper
zk_hosts: host1.mydomain.com:2181,host2 <...> 2181,host3.mydomain.com:2181

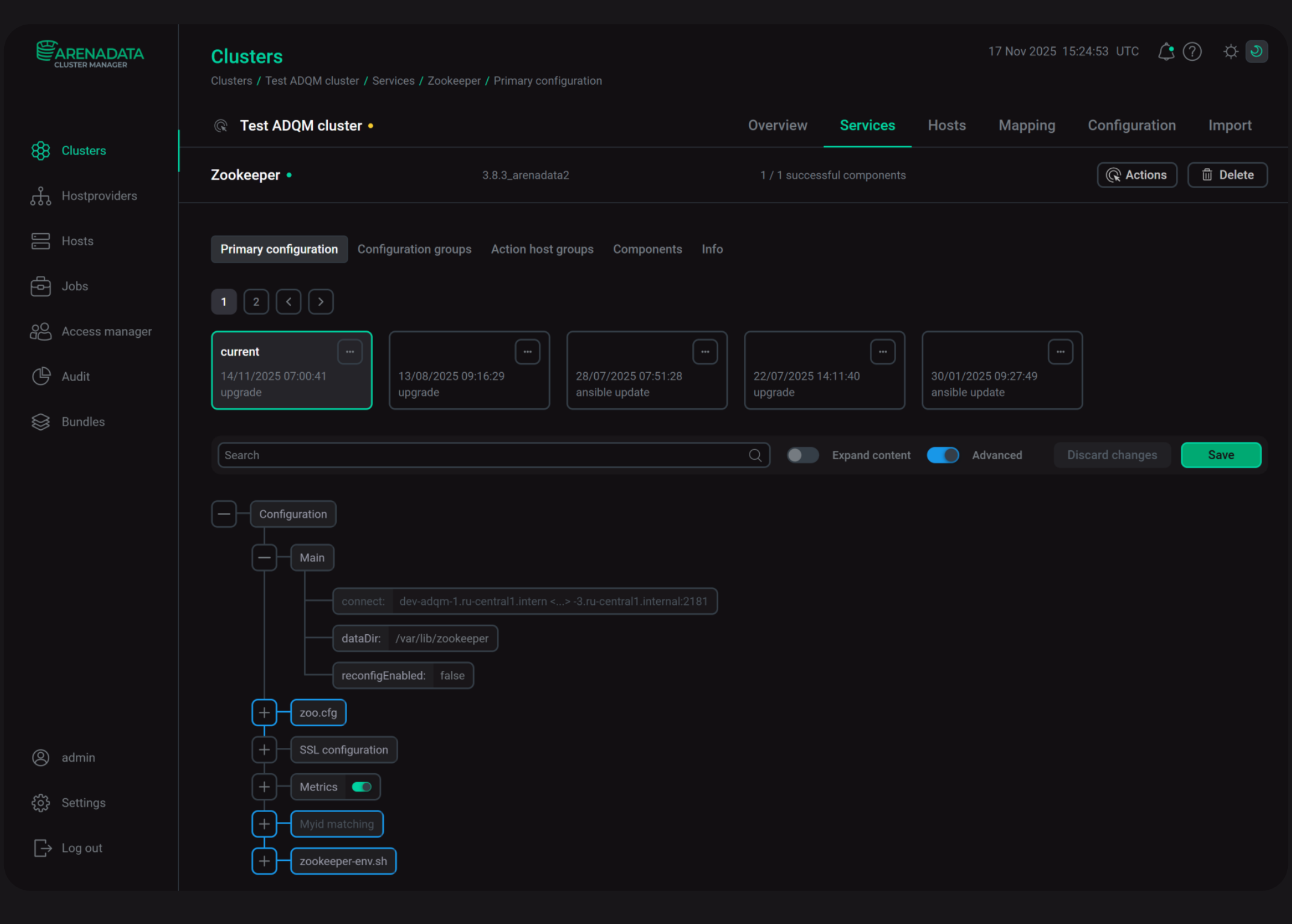
ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
17 Nov 2025 15:13:29 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
25.8.7.3_arenadata1
2 / 2 successful components
Actions
Delete
Add/Remove components
Check
Manage auto core dump
Reconfig
Restart
Start
Stop
Primary configuration
Configuration groups
Action host groups
Components
Info
Editing mode
current
Search
Expand content
Advanced
Discard changes
Save
Configuration
SSL Configuration
Enable gRPC
Enable LDAP
Kafka engine
Clickhouse Keeper (integrated)
Zookeeper
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
1
2
3

Подключение внешнего ZooKeeper

ZooKeeper как сервис ADQM

ADQM предоставляет специальный сервис, который можно добавить и установить в кластер, чтобы репликация и распределение запросов координировались через ZooKeeper. Для этого выполните следующие шаги в интерфейсе ADCM:

1. Добавьте сервис Zookeeper в кластер ADQM.
2. Установите компонент Zookeeper Server на нечетное количество хостов.
3. Настройте параметры сервиса Zookeeper (см. описания параметров сервиса в соответствующем разделе статьи Конфигурационные параметры).



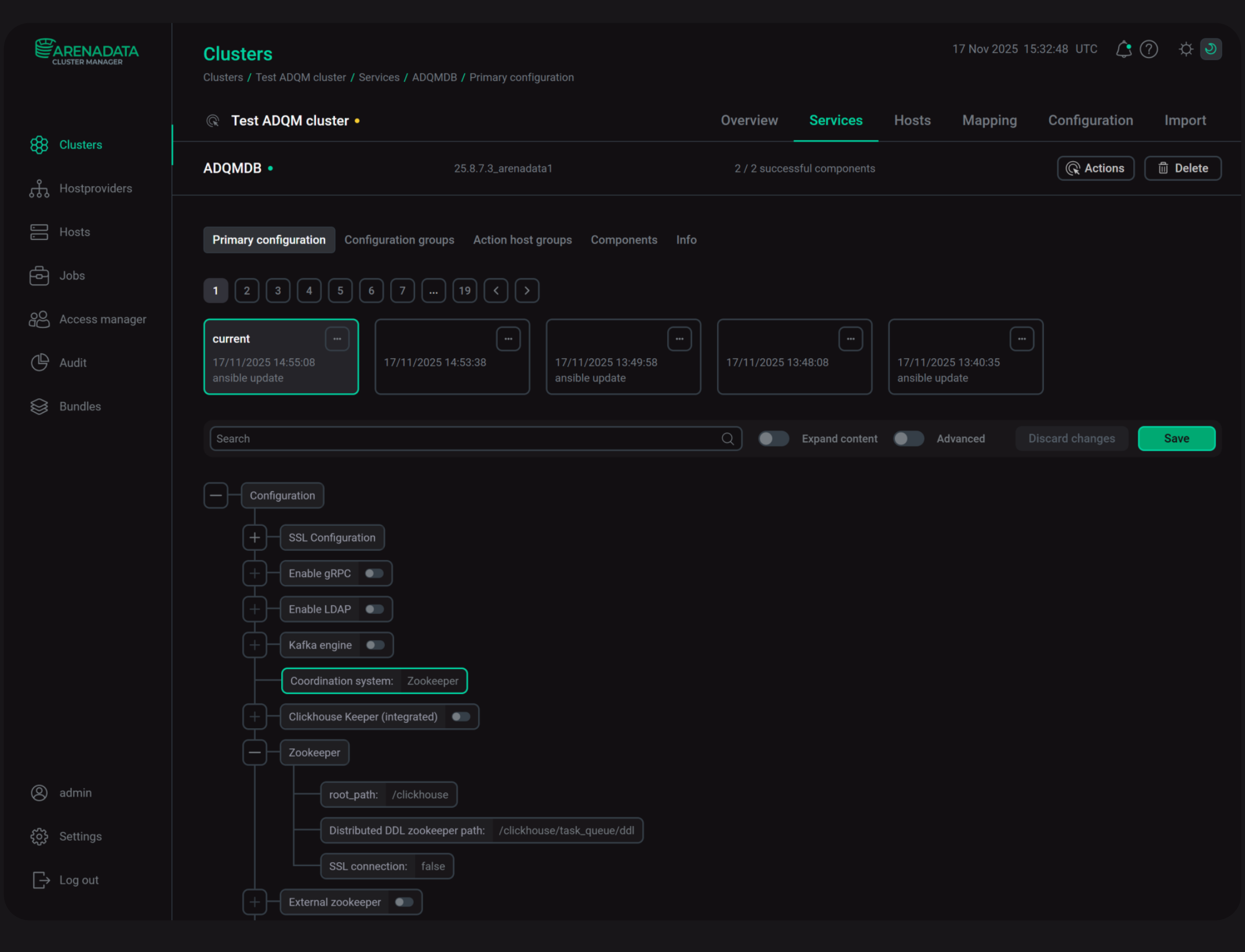
connect: dev-adqm-1.ru-central1.intern <...> -3.ru-central1.internal:2181
dataDir: /var/lib/zookeeper
reconfigEnabled: false

ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / Zookeeper / Primary configuration
17 Nov 2025 15:24:53 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Zookeeper
3.8.3_arenadata2
1 / 1 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
current
Search
Expand content
Advanced
Discard changes
Save
Configuration
Main
zoo.cfg
SSL configuration
Metrics
Myid matching
zookeeper-env.sh

Настройка сервиса Zookeeper

   Нажмите ***Save***, чтобы сохранить настройки.

4. Установите сервис.
5. На странице конфигурации сервиса ADQMDB установите значение `Zookeeper` для параметра ***Coordination system***.



Coordination system: Zookeeper
root_path: /clickhouse
Distributed DDL zookeeper path: /clickhouse/task_queue/ddl
SSL connection: false

ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
17 Nov 2025 15:32:48 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
25.8.7.3_arenadata1
2 / 2 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
current
Search
Expand content
Advanced
Discard changes
Save
Configuration
SSL Configuration
Enable gRPC
Enable LDAP
Kafka engine
Clickhouse Keeper (integrated)
Zookeeper
External zookeeper

Включение сервиса Zookeeper

   В секции ***Zookeeper*** можно настроить пути к узлам ZooKeeper.

6. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы включить использование установленного ZooKeeper в качестве сервиса координации для кластера ADQM.

> **ПРИМЕЧАНИЕ**
>
> - Если в кластере установлен сервис Zookeeper и одновременно включена опция ***External zookeeper***, для репликации данных и распределения запросов будет использоваться указанный внешний кластер ZooKeeper.
> - Чтобы переустановить компонент Zookeeper Server на другие хосты кластера, можно использовать действие сервиса Zookeeper ***Add/Remove components***.
>   После выполнения этого действия необходимо запустить действие ***Reconfig*** для сервиса ADQMDB, чтобы применить изменения.

Пример тестирования реплицируемых и распределенных таблиц

Конфигурирование кластера ADQM с ZooKeeper

1. Сконфигурируйте тестовый кластер — два шарда, в каждом по две реплики (в данном примере используются хосты *host-1*, *host-2*, *host-3* и *host-4*). В статье Конфигурирование логических кластеров в интерфейсе ADCM описано, как это сделать через интерфейс ADCM. В этом случае все необходимые секции (описанные ниже) добавятся в конфигурационный файл каждого сервера автоматически.
   - **Конфигурация логического кластера**
     В файле *config.xml* конфигурация логического кластера описывается в секции `remote_servers`:

```
<remote_servers>
    <default_cluster>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host-2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-3</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host-4</host>
                <port>9000</port>
            </replica>
        </shard>
    </default_cluster>
</remote_servers>
```

     В данном примере будет использоваться реплицируемая таблица ReplicatedMergeTree, поэтому параметр `internal_replication` для каждого шарда устанавливается в `true` — в этом случае репликацию данных будет выполнять реплицируемая таблица (то есть данные будут записываться в любую доступную реплику, другая реплика получит данные автоматически). Если бы использовались обычные таблицы, то параметр `internal_replication` надо было бы установить в `false`, чтобы репликацию данных выполняла таблица Distributed (данные будут записываться на все реплики шарда).

   - **Макросы**
     В настройках каждого сервера должна также присутствовать секция `macros`, в которой определяются идентификаторы шарда и реплики для автоматической подстановки соответствующих хосту значений при создании реплицируемых таблиц на кластере (`ON CLUSTER`). Например, макросы для *host-1* в кластере `default_cluster`:

```
<macros>
    <replica>1</replica>
    <shard>1</shard>
</macros>
```

     Если вы настраиваете логический кластер в интерфейсе ADCM через параметр ***Cluster Configuration*** и указываете имя кластера, например, `abc`, то ADQM автоматически пропишет в *config.xml* макросы следующего вида (для *host-1* в составе кластера `abc`, топология которого аналогична приведенной выше):

```
<macros>
    <abc_replica>1</abc_replica>
    <abc_shard>1</abc_shard>
</macros>
```

     В этом случае при создании реплицируемых таблиц нужно будет использовать переменные `{abc_shard}` и `{abc_replica}` в параметрах ReplicatedMergeTree.

2. Установите сервис Zookeeper на 3 хоста (*host-1*, *host-2*, *host-3*). После этого в конфигурационный файл *config.xml* добавятся секции:
   - `zookeeper` — список узлов ZooKeeper;
   - `distributed_ddl` — путь в ZooKeeper для очереди DDL-запросов (если несколько кластеров используют один и тот же ZooKeeper, этот путь должен быть уникальным для каждого кластера).

```
<zookeeper>
    <node>
        <host>host-1</host>
        <port>2181</port>
    </node>
    <node>
        <host>host-2</host>
        <port>2181</port>
    </node>
    <node>
        <host>host-3</host>
        <port>2181</port>
    </node>
    <root>/clickhouse</root>
</zookeeper>

<distributed_ddl>
    <path>/clickhouse/task_queue/ddl</path>
</distributed_ddl>
```

Реплицируемые таблицы

1. Создайте реплицируемую таблицу, выполнив следующий запрос на одном из хостов (например, на хосте *host-1*):

```
CREATE TABLE test_local ON CLUSTER default_cluster (id Int32, value_string String)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/test_local', '{replica}')
ORDER BY id;
```

   Этот запрос создаст таблицу `test_local` на всех хостах кластера `default_cluster`.

2. Зайдите на хост *host-2* и вставьте данные в таблицу:

```
INSERT INTO test_local VALUES (1, 'a');
```

3. Убедитесь, что данные реплицируются. Для этого выполните выборку данных из таблицы `test_local` на хосте *host-1*:

```
SELECT * FROM test_local;
```

   Если все работает правильно, в таблицу на хосте *host-1* автоматически скопируются данные, которые были записаны в реплику на хосте *host-2*:

```
┌─id─┬─value_string─┐
│  1 │ a            │
└────┴──────────────┘
```

Распределенные таблицы

Обратите внимание, запрос `SELECT` возвращает данные только из таблицы на том хосте, на котором выполняется запрос.

Вставьте данные в любую реплику на второй шард (например, в таблицу на хосте *host-3*):

```
INSERT INTO test_local VALUES (2, 'b');
```

Повторите запрос `SELECT` на хосте *host-1* или *host-2* — данные со второго шарда не попадают в выборку:

```
┌─id─┬─value_string─┐
│  1 │ a            │
└────┴──────────────┘
```

Чтобы получать данные со всех шардов, можно использовать распределенные таблицы (см. подробную информацию в разделе Распределенные таблицы).

1. Создайте таблицу на движке Distributed:

```
CREATE TABLE test_distr ON CLUSTER default_cluster AS default.test_local
ENGINE = Distributed(default_cluster, default, test_local, rand());
```

2. Выполните следующий запрос на любом хосте:

```
SELECT * FROM test_distr;
```

   В результирующую выборку включены данные с обоих шардов кластера:

```
┌─id─┬─value_string─┐
│  1 │ a            │
└────┴──────────────┘
┌─id─┬─value_string─┐
│  2 │ b            │
└────┴──────────────┘
```

ClickHouse Keeper

Елена Дворядкина

Содержание

Назад к содержанию

ClickHouse Keeper как сервис ADQM

Интегрированный ClickHouse Keeper

Пример тестирования реплицируемых и распределенных таблиц

- Конфигурирование кластера ADQM с ClickHouse Keeper
- Реплицируемые таблицы
- Распределенные таблицы

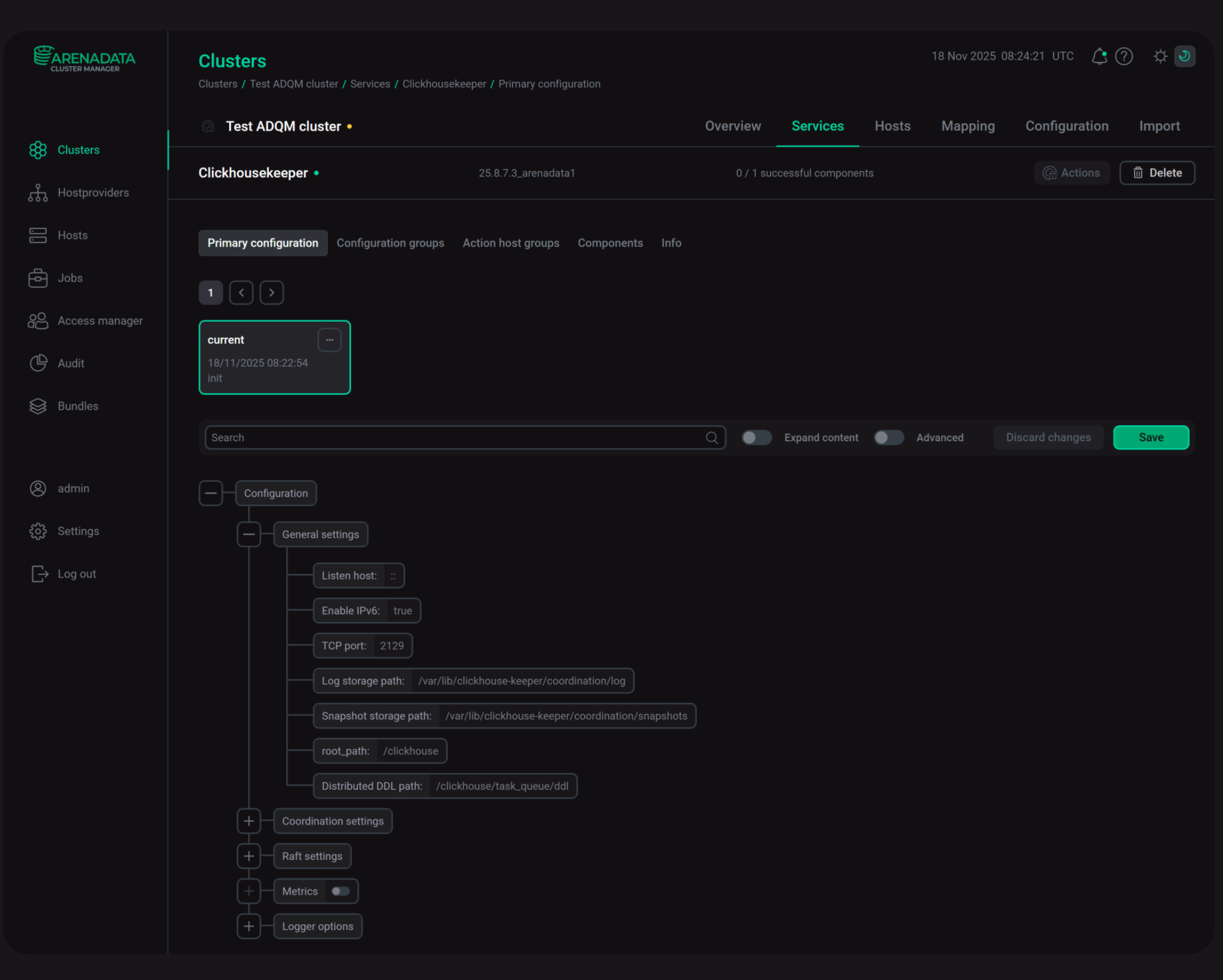
Чтобы использовать ClickHouse Keeper для репликации данных и распределения запросов в ADQM, можно:

- установить ClickHouse Keeper в кластер ADQM как отдельный сервис;
- использовать ClickHouse Keeper, интегрированный в сервер ClickHouse.

ClickHouse Keeper как сервис ADQM

Если необходимо разместить ClickHouse Keeper на выделенных хостах, установите ClickHouse Keeper как отдельный сервис в кластер ADQM. Для этого выполните следующие шаги в интерфейсе ADCM:

1. Добавьте сервис Clickhousekeeper в кластер ADQM.
2. Установите компонент Clickhouse Keeper Server на нечетное количество хостов.
3. Настройте параметры сервиса Clickhousekeeper (см. список и описания доступных параметров сервиса в статье Конфигурационные параметры).



ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters / Test ADQM cluster / Services / Clickhousekeeper / Primary configuration
18 Nov 2025 08:24:21 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
Search
Expand content
Advanced
Discard changes
Save

Clickhousekeeper 25.8.7.3_arenadata1 0 / 1 successful components

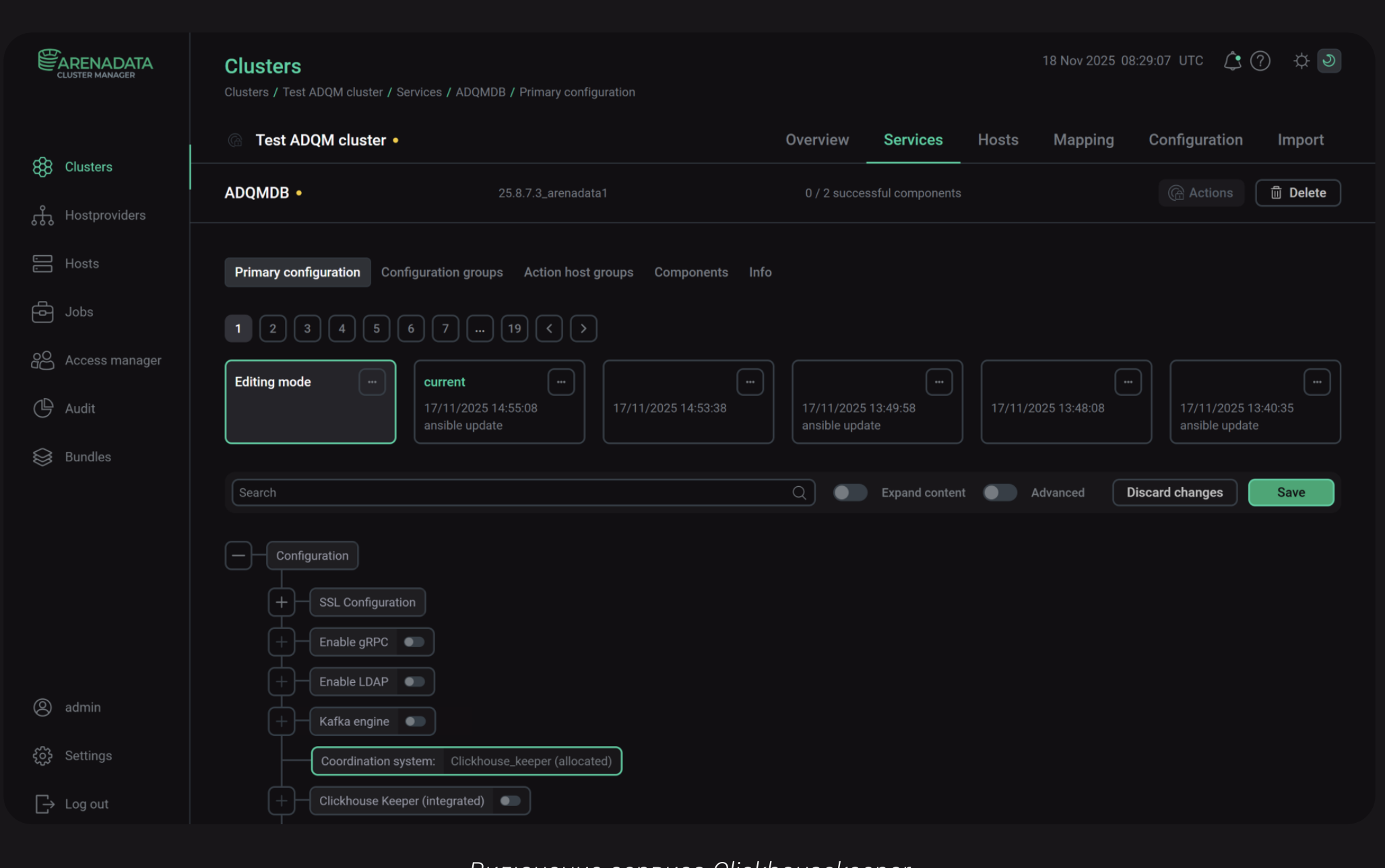
current 18/11/2025 08:22:54 init

Configuration
- General settings
  - Listen host: ::
  - Enable IPv6: true
  - TCP port: 2129
  - Log storage path: /var/lib/clickhouse-keeper/coordination/log
  - Snapshot storage path: /var/lib/clickhouse-keeper/coordination/snapshots
  - root_path: /clickhouse
  - Distributed DDL path: /clickhouse/task_queue/ddl
- Coordination settings
- Raft settings
- Metrics
- Logger options

Конфигурационные параметры сервиса Clickhousekeeper

Нажмите ***Save***, чтобы сохранить настройки.

4. Установите сервис.
   Если осуществляется перевод кластера с ZooKeeper, на этом этапе необходимо вручную перенести данные из ZooKeeper в Clickhouse Keeper. Снепшоты и логи в ClickHouse Keeper имеют несовместимые с ZooKeeper форматы, поэтому данные нужно сначала сконвертировать (см. раздел Migration from ZooKeeper в документации ClickHouse).
5. На странице конфигурации сервиса ADQMDB установите значение `Clickhouse_keeper (allocated)` для параметра ***Coordination system***.



ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
18 Nov 2025 08:29:07 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
Search
Expand content
Advanced
Discard changes
Save

ADQMDB 25.8.7.3_arenadata1 0 / 2 successful components

Editing mode | current 17/11/2025 14:55:08 ansible update | 17/11/2025 14:53:38 | 17/11/2025 13:49:58 ansible update | 17/11/2025 13:48:08 | 17/11/2025 13:40:35 ansible update

Configuration
- SSL Configuration
- Enable gRPC
- Enable LDAP
- Kafka engine
- Coordination system: Clickhouse_keeper (allocated)
- Clickhouse Keeper (integrated)

Включение сервиса Clickhousekeeper

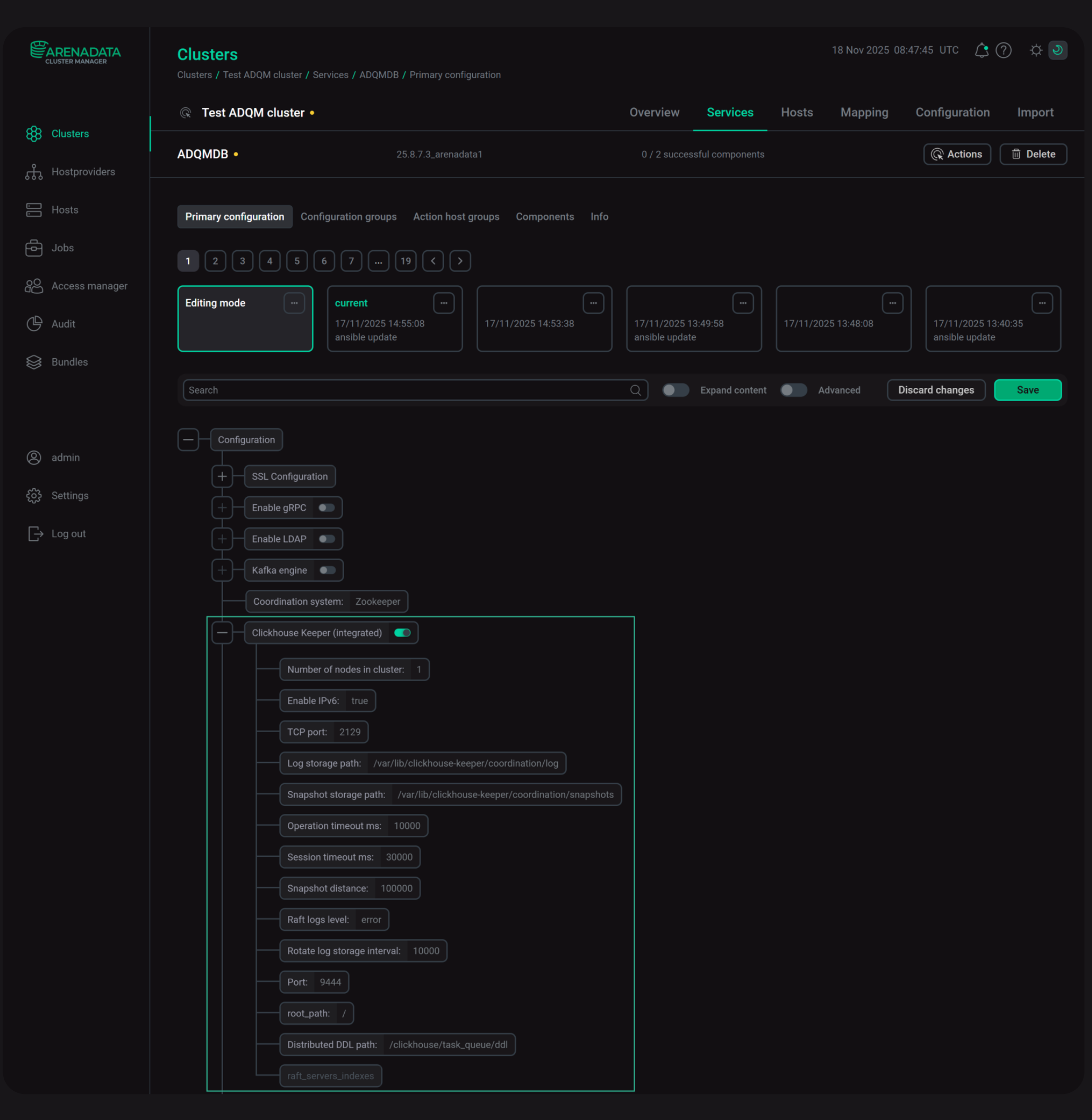
6. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы включить использование установленного ClickHouse Keeper в качестве сервиса координации для кластера ADQM.

> **ПРИМЕЧАНИЕ**
>
> Чтобы переустановить ClickHouse Keeper на другие хосты кластера, можно использовать действие сервиса Clickhousekeeper ***Add/Remove components***. После выполнения этого действия необходимо запустить действие ***Reconfig*** для сервиса ADQMDB, чтобы применить изменения.

Интегрированный ClickHouse Keeper

Если нет требования контролировать на каких хостах установлен ClickHouse Keeper (например, в тестовых или небольших кластерах), можно использовать внутренний ClickHouse Keeper — ADQM автоматически его сконфигурирует и развернет на хостах, где установлен компонент Clickhouse Server. Для этого в интерфейсе ADCM:

1. Активируйте ***Clickhouse Keeper (integrated)*** на странице конфигурации сервиса ADQMDB. Настройки в развернувшейся секции будут заполнены автоматически на основе параметров сервера ClickHouse. При необходимости значения параметров можно изменить (см. описания параметров в статье Конфигурационные параметры).



ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
18 Nov 2025 08:47:45 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
Search
Expand content
Advanced
Discard changes
Save

ADQMDB 25.8.7.3_arenadata1 0 / 2 successful components

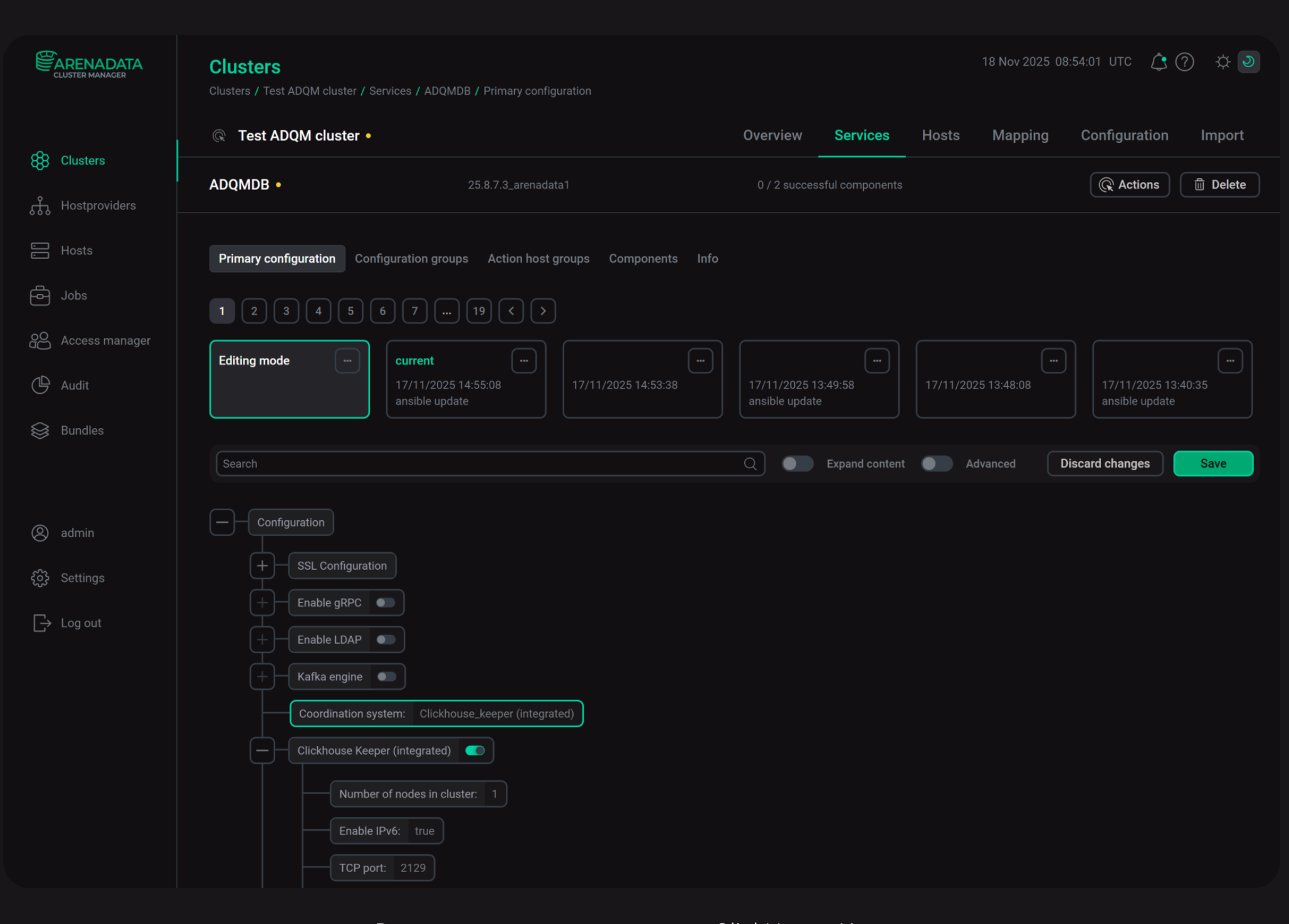
Configuration
- SSL Configuration
- Enable gRPC
- Enable LDAP
- Kafka engine
- Coordination system: Zookeeper
- Clickhouse Keeper (integrated)
  - Number of nodes in cluster: 1
  - Enable IPv6: true
  - TCP port: 2129
  - Log storage path: /var/lib/clickhouse-keeper/coordination/log
  - Snapshot storage path: /var/lib/clickhouse-keeper/coordination/snapshots
  - Operation timeout ms: 10000
  - Session timeout ms: 30000
  - Snapshot distance: 100000
  - Raft logs level: error
  - Rotate log storage interval: 10000
  - Port: 9444
  - root_path: /
  - Distributed DDL path: /clickhouse/task_queue/ddl
  - raft_servers_indexes

Конфигурационные параметры интегрированного ClickHouse Keeper

Если осуществляется переход с ZooKeeper, далее необходимо выполнить следующие шаги:

- Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы сконфигурировать и развернуть внутренний ClickHouse Keeper.
- Сконвертируйте и перенесите данные из ZooKeeper в ClickHouse Keeper.

2. На странице конфигурации сервиса ADQMDB установите значение `Clickhouse_keeper (integrated)` для параметра ***Coordination system***.



ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
18 Nov 2025 08:54:01 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
Search
Expand content
Advanced
Discard changes
Save

ADQMDB 25.8.7.3_arenadata1 0 / 2 successful components

Configuration
- SSL Configuration
- Enable gRPC
- Enable LDAP
- Kafka engine
- Coordination system: Clickhouse_keeper (integrated)
- Clickhouse Keeper (integrated)
  - Number of nodes in cluster: 1
  - Enable IPv6: true
  - TCP port: 2129

Включение интегрированного ClickHouse Keeper

Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы включить использование внутреннего ClickHouse Keeper в качестве сервиса координации для кластера ADQM.

Пример тестирования реплицируемых и распределенных таблиц

Конфигурирование кластера ADQM с ClickHouse Keeper

1. Сконфигурируйте тестовый кластер — два шарда, в каждом по две реплики (в данном примере используются хосты *host-1*, *host-2*, *host-3* и *host-4*). В статье Конфигурирование логических кластеров в интерфейсе ADCM описано, как это сделать через интерфейс ADCM. В этом случае все необходимые секции (описанные ниже) добавятся в конфигурационный файл каждого сервера автоматически.

   - **Конфигурация логического кластера**
     В файле *config.xml* конфигурация логического кластера описывается в секции `remote_servers`:

```
<remote_servers>
    <default_cluster>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-1</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host-2</host>
                <port>9000</port>
            </replica>
        </shard>
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>host-3</host>
                <port>9000</port>
            </replica>
            <replica>
                <host>host-4</host>
                <port>9000</port>
            </replica>
        </shard>
    </default_cluster>
</remote_servers>
```

     В данном примере будет использоваться реплицируемая таблица ReplicatedMergeTree, поэтому параметр `internal_replication` для каждого шарда устанавливается в `true` — в этом случае репликацию данных будет выполнять реплицируемая таблица (то есть данные будут записываться в любую доступную реплику, другая реплика получит данные автоматически). Если бы использовались обычные таблицы, то параметр `internal_replication` надо было бы установить в `false`, чтобы репликацию данных выполняла таблица Distributed (данные будут записываться на все реплики шарда).

   - **Макросы**
     В настройках каждого сервера должна также присутствовать секция `macros`, в которой определяются идентификаторы шарда и реплики для автоматической подстановки соответствующих хосту значений при создании реплицируемых таблиц на кластере (`ON CLUSTER`). Например, макросы для *host-1* в кластере `default_cluster`:

```
<macros>
    <replica>1</replica>
    <shard>1</shard>
</macros>
```

     Если вы настраиваете логический кластер в интерфейсе ADCM через параметр ***Cluster Configuration*** и указываете имя кластера, например, `abc`, то ADQM автоматически пропишет в *config.xml* макросы следующего вида (для *host-1* в составе кластера `abc`, топология которого аналогична приведенной выше):

```
<macros>
    <abc_replica>1</abc_replica>
    <abc_shard>1</abc_shard>
</macros>
```

     В этом случае при создании реплицируемых таблиц нужно будет использовать переменные `{abc_shard}` и `{abc_replica}` в параметрах ReplicatedMergeTree.

2. Установите сервис Clickhousekeeper на 3 хоста (*host-1*, *host-2*, *host-3*). После этого в конфигурационный файл *config.xml* добавятся секции:

   - `zookeeper` — список узлов ClickHouse Keeper;
   - `distributed_ddl` — путь в ClickHouse Keeper для очереди DDL-запросов (если несколько кластеров используют один и тот же ClickHouse Keeper, этот путь должен быть уникальным для каждого кластера).

```
<zookeeper>
    <node>
        <host>host-1</host>
        <port>2129</port>
    </node>
    <node>
        <host>host-2</host>
        <port>2129</port>
    </node>
    <node>
        <host>host-3</host>
        <port>2129</port>
    </node>
    <root>/clickhouse</root>
</zookeeper>

<distributed_ddl>
    <path>/clickhouse/task_queue/ddl</path>
</distributed_ddl>
```

Реплицируемые таблицы

1. Создайте реплицируемую таблицу, выполнив следующий запрос на одном из хостов (например, на хосте *host-1*):

```
CREATE TABLE test_local ON CLUSTER default_cluster (id Int32, value_string String)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/test_local', '{replica}')
ORDER BY id;
```

   Этот запрос создаст таблицу `test_local` на всех хостах кластера `default_cluster`.

2. Зайдите на хост *host-2* и вставьте данные в таблицу:

```
INSERT INTO test_local VALUES (1, 'a');
```

3. Убедитесь, что данные реплицируются. Для этого выполните выборку данных из таблицы `test_local` на хосте *host-1*:

```
SELECT * FROM test_local;
```

   Если все работает правильно, в таблицу на хосте *host-1* автоматически скопируются данные, которые были записаны в реплику на хосте *host-2*:

```
┌─id─┬─value_string─┐
│  1 │ a            │
└────┴──────────────┘
```

Распределенные таблицы

Обратите внимание, запрос `SELECT` возвращает данные только из таблицы на том хосте, на котором выполняется запрос.

Вставьте данные в любую реплику на второй шард (например, в таблицу на хосте *host-3*):

```
INSERT INTO test_local VALUES (2, 'b');
```

Повторите запрос `SELECT` на хосте *host-1* или *host-2* — данные со второго шарда не попадают в выборку:

```
┌─id─┬─value_string─┐
│  1 │ a            │
└────┴──────────────┘
```

Чтобы получать данные со всех шардов, можно использовать распределенные таблицы (см. подробную информацию в разделе Распределенные таблицы).

1. Создайте таблицу на движке Distributed:

```
CREATE TABLE test_distr ON CLUSTER default_cluster AS default.test_local
ENGINE = Distributed(default_cluster, default, test_local, rand());
```

2. Выполните следующий запрос на любом хосте:

```
SELECT * FROM test_distr;
```

   В результирующую выборку включены данные с обоих шардов кластера:

```
┌─id─┬─value_string─┐
│  1 │ a            │
└────┴──────────────┘
┌─id─┬─value_string─┐
│  2 │ b            │
└────┴──────────────┘
```

Примеры использования clickhouse-copier

Елена Дворядкина

Содержание

Назад к содержанию

- Использование clickhouse-copier
  - Шаг 1. Настройка соединения с сервисом координации
  - Шаг 2. Конфигурация заданий на копирование
  - Шаг 3. Запуск clickhouse-copier
- Пример 1. Шардирование таблицы
  - Исходные данные
  - Копирование
  - Проверка
- Пример 2. Перешардирование
  - Исходные данные
  - Копирование
  - Проверка

При установке ADQM на все хосты с сервисом ADQMDB по умолчанию устанавливается пакет adqm-clickhouse-copier, который предоставляет функциональность clickhouse-copier, позволяющую копировать между кластерами разных топологий и перешардировать большие объемы данных. В этой статье описывается как создавать задания на копирование и как запускать clickhouse-copier для их выполнения, а также приводятся примеры использования этого инструмента.

ВНИМАНИЕ

Clickhouse-copier не работает с типами данных Variant, Dynamic и JSON.

Использование clickhouse-copier

Чтобы скопировать/перешардировать данные с помощью clickhouse-copier, необходимо выполнить следующие шаги.

Шаг 1. Настройка соединения с сервисом координации

Для выполнения одного и того же задания можно запустить несколько экземпляров clickhouse-copier на разных хостах. Чтобы синхронизировать процесс копирования между хостами, требуется сервис координации — ZooKeeper или ClickHouse Keeper. Создайте XML-файл с параметрами подключения к ZooKeeper/ClickHouse Keeper — скопируйте в него секцию `zookeeper` из конфигурационного файла *config.xml*. Опционально в него также можно добавить секцию `logger` с настройками логирования. Скопируйте этот файл на все хосты, где будет запускаться clickhouse-copier.

Пример конфигурации параметров соединения с кластером ZooKeeper

Файл *keeper.xml*, который используется в примерах данной статьи:

```
<clickhouse>
    <zookeeper>
        <node>
            <host>dev-adqm1.ru-central1.internal</host>
            <port>2181</port>
        </node>
        <node>
            <host>dev-adqm2.ru-central1.internal</host>
            <port>2181</port>
        </node>
        <node>
            <host>dev-adqm3.ru-central1.internal</host>
            <port>2181</port>
        </node>
        <root>/clickhouse</root>
    </zookeeper>
</clickhouse>
```

Шаг 2. Конфигурация заданий на копирование

Создайте XML-файл с описанием заданий на копирование для clickhouse-copier (например, *task.xml*).

Структура конфигурации в общем виде

```
<clickhouse>
    <remote_servers>
        <source_cluster>
            <shard>
                    <replica>
                        ...
                    </replica>
                    ...
            </shard>
            ...
        </source_cluster>

        <destination_cluster>
            ...
        </destination_cluster>
    </remote_servers>

    <max_workers>2</max_workers>

    <settings_pull>
        <readonly>1</readonly>
    </settings_pull>

    <settings_push>
        <readonly>0</readonly>
    </settings_push>

    <settings>
        <connect_timeout>3</connect_timeout>
        <distributed_foreground_insert>1</distributed_foreground_insert>
    </settings>

    <tables>
        <!-- Задание на копирование одной таблицы -->
        <table_1>
            <cluster_pull>source_cluster</cluster_pull>
            <database_pull>source_database</database_pull>
            <table_pull>test_table</table_pull>

            <cluster_push>destination_cluster</cluster_push>
            <database_push>destination_database</database_push>
            <table_push>test_table_copied</table_push>

            <engine>
            ENGINE=ReplicatedMergeTree('/clickhouse/tables/{shard}/test_table_copied', '{replica}')
            PARTITION BY partition_key
            ORDER BY sorting_key
            </engine>

            <sharding_key>sharding_key_expr</sharding_key>

            <where_condition>where_condition_expr</where_condition>

            <enabled_partitions>
                <partition>'2024-02-01'</partition>
                <partition>'2024-03-02'</partition>
                ...
            </enabled_partitions>
        </table_1>

        <!-- Задание на копирование следующей таблицы -->
        <table_2>
            ...
        </table_2>
        ...
    </tables>
</clickhouse>
```

Параметры конфигурации

Параметр	Описание
remote_servers	Описание исходного кластера и кластера назначения из секции `remote_servers` конфигурационного файла *config.xml* (`source_cluster` и `destination_cluster` в приведенной выше конфигурации — названия этих кластеров). Для объединения хостов ADQM в логический кластер можно использовать ADCM, при этом описание кластера автоматически добавится в файл *config.xml* — подробнее в статье Конфигурирование логических кластеров в интерфейсе ADCM
max_workers	Максимальное количество экземпляров clickhouse-copier (workers), одновременно выполняющих задание на копирование. Если запустить больше экземпляров, лишние будут переведены в спящий режим
settings_pull, settings_push	Настройки, используемые для выборки данных (pull) из таблиц исходного кластера и вставки данных (push) в таблицы кластера назначения соответственно
settings	Общие настройки, используемые для операций выборки и вставки данных (перекрываются настройками `settings_pull` и `settings_push` соответственно)
tables	Описание заданий на копирование. Можно указать несколько заданий в одном узле ZooKeeper/ClickHouse Keeper — каждое задание для копирования одной таблицы (например, в приведенной выше конфигурации указано два задания — `table_1` и `table_2`). Задания будут выполняться последовательно, то есть копирование следующей таблицы не начнется, пока не будет скопирована предыдущая. Названия заданий в конфигурации могут быть произвольными. Для каждого задания нужно указать следующие параметры: ▪ `cluster_pull`, `database_pull`, `table_pull` — исходный кластер (из секции `remote_servers`), база данных в нем и таблицы, которые необходимо скопировать; ▪ `cluster_push`, `database_push`, `table_push` — кластер назначения (из секции `remote_servers`), база данных в нем и таблицы, в которые нужно вставить данные; ▪ `engine` — движок таблицы назначения (если таблица в кластере назначения не найдена, она будет создана на основе описаний столбцов исходной таблицы и табличного движка, указанного в этом параметре); ▪ `sharding_key` — ключ шардирования, который будет применяться при вставке данных в кластер назначения; ▪ `where_condition` — необязательное выражение, по которому будут фильтроваться данные при их выборке с хостов исходного кластера; ▪ `enabled_partitions` — партиции, которые должны быть скопированы (остальные партиции будут игнорироваться). Имена партиций должны иметь то же формат, что и столбец `partition` таблицы system.parts (текст в кавычках). Поскольку ключи партиционирования таблиц исходного кластера и кластера назначения могут быть разными, эти имена партиций определяют партиции в кластере назначения. Эта настройка является необязательной (если не указана, будут скопированы все партиции), но настоятельно рекомендуется указывать партиции явно. Если в кластере назначения уже есть некоторые готовые партиции, они будут удалены в начале копирования, так как будут считаться неполными данными от предыдущего копирования.

После завершения описания заданий на копирование загрузите его в ZooKeeper/ClickHouse Keeper — создайте znode (должен располагаться по определенному пути — *<znode-path-to-task>/description*) и сохраните в нем содержимое файла *task.xml*. Например, это можно сделать с помощью консоли ZooKeeper CLI, которая доступна на каждом хосте кластера ZooKeeper. Откройте консоль с помощью скрипта *zkCli.sh* и используйте команду `create`:

```
$ create /clickhouse/clickhouse ""
```

```
$ create /clickhouse/clickhouse/copier ""
```

```
$ create /clickhouse/clickhouse/copier/task ""
```

```
$ create /clickhouse/clickhouse/copier/task/description "$(cat <path-to>/task.xml)"
```

где `<path-to>` — путь к файлу *task.xml*.

Шаг 3. Запуск clickhouse-copier

Чтобы запустить экземпляр clickhouse-copier для выполнения задания на копирование, выполните следующую команду:

```
$ clickhouse-copier --config keeper.xml --task-path /clickhouse/copier/task
```

где:

- `config` — путь к файлу *keeper.xml* с параметрами подключения к ZooKeeper/ClickHouse Keeper;
- `task-path` — путь к znode, который используется для синхронизации процессов clickhouse-copier и хранения заданий на копирование данных.

Для этой команды также доступны следующие необязательные параметры:

- `daemon` — запустить clickhouse-copier в режиме демона;
- `task-file` — путь к файлу с конфигурацией задания для первоначальной загрузки в ZooKeeper/ClickHouse Keeper;
- `task-upload-force` — принудительно загрузить `task-file`, даже если znode уже существует (по умолчанию — `false`);
- `base-dir` — путь к логам и вспомогательным файлам на хосте, где запущен clickhouse-copier (если этот параметр не указан, подкаталоги для этих файлов создаются в каталоге, из которого был запущен clickhouse-copier).

Чтобы уменьшить сетевой трафик, рекомендуется запускать clickhouse-copier на том же хосте, где расположены исходные данные.

Пример 1. Шардирование таблицы

В этом примере показывается как шардировать таблицу — данные таблицы, реплицируемой на два хоста в рамках одного шарда, копируются в другой кластер и распределяются между двумя шардами, в каждом из которых две реплики.

Исходные данные

Создайте реплицируемую таблицу `data_table1` (в базе данных `default`) в кластере `source_cluster1`, который включает один шард с двумя репликами (описание кластера приведено ниже):

```
CREATE TABLE data_table1 ON CLUSTER source_cluster1 (value UInt64)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/data_table1', '{replica}') ORDER BY value;
```

Сгенерируйте тестовые данные для таблицы (1 миллион строк):

```
INSERT INTO data_table1 SELECT number FROM numbers(1000000);
```

В примерах данной статьи кластер назначения называется `default_cluster` — пример логического кластера, который автоматически формируется при установке ADQM на четыре хоста с фактором репликации равным `2`. Создайте в этом кластере новую базу данных `test_database`, куда clickhouse-copier будет копировать данные:

```
CREATE DATABASE test_database ON CLUSTER default_cluster;
```

Копирование

Создайте конфигурационный файл *task1.xml*, описывающий для clickhouse-copier задание копирования таблицы `data_table1` кластера `source_cluster1` в таблицу `data_table1_copied` кластера `default_cluster`.

task1.xml

```
<clickhouse>
    <remote_servers>
        <source_cluster1>
            <shard>
                <internal_replication>false</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm1.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
                <replica>
                    <host>dev-adqm2.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
        </source_cluster1>
        <default_cluster>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm1.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
                <replica>
                    <host>dev-adqm2.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm3.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
                <replica>
                    <host>dev-adqm4.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
        </default_cluster>
    </remote_servers>

    <max_workers>2</max_workers>

    <settings_pull>
        <readonly>1</readonly>
    </settings_pull>

    <settings_push>
        <readonly>0</readonly>
    </settings_push>

    <settings>
        <connect_timeout>3</connect_timeout>
        <distributed_foreground_insert>1</distributed_foreground_insert>
    </settings>

    <tables>
        <table_local>
            <cluster_pull>source_cluster1</cluster_pull>
            <database_pull>default</database_pull>
            <table_pull>data_table1</table_pull>

            <cluster_push>default_cluster</cluster_push>
            <database_push>test_database</database_push>
            <table_push>data_table1_copied</table_push>

            <engine>
            ENGINE=ReplicatedMergeTree('/clickhouse/tables/{shard}/data_table1_copied', '{replica}')
            ORDER BY (value)
            </engine>

            <sharding_key>rand()</sharding_key>
        </table_local>
    </tables>
</clickhouse>
```

Загрузите конфигурацию *task1.xml* в znode `/clickhouse/clickhouse/copier/task1/description` кластера ZooKeeper, который указан в *keeper.xml*.

Из каталога, где находится файл *keeper.xml*, запустите clickhouse-copier:

```
$ clickhouse-copier --config keeper.xml --task-path /clickhouse/copier/task1
```

Проверка

В результате копирования в кластере `default_cluster` на всех хостах в базе данных `test_database` будет создана реплицированная таблица. Это можно проверить с помощью запроса:

```
SELECT create_table_query FROM system.tables where name = 'data_table1_copied' and database ='test_database';
```

```
┌─create_table_query──────────────────────────────────────────────────────────────────────────────
─────────────┐
│ CREATE TABLE test_database.data_table1_copied (`value` UInt64)
│
│ ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/data_table1_copied', '{replica}') ORDER BY value SETTINGS
index_granularity = 8192   │
└─────────────────────────────────────────────────────────────────────────────────────────────────
─────────────┘
```

Данные распределены примерно поровну между двумя шардами кластера `default_cluster` — в таблице ниже приведены запросы для проверки, которые можно выполнить на хостах каждого шарда.

Запрос	Шард 1	Шард 2

`SELECT count() FROM test_database.data_table1_copied;`	`┌─count()─┐`
	`│  499045 │`
	`└─────────┘`
		`┌─count()─┐`
		`│  500955 │`
		`└─────────┘`

`SELECT * FROM test_database.data_table1_copied ORDER BY value ASC;`	`┌─value─┐`
	`│     2 │`
	`│     4 │`
	`│     5 │`
	`│     7 │`
	`│     9 │`
	`│    10 │`
	`│   ... │`
		`┌─value─┐`
		`│     0 │`
		`│     1 │`
		`│     3 │`
		`│     6 │`
		`│     8 │`
		`│    11 │`
		`│   ... │`

Чтобы отправлять запросы `SELECT` на все шарды сразу, можно использовать распределенную таблицу:

```
CREATE TABLE test_database.distr_table1 ON CLUSTER default_cluster AS test_database.data_table1_copied
ENGINE = Distributed(default_cluster, test_database, data_table1_copied, rand());
```

```
SELECT * FROM test_database.distr_table1 WHERE value < 10;
```

```
┌─value─┐
│     4 │
└───────┘
┌─value─┐
│     7 │
└───────┘
┌─value─┐
│     2 │
│     5 │
│     9 │
└───────┘
┌─value─┐
│     1 │
└───────┘
┌─value─┐
│     6 │
│     8 │
└───────┘
┌─value─┐
│     0 │
│     3 │
└───────┘
```

Пример 2. Перешардирование

В этом примере показывается как с помощью clickhouse-copier перешардировать данные — скопировать данные, распределенные по трем шардам кластера `source_cluster2` (в каждом шарде одна реплика), в кластер `default_cluster` с двумя шардами (в каждом шарде две реплики).

Исходные данные

Создайте таблицу `data_table2` в кластере `source_cluster2` (в базе данных `default`):

```
CREATE TABLE data_table2 ON CLUSTER source_cluster2 (value UInt64) ENGINE = MergeTree() ORDER BY value;
```

Создайте распределенную таблицу, которая будет обращаться к таблице `data_table2` и позволит выполнять запросы распределенно на нескольких серверах:

```
CREATE TABLE distr_table2 ON CLUSTER source_cluster2 AS default.data_table2
ENGINE = Distributed(source_cluster2, default, data_table2, rand());
```

Вставьте данные, направив запрос `INSERT` в распределенную таблицу, которая распределит данные по шардам случайным образом:

```
INSERT INTO distr_table2 SELECT number FROM numbers(1000000);
```

Примерное распределение данных по шардам приведено в следующей таблице.

Запрос	Шард 1	Шард 2	Шард 3

`SELECT count() FROM data_table2;`	`┌─count()─┐`
	`│  332973 │`
	`└─────────┘`
		`┌─count()─┐`
		`│  334408 │`
		`└─────────┘`
			`┌─count()─┐`
			`│  332619 │`
			`└─────────┘`

Копирование

Составьте XML-конфигурацию задания на копирование данных для clickhouse-copier (файл *task2.xml*).

task2.xml

```
<clickhouse>
    <remote_servers>
        <source_cluster2>
            <shard>
                <internal_replication>false</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm1.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
            <shard>
                <internal_replication>false</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm2.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
            <shard>
                <internal_replication>false</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm3.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
        </source_cluster2>
        <default_cluster>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm1.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
                <replica>
                    <host>dev-adqm2.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>dev-adqm3.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
                <replica>
                    <host>dev-adqm4.ru-central1.internal</host>
                    <port>9000</port>
                </replica>
            </shard>
        </default_cluster>
    </remote_servers>

    <max_workers>2</max_workers>

    <settings_pull>
        <readonly>1</readonly>
    </settings_pull>

    <settings_push>
        <readonly>0</readonly>
    </settings_push>

    <settings>
        <connect_timeout>3</connect_timeout>
        <distributed_foreground_insert>1</distributed_foreground_insert>
    </settings>

    <tables>
        <table_local>
            <cluster_pull>source_cluster2</cluster_pull>
            <database_pull>default</database_pull>
            <table_pull>data_table2</table_pull>

            <cluster_push>default_cluster</cluster_push>
            <database_push>test_database</database_push>
            <table_push>data_table2_copied</table_push>

            <engine>
            ENGINE=ReplicatedMergeTree('/clickhouse/tables/{shard}/data_table2_copied', '{replica}')
            ORDER BY (value)
            </engine>

            <sharding_key>rand()</sharding_key>
        </table_local>
    </tables>
</clickhouse>
```

Загрузите конфигурацию *task2.xml* в znode `/clickhouse/clickhouse/copier/task2/description`.

Запустите clickhouse-copier из каталога, где хранится *keeper.xml*:

```
$ clickhouse-copier --config keeper.xml --task-path /clickhouse/copier/task2
```

Проверка

Проверьте, что в базу данных `test_database` добавлена таблица `data_table2_copied`:

```
SHOW tables FROM test_database;
```

```
┌─name───────────────┐
│ data_table1_copied │
│ data_table2_copied │
└────────────────────┘
```

Посмотрите, как 1 миллион строк данных из исходной таблицы распределился по шардам кластера назначения.

Запрос	Шард 1	Шард 2

`SELECT count() FROM test_database.data_table2_copied;`	`┌─count()─┐`
	`│  500171 │`
	`└─────────┘`
		`┌─count()─┐`
		`│  499829 │`
		`└─────────┘`

`SELECT * FROM test_database.data_table2_copied limit 10;`	`┌─value─┐`
	`│     3 │`
	`│     5 │`
	`│     7 │`
	`│     8 │`
	`│    11 │`
	`│    13 │`
	`│    14 │`
	`│    18 │`
	`│    20 │`
	`│    22 │`
	`└───────┘`
		`┌─value─┐`
		`│     0 │`
		`│     1 │`
		`│     2 │`
		`│     4 │`
		`│     6 │`
		`│     9 │`
		`│    10 │`
		`│    12 │`
		`│    15 │`
		`│    16 │`
		`└───────┘`

Настройка хранилищ данных

Елена Дворядкина

Содержание

Назад к содержанию

Обзор

Диски

Политики хранения

Назначение таблице политики хранения

Пример конфигурирования многоуровневого хранилища

ADQM позволяет хранить табличные данные на нескольких дисках различных типов, группировать их в тома и автоматически перемещать данные между томами в соответствии с указанными условиями.

Обзор

Поддерживаются следующие варианты хранения данных:

- **В локальной файловой системе**. Возможно хранение данных на нескольких блочных устройствах — например, это может быть полезно при разделении данных на "горячие" (свежие данные, которые занимают малый объем и регулярно запрашиваются) и "холодные" (архивные данные, которые редко запрашиваются). В этом случае, "горячая" часть данных может быть размещена на быстрых локальных дисках (например, на NVMe SSD или в памяти), а "холодная" — на более медленных (например, HDD). Возможность добавления новых дисков также позволяет упростить процесс расширения хранилища.
- **На внешних дисках S3**. Обеспечивается возможность хранить огромные объемы информации с автоматической поддержкой отказоустойчивости, надежности и расширяемости хранилища.

Поверх дисков, на которых хранятся данные, можно создавать **кеш** (чтобы повысить скорость выборки данных с внешних дисков с высокой латентностью) и **зашифрованные диски** (чтобы зашифровать конфиденциальные данные, хранящиеся в кластере ADQM).

Основные термины, используемые при конфигурировании хранилища данных в ADQM:

- **Диск** (**disk**) — примонтированное в файловой системе дополнительное блочное устройство (локальный диск), внешнее хранилище S3, кеширующий или зашифрованный диск.
- **Том** (**volume**) — упорядоченный набор равноценных дисков.
- **Политика хранения** (**storage policy**) — набор томов с правилами записи данных таблицы.

Общая схема работы с хранилищами:

1. Создание и конфигурирование дисков.
2. Создание политики хранения с указанием дисков (в составе томов), на которых будут храниться данные.
3. Назначение таблице политики хранения.

Диски, тома и политики хранения настраиваются через конфигурационные параметры сервиса ADQMDB (см. соответствующие разделы ниже). Назначенные параметрам значения сохраняются внутри тега `<storage_configuration>` в файле */etc/clickhouse-server/config.d/storage.xml*.

Диски

Управление дисками в ADQM можно осуществлять через интерфейс ADCM:

1. Перейдите к окну конфигурирования сервиса ADQMDB (***ADQM cluster*** → ***Services*** → ***ADQMDB*** → ***Primary Configuration***).
2. Включите опцию ***Show advanced***.
3. Активируйте группу, соответствующую типу диска:
   - ***Enable additional local storage*** — локальные диски;
   - ***Enable S3 storage*** — диски S3;
   - ***Enable cache*** — кеширующие диски;
   - ***Enable encrypted storage*** — зашифрованные диски.
4. В развернувшейся секции можно добавлять/удалять и настраивать диски соответствующего типа. Ниже описаны параметры для каждого типа дисков.

Локальные диски

Параметр	Описание
disk_name	Уникальное имя диска
disk_path	Абсолютный путь к директории, в которую смонтировано блочное устройство. При указании относительного пути (то есть без начального `/`) возникает ошибка, информирующая о том, что директория не существует
keep_free_space_bytes	Размер зарезервированного свободного места на диске (в байтах)

Если опция ***Recursively create directories*** активирована, ADQM автоматически создаст указанные директории для локальных дисков (если эти директории не существуют) и назначит им владельца `clickhouse:clickhouse` с маской прав доступа `0770`.

Диски S3

Параметр	Описание
disk_name	Уникальное имя диска
endpoint	URL-адрес бакета с указанием пути к корневой директории на сервере, где хранятся данные
access_key_id	Идентификатор ключа доступа к S3
secret_access_key	Секретный ключ доступа к S3
proxy	Список прокси-серверов
custom_parameters	Другие параметры (полный список доступных для дисков S3 параметров описан в разделе Configuring the S3 disk документации ClickHouse)

Кеширующие диски

Параметр	Описание
parent_disk_name	Имя родительского диска (внешнего диска S3), поверх которого создается кеширующий
cache_disk_name	Уникальное имя кешируещего диска
cache_disk_path	Абсолютный путь к директории, в которую будет записываться кеш
max_size_bytes	Максимальный размер кеша (в байтах)
cache_on_write_operations	Включает кеширование данных при любых операциях записи (запросы `INSERT`, фоновые слияния)
enable_filesystem_query_cache_limit	Позволяет ограничить размер кеша, загружаемого в рамках каждого запроса (зависит от пользовательской настройки `max_query_cache_size`)
enable_cache_hits_threshold	Число, определяющее сколько раз данные должны быть считаны, прежде чем они будут закешированы
do_not_evict_index_and_mark_files	Позволяет не исключать небольшие часто используемые файлы в соответствии с политикой кеширования
max_file_segment_size_bytes	Максимальный размер одного кеш-файла (в байтах)
max_elements	Ограничение на количество кеш-файлов

Если опция ***Recursively create directories*** активирована, ADQM автоматически создаст указанные директории для хранения кеша (если эти директории не существуют) и назначит им владельца `clickhouse:clickhouse` с маской прав доступа `0770`.

Зашифрованные диски

Параметр	Описание
parent_disk_name	Имя родительского диска, поверх которого создается зашифрованный
encrypted_disk_name	Уникальное имя зашифрованного диска
encrypted_disk_path	Абсолютный или относительный путь к директории на диске, где будут сохранены данные в зашифрованном виде. При указании относительного пути создается поддиректория на родительском диске
encryption_algorithm	Алгоритм шифрования данных. Возможные значения: `AES_128_CTR`, `AES_192_CTR`, `AES_256_CTR`
encryption_key_hex	Ключ в виде шестнадцатиричного значения, используемый для шифрования (например, `00112233445566778899AABBCCDDEEFF`)

Если опция ***Recursively create directories*** активирована, ADQM автоматически создаст указанные директории для хранения зашифрованных данных (если эти директории не существуют) и назначит им владельца `clickhouse:clickhouse` с маской прав доступа `0770`.

Группу дисков по типу можно временно отключить, используя переключатель ***Enable...***, но учитывайте, что политика хранения не должна ссылаться на отсутствующий в конфигурации диск.

5. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы сохранить в конфигурации информацию о добавляемых дисках (при изменении существующих дисков необходимо также перезапустить сервис — для этого активируйте опцию ***Restart*** в окне настройки действия ***Reconfig***).

Информацию о сконфигурированных в ADQM дисках можно посмотреть в системной таблице system.disks:

```sql
SELECT
    name,
    path,
    formatReadableSize(free_space) AS free,
    formatReadableSize(total_space) AS total,
    formatReadableSize(keep_free_space) AS reserved
FROM system.disks;
```

При установке ADQM по умолчанию конфигурируется один локальный диск `default`. Данные будут сохраняться в каталог, указанный через параметр ***Data path*** в секции ***Directories*** на странице конфигурации сервиса ADQMDB (по умолчанию — */var/lib/clickhouse/*).

```
┌─name────┬─path─────────────────┬─free──────┬─total─────┬─reserved─┐
│ default │ /var/lib/clickhouse/ │ 39.33 GiB │ 49.99 GiB │ 0.00 B   │
└─────────┴──────────────────────┴───────────┴───────────┴──────────┘
```

Политики хранения

Политика хранения группирует диски в один или несколько томов, а также определяет правила записи данных таблицы на диски и то, как данные будут перемещаться между томами или отдельными дисками в томах.

Добавлять и конфигурировать политики хранения можно в интерфейсе ADCM:

1. Зайдите на страницу конфигурирования сервиса ADQMDB (***ADQM cluster*** → ***Services*** → ***ADQMDB*** → ***Primary Configuration***).
2. Включите опцию ***Show advanced***.
3. В секции ***Storage policies*** сконфигурируйте необходимые политики хранения в списке ***Policies***.

Параметры политики хранения

Параметр	Описание
policy_name	Уникальное имя политики
move_factor	Доля доступного свободного места на томе. Если места становится меньше, чем указанный фактор, данные автоматически начнут перемещаться на следующий том, если он есть. Для перемещения куски данных сортируются по размеру от большего к меньшему и выбираются куски, совокупный размер которых достаточен для соблюдения условия `move_factor`. Если совокупный размер всех кусков недостаточен, будут перемещены все куски
volumes	Список томов политики (политика должна содержать хотя бы один том). Конфигурационные параметры тома: ▪ ***volume_name*** — уникальное имя тома. ▪ ***max_data_part_size_bytes*** — максимальный размер куска данных, который может находиться на любом диске тома. Если в результате слияния размер куска ожидается больше значения `max_data_part_size_bytes`, этот кусок будет записан в следующий том. В основном эта функция позволяет хранить новые данные (небольшие куски) на горячем томе (SSD) и перемещать их на холодный том (HDD), когда они достигают большого размера (см. пример ниже). Не используйте этот параметр, если политика имеет только один том. ▪ ***prefer_not_to_merge*** — отключает слияние кусков данных, хранящихся на томе. Это позволяет контролировать работу ADQM с медленными дисками. ▪ ***volume_disks*** — список имен дисков внутри тома (том должен содержать хотя бы один диск). Если поверх диска с данными создан кеширующий диск, то для тома указывается кеширующий диск. Порядок томов в политике хранения важен, так как при достижении условий на переполнение тома данные переносятся на следующий том. Порядок дисков в томах также важен — данные записываются по очереди на каждый из них.

4. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB, чтобы сохранить в конфигурации информацию о добавляемых политиках хранения (при изменении существующих политик хранения необходимо также перезапустить сервис — для этого активируйте опцию ***Restart*** в окне настройки действия ***Reconfig***).

Получить информацию о политиках хранения, сконфигурированных в ADQM, можно из системной таблицы system.storage_policies:

```sql
SELECT policy_name, volume_name, disks FROM system.storage_policies;
```

По умолчанию ADQM содержит одну политику хранения `default` с одним томом, которая определяет, что все данные таблицы записываются на один диск `default`:

```
┌─policy_name─┬─volume_name─┬─disks───────┐
│ default     │ default     │ ['default'] │
└─────────────┴─────────────┴─────────────┘
```

Назначение таблице политики хранения

После того, как политики хранения сконфигурированы, их можно назначать таблицам. Для этого необходимо указать имя политики хранения в качестве значения параметра `storage_policy`, используя выражение `SETTINGS` при создании таблицы. Синтаксис запроса для создания таблицы MergeTree с указанием политики хранения данных в общем виде можно записать следующим образом:

```sql
CREATE TABLE <table_name> (<column_definitions>)
ENGINE = MergeTree
ORDER BY <sorting_key>
SETTINGS storage_policy = '<policy_name>';
```

По умолчанию таблице назначается политика хранения `default`.

Посмотреть, какая политика хранения используется таблицей и где хранятся данные, можно в полях `storage_policy` и `data_paths` системной таблицы system.tables. Например, выполните следующие запросы, чтобы создать таблицу без указания политики хранения, заполнить ее данными и вывести информацию о назначенной политике и директории для хранения данных:

```sql
CREATE TABLE test_table (id Int32) Engine=MergeTree ORDER BY id;
```

```sql
INSERT INTO test_table SELECT rand() FROM numbers(10000000);
```

```sql
SELECT name, storage_policy, data_paths FROM system.tables WHERE name = 'test_table' FORMAT Vertical;
```

```
Row 1:
──────
name:           test_table
storage_policy: default
data_paths:     ['/var/lib/clickhouse/store/50d/50d13b59-89d8-44ef-87f6-08ee8f71066c/']
```

Изменить политику хранения после создания таблицы можно с помощью запроса `ALTER TABLE … MODIFY SETTING`. При этом необходимо учесть, что новая политика должна содержать все тома и диски предыдущей политики с теми же именами.

Пример конфигурирования многоуровневого хранилища

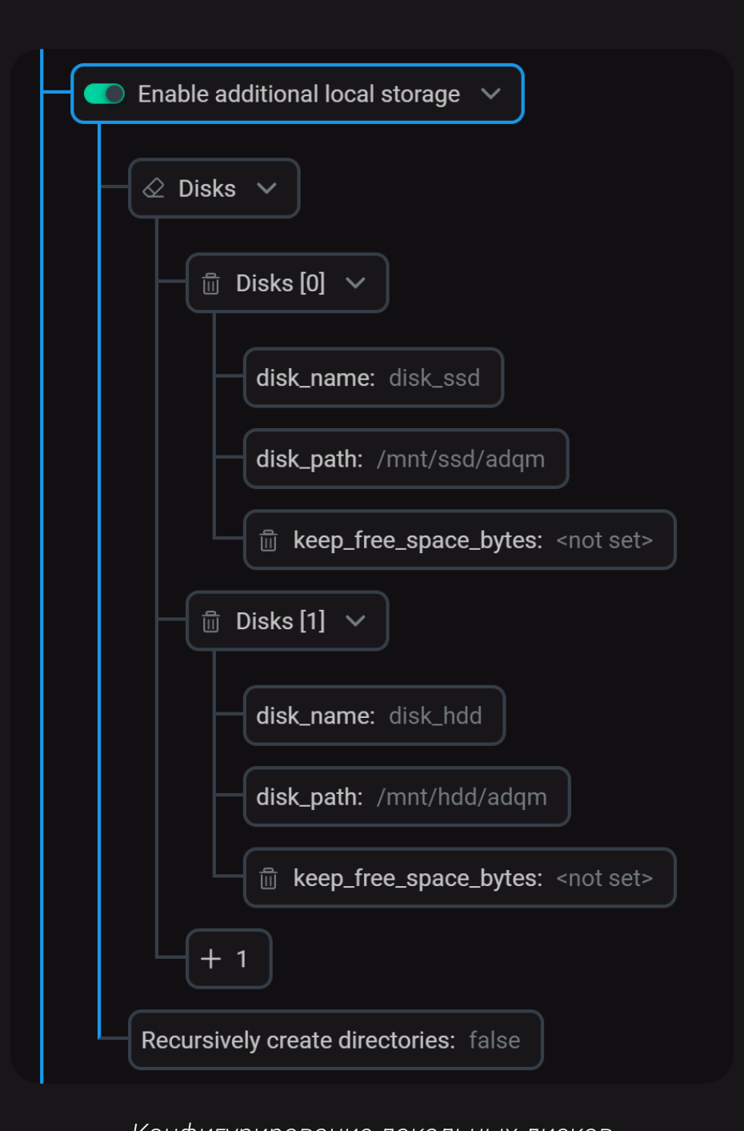
В данном примере для хранения данных используются два блочных устройства различных типов — SSD- и HDD-диск, примонтированные в следующие каталоги:

- */mnt/ssd/adqm/*
- */mnt/hdd/adqm/*

Каждому каталогу назначен владелец `clickhouse`:

```
$ sudo chown clickhouse:clickhouse -R /mnt/ssd/adqm/ /mnt/hdd/adqm/
```

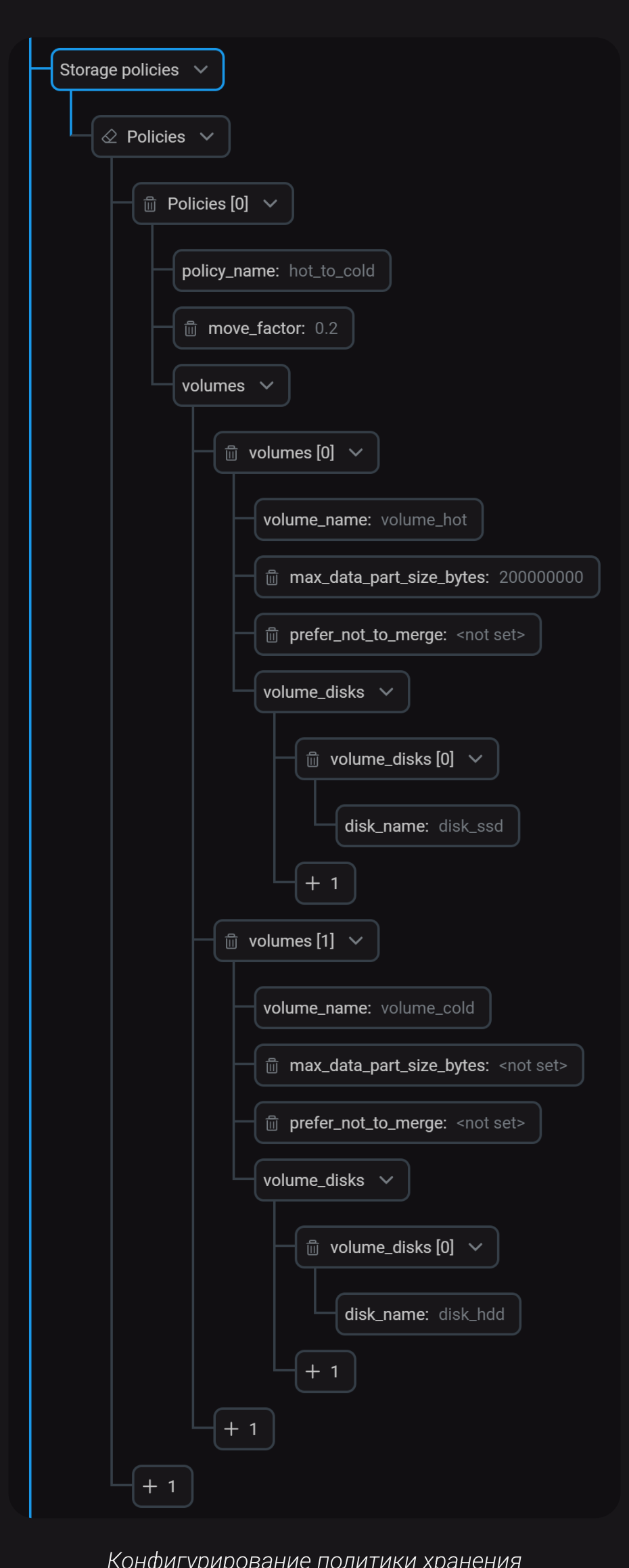
1. Добавьте локальные диски в ADQM на странице конфигурации сервиса ADQMDB в секции ***Enable additional local storage*** (убедитесь, что секция включена).



disk_name: disk_ssd
disk_path: /mnt/ssd/adqm
keep_free_space_bytes: <not set>
disk_name: disk_hdd
disk_path: /mnt/hdd/adqm
keep_free_space_bytes: <not set>
Recursively create directories: false

Конфигурирование локальных дисков

2. В секции ***Storage policies*** создайте политику (`hot_to_cold`), которая включает два тома для хранения "горячих" и "холодных" данных.



policy_name: hot_to_cold
move_factor: 0.2
volume_name: volume_hot
max_data_part_size_bytes: 200000000
prefer_not_to_merge: <not set>
disk_name: disk_ssd
volume_name: volume_cold
max_data_part_size_bytes: <not set>
prefer_not_to_merge: <not set>
disk_name: disk_hdd

Конфигурирование политики хранения

Том `volume_hot` содержит один SSD-диск (`disk_ssd`). Максимальный размер куска данных, который может храниться на этом томе — 200 МБ. Все куски данных таблицы больше 200 МБ будут сразу записываться на том `volume_cold`, который содержит один HDD-диск (`disk_hdd`). Табличный движок MergeTree постоянно выполняет фоновые слияния данных, объединяя с течением времени недавно вставленные небольшие куски в большие. То есть большие куски данных появляются после нескольких слияний, соответственно можно считать, что чем больше кусок, тем он старше.

Также при заполнении диска `disk_ssd` более чем на 80% данные будут переноситься на диск `disk_hdd` в фоновом режиме.

3. Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса, чтобы применить изменения.
   После этого, информация о дисках и политиках появится в соответствующих тегах внутри секции `<storage_configuration>` файла */etc/clickhouse-server/config.d/storage.xml*.

```xml
<storage_configuration>
    <disks>
        <disk_ssd>
            <type>local</type>
            <path>/mnt/ssd/adqm/</path>
        </disk_ssd>
        <disk_hdd>
            <type>local</type>
            <path>/mnt/hdd/adqm/</path>
        </disk_hdd>
    </disks>
    <policies>
        <hot_to_cold>
            <volumes>
                <volume_hot>
                    <disk>disk_ssd</disk>
                    <max_data_part_size_bytes>200000000</max_data_part_size_bytes>
                </volume_hot>
                <volume_cold>
                    <disk>disk_hdd</disk>
                </volume_cold>
            </volumes>
            <move_factor>0.2</move_factor>
        </hot_to_cold>
    </policies>
</storage_configuration>
```

4. Создайте таблицу, которая будет использовать новую политику хранения:

```sql
CREATE TABLE test_table1 (id Int32) Engine=MergeTree ORDER BY id SETTINGS storage_policy = 'hot_to_cold';
```

Заполните таблицу тестовыми данными, повторив несколько раз запрос на вставку данных:

```sql
INSERT INTO test_table1 SELECT rand() FROM numbers(10000000);
```

5. Подождите некоторое время, и выполните следующий запрос, чтобы посмотреть, как распределились данные по дискам:

```sql
SELECT
    disk_name,
    formatReadableSize(bytes_on_disk) AS size
FROM system.parts
WHERE (table = 'test_table1') AND active;
```

В результате слияния сформировался большой кусок данных, который был перемещен на "холодный" диск:

```
┌─disk_name─┬─size───────┐
│ disk_hdd  │ 530.49 MiB │
│ disk_ssd  │ 22.26 MiB  │
│ disk_ssd  │ 22.26 MiB  │
└───────────┴────────────┘
```

Системные таблицы

Елена Дворядкина

Содержание

Назад к содержанию

Обзор
Список системных таблиц

Обзор

Системные таблицы ADQM — это таблицы в базе данных `system`, предоставляющие системную информацию, которая может быть полезна администраторам и разработчикам для управления базами данных, оптимизации запросов, мониторинга производительности системы, анализа и устранения ошибок. Например, из системных таблиц можно получить:

- данные о состоянии сервера и его внутренних процессах;
- информацию о сконфигурированных кластерах, существующих базах данных, таблицах, столбцах и кусках данных;
- собираемые в режиме реального времени метрики сервера ClickHouse и данные о событиях, произошедших в системе;
- текущие значения конфигурационных параметров;
- справочную информацию о доступных опциях для построения запросов.

Данные большинства системных таблиц хранятся в оперативной памяти. Эти таблицы автоматически создаются при старте сервера ClickHouse и доступны только для чтения. Их нельзя удалять или изменять, но можно отсоединять.

К системным таблицам ADQM также относятся таблицы для хранения различных типов логов (лог-таблицы с именем `*_log`), которые начинают заполняться и становятся доступны из базы данных `system`, если соответствующее логирование активировано. В отличие от других системных таблиц, лог-таблицы основаны на табличном движке MergeTree. Данные этих таблиц хранятся в файловой системе и не удаляются автоматически, что гарантирует доступ к логам для анализа после перезапуска сервера. В статье how-to:logging.adoc#log_tables можно получить подробную информацию о том, какие типы логов можно записывать в системные таблицы, как включить ведение логов и настроить соответствующие лог-таблицы.

Список системных таблиц

Полный список доступных системных таблиц можно посмотреть через запрос:

```sql
SHOW TABLES FROM system;
```

Список доступных системных таблиц

Пример результата выполнения запроса `SHOW TABLES FROM system` (в списке отсутствуют таблицы `*_log`, так как логирование в ADQM не было включено):

```
    ┌─name─────────────────────────────┐
 1. │ aggregate_function_combinators │
 2. │ asynchronous_inserts           │
 3. │ asynchronous_loader            │
 4. │ asynchronous_metrics           │
 5. │ backups                        │
 6. │ build_options                  │
 7. │ certificates                   │
 8. │ clusters                       │
 9. │ collations                     │
10. │ columns                        │
11. │ contributors                   │
12. │ current_roles                  │
13. │ dashboards                     │
14. │ data_skipping_indices          │
15. │ data_type_families             │
16. │ database_engines               │
17. │ databases                      │
18. │ detached_parts                 │
19. │ dictionaries                   │
20. │ disks                          │
21. │ distributed_ddl_queue          │
22. │ distribution_queue             │
23. │ dns_cache                      │
24. │ dropped_tables                 │
25. │ dropped_tables_parts           │
26. │ enabled_roles                  │
27. │ errors                         │
28. │ events                         │
29. │ filesystem_cache               │
30. │ formats                        │
31. │ functions                      │
32. │ generateSeries                 │
33. │ generate_series                │
34. │ grants                         │
35. │ graphite_retentions            │
36. │ jemalloc_bins                  │
37. │ kafka_consumers                │
38. │ keywords                       │
39. │ licenses                       │
40. │ macros                         │
41. │ merge_tree_settings            │
42. │ merges                         │
43. │ metrics                        │
44. │ models                         │
45. │ moves                          │
46. │ mutations                      │
47. │ mysql_binlogs                  │
48. │ named_collections              │
49. │ numbers                        │
50. │ numbers_mt                     │
51. │ one                            │
52. │ part_moves_between_shards      │
53. │ parts                          │
54. │ parts_columns                  │
55. │ privileges                     │
56. │ processes                      │
57. │ projection_parts               │
58. │ projection_parts_columns       │
59. │ query_cache                    │
60. │ quota_limits                   │
61. │ quota_usage                    │
62. │ quotas                         │
63. │ quotas_usage                   │
64. │ remote_data_paths              │
65. │ replicas                       │
66. │ replicated_fetches             │
67. │ replicated_merge_tree_settings │
68. │ replication_queue              │
69. │ rocksdb                        │
70. │ role_grants                    │
71. │ roles                          │
72. │ row_policies                   │
73. │ s3queue                        │
74. │ scheduler                      │
75. │ schema_inference_cache         │
76. │ server_settings                │
77. │ settings                       │
78. │ settings_changes               │
79. │ settings_profile_elements      │
80. │ settings_profiles              │
81. │ stack_trace                    │
82. │ storage_policies               │
83. │ symbols                        │
84. │ table_engines                  │
85. │ table_functions                │
86. │ tables                         │
87. │ time_zones                     │
88. │ user_directories               │
89. │ user_processes                 │
90. │ users                          │
91. │ view_refreshes                 │
92. │ warnings                       │
93. │ zeros                          │
94. │ zeros_mt                       │
95. │ zookeeper                      │
96. │ zookeeper_connection           │
    └────────────────────────────────┘
```

Ниже системные таблицы ADQM сгруппированы по типу информации, которую они предоставляют.

Состояние сервера | Текущие процессы | Кеши

Системная таблица	Данные в таблице
merges	Информация о слияниях и мутациях кусков данных, которые выполняются в настоящее время для таблиц семейства MergeTree
mutations	Информация о ходе выполнения мутаций таблиц семейства MergeTree
processes	Информация обо всех запросах, выполняемых в данный момент
moves	Информация о текущих перемещениях кусков данных таблиц MergeTree на другие диски или тома хранилища (инициированных, например, условием TTL или запросом `ALTER TABLE … MOVE PARTITION|PART … TO DISK|VOLUME`)
replicated_fetches	Информация о выполняемых в данный момент фоновых операциях скачивания кусков данных с реплик
stack_trace	Трассировки стека всех серверных потоков (можно использовать для анализа состояния сервера)
asynchronous_inserts	Информация об ожидающих асинхронных вставках в очереди
asynchronous_loader	Информация и статус по последним асинхронным заданиям (например, для загрузки таблиц)
query_cache	Содержимое кеша запросов
dns_cache	Содержимое DNS-кеша
filesystem_cache	Информация о записях в кеше файловой системы для удаленных объектов

Кластеры | Шардирование | Репликация | ZooKeeper

Системная таблица	Данные в таблице
clusters	Информация о кластерах, описанных в конфигурационном файле
macros	Макросы, которые будут использоваться для автоматической подстановки идентификаторов шарда и реплики, соответствующих хосту, при создании реплицируемых таблиц в кластерах (параметры из секции `macros` конфигурации сервера)
replicas	Информация о реплицируемых таблицах, расположенных на сервере
replication_queue	Информация о задачах из очередей репликации, хранящихся в ZooKeeper или ClickHouse Keeper, для таблиц семейства ReplicationMergeTree
distribution_queue	Информация о локальных файлах, которые находятся в очереди для отправки на шарды (эти локальные файлы содержат новые куски данных, которые создаются путем вставки новых данных в таблицу Distributed в асинхронном режиме)
distributed_ddl_queue	Информация о распределенных DDL-запросах, которые были выполнены в кластере (запросы с `ON CLUSTER`)
zookeeper	Данные из кластера ZooKeeper или ClickHouse Keeper, описанного в конфигурации (можно получить список дочерних элементов для определенного узла или записанное внутри него значение)
zookeeper_connection	Текущие подключения к ZooKeeper или ClickHouse Keeper

Примеры

- Информация о сконфигурированных логических кластерах:

```sql
SELECT cluster, shard_num, replica_num, host_name FROM system.clusters;
```

- Проверка работоспособности реплики:

```sql
SELECT database, table, last_queue_update_exception, zookeeper_exception FROM system.replicas WHERE is_readonly;
```

  Если таблица заблокирована на запись (например, потому что в файле *config.xml* нет секции конфигурации ZooKeeper/ClickHouse Keeper или при переинициализации сессии в ZooKeeper/ClickHouse Keeper произошла ошибка), сообщения об ошибках можно посмотреть в полях `last_queue_update_exception` и `zookeeper_exception`.

- Состояние очереди репликации:

```sql
SELECT database, table, replica_name, type, num_postponed, postpone_reason, num_tries, last_exception
FROM system.replication_queue FORMAT Vertical;
```

Базы данных | Таблицы | Столбцы | Куски данных | Словари

Системная таблица	Данные в таблице
databases	Информация о базах данных, доступных для текущего пользователя
tables	Информация обо всех таблицах, о которых "знает" сервер (отключенные через `DETACH TABLE` таблицы не учитываются)
columns	Информация о столбцах таблиц
parts	Информация о кусках данных таблиц семейства MergeTree
parts_columns	Информация о столбцах всех существующих на данный момент кусков данных таблиц MergeTree
detached_parts	Информация об отсоединенных кусках данных таблиц семейства MergeTree
dropped_tables	Информация об удаленных из баз данных таблицах, очистка данных которых еще не выполнена полностью
dropped_tables_parts	Информация о кусках данных удаленных таблиц MergeTree, перечисленных в `system.dropped_tables`
projection_parts	Информация о существующих на данный момент кусках данных, созданных для всех проекций всех таблиц в кластере (это копии родительских кусков данных, в которых данные агрегированы или отсортированы в другом порядке). Эта таблица схожа с `system.parts`, но содержит дополнительную информацию о родительских кусках данных в столбцах с префиксом `parent_`
projection_parts_columns	Информация о столбцах всех существующих на данный момент кусков данных проекций таблиц MergeTree
data_skipping_indices	Информация о существующих индексах пропуска данных во всех таблицах
dictionaries	Информация о словарях
named_collections	Сконфигурированные именованные коллекции (named collections) — реквизиты для подключения к внешним источникам при использовании словарей, таблиц, табличных функций

Примеры

- Список столбцов таблицы с типами данных:

```sql
SELECT name, type FROM system.columns WHERE table = '<table_name>';
```

- Информация о кусках данных таблицы:

```sql
SELECT name, active, rows, data_compressed_bytes, data_uncompressed_bytes
FROM system.parts WHERE table = '<table_name>'
FORMAT Vertical;
```

  где `<table_name>` — имя таблицы.

Система управления доступом

Системная таблица	Данные в таблице
users	Информация об аккаунтах пользователей, настроенных на сервере
grants	Привилегии, предоставленные пользователям (список доступных для назначения привилегий можно посмотреть в системной таблице `system.privileges`)
roles, role_grants, current_roles, enabled_roles	Информация о ролях и их назначениях пользователям
row_policies	Информация о политиках доступа к строкам
settings_profiles	Сконфигурированные профили настроек
settings_profile_elements	Описание содержимого профиля настроек (настройки и их значения; ограничения; роли и пользователи, к которым применяются настройки; родительские профили настроек)
quota_limits, quota_usage, quotas, quotas_usage	Информация о квотах и их использовании

Хранилище данных

Системная таблица	Данные в таблице
disks	Информация о сконфигурированных дисках
storage_policies	Информация о сконфигурированных политиках хранения

Примеры

- Информация о дисках хранилища данных:

```sql
SELECT
    name,
    path,
    formatReadableSize(free_space) AS free,
    formatReadableSize(total_space) AS total,
    formatReadableSize(keep_free_space) AS reserved
FROM system.disks;
```

- Информация о сконфигурированных в ADQM политиках хранения:

```sql
SELECT policy_name, volume_name, disks FROM system.storage_policies;
```

Метрики сервера ClickHouse | Ошибки и предупреждения

Системная таблица	Данные в таблице
metrics	Метрики, которые рассчитываются в каждый момент времени и имеют актуальные текущие значения (см. ***Metrics*** в разделе ***Типы метрик*** статьи Метрики мониторинга кластера ADQM)
events	Информация о количестве событий, произошедших в системе (см. ***ProfileEvents*** в разделе ***Типы метрик*** статьи Метрики мониторинга кластера ADQM)
asynchronous_metrics	Метрики, которые периодически вычисляются в фоновом режиме (см. ***AsynchronousMetrics*** в разделе ***Типы метрик*** статьи Метрики мониторинга кластера ADQM)
user_processes	Информация об использовании памяти процессами, которые запустил пользователь, и событиях ProfileEvents для пользователя
errors	Информация обо всех произошедших ошибках (для каждого типа ошибки выводится код и сообщение об ошибке, общее количество срабатываний и время последнего срабатывания, трассировка стека для отладки)
warnings	Предупреждающие сообщения о конфигурации сервера (если таблица не пустая, содержащиеся в ней предупреждающие сообщения будут выводиться при подключении к серверу ClickHouse через clickhouse-client)
backups	Информация обо всех операциях `BACKUP` и `RESTORE` (таблица не является персистентной, то есть она содержит только операции, выполненные после последнего перезапуска сервера)
jemalloc_bins	Информация о распределении памяти, выполненном с помощью jemalloc в классах разных размеров (bins), агрегированная со всех арен (arenas). Эта статистика может быть не совсем точной из-за локального кеширования потоков в jemalloc

Примеры

- Информация об ошибках:

```sql
SELECT name, code, value, last_error_time FROM system.errors;
```

- Значение конкретной метрики (например, `Query` — количество выполняемых запросов):

```sql
SELECT * FROM system.metrics WHERE metric = 'Query';
```

Настройки

Системная таблица	Данные в таблице
settings	Настройки на уровне пользователя (которые можно изменять в рамках отдельного запроса или сессии)
server_settings	Глобальные настройки сервера, указанные в *config.xml*
merge_tree_settings	Список всех настроек табличного движка MergeTree, их текущие значения и значения по умолчанию, а также описания. Эти параметры можно настраивать при создании или модификации таблицы, используя выражение `SETTINGS` в запросе `CREATE TABLE` или `ALTER TABLE` соответственно
replicated_merge_tree_settings	Список всех настроек табличного движка ReplicatedMergeTree, их текущие значения и значения по умолчанию, а также описания. Эти параметры можно настраивать при создании или модификации таблицы, используя выражение `SETTINGS` в запросе `CREATE TABLE` или `ALTER TABLE` соответственно
settings_changes	Информация об изменениях настроек в различных версиях ClickHouse
build_options	Параметры сборки сервера ClickHouse

Примеры

Настройки, значения по умолчанию которых были изменены (например, при устранении неполадок может быть полезно проанализировать, не могло ли изменение каких-либо настроек повлиять на поведение системы):

```sql
SELECT name, value FROM system.settings WHERE changed;
```

```sql
SELECT name, value FROM system.server_settings WHERE changed;
```

Информация для разных табличных движков

Системная таблица	Данные в таблице
mysql_binlogs	Список активных бинарных журналов обновлений MySQL (файлов binlog) для баз данных MaterializedMySQL
kafka_consumers	Информация о потребителях Kafka (таблицах на базе движка Kafka)
rocksdb	Статистика RocksDB при настроенной в ADQM интеграции через табличный движок EmbeddedRocksDB
graphite_retentions	Информация о параметрах `graphite_rollup`, которые используются в таблицах GraphiteMergeTree
s3queue	Состояние S3Queue в памяти: какие файлы обрабатываются в данный момент, какие файлы обработаны или обработать не удалось

Данные для тестирования

ADQM предоставляет набор таблиц, которые могут быть полезны для выполнения тестирования и генерации последовательных значений.

Системная таблица	Данные в таблице
numbers	Таблица содержит один столбец `number` (типа UInt64) со всеми натуральными числами от `0` в отсортированном порядке. Чтение из таблицы не распараллеливается. Эту таблицу можно также получить с помощью табличной функции numbers
numbers_mt	Таблица, аналогичная `system.numbers`, но чтение распараллеливается. Числа могут возвращаться в произвольном порядке
zeros	Таблица содержит один столбец `zero` (типа UInt64) с нулями. Можно использовать для тестов производительности вместо `system.numbers` как самый быстрый способ создать большое количество строк
zeros_mt	Таблица, аналогичная `system.zeros`, но чтение распараллеливается
generate_series, generateSeries	Таблица содержит один столбец `generate_series` (типа UInt64) со всеми натуральными числами от `0` в отсортированном порядке. Чтобы получить такую же таблицу с арифметической прогрессией отсортированных натуральных чисел в указанном диапазоне с указанным шагом, можно также использовать табличные функции generate_series или generateSeries
one	Таблица содержит значение `0` в единственной строке с одним столбцом `dummy` типа UInt8 (аналог таблицы `DUAL` в других СУБД). Эта таблица используется, если в запросе `SELECT` не указана секция `FROM` — например, `SELECT 1`

Справочная информация

Системная таблица	Данные в таблице
functions	Поддерживаемые функции (обычные и агрегатные)
aggregate_function_combinators	Комбинаторы для агрегатных функций
table_functions	Поддерживаемые табличные функции
database_engines	Поддерживаемые движки баз данных
table_engines	Поддерживаемые табличные движки и их функциональные возможности
data_type_families	Поддерживаемые типы данных
formats	Поддерживаемые форматы входных и выходных данных
privileges	Поддерживаемые типы привилегий, которые можно предоставлять пользователям
keywords	Ключевые слова, которые можно использовать в синтаксисе запросов
time_zones	Часовые пояса, которые поддерживаются сервером ClickHouse
collations	Поддерживаемые значения collation (сравнение), которые можно использовать в запросах с `ORDER BY … COLLATE <'collation_name'>` для сортировки по строковым значениям с учетом алфавита указанного языка

Пример

Список доступных агрегатных функций:

```sql
SELECT name FROM system.functions WHERE is_aggregate = 1;
```

Управление настройками пользователя

Елена Дворядкина

Содержание

Назад к содержанию

Настройка параметров на уровне аккаунта пользователя

Настройка параметров на уровне сессии

Настройка параметров для отдельного запроса

Значения параметров по умолчанию

Примеры

В ADQM/ClickHouse есть две основные группы настроек:

- **Настройки на уровне сервера** — параметры, которые указываются в конфигурационном файле *config.xml* и применяются глобально для сервера ClickHouse. Эти настройки нельзя изменить на уровне пользователя, сессии или отдельных запросов. Список параметров и их значения можно посмотреть в системной таблице `system.server_settings`.
- **Настройки на уровне пользователя** — параметры, которые можно сконфигурировать для отдельного пользователя, сессии или конкретного запроса в зависимости от сценария использования и времени, в течение которого необходимо сохранять установленные значения актуальными. Полный список доступных параметров и их значения можно посмотреть в системной таблице `system.settings`. Способы настройки этих параметров описаны ниже.

Настройка параметров на уровне аккаунта пользователя

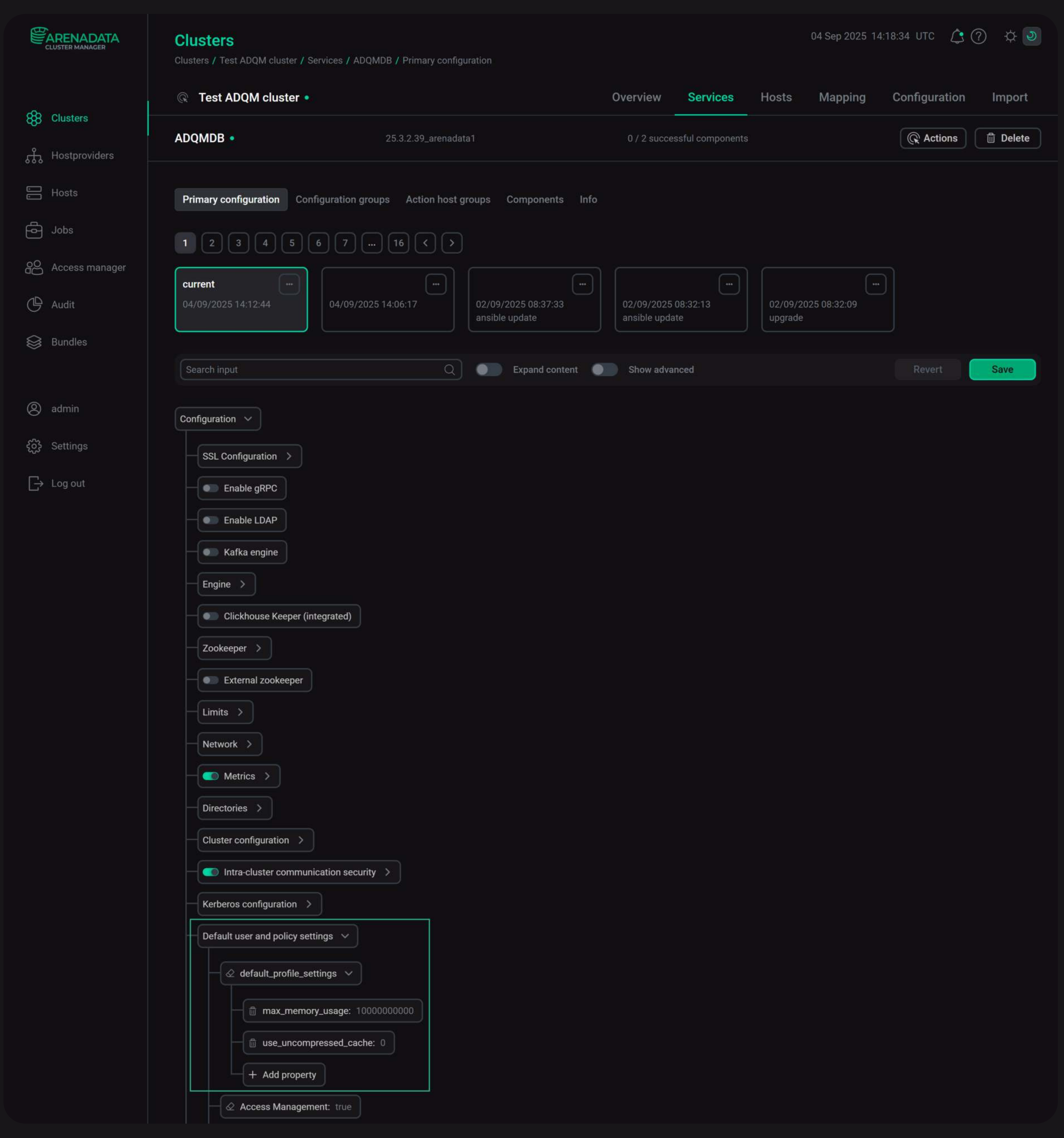
Чтобы указать значения параметров, которые будут применяться при входе пользователя в систему, используйте выражение `SETTINGS` в запросе `CREATE USER` при создании пользователя или `ALTER USER` при изменении существующего пользователя:

```
CREATE USER <user_name> ... SETTINGS <parameter_name> = <value>, ...;
```

Значения параметров можно также указать в профиле настроек, который затем назначить пользователю в запросе `CREATE USER` или `ALTER USER`:

```
CREATE SETTINGS PROFILE <profile_name> SETTINGS <parameter_name> = <value>, ...;
CREATE USER <user_name> SETTINGS PROFILE <profile_name>;
```

Управлять настройками профиля пользователя ADQM по умолчанию можно через интерфейс ADCM — используйте параметр ***default_profile_settings*** в секции ***Default user and policy settings*** на странице конфигурации сервиса ADQMDB, чтобы сконфигурировать список параметров для пользователя `default`.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
04 Sep 2025 14:18:34 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
25.3.2.39_arenadata1
0 / 2 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
1 2 3 4 5 6 7 ... 16
current 04/09/2025 14:12:44
04/09/2025 14:06:17
02/09/2025 08:37:33 ansible update
02/09/2025 08:32:13 ansible update
02/09/2025 08:32:09 upgrade
Search input
Expand content
Show advanced
Revert
Save
Configuration
SSL Configuration
Enable gRPC
Enable LDAP
Kafka engine
Engine
Clickhouse Keeper (integrated)
Zookeeper
External zookeeper
Limits
Network
Metrics
Directories
Cluster configuration
Intra-cluster communication security
Kerberos configuration
Default user and policy settings
default_profile_settings
Add property

max_memory_usage: 10000000000

use_uncompressed_cache: 0

Access Management: true

Управление настройками для пользователя ADQM по умолчанию через ADCM

После того, как необходимые параметры установлены, не забудьте нажать ***Save*** и выполнить действие ***Reconfig*** для сервиса ADQMDB, чтобы сохранить информацию об изменениях конфигурации и перезапустить сервис.

Настройка параметров на уровне сессии

С помощью оператора `SET` можно указать значение параметра на время текущей сессии (это значение временно заменит значение, установленное на уровне аккаунта пользователя):

```
SET <parameter_name> = <value>;
```

Запрос `SET` также позволяет установить все значения из указанного профиля настроек:

```
SET profile = '<profile_name>';
```

Применение настроек на уровне сессии может быть удобно для специального тестирования или когда нужно, чтобы определенные значения параметров действовали при выполнении запросов в рамках одной сессии, но не дольше.

Настройка параметров для отдельного запроса

Настройки также можно конфигурировать для каждого отдельного запроса. Для этого в тексте запроса после выражения `SETTINGS` перечислите параметры со значениями:

```
SELECT <column_list> FROM <table_name> SETTINGS <parameter_name> = <value>, ...;
```

Указанные настройки будут применяться только в рамках выполнения данного запроса — после его завершения для параметров будут действовать значения, указанные на уровне пользователя/сессии, или значения по умолчанию.

Значения параметров по умолчанию

Чтобы узнать, значения каких параметров были изменены для пользователя/сессии и отличаются от значений по умолчанию, выполните запрос:

```
SELECT name, value FROM system.settings WHERE changed;
```

Вернуть параметр к его значению по умолчанию можно следующей командой:

```
SET <parameter_name> = DEFAULT;
```

Примеры

Зайдите в консольный клиент clickhouse-client как пользователь `default`. Посмотрите значение параметра `max_columns_to_read`:

```
SELECT name, value FROM system.settings WHERE name = 'max_columns_to_read';
```

```
   ┌─name────────────────┬─value─┐
1. │ max_columns_to_read │ 0     │
   └─────────────────────┴───────┘
```

Создайте нового пользователя, указав через выражение `SETTINGS` для параметра `max_columns_to_read` значение, отличное от значения по умолчанию, например:

```
CREATE USER test_user IDENTIFIED WITH sha256_password BY 'qwerty123' SETTINGS max_columns_to_read = 20;
```

Зайдите в `clickhouse-client` как пользователь `test_user` и выполните команду:

```
SELECT name, value FROM system.settings WHERE changed;
```

Параметр `max_columns_to_read` появится в списке измененных настроек со значением, которое было установлено для текущего пользователя:

```
   ┌─name─────────────────────────────┬─value────────┐
1. │ max_columns_to_read              │ 20           │
     ...
   └──────────────────────────────────┴──────────────┘
```

Измените значение параметра на уровне сессии:

```
SET max_columns_to_read = 25;
```

Снова проверьте значение параметра:

```
SELECT name, value FROM system.settings WHERE changed;
```

```
   ┌─name─────────────────────────────┬─value────────┐
1. │ max_columns_to_read              │ 25           │
     ...
   └──────────────────────────────────┴──────────────┘
```

Если выйти из консольного клиента (завершить сессию) и снова зайти как пользователь `test_user`, можно убедиться, что параметр `max_columns_to_read` снова имеет значение `20`, присвоенное на уровне аккаунта пользователя.

Именованные коллекции

Елена Дворядкина

Содержание

Назад к содержанию

- Управление именованными коллекциями
- Переопределение параметров
- Создание именованных коллекций
  - DDL-запросы
  - XML-файлы
- Использование именованных коллекций
- Изменение и удаление именованных коллекций
  - Изменение именованных коллекций DDL
  - Удаление именованных коллекций DDL

Именованные коллекции (named collections) в ADQM/ClickHouse — предопределенные наборы параметров в виде пар ключ/значение, которые можно неоднократно использовать в SQL-запросах при подключении к внешним источникам данных (таким как PostgreSQL, Kafka, S3, MySQL и другим) через табличные функции, словари, интеграционные движки таблиц и баз данных.

Эта функциональность позволяет:

- делать SQL-запросы более короткими и понятными, избегая повторного определения одних и тех же параметров — значения параметров можно указать один раз в рамках именованной коллекции, а затем обращаться к этой коллекции в различных запросах по мере необходимости;
- централизованно управлять наборами параметров — конфигурация для интеграции с внешней системой хранится в одном месте, где можно, например, легко проанализировать ошибки подключения в случае их возникновения и обновить необходимые параметры;
- защитить конфиденциальные учетные данные (например, пароли) для интеграции с внешними системами, скрывая их от пользователей без прав доступа администратора.

ПРИМЕЧАНИЕ

В данной статье способы создания и использования именованных коллекций описаны на примере интеграции ADQM и ADPG. Для воспроизведения приведенных примеров подготовьте ADPG, как описано в статье Интеграция ADQM и ADPG: создайте пользователя PostgreSQL, от имени которого будет выполняться подключение к ADPG из ADQM, и две базы данных PostgreSQL с таблицами для тестирования.

Также можно посмотреть пример использования именованной коллекции для хранения параметров подключения к хранилищу S3 в статье Интеграция ADQM и S3.

Управление именованными коллекциями

Для управления именованными коллекциями (создания/изменения/удаления) можно использовать:

- DDL-запросы. Чтобы у пользователя было разрешение конфигурировать именованные коллекции через DDL-запросы, ему должна быть предоставлена привилегия `named_collection_control` — для этого в конфигурационном файле */etc/clickhouse-server/users.xml* (или в другом XML-файле директории */etc/clickhouse-server/users.d/*) установите значение параметра `named_collection_control` равным `1` в настройках пользователя.

  Пример

  Назначение привилегии `named_collection_control` пользователю `default` в файле *users.xml*:

  ```
  <users>
      <default>
          ...
          <named_collection_control>1</named_collection_control>
          ...
      </default>
  </users>
  ```

- XML-файлы. Именованные коллекции можно определять в конфигурационном файле */etc/clickhouse-server/config.xml* или в других XML-файлах директории */etc/clickhouse-server/config.d/*. Созданную или измененную в конфигурационном файле именованную коллекцию можно применять после перезапуска сервера ClickHouse.

Список всех существующих именованных коллекций (созданных с помощью DDL-запросов и описанных в XML-файлах) можно посмотреть в системной таблице `named_collections`.

Переопределение параметров

Параметры, указанные в именованной коллекции, могут быть переопределены в использующем эту коллекцию SQL-запросе. Для каждого параметра в составе именованной коллекции можно ограничить возможность изменения его значения с помощью специального флага:

- `OVERRIDABLE` (или `overridable="true"` в XML-конфигурации) — значение параметра можно переопределять;
- `NOT OVERRIDABLE` (или `overridable="false"` в XML-конфигурации) — значение параметра запрещено переопределять;
- флаг не указан — возможность переопределения значения параметра определяется настройкой allow_named_collection_override_by_default.

ВНИМАНИЕ

Если вы используете именованные коллекции с целью скрыть учетные данные для подключения к внешним системам от пользователей без доступа администратора, следует ограничить возможность переопределения соответствующих параметров.

Создание именованных коллекций

DDL-запросы

Создать именованную коллекцию можно с помощью запроса `CREATE NAMED COLLECTION`:

```
CREATE NAMED COLLECTION [IF NOT EXISTS] <collection_name> [ON CLUSTER <cluster_name>] AS
key1 = 'value1' [[NOT] OVERRIDABLE],
key2 = 'value2' [[NOT] OVERRIDABLE],
...;
```

Ключи в именованной коллекции должны совпадать с названиями параметров соответствующей функции, источника данных для словаря, движка таблицы или базы данных. Посмотреть точные названия параметров можно в документации ClickHouse — например, в статьях postgresql, mysql, s3, remote для соответствующих табличных функций или PostgreSQL, MySQL, Kafka для табличных движков.

При обращении к именованной коллекции, в которой указан недопустимый ключ, консольный клиент clickhouse-client выведет сообщение о соответствующей ошибке с указанием списка допустимых ключей (обязательных и опциональных). Например, если в именованной коллекции указан ключ `user_password`, при вызове функции `postgresql` со ссылкой на эту коллекцию `clickhouse-client` сообщит, что функция не поддерживает такой ключ:

```
DB::Exception: Unexpected key user_password in named collection. Required keys: database, db, password, table, user, username,
optional keys: addresses_expr, host, hostname, on_conflict, port, schema, use_table_cache. (BAD_ARGUMENTS)
```

Пример

Следующий запрос создает именованную коллекцию `my_adpg` с параметрами для подключения ADQM к ADPG:

```
CREATE NAMED COLLECTION my_adpg AS
host = '<adpg_host>',
port = 5432,
database = 'adpg_db1' NOT OVERRIDABLE,
user = 'adqm_user',
password = '<adqm_user_secret_password>';
```

Значения параметров (база данных `adpg_db1` и пользователь PostgreSQL `adqm_user`) здесь указаны в соответствии с примерами, описанными в статье Интеграция ADQM и ADPG — см. раздел ***Подготовка ADPG для тестовых примеров***.

XML-файлы

Чтобы определять именованные коллекции в XML-конфигурации сервера ClickHouse, используйте секцию `named_collections`:

```
<clickhouse>
    <named_collections>
        <named_collection1>
            <key1 overridable="true">value1</key1>
            <key2 overridable="false">value2</key2>
            <key3>value3</key3>
            ...
        </named_collection1>
        <named_collection2>
            ...
        </named_collection2>
        ...
    </named_collections>
</clickhouse>
```

Пример

Следующая конфигурация описывает именованную коллекцию `my_adpg_xml` с параметрами подключения к базе данных ADPG `adpg_db2` (детали создания этой базы данных также описаны в статье Интеграция ADQM и ADPG):

```
<clickhouse>
    <named_collections>
        <my_adpg_xml>
            <host>adpg_host</host>
            <port>5432</port>
            <database>adpg_db2</database>
            <user>adqm_user</user>
            <password>adqm_user_secret_password</password>
        </my_adpg_xml>
    </named_collections>
</clickhouse>
```

Использование именованных коллекций

После того как именованная коллекция создана, ее можно использовать в SQL-запросах, чтобы ссылаться на предопределенные параметры подключения к внешнему источнику данных при работе с табличной функцией, интеграционным табличным движком или движком баз данных, словарем.

Следующие примеры демонстрируют различные способы получить доступ к данным PostgreSQL из ADQM с использованием созданных выше именованных коллекций с параметрами подключения к ADPG.

- **Табличная функция postgresql**
  Чтение данных из таблицы ADPG `adpg_db1.table1` с помощью табличной функции postgresql (таблица не определена в именованной коллекции, поэтому ее необходимо указать отдельно в параметрах табличной функции):

  ```
  SELECT id, name FROM postgresql(my_adpg, table = 'table1');
  ```

  Запись данных в таблицу ADPG из ADQM:

  ```
  INSERT INTO TABLE FUNCTION postgresql(my_adpg, table = 'table1') (id, name) VALUES (3, 'three');
  ```

- **Табличный движок PostgreSQL**
  Создание таблицы ADQM на базе табличного движка PostgreSQL:

  ```
  CREATE TABLE psql_table (id Int32, name String) ENGINE = PostgreSQL(my_adpg, table = 'table1');
  ```

  Движок копирует данные из именованной коллекции при создании таблицы. Последующее изменение коллекции не будет влиять на существующие таблицы.

- **База данных PostgreSQL**
  Создание в ADQM базы данных на основе движка баз данных PostgreSQL (в эту базу данных будут автоматически добавлены таблицы из базы данных PostgreSQL `adpg_db1` — в данном примере одна таблица `table1`):

  ```
  CREATE DATABASE psql_database ENGINE = PostgreSQL(my_adpg);
  ```

- **Табличный движок MaterializedPostgreSQL**
  Создание таблицы MaterializedPostgreSQL для репликации данных таблицы ADPG `adpg_db2.table1` с указанием параметров подключения к ADQG через именованную коллекцию my_adpg_xml:

  ```
  SET allow_experimental_materialized_postgresql_table=1;
  ```

  ```
  CREATE TABLE psql_table_replica (id Int32, name String)
  ENGINE = MaterializedPostgreSQL(my_adpg_xml, table = 'table1');
  ```

- **База данных MaterializedPostgreSQL**
  Создание базы данных MaterializedPostgreSQL, в которую будут реплицироваться таблицы `table1` и `table2` из базы данных ADPG `adpg_db2`, с использованием именованной коллекции `my_adpg_xml` для указания параметров подключения для этой репликации:

  ```
  SET allow_experimental_database_materialized_postgresql=1;
  ```

  ```
  CREATE DATABASE psql_db_replica
  ENGINE = MaterializedPostgreSQL(my_adpg_xml)
  SETTINGS materialized_postgresql_tables_list = 'table1,table2';
  ```

- **Источник данных для словаря типа POSTGRESQL**
  Создание в ADQM словаря, который будет считывать данные из таблицы ADPG `adpg_db2.roles_dict` (именованная коллекция с настройками подключения к ADPG указывается с помощью параметра `name` в конфигурации источника данных для словаря):

  ```
  CREATE DICTIONARY roles_dict_psql (id Int32, role String DEFAULT 'no role assigned')
  PRIMARY KEY id
  SOURCE(POSTGRESQL(name my_adpg_xml table roles_dict))
  LIFETIME(MIN 10 MAX 20)
  LAYOUT(FLAT());
  ```

Изменение и удаление именованных коллекций

Именованные коллекции, созданные через DDL-запросы, можно изменять или удалять также с помощью DDL-запросов. Именованные коллекции, определенные в XML-файлах, можно изменять и удалять, редактируя или удаляя соответствующие им XML-описания.

Изменение именованных коллекций DDL

Изменить существующую именованную коллекцию можно с помощью запроса `ALTER NAMED COLLECTION`, используя ключевое слово `SET` для добавления/изменения параметров и `DELETE` — для удаления существующих параметров. Синтаксис запроса в общем виде:

```
ALTER NAMED COLLECTION [IF EXISTS] <collection_name> [ON CLUSTER <cluster_name>]
[SET key1 = 'value1' [[NOT] OVERRIDABLE], key2 = 'value2' [[NOT] OVERRIDABLE], ... ] |
[DELETE key3, key4, ... ];
```

Ниже приведены примеры использования запроса `ALTER NAMED COLLECTION` для изменения именованной коллекции my_adpg.

- **Добавление/изменение параметров**
  Изменение значения параметра `database` и добавление параметра `table` (при этом изначально установленный для параметра `database` флаг `NOT OVERRIDABLE` не меняется, то есть базу данных по-прежнему нельзя переопределить в SQL-запросе):

  ```
  ALTER NAMED COLLECTION my_adpg SET database = 'adpg_db2', table = 'table1';
  ```

  Изменение значения параметра `database` с разрешением его переопределять:

  ```
  ALTER NAMED COLLECTION my_adpg SET database = 'adpg_db1' OVERRIDABLE;
  ```

- **Удаление параметров**
  Удаление параметра `table` из коллекции `my_adpg`:

  ```
  ALTER NAMED COLLECTION my_adpg DELETE table;
  ```

  Если параметр необходимо перенастроить так, чтобы в качестве флага на разрешение/ограничение переопределения его значения применялась настройка по умолчанию (`allow_named_collection_override_by_default`), этот параметр нужно удалить из именованной коллекции и снова добавить без указания флага:

  ```
  ALTER NAMED COLLECTION my_adpg DELETE database;
  ```

  ```
  ALTER NAMED COLLECTION my_adpg SET database = 'adpg_db1';
  ```

  Изменение значения параметра `password` и удаление параметра `database` в одном запросе:

  ```
  ALTER NAMED COLLECTION my_adpg SET password = '<new_password>' DELETE database;
  ```

Удаление именованных коллекций DDL

Чтобы удалить именованную коллекцию, созданную через DDL-запрос, используйте запрос `DROP NAMED COLLECTION`:

```
DROP NAMED COLLECTION <collection_name>;
```

Метрики мониторинга кластера ADQM

Елена Дворядкина

Содержание

Назад к содержанию

Типы метрик

Просмотр метрик мониторинга

В статье описываются метрики мониторинга кластера ADQM. Для получения информации об установке мониторинга обратитесь к разделам:

- Online-установка
- Offline-установка

Типы метрик

Для кластера ADQM доступны две группы метрик: системные метрики и метрики сервера ClickHouse. Если мониторинг настроен через сервис Monitoring (доступен в ADQM, начиная с версии 23.8.2.7), метрики собираются со всех сервисов ADQM, включая ZooKeeper, ClickHouse Keeper, Chproxy.

Системные метрики показывают общие характеристики состояния хостов кластера, связанные, как правило, с потреблением ресурсов. Ниже представлен список системных метрик.

System metrics

Группа метрик	Описание
cpu	Уровень загруженности процессора
diskspace	Заполнение диска
files	Файловая статистика
iostat	Производительность операций ввода/вывода
loadavg	Средние значения загрузки системы
memory	Использование памяти
netstat	Статистика сетевых подключений
network	Производительность сетевых интерфейсов

Метрики сервера ClickHouse, доступные для кластера ADQM, включают:

- Metrics — метрики, которые рассчитываются в каждый момент времени и имеют актуальные текущие значения (например, количество одновременно обрабатываемых запросов или текущее значение задержки реплики).
- ProfileEvents — информация о количестве событий, произошедших в системе (например, количество запросов `SELECT`, обработанных с момента запуска сервера ClickHouse).
- AsynchronousMetrics — метрики, которые периодически вычисляются в фоновом режиме (например, объем используемой оперативной памяти).

В таблицах ниже приведены метрики сервера ClickHouse.

Metrics

Название метрики	Описание
ActiveAsyncDrainedConnections	Количество активных соединений, асинхронно опустошенных (все данные из которых были асинхронно извлечены)

ActiveSyncDrainedConnections	Количество активных соединений, синхронно опустошенных
AsyncDrainedConnections	Количество соединений, асинхронно опустошенных
AsynchronousReadWait	Количество потоков, ожидающих асинхронного чтения
BackgroundBufferFlushSchedulePoolTask	Количество активных задач в BackgroundBufferFlushSchedulePool (пул, который используется для выполнения фонового сброса данных в таблицах с движком Buffer)
BackgroundCommonPoolTask	Количество активных задач в соответствующем фоновом пуле
BackgroundDistributedSchedulePoolTask	Количество активных задач в BackgroundDistributedSchedulePool (пул, который используется для выполнения фоновых задач в распределенных таблицах)
BackgroundFetchesPoolTask	Количество активных скачиваний кусков данных для реплицируемых таблиц в соответствующем фоновом пуле
BackgroundMergesAndMutationsPoolTask	Количество активных слияний и мутаций в соответствующем фоновом пуле
BackgroundMessageBrokerSchedulePoolTask	Количество активных задач в BackgroundProcessingPool для фоновой потоковой передачи сообщений
BackgroundMovePoolTask	Количество активных задач в BackgroundProcessingPool для фоновых перемещений кусков данных
BackgroundSchedulePoolTask	Количество активных задач в BackgroundSchedulePool (пул, который используется для периодических задач в реплицируемых таблицах, таких как удаление старых кусков данных, изменение кусков данных, повторная инициализация реплики и других)
BrokenDistributedFilesToInsert	Количество файлов для асинхронной вставки в распределенные таблицы, которые были отмечены как поврежденные. При запуске сервера значение метрики `0`. Количество файлов для каждого шарда суммируется
CacheDetachedFileSegments	Количество существующих отделенных сегментов кеш-файла
CacheDictionaryUpdateQueueBatches	Количество "пакетов" (наборов ключей) в очереди обновлений в CacheDictionaries
CacheDictionaryUpdateQueueKeys	Точное количество ключей в очереди обновлений в CacheDictionaries
CacheFileSegments	Количество существующих сегментов кеш-файла
ContextLockWait	Количество потоков, ожидающих блокировки Context (глобальная блокировка)
DelayedInserts	Количество запросов `INSERT`, отложенных из-за большого количества активных кусков данных для партиции в таблице MergeTree
DictCacheRequests	Количество запросов к источникам данных словарей, которые хранятся в кеше
DiskSpaceReservedForMerge	Место на диске, зарезервированное для текущих фоновых слияний (немного больше, чем общий размер сливающихся в данный момент кусков данных)
DistributedFilesToInsert	Количество файлов, ожидающих обработки для асинхронной вставки в распределенные таблицы. Количество файлов для каждого шарда суммируется
DistributedSend	Количество подключений к удаленным серверам для отправки данных на вставку в распределенные таблицы (в синхронном и асинхронном режиме)
EphemeralNode	Количество эфемерных узлов в ZooKeeper
FilesystemCacheElements	Кешируемые элементы файловой системы (сегменты файлов)
FilesystemCacheReadBuffers	Количество активных кеш-буферов
FilesystemCacheSize	Размер кеша файловой системы в байтах

GlobalThread	Количество потоков в глобальном пуле потоков
GlobalThreadActive	Количество потоков в глобальном пуле потоков, выполняющих задачу
HTTPConnection	Количество подключений к HTTP-серверу
InterserverConnection	Количество подключений других реплик для получения кусков данных
KafkaAssignedPartitions	Количество партиций, которым в данный момент назначены таблицы Kafka
KafkaBackgroundReads	Количество фоновых чтений, осуществляемых в данный момент (заполнение материализованных представлений из Kafka)
KafkaConsumers	Количество активных потребителей (consumers) Kafka
KafkaConsumersInUse	Количество потребителей, которые в данный момент используются прямыми или фоновыми чтениями
KafkaConsumersWithAssignment	Количество активных потребителей Kafka, которым назначены какие-либо партиции
KafkaLibrdkafkaThreads	Количество активных потоков библиотеки librdkafka
KafkaProducers	Количество созданных активных производителей (producers) Kafka
KafkaWrites	Количество запущенных в данный момент записей в Kafka
KeeperAliveConnections	Количество активных подключений
KeeperOutstandingRequets	Количество невыполненных запросов
LocalThread	Количество потоков в локальных пулах потоков. Потоки в локальных пулах потоков берутся из глобального пула потоков
LocalThreadActive	Количество потоков в локальных пулах потоков, выполняющих задачу
MMappedFileBytes	Суммарный размер областей mmapped-файлов (в байтах)
MMappedFiles	Общее количество mmapped-файлов (файлы, отображаемые в память)
MaxDDLEntryID	Максимальный ID обработанной DDL-команды
MaxPushedDDLEntryID	Максимальный ID DDL-команды, отправленной в ZooKeeper
MemoryTracking	Общий объем памяти (в байтах), занятой сервером
Merge	Количество выполняемых фоновых слияний
MySQLConnection	Количество клиентских подключений по протоколу MySQL
NetworkReceive	Количество потоков, получающих данные из сети (учитывается только сетевое взаимодействие, связанное с ClickHouse, а не со сторонними библиотеками)
NetworkSend	Количество потоков, отправляющих данные в сеть (учитывается только сетевое взаимодействие, связанное с ClickHouse, а не со сторонними библиотеками)
OpenFileForRead	Количество файлов, открытых для чтения
OpenFileForWrite	Количество файлов, открытых для записи
PartMutation	Количество мутаций (`ALTER DELETE` / `UPDATE`)
PartsActive	Количество активных кусков данных, используемых в текущих и предстоящих запросах `SELECT`
PartsCommitted	Устаревшая метрика, см. PartsActive

PartsCompact	Количество кусков данных в формате Compact
PartsDeleteOnDestroy	Количество кусков данных, перемещенных на другой диск, которые должны быть удалены собственным деструктором
PartsDeleting	Количество неактивных кусков данных, которые удаляются в данный момент
PartsInMemory	Количество кусков данных в оперативной памяти (in-memory data parts)
PartsOutdated	Количество неактивных кусков данных, которые могут быть использованы только в текущих запросах `SELECT` и удалены после их завершения
PartsPreActive	Количество кусков данных, которые есть в списке data_parts, но не используются для запросов `SELECT`
PartsPreCommitted	Устаревшая метрика, см. PartsPreActive
PartsTemporary	Количество кусков данных, которые генерируются в настоящий момент, их нет в списке data_parts
PartsWide	Количество кусков данных в формате Wide
PendingAsyncInsert	Количество асинхронных вставок, ожидающих сброса
PostgreSQLConnection	Количество клиентских подключений по протоколу PostgreSQL
Query	Количество выполняемых запросов
QueryPreempted	Количество запросов, остановленных и находящихся в режиме ожидания в соответствии с настройкой `priority`
QueryThread	Количество потоков обработки запроса
RWLockActiveReaders	Количество потоков, удерживающих блокировку чтения в таблице RWLock
RWLockActiveWriters	Количество потоков, удерживающих блокировку записи в таблице RWLock
RWLockWaitingReaders	Количество потоков, ожидающих чтения в таблице RWLock
RWLockWaitingWriters	Количество потоков, ожидающих записи в таблицу RWLock
Read	Количество системных вызовов чтения (read, pread, io_getevents, и т.д.), которые в данный момент выполняются
ReadonlyReplica	Количество реплицируемых таблиц, которые в настоящее время находятся в состоянии только для чтения из-за повторной инициализации после потери соединения с ZooKeeper или из-за того, что были созданы без настроенного ZooKeeper
ReplicatedChecks	Количество кусков данных, проверяемых на согласованность
ReplicatedFetch	Количество кусков данных, извлекаемых из реплики
ReplicatedSend	Количество кусков данных, отправляемых на реплики
Revision	Ревизия сервера. Это число увеличивается для каждого релиза или релиз-кандидата, кроме релизов патчей
S3Requests	Количество запросов S3
SendExternalTables	Количество подключений, которые отправляют данные для внешних таблиц на удаленные серверы. Внешние таблицы используются для реализации операторов `GLOBAL IN` и `GLOBAL JOIN` с распределенными подзапросами
SendScalars	Количество подключений, которые отправляют данные для скаляров на удаленные серверы
StorageBufferBytes	Количество байтов в буферах таблиц Buffer

StorageBufferRows	Количество строк в буферах таблиц Buffer
SyncDrainedConnections	Количество соединений, синхронно опустошенных (все данные из которых были синхронно извлечены)
TCPConnection	Количество подключений к TCP-серверу (клиенты с интерфейсом Native), включая подключения сервер-сервер для выполнения распределенных запросов
TablesToDropQueueSize	Количество удаленных таблиц, ожидающих фонового удаления данных
VersionInteger	Версия сервера в виде одного целого числа (например, версия `11.22.33` преобразуется в `11022033`)
Write	Количество системных вызовов записи (write, pwrite, io_getevents, и т.д.), которые в данный момент выполняются
ZooKeeperRequest	Количество запросов к ZooKeeper, выполняющихся в данный момент
ZooKeeperSession	Количество подключений к ZooKeeper. Должно быть не более одного, потому что использование более одного подключения к ZooKeeper может привести к ошибкам из-за отсутствия линеаризуемости (устаревшие чтения), которую допускает модель согласованности ZooKeeper
ZooKeeperWatch	Количество watches (подписок на события) в ZooKeeper

ProfileEvents

Название события	**Описание**
AIORead	Количество операций чтения через интерфейс Linux или FreeBSD AIO
AIOReadBytes	Количество байтов, прочитанных через интерфейс Linux или FreeBSD AIO
AIOWrite	Количество операций записи через интерфейс Linux или FreeBSD AIO
AIOWriteBytes	Количество байтов, записанных через интерфейс Linux или FreeBSD AIO
AggregationHashTablesInitializedAsTwoLevel	Сколько хеш-таблиц было инициализировано как двухуровневые для агрегирования
AggregationPreallocatedElementsInHashTables	Сколько элементов было предварительно выделено в хеш-таблицах для агрегирования
ArenaAllocBytes	Количество байтов, выделенных во внутренней памяти, используемой для небольших объектов
ArenaAllocChunks	Общее количество блоков памяти, выделенных в памяти, используемой для небольших объектов
AsyncInsertBytes	Размер данных (в байтах) асинхронных запросов `INSERT`
AsyncInsertQuery	То же, что и InsertQuery, но только для асинхронных запросов `INSERT`
AsynchronousReadWaitMicroseconds	Время ожидания асинхронного чтения
CachedReadBufferCacheWriteBytes	Количество байтов, записанных из источника (например, remote fs) в кеш файловой системы
CachedReadBufferCacheWriteMicroseconds	Время, затраченное на запись данных в кеш файловой системы
CachedReadBufferReadFromCacheBytes	Количество байтов, прочитанных из кеша файловой системы
CachedReadBufferReadFromCacheMicroseconds	Время чтения из кеша файловой системы

CachedReadBufferReadFromSourceBytes	Количество байтов, прочитанных из источника кеша файловой системы (например, remote fs)
CachedReadBufferReadFromSourceMicroseconds	Время чтения из источника кеша файловой системы (например, из удаленной файловой системы)
CachedWriteBufferCacheWriteBytes	Количество байтов, записанных из источника (например, remote fs) в кеш файловой системы
CachedWriteBufferCacheWriteMicroseconds	Время, затраченное на запись данных в кеш файловой системы
CannotRemoveEphemeralNode	Сколько раз возникала ошибка при попытке удалить эфемерный узел (при этом гарантируется, что сессия истечет и эфемерный узел будет удален, даже если произошла ошибка при попытке удалить его вручную)
CannotWriteToWriteBufferDiscard	Количество трассировок стека (stack traces), выданных профилировщиком запросов или обработчиком сигналов из-за переполнения канала или невозможности записи в канал
CompileExpressionsBytes	Количество байтов, используемых для компиляции выражений
CompileExpressionsMicroseconds	Общее время, затраченное на компиляцию выражений в код LLVM
CompileFunction	Сколько раз инициировалась компиляция сгенерированного кода LLVM (для создания объединенной функции для сложных выражений)
CompiledFunctionExecute	Сколько раз выполнялась скомпилированная функция
CompressedReadBufferBlocks	Количество сжатых блоков (блоков данных, которые сжимаются независимо друг от друга), считанных из сжатых источников (файлов, сети)
CompressedReadBufferBytes	Количество несжатых байтов (количество байтов после декомпрессии), прочитанных из сжатых источников (файлов, сети)
ContextLock	Сколько раз блокировка Context (глобальная блокировка) была получена или предпринята попытка ее получить
CreatedHTTPConnections	Общее количество созданных HTTP-соединений (счетчик увеличивается каждый раз, когда создается соединение)
CreatedLogEntryForMerge	Количество успешно созданных записей в логах о слиянии кусков данных в таблицах ReplicatedMergeTree
CreatedLogEntryForMutation	Количество успешно созданных записей в логах об изменении кусков данных в таблицах ReplicatedMergeTree
CreatedReadBufferDirectIO	Сколько раз буфер чтения с `O_DIRECT` создавался для чтения данных (при выборе среди других методов чтения)
CreatedReadBufferDirectIOFailed	Количество попыток создания буфера чтения с `O_DIRECT` для чтения данных (при выборе среди других методов чтения), когда операционная система этого не разрешала (из-за отсутствия поддержки файловой системы или по другим причинам), после чего использовался обычный метод чтения
CreatedReadBufferMMap	Количество раз, когда был создан буфер чтения с использованием `mmap` для чтения данных (при выборе среди других методов чтения)
CreatedReadBufferMMapFailed	Количество попыток создания буфера чтения с `mmap` для чтения данных (при выборе среди других методов чтения), когда операционная система этого не разрешала (из-за отсутствия поддержки файловой системы или по другим причинам), после чего использовался обычный метод чтения
CreatedReadBufferOrdinary	Сколько раз создавался буфер чтения для обычных файлов
DNSError	Общее количество ошибок при разрешении DNS
DataAfterMergeDiffersFromReplica	Количество раз, когда данные после слияния были не идентичны по байтам данным на других репликах. Причин может быть несколько: ▪ использование более новой версии библиотеки сжатия после обновления сервера;

	▪ использование другого метода сжатия; ▪ алгоритм недетерминированного сжатия (крайне маловероятен); ▪ недетерминированный алгоритм слияния из-за логической ошибки в коде; ▪ повреждение данных в памяти из-за ошибки в коде; ▪ повреждение данных в памяти из-за проблем с оборудованием; ▪ ручная модификация исходных данных после запуска сервера; ▪ ручная модификация контрольных сумм, хранящихся в ZooKeeper; ▪ параметры, связанные с форматом куска данных, такие как `enable_mixed_granularity_parts`, различаются на разных репликах. Сервер успешно обнаружит эту ситуацию и загрузит объединенный кусок данных из реплики, чтобы получить идентичный по байтам результат
DataAfterMutationDiffersFromReplica	Количество раз, когда данные после мутации были не идентичны по байтам данным на других репликах. Помимо причин, описанных в DataAfterMergeDiffersFromReplica, это также возможно из-за недетерминированной мутации
DelayedInserts	Количество раз, когда вставка данных в таблицу MergeTree (запрос `INSERT`) откладывалась из-за большого количества активных кусков данных для партиции
DelayedInsertsMilliseconds	Общее время (в миллисекундах), когда запрос `INSERT` на вставку данных в таблицу MergeTree был отложен из-за большого количества активных кусков данных для партиции
DictCacheKeysExpired	Количество найденных в кеше, но устаревших ключей, которые искались в хранящихся в кеше словарях
DictCacheKeysHit	Количество найденных в кеше ключей, которые искались в хранящихся в кеше словарях
DictCacheKeysNotFound	Количество ключей, которые искались в словарях, хранящихся в кеше, но не были найдены
DictCacheKeysRequested	Количество ключей, запрошенных из источника данных для словарей, хранящихся в кеше
DictCacheKeysRequestedFound	Количество ключей, запрошенных из источника данных для словарей, которые хранятся в кеше, и найденных в источнике данных
DictCacheKeysRequestedMiss	Количество ключей, запрошенных из источника данных для словарей, которые хранятся в кеше, но не найденных в источнике данных
DictCacheLockReadNs	Время (в наносекундах), затраченное на ожидание блокировки чтения при поиске данных для словарей, хранящихся в кеше
DictCacheLockWriteNs	Время (в наносекундах), затраченное на ожидание блокировки записи при обновлении данных для словарей, хранящихся в кеше
DictCacheRequestTimeNs	Время (в наносекундах), затраченное на запросы к внешним источникам данных для словарей, хранящихся в кеше
DictCacheRequests	Количество массовых запросов к внешним источникам данных для словарей, хранящихся в кеше
DirectorySync	Количество вызовов функции `F_FULLFSYNC` / `fsync` / `fdatasync` для каталогов
DirectorySyncElapsedMicroseconds	Общее время ожидания системного вызова `F_FULLFSYNC` / `fsync` / `fdatasync` для каталогов
DiskReadElapsedMicroseconds	Общее время ожидания чтения системного вызова (включая время чтения из кеша страниц)
DiskWriteElapsedMicroseconds	Общее время ожидания системного вызова записи (включая время записи в кеш страниц)
DistributedConnectionFailAtAll	Общее количество случаев сбоя распределенного подключения после завершения всех попыток
DistributedConnectionFailTry	Общее количество случаев сбоя распределенного подключения с повторной попыткой
DistributedConnectionMissingTable	Сколько раз реплика исключалась из распределенного запроса, потому что она не содержала таблицы, необходимой для запроса

DistributedConnectionStaleReplica	Сколько раз реплика исключалась из распределенного запроса, потому что для некоторой таблицы, необходимой для запроса, задержка репликации превышала настроенное пороговое значение
DistributedDelayedInserts	Сколько раз запрос `INSERT` на вставку данных в распределенную таблицу был отложен из-за большого количества ожидающих обработки байтов
DistributedDelayedInsertsMilliseconds	Общее время (в миллисекундах), когда запрос `INSERT` на вставку данных в распределенную таблицу был отложен из-за большого количества ожидающих обработки байтов
DistributedRejectedInserts	Сколько раз вставка данных (`INSERT`) в распределенную таблицу была отклонена с исключением `Too many bytes` из-за большого количества ожидающих обработки байтов
DistributedSyncInsertionTimeoutExceeded	Сколько раз было превышено время ожидания шардов во время синхронной вставки в распределенную таблицу (`insert_distributed_sync = 1`)
DuplicatedInsertedBlocks	Количество дедупликаций блока данных, вставленного (`INSERT`) в таблицу ReplicatedMergeTree
ExecuteShellCommand	Количество выполненных команд оболочки (shell)
ExternalAggregationCompressedBytes	Количество байтов, записанных на диск для агрегации во внешней памяти
ExternalAggregationMerge	Количество временных файлов, которые были объединены для агрегации во внешней памяти
ExternalAggregationUncompressedBytes	Объем данных (несжатых, перед сжатием), записанных на диск для агрегации во внешней памяти
ExternalAggregationWritePart	Сколько раз временный файл был записан на диск для агрегации во внешней памяти
ExternalDataSourceLocalCacheReadBytes	Количество байтов, прочитанных из буфера локального кеша в RemoteReadBufferCache
ExternalSortMerge	Количество раз, когда временные файлы были объединены для сортировки во внешней памяти
ExternalSortWritePart	Сколько раз временный файл был записан на диск для сортировки во внешней памяти
FailedInsertQuery	Количество невыполненных (неудачных) запросов `INSERT`
FailedQuery	Количество невыполненных (неудачных) запросов
FailedSelectQuery	Количество невыполненных (неудачных) запросов `SELECT`
FileOpen	Количество открытых файлов
FileSegmentCacheWriteMicroseconds	Метрика сегмента файла. Время, затраченное на запись данных в кеш
FileSegmentPredownloadMicroseconds	Метрика сегмента файла. Время, затраченное на предварительную загрузку данных в кеш. Предварительная загрузка — завершение загрузки сегмента файла (после неудачной попытки это сделать) до момента, когда текущий поток был запрошен
FileSegmentReadMicroseconds	Метрика сегмента файла. Время чтения из файла
FileSegmentUsedBytes	Метрика сегмента файла. Сколько байтов было фактически использовано из текущего сегмента файла
FileSegmentWaitReadBufferMicroseconds	Метрика сегмента файла. Время ожидания внутреннего буфера чтения (включая ожидание кеша)
FileSync	Количество вызовов функции `F_FULLFSYNC` / `fsync` / `fdatasync` для файлов
FileSyncElapsedMicroseconds	Общее время ожидания системного вызова `F_FULLFSYNC` / `fsync` / `fdatasync` для файлов
FunctionExecute	Общее количество функций, выполненных системой
HardPageFaults	Количество аппаратных прерываний (отказы страниц, для обработки которых требуется восстановление страниц с диска)

HedgedRequestsChangeReplica	Общее количество раз, когда время ожидания для изменения реплики истекло в хедж-запросах (hedged requests)
IOBufferAllocBytes	Общее количество байтов, выделенных в буферах ввода/вывода
IOBufferAllocs	Общее количество аллокаций в буферах ввода/вывода
InsertQuery	То же, что Query, но только для запросов `INSERT`
InsertQueryTimeMicroseconds	Общее время выполнения запросов `INSERT`
InsertedBytes	Количество байтов (несжатых; для столбцов, как они хранятся в памяти), вставленных через запросы `INSERT` во все таблицы
InsertedCompactParts	Количество кусков данных, вставленных в формате Compact
InsertedInMemoryParts	Количество кусков данных, вставленных в формате InMemory
InsertedRows	Количество строк, вставленных через запросы `INSERT` во все таблицы
InsertedWideParts	Количество кусков данных, вставленных в формате Wide
KafkaBackgroundReads	Количество фоновых чтений, заполняющих материализованные представления из Kafka с момента запуска сервера
KafkaCommitFailures	Количество неудачных коммитов смещений (offsets) в Kafka (обычно это признак дублирования данных)
KafkaCommits	Количество успешных коммитов смещений (offsets) в Kafka (обычно значение должно быть таким же, как KafkaBackgroundReads)
KafkaConsumerErrors	Количество ошибок, о которых сообщила библиотека librdkafka во время выполнения `poll`
KafkaDirectReads	Количество прямых выборок из таблиц Kafka с момента запуска сервера
KafkaMessagesFailed	Количество сообщений Kafka, которые ClickHouse не смог распарсить
KafkaMessagesPolled	Количество сообщений Kafka, отправленных из librdkafka в ClickHouse
KafkaMessagesProduced	Количество сообщений, отправленных в Kafka
KafkaMessagesRead	Количество сообщений Kafka, уже обработанных ClickHouse
KafkaProducerErrors	Количество ошибок при отправке сообщений в Kafka
KafkaProducerFlushes	Количество явных сбросов в Kafka producer
KafkaRebalanceAssignments	Количество назначений партиций (заключительный этап перебалансировки групп потребителей)
KafkaRebalanceErrors	Количество неудачных перебалансировок групп потребителей
KafkaRebalanceRevocations	Количество отзывов партиций (первый этап перебалансировки группы потребителей)
KafkaRowsRead	Количество строк, выделенных из сообщений Kafka
KafkaRowsRejected	Количество выделенных строк, которые впоследствии были отклонены (из-за перебалансировок/ошибок или по аналогичным причинам). Эти строки будут снова использованы после перебалансировки
KafkaRowsWritten	Количество строк, вставленных в таблицы Kafka
KafkaWrites	Количество записей (вставок) в таблицы Kafka
KeeperCommits	Количество успешных коммитов

KeeperCommitsFailed	Количество неудачных коммитов
KeeperLatency	Задержка keeper
KeeperPacketsReceived	Пакеты, полученные сервером keeper
KeeperPacketsSent	Пакеты, отправленные сервером keeper
KeeperReadSnapshot	Количество прочитанных снепшотов (сериализация)
KeeperRequestTotal	Общее количество запросов на сервере keeper
KeeperSaveSnapshot	Количество сохраненных снепшотов
KeeperSnapshotApplys	Количество применений снепшотов
KeeperSnapshotApplysFailed	Количество неудачных попыток применения снепшотов
KeeperSnapshotCreations	Количество созданных снепшотов
KeeperSnapshotCreationsFailed	Количество неудачных попыток создания снепшотов
MMappedFileCacheHits	Количество раз, когда файл был найден в кеше MMap (для метода `mmap`), поэтому не нужно было повторно выполнять `mmap`
MMappedFileCacheMisses	Количество раз, когда файл не был найден в кеше MMap (для метода `mmap`), поэтому нужно было снова выполнить `mmap`
MainConfigLoads	Количество перезагрузок основной конфигурации
MarkCacheHits	Сколько раз запись была найдена в кеше засечек (mark cache), поэтому не нужно было загружать файл засечек
MarkCacheMisses	Количество промахов (cache miss) кеша засечек, который используется в движке MergeTree для более быстрого чтения данных
MemoryOvercommitWaitTimeMicroseconds	Общее время ожидания освобождения памяти в OvercommitTracker
Merge	Количество запущенных фоновых слияний
MergeTreeDataProjectionWriterBlocks	Количество блоков, вставленных через запрос `INSERT` в проекцию таблиц MergeTree. Каждый блок формирует кусок данных нулевого уровня
MergeTreeDataProjectionWriterBlocksAlreadySorted	Количество блоков, вставленных через запрос `INSERT` в проекцию таблиц MergeTree, которые оказались уже отсортированными
MergeTreeDataProjectionWriterCompressedBytes	Количество байтов, записанных в файловую систему для данных, вставленных через запрос `INSERT` в проекцию таблиц MergeTree
MergeTreeDataProjectionWriterRows	Количество строк, вставленных через запрос `INSERT` в проекцию таблиц MergeTree
MergeTreeDataProjectionWriterUncompressedBytes	Количество несжатых байтов (для столбцов, как они хранятся в памяти), вставленных через запрос `INSERT` в проекцию таблиц MergeTree
MergeTreeDataWriterBlocks	Количество блоков, вставленных через запросы `INSERT` в таблицы MergeTree. Каждый блок формирует кусок данных нулевого уровня
MergeTreeDataWriterBlocksAlreadySorted	Количество блоков, вставленных через запросы `INSERT` в таблицы MergeTree, которые уже отсортированы
MergeTreeDataWriterCompressedBytes	Количество записанных в файловую систему байтов для данных, вставленных через запросы `INSERT` в таблицы MergeTree
MergeTreeDataWriterRows	Количество строк, вставленных через запросы `INSERT` в таблицы MergeTree

MergeTreeDataWriterUncompressedBytes	Количество несжатых байтов (для столбцов, как они хранятся в памяти), вставленных через запросы `INSERT` в таблицы MergeTree
MergeTreeMetadataCacheDelete	Количество операций `Delete` rocksdb (используется для кеша метаданных MergeTree)
MergeTreeMetadataCacheGet	Количество операций `Get` rocksdb (используется для кеша метаданных MergeTree)
MergeTreeMetadataCacheHit	Сколько раз выполнялось чтение метафайла из кеша метаданных MergeTree
MergeTreeMetadataCacheMiss	Сколько раз чтение метафайла из кеша метаданных MergeTree не выполнилось
MergeTreeMetadataCachePut	Количество операций `Put` rocksdb (используется для кеша метаданных MergeTree)
MergeTreeMetadataCacheSeek	Количество операций `Seek` rocksdb (используется для кеша метаданных MergeTree)
MergedIntoCompactParts	Количество кусков данных, объединенных в формате Compact
MergedIntoInMemoryParts	Количество кусков данных, объединенных в формате InMemory
MergedIntoWideParts	Количество кусков данных, объединенных в формате Wide
MergedRows	Количество строк, считанных для фоновых слияний, то есть количество строк перед слиянием
MergedUncompressedBytes	Количество несжатых байтов (для столбцов, как они хранятся в памяти), которые были считаны для фонового слияния, то есть число байтов до слияния
MergesTimeMilliseconds	Общее время, затраченное на фоновые слияния
NetworkReceiveBytes	Общее количество байтов, полученных из сети (учитывается только сетевое взаимодействие, связанное с ClickHouse, а не со сторонними библиотеками)
NetworkReceiveElapsedMicroseconds	Общее время получения данных или ожидания получения данных из сети (учитывается только сетевое взаимодействие, связанное с ClickHouse, а не со сторонними библиотеками)
NetworkSendBytes	Общее количество байтов, отправленных в сеть (учитывается только сетевое взаимодействие, связанное с ClickHouse, а не со сторонними библиотеками)
NetworkSendElapsedMicroseconds	Общее время отправки данных или ожидания отправки данных в сеть (учитывается только сетевое взаимодействие, связанное с ClickHouse, а не со сторонними библиотеками)
NotCreatedLogEntryForMerge	Сколько раз запись в логе об объединении кусков данных в ReplicatedMergeTree не создалась из-за одновременного обновления лога другой репликой
NotCreatedLogEntryForMutation	Сколько раз запись в логе об изменении куска данных в ReplicatedMergeTree не создалась из-за одновременного обновления лога другой репликой
OSCPUVirtualTimeMicroseconds	Процессорное время, затраченное на выполнение задач, с точки зрения операционной системы. Не включает вынужденное ожидание из-за виртуализации
OSCPUWaitMicroseconds	Общее время, в течение которого поток был готов к выполнению, но ожидал распределения на процессорное ядро операционной системой
OSIOWaitMicroseconds	Общее время, затраченное потоком на ожидание результата операции ввода-вывода, в пространстве операционной системы. Это настоящий ввод-вывод, который не включает кеш страниц
OSReadBytes	Количество байтов, прочитанных с дисков или блочных устройств. Не включает байты, прочитанные из кеша страниц. Может включать избыточные данные из-за размера блока, упреждающего чтения и т.д.
OSReadChars	Количество байтов, прочитанных из файловой системы, включая кеш страниц
OSWriteBytes	Количество байтов, записанных на диски или блочные устройства. Не включает байты, которые находятся в грязных страницах кеша страниц. Может не включать данные, которые были записаны операционной системой асинхронно
OSWriteChars	Количество байтов, записанных в файловую систему, включая кеш страниц

ObsoleteReplicatedParts	Количество раз, когда кусок данных был покрыт другим куском данных, полученным из реплики (таким образом, покрытый кусок данных помечается как устаревший и больше не используется)
OpenedFileCacheHits	Количество раз, когда файл был найден в кеше открытых файлов, поэтому не нужно было открывать его снова
OpenedFileCacheMisses	Количество промахов кеша открытых файлов
OtherQueryTimeMicroseconds	Общее время выполнения всех запросов кроме `SELECT` и `INSERT`
OverflowAny	Количество раз, когда `GROUP BY` выполнялось приближенно: когда агрегирование выполнялось только поверх первых уникальных ключей `max_rows_to_group_by`, а другие ключи игнорировались из-за настройки `group_by_overflow_mode = any`
OverflowBreak	Количество раз, когда обработка данных отменялась из-за ограничения сложности запроса настройкой `overflow_mode = break` и был получен неполный результат
OverflowThrow	Количество раз, когда обработка данных отменялась из-за ограничения сложности запроса настройкой `overflow_mode = throw` и выбрасывалось исключение
PerfAlignmentFaults	Количество ошибок выравнивания. Такие ошибки возникают, когда происходит не выровненный доступ к памяти (возможно только на некоторых архитектурах, никогда на x86). Ядро может обрабатывать их, но это снижает производительность
PerfBranchInstructions	Устаревшие команды ветвления. До Linux 2.6.35 использовалось неправильное событие на процессорах AMD
PerfBranchMisses	Неправильно предсказанные команды ветвления
PerfBusCycles	Количество циклов шины, которое может отличаться от общего количества циклов
PerfCacheMisses	Промахи кеша. Обычно эта метрика указывает на промахи кеша последнего уровня и предназначена для использования в сочетании с событием PerfCacheReferences для расчета частоты промахов кеша
PerfCacheReferences	Доступ к кешу. Обычно эта метрика указывает на доступ к кешу последнего уровня, но может варьироваться в зависимости от процессора. Метрика может включать в себя предварительные выборки и сообщения согласованности, что тоже зависит от процессора
PerfContextSwitches	Количество переключений контекста
PerfCpuClock	Счетчик времени CPU, таймер с высоким разрешением для каждого CPU
PerfCpuCycles	Общее количество циклов CPU. Будьте осторожны с тем, что происходит во время масштабирования частоты процессора
PerfCpuMigrations	Сколько раз процесс был мигрирован на новый CPU
PerfDataTLBMisses	Количество промахов data-TLB (буфер для обрабатываемых данных)
PerfDataTLBReferences	Количество раз, когда адрес был найден в data-TLB
PerfEmulationFaults	Количество ошибок эмуляции. Ядро иногда перехватывает невыполненные инструкции и эмулирует их для пользовательского пространства. Это может негативно сказаться на производительности
PerfInstructionTLBMisses	Количество промахов instruction-TLB (буфер для страниц, содержащих исполняемый код)
PerfInstructionTLBReferences	Количество раз, когда адрес был найден в instruction-TLB
PerfInstructions	Устаревшие команды. Будьте внимательны, на них могут влиять различные проблемы, в первую очередь счетчики аппаратных прерываний
PerfLocalMemoryMisses	Ошибки чтения памяти локального узла NUMA
PerfLocalMemoryReferences	Количество чтений памяти локального узла NUMA

PerfMinEnabledRunningTime	Время выполнения события с минимальным разрешенным временем. Используется для отслеживания количества мультиплексирования событий
PerfMinEnabledTime	Для всех событий — минимальное время, в течение которого событие отслеживания было активировано. Используется для отслеживания влияния мультиплексирования событий
PerfRefCpuCycles	Общее количество циклов, не зависит от масштабирования частоты процессора
PerfStalledCyclesBackend	Количество простаивающих циклов во время завершения инструкций
PerfStalledCyclesFrontend	Количество простаивающих циклов во время передачи инструкций на выполнение
PerfTaskClock	Счетчик тактов для выполняемой задачи
PolygonsAddedToPool	Сколько раз полигон был добавлен в кеш (пул) для функции `pointInPolygon`
PolygonsInPoolAllocatedBytes	Количество байтов для полигонов, добавляемых в кеш (пул) для функции `pointInPolygon`
Query	Количество запросов, которые необходимо интерпретировать и потенциально выполнить. Не включаются запросы, которые не удалось распарсить или которые были отклонены из-за ограничений размера AST, ограничений квот или ограничений на количество одновременно выполняемых запросов. Могут включаться внутренние запросы, инициированные ClickHouse. Подзапросы не учитываются
QueryMaskingRulesMatch	Количество раз, когда правила маскировки запросов были успешно сопоставлены
QueryMemoryLimitExceeded	Количество раз, когда был превышен лимит памяти для запроса
QueryProfilerRuns	Количество запусков QueryProfiler
QueryProfilerSignalOverruns	Количество раз, когда прерывалась обработка сигнала профилировщика запросов из-за переполнения, плюс количество сигналов, которые операционная система не доставила из-за переполнения
QueryTimeMicroseconds	Общее время выполнения всех запросов
RWLockAcquiredReadLocks	Сколько раз блокировка чтения была получена в RWLock
RWLockAcquiredWriteLocks	Сколько раз блокировка записи была получена (в нагруженной RWLock)
RWLockReadersWaitMilliseconds	Общее время (в миллисекундах), в течение которого потоки на чтение ждали получения блокировки в RWLock
RWLockWritersWaitMilliseconds	Общее время (в миллисекундах), затраченное на ожидание получения блокировки записи (в нагруженной RWLock)
ReadBackoff	Сколько раз количество потоков обработки запросов уменьшалось из-за медленного чтения
ReadBufferFromFileDescriptorRead	Количество операций чтения (read/pread) из файлового дескриптора. Сокеты не включаются
ReadBufferFromFileDescriptorReadBytes	Количество байтов, прочитанных из файловых дескрипторов. Если файл сжат, метрика покажет размер сжатых данных
ReadBufferFromFileDescriptorReadFailed	Сколько раз чтение (read/pread) из файлового дескриптора было неудачным
ReadBufferFromS3Bytes	Количество байтов, прочитанных из S3
ReadBufferFromS3Microseconds	Время, затраченное на чтение из S3
ReadBufferFromS3RequestsErrors	Количество исключений при чтении из S3
ReadBufferSeekCancelConnection	Количество операций поиска, приводящих к новому соединению (s3, http)
ReadCompressedBytes	Количество байтов (до декомпрессии), прочитанных из сжатых источников (файлы, сеть)
RealTimeMicroseconds	Общее время, затраченное на выполнение запросов и других задач

RegexpCreated	Скомпилированные регулярные выражения. Идентичные регулярные выражения компилируются только один раз и кешируются навсегда
RejectedInserts	Сколько раз вставка данных в таблицу MergeTree через запрос `INSERT` была отклонена с исключением `Too many parts` из-за большого количества активных кусков данных для партиции
RemoteFSBuffers	Количество буферов, созданных для асинхронного чтения из удаленной файловой системы
RemoteFSCacheDownloadBytes	Количество байтов, скачанных из удаленного кеша файловой системы
RemoteFSCacheReadBytes	Количество байтов, прочитанных из удаленного кеша файловой системы
RemoteFSCancelledPrefetches	Количество отмененных предварительных загрузок (из-за операции поиска)
RemoteFSLazySeeks	Количество отложенных операций поиска
RemoteFSPrefetchedReads	Количество чтений из буфера для предварительно загружаемых данных
RemoteFSPrefetches	Количество предварительных загрузок, выполненных при асинхронным чтении из удаленной файловой системы
RemoteFSReadBytes	Количество байтов, прочитанных из удаленной файловой системы
RemoteFSReadMicroseconds	Время, затраченное на чтение данных из удаленной файловой системы
RemoteFSSeeks	Общее количество операций поиска для асинхронного буфера
RemoteFSSeeksWithReset	Количество операций поиска, приводящих к новому соединению
RemoteFSUnprefetchedReads	Количество чтений из буфера, из которого данные читаются без предварительной загрузки
RemoteFSUnusedPrefetches	Количество неиспользуемых предварительных загрузок в момент деструкции буфера
ReplicaPartialShutdown	Сколько раз реплицируемая таблица должна деинициализировать свое состояние из-за истечения срока действия сеанса в ZooKeeper. Состояние повторно инициализируется каждый раз, когда ZooKeeper снова доступен
ReplicatedDataLoss	Сколько раз запрашиваемый кусок данных не существовал ни на одной реплике (даже на репликах, которые сейчас отключены). Эти куски данных потеряны. Это нормально из-за асинхронной репликации (если вставки кворума не были включены) — если реплика, на которую записывался кусок данных, вышла из строя, она не будет содержать этот кусок данных, когда снова станет доступной после сбоя
ReplicatedPartChecks	Сколько раз нужно было выполнить расширенный поиск куска данных на репликах или уточнить необходимость в существующем куске данных
ReplicatedPartChecksFailed	Количество раз, когда расширенный поиск куска данных на репликах не дал результата или когда был найден и удален неожидаемый кусок
ReplicatedPartFailedFetches	Сколько раз кусок данных не удалось загрузить из реплики таблицы ReplicatedMergeTree
ReplicatedPartFetches	Сколько раз кусок данных загружался из реплики таблицы ReplicatedMergeTree
ReplicatedPartFetchesOfMerged	Количество раз, когда предпочиталось загружать уже объединеный кусок данных из реплики таблицы ReplicatedMergeTree вместо того, чтобы выполнять слияние самостоятельно (обычно предпочтительно выполнять слияние самостоятельно для экономии сетевого трафика). Это происходит, когда нет всех исходных кусков данных для выполнения слияния или когда кусок данных достаточно старый
ReplicatedPartMerges	Количество раз, когда куски данных таблиц ReplicatedMergeTree были успешно объединены
ReplicatedPartMutations	Количество раз, когда куски данных таблиц ReplicatedMergeTree были успешно изменены
S3ReadBytes	Количество байтов, прочитанных из хранилища S3

S3ReadMicroseconds	Время выполнения запросов `GET` и `HEAD` к хранилищу S3
S3ReadRequestsCount	Количество запросов `GET` и `HEAD` к хранилищу S3
S3ReadRequestsErrors	Количество ошибок без регулирования в запросах `GET` и `HEAD` к хранилищу S3
S3ReadRequestsRedirects	Количество редиректов в запросах `GET` и `HEAD` к хранилищу S3
S3ReadRequestsThrottling	Количество ошибок 429 и 503 в запросах `GET` и `HEAD` к хранилищу S3
S3WriteBytes	Количество байтов, записанных в хранилище S3
S3WriteMicroseconds	Время выполнения запросов `POST`, `DELETE`, `PUT` и `PATCH` к хранилищу S3
S3WriteRequestsCount	Количество запросов `POST`, `DELETE`, `PUT` и `PATCH` к хранилищу S3
S3WriteRequestsErrors	Количество ошибок без регулирования в запросах `POST`, `DELETE`, `PUT` и `PATCH` к хранилищу S3
S3WriteRequestsRedirects	Количество редиректов в запросах `POST`, `DELETE`, `PUT` и `PATCH` к хранилищу S3
S3WriteRequestsThrottling	Количество ошибок 429 и 503 в запросах `POST`, `DELETE`, `PUT` и `PATCH` к хранилищу S3
ScalarSubqueriesCacheMiss	Сколько раз чтение из скалярного подзапроса не кешировалось и должно было быть полностью вычислено
ScalarSubqueriesGlobalCacheHit	Сколько раз чтение из скалярного подзапроса выполнялось с использованием глобального кеша
ScalarSubqueriesLocalCacheHit	Сколько раз чтение из скалярного подзапроса выполнялось с использованием локального кеша
SchemaInferenceCacheEvictions	Сколько раз схема была удалена из кеша из-за переполнения
SchemaInferenceCacheHits	Сколько раз схема из кеша использовалась для вывода схемы
SchemaInferenceCacheInvalidations	Сколько раз схема в кеше становилась недействительной из-за изменений в данных
SchemaInferenceCacheMisses	Сколько раз схема не находилась в кеше при выводе схемы
Seek	Количество вызовов функции `lseek`
SelectQuery	То же, что Query, но только для запросов `SELECT`
SelectQueryTimeMicroseconds	Общее время выполнения запросов `SELECT`
SelectedBytes	Количество байтов (несжатых; для столбцов, как они хранятся в памяти), полученных через запрос `SELECT` из всех таблиц
SelectedMarks	Количество засечек (гранул), выбранных для чтения из таблицы MergeTree
SelectedParts	Количество кусков данных, выбранных для чтения из таблицы MergeTree
SelectedRanges	Количество несмежных диапазонов во всех кусках данных, выбранных для чтения из таблицы MergeTree
SelectedRows	Количество строк, полученных через запрос `SELECT` из всех таблиц
SleepFunctionCalls	Количество вызовов функции сна (`sleep`, `sleepEachRow`)
SleepFunctionMicroseconds	Время, затраченное на режим сна из-за вызова функции сна
SlowRead	Количество операций чтения из файла, которые были медленными. Указывает на перегрузку системы. Пороговые значения управляются настройками `read_backoff_*`
SoftPageFaults	Количество программных прерываний (отказы страниц, которые могут быть устранены без загрузки страниц с диска)

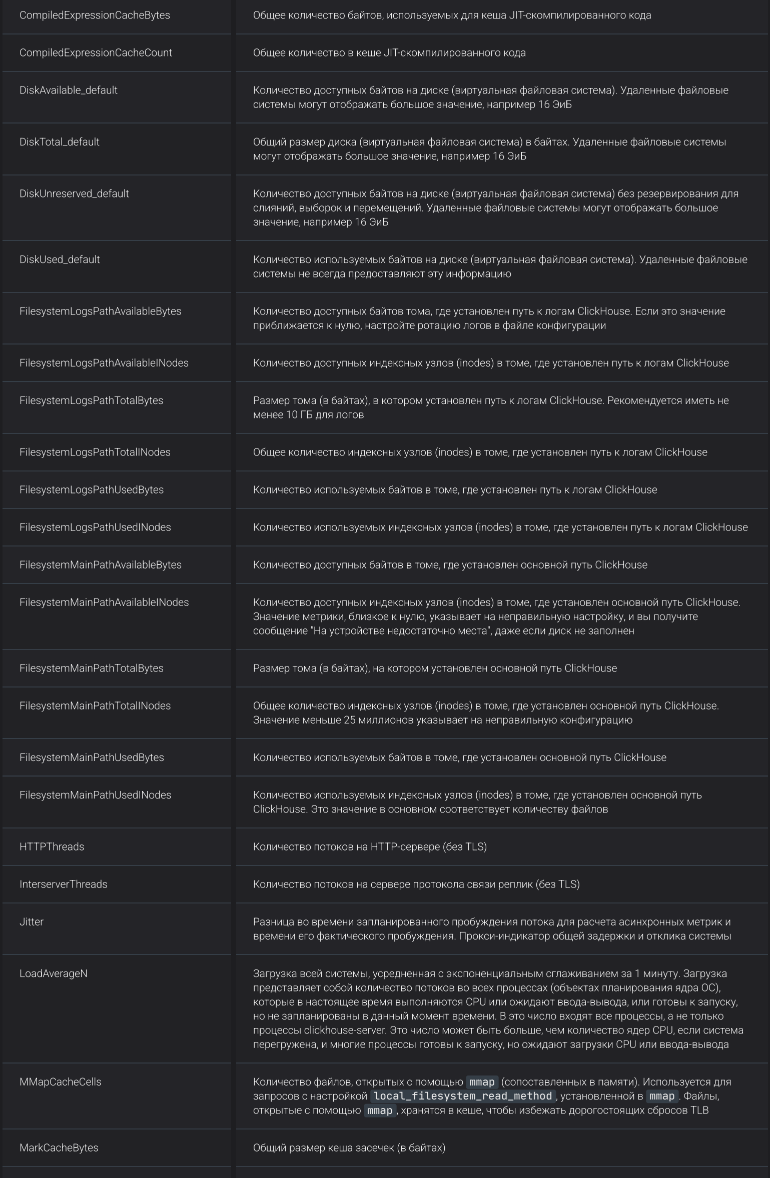
StorageBufferErrorOnFlush	Количество раз, когда буфер в таблице Buffer не мог быть очищен из-за ошибки записи в таблице назначения
StorageBufferFlush	Сколько раз буфер в таблице Buffer очищался
StorageBufferLayerLockReadersWaitMilliseconds	Время ожидания буферного слоя во время чтения
StorageBufferLayerLockWritersWaitMilliseconds	Время ожидания свободного буферного слоя для записи (можно использовать для настройки буферных слоев)
StorageBufferPassedAllMinThresholds	Количество раз, когда был достигнут критерий минимальных порогов для очистки буфера в таблице Buffer
StorageBufferPassedBytesFlushThreshold	Количество раз, когда было достигнуто пороговое значение сброса только в фоновом режиме (в байтах), чтобы очистить буфер в таблице Buffer (метрика для экспертов)
StorageBufferPassedBytesMaxThreshold	Количество раз, когда был достигнут порог максимального количества байтов для очистки буфера в таблице Buffer
StorageBufferPassedRowsFlushThreshold	Количество раз, когда было достигнуто пороговое значение сброса только в фоновом режиме (в строках), чтобы очистить буфер в таблице Buffer (метрика для экспертов)
StorageBufferPassedRowsMaxThreshold	Количество раз, когда был достигнут порог максимального количества строк для очистки буфера в таблице Buffer
StorageBufferPassedTimeFlushThreshold	Количество раз, когда было достигнуто пороговое значение сброса только в фоновом режиме по времени, чтобы очистить буфер в таблице Buffer (метрика для экспертов)
StorageBufferPassedTimeMaxThreshold	Количество раз, когда был достигнут максимальный порог времени для очистки буфера в таблице Buffer
SystemTimeMicroseconds	Общее время, затраченное на обработку (запросы и другие задачи) потоков, выполняющих инструкции CPU в пространстве ядра ОС. Включается время, когда конвейер CPU был остановлен из-за неудачных обращений в кеш, неправильных прогнозирований ветвления, гиперпоточности и т.д.
TableFunctionExecute	Сколько раз выполнялись табличные функции
ThreadPoolReaderPageCacheHit	Сколько раз чтение внутри ThreadPoolReader выполнялось из кеша страницы
ThreadPoolReaderPageCacheHitBytes	Количество байтов, прочитанных внутри ThreadPoolReader, когда это было сделано из кеша страниц
ThreadPoolReaderPageCacheHitElapsedMicroseconds	Время, затраченное на чтение данных из кеша страниц в ThreadPoolReader
ThreadPoolReaderPageCacheMiss	Сколько раз чтение в ThreadPoolReader не выполнилось из кеша страниц и передалось в пул потоков
ThreadPoolReaderPageCacheMissBytes	Количество байтов, прочитанных в ThreadPoolReader, когда чтение не было выполнено из кеша страниц и было передано в пул потоков
ThreadPoolReaderPageCacheMissElapsedMicroseconds	Время, затраченное на чтение данных внутри асинхронного задания (job) в ThreadPoolReader, когда чтение не было выполнено из кеша страниц
ThreadpoolReaderReadBytes	Количество байтов, прочитанных из задания пула потоков при асинхронном чтении
ThreadpoolReaderTaskMicroseconds	Время, затраченное на получение данных при асинхронном чтении
ThrottlerSleepMicroseconds	Общее время бездействия запроса для соответствия всем параметрам регулирования
UncompressedCacheHits	Количество раз, когда блок данных был найден в несжатом кеше (декомпрессии удалось избежать)
UncompressedCacheMisses	Количество раз, когда блок данных не был найден в несжатом кеше (требовалась декомпрессия)
UncompressedCacheWeightLost	Количество байтов, удаленных из несжатого кеша

UserTimeMicroseconds	Общее время, затраченное на обработку (запросы и другие задачи) потоков, выполняющих инструкции CPU в пользовательском пространстве. Включается время, когда конвейер CPU был остановлен из-за неудачных обращений в кеш, неправильных прогнозирований ветвления, гиперпоточности и т.д.
WriteBufferFromFileDescriptorWrite	Количество операций записи (write/pwrite) в файловый дескриптор. Сокеты не включаются
WriteBufferFromFileDescriptorWriteBytes	Количество байтов, записанных в файловые дескрипторы. Если файл сжат, метрика покажет размер сжатых данных
WriteBufferFromFileDescriptorWriteFailed	Количество неудачных попыток записи (write/pwrite) в дескриптор файла
WriteBufferFromS3Bytes	Количество байтов, записанных в S3
ZooKeeperBytesReceived	Общее количество байтов, полученных от ZooKeeper
ZooKeeperBytesSent	Общее количество байтов, отправленных в ZooKeeper
ZooKeeperCheck	Количество запросов `check` к ZooKeeper. Обычно они не имеют смысла по отдельности, только как часть сложной транзакции
ZooKeeperClose	Количество раз, когда соединение с ZooKeeper было закрыто добровольно
ZooKeeperCreate	Сколько раз узел был создан в ZooKeeper
ZooKeeperExists	Сколько раз выполнялась проверка существования узла в ZooKeeper
ZooKeeperGet	Сколько раз данные были получены из ZooKeeper
ZooKeeperHardwareExceptions	Количество исключений при работе с ZooKeeper, связанных с сетью (потеря соединения и т.д.)
ZooKeeperInit	Количество инициализаций сеанса ZooKeeper
ZooKeeperList	Сколько раз команда `list` выполнялась в ZooKeeper
ZooKeeperMulti	Сколько раз команда `multi` выполнялась в ZooKeeper
ZooKeeperOtherExceptions	Количество исключений при работе с ZooKeeper, кроме ZooKeeperUserExceptions и ZooKeeperHardwareExceptions
ZooKeeperRemove	Количество запросов `remove` в ZooKeeper
ZooKeeperSet	Количество запросов `set` в ZooKeeper
ZooKeeperSync	Количество запросов `sync` в ZooKeeper (эти запросы редко нужны или используются)
ZooKeeperTransactions	Количество транзакций, выполненных в ZooKeeper
ZooKeeperUserExceptions	Количество исключений при работе с ZooKeeper, связанных с данными (нет узла, неподходящая версия и т.д.)
ZooKeeperWaitMicroseconds	Общее время ожидания ZooKeeper (в микросекундах)
ZooKeeperWatchResponse	Количество раз, когда уведомление о просмотре было получено от ZooKeeper

AsynchronousMetrics

Название метрики	**Описание**
AsynchronousMetricsCalculationTimeSpent	Время в секундах, затраченное на расчет асинхронных метрик

BlockActiveTime_vda	Время в секундах, в течение которого блочное устройство ставило запросы ввода-вывода в очередь. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockDiscardBytes_vda	Количество отброшенных байтов на блочном устройстве. Эти операции актуальны для SSD. Операции отмены не используются ClickHouse, но могут использоваться другими процессами в системе. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockDiscardMerges_vda	Количество операций отмены, запрошенных с блочного устройства и объединенных вместе планировщиком ввода-вывода операционной системы. Эти операции актуальны для SSD. Операции отмены не используются ClickHouse, но могут использоваться другими процессами в системе. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockDiscardOps_vda	Количество операций отмены, запрошенных с блочного устройства. Эти операции актуальны для SSD. Операции отмены не используются ClickHouse, но могут использоваться другими процессами в системе. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockDiscardTime_vda	Время (в секундах), затраченное на операции отмены, запрошенные с блочного устройства, суммированное по всем операциям. Эти операции актуальны для SSD. Операции отмены не используются ClickHouse, но могут использоваться другими процессами в системе. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockInFlightOps_vda	Количество запросов ввода-вывода, отправленных драйверу устройства, но еще не выполненных. Сюда не входят запросы ввода-вывода, находящиеся в очереди, но еще не переданные драйверу устройства. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockQueueTime_vda	Время ожидания запросов ввода-вывода на этом блочном устройстве (в миллисекундах). Если имеется несколько ожидающих запросов ввода-вывода, значение метрики будет увеличиваться как произведение числа миллисекунд, умноженного на количество ожидающих запросов. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockReadBytes_vda	Количество байтов, прочитанных с блочного устройства. Это значение может быть меньше, чем количество байтов, прочитанных из файловой системы, из-за использования кеша страниц операционной системы, который сохраняет операции ввода-вывода. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockReadMerges_vda	Количество операций чтения, запрошенных с блочного устройства и объединенных вместе планировщиком ввода-вывода операционной системы. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockReadOps_vda	Количество операций чтения, запрошенных с блочного устройства. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockReadTime_vda	Время (в секундах), затраченное на операции чтения, запрошенные с блочного устройства, суммированное по всем операциям. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockWriteBytes_vda	Количество байтов, записанных на блочное устройство. Это значение может быть меньше, чем количество байтов, записанных в файловую систему, из-за использования кеша страниц операционной системы, который сохраняет операции ввода-вывода. Запись на блочное устройство может произойти позже, чем соответствующая запись в файловую систему, из-за сквозного кеширования. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockWriteMerges_vda	Количество операций записи, запрошенных с блочного устройства и объединенных вместе планировщиком ввода-вывода операционной системы. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockWriteOps_vda	Количество операций записи, запрошенных с блочного устройства. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
BlockWriteTime_vda	Время (в секундах), затраченное на операции записи, запрошенные с блочного устройства, суммированное по всем операциям. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
CPUFrequencyMHz_N	Текущая частота процессора (в МГц). Большинство современных процессоров динамически регулируют частоту для энергосбережения и ускорения Turbo Boost



CompiledExpressionCacheBytes	Общее количество байтов, используемых для кеша JIT-скомпилированного кода
CompiledExpressionCacheCount	Общее количество в кеше JIT-скомпилированного кода
DiskAvailable_default	Количество доступных байтов на диске (виртуальная файловая система). Удаленные файловые системы могут отображать большое значение, например 16 ЭиБ
DiskTotal_default	Общий размер диска (виртуальная файловая система) в байтах. Удаленные файловые системы могут отображать большое значение, например 16 ЭиБ
DiskUnreserved_default	Количество доступных байтов на диске (виртуальная файловая система) без резервирования для слияний, выборок и перемещений. Удаленные файловые системы могут отображать большое значение, например 16 ЭиБ
DiskUsed_default	Количество используемых байтов на диске (виртуальная файловая система). Удаленные файловые системы не всегда предоставляют эту информацию
FilesystemLogsPathAvailableBytes	Количество доступных байтов тома, где установлен путь к логам ClickHouse. Если это значение приближается к нулю, настройте ротацию логов в файле конфигурации
FilesystemLogsPathAvailableINodes	Количество доступных индексных узлов (inodes) в томе, где установлен путь к логам ClickHouse
FilesystemLogsPathTotalBytes	Размер тома (в байтах), в котором установлен путь к логам ClickHouse. Рекомендуется иметь не менее 10 ГБ для логов
FilesystemLogsPathTotalINodes	Общее количество индексных узлов (inodes) в томе, где установлен путь к логам ClickHouse
FilesystemLogsPathUsedBytes	Количество используемых байтов в томе, где установлен путь к логам ClickHouse
FilesystemLogsPathUsedINodes	Количество используемых индексных узлов (inodes) в томе, где установлен путь к логам ClickHouse
FilesystemMainPathAvailableBytes	Количество доступных байтов в томе, где установлен основной путь ClickHouse
FilesystemMainPathAvailableINodes	Количество доступных индексных узлов (inodes) в томе, где установлен основной путь ClickHouse. Значение метрики, близкое к нулю, указывает на неправильную настройку, и вы получите сообщение "На устройстве недостаточно места", даже если диск не заполнен
FilesystemMainPathTotalBytes	Размер тома (в байтах), на котором установлен основной путь ClickHouse
FilesystemMainPathTotalINodes	Общее количество индексных узлов (inodes) в томе, где установлен основной путь ClickHouse. Значение меньше 25 миллионов указывает на неправильную конфигурацию
FilesystemMainPathUsedBytes	Количество используемых байтов в томе, где установлен основной путь ClickHouse
FilesystemMainPathUsedINodes	Количество используемых индексных узлов (inodes) в томе, где установлен основной путь ClickHouse. Это значение в основном соответствует количеству файлов
HTTPThreads	Количество потоков на HTTP-сервере (без TLS)
InterserverThreads	Количество потоков на сервере протокола связи реплик (без TLS)
Jitter	Разница во времени запланированного пробуждения потока для расчета асинхронных метрик и времени его фактического пробуждения. Прокси-индикатор общей задержки и отклика системы
LoadAverageN	Загрузка всей системы, усредненная с экспоненциальным сглаживанием за 1 минуту. Загрузка представляет собой количество потоков во всех процессах (объектах планирования ядра ОС), которые в настоящее время выполняются CPU или ожидают ввода-вывода, или готовы к запуску, но не запланированы в данный момент времени. В это число входят все процессы, а не только процессы clickhouse-server. Это число может быть больше, чем количество ядер CPU, если система перегружена, и многие процессы готовы к запуску, но ожидают загрузки CPU или ввода-вывода
MMapCacheCells	Количество файлов, открытых с помощью `mmap` (сопоставленных в памяти). Используется для запросов с настройкой `local_filesystem_read_method`, установленной в `mmap`. Файлы, открытые с помощью `mmap`, хранятся в кеше, чтобы избежать дорогостоящих сбросов TLB
MarkCacheBytes	Общий размер кеша засечек (в байтах)

MarkCacheFiles	Общее количество файлов засечек, закешированных в кеше засечек
MaxPartCountForPartition	Максимальное количество кусков данных на партицию по всем партициям всех таблиц семейства MergeTree. Значения больше `300` указывают на неправильную конфигурацию, перегрузку или массовую загрузку данных
MemoryCode	Объем виртуальной памяти (в байтах), сопоставляемой для страниц машинного кода серверного процесса
MemoryDataAndStack	Объем виртуальной памяти (в байтах), сопоставляемой для использования стека и выделенной памяти. Не указано, включает ли он стеки для каждого потока и большую часть выделенной памяти, которая выделяется системным вызовом `mmap`. Эта метрика существует только для полноты. Для мониторинга рекомендуется использовать метрику MemoryResident
MemoryResident	Объем физической памяти (в байтах), используемой серверным процессом
MemoryShared	Объем памяти (в байтах), используемой серверным процессом, который также используется другими процессами. ClickHouse не использует разделяемую память, но часть памяти может быть помечена операционной системой как разделяемая по своим причинам. За этой метрикой можно не следить, она существует только для полноты
MemoryVirtual	Размер виртуального адресного пространства (в байтах), выделенного серверным процессом. Размер виртуального адресного пространства обычно намного превышает потребление физической памяти и не должен использоваться в качестве оценки потребления памяти. Большие значения этой метрики абсолютно нормальны и имеют только технический смысл
NetworkReceiveBytes_eth0	Количество байтов, полученных через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
NetworkReceiveDrop_eth0	Количество пакетов, которые были потеряны при получении через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
NetworkReceiveErrors_eth0	Количество раз, когда произошла ошибка при приеме данных через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
NetworkReceivePackets_eth0	Количество сетевых пакетов, полученных через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
NetworkSendBytes_eth0	Количество байтов, отправленных через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
NetworkSendDrop_eth0	Количество пакетов, которые были потеряны при отправке через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, не только процессы clickhouse-server
NetworkSendErrors_eth0	Количество ошибок при отправке через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, не только процессы clickhouse-server
NetworkSendPackets_eth0	Количество сетевых пакетов, отправленных через сетевой интерфейс. Это общесистемная метрика, она включает в себя все процессы на хост-машине, не только процессы clickhouse-server
NumberOfDatabases	Общее количество баз данных на сервере
NumberOfTables	Общее количество таблиц суммарно по базам данных на сервере, за исключением баз данных, которые не могут содержать таблицы MergeTree. Исключены движки баз данных, которые генерируют набор таблиц на лету, такие как Lazy, MySQL, PostgreSQL, SQlite
OSContextSwitches	Количество переключений контекста (context switch), выполненных системой на хост-компьютере. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы
OSGuestNiceTime	Соотношение времени работы виртуального CPU для гостевых операционных систем под управлением ядра Linux, когда для гостя был установлен более высокий приоритет (см. `man procfs`). Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Эта метрика не относится к ClickHouse, а существует для полноты. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`

OSGuestNiceTimeCPUN	Соотношение времени, затрачиваемого на запуск виртуального CPU для гостевых операционных систем под управлением ядра Linux, когда для гостя был установлен более высокий приоритет (см. `man procfs`). Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Эта метрика не относится к ClickHouse, а существует для полноты. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSGuestNiceTimeNormalized	Значение аналогично OSGuestNiceTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSGuestTime	Соотношение времени работы виртуального процессора для гостевых операционных систем под управлением ядра Linux (см. `man procfs`). Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Эта метрика не относится к ClickHouse, а существует для полноты. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSGuestTimeCPUN	Соотношение времени работы виртуального процессора для гостевых операционных систем под управлением ядра Linux (см. `man procfs`). Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Эта метрика не относится к ClickHouse, а существует для полноты. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSGuestTimeNormalized	Значение аналогично OSGuestTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSIOWaitTime	Соотношение времени, в течение которого ядро CPU не выполняло код, и ядро операционной системы не запускало какой-либо другой процесс на этом CPU, поскольку процессы ожидали ввода-вывода. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSIOWaitTimeCPUN	Соотношение времени, в течение которого ядро CPU не выполняло код, и ядро операционной системы не запускало какой-либо другой процесс на этом CPU, поскольку процессы ожидали ввода-вывода. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSIOWaitTimeNormalized	Значение аналогично OSIOWaitTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSIdleTime	Соотношение времени бездействия ядра CPU (даже не готово к запуску процесса, ожидающего ввода-вывода) в пространстве ядра операционной системы. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Метрика не включает время, когда CPU был недостаточно загружен из-за внутренних причин CPU (загрузки памяти, остановки конвейера, неправильные прогнозы ветвления, запуск другого ядра SMT). Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSIdleTimeCPUN	Соотношение времени бездействия ядра CPU (даже не готово к запуску процесса, ожидающего ввода-вывода) в пространстве ядра операционной системы. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Метрика не включает время, когда CPU был недостаточно загружен из-за внутренних причин CPU (загрузки памяти, остановки конвейера, неправильные прогнозы ветвления, запуск другого ядра SMT). Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSIdleTimeNormalized	Значение аналогично OSIdleTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSInterrupts	Количество прерываний на хост-машине. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server

OSIrqTime	Соотношение времени, затрачиваемого на выполнение запросов аппаратного прерывания на CPU. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Большое число этой метрики может указывать на неправильную конфигурацию оборудования или очень высокую загруженность сети. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSIrqTimeCPUN	Соотношение времени, затрачиваемого на выполнение запросов аппаратного прерывания на CPU. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Большое число этой метрики может указывать на неправильную конфигурацию оборудования или очень высокую загруженность сети. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSIrqTimeNormalized	Значение аналогично OSIrqTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSMemoryAvailable	Объем памяти (в байтах), доступной для использования программами. Значение, близкое к OSMemoryFreePlusCached. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
OSMemoryBuffers	Объем памяти (в байтах), используемой буферами ядра операционной системы. Обычно это значение должно быть небольшим, а большие значения могут указывать на неправильную настройку операционной системы. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
OSMemoryCached	Объем памяти (в байтах), используемой кешем страниц операционной системы. Как правило, почти вся доступная память используется кешем страниц операционной системы — высокие значения этой метрики являются нормальными и ожидаемыми. Это общесистемная метрика, она включает в себя все процессы на хост-машине, не только процессы clickhouse-server
OSMemoryFreePlusCached	Объем свободной памяти плюс кеш-память страниц операционной системы в хост-системе (в байтах). Эта память доступна для использования программами. Значение должно быть очень похоже на OSMemoryAvailable. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
OSMemoryFreeWithoutCached	Объем свободной памяти в хост-системе (в байтах). Значение не включает кеш-память страниц операционной системы. Кеш-память страниц также доступна для использования программами, поэтому значение этой метрики может сбивать с толку (см. также метрики OSMemoryAvailable и OSMemoryFreePlusCached). Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только clickhouse-server
OSMemoryTotal	Общий объем памяти в хост-системе (в байтах)
OSNiceTime	Соотношение времени, в течение которого ядро CPU выполняло код пользовательского пространства с более высоким приоритетом. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSNiceTimeCPUN	Соотношение времени, в течение которого ядро CPU выполняло код пользовательского пространства с более высоким приоритетом. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSNiceTimeNormalized	Значение аналогично OSNiceTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSOpenFiles	Общее количество открытых файлов на хост-машине. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
OSProcessesBlocked	Количество потоков, заблокированных в ожидании завершения ввода-вывода (`man procfs`). Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
OSProcessesCreated	Количество созданных процессов. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server

OSProcessesRunning	Количество выполнимых (запущенных или готовых к запуску) потоков операционной системой. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server
OSSoftIrqTime	Соотношение времени, затрачиваемого на выполнение запросов программного прерывания на CPU. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Большое число этой метрики может указывать на неэффективное программное обеспечение, работающее в системе. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSSoftIrqTimeCPUN	Соотношение времени, затрачиваемого на выполнение запросов программного прерывания на CPU. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Большое число этой метрики может указывать на неэффективное программное обеспечение, работающее в системе. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSSoftIrqTimeNormalized	Значение аналогично OSSoftIrqTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSStealTime	Соотношение времени, затрачиваемого CPU в других операционных системах при работе в виртуализированной среде. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Не все виртуализированные среды предоставляют эту метрику, большинство из них ее не имеют. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSStealTimeCPUN	Соотношение времени, затрачиваемого CPU в других операционных системах при работе в виртуализированной среде. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Не все виртуализированные среды предоставляют эту метрику, большинство из них ее не имеют. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSStealTimeNormalized	Значение аналогично OSStealTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSSystemTime	Соотношение времени, в течение которого ядро CPU выполняло код ядра операционной системы. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSSystemTimeCPUN	Соотношение времени, в течение которого ядро CPU выполняло код ядра операционной системы. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSSystemTimeNormalized	Значение аналогично OSSystemTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и получить средний показатель использования ресурсов
OSThreadsRunnable	Количество выполнимых потоков, как это "видит" планировщик ядра операционной системы
OSThreadsTotal	Общее количество потоков, как это "видит" планировщик ядра ОС
OSUptime	Время работы хост-сервера (машины, на которой работает ClickHouse) в секундах
OSUserTime	Соотношение времени, в течение которого ядро CPU выполняло код пользовательского пространства. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение метрики включает также время, когда CPU был недостаточно загружен из-за внутренних причин CPU (загрузки памяти, остановки конвейера, неправильные прогнозы ветвления, запуск другого ядра SMT). Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`

OSUserTimeCPUN	Соотношение времени, в течение которого ядро CPU выполняло код пользовательского пространства. Это общесистемная метрика, она включает в себя все процессы на хост-машине, а не только процессы clickhouse-server. Значение метрики включает также время, когда CPU был недостаточно загружен из-за внутренних причин CPU (загрузки памяти, остановки конвейера, неправильные прогнозы ветвления, запуск другого ядра SMT). Значение для одного ядра CPU будет находиться в интервале `[0..1]`. Значение для всех ядер CPU рассчитывается как сумма по ним `[0..количество ядер]`
OSUserTimeNormalized	Значение аналогично OSUserTime, но делится на количество ядер CPU, чтобы оно находилось в интервале `[0..1]` независимо от количества ядер. Это позволяет усреднить значения этой метрики по нескольким серверам в кластере, даже если количество ядер неравномерно, и все же получить средний показатель использования ресурсов
ReplicasMaxAbsoluteDelay	Максимальная разница в секундах между самым свежим реплицированным куском данных и самым свежим куском данных, который еще предстоит реплицировать, между реплицируемыми таблицами. Очень высокое значение этой метрики указывает на реплику без данных
ReplicasMaxInsertsInQueue	Максимальное количество операций `INSERT` в очереди (еще предстоит реплицировать) для реплицируемых таблиц
ReplicasMaxMergesInQueue	Максимальное количество операций слияния в очереди (еще не примененных) для реплицируемых таблиц
ReplicasMaxQueueSize	Максимальный размер очереди (по количеству операций, таких как получение, слияние) для реплицируемых таблиц
ReplicasMaxRelativeDelay	Максимальная разница между задержкой реплики и задержкой самой последней реплики той же таблицы для реплицируемых таблиц
ReplicasSumInsertsInQueue	Сумма операций `INSERT` в очереди (еще предстоит реплицировать) для реплицируемых таблиц
ReplicasSumMergesInQueue	Сумма операций слияния в очереди (еще предстоит применить) для реплицируемых таблиц
ReplicasSumQueueSize	Суммарный размер очереди (по количеству операций, таких как получение, слияние) для реплицируемых таблиц
TCPThreads	Количество потоков на сервере протокола TCP (без TLS)
TotalBytesOfMergeTreeTables	Общее количество байтов (сжатых, включая данные и индексы), хранящихся во всех таблицах семейства MergeTree
TotalPartsOfMergeTreeTables	Общее количество кусков данных во всех таблицах семейства MergeTree. Числа больше `10000` негативно повлияют на время запуска сервера и могут свидетельствовать о необоснованном выборе ключа партиционирования
TotalRowsOfMergeTreeTables	Общее количество строк (записей), хранящихся во всех таблицах семейства MergeTree
UncompressedCacheBytes	Общий размер несжатого кеша (в байтах). Несжатый кеш обычно не повышает производительность, и его следует избегать
UncompressedCacheCells	Общее количество записей в несжатом кеше. Каждая запись представляет собой распакованный блок данных. Несжатый кеш обычно не повышает производительность, и его следует избегать
Uptime	Время работы сервера в секундах. Включает время, затрачиваемое на инициализацию сервера перед приемом соединений
jemalloc.arenas.all.dirty_purged	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.arenas.all.muzzy_purged	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.arenas.all.pactive	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.arenas.all.pdirty	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.arenas.all.pmuzzy	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.background_thread.num_runs	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)

jemalloc.background_thread.num_threads	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.background_thread.run_intervals	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.active	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.allocated	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.epoch	Внутренний номер инкрементного обновления статистики jemalloc, используемый во всех других метриках jemalloc
jemalloc.mapped	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.metadata	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.metadata_thp	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.resident	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)
jemalloc.retained	Внутренняя метрика низкоуровневого распределителя памяти (см. jemalloc)

Просмотр метрик мониторинга

Метрики сервера ClickHouse можно посмотреть в системных таблицах system.metrics, system.events и system.asynchronous_metrics.

Пример запроса:

```sql
SELECT * FROM system.metrics LIMIT 5;
```

Вывод на экран:

```
┌─metric──────────┬─value─┬─description─────────────────────────────────────┐
│ Query           │     1 │ Number of executing queries                     │
│ Merge           │     0 │ Number of executing background merges           │
│ PartMutation    │     0 │ Number of mutations (ALTER DELETE/UPDATE)       │
│ ReplicatedFetch │     0 │ Number of data parts being fetched from replica │
│ ReplicatedSend  │     0 │ Number of data parts being sent to replicas     │
└─────────────────┴───────┴─────────────────────────────────────────────────┘
```

Для просмотра метрик можно также использовать веб-интерфейсы Prometheus и Grafana.

Логирование

Елена Дворядкина

Содержание

Назад к содержанию

Лог-файлы

Лог-таблицы

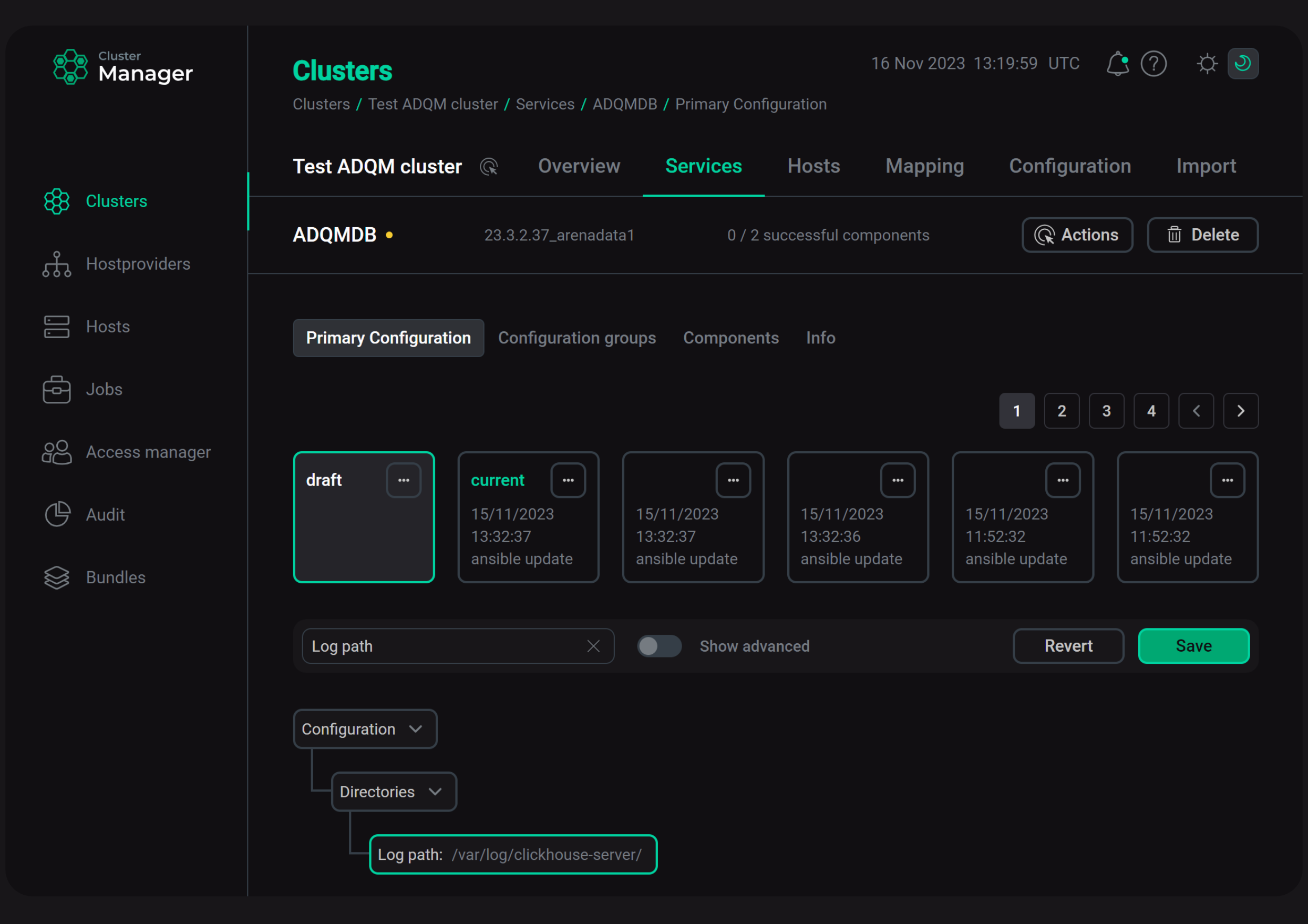
В процессе работы ADQM генерирует текстовые лог-файлы, которые могут быть полезны при анализе причин возникновения различных ошибок. Дополнительно можно включить ведение других специализированных логов (например, собирать информацию о выполнении запросов, зависимых представлениях, потоках, трассировках стека), которые будут сохраняться в системные таблицы ADQM.

Лог-файлы

По умолчанию ADQM записывает следующие логи в каталог */var/log/clickhouse-server*:

- *clickhouse-server.log* — основной лог сервера ADQM (уровень детализации записей настраивается специальным параметром — см. описание параметров логирования ниже);
- *clickhouse-server.err.log* — лог ошибок (содержит записи уровней `warning`, `error`, `critical` и `fatal`).

Изменить каталог хранения лог-файлов можно через параметр ***Log path*** в секции ***Directories*** на странице конфигурации сервиса ADQMDB интерфейса ADCM.



Cluster Manager
Clusters
16 Nov 2023 13:19:59 UTC
Clusters / Test ADQM cluster / Services / ADQMDB / Primary Configuration
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
23.3.2.37_arenadata1
0 / 2 successful components
Actions
Delete
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Primary Configuration
Configuration groups
Components
Info
1
2
3
4
draft
current
15/11/2023 13:32:37 ansible update
15/11/2023 13:32:37 ansible update
15/11/2023 13:32:36 ansible update
15/11/2023 11:52:32 ansible update
15/11/2023 11:52:32 ansible update
Log path
Show advanced
Revert
Save
Configuration
Directories

Log path: /var/log/clickhouse-server/

Установка пути к каталогу, содержащему лог-файлы

На этой же странице в секции ***Log settings*** (секция доступна при включенной опции ***Show advanced***) можно настроить параметры логирования:

- ***Logging level*** — уровень детализации сообщений, записываемых в лог-файл *clickhouse-server.log*. Возможные значения: `trace`, `debug`, `information`, `notice`, `warning`, `error`, `critical`, `fatal`, `none`.
  Уровень логирования для *clickhouse-server.err.log* отдельно не конфигурируется.
- ***Log file size, MB*** — размер лог-файла (в МБ). Как только файл достигает указанного размера, он архивируется и переименовывается, а на его месте создается новый лог-файл.
- ***Log files count*** — количество архивных лог-файлов, которые хранятся в ADQM.

ВАЖНО

После того, как необходимые параметры установлены, не забудьте нажать ***Save*** и выполнить действие ***Reconfig*** для сервиса ADQMDB, чтобы сохранить информацию об изменениях конфигурации.

Чтобы просмотреть содержимое лог-файла, можно использовать следующую команду:

```
$ sudo less /var/log/clickhouse-server/clickhouse-server.log
```

Пример вывода на экран:

```
2023.09.08 10:59:14.549245 [ 31027 ] {} <Information> Application: Will watch for the process with pid 31036
2023.09.08 10:59:14.549357 [ 31036 ] {} <Information> Application: Forked a child process to watch
2023.09.08 10:59:14.549768 [ 31036 ] {} <Information> SentryWriter: Sending crash reports is disabled
2023.09.08 10:59:14.549851 [ 31036 ] {} <Trace> Pipe: Pipe capacity is 1.00 MiB
2023.09.08 10:59:14.674600 [ 31036 ] {} <Information> : Starting ClickHouse 22.3.7.28 with revision 54460, build id:
6F5FF3E355A49AA523CB708AFAC0EB6C5EE73278, PID 31036
2023.09.08 10:59:14.674685 [ 31036 ] {} <Information> Application: starting up
2023.09.08 10:59:14.674703 [ 31036 ] {} <Information> Application: OS name: Linux, version: 3.10.0-1160.66.1.el7.x86_64,
architecture: x86_64
2023.09.08 10:59:14.678336 [ 31036 ] {} <Trace> AsynchronousMetrics: Scanning /sys/class/thermal
2023.09.08 10:59:14.678369 [ 31036 ] {} <Trace> AsynchronousMetrics: Scanning /sys/block
2023.09.08 10:59:14.678516 [ 31036 ] {} <Trace> AsynchronousMetrics: Scanning /sys/devices/system/edac
2023.09.08 10:59:14.678536 [ 31036 ] {} <Trace> AsynchronousMetrics: Scanning /sys/class/hwmon
2023.09.08 10:59:14.981938 [ 31036 ] {} <Warning> Application: Calculated checksum of the binary:
1841F46AA9C864447AE24AC0D983E86C. There is no information about the reference checksum.
2023.09.08 10:59:14.991365 [ 31036 ] {} <Debug> Application: rlimit on number of file descriptors is 500000
2023.09.08 10:59:14.991394 [ 31036 ] {} <Debug> Application: Initializing DateLUT.
2023.09.08 10:59:14.991397 [ 31036 ] {} <Trace> Application: Initialized DateLUT with time zone 'UTC'.
2023.09.08 10:59:14.991426 [ 31036 ] {} <Debug> Application: Setting up /var/lib/clickhouse/tmp/ to store temporary data in it
2023.09.08 10:59:14.993872 [ 31036 ] {} <Debug> Application: Initiailizing interserver credentials.
2023.09.08 10:59:14.994558 [ 31036 ] {} <Debug> ConfigReloader: Loading config '/etc/clickhouse-server/config.xml'
2023.09.08 10:59:14.995658 [ 31036 ] {} <Debug> ConfigReloader: Loaded config '/etc/clickhouse-server/config.xml', performing
update on configuration
...
```

В случае успешного запуска сервера ADQM, лог содержит следующие строки:

- `<Information> Application: starting up` — сервер запускается.
- `<Information> Application: Ready for connections.` — сервер запущен и готов принимать соединения.

Если в конфигурации ошибка и сервер не запустился, в логе будет строка с маркером `<Error>` и описанием ошибки. Например:

```
...
2023.09.08 10:59:14.991426 [ 31036 ] {} <Error> Application: DB::Exception: A setting 'query_profiler_cpu_time_period_ns' appeared
at top level in config /etc/clickhouse-server/config.xml. But it is user-level setting that should be located in users.xml inside
<profiles> section for specific profile. You can add it to <profiles><default> if you want to change default value of this
setting. You can also disable the check - specify <skip_check_for_incorrect_settings>1</skip_check_for_incorrect_settings> in the
main configuration file.
...
```

Если сообщения об ошибке нет в конце файла, просмотрите весь файл начиная со строки `<Information> Application: starting up`.

Лог-таблицы

ADQM поддерживает несколько системных таблиц для хранения различных типов логов. Чтобы начать логирование, необходимо активировать соответствующий параметр в секции ***Log settings*** на странице конфигурации сервиса ADQMDB — после этого ADQM создаст системную таблицу с тем же именем, что и параметр, и начнет записывать в нее соответствующий системный лог. Параметры для включения поддерживаемых типов логирования перечислены в следующей таблице, которая также содержит описание данных, хранящихся в логах.

Типы логов в ADQM

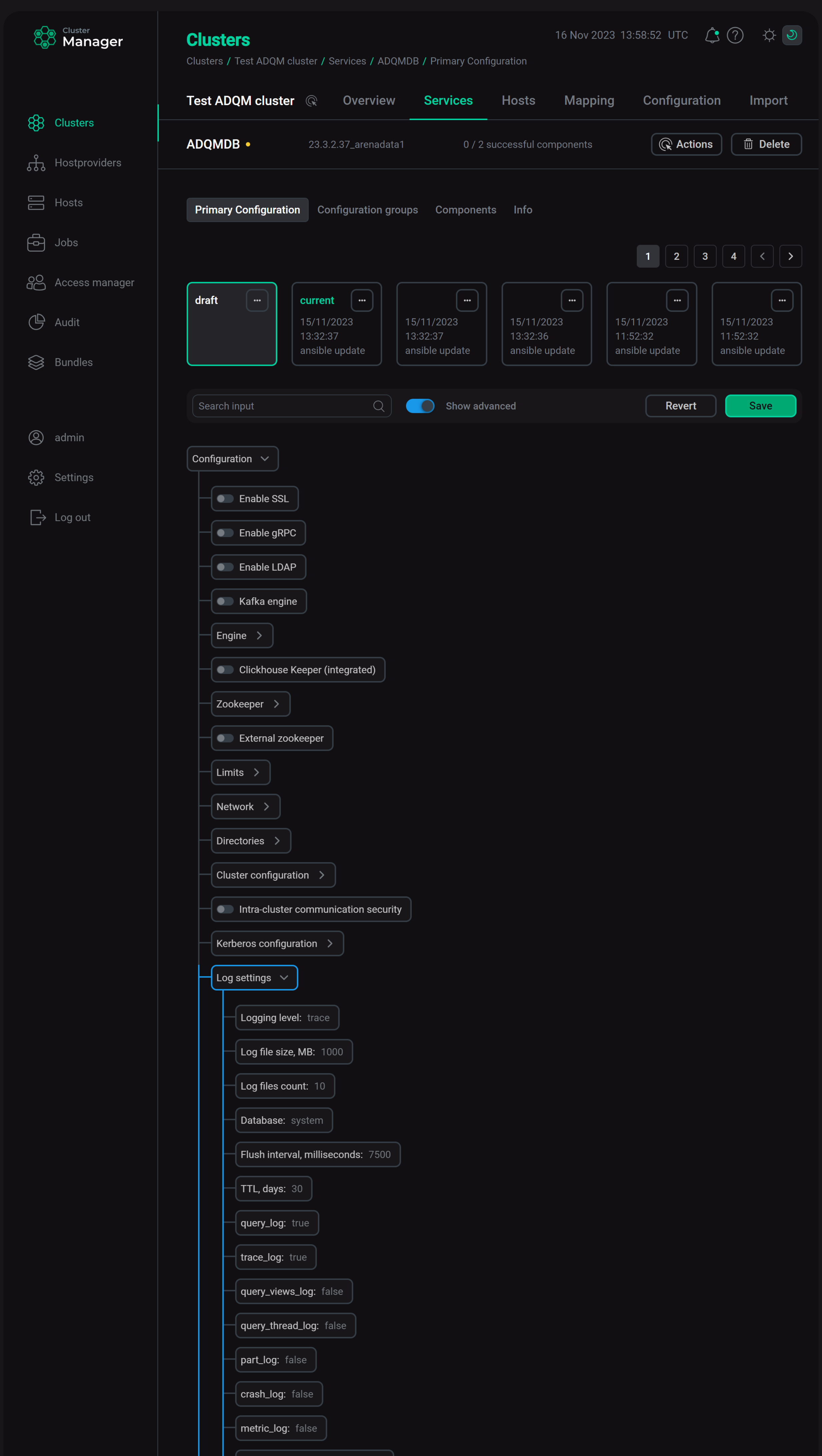
Параметр включения логирования	Лог-таблица	Описание данных лог-таблицы
query_log	query_log	Информация о выполняемых запросах (например, время начала выполнения запроса, длительность обработки, сообщения об ошибках)
trace_log	trace_log	Трассировки стека, собранные профилировщиком запросов (полезны для глубокого анализа производительности)
query_views_log	query_views_log	Информация о зависимых представлениях, выполняемых при обработке запросов, принятых с настройкой `log_query_views = 1` (например, тип представления, продолжительность выполнения представления)
query_thread_log	query_thread_log	Информация о потоках выполнения запросов, принятых с настройкой `log_query_threads = 1` (например, имя потока, время начала и завершения выполнения запроса потоком, продолжительность обработки запроса, объем прочитанных и записанных данных в байтах/строках)
part_log	part_log	Информация о событиях, связанных с данными таблиц семейства MergeTree (например, вставка/объединение/удаление кусков данных, загрузка данных с реплики, перемещение данных между дисками)
crash_log	crash_log	Информация о трассировках стека для фатальных ошибок
session_log	session_log	Информация обо всех успешных и неудачных событиях входа и выхода из системы
metric_log	metric_log	История значений метрик из таблиц `system.metrics` и `system.events`
asynchronous_metric_log	asynchronous_metric_log	История значений метрик из таблицы `system.asynchronous_metrics`
asynchronous_insert_log	asynchronous_insert_log	Информация об асинхронных вставках

ПРИМЕЧАНИЕ

Данные таблиц с системными логами хранятся в файловой системе и не удаляются автоматически. Если удалить таблицу из файловой системы, ADQM снова создаст пустую таблицу во время следующей записи данных в лог. Если схема системной таблицы изменилась при обновлении сервера ADQM, текущая таблица переименовывается и создается новая таблица.

В секции ***Log settings*** можно установить следующие настройки, общие для всех лог-таблиц:

- ***Database*** — база данных, в которой хранятся таблицы;
- ***Flush interval, milliseconds*** — интервал сброса данных из буфера памяти в таблицы (в миллисекундах);
- ***TTL, days*** — время жизни логов (в днях).



Cluster Manager
Clusters
16 Nov 2023 13:58:52 UTC
Clusters / Test ADQM cluster / Services / ADQMDB / Primary Configuration
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
23.3.2.37_arenadata1
0 / 2 successful components
Actions
Delete
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Primary Configuration
Configuration groups
Components
Info
1
2
3
4
draft
current
15/11/2023 13:32:37 ansible update
15/11/2023 13:32:37 ansible update
15/11/2023 13:32:36 ansible update
15/11/2023 11:52:32 ansible update
15/11/2023 11:52:32 ansible update
Search input
Show advanced
Revert
Save
Configuration
Enable SSL
Enable gRPC
Enable LDAP
Kafka engine
Engine
Clickhouse Keeper (integrated)
Zookeeper
External zookeeper
Limits
Network
Directories
Cluster configuration
Intra-cluster communication security
Kerberos configuration
Log settings

Logging level: trace
Log file size, MB: 1000
Log files count: 10
Database: system
Flush interval, milliseconds: 7500
TTL, days: 30
query_log: true
trace_log: true
query_views_log: false
query_thread_log: false
part_log: false
crash_log: false
metric_log: false
asynchronous_metric_log: false

Настройки логирования

Базы данных

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
- Создание базы данных
- Особенности движка Atomic
  - UUID таблиц
  - RENAME TABLE
  - EXCHANGE TABLES/DICTIONARIES
  - DROP/DETACH TABLE

Обзор

Чтобы оптимизировать выполнение специализированных задач, ADQM поддерживает **движки баз данных** различного назначения:

- Atomic — основной движок баз данных ADQM, который используется по умолчанию (см. Особенности движка Atomic ниже);
- MySQL — движок для доступа к внешней базе данных MySQL;
- MaterializedMySQL — экспериментальный движок для репликации базы данных MySQL в ADQM;
- PostgreSQL — движок для доступа к внешней базе данных PostgreSQL;
- MaterializedPostgreSQL — экспериментальный движок для репликации базы данных PostgreSQL в ADQM;
- SQLite — движок для доступа к базе данных SQLite;
- Lazy — движок для работы с таблицами семейства Log, которые сохраняются только в оперативной памяти;
- Replicated — экспериментальная версия движка Atomic с поддержкой репликации полной базы данных.

Полную информацию по всем движкам баз данных можно получить в документации ClickHouse.

По умолчанию в ADQM есть следующие базы данных:

- `system` — база данных с системными таблицами, которые содержат информацию о состоянии сервера и его внутренних процессах, а также системные логи (таблицы `*_log`);
- `INFORMATION_SCHEMA` (`information_schema`) — системная база данных, содержащая представления, с помощью которых можно получить информацию о метаданных объектов базы данных;
- `default` — база данных Atomic, которая создается по умолчанию при установке ADQM (конфигурационный параметр `Default database` сервиса ADQMDB).

Посмотреть список всех баз данных на сервере ADQM можно с помощью запроса SHOW DATABASES:

```
SHOW DATABASES;
```

Пример результата выполнения команды:

```
--name----------------
│ INFORMATION_SCHEMA │
│ default            │
│ information_schema │
│ system             │
│ test_database      │
----------------------
```

Создание базы данных

Для создания базы данных используется запрос CREATE DATABASE:

```
CREATE DATABASE <database_name> [ENGINE = <engine_name>];
```

Если в запросе не указать параметр `ENGINE`, он создаст базу данных Atomic. Ниже перечислены основные особенности движка Atomic.

Особенности движка Atomic

UUID таблиц

У каждой таблицы в базе данных Atomic есть уникальный UUID. Формат UUID — `xxxyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy`.

Данные таблицы хранятся в папке */var/lib/clickhouse/store/xxx/xxxyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy*, на которую указывает ссылка */var/lib/clickhouse/data/<database_name>/<table_name>*, где `<database_name>` — имя базы данных, `<table_name>` — имя таблицы.

За счет того, что имя таблицы является ссылкой в файловой системе, операции `RENAME TABLE` и `DROP TABLE` выполняются быстро и не являются блокирующими. Также доступны атомарные операции `EXCHANGE TABLES` и `EXCHANGE DICTIONARIES`.

RENAME TABLE

Запрос на переименование таблицы RENAME TABLE выполняется без изменения UUID и перемещения табличных данных. Этот запрос не ожидает завершения использующих таблицу запросов и выполняется мгновенно — при задании нового имени таблицы сразу появляется новая ссылка на папку с данными таблицы, после чего старая ссылка удаляется.

EXCHANGE TABLES/DICTIONARIES

Запрос EXCHANGE атомарно обменивает имена между двумя таблицами или двумя словарями (словарь — это хранилище данных типа ключ/значение, которое полностью или частично хранится в оперативной памяти сервера ADQM и может быть использовано в качестве справочника для подстановки значений по ключам в итоговую выборку данных).

Например, вместо неатомарной операции:

```
RENAME TABLE new_table TO tmp, old_table TO new_table, tmp TO old_table;
```

можно использовать один атомарный запрос:

```
EXCHANGE TABLES new_table AND old_table;
```

DROP/DETACH TABLE

Таблицу можно удалить с помощью запроса DROP TABLE или отключить (сделать таблицу "невидимой" для сервера) с помощью запроса DETACH TABLE.

При выполнении запроса `DROP TABLE` данные не удаляются немедленно. Таблица помечается как удаленная (к ней не могут приходить новые запросы), метаданные перемещаются в папку */var/lib/clickhouse/metadata_dropped/*, а база данных уведомляет об этом фоновый поток. Настройка database_atomic_delay_before_drop_table_sec задает задержку перед окончательным удалением данных.

Можно активировать синхронный режим с помощью модификатора `SYNC`. При активации настройки database_atomic_wait_for_drop_and_detach_synchronously этот модификатор добавляется ко всем запросам `DROP` и `DETACH`. В этом случае таблица будет удалена сразу после завершения `SELECT`, `INSERT` и других запросов, которые ее используют.

Движки таблиц

Елена Дворядкина

Содержание

Назад к содержанию

- Семейства движков таблиц
- Семейство MergeTree
  - Особенности таблиц семейства MergeTree
  - Вставка и хранение данных
  - Специальные версии движка MergeTree
  - Движки реплицируемых таблиц

Движок таблицы в ADQM/ClickHouse — это тип таблицы, который определяет:

- место и способ хранения данных;
- поддерживаемые запросы;
- конкурентный доступ к данным;
- возможность использования индексов;
- возможность многопоточного выполнения запроса;
- параметры репликации данных.

Семейства движков таблиц

ADQM/ClickHouse поддерживает множество движков таблиц, которые оптимизированы для решения различных задач. Все движки разбиты на семейства (группы):

- **MergeTree** — наиболее универсальные и функциональные движки таблиц для хранения больших данных и высокопроизводительного поиска. Основное свойство этих движков — быстрая вставка данных с последующей фоновой обработкой. Подробнее в разделе Семейство MergeTree ниже.
- **Log** — простые движки для сценариев, когда необходимо быстро записывать и затем целиком читать множество небольших таблиц (до 1 миллиона строк).
- **Интеграционные** — семейство движков для доступа к внешним системам хранения данных (ODBC, JDBC, MySQL, MongoDB, HDFS, S3, Kafka, EmbeddedRocksDB, RabbitMQ, PostgreSQL). Подробную информацию об этих движках и примеры их использования можно получить в разделе Table Engines for Integrations документации ClickHouse.
- **Специальные** — семейство движков для специальных задач, например:
  - Distributed — движок для распределенной обработки запросов на нескольких серверах (подробнее о создании и использовании таблиц типа Distributed в статье Шардирование).
  - Dictionary — движок, который отображает данные словаря как таблицу.
  - Merge (не путать с движком MergeTree) — движок, который позволяет читать данные одновременно из произвольного количества других таблиц. Таблица Merge не хранит данные самостоятельно, и запись в такую таблицу не поддерживается.
  - Другие движки — подробнее в разделе Special Table Engines документации ClickHouse.

Получить информацию о функциональных возможностях движков таблиц можно с помощью системной таблицы system.table_engines, например:

```sql
SELECT * FROM system.table_engines WHERE name in ('MergeTree', 'TinyLog', 'MySQL');
```

Результат выполнения запроса:

```
--name--------supports_settings---supports_skipping_indices---supports_projections---supports_sort_order---supports_ttl---
supports_replication---supports_deduplication---supports_parallel_insert--
| MySQL     |                 1 |                         0 |                    0 |                   0 |            0 |
0 |                      0 |                        0 |
| MergeTree |                 1 |                         1 |                    1 |                   1 |            1 |
0 |                      0 |                        1 |
| TinyLog   |                 1 |                         0 |                    0 |                   0 |            0 |
0 |                      0 |                        0 |
---------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------
```

Семейство MergeTree

Особенности таблиц семейства MergeTree

MergeTree — основной движок таблиц для работы с большими данными в ADQM. Этот движок, а также все его специальные версии, поддерживают:

- партиционирование, то есть объединение данных таблицы в **партиции** (наборы записей, которыми можно гибко оперировать — отсоединять/присоединять, переносить и т.д.) по какому-либо критерию (например, по месяцу/дню/типу события);
- репликацию данных;
- семплирование данных;
- оптимизированный поиск при помощи разреженного индекса по первичному ключу.

Вставка и хранение данных

Особенности хранения данных в таблицах семейства MergeTree:

- При вставке данные записываются не единым целым, а в виде отдельных частей — **кусков** (data parts). Затем куски поэтапно объединяются между собой в фоновом режиме.
- Данные в каждом куске отсортированы по первичному ключу.
- Данные, относящиеся к разным партициям, разбиваются на разные куски. Куски разных партиций не сливаются между собой.
- Процесс слияния данных управляется системой (система стремится к состоянию, когда каждая партиция содержит всего один отсортированный кусок данных), но его можно форсировать с помощью команды OPTIMIZE.
- Каждый кусок данных логически делится на гранулы. **Гранула** — это минимальный неделимый набор данных, который считывается при выборке данных. Размер гранул задается настройками движка `index_granularity` (максимальное количество строк данных в грануле) и `index_granularity_bytes` (максимальный размер гранулы в байтах).
- Каждый кусок хранится в отдельной директории внутри директории таблицы. Куски данных могут храниться как в колоночном формате (каждый столбец хранится в отдельном файле), так и в компактном (все столбцы хранятся в одном файле).
- В конкретный момент времени в таблице может храниться более одной копии данных — неслитые между собой куски и их объединенный вариант.
- Старые куски, данные которых были слиты в новый, постепенно удаляются. Время хранения неактивных кусков задает табличная настройка `old_parts_lifetime`.
- Кусками можно манипулировать как партициями (например, доступны операции перемещения, отсоединения и другие).

Специальные версии движка MergeTree

Семейство MergeTree включает также следующие движки таблиц, которые наследуют функциональность MergeTree, но имеют свои особенности:

- ReplacingMergeTree — удаляет дублирующиеся записи с одинаковым значением ключа сортировки. Подходит для фоновой чистки дублирующихся данных в целях экономии места, но не дает гарантии отсутствия дубликатов.
- SummingMergeTree — при слиянии кусков данных все строки с одинаковым ключом сортировки заменяются на одну, которая хранит суммы значений столбцов с числовым типом данных.
- AggregatingMergeTree — используется для создания материализованных представлений (materialized views).
- CollapsingMergeTree и VersionedCollapsingMergeTree — добавляют в алгоритм слияния кусков данных логику сворачивания (удаления) строк.
- GraphiteMergeTree — движок для прореживания и агрегирования/усреднения данных Graphite.

Движки реплицируемых таблиц

Для создания реплицируемых таблиц используются движки семейства MergeTree с префиксом `Replicated`:

- ReplicatedMergeTree
- ReplicatedReplacingMergeTree
- ReplicatedSummingMergeTree
- ReplicatedAggregatingMergeTree
- ReplicatedCollapsingMergeTree
- ReplicatedVersionedCollapsingMergeTree
- ReplicatedGraphiteMergeTree

Пример использования движка ReplicatedMergeTree для создания реплицируемых таблиц можно посмотреть в статье Типовой кластер.

Обзор таблиц

Елена Дворядкина

Содержание

Назад к содержанию

- Создание таблиц MergeTree
  - Обзор синтаксиса
  - Примеры
- Просмотр списка таблиц
- Получение информации о таблице
  - Структура таблицы
  - Метаданные таблицы
- Добавление данных в таблицу
- Чтение данных из таблицы
- Модификация таблицы
- Отключение и подключение таблицы
- Удаление таблицы

Основные операции с таблицами выполняются с помощью SQL-запросов, каждый из которых имеет свой синтаксис и особенности использования. В этой статье приведен базовый синтаксис и примеры запросов для создания, изменения и удаления таблиц, а также для просмотра информации о таблицах, чтения и записи данных.

Создание таблиц MergeTree

Обзор синтаксиса

Чтобы создать таблицу на сервере ADQM, необходимо использовать запрос `CREATE TABLE` с указанием движка таблицы с помощью выражения `ENGINE`. Полную информацию о запросе `CREATE TABLE` можно получить в документации ClickHouse.

Ниже приведен базовый синтаксис запроса для создания таблицы MergeTree — основного типа таблиц для работы с большими данными в ADQM.

```
CREATE TABLE [IF NOT EXISTS] [<db_name>.]<table_name> [ON CLUSTER <cluster_name>]
(   <column_name1> [<data_type1>] [NULL|NOT NULL] [DEFAULT|MATERIALIZED|EPHEMERAL|ALIAS <expr1>] [CODEC(<compression_codec1>)]
[TTL <expr1>],
    <column_name2> [<data_type2>] [NULL|NOT NULL] [DEFAULT|MATERIALIZED|EPHEMERAL|ALIAS <expr2>] [CODEC(<compression_codec2>)]
[TTL <expr2>],
    ...
    CONSTRAINT <constraint_name_1> CHECK <boolean_expr_1>,
    CONSTRAINT <constraint_name_2> CHECK <boolean_expr_2>,
    ...
    INDEX <index_name1> <expr1> TYPE <index_type1> GRANULARITY <granularity_value1>,
    INDEX <index_name2> <expr2> TYPE <index_type2> GRANULARITY <granularity_value2>)
ENGINE = MergeTree()
ORDER BY <expr>
[PARTITION BY <expr>]
[PRIMARY KEY <expr>]
[SAMPLE BY <expr>]
[TTL <expr>
    [DELETE|TO DISK 'xxx'|TO VOLUME 'xxx' [, ...] ]
    [WHERE <conditions>]
    [GROUP BY <key_expr> [SET v1 = aggr_func(v1) [, v2 = aggr_func(v2) ...]] ] ]
[SETTINGS name=value, ...];
```

Параметры запроса

Один из основных вариантов синтаксиса запроса `CREATE TABLE` — описание столбцов, ограничений и индексов в качестве параметров запроса.

Описания столбцов

Столбцы таблицы MergeTree описываются следующим образом:

```
<column_name> [<data_type>] [NULL|NOT NULL] [DEFAULT|MATERIALIZED|EPHEMERAL|ALIAS <expr>]
[CODEC(<compression_codec>)] [TTL <expr>]
```

Описание столбца содержит:

- имя столбца;
- тип данных (зависимость названия типа данных от регистра можно проверить в системной таблице system.data_type_families);
- модификатор `NULL` или `NOT NULL`, определяющий может ли тип данных столбца быть Nullable;
- выражение для значения по умолчанию с указанием его типа — `DEFAULT`, `MATERIALIZED`, `EPHEMERAL` или `ALIAS`;
- кодек сжатия столбца;
- выражение TTL, определяющее время хранения значений в столбце.

Ограничения (constraints)

Ограничения на значения в столбцах таблицы описываются следующим образом:

```
CONSTRAINT <constraint_name> CHECK <boolean_expr>
```

Описание ограничения содержит:

- имя ограничения;
- логическое выражение, которому должны удовлетворять данные для успешного выполнения запросов `INSERT`.

Индексы пропуска данных

Для таблиц семейства MergeTree можно задавать индексы пропуска данных, которые агрегируют данные в блоки по заданным выражениям и потом при выполнении запроса `SELECT` позволяют пропускать блоки, в которых гарантированно нет запрашиваемых данных. Таким образом можно уменьшить объем данных, читаемых с диска.

Индексы пропуска данных описываются в следующем формате:

```
INDEX <index_name> <expr> TYPE <index_type> GRANULARITY <granularity_value>
```

При описании индекса задается:

- имя индекса;
- выражение, по которому будут агрегироваться данные;
- тип индекса. Наиболее часто используемые индексы:
  - `minmax` — хранит минимум и максимум выражения;
  - `set(max_rows)` — хранит уникальные значения выражения в количестве, не большем чем `max_rows`;
  - `bloom_filter([false_positive])` — фильтр Блума для указанных столбцов. Необязательный параметр `false_positive` — это вероятность получения ложноположительного срабатывания. Возможные значения: `(0, 1)`. Значение по умолчанию: `0.025`.
- гранулярность — количество гранул данных в пропускаемом блоке.

Добавлять и удалять столбцы, ограничения и индексы можно и после создания таблицы с помощью запросов `ALTER TABLE` (см. раздел Модификация таблицы ниже).

Основные выражения

ON CLUSTER	По умолчанию таблица создается только на том сервере, на котором выполняется запрос. Чтобы создать таблицу на всех серверах кластера, нужно использовать выражение `ON CLUSTER` с указанием имени кластера
ENGINE	Движок таблицы с параметрами. Названия движков таблиц являются регистрозависимыми. Их можно посмотреть в системной таблице system.table_engines
ORDER BY	Ключ сортировки в виде одного или нескольких столбцов. ADQM использует ключ сортировки в качестве первичного ключа, если первичный ключ не задан с помощью выражения `PRIMARY KEY`
PARTITION BY	Ключ партиционирования. Ключом партиционирования может быть столбец таблицы или произвольное выражение по столбцу. Например, чтобы задать партиционирование по месяцам, можно использовать выражение `toYYYYMM(date_column)`, где `date_column` — столбец с датой типа Date. Ключом партиционирования также может быть кортеж из столбцов или выражений
PRIMARY KEY	Первичный ключ. По умолчанию первичный ключ совпадает с ключом сортировки `ORDER BY`, поэтому в большинстве случаев выражение `PRIMARY KEY` отдельно использовать не нужно. Если задается первичный ключ, отличный от ключа сортировки, то кортеж выражений первичного ключа должен быть префиксом кортежа выражений ключа сортировки
SAMPLE BY	Выражение для семплирования
TTL	Список правил TTL (time-to-live), определяющих сроки хранения строк в таблице, а также как данные должны быть обработаны по истечении указанного времени (например, устаревшие данные могут быть автоматически удалены, перемещены на другие диски/тома, сжаты другим кодеком или агрегированы/сгруппированы — подробнее в статье Использование TTL)
SETTINGS	Дополнительные параметры, регулирующие поведение таблицы MergeTree. Список настроек таблиц семейства MergeTree и их значений можно получить с помощью системной таблицы system.merge_tree_settings

В запросе `CREATE TABLE` (а также в других запросах, приведенных в этой статье ниже) `<table_name>` — имя таблицы, `<db_name>` — имя базы данных, в которой находится таблица. Если база данных не указана, запрос применяется к таблице в текущей базе данных сессии. Чтобы установить текущую базу данных, используйте следующий запрос:

```
USE <db_name>;
```

Примеры

Создание таблицы MergeTree

Следующий запрос создает таблицу **MergeTree** с тремя столбцами, которые будут хранить числовые идентификаторы, строковые значения и даты. В качестве ключа сортировки задается кортеж из столбца с идентификаторами и столбца с датами (первичный ключ `PRIMARY KEY` будет совпадать с ключом сортировки), а также устанавливается партиционирование по дням.

```
CREATE TABLE table_mergeTree
(
    id Int32,
    value String,
    date Date
)
ENGINE = MergeTree
ORDER BY (id, date)
PARTITION BY toYYYYMMDD(date);
```

Движок MergeTree не имеет параметров, его можно указывать как `ENGINE = MergeTree()` или `ENGINE = MergeTree`.

Создание таблицы ReplacingMergeTree

Таблица **ReplacingMergeTree** выполняет удаление дублирующихся записей с одинаковым значением ключа сортировки (`ORDER BY`, не `PRIMARY KEY`) во время слияния кусков данных.

Движок ReplacingMergeTree имеет один необязательный параметр `ver` — столбец (типа UInt*, Date или DateTime) с версией записи данных.

Из всех строк с одинаковым значением ключа сортировки ReplacingMergeTree оставляет только одну:

- **Последнюю по времени создания**, если параметр `ver` не задан.

```
CREATE TABLE table_replacing_1
(
    id Int32,
    value String,
    date Date,
    ver Int32
)
ENGINE = ReplacingMergeTree()
ORDER BY id;

INSERT INTO table_replacing_1 VALUES(1, 'value1', '2022-01-01', 1), (1, 'value2', '2022-01-01', 2), (1, 'value3', '2022-01-01',
-1);

SELECT * FROM table_replacing_1 FINAL;
```

В результате остается последняя добавленная строка:

```
--id---value---------date---ver--
│  1 │ value3 │ 2022-01-01 │  -1 │
---------------------------------
```

- **С максимальной версией**, если параметр `ver` задан.

```
CREATE TABLE table_replacing_2
(
    id Int32,
    value String,
    date Date,
    ver Int32
)
ENGINE = ReplacingMergeTree(ver)
ORDER BY id;

INSERT INTO table_replacing_2 VALUES(1, 'value1', '2022-01-01', 1), (1, 'value2', '2022-01-01', 2), (1, 'value3', '2022-01-01',
-1);

SELECT * FROM table_replacing_2 FINAL;
```

В результате остается строка с большим значением в столбце `ver`.

```
--id---value---------date---ver--
│  1 │ value2 │ 2022-01-01 │   2 │
---------------------------------
```

Дедупликация данных происходит на уровне партиций. Если таблица партиционирована, то дубликаты между партициями не удаляются.

Так как дедупликация производится во время слияния данных, которое происходит в фоновом режиме, некоторая часть данных может остаться необработанной в момент, когда выполняется запрос `SELECT`. Чтобы вызвать внеочередное слияние и получить таблицу или партицию без дубликатов, можно использовать модификатор `FINAL` после имени таблицы в запросе `SELECT` или предварительно выполнить запрос `OPTIMIZE` для таблицы или партиции.

Создание таблицы SummingMergeTree

Движок **SummingMergeTree** отслеживает строки с одинаковыми ключами сортировки и заменяет их на одну, суммируя значения в числовых столбцах. Значения в столбцах других типов выбираются произвольно из какой-либо строки.

Движок SummingMergeTree имеет один необязательный параметр `columns` — имена столбцов, в которых будут суммироваться данные (столбцы должны иметь числовой тип и не должны входить в ключ сортировки).

- Если параметр `columns` не задан, то ADQM суммирует значения во всех столбцах с числовым типом данных, не входящих в ключ сортировки.

```
CREATE TABLE table_sum_1
(
    id UInt32,
    value1 Int32,
    value2 Int32,
    value3 Int32
)
ENGINE = SummingMergeTree()
ORDER BY id;

INSERT INTO table_sum_1 VALUES(1, 1, 1, 1), (1, 2, 2, 2), (1, 3, 3, 3);

SELECT * FROM table_sum_1;
```

В результате значения во всех числовых столбцах просуммировались:

```
--id---value1---value2---value3--
│  1 │      6 │      6 │      6 │
---------------------------------
```

- Если параметр `columns` задан, ADQM суммирует значения в заданных столбцах.

```
CREATE TABLE table_sum_2
(
    id UInt32,
    value1 Int32,
    value2 Int32,
    value3 Int32
)
ENGINE = SummingMergeTree(value1, value2)
ORDER BY id;

INSERT INTO table_sum_2 VALUES(1, 1, 1, 1), (1, 2, 2, 2), (1, 3, 3, 3);

SELECT * FROM table_sum_2;
```

В результате значения в столбцах `value1` и `value2` просуммировались, а значение столбца `value3` взялось из произвольной строки:

```
--id---value1---value2---value3--
│  1 │      6 │      6 │      1 │
---------------------------------
```

Создание таблицы AggregatingMergeTree

В таблицах **AggregatingMergeTree** все строки с одинаковым ключом сортировки заменяются на одну (в пределах одного куска данных), которая хранит объединение состояний агрегатных функций. Такие таблицы могут использоваться для инкрементальной агрегации данных, в том числе для агрегирующих материализованных представлений.

Движок AggregatingMergeTree не имеет параметров. Ниже приведен пример создания агрегирующего материализованного представления.

1. Создайте таблицу `table_sales` типа MergeTree для хранения данных о продажах:

```
CREATE TABLE table_sales
(
    user_id Int64,
    store_id Int32,
    product_id Int32,
    date Date,
    sum Decimal(20, 2)
)
ENGINE = MergeTree
PARTITION BY date
ORDER BY (store_id, product_id, user_id);
```

2. Используя запрос CREATE MATERIALIZED VIEW, создайте материализованное представление типа AggregatingMergeTree, которое "смотрит" на таблицу `table_sales` и хранит промежуточные состояния (state) для агрегатных функций:

```
CREATE MATERIALIZED VIEW table_aggregating
(
    store_id Int32,
    product_id Int32,
    date Date,
    s AggregateFunction(sum, Decimal(20, 2)),
    c AggregateFunction(count, Decimal(20, 2))
)
ENGINE = AggregatingMergeTree
PARTITION BY date
ORDER BY (store_id, date) AS
SELECT
    store_id,
    product_id,
    date,
    sumState(sum) AS s,
    countState(sum) AS c
FROM table_sales
GROUP BY
    store_id,
    date,
    product_id;
```

При вставке данных в таблицу `table_sales` в материализованном представлении `table_aggregating` будет происходит автоматическое обновление агрегатных состояний для всех измененных ключей, указанных в `GROUP BY`.

3. Чтобы получить агрегированные данные, выполните запрос `SELECT … GROUP BY …` из представления `table_aggregating`:

```
SELECT
    store_id,
    product_id,
    date,
    sumMerge(s),
    countMerge(c)
FROM table_aggregating
GROUP BY
    store_id,
    date,
    product_id
LIMIT 1;
```

Создание таблицы CollapsingMergeTree

Движок таблиц **CollapsingMergeTree** позволяет быстро записывать постоянно изменяющиеся состояния объектов и удаляет старые состояния объектов в фоновом режиме, что значительно сокращает объем хранения.

CollapsingMergeTree асинхронно удаляет (сворачивает) пары строк, которые содержат одинаковые значения во всех столбцах в ключе сортировки (`ORDER BY`), но разные значения в специальном столбце `sign`. Этот столбец имеет тип данных Int8 и может содержать значения `1` и `-1`, по которым определяется тип строки:

- `1` — обозначает, что строка является строкой состояния;
- `-1` — обозначает, что строка является строкой отмены состояния.

Столбец `sign` указывается в качестве параметра движка CollapsingMergeTree.

Подробно алгоритм сворачивания строк описан в статье CollapsingMergeTree документации ClickHouse.

Ниже приведен пример преобразования данных таблицей CollapsingMergeTree.

1. Создайте таблицу типа CollapsingMergeTree и укажите столбец, по значениям которого будет определяться тип строки:

```
CREATE TABLE table_collapsing
(
    id UInt32,
    pageViews UInt8,
    duration UInt8,
    sign Int8
)
ENGINE = CollapsingMergeTree(sign)
ORDER BY id;
```

2. Выполните два запроса `INSERT`, чтобы создать два отдельных куска данных. Во втором запросе первая запись отменяет предыдущее состояние объекта, а вторая содержит текущее состояние.

```
INSERT INTO table_collapsing VALUES (1, 7, 100, 1);
INSERT INTO table_collapsing VALUES (1, 7, 100, -1), (1, 8, 150, 1);
```

3. Используйте запрос `SELECT`, чтобы проверить, что в таблице удалились строки, соответствующие устаревшему состоянию объекта, и осталось только последнее состояние объекта. Так как ADQM объединяет вставленные куски данных в неизвестный момент времени, то для получения полностью "свернутых" данных из таблицы CollapsingMergeTree, требуется агрегирование. Если необходимо выбирать данные без агрегации (например, проверить наличие строк, последние значения которых удовлетворяют некоторым условиям), можно использовать модификатор `FINAL` для выражения `FROM`.

```
SELECT * FROM table_collapsing FINAL;
```

Результат выполнения запроса:

```
--id---pageViews---duration---sign--
│  1 │         8 │      150 │    1 │
------------------------------------
```

Создание таблицы VersionedCollapsingMergeTree

Движок **VersionedCollapsingMergeTree** имеет то же назначение, что и CollapsingMergeTree, но использует другой алгоритм удаления (сворачивания) строк, который позволяет вставлять данные в любом порядке в несколько потоков (CollapsingMergeTree требует строго последовательную вставку данных).

Параметры движка VersionedCollapsingMergeTree:

- `sign` — имя столбца (типа Int8), значения которого обозначают тип строки (`1` — строка состояния объекта; `-1` — строка отмены состояния, которая копирует все поля отменяемого состояния за исключением `sign`);
- `version` — имя столбца (типа UInt), который содержит версию состояния объекта (идентифицирует каждое состояние объекта отдельным номером).

Ниже приведен пример преобразования данных таблицей VersionedCollapsingMergeTree.

1. Создайте таблицу типа VersionedCollapsingMergeTree и определите столбцы, по которым будет определяться тип строки и версия состояния объекта:

```
CREATE TABLE table_versioned_collapsing
(
    id UInt32,
    pageViews UInt8,
    duration UInt8,
    sign Int8,
    version UInt8
)
ENGINE = VersionedCollapsingMergeTree(sign, version)
ORDER BY id;
```

2. Используйте два запроса `INSERT` для создания двух отдельных кусков данных. Во втором запросе первая запись отменяет предыдущее состояние объекта, а вторая содержит текущее состояние.

```
INSERT INTO table_versioned_collapsing VALUES (1, 7, 100, 1, 1);
INSERT INTO table_versioned_collapsing VALUES (1, 7, 100, -1, 1), (1, 8, 150, 1, 2);
```

3. Используйте запрос `SELECT`, чтобы проверить, что таблица удалила строки, соответствующие устаревшему состоянию объекта, и оставила только последнее состояние объекта. Так как ADQM объединяет вставленные куски данных в неизвестный момент времени, то для получения полностью "свернутых" данных из таблицы VersionedCollapsingMergeTree, требуется агрегирование. Если необходимо выбирать данные без агрегации (например, проверить наличие строк, последние значения которых удовлетворяют некоторым условиям), можно использовать модификатор `FINAL` для выражения `FROM`.

```
SELECT * FROM table_versioned_collapsing FINAL;
```

Результат выполнения запроса:

```
--id---pageViews---duration---sign---version--
│  1 │         8 │      150 │    1 │       2 │
----------------------------------------------
```

Создание реплицируемых таблиц

Для создания реплицируемых таблиц используются движки семейства MergeTree с префиксом Replicated.

Метаинформация, относящаяся к процессу репликации, хранится в ZooKeeper. Соответственно, движки **Replicated*** имеют два обязательных параметра для формирования отдельного каталога в ZooKeeper для каждой реплицируемой таблицы:

- путь к таблице в ZooKeeper;
- имя реплики в ZooKeeper (идентифицирует разные реплики одной и той же таблицы).

Также каждый движок с префиксом `Replicated` имеет параметры движка, которому соответствует данная реплицируемая версия. Например, движок ReplicatedSummingMergeTree, как и движок SummingMergeTree, имеет необязательный параметр `columns` — список столбцов, в которых будут суммироваться данные.

Пример создания реплицируемой таблицы ReplicatedMergeTree можно посмотреть в статье Типовой кластер.

Просмотр списка таблиц

Получить имена всех таблиц, существующих в базе данных, можно следующими способами:

- Использовать запрос SHOW TABLES.

```
SHOW TABLES FROM <database_name>;
```

- Сделать выборку из системной таблицы system.tables.

```
SELECT name FROM system.tables WHERE database='<database_name>';
```

Получение информации о таблице

Структура таблицы

Описание столбцов таблицы можно получить с помощью запроса DESCRIBE TABLE:

```
DESCRIBE TABLE [<db_name>.]<table_name>;
```

Этот запрос возвращает список столбцов таблицы с их именами и типами данных, а также следующую информацию по каждому столбцу:

- тип выражения для значения по умолчанию (`DEFAULT`, `MATERIALIZED`, `EPHEMERAL` или `ALIAS`);
- выражение для значения по умолчанию;
- комментарий;
- кодек, который применяется к столбцу;
- выражение TTL.

Пример

Запрос:

```
CREATE TABLE test_table
(
    id Int32,
    description String DEFAULT 'unknown' COMMENT 'This column contains descriptions' CODEC(LZ4),
    date Date COMMENT 'This column contains event dates'
)
ENGINE = MergeTree
ORDER BY id;

DESCRIBE TABLE test_table;
```

Результат:

```
--name---------type-----default_type---default_expression---comment----------------------------codec_expression---
ttl_expression--
│ id          │ Int32  │              │                    │                                  │                 │
│
│ description │ String │ DEFAULT      │ 'unknown'          │ This column contains descriptions │ LZ4             │
│
│ date        │ Date   │              │                    │ This column contains event dates  │                 │
│
-------------------------------------------------------------------------------------------------------------------
------
```

Метаданные таблицы

Метаданные таблицы можно получить из системной таблицы system.tables. Синтаксис запроса:

```
SELECT * FROM system.tables WHERE name = '<table_name>' [FORMAT <format_name>];
```

Пример

Запрос:

```
SELECT * FROM system.tables WHERE name = 'test_table' FORMAT Vertical;
```

Результат:

```
Row 1:
──────
database:                      default
name:                          test_table
uuid:                          7239d59f-9144-4600-b47f-c9dd02330242
engine:                        MergeTree
is_temporary:                  0
data_paths:                    ['/var/lib/clickhouse/store/723/7239d59f-9144-4600-b47f-c9dd02330242/']
metadata_path:                 /var/lib/clickhouse/store/463/4632cdc5-46bc-4d4e-ad36-27b8d101e2e1/test_table.sql
metadata_modification_time:    2022-12-21 07:04:25
dependencies_database:         []
dependencies_table:            []
create_table_query:            CREATE TABLE default.test_table (`id` Int32, `v` String, `d` Date, `v2` String CODEC(ZSTD(10)))
ENGINE = MergeTree PARTITION BY d ORDER BY id SETTINGS index_granularity = 8192 COMMENT 'This is a test table'
engine_full:                   MergeTree PARTITION BY d ORDER BY id SETTINGS index_granularity = 8192
as_select:
partition_key:                 d
sorting_key:                   id
primary_key:                   id
sampling_key:
storage_policy:                default
total_rows:                    1000000
total_bytes:                   56688997
lifetime_rows:                 NULL
lifetime_bytes:                NULL
comment:                       This is a test table
has_own_data:                  1
loading_dependencies_database: []
loading_dependencies_table:    []
loading_dependent_database:    []
loading_dependent_table:       []
```

Добавление данных в таблицу

Для добавления данных в таблицу используется запрос INSERT INTO, в котором после выражения `VALUES` через запятую перечисляются значения столбцов (или кортежи данных, если добавляется более одной строки):

```
INSERT INTO [<db_name>.]<table_name> [(<column_name1>, <column_name2>, <column_name3>)] VALUES (v11, v12, v13), (v21, v22, v23),
...;
```

Пример

1. Создайте таблицу с тремя столбцами:

```
CREATE TABLE test_table
(
    a Int32,
    b String,
    c Int32
)
ENGINE = MergeTree
ORDER BY a;
```

2. Вставьте две строки с указанными значениями для всех столбцов таблицы:

```
INSERT INTO test_table VALUES (1, 'a', 1), (2, 'b', 2);
```

Результат:

```
--a---b---c--
│ 1 │ a │ 1 │
│ 2 │ b │ 2 │
-------------
```

3. Вставьте две строки, указав значения только для столбцов `a` и `b`:

```
INSERT INTO test_table (a, b) VALUES (3, 'c'), (4, 'd');
```

В результате данные из запроса `INSERT INTO` записываются в столбцы `a` и `b`, а столбец `c` заполняется нулями, так как для него не задано значение по умолчанию:

```
--a---b---c--
│ 3 │ c │ 0 │
│ 4 │ d │ 0 │
-------------
```

Запрос `INSERT INTO` позволяет также копировать данные из одной таблицы в другую. Для этого используется комбинация выражений `INSERT INTO` и `SELECT`:

```
INSERT INTO [<db_name>.]<table_name> [(<column_name1>, <column_name2>, <column_name3>)] SELECT ... ;
```

Пример

Приведенный ниже пример копирует в таблицу `new_table` первые три строки из таблицы с той же структурой `source_table`, которая содержит следующие данные:

```
--a---b----c--
│ 1 │ a │  1 │
│ 1 │ a │  1 │
│ 2 │ b │  2 │
│ 3 │ c │  0 │
│ 6 │ e │ 12 │
│ 9 │ a │  9 │
--------------
```

1. Создайте таблицу `new_table` с тремя столбцами:

```
CREATE TABLE new_table
(
    d Int32,
    f String,
    g Int32
)
ENGINE = MergeTree
ORDER BY d;
```

2. Вставьте в новую таблицу три строки из таблицы `source_table` с помощью следующего запроса:

```
INSERT INTO new_table SELECT * FROM source_table LIMIT 3;
```

Результат выполнения запроса:

```
--d---f---g--
│ 1 │ a │ 1 │
│ 1 │ a │ 1 │
│ 2 │ b │ 2 │
-------------
```

Чтение данных из таблицы

Для чтения данных из таблицы используется запрос `SELECT`. В запросах этого типа можно использовать различные модификаторы и выражения для дополнительной обработки считываемых данных. Например, с помощью специальных выражений данные, которые будет возвращать запрос, можно отфильтровать/отсортировать/сгруппировать, ограничить количество строк в выборке, задать формат вывода данных. Подробнее все выражения, доступные для использования в запросах `SELECT`, описаны в разделе SELECT Query документации ClickHouse.

Чтение всех данных

Чтобы считать полностью все данные из таблицы, нужно использовать символ `*` после выражения `SELECT`:

```
SELECT * FROM [<db_name>.]<table_name>;
```

Например, следующий запрос считывает данные из всех столбцов таблицы `test_table`:

```
SELECT * FROM test_table;
```

Результат:

```
--a---b----c--
│ 1 │ a │  1 │
│ 1 │ a │  1 │
│ 2 │ b │  2 │
│ 3 │ c │  0 │
│ 6 │ e │ 12 │
│ 9 │ a │  9 │
--------------
```

Чтение данных из определенных столбцов

Чтобы считать данные из определенных столбцов, нужно перечислить имена этих столбцов после выражения `SELECT`:

```
SELECT <column_name1>[, ...] FROM [<db_name>.]<table_name>;
```

Например, следующий запрос считывает данные из столбцов `b` и `c` таблицы `test_table`:

```
SELECT b, c FROM test_table;
```

Результат:

```
--b----c--
│ a │  1 │
│ a │  1 │
│ b │  2 │
│ c │  0 │
│ e │ 12 │
│ a │  9 │
----------
```

Выборка определенного количества строк

Чтобы выбрать из таблицы определенное количество строк, нужно использовать выражение `LIMIT` в запросе `SELECT`:

```
SELECT * FROM [<db_name>.]<table_name> LIMIT [n, ]m;
```

В результате выполнения этого запроса из таблицы выбираются первые `m` строк после пропуска первых `n` строк.

Например, следующий запрос считывает первые три строки из таблицы `test_table`:

```
SELECT * FROM test_table LIMIT 3;
```

Результат:

```
--a---b----c--
│ 1 │ a │  1 │
│ 1 │ a │  1 │
│ 2 │ b │  2 │
--------------
```

Чтение данных по заданному условию

Задать условие для фильтрации считываемых данных можно с помощью выражения `WHERE`:

```
SELECT <column_name1>[, ...] FROM [<db_name>.]<table_name> WHERE <filter_expr>;
```

Например, следующий запрос выводит значения столбца `c`, если они больше `1`, но меньше `10`:

```
SELECT c FROM test_table WHERE (c > 1) AND (c < 10);
```

Результат:

```
---c--
│  2 │
│  9 │
------
```

Сортировка результатов

Отсортировать результаты запроса `SELECT` можно с помощью выражения `ORDER BY`:

```
SELECT <column_name1>[, ...] FROM [<db_name>.]<table_name> ORDER BY <expr_list>;
```

Например, следующий запрос сортирует данные по значениям в столбце `c` в порядке возрастания:

```
SELECT * FROM test_table ORDER BY c;
```

Результат:

```
--a---b----c--
│ 3 │ c │  0 │
│ 1 │ a │  1 │
│ 1 │ a │  1 │
│ 2 │ b │  2 │
│ 9 │ a │  9 │
│ 6 │ e │ 12 │
--------------
```

Модификация таблицы

Для таблиц семейства MergeTree, а также таблиц типа Merge и Distributed, поддерживается запрос `ALTER TABLE`, который позволяет изменять структуру таблицы, ее настройки или данные. Например, ниже приведен синтаксис запроса для изменения столбцов таблицы:

```
ALTER TABLE [<db_name>].<table_name> [ON CLUSTER <cluster_name>] ADD|DROP|CLEAR|COMMENT|MODIFY COLUMN ...;
```

Манипуляции со столбцами

Следующие выражения в запросах `ALTER TABLE` позволяют выполнять различные действия над столбцами таблицы.

Выражение	Описание
ADD COLUMN	Добавляет столбец в таблицу. Выражения `AFTER | FIRST` используются, чтобы указать куда вставлять новый столбец — после указанного столбца `after_column_name` или в начало таблицы. Если ни одно из этих выражений не используется, столбец добавляется в конец таблицы
DROP COLUMN	Удаляет столбец с заданным именем
RENAME COLUMN	Переименовывает существующий столбец. Столбцы, которые являются частью первичного ключа (`PRIMARY KEY`) или ключа сортировки (`ORDER BY`), не могут быть переименованы
CLEAR COLUMN	Сбрасывает все значения в столбце для заданной партиции
COMMENT COLUMN	Добавляет комментарий к столбцу. Если комментарий уже существует, то он заменяется на новый, так как столбец может содержать только один комментарий
MODIFY COLUMN	Изменяет настройки столбца (тип данных, выражение для значения по умолчанию, кодек сжатия данных, TTL, позицию столбца в таблице)

Синтаксис

ADD COLUMN

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    ADD COLUMN [IF NOT EXISTS] <new_column_name>
[<data_type>] [<default_expr>]
[CODEC(<codec_name>)] [TTL <expr>] [AFTER
<after_column_name> | FIRST];
```

DROP COLUMN

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    DROP COLUMN [IF EXISTS] <column_name>;
```

RENAME COLUMN

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    RENAME COLUMN [IF EXISTS] <old_name> to
<new_name>;
```

CLEAR COLUMN

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    CLEAR COLUMN [IF EXISTS] <column_name> IN
PARTITION <partition_name>;
```

COMMENT COLUMN

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    COMMENT COLUMN [IF EXISTS] <column_name> 'Text
comment';
```

MODIFY COLUMN

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    MODIFY COLUMN [IF EXISTS] <column_name>
[<data_type>] [<default_expr>]
[CODEC(<codec_name>)] [TTL <expr>] [AFTER
<after_column_name> | FIRST];
```

Манипуляции с ключевыми выражениями таблиц

Для таблиц семейства MergeTree (в том числе реплицируемых таблиц) поддерживается запрос на изменение ключа сортировки таблицы:

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER <cluster_name>] MODIFY ORDER BY <new_expression>;
```

Манипуляции с индексами

С помощью следующих запросов можно добавить или удалить индекс пропуска данных:

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER <cluster_name>] ADD INDEX <index_name> <expression> TYPE <index_type>
GRANULARITY  <granularity_value> [FIRST|AFTER name];
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER <cluster_name>] DROP INDEX <index_name>;
```

Эти операции поддерживаются только таблицами семейства MergeTree.

Манипуляции с ограничениями (constraints)

Добавить или удалить ограничение можно с помощью запросов:

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER <cluster_name>] ADD CONSTRAINT <constraint_name> CHECK <expression>;
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER <cluster_name>] DROP CONSTRAINT <constraint_name>;
```

Манипуляции с партициями и кусками данных

Для работы с партициями и кусками данных можно использовать следующие выражения в запросах `ALTER TABLE`.

Выражение	Описание
DETACH PARTITION|PART	Переносит партицию/кусок в директорию *detached* (сервер не будет "видеть" партицию/кусок, пока не будет выполнен запрос `ATTACH PARTITION|PART`). Запрос реплицируется — данные перемещаются в директорию *detached* на всех репликах
DROP PARTITION|PART	Удаляет партицию/кусок. Запрос реплицируется — данные удаляются на всех репликах
DROP DETACHED PARTITION|PART	Удаляет партицию/кусок из директории *detached*
ATTACH PARTITION|PART	Добавляет партицию/кусок в таблицу из директории *detached*. Запрос реплицируется
ATTACH PARTITION FROM	Копирует партицию из одной таблицы в другую. Таблицы должны иметь одинаковые: - структуры; - ключи партиционирования, ключи сортировки и первичные ключи; - политики хранения данных (диск, на котором хранится партиция, должен быть доступен для обеих таблиц).
REPLACE PARTITION FROM	Копирует партицию из одной таблицы в другую с заменой существующей партиции в таблице назначения (данные из таблицы-источника не удаляются). Таблицы должны иметь одинаковые: - структуры; - ключи партиционирования, ключи сортировки и первичные ключи; - политики хранения данных (диск, на котором хранится партиция, должен быть доступен для обеих таблиц).
MOVE PARTITION TO TABLE	Перемещает партицию из одной таблицы в другую с удалением данных из таблицы-источника. Таблицы должны иметь одинаковые: - структуры; - ключи партиционирования, ключи сортировки и первичные ключи; - политики хранения данных (диск, на котором хранится партиция, должен быть доступен для обеих таблиц); - семейства движков (реплицируемые или нереплицируемые).
CLEAR COLUMN IN PARTITION	Удаляет все значения в столбце для заданной партиции. Если для столбца задано значение по умолчанию, то будет выставлено это значение
CLEAR INDEX IN PARTITION	Очищает построенные вторичные индексы для заданной партиции
FREEZE PARTITION	Создает резервную копию партиции (копирует только данные, но не метаданные таблицы). Если выражение `PARTITION` не используется, резервные копии будут созданы для всех партиций. Запрос не реплицируется — резервная копия создается только на локальном сервере. Чтобы сделать резервную копию метаданных таблицы, необходимо скопировать файл */var/lib/clickhouse/metadata/database/table.sql*
UNFREEZE PARTITION	Удаляет с диска резервную копию партиции. Если секция `PARTITION` не используется, запрос удаляет резервную копию всех партиций сразу
FETCH PARTITION|PART	Скачивает партицию/кусок с другого сервера. Запрос работает только для реплицируемых таблиц
MOVE PARTITION|PART	Перемещает партицию/кусок на другой диск или том. Запрос не реплицируется, так как на разных репликах могут быть различные конфигурации политик хранения
UPDATE IN PARTITION	Обновляет соответствующие указанному условию данные в партиции
DELETE IN PARTITION	Удаляет соответствующие указанному условию данные в партиции

Синтаксис

DETACH PARTITION|PART

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    DETACH PARTITION|PART <partition_expr>;
```

DROP PARTITION|PART

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    DROP PARTITION|PART <partition_expr>;
```

DROP DETACHED PARTITION|PART

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    DROP DETACHED PARTITION|PART <partition_expr>;
```

ATTACH PARTITION|PART

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    ATTACH PARTITION|PART <partition_expr>;
```

ATTACH PARTITION FROM

```
ALTER TABLE <dest_table_name> [ON CLUSTER
<cluster_name>]
    ATTACH PARTITION <partition_expr> FROM
<source_table_name>;
```

REPLACE PARTITION FROM

```
ALTER TABLE <dest_table_name> [ON CLUSTER
<cluster_name>]
    REPLACE PARTITION <partition_expr> FROM
<source_table_name>;
```

MOVE PARTITION TO TABLE

```
ALTER TABLE <source_table_name> [ON CLUSTER
<cluster_name>]
    MOVE PARTITION <partition_expr> TO TABLE
<dest_table_name>;
```

CLEAR COLUMN IN PARTITION

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    CLEAR COLUMN <column_name> IN PARTITION
<partition_expr>;
```

CLEAR INDEX IN PARTITION

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    CLEAR INDEX <index_name> IN PARTITION
<partition_expr>;
```

FREEZE PARTITION

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    FREEZE [PARTITION <partition_expr>] [WITH NAME
'backup_name'];
```

UNFREEZE PARTITION

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    UNFREEZE [PARTITION <part_expr>] WITH NAME
'backup_name';
```

FETCH PARTITION|PART

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    FETCH PARTITION|PART <partition_expr> FROM
'path-in-zookeeper';
```

MOVE PARTITION|PART

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    MOVE PARTITION|PART <partition_expr> TO
DISK|VOLUME 'disk_name';
```

UPDATE IN PARTITION

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    UPDATE <column_name1> = <expr1> [, ...] [IN
PARTITION <partition_expr>] WHERE <filter_expr>;
```

DELETE IN PARTITION

```
ALTER TABLE [<db_name>.]<table_name> [ON CLUSTER
<cluster_name>]
    DELETE [IN PARTITION <partition_expr>] WHERE
<filter_expr>;
```

Как задать партицию в запросе

Чтобы задать партицию в запросе `ALTER TABLE … PARTITION`, можно использовать:

- Имя партиции в одинарных кавычках (кавычки можно не использовать, если выражение, по которому задано партиционирование, типа Int*). Например: `ALTER TABLE test_table DROP PARTITION '2022-12-26';`.
- Кортеж констант или выражений, совпадающий по типам с кортежем ключа партиционирования таблицы. Если ключ партиционирования таблицы состоит из одного элемента, выражение нужно обернуть в функцию `tuple(…)`. Например, `ALTER TABLE test_table DROP PARTITION tuple(toYYYYMMDD(toDate('2022-12-26')));`.
- Строковый идентификатор партиции в одинарных кавычках, указанный для выражения `PARTITION ID`. Например, `ALTER TABLE test_table DROP PARTITION ID '20221226';`.

Посмотреть имя и идентификатор партиции можно в столбцах `partition` и `partition_id` системной таблицы system.parts:

```
SELECT partition, partition_id FROM system.parts WHERE table = '<table_name>' and active = 1;
```

Чтобы задать имя куска партиции в запросах `ATTACH PART` и `DROP DETACHED PART`, используется значение из столбца `name` системной таблицы system.detached_parts (в одинарных кавычках).

Обновление и удаление существующих данных таблицы описано в статье Обновление и удаление строк.

Отключение и подключение таблицы

Таблицу можно отключить с помощью запроса `DETACH TABLE`:

```
DETACH TABLE [IF EXISTS] [<db_name.]<table_name> [ON CLUSTER <cluster_name>] [PERMANENTLY];
```

При выполнении этого запроса таблица становится "невидимой" для сервера, но ее данные и метаданные не удаляются. Если отключение не было перманентным (ключевое слово `PERMANENTLY` не использовалось в запросе), при следующем запуске сервер прочитает метаданные и снова будет "видеть" таблицу. Если таблица была отключена перманентно, то сервер не подключит ее обратно автоматически.

Независимо от того, каким способом таблица была отключена, ее можно подключить обратно с помощью запроса `ATTACH`:

```
ATTACH TABLE [IF NOT EXISTS] [<db_name.]<table_name> [ON CLUSTER <cluster_name>];
```

Удаление таблицы

Удалить таблицу можно с помощью запроса `DROP TABLE`:

```
DROP TABLE [IF EXISTS] [<db_name.]<table_name> [ON CLUSTER <cluster_name>];
```

Обновление и удаление строк

Елена Дворядкина

Содержание

Назад к содержанию

- Обновление данных
- Удаление данных
  - ALTER TABLE…DELETE
  - DELETE FROM

Концептуально ADQM не предназначен для обновления и удаления данных, однако поддерживает возможность выполнения **мутаций** — операций изменения/удаления данных `ALTER TABLE … UPDATE/DELETE`. Область применения мутаций — изменения, затрагивающие большое количество строк в таблице. Эти операции являются тяжеловесными и не подходят для частого использования (по возможности их лучше избегать).

После того как запрос на изменение/удаление данных отправлен на выполнение, его нельзя откатить и выполнение мутации будет продолжаться даже после перезапуска сервера ClickHouse. Чтобы отслеживать процесс выполнения мутации, используйте системную таблицу system.mutations. Если по какой-либо причине мутация "зависла" или выполняется некорректно, ее можно прервать с помощью запроса KILL MUTATION (дальнейшая обработка будет остановлена, но все уже примененные к данным изменения останутся).

Обновление данных

Обновить данные в таблице можно с помощью запроса `ALTER TABLE…UPDATE`:

```
ALTER TABLE [<db_name>.]<table_name>
[ON CLUSTER <cluster_name>]
UPDATE <column_name> = <expression>[, ...]
[IN PARTITION <partition_expr>]
WHERE <filter_expression>;
```

В этом запросе:

- `[<db_name>.]<table_name>` — имя таблицы, в которой будут обновляться данные. Если имя базы данных не указано, таблица ищется в текущей базе данных.
- `[ON CLUSTER <cluster_name>]` — имя кластера, на всех серверах которого будет выполняться запрос. Если выражение `ON CLUSTER` не используется, запрос на обновление данных в таблице выполняется только на одном сервере.
- `<column_name> = <expression>` — команда обновления значений в столбце. Значения столбца `<column_name>` заменяются на вычисленное значение выражения `<expression>` в каждой строке, для которой выражение `<filter_expression>` возвращает ненулевое значение.

  Значение выражения `<expression>` должно быть того же типа данных, что и столбец `<column_name>`, или должно быть приведено к тому же типу с помощью оператора `CAST`.

  Один запрос может содержать несколько команд `<column_name> = <expression>`, разделенных запятыми. Изменение значений нескольких столбцов одним запросом может быть более эффективным, чем выполнение нескольких запросов, каждый из которых обновляет один столбец.

  Изменение столбцов, которые используются при вычислении первичного ключа или ключа сортировки, не поддерживается.
- `[IN PARTITION <partition_expr>]` — партиция, в которой будут обновляться данные. Использование выражения `IN PARTITION` помогает ускорить выполнение операции `UPDATE`, когда нужно точечно обновить данные в таблице, которая разбита на множество партиций.
- `<filter_expression>` — выражение, возвращающее значение типа UInt8 для каждой строки таблицы, по которому будет определяться, в каких строках нужно обновить данные.

Пример

Исходная таблица:

```
--a---b----c---------date--
| 1 | a | 10 | 2022-12-10 |
| 2 | b | 20 | 2022-12-11 |
| 3 | c | 30 | 2022-12-12 |
| 4 | d | 40 | 2022-12-13 |
| 5 | e | 50 | 2022-12-14 |
| 6 | f | 60 | 2022-12-15 |
---------------------------
```

Следующий запрос заменяет значения столбцов `b` и `c` в строках, в которых значение столбца `date` отвечает заданному условию:

```
ALTER TABLE test_table UPDATE b = repeat(b, 3), c = c/10 WHERE date > '2022-12-13';
```

Результат выполнения запроса обновления данных:

```
--a---b------c---------date--
| 1 | a   | 10 | 2022-12-10 |
| 2 | b   | 20 | 2022-12-11 |
| 3 | c   | 30 | 2022-12-12 |
| 4 | d   | 40 | 2022-12-13 |
| 5 | eee |  5 | 2022-12-14 |
| 6 | fff |  6 | 2022-12-15 |
-----------------------------
```

Для реплицируемых таблиц запросы `ALTER TABLE…UPDATE` выполняются асинхронно. При необходимости синхронизировать обновление данных на репликах можно с помощью настройки mutations_sync.

Удаление данных

Для удаления строк данных из таблицы по заданному условию могут использоваться следующие запросы:

- `ALTER TABLE…DELETE` — поддерживается всеми движками таблиц.
- `DELETE FROM` — поддерживается только движками таблиц семейства MergeTree (начиная с версии ADQM 22.8).

В большинстве случаев запросы `DELETE FROM` выполняются значительно быстрее, чем `ALTER TABLE…DELETE` (эта операция требует слияния данных перед их удалением). `DELETE FROM` считается легковесной операцией, которая эффективна для удаления небольшого количества строк. Для некоторых сценариев (например, для нечастого выполнения массового удаления данных) лучше подходит `ALTER TABLE…DELETE`.

ALTER TABLE…DELETE

Синтаксис запроса:

```
ALTER TABLE [<db_name>.]<table_name>
[ON CLUSTER <cluster_name>]
DELETE [IN PARTITION <partition_expr>]
WHERE <filter_expression>;
```

Запрос удаляет из таблицы `<table_name>` строки, для которых выражение `<filter_expression>` возвращает ненулевое значение (типа UInt8). Чтобы удалить строки только из определенной партиции, необходимо указать нужную партицию с помощью выражения `IN PARTITION`.

Пример

Исходная таблица:

```
--a---b------c---------date--
| 1 | a   | 10 | 2022-12-10 |
| 2 | b   | 20 | 2022-12-11 |
| 3 | c   | 30 | 2022-12-12 |
| 4 | d   | 40 | 2022-12-13 |
| 5 | eee |  5 | 2022-12-14 |
| 6 | fff |  6 | 2022-12-15 |
-----------------------------
```

Следующий запрос удаляет из таблицы `test_table` строки, в которых значения столбцов `date` и `a` удовлетворяют заданным условиям:

```
ALTER TABLE test_table DELETE WHERE date < '2022-12-13' AND a = 2;
```

Результат выполнения запроса:

```
--a---b------c---------date--
| 1 | a   | 10 | 2022-12-10 |
| 3 | c   | 30 | 2022-12-12 |
| 4 | d   | 40 | 2022-12-13 |
| 5 | eee |  5 | 2022-12-14 |
| 6 | fff |  6 | 2022-12-15 |
-----------------------------
```

Для реплицируемых таблиц запросы `ALTER TABLE…DELETE` выполняются асинхронно. Если необходимо синхронизировать операцию удаления на репликах, измените значение настройки mutations_sync.

DELETE FROM

Синтаксис запроса:

```
DELETE FROM [<db_name>.]<table_name> WHERE <filter_expression>;
```

Запрос удаляет из таблицы `<table_name>` строки, для которых выражение `<filter_expression>` возвращает ненулевое значение (типа UInt8). Удаляемые строки сразу помечаются как удаленные и автоматически исключаются из всех последующих запросов. Удаление данных происходит асинхронно в фоновом режиме (синхронность выполнения запросов определяется параметром mutations_sync).

Пример

Исходная таблица:

```
--a---b-------c---------date--
| 1 | a    | 10 | 2022-12-10 |
| 3 | c    | 30 | 2022-12-12 |
| 4 | d    | 40 | 2022-12-13 |
| 5 | eee  |  5 | 2022-12-14 |
| 6 | fff  |  6 | 2022-12-15 |
| 7 | aabb | 10 | 2022-12-16 |
| 8 | bbaa | 20 | 2022-12-17 |
------------------------------
```

Следующий запрос удаляет из таблицы `test_table` строки, в которых значения в столбце `b` содержат текст `aa`:

```
DELETE FROM test_table WHERE b LIKE '%aa%';
```

Результат выполнения запроса:

```
--a---b------c---------date--
| 1 | a   | 10 | 2022-12-10 |
| 3 | c   | 30 | 2022-12-12 |
| 4 | d   | 40 | 2022-12-13 |
| 5 | eee |  5 | 2022-12-14 |
| 6 | fff |  6 | 2022-12-15 |
-----------------------------
```

Сжатие данных

Елена Дворядкина

Содержание

Назад к содержанию

- Кодеки сжатия данных
  - Кодеки общего назначения
  - Специализированные кодеки
- Применение кодеков сжатия
  - На уровне сервера
  - На уровне столбцов
- Общие рекомендации
- Пример

Важную роль в обеспечении высокой производительности ADQM играет сжатие — или компрессия — данных (data compression). Например, с помощью сжатия можно уменьшить объем данных, считываемых при выполнении запроса, и тем самым ускорить этот запрос — см. пример в статье Повышение производительности запросов.

ADQM поддерживает сжатие данных для следующих табличных движков:

- MergeTree — возможно установить алгоритм сжатия по умолчанию для таблиц, а также указать свой метод сжатия (или их комбинацию) для каждого отдельного столбца при создании/модификации таблицы (см. раздел Применение кодеков сжатия ниже).
- Log — метод сжатия LZ4 используется по умолчанию, поддерживается возможность устанавливать сжатие по столбцам.
- Set, Join — поддерживается только сжатие по умолчанию.

Кодеки сжатия данных

Алгоритм сжатия, применяемый к данным ADQM, определяется **кодеком**. В ADQM доступны следующие кодеки сжатия данных:

- Кодеки общего назначения, позволяющие находить оптимальное соотношение между потреблением дискового пространства и ресурсов CPU.
- Специализированные кодеки для определенных типов данных. Некоторые из этих кодеков не сжимают данные самостоятельно, а выполняют их предварительную обработку (учитывая особенности типа данных) для кодеков общего назначения, которые в свою очередь сжимают подготовленные данные эффективнее, чем неподготовленные.

Кодеки общего назначения

Кодек	Алгоритм сжатия	Уровень сжатия (level)
NONE	Сжатие данных не применяется	—
LZ4	Алгоритм сжатия без потерь LZ4. Обеспечивает высокую скорость и хорошую степень сжатия. Используется по умолчанию	—
LZ4HC[(level)]	Алгоритм LZ4 HC (high compression) с настраиваемым уровнем сжатия. По сравнению с LZ4 сжимает данные лучше, но медленнее, при этом обеспечивает такую же быструю декомпрессию	Возможные уровни сжатия — `[1, 12]`, рекомендуемый диапазон — `[4, 9]`. Уровень сжатия по умолчанию (если не указан или указано значение, меньшее или равное `0`) — `9`
ZSTD[(level)]	Алгоритм сжатия ZSTD. Обеспечивает как скорость, так и высокую степень сжатия, которую можно настроить. Без применения специализированного кодека подготовки данных часто превосходит другие методы сжатия или, по крайней мере, конкурентоспособен	Возможные уровни сжатия — `[1, 22]`. Уровень сжатия по умолчанию — `1`. Уровни сжатия выше `3` редко приводят к значительному улучшению результата, однако высокие уровни сжатия (которые предполагают лучшее сжатие, но большее использование CPU) могут быть полезны для асимметричных сценариев — например, "одно сжатие, многократная декомпрессия". В любом случае рекомендуется проводить тестирование
ZSTD_QAT[(level)]	Алгоритм сжатия ZSTD с настраиваемым уровнем сжатия, который реализуют Intel QATlib и Intel QAT ZSTD Plugin. Особенности использования: - Кодек ZSTD_QAT отключен по умолчанию и может использоваться только после включения параметра: `SET enable_zstd_qat_codec = 1`. - Для сжатия ZSTD_QAT пытается использовать устройство разгрузки Intel QuickAssist Technology (QAT). Если такое устройство не найдено, сжатие выполняется программно. - Декомпрессия всегда выполняется программно.	Возможные уровни сжатия — `[1, 12]`, рекомендуемый диапазон — `[6, 12]`. Уровень сжатия по умолчанию — `1`
DEFLATE_QPL	Алгоритм сжатия Deflate, реализованный Intel Query Processing Library (Intel QPL). Особенности использования: - Кодек DEFLATE_QPL отключен по умолчанию и может использоваться только после включения параметра: `SET enable_deflate_qpl_codec = 1`. - DEFLATE_QPL работает лучше всего, если в системе установлено устройство разгрузки Intel In-Memory Analytics Accelerator (IAA) — см. Accelerator Configuration и Benchmark with DEFLATE_QPL. Техническая поддержка по этому кодеку для ADQM предоставляется в ограниченной форме	—

Специализированные кодеки

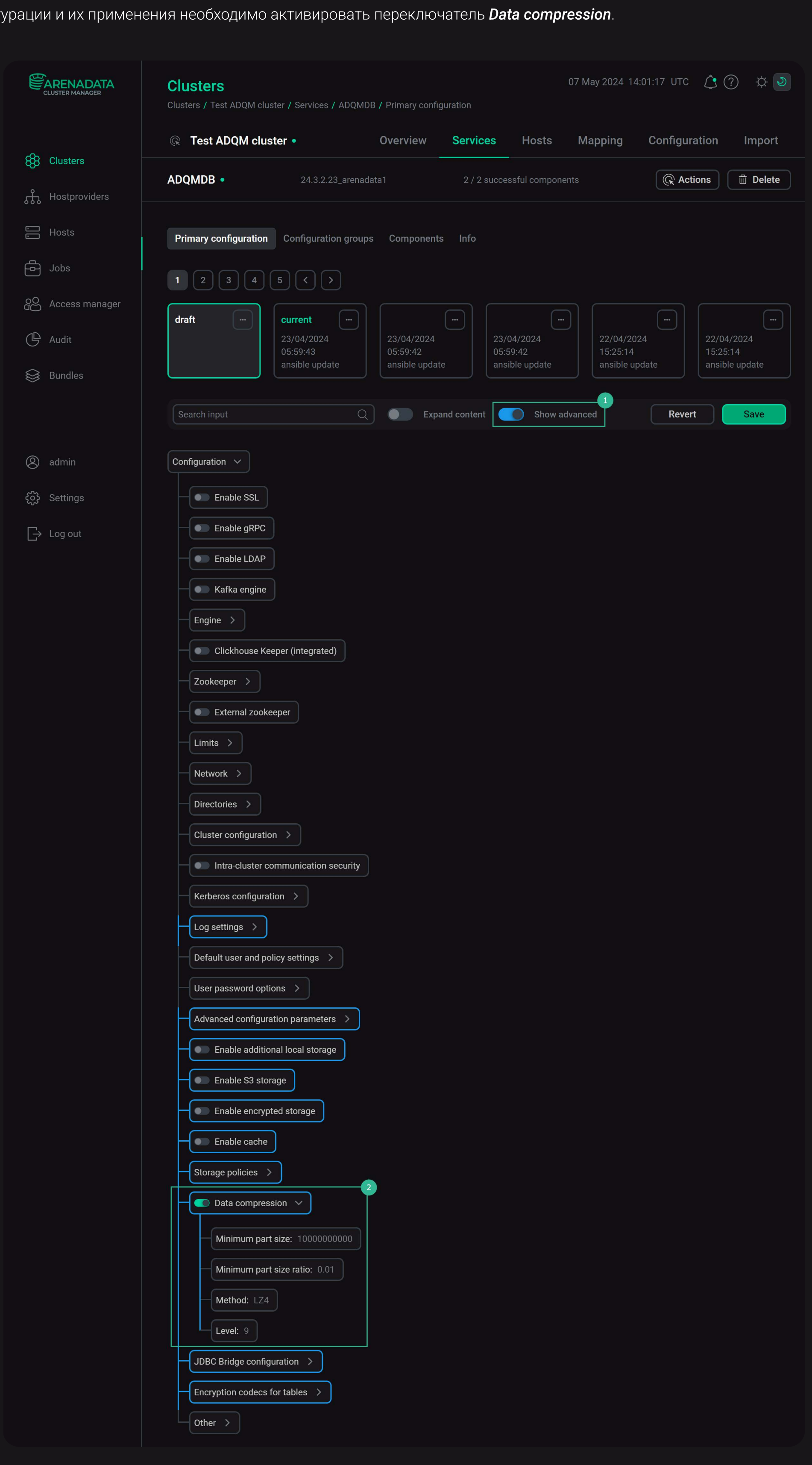
Кодек	Описание	Параметры
Delta(delta_bytes)	Заменяет исходные значения разницей (дельтой) двух соседних значений (за исключением первого значения, которое остается неизменным). Хорошо работает с монотонно возрастающими значениями. Delta — кодек подготовки данных для дальнейшего сжатия, то есть его нельзя использовать отдельно (без дополнительного кодека сжатия)	`delta_bytes` — ограничение (в байтах) для хранения разниц, то есть это максимальный размер исходных значений. Возможные значения `delta_bytes` — `1`, `2`, `4`, `8`. Значение по умолчанию равно `sizeof(type)`, если результат `1`, `2`, `4` или `8`. Во всех остальных случаях — `1`
DoubleDelta(bytes_size)	Вычисляет разницу между соседними дельтами и сохраняет ее в компактном бинарном виде. Оптимальная степень сжатия достигается для монотонных последовательностей с постоянным шагом (например, временных рядов). Может использоваться с любым типом данных фиксированного размера. Реализует алгоритм, используемый в Gorilla TSDB, адаптируя его для работы с 64-битными типами данных. Использует 1 дополнительный бит для 32-битных дельт: 5-битные префиксы вместо 4-битных. Подробнее читайте в разделе Compressing Time Stamps статьи ***Gorilla: A Fast, Scalable, In-Memory Time Series Database***. DoubleDelta — кодек подготовки данных для дальнейшего сжатия, то есть его нельзя использовать отдельно (без дополнительного кодека сжатия)	Возможные значения `bytes_size` — `1`, `2`, `4`, `8`. Значение по умолчанию равно `sizeof(type)`, если результат `1`, `2`, `4` или `8`. Во всех остальных случаях — `1`
GCD	Вычисляет наибольший общий знаменатель (Greatest Common Denominator, GCD) значений в столбце, затем делит каждое значение на GCD. Может использоваться с целочисленными, десятичными столбцами и столбцами типа DateTime. Также хорошо подходит для столбцов, значения которых изменяются (увеличиваются или уменьшаются) кратно GCD — например, `24`, `28`, `16`, `24`, `8`, `24` (`GCD = 4`). GCD — кодек подготовки данных для дальнейшего сжатия, то есть его нельзя использовать отдельно (без дополнительного кодека сжатия)	—
Gorilla(bytes_size)	Вычисляет XOR между текущим и предыдущим значением с плавающей запятой и записывает результат в компактной бинарной форме. Чем меньше разница между последовательными значениями (то есть чем медленнее изменяются значения ряда), тем лучше степень сжатия. Реализует алгоритм, используемый в Gorilla TSDB, адаптируя его для работы с 64-битными значениями. Дополнительную информацию можно найти в разделе Compressing values статьи ***Gorilla: A Fast, Scalable, In-Memory Time Series Database***	Возможные значения `bytes_size` — `1`, `2`, `4`, `8`. Значение по умолчанию равно `sizeof(type)`, если результат `1`, `2`, `4` или `8`. Во всех остальных случаях — `1`
FPC(level, float_size)	Повторно прогнозирует следующее значение с плавающей запятой в последовательности, используя лучший из двух предикторов, затем вычисляет XOR фактического значения с предсказанным значением и сжимает результат с ведущим нулем. Подобно Gorilla, FPC эффективно сохраняет ряды медленно изменяющихся значений с плавающей запятой. Для 64-битных значений FPC работает быстрее, чем Gorilla; для 32-битных значений — оценка может измениться. См. подробное описание алгоритма в статье High Throughput Compression of Double-Precision Floating-Point Data	Возможные значения параметра `level` — `[1, 28]`, значение по умолчанию — `12`. Возможные значения `float_size` — `4`, `8`, значение по умолчанию — `sizeof(type)`, если тип — Float. Во всех остальных случаях — `4`
T64	Обрезает неиспользуемые старшие биты целочисленных значений (включая Enum, Date и DateTime). На каждом шаге алгоритма кодек помещает блок из 64 значений в битовую матрицу размером 64x64, транспонирует ее, обрезает неиспользуемые биты значений и возвращает то, что осталось, в виде последовательности. Неиспользуемые биты — это биты, которые не изменяются от минимального к максимальному на всем диапазоне значений куска данных, для которого используется сжатие	—

Кодеки DoubleDelta и Gorilla используются в Gorilla TSDB в качестве компонентов алгоритма сжатия. Подход Gorilla эффективен в сценариях, когда данные представляют собой последовательность медленно меняющихся значений с их временными метками. Временные метки эффективно сжимаются кодеком DoubleDelta, а значения — кодеком Gorilla.

Применение кодеков сжатия

На уровне сервера

По умолчанию к столбцам таблиц семейства MergeTree ADQM применяет метод сжатия, определенный в конфигурации сервера. Изменить настройки сжатия по умолчанию можно в секции ***Data compression*** на странице конфигурации сервиса ADQMDB. Эта секция становится видимой при включенной опции ***Show advanced***. Для доступа к параметрам конфигурации и их применения необходимо активировать переключатель ***Data compression***.



Minimum part size: 10000000000

Minimum part size ratio: 0.01

Method: LZ4

Level: 9

Настройка сжатия данных таблиц MergeTree на уровне сервера

После определения параметров сжатия данных нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB.

ПРИМЕЧАНИЕ

Изменение настроек сжатия по умолчанию не влияет на уже существующие таблицы.

На уровне столбцов

Метод сжатия можно указать для каждого столбца отдельно в запросе CREATE TABLE с помощью выражения `CODEC`:

```
CREATE TABLE <table_name>
(   <column_name1> <data_type1> CODEC(<compression_codec1>) [...],
    <column_name2> <data_type2> CODEC(<compression_codec2>) [...],
    ...)
ENGINE = MergeTree()
...;
```

Если для столбца указан кодек Default (`CODEC(Default)`) или кодек не указан, используется алгоритм сжатия по умолчанию, установленный на уровне сервера.

Кодеки можно последовательно комбинировать — например, при использовании специализированного кодека, чтобы подготовить данные для более эффективного сжатия кодеком общего назначения (`CODEC(Delta, ZSTD)`).

Изменить кодек сжатия столбца уже существующей таблицы можно с помощью следующего запроса:

```
ALTER TABLE <table_name> MODIFY COLUMN <column_name> CODEC(<new_compression_codec>);
```

Общие рекомендации

Для наиболее эффективного сжатия данных рекомендуется:

- Выбирать наиболее подходящий под конкретную задачу алгоритм сжатия, учитывая тип и характер данных (если известны).
- При выборе уровня сжатия данных следует помнить: чем выше этот уровень — тем больше, но дольше по времени сжатие данных. Фактически сжатие данных позволяет улучшить показатели I/O и уменьшить размер данных за счет ресурсов CPU. Постарайтесь найти оптимальные настройки — чтобы избежать длительного сжатия и медленного сканирования данных.
- Конечная производительность сжатых таблиц зависит от множества факторов, включая аппаратное обеспечение, параметры запросов и так далее. Всегда проводите тестирование — чтобы определить, насколько эффективно сжатие данных в конкретном окружении.

РЕКОМЕНДАЦИЯ

Необходимо всегда соблюдать баланс между доступными ресурсами CPU и требуемой скоростью сжатия.

Пример

Создайте таблицу MergeTree со столбцами типа Int64 и Float64, применяя к ним кодеки сжатия данных LZ4 и ZSTD в комбинации с различными специализированными кодеками подготовки данных:

```
CREATE TABLE compression_test (
    n Int32,
    i64 Int64 DEFAULT n CODEC(NONE),
    i64_lz4 Int64 DEFAULT n CODEC(LZ4),
    i64_delta_lz4 Int64 DEFAULT n CODEC(Delta, LZ4),
    i64_doubledelta_lz4 Int64 DEFAULT n CODEC(DoubleDelta, LZ4),
    i64_t64_lz4 Int64 DEFAULT n CODEC(T64, LZ4),
    i64_zstd Int64 DEFAULT n CODEC(ZSTD),
    i64_delta_zstd Int64 DEFAULT n CODEC(Delta, ZSTD),
    i64_doubledelta_zstd Int64 DEFAULT n CODEC(DoubleDelta, ZSTD),
    i64_t64_zstd Int64 DEFAULT n CODEC(T64, ZSTD),
    f64 Float64 DEFAULT n CODEC(NONE),
    f64_lz4 Float64 DEFAULT n CODEC(LZ4),
    f64_delta_lz4 Float64 DEFAULT n CODEC(Delta, LZ4),
    f64_doubledelta_lz4 Float64 DEFAULT n CODEC(DoubleDelta, LZ4),
    f64_gorilla_lz4 Float64 DEFAULT n CODEC(Gorilla, LZ4),
    f64_fpc_lz4 Float64 DEFAULT n CODEC(FPC, LZ4),
    f64_zstd Float64 DEFAULT n CODEC(ZSTD),
    f64_delta_zstd Float64 DEFAULT n CODEC(Delta, ZSTD),
    f64_doubledelta_zstd Float64 DEFAULT n CODEC(DoubleDelta, ZSTD),
    f64_gorilla_zstd Float64 DEFAULT n CODEC(Gorilla, ZSTD),
    f64_fpc_zstd Float64 DEFAULT n CODEC(FPC, ZSTD))
Engine = MergeTree()
ORDER BY tuple();
```

Сгенерируйте тестовые данные для таблицы (1 миллион строк) так, чтобы каждый столбец содержал монотонную последовательность чисел со случайным инкрементом:

```
INSERT INTO compression_test (n) SELECT number*1000+(rand()%100) FROM numbers(1000000);
```

Используйте системную таблицу system.columns, чтобы посмотреть размер данных в сжатом и несжатом виде для каждого столбца и кодеки сжатия столбцов:

```
SELECT
    name,
    type,
    formatReadableSize(data_uncompressed_bytes) AS uncompressed,
    formatReadableSize(data_compressed_bytes) AS compressed,
    round(data_uncompressed_bytes / data_compressed_bytes, 2) AS ratio,
compression_codec codec
FROM system.columns
WHERE table = 'compression_test' AND name != 'n';
```

В результате этого тестирования видно, что применение специализированных кодеков может существенно повлиять на уровень сжатия данных кодеками общего назначения в зависимости от типа данных столбца. Например, для типа Int64 оба алгоритма (LZ4 и ZSTD) обеспечивают лучшее сжатие вместе с кодеком DoubleDelta, а для типа Float64 — ZSTD вместе с Delta или FPC.

```
    ┌─name─────────────────┬─type────┬─uncompressed─┬─compressed─┬─ratio─┬─codec────────────────────────┐
 1. │ i64                  │ Int64   │ 7.63 MiB     │ 7.63 MiB   │     1 │ CODEC(NONE)                  │
 2. │ i64_lz4              │ Int64   │ 7.63 MiB     │ 4.78 MiB   │   1.6 │ CODEC(LZ4)                   │
 3. │ i64_delta_lz4        │ Int64   │ 7.63 MiB     │ 2.89 MiB   │  2.64 │ CODEC(Delta(8), LZ4)         │
 4. │ i64_doubledelta_lz4  │ Int64   │ 7.63 MiB     │ 1.23 MiB   │  6.21 │ CODEC(DoubleDelta, LZ4)      │
 5. │ i64_t64_lz4          │ Int64   │ 7.63 MiB     │ 1.62 MiB   │  4.72 │ CODEC(T64, LZ4)              │
 6. │ i64_zstd             │ Int64   │ 7.63 MiB     │ 1.97 MiB   │  3.87 │ CODEC(ZSTD(1))               │
 7. │ i64_delta_zstd       │ Int64   │ 7.63 MiB     │ 1.36 MiB   │  5.62 │ CODEC(Delta(8), ZSTD(1))     │
 8. │ i64_doubledelta_zstd │ Int64   │ 7.63 MiB     │ 1.22 MiB   │  6.23 │ CODEC(DoubleDelta, ZSTD(1))  │
 9. │ i64_t64_zstd         │ Int64   │ 7.63 MiB     │ 1.42 MiB   │  5.36 │ CODEC(T64, ZSTD(1))          │
10. │ f64                  │ Float64 │ 7.63 MiB     │ 7.63 MiB   │     1 │ CODEC(NONE)                  │
11. │ f64_lz4              │ Float64 │ 7.63 MiB     │ 5.01 MiB   │  1.52 │ CODEC(LZ4)                   │
12. │ f64_delta_lz4        │ Float64 │ 7.63 MiB     │ 2.90 MiB   │  2.63 │ CODEC(Delta(8), LZ4)         │
13. │ f64_doubledelta_lz4  │ Float64 │ 7.63 MiB     │ 3.47 MiB   │   2.2 │ CODEC(DoubleDelta, LZ4)      │
14. │ f64_gorilla_lz4      │ Float64 │ 7.63 MiB     │ 2.66 MiB   │  2.87 │ CODEC(Gorilla, LZ4)          │
15. │ f64_fpc_lz4          │ Float64 │ 7.63 MiB     │ 3.11 MiB   │  2.45 │ CODEC(FPC(12), LZ4)          │
16. │ f64_zstd             │ Float64 │ 7.63 MiB     │ 2.22 MiB   │  3.44 │ CODEC(ZSTD(1))               │
17. │ f64_delta_zstd       │ Float64 │ 7.63 MiB     │ 1.49 MiB   │  5.11 │ CODEC(Delta(8), ZSTD(1))     │
18. │ f64_doubledelta_zstd │ Float64 │ 7.63 MiB     │ 2.05 MiB   │  3.72 │ CODEC(DoubleDelta, ZSTD(1))  │
19. │ f64_gorilla_zstd     │ Float64 │ 7.63 MiB     │ 2.32 MiB   │  3.29 │ CODEC(Gorilla, ZSTD(1))      │
20. │ f64_fpc_zstd         │ Float64 │ 7.63 MiB     │ 1.87 MiB   │  4.08 │ CODEC(FPC(12), ZSTD(1))      │
    └──────────────────────┴─────────┴──────────────┴────────────┴───────┴──────────────────────────────┘
```

При этом, например, на случайных данных Delta и DoubleDelta не очень хорошо работают как видно в примере ниже. Такие данные лучше сжимаются с использованием T64 для целочисленных значение и Gorilla для чисел с плавающей запятой — то есть эти кодеки могут быть рекомендованы к использованию, когда характер данных неизвестен.

```
CREATE TABLE compression_test_rand AS compression_test;
```

```
INSERT INTO compression_test_rand (n) SELECT rand()%(1000000) FROM numbers(1000000);
```

```
SELECT
    name,
    type,
    formatReadableSize(data_uncompressed_bytes) AS uncompressed,
    formatReadableSize(data_compressed_bytes) AS compressed,
    round(data_uncompressed_bytes / data_compressed_bytes, 2) AS ratio,
compression_codec codec
FROM system.columns
WHERE table = 'compression_test_rand' AND name != 'n';
```

```
    ┌─name─────────────────┬─type────┬─uncompressed─┬─compressed─┬─ratio─┬─codec────────────────────────┐
 1. │ i64                  │ Int64   │ 7.63 MiB     │ 7.63 MiB   │     1 │ CODEC(NONE)                  │
 2. │ i64_lz4              │ Int64   │ 7.63 MiB     │ 4.40 MiB   │  1.73 │ CODEC(LZ4)                   │
 3. │ i64_delta_lz4        │ Int64   │ 7.63 MiB     │ 4.63 MiB   │  1.65 │ CODEC(Delta(8), LZ4)         │
 4. │ i64_doubledelta_lz4  │ Int64   │ 7.63 MiB     │ 4.43 MiB   │  1.72 │ CODEC(DoubleDelta, LZ4)      │
 5. │ i64_t64_lz4          │ Int64   │ 7.63 MiB     │ 2.40 MiB   │  3.18 │ CODEC(T64, LZ4)              │
 6. │ i64_zstd             │ Int64   │ 7.63 MiB     │ 3.32 MiB   │   2.3 │ CODEC(ZSTD(1))               │
 7. │ i64_delta_zstd       │ Int64   │ 7.63 MiB     │ 3.42 MiB   │  2.23 │ CODEC(Delta(8), ZSTD(1))     │
 8. │ i64_doubledelta_zstd │ Int64   │ 7.63 MiB     │ 3.84 MiB   │  1.99 │ CODEC(DoubleDelta, ZSTD(1))  │
 9. │ i64_t64_zstd         │ Int64   │ 7.63 MiB     │ 2.39 MiB   │  3.19 │ CODEC(T64, ZSTD(1))          │
10. │ f64                  │ Float64 │ 7.63 MiB     │ 7.63 MiB   │     1 │ CODEC(NONE)                  │
11. │ f64_lz4              │ Float64 │ 7.63 MiB     │ 4.67 MiB   │  1.63 │ CODEC(LZ4)                   │
12. │ f64_delta_lz4        │ Float64 │ 7.63 MiB     │ 4.85 MiB   │  1.57 │ CODEC(Delta(8), LZ4)         │
13. │ f64_doubledelta_lz4  │ Float64 │ 7.63 MiB     │ 5.94 MiB   │  1.28 │ CODEC(DoubleDelta, LZ4)      │
14. │ f64_gorilla_lz4      │ Float64 │ 7.63 MiB     │ 3.12 MiB   │  2.44 │ CODEC(Gorilla, LZ4)          │
15. │ f64_fpc_lz4          │ Float64 │ 7.63 MiB     │ 5.03 MiB   │  1.52 │ CODEC(FPC(12), LZ4)          │
16. │ f64_zstd             │ Float64 │ 7.63 MiB     │ 3.26 MiB   │  2.34 │ CODEC(ZSTD(1))               │
17. │ f64_delta_zstd       │ Float64 │ 7.63 MiB     │ 3.44 MiB   │  2.22 │ CODEC(Delta(8), ZSTD(1))     │
18. │ f64_doubledelta_zstd │ Float64 │ 7.63 MiB     │ 4.07 MiB   │  1.87 │ CODEC(DoubleDelta, ZSTD(1))  │
19. │ f64_gorilla_zstd     │ Float64 │ 7.63 MiB     │ 3.05 MiB   │   2.5 │ CODEC(Gorilla, ZSTD(1))      │
20. │ f64_fpc_zstd         │ Float64 │ 7.63 MiB     │ 4.20 MiB   │  1.82 │ CODEC(FPC(12), ZSTD(1))      │
    └──────────────────────┴─────────┴──────────────┴────────────┴───────┴──────────────────────────────┘
```

Использование TTL для управления данными

Елена Дворядкина

Содержание

Назад к содержанию

Обзор
Когда срабатывают события TTL
TTL таблицы
- Создание таблицы с TTL
- Изменение TTL
- Удаление TTL
- Примеры

TTL столбцов
- Установка TTL для столбцов
- Изменение TTL
- Удаление TTL
- Примеры

Обзор

Для таблиц семейства MergeTree доступна возможность устанавливать TTL (time-to-live) — интервал времени, по истечении которого старые данные будут удаляться в фоновом режиме. TTL можно установить как для всей таблицы (будут удаляться целые строки данных с устаревшими значениями), так и для каждого отдельного столбца (устаревшие значения в столбце будут заменяться значениями по умолчанию для типа данных столбца).

С помощью TTL на уровне таблицы можно также запланировать автоматическое выполнение следующих действий для строк данных:

- Перемещение данных между дисками или томами хранилища ADQM — может быть полезно при развертывании архитектуры с разделением данных на "горячие", "теплые" и "холодные" (см. Настройка хранилищ данных).
- Обобщение данных (агрегирование и выполнение различных вычислений) — можно снизить уровень детализации старых данных в целях экономии ресурсов для их хранения.
- Применение более высокого уровня сжатия для старых данных.

Таким образом, управляя жизненным циклом данных с помощью механизма TTL, можно оптимизировать хранилище и повысить производительность запросов.

Когда срабатывают события TTL

Удаление строк с истекшим сроком хранения, установленным через TTL, не происходит немедленно — оно выполняется при слиянии кусков данных. ADQM с определенной периодичностью выполняет "внеплановые" слияния для удаления или сжатия данных по правилам TTL. Изменить частоту выполнения этих слияний можно с помощью специальных параметров таблицы MergeTree:

- `merge_with_ttl_timeout` — минимальное время в секундах между слияниями для удаления данных по условию TTL;
- `merge_with_recompression_ttl_timeout` — минимальное время в секундах между слияниями для сжатия данных по условию TTL.

Значения этих параметров по умолчанию — 14400 секунд. Это значит, что удаление/сжатие устаревших по TTL данных будет выполняться каждые 4 часа или при фоновом слиянии кусков данных. Если нужно, чтобы эти операции применялись чаще, измените приведенные выше настройки. При этом стоит учитывать, что если значения этих параметров слишком низкие, будет выполняться много "внеплановых" слияний, а это может привести к потреблению значительной части ресурсов сервера — поэтому не рекомендуется использовать значения меньше 300 секунд.

Также в таблицах MergeTree есть настройка `ttl_only_drop_parts`, значение которой влияет на удаление данных согласно TTL следующим образом:

- `0` (по умолчанию) — удаляются только записи, устаревшие в соответствии с их временем жизни (TTL);
- `1` — удаляется кусок данных целиком, когда в нем все записи устарели.

Удаление целых кусков данных вместо удаления отдельных записей позволяет установить меньший тайм-аут `merge_with_ttl_timeout` и уменьшить влияние операций TTL на производительность системы.

Операции перемещения между дисками/томами или повторного сжатия данных выполняются только когда устаревают все данные в куске данных.

При выполнении запроса `SELECT` между слияниями можно получить устаревшие данные в таблице. Чтобы этого избежать, перед выборкой можно выполнить запрос `OPTIMIZE … FINAL` (форсирует слияние кусков данных), но не рекомендуется делать это слишком часто, особенно для больших таблиц.

TTL таблицы

Создание таблицы с TTL

Чтобы установить TTL для таблицы при ее создании, используйте выражение `TTL` в конце описания таблицы в запросе CREATE TABLE. После выражения `TTL` можно указать одно правило для удаления старых данных и/или несколько правил TTL для обработки данных другим способом (например, для перемещения данных на другой диск/том или изменения уровня их сжатия).

Синтаксис запроса для создания таблицы с TTL в общем виде:

```
CREATE TABLE <table_name>
(   <timestamp_column> Date|DateTime ...,
    <column1> ...,
    <column2> ...,
    ...)
ENGINE = MergeTree()
TTL <ttl_expr> [<ttl_action>] [, <ttl_expr2> [<ttl_action2>]] [, ...];
```

где каждое правило TTL включает:

- `<ttl_expr>` — выражение TTL, которое определяет когда данные будут считаться устаревшими;
- `<ttl_action>` — действие, которое будет выполняться по истечении определяемого выражением TTL времени.

Выражение TTL должно возвращать значение типа Date или DateTime и обычно указывается в следующем формате:

```
<timestamp_column> [+ INTERVAL <time_interval> <interval_type>]
```

где:

- `<timestamp_column>` — имя столбца типа Date или DateTime с временными метками, относительно которых будет определяться время хранения данных;
- `INTERVAL <time_interval> <interval_type>` — значение типа Interval, которое указывает длительность хранения строки данных относительно временной метки в столбце `<timestamp_column>`. В структуру этого значения входит интервал времени как целое число без знака и тип интервала — например, `1 MONTH`, `3 DAY`, `5 HOUR`.

Чтобы указать интервал времени, вместо оператора `INTERVAL` можно использовать функцию преобразования toInterval. В этом случае выражение TTL может выглядеть, например, следующим образом:

```
<timestamp_column> toIntervalMonth(1)
<timestamp_column> toIntervalDay(3)
<timestamp_column> toIntervalHour(5)
```

Действия, которые должны выполняться для строк данных с истекшим согласно выражению TTL временем хранения, указываются с помощью перечисленных ниже выражений.

Выражение	Синтаксис TTL в общем виде	Описание действия
DELETE	`TTL <ttl_expr> [DELETE] [WHERE <filter_expr>]`	Удаление строк данных с истекшим сроком хранения. Это действие применяется по умолчанию (то есть строки удаляются, если после выражения `TTL` явно не указано иное действие)
TO DISK	`TTL <ttl_expr> TO DISK '<disk_name>'`	Перемещение данных на указанный диск
TO VOLUME	`TTL <ttl_expr> TO VOLUME '<volume_name>'`	Перемещение данных на указанный том
RECOMPRESS	`TTL <ttl_expr> RECOMPRESS CODEC(<compression_codec>)`	Сжатие данных указанным кодеком
GROUP BY		Агрегирование данных. При определении правила агрегирования устаревших данных необходимо учитывать следующие особенности: - Столбцы выражения `GROUP BY` (`<key_expr>`), по которым агрегируются данные, должны быть префиксом первичного ключа таблицы. - Если столбец таблицы не является частью выражения `GROUP BY` и не указан явно в выражении `SET`, в результирующих строках он будет содержать случайное значение, взятое из одной из сгруппированных строк (как будто к нему применяется агрегатная функция `any`).

GROUP BY

```
TTL <ttl_expr>
    GROUP BY <key_expr>
    [SET <column_name> = aggr_func(<column_name>)[,
...]]
    [WHERE <filter_expr>]
```

С помощью выражения `WHERE` можно указать условие удаления (`DELETE`) или агрегирования (`GROUP BY`) устаревших строк. Для перемещения и сжатия данных условие `WHERE` не применимо.

Изменение TTL

Изменить TTL существующей таблицы можно с помощью следующего запроса:

```
ALTER TABLE <table_name> MODIFY TTL <new_ttl_rules>;
```

После выполнения этого запроса при каждой вставке новых данных или создании новых кусков данных в результате фонового слияния TTL будет рассчитываться в соответствии с новым правилом.

Чтобы пересчитать TTL для уже существующих данных, необходимо выполнить материализацию TTL. Настройка `materialize_ttl_after_modify` (включена по умолчанию) контролирует, будет ли материализация TTL осуществляться автоматически после выполнения запроса `ALTER TABLE … MODIFY TTL`. Также можно форсировать материализацию TTL следующим запросом:

```
ALTER TABLE <table_name> MATERIALIZE TTL;
```

Материализация TTL может быть достаточно тяжелой операцией (особенно для больших таблиц), так как при этом пересчитывается TTL во всех кусках данных (обновляются файлы *ttl.txt*) и перезаписываются данные во всех столбцах, что может привести к большому количеству операций чтения/записи. Чтобы облегчить операцию материализации TTL, можно в настройках таблицы MergeTree установить параметр `materialize_ttl_recalculate_only = 1` — в этом случае будет выполняться только обновление *ttl.txt* (довольно легкая операция, так как считываются только столбцы, используемые в TTL) и копирование столбцов через жесткие ссылки, а через некоторое время данные с истекшим сроком хранения будут удалены во время слияния при включенной настройке `ttl_only_drop_parts`.

В некоторых случаях оптимальным решением будет отключить материализацию TTL (`materialize_ttl_after_modify = 0`) и удалить/переместить старые партиции вручную (с помощью запросов `ALTER TABLE … DROP|MOVE PARTITION` — см. Манипуляции с партициями и кусками данных).

Удаление TTL

Чтобы удалить правила TTL из таблицы, используйте запрос:

```
ALTER TABLE <table_name> REMOVE TTL;
```

Примеры

Автоматическое удаление данных с истекшим сроком хранения

1. Создайте таблицу с правилом TTL для удаления строк, которые будут считаться устаревшими, если с момента времени в столбце `time` прошло 10 минут:

```
CREATE TABLE test_table (time DateTime, value UInt64)
ENGINE = MergeTree
ORDER BY time
TTL time + INTERVAL 10 MINUTE;
```

Заполните таблицу тестовыми данными:

```
INSERT INTO test_table
SELECT now() - toIntervalMinute(number), number
FROM numbers(10);
```

```
SELECT * FROM test_table;
```

```
    ┌────────────────time─┬─value─┐
 1. │ 2024-06-07 13:15:49 │     9 │
 2. │ 2024-06-07 13:16:49 │     8 │
 3. │ 2024-06-07 13:17:49 │     7 │
 4. │ 2024-06-07 13:18:49 │     6 │
 5. │ 2024-06-07 13:19:49 │     5 │
 6. │ 2024-06-07 13:20:49 │     4 │
 7. │ 2024-06-07 13:21:49 │     3 │
 8. │ 2024-06-07 13:22:49 │     2 │
 9. │ 2024-06-07 13:23:49 │     1 │
10. │ 2024-06-07 13:24:49 │     0 │
    └─────────────────────┴───────┘
```

2. Через 5 минут после вставки данных выполните запрос:

```
OPTIMIZE TABLE test_table FINAL;
```

Повторите запрос на выборку данных из таблицы:

```
SELECT * FROM test_table;
```

В результате видно, что первые пять строк удалены из таблицы, так как для каждой из них прошло 10 минут с момента времени, которое представлено значением столбца `time`:

```
   ┌────────────────time─┬─value─┐
1. │ 2024-06-07 13:20:49 │     4 │
2. │ 2024-06-07 13:21:49 │     3 │
3. │ 2024-06-07 13:22:49 │     2 │
4. │ 2024-06-07 13:23:49 │     1 │
5. │ 2024-06-07 13:24:49 │     0 │
   └─────────────────────┴───────┘
```

Если не использовать запрос `OPTIMIZE TABLE … FINAL` и выполнить запрос `SELECT` между слияниями, в таблице все еще можно увидеть устаревшие данные.

Фильтрация строк для удаления

1. Создайте таблицу оповещений (`alerts`) с правилом TTL для удаления строк со значением `Critical` в столбце `alert` и значением времени в столбце `time`, с момента которого прошло 10 минут:

```
CREATE TABLE alerts (time DateTime, value UInt64, alert String)
ENGINE = MergeTree
ORDER BY time
TTL time + INTERVAL 10 MINUTE DELETE WHERE alert = 'Critical';
```

Заполните таблицу тестовыми данными:

```
INSERT INTO alerts
SELECT now() - toIntervalMinute(number), number, ['Warning', 'Critical', 'Info'][(rand() % 3) + 1]
FROM numbers(10);
```

```
SELECT * FROM alerts;
```

```
    ┌────────────────time─┬─value─┬─alert────┐
 1. │ 2024-06-10 07:14:56 │     9 │ Critical │
 2. │ 2024-06-10 07:15:56 │     8 │ Warning  │
 3. │ 2024-06-10 07:16:56 │     7 │ Info     │
 4. │ 2024-06-10 07:17:56 │     6 │ Critical │
 5. │ 2024-06-10 07:18:56 │     5 │ Warning  │
 6. │ 2024-06-10 07:19:56 │     4 │ Info     │
 7. │ 2024-06-10 07:20:56 │     3 │ Critical │
 8. │ 2024-06-10 07:21:56 │     2 │ Warning  │
 9. │ 2024-06-10 07:22:56 │     1 │ Warning  │
10. │ 2024-06-10 07:23:56 │     0 │ Critical │
    └─────────────────────┴───────┴──────────┘
```

2. Через 10 минут после вставки данных инициируйте слияние кусков данных таблицы и посмотрите ее содержимое:

```
OPTIMIZE TABLE alerts FINAL;
```

```
SELECT * FROM alerts;
```

Из таблицы удалены все старые оповещения типа `Critical`:

```
   ┌────────────────time─┬─value─┬─alert───┐
1. │ 2024-06-10 07:15:56 │     8 │ Warning │
2. │ 2024-06-10 07:16:56 │     7 │ Info    │
3. │ 2024-06-10 07:18:56 │     5 │ Warning │
4. │ 2024-06-10 07:19:56 │     4 │ Info    │
5. │ 2024-06-10 07:21:56 │     2 │ Warning │
6. │ 2024-06-10 07:22:56 │     1 │ Warning │
   └─────────────────────┴───────┴─────────┘
```

Сохранение данных в другую таблицу перед удалением

Если необходимо переместить данные в другую таблицу перед их удалением из основной таблицы, можно использовать TTL в сочетании с материализованным представлением (materialized view).

1. Создайте таблицу `alerts` с правилом TTL для удаления оповещений через 10 минут относительно времени, указанного в столбце `time`, и таблицу `alerts_history`, в которую будут сохраняться оповещения типа `Critical` из таблицы `alerts`:

```
DROP TABLE IF EXISTS alerts;
```

```
CREATE TABLE alerts (time DateTime, value UInt64, alert String)
ENGINE = MergeTree
ORDER BY time
TTL time + INTERVAL 10 MINUTE DELETE;
```

```
CREATE TABLE alerts_history (time DateTime, value UInt64, alert String)
ENGINE = MergeTree
ORDER BY time;
```

Обратите внимание, таблица `alerts_history` имеет ту же структуру, что и `alerts`, но не стоит использовать запрос `CREATE TABLE alerts_history AS alerts`, так как в этом случае правило TTL тоже скопируется, а это не нужно.

2. Создайте материализованное представление для автоматической загрузки оповещений `Critical` в таблицу `alerts_history` при вставке в `alerts`:

```
CREATE MATERIALIZED VIEW alerts_history_mv TO alerts_history AS
SELECT * FROM alerts WHERE alert = 'Critical';
```

3. Вставьте тестовые данные в таблицу `alerts`:

```
INSERT INTO alerts
SELECT now() - toIntervalMinute(number), number, ['Warning', 'Critical', 'Info'][(rand() % 3) + 1]
FROM numbers(10);
```

```
SELECT * FROM alerts;
```

```
    ┌────────────────time─┬─value─┬─alert────┐
 1. │ 2024-06-10 08:02:16 │     9 │ Warning  │
 2. │ 2024-06-10 08:03:16 │     8 │ Warning  │
 3. │ 2024-06-10 08:04:16 │     7 │ Info     │
 4. │ 2024-06-10 08:05:16 │     6 │ Critical │
 5. │ 2024-06-10 08:06:16 │     5 │ Warning  │
 6. │ 2024-06-10 08:07:16 │     4 │ Warning  │
 7. │ 2024-06-10 08:08:16 │     3 │ Critical │
 8. │ 2024-06-10 08:09:16 │     2 │ Warning  │
 9. │ 2024-06-10 08:10:16 │     1 │ Critical │
10. │ 2024-06-10 08:11:16 │     0 │ Warning  │
    └─────────────────────┴───────┴──────────┘
```

4. Подождите 10 минут и выполните запрос:

```
OPTIMIZE TABLE alerts FINAL;
```

В результате, TTL удалит записи из таблицы `alerts`, но записи со значением `Critical` в столбце `alert` будут сохранены в таблице `alerts_history`:

```
SELECT * FROM alerts_history;
```

```
   ┌────────────────time─┬─value─┬─alert────┐
1. │ 2024-06-10 08:05:16 │     6 │ Critical │
2. │ 2024-06-10 08:08:16 │     3 │ Critical │
3. │ 2024-06-10 08:10:16 │     1 │ Critical │
   └─────────────────────┴───────┴──────────┘
```

Агрегирование устаревших данных

1. Создайте таблицу `alerts` и заполните ее тестовыми данными:

```
DROP TABLE IF EXISTS alerts;
```

```
CREATE TABLE alerts (time DateTime, value UInt64, alert String)
ENGINE = MergeTree
ORDER BY (toHour(time), alert);
```

```
INSERT INTO alerts
SELECT now() - toIntervalMinute(number), toInt64(randUniform(0, 100)), ['Warning', 'Critical', 'Info'][(rand() % 3) + 1]
FROM numbers(15);
```

```
SELECT * FROM alerts;
```

```
    ┌────────────────time─┬─value─┬─alert────┐
 1. │ 2024-07-26 21:51:41 │     5 │ Critical │
 2. │ 2024-07-26 21:48:41 │    18 │ Critical │
 3. │ 2024-07-26 21:46:41 │    53 │ Critical │
 4. │ 2024-07-26 21:43:41 │    33 │ Critical │
 5. │ 2024-07-26 21:50:41 │    98 │ Info     │
 6. │ 2024-07-26 21:49:41 │    74 │ Info     │
 7. │ 2024-07-26 21:47:41 │    73 │ Info     │
 8. │ 2024-07-26 21:42:41 │    52 │ Info     │
 9. │ 2024-07-26 21:41:41 │    92 │ Info     │
10. │ 2024-07-26 21:39:41 │    70 │ Info     │
11. │ 2024-07-26 21:38:41 │    81 │ Info     │
12. │ 2024-07-26 21:37:41 │    72 │ Info     │
13. │ 2024-07-26 21:45:41 │     7 │ Warning  │
14. │ 2024-07-26 21:44:41 │    96 │ Warning  │
15. │ 2024-07-26 21:40:41 │     0 │ Warning  │
    └─────────────────────┴───────┴──────────┘
```

2. Через пять минут после вставки данных (относительно сгенерированных выше временных значений — в `21:56:41`) добавьте в таблицу правило TTL, по которому устаревшие строки данных (если с момента времени в столбце `time` прошло 10 минут) будут сгруппированы по времени и типу оповещения, при этом в каждой группе в столбце `value` будет сохраняться максимальное значение, а в столбце `time` — самое последнее время:

```
ALTER TABLE alerts MODIFY TTL time + INTERVAL 10 MINUTE GROUP BY toHour(time), alert SET value = max(value), time =
max(time);
```

Обратите внимание, столбцы `GROUP BY` в TTL должны быть префиксом первичного ключа таблицы (`ORDER BY`, если отдельно не указан `PRIMARY KEY`).

3. Выполните материализацию TTL, чтобы применить правило TTL к существующим данным таблицы:

```
ALTER TABLE alerts MATERIALIZE TTL;
```

4. Проверьте содержимое таблицы:

```
SELECT * FROM alerts;
```

Строки, для которых выполнилось правило TTL (с момента времени `time` прошло 10 минут), сгруппировались, остальные строки остались в прежнем виде:

```
   ┌────────────────time─┬─value─┬─alert────┐
1. │ 2024-07-26 21:51:41 │     5 │ Critical │
2. │ 2024-07-26 21:48:41 │    18 │ Critical │
3. │ 2024-07-26 21:46:41 │    53 │ Critical │
4. │ 2024-07-26 21:50:41 │    98 │ Info     │
5. │ 2024-07-26 21:49:41 │    74 │ Info     │
6. │ 2024-07-26 21:47:41 │    73 │ Info     │
7. │ 2024-07-26 21:42:41 │    92 │ Info     │
8. │ 2024-07-26 21:45:41 │    96 │ Warning  │
   └─────────────────────┴───────┴──────────┘
```

Если еще через 10 минут выполнить оптимизацию, данные будут доагрегированы — в таблице останется по одной строке для самого последнего оповещения каждого типа:

```
OPTIMIZE TABLE alerts FINAL;
```

```
   ┌────────────────time─┬─value─┬─alert────┐
1. │ 2024-07-26 21:51:41 │    53 │ Critical │
2. │ 2024-07-26 21:50:41 │    98 │ Info     │
3. │ 2024-07-26 21:45:41 │    96 │ Warning  │
   └─────────────────────┴───────┴──────────┘
```

Изменение сжатия для устаревших данных

1. Создайте таблицу с правилом TTL, по которому устаревшие данные будут сжиматься кодеком `LZ4HC(10)`, а также укажите параметр `merge_with_recompression_ttl_timeout = 1200` (в секундах), чтобы определить, как часто будет запускаться действие TTL для сжатия старых данных (в данном примере каждые 20 минут):

```
DROP TABLE IF EXISTS alerts;
```

```
CREATE TABLE alerts (time DateTime, value UInt64, alert String)
ENGINE = MergeTree
ORDER BY time
TTL time + INTERVAL 10 MINUTE RECOMPRESS CODEC(LZ4HC(10))
SETTINGS merge_with_recompression_ttl_timeout = 1200;
```

2. Вставьте данные:

```
INSERT INTO alerts
SELECT now() - toIntervalMinute(number), number, ['Warning', 'Critical', 'Info'][(rand() % 3) + 1]
FROM numbers(100);
```

3. Посмотрите информацию о кусках данных таблицы `alerts` в системной таблице `system.parts`:

```
SELECT name, active, rows, data_compressed_bytes, data_uncompressed_bytes
FROM system.parts WHERE table = 'alerts'
FORMAT Vertical;
```

Вставленные 100 строк данных записались в виде одного куска:

```
Row 1:
──────
name:                    all_1_1_0
active:                  1
rows:                    100
data_compressed_bytes:   1114
data_uncompressed_bytes: 1933
```

4. Повторите приведенные выше запросы `INSERT INTO` и `SELECT … FROM system.parts`. В результате в таблице будет два куска данных:

```
Row 1:
──────
name:                    all_1_1_0
active:                  1
rows:                    100
data_compressed_bytes:   1114
data_uncompressed_bytes: 1933

Row 2:
──────
name:                    all_2_2_0
active:                  1
rows:                    100
data_compressed_bytes:   1125
data_uncompressed_bytes: 1931
```

5. Через некоторое время повторите запрос `SELECT … FROM system.parts`. В результате вставленные куски данных сливаются в один кусок `all_1_2_1`, который содержит 200 строк. Размер несжатых данных в этом куске равен сумме несжатых данных объединенных кусков, а размер сжатых данных меньше, так как при слиянии устаревшие данные были сжаты кодеком LZ4HC(10):

```
Row 1:
──────
name:                    all_1_2_1
active:                  1
rows:                    200
data_compressed_bytes:   1509
data_uncompressed_bytes: 3864
```

Перемещение данных между горячим и холодным хранилищами

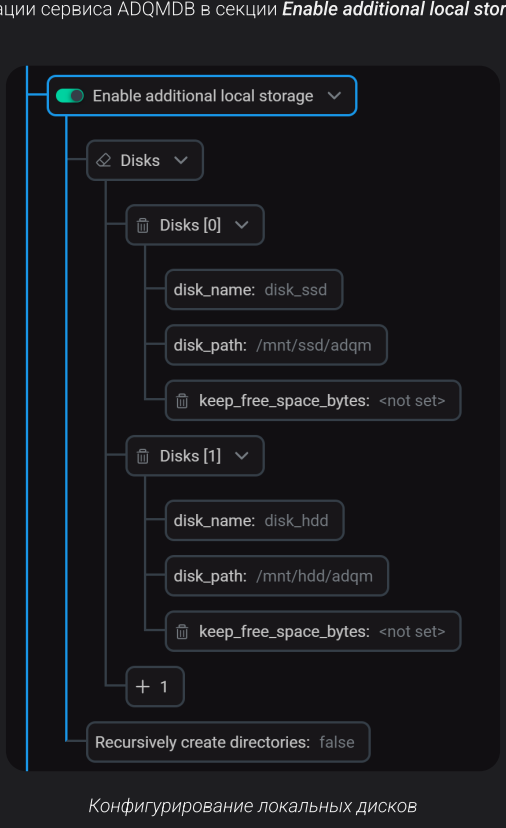
В данном примере для хранения "горячих" и "холодных" данных будут использоваться разные диски — SSD- и HDD-диск, примонтированные в следующие каталоги:

- */mnt/ssd/adqm/*
- */mnt/hdd/adqm/*

Каждому каталогу назначен владелец `clickhouse`:

```
$ sudo chown clickhouse:clickhouse -R /mnt/ssd/adqm/ /mnt/hdd/adqm/
```

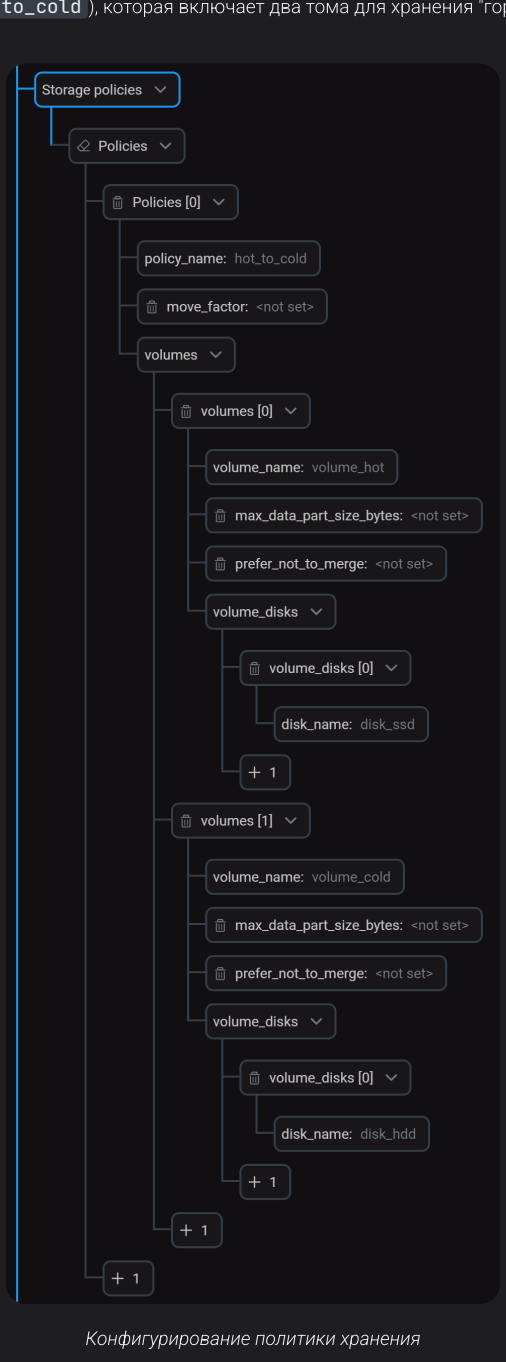
1. В интерфейсе ADCM сконфигурируйте многоуровневое хранилище для ADQM следующим образом.
Добавьте локальные диски на странице конфигурации сервиса ADQMDB в секции ***Enable additional local storage*** (секция должна быть активирована).



disk_name: disk_ssd
disk_path: /mnt/ssd/adqm
keep_free_space_bytes: <not set>
Disks [1]
disk_name: disk_hdd
disk_path: /mnt/hdd/adqm
keep_free_space_bytes: <not set>
Recursively create directories: false

Конфигурирование локальных дисков

В секции ***Storage policies*** создайте политику (`hot_to_cold`), которая включает два тома для хранения "горячих" и "холодных" данных.



policy_name: hot_to_cold
move_factor: <not set>
volume_name: volume_hot
max_data_part_size_bytes: <not set>
prefer_not_to_merge: <not set>
disk_name: disk_ssd
volume_name: volume_cold
max_data_part_size_bytes: <not set>
prefer_not_to_merge: <not set>
disk_name: disk_hdd

Конфигурирование политики хранения

Нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса, чтобы применить изменения.

После этого информация о дисках и политике появится в соответствующих тегах внутри секции `<storage_configuration>` файла */etc/clickhouse-server/config.d/storage.xml*.

storage.xml

```
<storage_configuration>
    <disks>
        <disk_ssd>
            <type>local</type>
            <path>/mnt/ssd/adqm/</path>
        </disk_ssd>
        <disk_hdd>
            <type>local</type>
            <path>/mnt/hdd/adqm/</path>
        </disk_hdd>
    </disks>
    <policies>
        <hot_to_cold>
            <volumes>
                <volume_hot>
                    <disk>disk_ssd</disk>
                </volume_hot>
                <volume_cold>
                    <disk>disk_hdd</disk>
                </volume_cold>
            </volumes>
        </hot_to_cold>
    </policies>
</storage_configuration>
```

Доступные диски можно посмотреть в системной таблице `system.disks`, а политики хранения — в таблице `system.storage_policies`:

```
SELECT name, path, free_space, total_space FROM system.disks;
```

```
   ┌─name─────┬─path─────────────────┬──free_space─┬─total_space─┐
1. │ default  │ /var/lib/clickhouse/ │ 45148381184 │ 53674487808 │
2. │ disk_hdd │ /mnt/hdd/adqm/       │ 45148381184 │ 53674487808 │
3. │ disk_ssd │ /mnt/ssd/adqm/       │ 45148381184 │ 53674487808 │
   └──────────┴──────────────────────┴─────────────┴─────────────┘
```

```
SELECT policy_name, volume_name, disks FROM system.storage_policies;
```

```
   ┌─policy_name─┬─volume_name─┬─disks────────┐
1. │ default     │ default     │ ['default']  │
2. │ hot_to_cold │ volume_hot  │ ['disk_ssd'] │
3. │ hot_to_cold │ volume_cold │ ['disk_hdd'] │
   └─────────────┴─────────────┴──────────────┘
```

2. Создайте таблицу с правилом TTL, по которому данные будут перемещаться между томами `volume_hot` и `volume_cold`:

```
DROP TABLE IF EXISTS alerts;
```

```
CREATE TABLE alerts (time DateTime, value UInt64, alert String)
ENGINE = MergeTree
ORDER BY time
TTL time TO VOLUME 'volume_hot', time + INTERVAL 5 MINUTE TO VOLUME 'volume_cold'
SETTINGS storage_policy = 'hot_to_cold';
```

3. Вставьте в таблицу тестовые данные:

```
INSERT INTO alerts
    SELECT
    now() - toIntervalMinute(number),
    number,
    ['Warning', 'Critical', 'Info'][(rand() % 3) + 1]
FROM numbers(5);
```

Повторите запрос `INSERT INTO` через некоторое время.

В результате в таблице будет два куска данных, расположенных на диске `disk_ssd` (том `volume_hot`):

```
SELECT name, rows, active, disk_name FROM system.parts WHERE table = 'alerts';
```

```
   ┌─name──────┬─rows─┬─active─┬─disk_name─┐
1. │ all_1_1_0 │    5 │      1 │ disk_ssd  │
2. │ all_2_2_0 │    5 │      1 │ disk_ssd  │
   └───────────┴──────┴────────┴───────────┘
```

4. Через 5 минут после первой вставки первый кусок данных будет перемещен на диск `disk_hdd` (том `volume_cold`). Чтобы убедиться в этом, повторите запрос `SELECT … FROM system.parts`:

```
   ┌─name──────┬─rows─┬─active─┬─disk_name─┐
1. │ all_1_1_0 │    5 │      1 │ disk_hdd  │
2. │ all_2_2_0 │    5 │      1 │ disk_ssd  │
   └───────────┴──────┴────────┴───────────┘
```

TTL столбцов

В столбце с установленным TTL по истечении указанного срока хранения значений ADQM заменяет их значениями по умолчанию для типа данных столбца (или значением, указанным через `DEFAULT`). Если все значения столбца в куске данных устарели, этот столбец удаляется из каталога куска данных в файловой системе.

ВАЖНО

TTL нельзя использовать для ключевых столбцов.

Установка TTL для столбцов

Для указания времени хранения значений в столбце используйте выражение `TTL` в описании столбца при создании таблицы с помощью запроса `CREATE TABLE`:

```
CREATE TABLE <table_name>
(   <timestamp_column> Date|DateTime,
    ...,
    <column_name1> [<data_type1>] TTL <timestamp_column> + INTERVAL <time_interval1> <interval_type1>,
    <column_name2> [<data_type2>] TTL <timestamp_column> + INTERVAL <time_interval2> <interval_type2>,
    ...)
ENGINE = MergeTree()
...;
```

Выражение TTL, определяющее когда значения столбца должны быть удалены, указывается тем же способом, что и выражение TTL на уровне таблицы.

Изменение TTL

Чтобы добавить TTL в столбец существующей таблицы или изменить TTL столбца, используйте запрос `ALTER TABLE … MODIFY COLUMN`:

```
ALTER TABLE <table_name> MODIFY COLUMN <column_name> <data_type> TTL <new_ttl_expr>;
```

Удаление TTL

Удалить TTL столбца можно через следующий запрос:

```
ALTER TABLE <table_name> MODIFY COLUMN <column_name> REMOVE TTL;
```

Примеры

Создание таблицы с TTL для столбцов

1. Создайте таблицу и установите следующие правила TTL для столбцов:

   - `value1` — значение в столбце заменяется на `100`, если прошло 5 минут с момента времени, указанного в столбце `time`;
   - `value2` — устанавливается значение по умолчанию (`0` для типа данных Int), если с момента времени `time` прошло 10 минут.

```
CREATE TABLE column_ttl_test
(
    time DateTime,
    value1 Int DEFAULT 100 TTL time + INTERVAL 5 MINUTE,
    value2 Int TTL time + INTERVAL 10 MINUTE,
    alert String
)
ENGINE = MergeTree
ORDER BY time
SETTINGS min_rows_for_wide_part = 0, min_bytes_for_wide_part = 0;
```

Настройки `min_rows_for_wide_part` и `min_bytes_for_wide_part` используются, чтобы куски данных хранились в формате `Wide` (для тестовых целей) — каждый столбец в отдельном файле.

2. Вставьте в таблицу данные:

```
INSERT INTO column_ttl_test
SELECT
        now() - toIntervalMinute(number),
        number,
        number + 10,
        ['Warning', 'Critical', 'Info'][(rand() % 3) + 1]
FROM numbers(10);
```

Первые пять строк в столбце `value1` содержат `100`, так как на момент вставки значения времени в столбце `time` были уже устаревшими:

```
SELECT * FROM column_ttl_test;
```

```
    ┌────────────────time─┬─value1─┬─value2─┬─alert────┐
 1. │ 2024-06-25 11:42:36 │    100 │     19 │ Warning  │
 2. │ 2024-06-25 11:43:36 │    100 │     18 │ Critical │
 3. │ 2024-06-25 11:44:36 │    100 │     17 │ Warning  │
 4. │ 2024-06-25 11:45:36 │    100 │     16 │ Info     │
 5. │ 2024-06-25 11:46:36 │    100 │     15 │ Info     │
 6. │ 2024-06-25 11:47:36 │      4 │     14 │ Info     │
 7. │ 2024-06-25 11:48:36 │      3 │     13 │ Warning  │
 8. │ 2024-06-25 11:49:36 │      2 │     12 │ Info     │
 9. │ 2024-06-25 11:50:36 │      1 │     11 │ Critical │
10. │ 2024-06-25 11:51:36 │      0 │     10 │ Critical │
    └─────────────────────┴────────┴────────┴──────────┘
```

3. Через 3 минут после вставки данных в таблицу, выполните запрос:

```
OPTIMIZE TABLE column_ttl_test FINAL;
```

Снова запросите данные из таблицы. В результате будут заменены значения столбца `value1` в строках 6-8 и столбца `value2` в строках 1-3:

```
    ┌────────────────time─┬─value1─┬─value2─┬─alert────┐
 1. │ 2024-06-25 11:42:36 │    100 │      0 │ Warning  │
 2. │ 2024-06-25 11:43:36 │    100 │      0 │ Critical │
 3. │ 2024-06-25 11:44:36 │    100 │      0 │ Warning  │
 4. │ 2024-06-25 11:45:36 │    100 │     16 │ Info     │
 5. │ 2024-06-25 11:46:36 │    100 │     15 │ Info     │
 6. │ 2024-06-25 11:47:36 │    100 │     14 │ Info     │
 7. │ 2024-06-25 11:48:36 │    100 │     13 │ Warning  │
 8. │ 2024-06-25 11:49:36 │    100 │     12 │ Info     │
 9. │ 2024-06-25 11:50:36 │      1 │     11 │ Critical │
10. │ 2024-06-25 11:51:36 │      0 │     10 │ Critical │
    └─────────────────────┴────────┴────────┴──────────┘
```

4. Еще через 2 минуты (5 минут после первоначальной вставки данных в таблицу) все значения в столбце `value1` будут считаться устаревшими (заменены на `100`):

```
OPTIMIZE TABLE column_ttl_test FINAL;
```

```
SELECT * FROM column_ttl_test;
```

```
    ┌────────────────time─┬─value1─┬─value2─┬─alert────┐
 1. │ 2024-06-25 11:42:36 │    100 │      0 │ Warning  │
 2. │ 2024-06-25 11:43:36 │    100 │      0 │ Critical │
 3. │ 2024-06-25 11:44:36 │    100 │      0 │ Warning  │
 4. │ 2024-06-25 11:45:36 │    100 │      0 │ Info     │
 5. │ 2024-06-25 11:46:36 │    100 │      0 │ Info     │
 6. │ 2024-06-25 11:47:36 │    100 │     14 │ Info     │
 7. │ 2024-06-25 11:48:36 │    100 │     13 │ Warning  │
 8. │ 2024-06-25 11:49:36 │    100 │     12 │ Info     │
 9. │ 2024-06-25 11:50:36 │    100 │     11 │ Critical │
10. │ 2024-06-25 11:51:36 │    100 │     10 │ Critical │
    └─────────────────────┴────────┴────────┴──────────┘
```

После этого столбец `value1` будет удален из каталога куска данных в файловой системе. Проверить это можно следующим образом.

В таблице `system.parts` посмотрите путь к каталогу с файлами активного куска данных (`all_1_1_3`):

```
SELECT name, active, path FROM system.parts WHERE table = 'column_ttl_test';
```

```
   ┌─name──────┬─active─┬─path──────────────────────────────────────────────────────────────────────┐
1. │ all_1_1_0 │      0 │ /var/lib/clickhouse/store/62a/62a16a55-c91c-4c6b-9386-f4347ffac137/all_1_1_0/ │
2. │ all_1_1_1 │      0 │ /var/lib/clickhouse/store/62a/62a16a55-c91c-4c6b-9386-f4347ffac137/all_1_1_1/ │
3. │ all_1_1_2 │      0 │ /var/lib/clickhouse/store/62a/62a16a55-c91c-4c6b-9386-f4347ffac137/all_1_1_2/ │
4. │ all_1_1_3 │      1 │ /var/lib/clickhouse/store/62a/62a16a55-c91c-4c6b-9386-f4347ffac137/all_1_1_3/ │
   └───────────┴────────┴───────────────────────────────────────────────────────────────────────────┘
```

Убедитесь, что в этом каталоге нет файлов с данными столбца `value1` (*value1.bin* и *value1.cmrk2*):

```
$ sudo ls /var/lib/clickhouse/store/62a/62a16a55-c91c-4c6b-9386-f4347ffac137/all_1_1_3/
```

```
alert.bin    checksums.txt  count.txt                     metadata_version.txt  serialization.json  time.cmrk2  value2.bin
alert.cmrk2  columns.txt    default_compression_codec.txt primary.cidx          time.bin            ttl.txt     value2.
cmrk2
```

В файле *columns.txt* нет информации о столбце `value1`:

```
$ sudo cat /var/lib/clickhouse/store/62a/62a16a55-c91c-4c6b-9386-f4347ffac137/all_1_1_3/columns.txt
```

```
columns format version: 1
3 columns:
`time` DateTime
`value2` Int32
`alert` String
```

Изменение TTL столбца в существующей таблице

Используйте запрос `ALTER TABLE … MODIFY COLUMN … TTL`, чтобы изменить TTL столбца `value2` таблицы, созданной в предыдущем примере:

```
ALTER TABLE column_ttl_test MODIFY COLUMN value2 Int TTL time + INTERVAL 1 HOUR;
```

Проверьте текущие настройки TTL с помощью запроса `SHOW CREATE TABLE`:

```
SHOW CREATE TABLE column_ttl_test;
```

Новое правило для столбца `value2` применилось:

```
CREATE TABLE default.column_ttl_test
(
    `time` DateTime,
    `value1` Int32 DEFAULT 100 TTL time + toIntervalMinute(5),
    `value2` Int32 TTL time + toIntervalHour(1),
    `alert` String
)
ENGINE = MergeTree
ORDER BY time
SETTINGS min_rows_for_wide_part = 0, min_bytes_for_wide_part = 0, index_granularity = 8192
```

Индексы

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
- Разреженный первичный индекс
  - Назначение первичного ключа
  - Составной первичный ключ
- Индексы пропуска данных
  - Управление индексами
  - Типы индекса
- Получение информации об индексах

Обзор

Индексирование — способ повышения производительности базы данных. Индексы представляют собой специальные структуры данных, обеспечивающие быстрый поиск запрашиваемых данных по значениям ключевого столбца (или набора столбцов) без выполнения полного сканирования таблицы.

ADQM поддерживает следующие типы индексов для таблиц MergeTree:

- Первичный индекс. Первичный индекс в ADQM отличается от индекса в традиционных реляционных базах данных и строится таким образом, чтобы аналитические запросы на больших объемах данных выполнялись наиболее эффективно — он хранит не единичные значения ключевых столбцов для каждой строки, а одну запись на диапазон из N упорядоченных по ключевым столбцам строк (каждая запись индекса — значения ключевых столбцов в первой строке соответствующего диапазона). Правильно настроенный первичный индекс таблицы существенно сокращает объем данных, которые необходимо считать с диска для выполнения запроса, и позволяет быстро находить диапазоны строк с нужными значениями. Ключ, по которому строится первичный индекс, указывается при создании таблицы и не может быть изменен после этого.
- Индексы пропуска данных (вторичные индексы). Когда первичного индекса недостаточно для обеспечения высокой производительности запросов, можно добавить в таблицу индексы пропуска данных — вторичные индексы в ADQM, позволяющие при выполнении запроса пропускать чтение блоков данных (могут быть достаточно большими), которые гарантированно не содержат запрашиваемых значений. Например, такие индексы могут быть эффективны для полнотекстового поиска или при проверках на вхождение значений в определенные списки. В отличие от первичного индекса индексы пропуска данных можно добавлять в существующие таблицы и удалять.

Каждый тип индекса ADQM имеет свои преимущества и особенности, поэтому при проектировании индексов таблицы важно учитывать какие запросы будут наиболее часто направляться в таблицу, характеристики используемых в запросах столбцов (тип данных, кардинальность), желаемый баланс между производительностью чтения/записи и потреблением ресурсов системы. Чтобы выбрать наиболее эффективный индекс, часто бывает необходимо провести тестирование нескольких вариантов с различными типами индексов, гранулярностью и другими параметрами.

> **РЕКОМЕНДАЦИЯ**
>
> В статье how-to:performance-tuning/improve-query-performance.adoc#indexes_example можно найти примеры использования в таблицах индексов различных типов и посмотреть, как они влияют на скорость выполнения запросов.

Разреженный первичный индекс

Данные таблицы MergeTree делятся на части — **куски данных (data parts)**, строки внутри которых сортируются по столбцам первичного ключа. Каждый кусок данных логически делится на **гранулы (granules)**. Гранула — минимальный неделимый набор данных, состоящий из целого количества строк, который считывается при выборке данных. Первая строка гранулы помечается значениями столбцов первичного ключа для этой строки — такая отметка называется **засечкой (mark)**. Для каждого куска данных создается свой индекс — файл с засечками. Таким образом, в ADQM не индексируется каждая отдельная строка, а первичный индекс куска данных имеет одну запись (засечку) на группу строк (гранулу), то есть используется **разреженный индекс (sparse index)**. Вместо непосредственного поиска отдельных строк по всей таблице разреженный первичный индекс позволяет быстро (с помощью бинарного поиска) идентифицировать гранулы, которые могут включать строки, соответствующие фильтру запроса. Затем найденные группы потенциально подходящих строк параллельно загружаются в память для поиска в них строк, точно соответствующих условию фильтрации.

Такой тип индексирования делает первичный индекс небольшим (он полностью помещается в оперативную память) и в то же время позволяет осуществлять поиск по большому количеству строк за считанные секунды, пропуская терабайты или даже петабайты неподходящих данных (хотя разреженный индекс допускает чтение лишних строк).

Назначение первичного ключа

Чтобы первичный индекс работал наиболее эффективно, первичный ключ должен включать столбец, по которому будет фильтроваться большинство запросов, обращающихся к таблице. Первичный ключ может содержать несколько столбцов — при определении составного первичного ключа обратите внимание на приведенные ниже рекомендации.

Чтобы назначить таблице MergeTree первичный ключ, используйте выражение `ORDER BY` или `PRIMARY KEY` в запросе `CREATE TABLE` при создании таблицы:

```
CREATE TABLE <table_name> (<column_name> <column_type>, ...)
ENGINE = MergeTree
ORDER BY <sorting_key_expr>
[PRIMARY KEY <primary_key_expr>];
```

- `ORDER BY` — ключ сортировки в виде одного или нескольких столбцов таблицы. ADQM использует ключ сортировки в качестве первичного ключа, если первичный ключ не задан с помощью выражения `PRIMARY KEY`.
- `PRIMARY KEY` — первичный ключ. По умолчанию первичный ключ совпадает с ключом сортировки `ORDER BY`, поэтому в большинстве случаев выражение `PRIMARY KEY` отдельно использовать не нужно. Но при необходимости можно указать первичный ключ (столбцы, значения которых будут записаны как засечки в индексный файл), отличный от ключа сортировки (столбцы, по которым будут упорядочены строки в кусках данных). В этом случае кортеж столбцов первичного ключа должен быть префиксом кортежа столбцов ключа сортировки (например, если первичный ключ — `(a, b)`, то ключ сортировки должен быть `(a, b, c…)`).

Для каждой таблицы может быть определен только один первичный ключ. Можно создать таблицу без первичного ключа — для этого используйте синтаксис `ORDER BY tuple()`.

Значения в столбцах первичного ключа не обязательно должны быть уникальными — ADQM допускает вставку в таблицу множества строк с одинаковыми значениями в ключевых столбцах.

Составной первичный ключ

В составном первичном ключе важен порядок столбцов, который может существенно влиять на:

- эффективность фильтрации по дополнительным ключевым столбцам в запросах;
- степень сжатия файлов данных таблицы.

Когда запрос фильтрует данные по первому ключевому столбцу, ADQM запускает алгоритм бинарного поиска по индексным засечкам этого столбца. Для фильтрации по другим ключевым столбцам применяется менее эффективный алгоритм generic exclusion search. Например, первичный ключ `(a, b)` будет полезен для ускорения запросов, фильтрующих данные по столбцу `a`, но не улучшит производительность запросов с фильтрацией по столбцу `b`, несмотря на то, что столбец `b` является частью составного первичного ключа. И наоборот, первичный индекс таблицы с ключом `(b, a)` будет ускорять запросы с фильтрацией по столбцу `b`, но не оптимизирует запросы, фильтрующие данные по столбцу `a`.

Если нужно ускорить оба типа запросов (с фильтрацией по первому столбцу и с фильтрацией по второму столбцу) и при этом кардинальность (количество различных значений) столбцов составного первичного ключа одинаково высокая, имеет смысл удалить второй ключевой столбец из первичного индекса, что приведет к меньшему потреблению памяти индексом, и вместо этого использовать несколько первичных индексов. Чтобы это реализовать, можно:

- создать вторую таблицу с другим первичным ключом;
- создать материализованное представление для существующей таблицы;
- добавить проекцию существующей таблицы.

Если ключевые столбцы составного первичного ключа имеют большие различия по кардинальности, то для ускорения запросов нужно упорядочить столбцы первичного ключа по кардинальности в порядке возрастания. Чем выше разница в кардинальности между ключевыми столбцами, тем большее значение имеет порядок этих столбцов в ключе.

Индексы пропуска данных

Для таблиц семейства MergeTree можно также указать **индексы пропуска данных (data skipping indexes)**. Индекс пропуска данных агрегирует некоторую информацию по заданному выражению (столбцу) в блоках данных. Затем эта информация используются при выполнении запросов `SELECT` для уменьшения объема считываемых с диска данных путем пропуска блоков данных, в которых гарантированно нет совпадений по условию фильтрации запроса. Пропускаемый блок состоит из гранул данных в количестве, равном гранулярности данного индекса.

Управление индексами

Индексы пропуска данных описываются с помощью выражения `INDEX` в секции описания столбцов запроса `CREATE TABLE`:

```
CREATE TABLE <table_name>
(   <column_name> <column_type>, ...
    INDEX <index_name> <index_expr> TYPE <index_type> [GRANULARITY <granularity_value>], ...)
ENGINE = MergeTree
...;
```

где:

- `<index_name>` — имя индекса;
- `<index_expr>` — выражение, по которому будет агрегироваться информация (часто это имя столбца);
- `<index_type>` — тип индекса, определяющий, какая дополнительная информация о содержимом блока данных будет храниться в индексе (эта информация позволит быстро находить нужные блоки и пропускать все остальные);
- `<granularity_value>` — гранулярность индекса, то есть количество гранул данных в пропускаемом блоке (значение по умолчанию — `1`).

В существующую таблицу индекс пропуска данных можно добавить с помощью следующего запроса:

```
ALTER TABLE <table_name> ADD INDEX <index_name> <index_expr> TYPE <index_type> [GRANULARITY <granularity_value>];
```

Когда индекс добавляется в существующую таблицу, он не обновляется автоматически. `ALTER TABLE` изменяет метаданные, поэтому индекс будет рассчитываться только для новых данных, вставляемых в таблицу. Чтобы применить индекс к существующим данным, выполните дополнительную команду:

```
ALTER TABLE <table_name> MATERIALIZE INDEX <index_name>;
```

Удалить индексы можно запросами:

- `ALTER TABLE <table_name> DROP INDEX <index_name>` — удаляет описание индекса из метаданных таблицы и удаляет индексные файлы с диска;
- `ALTER TABLE <table_name> CLEAR INDEX <index_name>` — удаляет файлы индекса с диска без удаления описания индекса из метаданных.

Типы индекса

MinMax

Индекс `minmax` хранит минимальное и максимальное значение столбца (или выражения) для каждого блока.

Set

Индекс `set(<max_rows>)` хранит уникальные значения столбца (или выражения) в блоке данных в количестве, не большем чем `<max_rows>` (`0` означает, что ограничений нет). Когда индексированный столбец используется в выражении `WHERE`, ADQM может считать небольшой набор значений вместо полного столбца. Этот тип индекса хорошо работает, если столбец имеет низкую кардинальность (небольшое количество уникальных значений) в каждом наборе гранул, но высокую кардинальность в целом.

Bloom Filter

Поддерживается три типа индекса на основе фильтра Блума:

- `bloom_filter([<false_positive>])` — фильтр Блума для столбца. Необязательный параметр `<false_positive>` определяет допустимую частоту получения ложноположительных ответов от фильтра. Возможны значения в диапазоне `(0, 1)`. Значение по умолчанию: `0.025`.
  Этот тип индекса поддерживается для следующих типов данных: Int*, UInt*, Float*, Enum, Date, DateTime, String, FixedString, Array, LowCardinality, Nullable, UUID и Map.
- `tokenbf_v1(<size_of_bloom_filter_in_bytes>, <number_of_hash_functions>, <random_seed>)` — входная строка разбивается на токены (последовательности символов, разделенные не буквенно-цифровыми символами), а затем токены сохраняются в фильтре Блума. Подходит для поиска точного совпадения в строке (например, когда нужно найти определенную часть URL-адреса или параметр запроса в столбце `URL`).
  Все три параметра индекса связаны с настройкой фильтра Блума:
  - `<size_of_bloom_filter_in_bytes>` — размер фильтра в байтах (фильтры большего размера имеют меньше ложноположительных срабатываний);
  - `<number_of_hash_functions>` — количество хеш-функций, использующихся в фильтре Блума;
  - `<random_seed>` — состояние генератора случайных чисел для хеш-функций фильтра Блума.

  Этот тип индекса работает только с данными типа String, FixedString или Map.
- `ngrambf_v1(<n>, <size_of_bloom_filter_in_bytes>, <number_of_hash_functions>, <random_seed>)` — входная строка разбивается на n-граммы (подстрока из n символов), а затем сохраняется в фильтре Блума. Подходит для полнотекстового поиска, особенно на языках без разрывов слов, таких как китайский. Первый параметр `<n>` задает размер n-грамм, остальные — аналогичны `tokenbf_v1`. Этот тип индекса работает только с типами данных String, FixedString и Map.

Inverted

Инвертированный индекс `inverted([<n>], [<max_rows_per_posting_list>])` хранит сопоставление уникальных слов (или n-грамм) текстового столбца с указателями на их расположение в таблице (номера строк, в которых они содержатся). Используется для оптимизации полнотекстового поиска по столбцам типов String, FixedString, Array(String), Array(FixedString), Map(String) и Map(String). В настоящее время работает в экспериментальном режиме. Более подробную информацию и пример использования можно получить в статье Полнотекстовый поиск.

Получение информации об индексах

Информацию о существующих индексах можно получить из системных таблиц:

- `system.tables` — поле `primary_key` содержит информацию о первичном ключе таблицы.

  Пример

  ```
  SELECT primary_key FROM system.tables WHERE table='table_compound_primary_key';
  ```

  ```
  ┌─primary_key─┐
  │ a, b        │
  └─────────────┘
  ```

- `system.data_skipping_indices` — содержит информацию об индексах пропуска данных во всех таблицах.

  Пример

  ```
  SELECT type_full,expr FROM system.data_skipping_indices WHERE table='table_skip_index';
  ```

  ```
  ┌─type_full─┬─expr─┐
  │ set(100)  │ b    │
  └───────────┴──────┘
  ```

Также можно использовать запросы:

- `SHOW CREATE TABLE <table_name>`.

  Пример

  ```
  SHOW CREATE TABLE table_skip_index;
  ```

  ```
  CREATE TABLE default.table_skip_index
  (
      `a` UInt64,
      `b` UInt64,
      INDEX my_index b TYPE set(100) GRANULARITY 2
  )
  ENGINE = MergeTree
  PRIMARY KEY a
  ORDER BY a
  SETTINGS index_granularity = 10000
  ```

- `SHOW {INDEX|INDEXES|INDICES|KEYS} FROM <table_name>`.

  Пример

  ```
  SHOW INDEXES FROM table_skip_index;
  ```

  ```
  ┌─table────────────┬─non_unique─┬─key_name─┬─seq_in_index─┬─column_name─┬
  ┬─index_type─┬─comment─┬─index_comment─┬─visible─┬─expression─┐
  │ table_skip_index │          1 │ PRIMARY  │            1 │ a           │ ... │ PRIMARY    │         │               │ YES
  │            │
  └──────────────────┴────────────┴──────────┴──────────────┴─────────────┴─
  ─┴────────────┴─────────┴───────────────┴─────────┴────────────┘
  ┌─table────────────┬─non_unique─┬─key_name─┬─seq_in_index─┬─column_name─┬─
  ┬─index_type─┬─comment─┬─index_comment─┬─visible─┬─expression─┐
  │ table_skip_index │          1 │ my_index │            1 │             │ ... │ SET        │         │               │ YES
  │ b          │
  └──────────────────┴────────────┴──────────┴──────────────┴─────────────┴─
  ┴────────────┴─────────┴───────────────┴─────────┴────────────┘
  ```

Чтобы убедиться в использовании индексов для конкретного запроса, можно проанализировать выполнение этого запроса используя команду EXPLAIN с параметром `indexes`.

Пример

```
EXPLAIN indexes = 1 SELECT * FROM table_skip_index WHERE b=100 OR b=555;
```

```
┌─explain─────────────────────────────────────────┐
│ Expression ((Project names + Projection))       │
│   Expression                                    │
│     ReadFromMergeTree (default.table_skip_index) │
│     Indexes:                                    │
│       PrimaryKey                                │
│         Condition: true                         │
│         Parts: 6/6                              │
│         Granules: 10004/10004                   │
│       Skip                                      │
│         Name: my_index                          │
│         Description: set GRANULARITY 2          │
│         Parts: 1/6                              │
│         Granules: 4/10004                       │
└─────────────────────────────────────────────────┘
```

Словари

Елена Дворядкина

Содержание

Назад к содержанию

- Создание и конфигурирование словаря
  - Ключ и атрибуты словаря
  - Источник данных для словаря
  - Размещение словаря в памяти
  - Обновление словаря
  - Пример
- Просмотр информации о словарях
- Загрузка словарей
- Использование словаря
  - dictGet, dictGetOrDefault, dictGetOrNull
  - dictHas
  - Пример
- Отключение и подключение словаря
- Удаление словаря

Словарь — хранилище данных типа ключ/значение, которое полностью или частично хранится в оперативной памяти сервера ADQM и может быть использовано в качестве справочника для подстановки значений по ключам в итоговую выборку данных. Словари являются более простой в использовании альтернативой оператору JOIN.

Источником данных для словаря может быть локальный файл (текстовый или исполняемый), HTTP(s) ресурс или другая СУБД. ADQM автоматически обновляет словари и подгружает отсутствующие или измененные значения из источника. Доступ к словарям из ADQM осуществляется только на чтение.

Настроить ADQM для подключения словарей из внешних источников можно с помощью конфигурационных XML-файлов или DDL-запросов. В этой статье описывается второй подход (рекомендуемый) — приводится базовый синтаксис и примеры запросов для создания/конфигурирования/удаления словарей в ADQM, а также для чтения данных из словарей с помощью специальных функций.

Создание и конфигурирование словаря

Чтобы создать и настроить словарь в ADQM, используйте запрос `CREATE DICTIONARY`. Общий синтаксис:

```
CREATE DICTIONARY [OR REPLACE][IF NOT EXISTS] [<database_name>.]<dictionary_name> [ON CLUSTER <cluster_name>]
(   <key1_name> <key1_data_type>,
    <key2_name> <key2_data_type>,
    <attr1_name> <attr1_data_type> [DEFAULT <default_value1>] [EXPRESSION <expr1>] [HIERARCHICAL] [INJECTIVE] [IS_OBJECT_ID],
    <attr2_name> <attr2_data_type> [DEFAULT <default_value2>] [EXPRESSION <expr2>] [HIERARCHICAL] [INJECTIVE] [IS_OBJECT_ID])
PRIMARY KEY <key1_name>, <key2_name>
SOURCE(<SOURCE_TYPE>([<param1_name> <param1_value> ... <paramN_name> <paramN_value>]))
LAYOUT(<LAYOUT_NAME>([<param_name> <param_value>]))
LIFETIME({MIN <min_value> MAX <max_value> | <max_value>})
[SETTINGS(<setting_name> = <setting_value>, <setting_name> = <setting_value>, ...)]
[COMMENT '<comment_text>']
```

где:

- `PRIMARY KEY` — ключ словаря;
- `SOURCE` — источник данных для словаря;
- `LAYOUT` — способ хранения словаря в памяти;
- `LIFETIME` — частота обновления словаря;
- `SETTINGS` — дополнительные настройки, которые можно указать для словаря, источником данных которого является локальный файл, ресурс HTTP(s) или таблица ClickHouse;
- `COMMENT` — комментарий к словарю.

Ключ и атрибуты словаря

Структуру словаря описывают ключ и атрибуты, которые указываются как параметры в запросе `CREATE DICTIONARY`.

- **Ключ** — столбец в источнике словаря с уникальными значениями, которые используются для идентификации записей и поиска данных.

  ADQM поддерживает следующие виды ключей:

  - Числовой ключ — столбец типа UInt64.

    ```
    CREATE DICTIONARY <dictionary_name> (
        <key_name> UInt64,
        ...
        )
    PRIMARY KEY <key_name>
    ...
    ```

    где `<key_name>` — имя столбца с ключами.

  - Составной ключ — кортеж из столбцов произвольных типов. Может состоять из одного элемента (например, можно использовать строку в качестве ключа).

    ```
    CREATE DICTIONARY <dictionary_name> (
        <key1_name> <key1_data_type>,
        <key2_name> <key2_data_type>
        ...
        )
    PRIMARY KEY <key1_name>, <key2_name>
    ...
    ```

    где `<key1_name>`, `<key2_name>` — имена столбцов с ключами, `<key1_data_type>`, `<key2_data_type>` — типы ключей.

- **Атрибуты** — столбцы в источнике словаря, значения которых можно получить по ключу. Атрибуты описываются следующим образом:

```
CREATE DICTIONARY <dictionary_name> (
    ...
    <attribute_name> <attribute_data_type> [DEFAULT <default_value>] [EXPRESSION <expr>] [HIERARCHICAL] [INJECTIVE] [IS_OBJECT_ID]
    )
...
```

Параметры и основные выражения

<attribute_name>	Имя столбца в источнике данных
<attribute_data_type>	Тип данных, к которому будет приводиться значение из таблицы источника словаря. Поддерживаются типы: UInt8, UInt16, UInt32, UInt64, Int8, Int16, Int32, Int64, Float32, Float64, UUID, Decimal32, Decimal64, Decimal128, Decimal256, String, Array. Возможность использовать типы Nullable зависит от способа размещения словаря в памяти (например, для атрибутов словарей `IP_TRIE` в настоящее время типы Nullable не поддерживаются)
DEFAULT	Значение по умолчанию для элемента, не найденного по ключу. Значение NULL можно указывать, только если тип атрибута Nullable
EXPRESSION	Выражение, которое выполняется для значения
HIERARCHICAL	Указывает, что атрибут содержит значение родительского ключа для текущего ключа (используется для конфигурирования иерархического словаря)
INJECTIVE	Указывает, что атрибут является инъективным, то есть разным ключам соответствуют обязательно разные значения атрибута. В этом случае если `GROUP BY` использует функцию, которая извлекает значение атрибута по ключу, эта функция автоматически удаляется из `GROUP BY`, что оптимизирует процесс агрегирования данных
IS_OBJECT_ID	Указывает, что запрос выполняется для документа MongoDB по `ObjectID`

Источник данных для словаря

Источник данных для словаря в запросе `CREATE DICTIONARY` указывается с помощью выражения `SOURCE`:

```
CREATE DICTIONARY <dictionary_name> (...)
...
SOURCE(<SOURCE_TYPE>([<param1_name> <param1_value> ... <paramN_name> <paramN_value>]))
...
```

где `<SOURCE_TYPE>` — тип источника данных, `<paramX_name> <paramX_value>` — настройки источника.

Возможные значения <SOURCE_TYPE>

Значение	Источник данных
FILE	Локальный файл (файл должен быть расположен в папке *user_files*)
HTTP	HTTP-ресурс
MYSQL	Таблица из базы данных MySQL
CLICKHOUSE	Таблица из базы данных ClickHouse
MONGODB	Коллекция из базы данных MongoDB
REDIS	База данных Redis
CASSANDRA	Семейство столбцов (таблица) из пространства ключей (базы данных) Cassandra
POSTGRESQL	Таблица из базы данных PostgreSQL (см. пример в статье Интеграция ADQM и ADPG)
ODBC	Таблица из любой базы данных, которая имеет драйвер ODBC

Примеры подключения словарей к различным типам источников приведены в разделе Dictionary Sources документации ClickHouse.

Размещение словаря в памяти

Способ хранения словаря в памяти указывается с помощью выражения `LAYOUT` в запросе `CREATE DICTIONARY`:

```
CREATE DICTIONARY <dictionary_name> (...)
...
LAYOUT(<LAYOUT_TYPE>([<param_name> <param_value>]))
...
```

где `<LAYOUT_TYPE>` — способ хранения словаря в памяти.

Возможные значения <LAYOUT_TYPE>

Значение	Способ хранения словаря
FLAT	Словарь полностью хранится в оперативной памяти в виде плоских массивов
HASHED, COMPLEX_KEY_HASHED	Словарь полностью хранится в оперативной памяти в виде хеш-таблиц
SPARSE_HASHED, COMPLEX_KEY_SPARSE_HASHED	Словарь полностью хранится в оперативной памяти в виде хеш-таблиц аналогично `HASHED` / `COMPLEX_KEY_HASHED`, но занимает меньше места в памяти и генерирует более высокую загрузку CPU
HASHED_ARRAY, COMPLEX_KEY_HASHED_ARRAY	Словарь полностью хранится в оперативной памяти. Каждый атрибут хранится в массиве. Ключ хранится в виде хеш-таблицы, где его значение является индексом в массиве атрибутов
RANGE_HASHED, COMPLEX_KEY_RANGE_HASHED	Словарь хранится в оперативной памяти в виде хеш-таблицы с упорядоченным массивом диапазонов и соответствующих значений
CACHE, COMPLEX_KEY_CACHE	Словарь хранится в кеше, состоящем из фиксированного количества ячеек, которые содержат часто используемые элементы
SSD_CACHE, COMPLEX_KEY_SSD_CACHE	Словарь хранится в кеше аналогично `SSD_CACHE` / `COMPLEX_KEY_SSD_CACHE`, но данные хранятся на SSD, а индекс в оперативной памяти
DIRECT, COMPLEX_KEY_DIRECT	Словарь не хранит данные локально и взаимодействует с источником непосредственно в момент запроса
IP_TRIE	Тип размещения словаря для получения метаданных (например, номер `AS` или код страны) по IP-адресам. У словаря должен быть составной ключ, который содержит один элемент типа String — IP-адрес

Значения с префиксом `COMPLEX_KEY` используются для словарей с составным ключом.

Особенности всех вариантов размещения словарей в памяти описаны в разделе Storing Dictionaries in Memory документации ClickHouse.

Обновление словаря

С помощью выражения `LIFETIME` в запросе `CREATE DICTIONARY` можно указать интервал автоматического обновления словаря в секундах.

```
CREATE DICTIONARY <dictionary_name> (...)
...
LIFETIME(<max_value> | MIN <min_value> MAX <max_value>)
...
```

где:

- `<max_value>` — интервал обновления словаря. При `LIFETIME(0)` обновление словаря отключено.
- `MIN <min_value> MAX <max_value>` — интервал, внутри которого случайным образом выбирается время для обновления словаря. Такая настройка может быть полезна для распределения нагрузки на источник данных при обновлении словаря на большом количестве серверов. При `MIN 0 MAX 0` словарь не перезагружается по истечении времени. В этом случае данные словаря могут быть перезагружены командой SYSTEM RELOAD DICTIONARY.

Обновление словаря (кроме загрузки при первом использовании) не блокирует запросы — во время обновления используется старая версия словаря. Если при обновлении возникнет ошибка, то ошибка запишется в лог сервера, а запросы продолжат использовать старую версию словаря.

Логика обновления словаря зависит от типа источника и способа хранения словаря в памяти — подробнее об этом в разделе Dictionary Updates документации ClickHouse.

Пример

1. Создайте таблицу `roles_dict`, которая будет источником данных для словаря:

```
CREATE TABLE roles_dict (`id` Int32, `role` String) ENGINE = MergeTree() ORDER BY id;
```

```
INSERT INTO roles_dict VALUES (10, 'admin'), (20, 'owner'), (30, 'author'), (40, 'reviewer'), (50, 'editor'), (60, 'view only');
```

```
--id-┬-role-------
│ 10 │ admin     │
│ 20 │ owner     │
│ 30 │ author    │
│ 40 │ reviewer  │
│ 50 │ editor    │
│ 60 │ view only │
------------------
```

2. Создайте на кластере словарь, который будет считывать данные из таблицы `roles_dict`:

```
CREATE DICTIONARY roles_dict_clickhouse (`id` Int32, `role` String DEFAULT 'no role assigned')
PRIMARY KEY id
SOURCE(CLICKHOUSE(HOST 'localhost' USER 'default' PASSWORD '' DATABASE 'default' TABLE 'roles_dict'))
LIFETIME(MIN 10 MAX 20)
LAYOUT(FLAT());
```

Просмотр информации о словарях

Информацию о словарях, сконфигурированных на сервере, можно получить из системной таблицы system.dictionaries:

```
SELECT * FROM system.dictionaries WHERE name = '<dictionary_name>' [FORMAT <format_name>];
```

Пример

Запрос:

```
SELECT * FROM system.dictionaries WHERE name = 'roles_dict_clickhouse' FORMAT Vertical;
```

Пример вывода на экран (словарь сконфигурирован, но не загружен):

```
Row 1:
──────
database:                           default
name:                               roles_dict_clickhouse
uuid:                               4e28febc-b472-423c-bc14-bd29f195390f
status:                             NOT_LOADED
origin:                             4e28febc-b472-423c-bc14-bd29f195390f
type:
key.names:                          ['id']
key.types:                          ['UInt64']
attribute.names:                    ['role']
attribute.types:                    ['String']
bytes_allocated:                    0
hierarchical_index_bytes_allocated: 0
query_count:                        0
hit_rate:                           0
found_rate:                         0
element_count:                      0
load_factor:                        0
source:
lifetime_min:                       0
lifetime_max:                       0
loading_start_time:                 1970-01-01 00:00:00
last_successful_update_time:        1970-01-01 00:00:00
loading_duration:                   0
last_exception:
comment:
```

Чтобы получить имена всех словарей, существующих в базе данных, можно использовать запрос SHOW DICTIONARIES:

```
SHOW DICTIONARIES FROM <database_name>;
```

Загрузка словарей

Настройка сервера dictionaries_lazy_load определяет, когда словари должны загружаться — автоматически при запуске сервера или при первом использовании словаря.

Принудительную перезагрузку всех словарей сразу или отдельного словаря можно выполнить с помощью запросов:

```
SYSTEM RELOAD DICTIONARIES;
```

```
SYSTEM RELOAD DICTIONARY <dictionary_name>;
```

После загрузки словаря в таблице system.dictionaries меняется статус словаря на `LOADED`, а также записываются значения конфигурационных параметров словаря и различные метрики (такие как количество занятой словарем RAM или количество запросов к словарю с момента его успешной загрузки). Например:

```
SELECT * FROM system.dictionaries WHERE name = 'roles_dict_clickhouse' FORMAT Vertical;
```

```
Row 1:
──────
database:                           default
name:                               roles_dict_clickhouse
uuid:                               4e28febc-b472-423c-bc14-bd29f195390f
status:                             LOADED
origin:                             4e28febc-b472-423c-bc14-bd29f195390f
type:                               Flat
key.names:                          ['id']
key.types:                          ['UInt64']
attribute.names:                    ['role']
attribute.types:                    ['String']
bytes_allocated:                    21048
hierarchical_index_bytes_allocated: 0
query_count:                        0
hit_rate:                           1
found_rate:                         0
element_count:                      6
load_factor:                        0.005859375
source:                             ClickHouse: default.roles_dict
lifetime_min:                       10
lifetime_max:                       20
loading_start_time:                 2023-03-14 11:05:13
last_successful_update_time:        2023-03-14 11:05:13
loading_duration:                   0.001
last_exception:
comment:
```

Использование словаря

Для работы со словарями ADQM поддерживает набор специальных функций.

dictGet, dictGetOrDefault, dictGetOrNull

Функции `dictGet`, `dictGetOrDefault` и `dictGetOrNull` извлекают значения из словаря.

```
dictGet('<dictionary_name>', <attr_name>, <id_expr>)
dictGetOrDefault('<dictionary_name>', <attr_name>, <id_expr>, <default_value_expr>)
dictGetOrNull('<dictionary_name>', <attr_name>, <id_expr>)
```

Аргументы

<dictionary_name>	Имя словаря. Указывается как строковый литерал
<attr_name>	Имя столбца в источнике словаря (строковый литерал) или кортеж имен столбцов (tuple)
<id_expr>	Значение ключа словаря. Указывается как выражение, возвращающее значение типа UInt64, или кортеж выражений (tuple), в зависимости от того, числовой или составной ключ определен в конфигурации словаря
<default_value_expr>	Значение, возвращаемое, если в словаре не найдена строка с заданным ключом `<id_expr>`. Указывается как выражение, возвращающее значение типа данных, сконфигурированного для атрибута `<attr_name>`, или кортеж выражений

Возвращаемое значение:

- значение атрибута, соответствующее ключу `<id_expr>`, если ADQM смог привести это значение к типу данных, указанному для атрибута в конфигурации словаря;
- если ключ `<id_expr>` в словаре не найден:
  - `dictGet` возвращает значение, указанное через выражение `DEFAULT` для атрибута при конфигурировании словаря;
  - `dictGetOrDefault` возвращает значение `<default_value_expr>`;
  - `dictGetOrNull` возвращает NULL.

Например, следующий запрос возвращает из словаря `roles_dict_clickhouse` роль с идентификатором `20`:

```
SELECT dictGet('roles_dict_clickhouse', 'role', 20);
```

```
┌─dictGet('roles_dict_clickhouse', 'role', 20)─┐
│ owner                                        │
└──────────────────────────────────────────────┘
```

ADQM поддерживает также специализированные функции, которые приводят значения атрибутов словаря к определенному типу данных независимо от конфигурации словаря:

- `dictGetInt8`, `dictGetInt16`, `dictGetInt32`, `dictGetInt64`;
- `dictGetUInt8`, `dictGetUInt16`, `dictGetUInt32`, `dictGetUInt64`;
- `dictGetFloat32`, `dictGetFloat64`;
- `dictGetDate`;
- `dictGetDateTime`;
- `dictGetUUID`;
- `dictGetString`.

Синтаксис этих функций аналогичен синтаксису `dictGet`, который приведен выше. Любую из этих функций можно использовать с суффиксом `OrDefault` (например, `dictGetInt8OrDefault`) и указать в аргументах значение `<default_value_expr>`, которое будет возвращаться, если ключ в словаре не найден.

dictHas

Функция `dictHas` проверяет, присутствует ли запись с указанным ключом в словаре.

```
dictHas('<dictionary_name>', <id_expr>)
```

Аргументы

<dictionary_name>	Имя словаря. Указывается как строковый литерал
<id_expr>	Значение ключа словаря. Указывается как выражение, возвращающее значение типа UInt64, или кортеж выражений (tuple), в зависимости от того, числовой или составной ключ определен в конфигурации словаря

Функция возвращает:

- `0` — если ключ в словаре не найден;
- `1` — если ключ есть.

Например, следующий запрос проверяет, есть ли в словаре `roles_dict_clickhouse` роль с идентификатором `100`:

```
SELECT dictHas('roles_dict_clickhouse', 100);
```

```
┌─dictHas('roles_dict_clickhouse', 100)─┐
│                                     0 │
└───────────────────────────────────────┘
```

Пример

1. Создайте таблицу `users`, которая содержит имена пользователей и идентификаторы назначенных пользователям ролей:

```
CREATE TABLE users (user_id Int32, name String, role_id Int32) ENGINE = MergeTree ORDER BY user_id;
```

```
INSERT INTO users VALUES (1, 'john', 10), (2, 'mary', 30), (3, 'andrew', 40), (4, 'harry', 70), (5, 'ann', 50);
```

```
--user_id---name-----role_id--
│       1 │ john   │      10 │
│       2 │ mary   │      30 │
│       3 │ andrew │      40 │
│       4 │ harry  │      70 │
│       5 │ ann    │      50 │
------------------------------
```

2. Выполните следующий запрос, чтобы считать имена пользователей из таблицы `users` и найти названия их ролей в словаре `roles_dict_clickhouse` по идентификаторам `role_id`:

```
SELECT  name AS user, dictGet('roles_dict_clickhouse', 'role', toUInt64(role_id)) AS role FROM users;
```

Результат:

```
--user-----role--------------
│ john   │ admin            │
│ mary   │ author           │
│ andrew │ reviewer         │
│ harry  │ no role assigned │
│ ann    │ editor           │
-----------------------------
```

Отключение и подключение словаря

Словарь можно отключить с помощью запроса `DETACH DICTIONARY`:

```
DETACH DICTIONARY [IF EXISTS] [<database_name.]<dictionary_name> [ON CLUSTER <cluster_name>] [PERMANENTLY];
```

При выполнении этого запроса словарь становится "невидимым" для сервера, но его метаданные не удаляются. Если отключение не было перманентным (ключевое слово `PERMANENTLY` не использовалось в запросе), при следующем запуске сервер прочитает метаданные и снова будет "видеть" словарь. Если словарь был отключен перманентно, то сервер не подключит его обратно автоматически.

Независимо от того, каким способом словарь был отключен, его можно подключить обратно с помощью запроса `ATTACH`:

```
ATTACH DICTIONARY  [IF NOT EXISTS] [<database_name.]<dictionary_name> [ON CLUSTER <cluster_name>];
```

Удаление словаря

Удалить словарь можно с помощью запроса `DROP DICTIONARY`:

```
DROP DICTIONARY [IF EXISTS] [<database_name>.]<dictionary_name> [ON CLUSTER <cluster_name>];
```

Материализованные представления

Елена Дворядкина

Содержание

Назад к содержанию

- Создание
  - Обзор синтаксиса
  - Пример
- Изменение
  - Пример

Материализованное представление (materialized view) вычисляет промежуточные агрегатные состояния для данных другой таблицы (source table) согласно запросу `SELECT`, который указывается в описании представления, и сохраняет результаты в свою внутреннюю или отдельную таблицу (target table). Обновление агрегатных состояний происходит автоматически при каждой вставке новых данных в исходную таблицу. Использование материализованных представлений позволяет оптимизировать время выполнения сложных запросов (с объединениями и агрегатами), обращающихся к большому количеству записей — за счет предварительного вычисления итоговых значений сокращается объем данных, которые необходимо обработать для выполнения запросов `SELECT`.

Создание

Обзор синтаксиса

Чтобы создать материализованное представление, используйте запрос `CREATE MATERIALIZED VIEW`. Общий синтаксис:

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [<db_name>.]<mat_view_name> [ON CLUSTER <cluster_name>]
[TO [<db_name>.]<target_table>]
[ENGINE = <engine>]
[POPULATE]
AS SELECT
    ...
FROM [<db_name>.]<source_table>
...;
```

Для описания материализованного представления используются следующие выражения.

TO	Целевая таблица (target table), в которую будут сохраняться данные из материализованного представления
ENGINE	Движок внутренней таблицы материализованного представления, где будут храниться агрегированные данные, если отдельная целевая таблица не была заранее создана и явно указана через выражение `TO` в описании материализованного представления
POPULATE	Указывает, что при создании материализованного представления в него будут добавлены данные, уже содержащиеся в исходной таблице. Использовать `POPULATE` не рекомендуется, так как в представление не попадут данные, добавляемые в исходную таблицу во время создания представления. Кроме того, ключевое слово `POPULATE` не указывается, если описание материализованного представления включает выражение `TO` — в этом случае представление будет обрабатывать и сохранять только данные, добавляемые в исходную таблицу после создания представления
AS SELECT … FROM …	Запрос, который считывает новые данные, вставляемые в исходную таблицу, и определяет, как эти данные преобразовывать для загрузки в материализованное представление (или целевую таблицу, если она указана). Запрос `SELECT` может содержать выражения `DISTINCT`, `GROUP BY`, `ORDER BY`, `LIMIT` — соответствующие преобразования будут выполняться независимо для каждого блока вставляемых данных. Например, при наличии `GROUP BY` данные будут агрегироваться при вставке в рамках только одного блока вставляемых данных. Далее данные не будут доагрегированы. Исключением является использование табличного движка, указанного в описании представления через `ENGINE`, который выполняет агрегацию данных самостоятельно (например, `SummingMergeTree`). Изменить запрос `SELECT` после создания материализованного представления можно с помощью запроса ALTER TABLE … MODIFY QUERY — но учитывайте, что в настоящее время этот тип запроса является экспериментальным и имеет некоторые ограничения

ПРИМЕЧАНИЕ

Изменения существующих данных в исходной таблице (например, обновление, удаление и т.д.) не влияют на материализованное представление.

В ADQM материализованные представления выглядят так же, как обычные таблицы. Например, они включаются в результат запроса `SHOW TABLES`.

Чтобы удалить материализованное представление, выполните запрос `DROP VIEW` или `DROP TABLE`.

Пример

В примере ниже показано, как создать материализованное представление для предварительной агрегации данных, чтобы ускорить последующие выполнения запросов `SELECT`.

1. Создайте таблицу, которая будет хранить исходные данные о просмотрах страниц:

```
CREATE TABLE page_views
(
    time DateTime,
    page_name String,
    time_on_page UInt64
)
ENGINE = MergeTree
ORDER BY time;
```

2. Создайте целевую таблицу, в которую будут записываться количество просмотров и среднее время просмотра страниц по месяцам:

```
CREATE TABLE page_views_monthly
(
    page_name String,
    month Date,
    page_view_count UInt64,
    avg_time_on_page AggregateFunction(avg, UInt64)
)
ENGINE = SummingMergeTree()
ORDER BY (page_name, month);
```

Для колонки со средним значением времени просмотра страниц используется тип данных AggregateFunction, который позволяет хранить частично агрегированные данные. Табличный движок SummingMergeTree, который используется в примере, требует этот тип данных для всех агрегатов, кроме `sum` и `count`. Для работы с материализованными представлениями можно использовать и другие движки таблиц. Например, движок AggregatingMergeTree — этот движок работает только с агрегатными функциями (то есть если бы в запросе выше использовался движок AggregatingMergeTree, для колонки `page_view_count` также нужно было бы использовать тип AggregateFunction).

3. Создайте материализованное представление:

```
CREATE MATERIALIZED VIEW page_views_monthly_mv
TO page_views_monthly
AS SELECT
    toDate(toStartOfMonth(time)) AS month,
    page_name,
    count() as page_view_count,
    avgState(time_on_page) AS avg_time_on_page
FROM page_views
GROUP BY page_name, month
ORDER BY page_name, month;
```

При вставке данных в исходную таблицу `page_views` материализованное представление преобразует вставляемые данные, как указано в запросе `SELECT`, и поместит результат в целевую таблицу `page_views_monthly`. Для сохранения промежуточных агрегатных состояний в столбец типа AggregateFunction (`avg_time_on_page`) в запросе `SELECT` должна использоваться соответствующая агрегатная функция с суффиксом `State` (`avgState`).

Если на этом этапе выполнить запрос `SHOW TABLES`, вывод на экран будет включать созданные таблицы и материализованное представление:

```
┌─name──────────────────┐
│ page_views            │
│ page_views_monthly    │
│ page_views_monthly_mv │
└───────────────────────┘
```

4. Заполните исходную таблицу тестовыми данными. Например, сгенерируйте случайным образом 1 миллион просмотров для 4 страниц:

```
INSERT INTO page_views SELECT
    toDate('2023-01-01 00:00:00') + (rand() % 182) AS time,
    ['landing', 'products', 'docs', 'about'][toInt32(randBinomial(4, 0.01)) + 1],
    rand() % 15
FROM numbers(1000000);
```

5. Сравните скорость выполнения запроса `SELECT` при выборке данных из исходной таблицы с необработанными входящими данными и таблицы с предварительно агрегированными данными по месяцам.

Сделайте выборку данных из таблицы `page_views`:

```
SELECT
    page_name,
    toDate(toStartOfMonth(time)) AS month,
    count() AS page_view_count,
    round(avg(time_on_page), 2) AS avg_time_on_page
FROM page_views
GROUP BY page_name, month
ORDER BY page_name, month;
```

```
┌─page_name─┬──────month─┬─page_view_count─┬─avg_time_on_page─┐
│ about     │ 2023-02-01 │               2 │                7 │
│ about     │ 2023-03-01 │               1 │                7 │
│ docs      │ 2023-01-01 │              95 │             7.21 │
│ docs      │ 2023-02-01 │              93 │             6.42 │
│ docs      │ 2023-03-01 │             102 │             6.72 │
│ docs      │ 2023-04-01 │              84 │             7.35 │
│ docs      │ 2023-05-01 │             101 │             7.18 │
│ docs      │ 2023-06-01 │              92 │             7.72 │
│ docs      │ 2023-07-01 │               5 │              8.6 │
│ landing   │ 2023-01-01 │          163242 │             7.01 │
│ landing   │ 2023-02-01 │          147695 │                7 │
│ landing   │ 2023-03-01 │          163930 │             6.99 │
│ landing   │ 2023-04-01 │          158155 │             7.01 │
│ landing   │ 2023-05-01 │          163263 │             6.99 │
│ landing   │ 2023-06-01 │          158840 │                7 │
│ landing   │ 2023-07-01 │            5396 │             7.02 │
│ products  │ 2023-01-01 │            6667 │             7.05 │
│ products  │ 2023-02-01 │            6099 │             6.99 │
│ products  │ 2023-03-01 │            6589 │             7.02 │
│ products  │ 2023-04-01 │            6391 │             7.01 │
│ products  │ 2023-05-01 │            6428 │             6.98 │
│ products  │ 2023-06-01 │            6517 │             7.04 │
│ products  │ 2023-07-01 │             213 │             7.48 │
└───────────┴────────────┴─────────────────┴──────────────────┘

23 rows in set. Elapsed: 0.023 sec. Processed 1.00 million rows, 28.04 MB (43.43 million rows/s., 1.22 GB/s.)
```

Сделайте выборку из таблицы `page_views_monthly` — чтение данных из этой таблицы выполнится в 7 раз быстрее, чем из исходной таблицы (на большем объеме данных разница будет еще более заметна). Обратите внимание, в запросе `SELECT` используется та же агрегатная функция, что и при вставке данных, но с суффиксом `Merge` — это необходимо, чтобы объединить множество промежуточных агрегатных состояний и вернуть полную агрегацию данных (см. AggregateFunction):

```
SELECT
    page_name,
    month,
    sum(page_view_count) AS page_view_count,
    round(avgMerge(avg_time_on_page), 2) AS avg_time_on_page
FROM page_views_monthly
GROUP BY page_name, month
ORDER BY page_name, month;
```

```
┌─page_name─┬──────month─┬─page_view_count─┬─avg_time_on_page─┐
│ about     │ 2023-02-01 │               2 │                7 │
│ about     │ 2023-03-01 │               1 │                7 │
│ docs      │ 2023-01-01 │              95 │             7.21 │
│ docs      │ 2023-02-01 │              93 │             6.42 │
│ docs      │ 2023-03-01 │             102 │             6.72 │
│ docs      │ 2023-04-01 │              84 │             7.35 │
│ docs      │ 2023-05-01 │             101 │             7.18 │
│ docs      │ 2023-06-01 │              92 │             7.72 │
│ docs      │ 2023-07-01 │               5 │              8.6 │
│ landing   │ 2023-01-01 │          163242 │             7.01 │
│ landing   │ 2023-02-01 │          147695 │                7 │
│ landing   │ 2023-03-01 │          163930 │             6.99 │
│ landing   │ 2023-04-01 │          158155 │             7.01 │
│ landing   │ 2023-05-01 │          163263 │             6.99 │
│ landing   │ 2023-06-01 │          158840 │                7 │
│ landing   │ 2023-07-01 │            5396 │             7.02 │
│ products  │ 2023-01-01 │            6667 │             7.05 │
│ products  │ 2023-02-01 │            6099 │             6.99 │
│ products  │ 2023-03-01 │            6589 │             7.02 │
│ products  │ 2023-04-01 │            6391 │             7.01 │
│ products  │ 2023-05-01 │            6428 │             6.98 │
│ products  │ 2023-06-01 │            6517 │             7.04 │
│ products  │ 2023-07-01 │             213 │             7.48 │
└───────────┴────────────┴─────────────────┴──────────────────┘

23 rows in set. Elapsed: 0.003 sec.
```

Изменение

Запрос `SELECT`, который был указан при создании материализованного представления, можно изменить с помощью выражения `ALTER TABLE…MODIFY QUERY`. Эта команда не меняет структуру и описания столбцов внутренней таблицы материализованного представления, поэтому не рекомендуется ее использование для изменения материализованных представлений, созданных без указания целевой таблицы через выражение `TO`.

Перед выполнением команды `ALTER TABLE…MODIFY QUERY`, активируйте опцию: `SET allow_experimental_alter_materialized_view_structure=1`.

Пример

1. С помощью выражения ALTER TABLE…ADD COLUMN добавьте столбец `browser` в исходную таблицу `page_views` и целевую таблицу `page_views_monthly` (столбцы материализованного представления и целевой таблицы не обязательно должны совпадать, поэтому изменение структуры целевой таблицы не нарушит вставку данных):

```
ALTER TABLE page_views ADD COLUMN browser String;
```

```
ALTER TABLE page_views_monthly ADD COLUMN browser String after month,
MODIFY ORDER BY (page_name, month, browser);
```

2. Вставьте данные в исходную таблицу:

```
INSERT INTO page_views SELECT
    toDate('2023-01-01 00:00:00') + (rand() % 182) AS time,
    ['landing', 'products', 'docs', 'about'][toInt32(randBinomial(4, 0.01)) + 1],
    rand() % 15,
    ['firefox', 'safary', 'chrome'][(number % 3) + 1]
FROM numbers(1000000);
```

Если на этом этапе сделать выборку данных из целевой таблицы, столбец `browser` будет пустым:

```
SELECT
    page_name,
    month,
    browser,
    sum(page_view_count) AS page_view_count,
    round(avgMerge(avg_time_on_page), 2) AS avg_time_on_page
FROM page_views_monthly
GROUP BY page_name, month, browser
ORDER BY page_name, month ASC, browser;
```

```
┌─page_name─┬──────month─┬─browser─┬─page_view_count─┬─avg_time_on_page─┐
│ about     │ 2023-01-01 │         │               1 │               14 │
│ about     │ 2023-02-01 │         │               3 │             6.67 │
│ about     │ 2023-03-01 │         │               2 │              8.5 │
│ about     │ 2023-05-01 │         │               1 │               12 │
│ docs      │ 2023-01-01 │         │             203 │             7.13 │
│ docs      │ 2023-02-01 │         │             175 │             6.91 │
│ docs      │ 2023-03-01 │         │             185 │             7.21 │
│ docs      │ 2023-04-01 │         │             175 │             7.15 │
│ docs      │ 2023-05-01 │         │             182 │             7.53 │
│ docs      │ 2023-06-01 │         │             190 │             7.94 │
│ docs      │ 2023-07-01 │         │               8 │             9.88 │
│ landing   │ 2023-01-01 │         │          326986 │             7.01 │
│ landing   │ 2023-02-01 │         │          295510 │             6.99 │
│ landing   │ 2023-03-01 │         │          327021 │                7 │
│ landing   │ 2023-04-01 │         │          316761 │                7 │
│ landing   │ 2023-05-01 │         │          327207 │                7 │
│ landing   │ 2023-06-01 │         │          316732 │             6.99 │
│ landing   │ 2023-07-01 │         │           10702 │             7.05 │
│ products  │ 2023-01-01 │         │           13224 │             7.07 │
│ products  │ 2023-02-01 │         │           12103 │             7.02 │
│ products  │ 2023-03-01 │         │           13225 │             6.99 │
│ products  │ 2023-04-01 │         │           12807 │             6.98 │
│ products  │ 2023-05-01 │         │           13156 │             6.95 │
│ products  │ 2023-06-01 │         │           13000 │             7.01 │
│ products  │ 2023-07-01 │         │             441 │             6.95 │
└───────────┴────────────┴─────────┴─────────────────┴──────────────────┘
```

3. Чтобы данные из нового столбца исходной таблицы попадали в соответствующий столбец целевой таблицы, необходимо изменить запрос `SELECT` материализованного представления. Для этого выполните команду `ALTER TABLE…MODIFY QUERY`:

```
SET allow_experimental_alter_materialized_view_structure=1;
```

```
ALTER TABLE page_views_monthly_mv
MODIFY QUERY
SELECT
    toDate(toStartOfMonth(time)) AS month,
    page_name,
    browser,
    count() AS page_view_count,
    avgState(time_on_page) AS avg_time_on_page
FROM page_views
GROUP BY page_name, month, browser
ORDER BY page_name, month, browser;
```

4. Добавьте еще раз данные в исходную таблицу:

```
INSERT INTO page_views SELECT
    toDate('2023-01-01 00:00:00') + (rand() % 182) AS time,
    ['landing', 'products', 'docs', 'about'][toInt32(randBinomial(4, 0.01)) + 1],
    rand() % 15,
    ['firefox', 'safary', 'chrome'][(number % 3) + 1]
FROM numbers(1000000);
```

Убедитесь, что данные из столбца `browser` теперь учитываются в целевой таблице:

```
SELECT
    page_name,
    month,
    browser,
    sum(page_view_count) AS page_view_count,
    round(avgMerge(avg_time_on_page), 2) AS avg_time_on_page
FROM page_views_monthly
GROUP BY page_name, month, browser
ORDER BY page_name, month ASC, browser;
```

```
┌─page_name─┬──────month─┬─browser─┬─page_view_count─┬─avg_time_on_page─┐
│ about     │ 2023-01-01 │         │               1 │               14 │
│ about     │ 2023-02-01 │         │               3 │             6.67 │
│ about     │ 2023-02-01 │ safary  │               1 │                7 │
│ about     │ 2023-03-01 │         │               2 │              8.5 │
│ about     │ 2023-04-01 │ firefox │               1 │                1 │
│ about     │ 2023-05-01 │         │               1 │               12 │
│ about     │ 2023-05-01 │ chrome  │               1 │               14 │
│ about     │ 2023-05-01 │ safary  │               1 │                6 │
│ docs      │ 2023-01-01 │         │             203 │             7.13 │
│ docs      │ 2023-01-01 │ chrome  │              37 │             6.92 │
│ docs      │ 2023-01-01 │ firefox │              38 │             5.58 │
│ docs      │ 2023-01-01 │ safary  │              29 │             5.79 │
│ docs      │ 2023-02-01 │         │             175 │             6.91 │
│ docs      │ 2023-02-01 │ chrome  │              30 │             6.77 │
│ docs      │ 2023-02-01 │ firefox │              26 │             6.65 │
│ docs      │ 2023-02-01 │ safary  │              18 │             5.89 │
│ docs      │ 2023-03-01 │         │             185 │             7.21 │
│ docs      │ 2023-03-01 │ chrome  │              42 │             7.83 │
│ docs      │ 2023-03-01 │ firefox │              31 │             7.65 │
│ docs      │ 2023-03-01 │ safary  │              33 │             5.88 │
│ docs      │ 2023-04-01 │         │             175 │             7.15 │
│ docs      │ 2023-04-01 │ chrome  │              28 │             8.25 │
│ docs      │ 2023-04-01 │ firefox │              25 │             6.32 │
│ docs      │ 2023-04-01 │ safary  │              26 │             6.58 │
│ docs      │ 2023-05-01 │         │             182 │             7.53 │
│ docs      │ 2023-05-01 │ chrome  │              32 │             7.03 │
│ docs      │ 2023-05-01 │ firefox │              20 │             7.45 │
│ docs      │ 2023-05-01 │ safary  │              34 │             6.59 │
│ docs      │ 2023-06-01 │         │             190 │             7.94 │
│ docs      │ 2023-06-01 │ chrome  │              35 │             8.37 │
│ docs      │ 2023-06-01 │ firefox │              29 │             6.69 │
│ docs      │ 2023-06-01 │ safary  │              27 │             7.15 │
│ docs      │ 2023-07-01 │         │               8 │             9.88 │
│ docs      │ 2023-07-01 │ chrome  │               2 │                2 │
│ docs      │ 2023-07-01 │ firefox │               1 │                1 │
│ docs      │ 2023-07-01 │ safary  │               1 │                1 │
│ landing   │ 2023-01-01 │         │          326986 │             7.01 │
│ landing   │ 2023-01-01 │ chrome  │           54763 │             6.99 │
│ landing   │ 2023-01-01 │ firefox │           54665 │             7.02 │
│ landing   │ 2023-01-01 │ safary  │           54294 │                7 │
│ landing   │ 2023-02-01 │         │          295510 │             6.99 │
│ landing   │ 2023-02-01 │ chrome  │           48944 │             7.03 │
│ landing   │ 2023-02-01 │ firefox │           49031 │             7.03 │
│ landing   │ 2023-02-01 │ safary  │           49298 │                7 │
│ landing   │ 2023-03-01 │         │          327021 │                7 │
│ landing   │ 2023-03-01 │ chrome  │           54381 │             6.98 │
│ landing   │ 2023-03-01 │ firefox │           54433 │             7.03 │
│ landing   │ 2023-03-01 │ safary  │           54556 │             7.01 │
│ landing   │ 2023-04-01 │         │          316761 │                7 │
│ landing   │ 2023-04-01 │ chrome  │           52931 │             6.95 │
│ landing   │ 2023-04-01 │ firefox │           52706 │             6.99 │
│ landing   │ 2023-04-01 │ safary  │           53075 │             6.99 │
│ landing   │ 2023-05-01 │         │          327207 │                7 │
│ landing   │ 2023-05-01 │ chrome  │           54538 │             6.99 │
│ landing   │ 2023-05-01 │ firefox │           54733 │             7.02 │
│ landing   │ 2023-05-01 │ safary  │           54618 │             7.02 │
│ landing   │ 2023-06-01 │         │          316732 │             6.99 │
│ landing   │ 2023-06-01 │ chrome  │           52763 │             6.97 │
│ landing   │ 2023-06-01 │ firefox │           52955 │                7 │
│ landing   │ 2023-06-01 │ safary  │           52795 │             7.01 │
│ landing   │ 2023-07-01 │         │           10702 │             7.05 │
│ landing   │ 2023-07-01 │ chrome  │            1762 │             7.08 │
│ landing   │ 2023-07-01 │ firefox │            1747 │             6.98 │
│ landing   │ 2023-07-01 │ safary  │            1767 │             7.09 │
│ products  │ 2023-01-01 │         │           13224 │             7.07 │
│ products  │ 2023-01-01 │ chrome  │            2208 │             7.04 │
│ products  │ 2023-01-01 │ firefox │            2210 │             6.99 │
│ products  │ 2023-01-01 │ safary  │            2273 │             6.99 │
│ products  │ 2023-02-01 │         │           12103 │             7.02 │
│ products  │ 2023-02-01 │ chrome  │            1970 │              7.1 │
│ products  │ 2023-02-01 │ firefox │            1978 │             6.88 │
│ products  │ 2023-02-01 │ safary  │            1870 │             7.01 │
│ products  │ 2023-03-01 │         │           13225 │             6.99 │
│ products  │ 2023-03-01 │ chrome  │            2189 │             6.97 │
│ products  │ 2023-03-01 │ firefox │            2186 │             6.94 │
│ products  │ 2023-03-01 │ safary  │            2251 │             7.18 │
│ products  │ 2023-04-01 │         │           12807 │             6.98 │
│ products  │ 2023-04-01 │ chrome  │            2252 │             6.91 │
│ products  │ 2023-04-01 │ firefox │            2210 │              7.1 │
│ products  │ 2023-04-01 │ safary  │            2089 │             7.04 │
│ products  │ 2023-05-01 │         │           13156 │             6.95 │
│ products  │ 2023-05-01 │ chrome  │            2214 │                7 │
│ products  │ 2023-05-01 │ firefox │            2188 │             6.94 │
│ products  │ 2023-05-01 │ safary  │            2151 │             6.98 │
│ products  │ 2023-06-01 │         │           13000 │             7.01 │
│ products  │ 2023-06-01 │ chrome  │            2134 │             6.89 │
│ products  │ 2023-06-01 │ firefox │            2054 │             7.18 │
│ products  │ 2023-06-01 │ safary  │            2062 │             6.91 │
│ products  │ 2023-07-01 │         │             441 │             6.95 │
│ products  │ 2023-07-01 │ chrome  │              77 │             6.65 │
│ products  │ 2023-07-01 │ firefox │              67 │             7.27 │
│ products  │ 2023-07-01 │ safary  │              64 │             7.16 │
└───────────┴────────────┴─────────┴─────────────────┴──────────────────┘
```

Проекции

Елена Дворядкина

Содержание

Назад к содержанию

Обзор

Управление проекциями

- Создание/удаление проекций
- Настройки для применения проекций

Информация о проекциях

Примеры

- Фильтрация без использования первичного ключа
- Предварительное агрегирование значений
- Просмотр информации о проекциях

Обзор

Проекция (projection) — дополнительная скрытая таблица, которая хранит данные исходной таблицы в преобразованном виде, оптимальном для выполнения какого-либо типа запросов. Проекция может хранить строки данных отсортированными в отличном от исходной таблицы порядке (и, следовательно, иметь другой первичный индекс) или вычислять агрегированные значения по столбцам исходной таблицы. Таким образом, эта функциональность может быть полезна для оптимизации:

- запросов с фильтрацией по столбцам исходной таблицы, которые не являются частью ее первичного ключа;
- запросов, агрегирующих значения столбцов исходной таблицы (предварительно выполняемая проекциями агрегация данных может снизить потребление как вычислительных ресурсов, так и ресурсов I/O).

Проекции поддерживаются только таблицами семейства MergeTree (включая Replicated*). Для таблицы можно определить одну или несколько проекций — все они будут автоматически синхронизироваться с исходной таблицей, то есть обновляться при каждой вставке новых данных в исходную таблицу.

Проекции похожи на материализованные представления (которые также позволяют предварительно вычислять агрегаты и использовать иные условия сортировки для данных другой таблицы), но определяются на уровне кусков данных — для каждого куска данных создается проекция данных в нем на основе запроса проекции. Это гарантирует согласованность данных проекций с данными исходной таблицы и позволяет ADQM автоматически выбирать и использовать оптимальные проекции для выполнения пользовательских запросов (проекция считается оптимальной, если она возвращает соответствующий запросу результат, требуя при этом чтения наименьшего объема данных).

В отличие от материализованных представлений проекции не требуют создания вручную отдельных дополнительных таблиц с различными структурами, о которых нужно помнить и каждый раз при формулировке запроса самостоятельно выбирать в какую таблицу его направить, чтобы обеспечить высокую скорость его выполнения. С использованием проекций достаточно направлять все запросы в исходную таблицу, при этом ADQM будет автоматически определять есть ли у таблицы проекция, которая может ускорить выполнение запроса, и если такая проекция есть, перенаправит запрос в нее — то есть пользователю ничего не надо знать о проекциях.

ВНИМАНИЕ

Так как каждая проекция создает новую скрытую таблицу (файлы которой хранятся в подкаталогах *<projection_name>.proj* внутри каталогов кусков данных исходной таблицы MergeTree), может потребоваться больше ресурсов I/O и места на диске. Например, если проекция создается для изменения ключа сортировки, в нее продублируются все данные из исходной таблицы.

Управление проекциями

Создание/удаление проекций

Проекцию можно указать при создании таблицы MergeTree в секции описания столбцов с помощью выражения `PROJECTION`. Синтаксис запроса для создания таблицы с проекцией в общем виде:

```
CREATE TABLE <table_name>
(   <column1> ...,
    <column2> ...,
    ...
    PROJECTION <projection_name> (SELECT <column_list> [ORDER BY <sorting_key>] [GROUP BY <grouping_key>])
)
ENGINE = MergeTree()
...;
```

После имени проекции в скобках указывается **запрос проекции**, который определяет как данные исходной таблицы будут преобразовываться для сохранения в таблице проекции. Запрос проекции может включать одно из выражений:

- `ORDER BY` — ключ, на основе которого будут сортироваться строки данных и устанавливаться первичный ключ в проекции;
- `GROUP BY` — ключ, по которому данные исходной таблицы будут группироваться для вычисления агрегаций в проекции. В этом случае для скрытой таблицы проекции используется движок AggregatingMergeTree, а все агрегатные функции преобразуются в AggregateFunction.

Управлять проекциями существующих таблиц можно с помощью приведенных ниже запросов `ALTER TABLE`.

- Добавление проекции в существующую таблицу:

  ```
  ALTER TABLE [<db_name.]<table_name> [ON CLUSTER <cluster_name>]
      ADD PROJECTION [IF NOT EXISTS] <projection_name> (SELECT <column_list> [ORDER BY <sorting_key>] [GROUP BY <grouping_key>]);
  ```

  Этот запрос добавляет описание проекции в метаданные таблицы MergeTree, но не заполняет проекцию данными (она будет перестраиваться только при вставке новых данных в исходную таблицу или при материализации).

- Материализация проекции, то есть заполнение проекции текущими данными исходной таблицы (можно выполнить для отдельной партиции):

  ```
  ALTER TABLE [<db_name.]<table_name> [ON CLUSTER <cluster_name>]
      MATERIALIZE PROJECTION [IF EXISTS] <projection_name> [IN PARTITION <partition_name>];
  ```

- Удаление файлов проекции с диска без удаления ее описания из метаданных исходной таблицы (можно выполнить для отдельной партиции):

  ```
  ALTER TABLE [<db_name.]<table_name> [ON CLUSTER <cluster_name>]
      CLEAR PROJECTION [IF EXISTS] <projection_name> [IN PARTITION <partition_name>];
  ```

- Удаление описания проекции из метаданных исходной таблицы и удаление файлов проекции с диска:

  ```
  ALTER TABLE [<db_name.]<table_name> [ON CLUSTER <cluster_name>] DROP PROJECTION [IF EXISTS] <partition_name>;
  ```

Команды `ADD PROJECTION`, `CLEAR PROJECTION` и `DROP PROJECTION` являются легковесными, поскольку они только меняют метаданные или удаляют файлы.

Настройки для применения проекций

Чтобы указать ADQM нужно ли использовать проекции при выполнении запросов, используйте следующие параметры (настраиваются на уровне сессии):

- `optimize_use_projections` — включает/отключает возможность автоматического использования проекций при выполнении запросов `SELECT` (значение по умолчанию — `1`, то есть проекции могут быть использованы);
- `force_optimize_projection` — включает/отключает обязательное использование проекций при выполнении запросов `SELECT`, если поддержка проекций включена настройкой `optimize_use_projections` (значение по умолчанию — `0`).

Информация о проекциях

Информацию о проекциях можно получить из системных таблиц:

- `system.projection_parts` — информация о существующих на данный момент кусках данных проекций всех таблиц MergeTree в кластере (информация о соответствующих кусках данных исходной таблицы содержится в столбцах с префиксом `parent_`);
- `system.projection_parts_columns` — информация о столбцах всех существующих на данный момент кусков данных проекций таблиц MergeTree.

Так как таблицы проекций являются скрытыми, они не включаются в список `SHOW TABLES`.

Примеры

Создайте таблицу MergeTree, в которой будут храниться данные о просмотрах страниц (в примерах ниже для этой таблицы будут определены проекции):

```
CREATE TABLE page_views (time DateTime, page_name String, time_on_page UInt64) ENGINE = MergeTree ORDER BY time;
```

Заполните таблицу тестовыми данными. Например, сгенерируйте случайным образом 1 миллион просмотров для 4 страниц приблизительно за последние 3 года относительного текущего момента:

```
INSERT INTO page_views SELECT
    now() - toIntervalMinute(rand() % 500000 + number),
    ['landing', 'products', 'docs', 'about'][toInt32(randBinomial(4, 0.01)) + 1],
    rand() % 15
FROM numbers(1000000);
```

Фильтрация без использования первичного ключа

Если выполнить запрос с фильтрацией по столбцу `time`, который входит в первичный ключ таблицы `page_views`, полное сканирование таблицы не требуется:

```
SELECT count() FROM page_views WHERE toYear(time) = '2022';
```

```
   ┌─count()─┐
1. │  388601 │
   └─────────┘

1 row in set. Elapsed: 0.004 sec. Processed 393.22 thousand rows, 1.57 MB (99.18 million rows/s., 396.72 MB/s.)
```

При фильтрации данных не по ключевому столбцу выполняется полное сканирование таблицы, например:

```
SELECT count() FROM page_views WHERE page_name = 'products';
```

```
   ┌─count()─┐
1. │   38729 │
   └─────────┘

1 row in set. Elapsed: 0.006 sec. Processed 1.00 million rows, 16.04 MB (156.85 million rows/s., 2.52 GB/s.)
```

Для оптимизации таких запросов в таблицу можно добавить проекцию с другим первичным индексом. Например, создайте проекцию `page_name_projection`, в которой строки сортируются по столбцу `page_name`:

```
ALTER TABLE page_views ADD PROJECTION page_name_projection (SELECT * ORDER BY page_name);
```

Выполните материализацию проекции, чтобы построить ее на основе текущих данных таблицы `page_views`:

```
ALTER TABLE page_views MATERIALIZE PROJECTION page_name_projection;
```

Повторите запрос с фильтрацией по столбцу `page_name`:

```
SELECT count() FROM page_views WHERE page_name = 'products';
```

Теперь для выполнения запроса считывается 41.54 тысяч строк вместо 1 миллиона:

```
   ┌─count()─┐
1. │   38729 │
   └─────────┘

1 row in set. Elapsed: 0.003 sec. Processed 41.54 thousand rows, 703.30 KB (12.48 million rows/s., 211.24 MB/s.)
```

Используйте выражение EXPLAIN, чтобы убедиться, что для выполнения запроса ADQM использует проекцию (`ReadFromMergeTree (page_name_projection)`):

```
EXPLAIN SELECT count() FROM page_views WHERE page_name = 'products';
```

```
   ┌─explain────────────────────────────────────┐
1. │ Expression ((Project names + Projection))  │
2. │   Aggregating                              │
3. │     Filter                                 │
4. │       ReadFromMergeTree (page_name_projection) │
   └────────────────────────────────────────────┘
```

Предварительное агрегирование значений

Выполните запрос, возвращающий количество просмотров и среднее время просмотра страниц по месяцам:

```
SELECT
    page_name,
    toStartOfMonth(time) AS month,
    count() AS page_view_count,
    round(avg(time_on_page), 2) AS avg_time_on_page
FROM page_views
GROUP BY page_name, month
ORDER BY page_name, month;
```

Для выполнения этого запроса считываются все строки таблицы `page_views`:

```
Query id: 303db76e-1e68-4454-91ea-a951c36d170b

     ┌─page_name─┬──────month─┬─page_view_count─┬─avg_time_on_page─┐
  1. │ about     │ 2022-03-01 │               1 │                9 │
  2. │ about     │ 2023-06-01 │               1 │                6 │

      ...

105. │ products  │ 2024-06-01 │             313 │             6.87 │
106. │ products  │ 2024-07-01 │             116 │             6.26 │
107. │ products  │ 2024-08-01 │               7 │                7 │
     └───────────┴────────────┴─────────────────┴──────────────────┘

107 rows in set. Elapsed: 0.018 sec. Processed 1.00 million rows, 28.04 MB (54.22 million rows/s., 1.52 GB/s.)
```

Создайте и материализуйте проекцию, вычисляющую количество просмотров и среднее время просмотра для каждой страницы в таблице `page_views` по месяцам:

```
ALTER TABLE page_views ADD PROJECTION page_views_monthly_projection
    (
        SELECT
            page_name,
            toStartOfMonth(time),
            count(),
            round(avg(time_on_page), 2)
        GROUP BY
            page_name,
            toStartOfMonth(time)
    );
```

```
ALTER TABLE page_views MATERIALIZE PROJECTION page_views_monthly_projection;
```

Повторите запрос:

```
SELECT
    page_name,
    toStartOfMonth(time) AS month,
    count() AS page_view_count,
    round(avg(time_on_page), 2) AS avg_time_on_page
FROM page_views
GROUP BY page_name, month
ORDER BY page_name, month;
```

В результате получены те же данные, но производительность стала лучше:

```
Query id: acb9a600-cbd7-44a4-aa61-ee5f8cdd941d

     ┌─page_name─┬──────month─┬─page_view_count─┬─avg_time_on_page─┐
  1. │ about     │ 2022-03-01 │               1 │                9 │
  2. │ about     │ 2023-06-01 │               1 │                6 │

      ...

105. │ products  │ 2024-06-01 │             313 │             6.87 │
106. │ products  │ 2024-07-01 │             116 │             6.26 │
107. │ products  │ 2024-08-01 │               7 │                7 │
     └───────────┴────────────┴─────────────────┴──────────────────┘

107 rows in set. Elapsed: 0.004 sec.
```

С помощью `EXPLAIN` убедитесь, что для выполнения этого запроса используется проекция:

```
EXPLAIN SELECT
    page_name,
    toStartOfMonth(time) AS month,
    count() AS page_view_count,
    round(avg(time_on_page), 2) AS avg_time_on_page
FROM page_views
GROUP BY page_name, month
ORDER BY page_name, month;
```

```
   ┌─explain─────────────────────────────────────────────────────────────────────┐
1. │ Expression ((Project names + (Before ORDER BY + Projection) [lifted up part])) │
2. │   Sorting (Sorting for ORDER BY)                                            │
3. │     Expression ((Before ORDER BY + Projection))                             │
4. │       Aggregating                                                           │
5. │         Expression                                                          │
6. │           ReadFromMergeTree (page_views_monthly_projection)                 │
   └─────────────────────────────────────────────────────────────────────────────┘
```

Также можно посмотреть лог-таблицу `system.query_log`, в которой поле `projections` покажет имя используемой проекции (или будет пустым, если ни одна проекция не использовалась):

```
SELECT query, formatReadableQuantity(read_rows) AS read_rows, projections FROM system.query_log
WHERE type='QueryFinish' AND query_id='acb9a600-cbd7-44a4-aa61-ee5f8cdd941d';\G
```

```
Row 1:
──────
query:       SELECT
                 page_name,
                 toStartOfMonth(time) AS month,
                 count() AS page_view_count,
                 round(avg(time_on_page), 2) AS avg_time_on_page
             FROM page_views
             GROUP BY page_name, month
             ORDER BY page_name, month;
read_rows:   107.00
projections: ['default.page_views.page_views_monthly_projection']
```

Проекция, агрегирующая значения исходной таблицы, используется не только для запросов, полностью совпадающих с запросом проекции, но и для запросов на выборку части соответствующих агрегатов с группировкой по некоторым столбцам из `GROUP BY` запроса проекции. Например:

```
EXPLAIN SELECT page_name, count() AS page_view_count FROM page_views GROUP BY page_name;
```

```
   ┌─explain─────────────────────────────────────────┐
1. │ Expression ((Project names + Projection))       │
2. │   Aggregating                                   │
3. │     Expression                                  │
4. │       ReadFromMergeTree (page_views_monthly_projection) │
   └─────────────────────────────────────────────────┘
```

Просмотр информации о проекциях

Получите информацию об активных кусках данных исходной таблицы из системной таблицы `system.parts`:

```
SELECT name, active, path FROM system.parts WHERE table='page_views';
```

```
   ┌─name────────┬─active─┬─path──────────────────────────────────────────────────────────────────┐
1. │ all_1_1_0_3 │      1 │ /var/lib/clickhouse/store/7f1/7f1aea44-35a3-4b96-a5c9-07640aec1340/all_1_1_0_3/ │
   └─────────────┴────────┴───────────────────────────────────────────────────────────────────────┘
```

В каталоге куска данных таблицы `page_views` присутствуют подкаталоги ее проекций (*page_name_projection.proj* и *page_views_monthly_projection.proj*):

```
$ sudo ls /var/lib/clickhouse/store/7f1/7f1aea44-35a3-4b96-a5c9-07640aec1340/all_1_1_0_3/
```

```
checksums.txt  default_compression_codec.txt  page_name.cmrk2                     primary.cidx        time.cmrk2
columns.txt    metadata_version.txt           page_name_projection.proj           serialization.json  time_on_page.bin
count.txt      page_name.bin                  page_views_monthly_projection.proj  time.bin            time_on_page.cmrk2
```

В подкаталоге каждой проекции содержатся файлы данных столбцов (*.bin*), файлы засечек столбцов (*.cmrk2*) и файл первичного индекса (*primary.cidx*) скрытой таблицы проекции:

```
$ sudo ls /var/lib/clickhouse/store/7f1/7f1aea44-35a3-4b96-a5c9-07640aec1340/all_1_1_0_3/page_name_projection.proj
```

```
checksums.txt  default_compression_codec.txt  page_name.cmrk2     time.bin          time_on_page.cmrk2
columns.txt    metadata_version.txt           primary.cidx        time.cmrk2
count.txt      page_name.bin                  serialization.json  time_on_page.bin
```

В таблице `system.projection_parts` посмотрите сколько места на диске занимают существующие проекции:

```
SELECT
    name,
    parent_name,
    formatReadableSize(bytes_on_disk) AS bytes,
    formatReadableSize(parent_bytes_on_disk) AS parent_bytes,
    bytes_on_disk / parent_bytes_on_disk AS ratio
FROM system.projection_parts;
```

```
   ┌─name──────────────────────────┬─parent_name─┬─bytes────┬─parent_bytes─┬──────────────────ratio─┐
1. │ page_name_projection          │ all_1_1_0_3 │ 5.54 MiB │ 11.24 MiB    │    0.49312069529045727 │
2. │ page_views_monthly_projection │ all_1_1_0_3 │ 1.40 KiB │ 11.24 MiB    │ 0.00012159165770921657 │
   └───────────────────────────────┴─────────────┴──────────┴──────────────┴────────────────────────┘
```

Из таблицы `system.projection_parts_columns` получите типы столбцов проекций:

```
SELECT name as projection, column, type as column_type, column_position FROM system.projection_parts_columns;
```

```
   ┌─projection────────────────────┬─column───────────────┬─column_type────────────────────┬─column_position─┐
1. │ page_name_projection          │ time                 │ DateTime                       │               1 │
2. │ page_name_projection          │ page_name            │ String                         │               2 │
3. │ page_name_projection          │ time_on_page         │ UInt64                         │               3 │
4. │ page_views_monthly_projection │ page_name            │ String                         │               1 │
5. │ page_views_monthly_projection │ toStartOfMonth(time) │ Date                           │               2 │
6. │ page_views_monthly_projection │ count()              │ AggregateFunction(count)       │               3 │
7. │ page_views_monthly_projection │ avg(time_on_page)    │ AggregateFunction(avg, UInt64) │               4 │
   └───────────────────────────────┴──────────────────────┴────────────────────────────────┴─────────────────┘
```

Пользовательские функции и таблицы Executable

Елена Дворядкина

Содержание

Назад к содержанию

Пользовательские функции SQL
- Пример

Исполняемые пользовательские функции
- Пример

Табличная функция executable
- Пример

Табличные движки Executable и ExecutablePool
- Пример

Параметры выполнения скрипта

В данной статье описаны способы расширения встроенной функциональности ADQM/ClickHouse для выполнения специализированных задач:

- пользовательские функции SQL — создаются из лямбда-выражений, которые могут использовать конструкции и функции SQL;
- исполняемые пользовательские функции | табличная функция executable | табличные движки Executable и ExecutablePool — инструменты, позволяющие вызывать внешние исполняемые программы или скрипты (например, Bash, Python, C++) для обработки данных.

Создание таблицы для выполнения примеров

Создайте таблицу `sales` с тестовыми данными, которые будут использоваться в примерах, приведенных в данной статье:

```
CREATE TABLE sales (sale_id UInt32, product_name String, product_price UInt32, quantity UInt8)
ENGINE = MergeTree() ORDER BY (sale_id);
```

```
INSERT INTO sales VALUES
    (1, 'product_1', 100, 15),
    (2, 'product_2', 200, 20),
    (3, 'product_3', 300, 5),
    (4, 'product_2', 200, 25),
    (5, 'product_1', 100, 50);
```

```
   ┌─sale_id─┬─product_name─┬─product_price─┬─quantity─┐
1. │       1 │ product_1    │           100 │       15 │
2. │       2 │ product_2    │           200 │       20 │
3. │       3 │ product_3    │           300 │        5 │
4. │       4 │ product_2    │           200 │       25 │
5. │       5 │ product_1    │           100 │       50 │
   └─────────┴──────────────┴───────────────┴──────────┘
```

Пользовательские функции SQL

Чтобы создать функцию из лямбда-выражения, используйте запрос `CREATE FUNCTION`:

```
CREATE FUNCTION <function_name> [ON CLUSTER <cluster_name>] AS (<parameter_0>, <parameter_1> ...) -> <expression>;
```

где:

- `<function_name>` — имя функции, которое должно быть уникально среди всех пользовательских и системных функций;
- `<parameter_0>, <parameter_1> …` — список параметров, в котором должны быть перечислены все используемые функцией переменные;
- `<expression>` — выражение, состоящее из параметров функции, констант, операторов и вызовов других функций. Рекурсивные функции не допускаются.

Удалить пользовательскую функцию SQL можно запросом `DROP FUNCTION <function_name>`.

Пример

Создайте функцию, вычисляющую произведение аргументов:

```
CREATE FUNCTION sql_udf AS (a, b) -> a * b;
```

Убедитесь, что функция добавлена в список функций ADQM:

```
SELECT name, create_query FROM system.functions WHERE origin = 'SQLUserDefined';
```

```
   ┌─name────┬─create_query─────────────────────────────┐
1. │ sql_udf │ CREATE FUNCTION sql_udf AS (a, b) -> (a * b) │
   └─────────┴──────────────────────────────────────────┘
```

Используйте созданную функцию для расчета общей стоимости каждой продажи в тестовой таблице `sales`:

```
SELECT sale_id, product_price, quantity, sql_udf(product_price, quantity) AS total_price FROM sales;
```

```
   ┌─sale_id─┬─product_price─┬─quantity─┬─total_price─┐
1. │       1 │           100 │       15 │        1500 │
2. │       2 │           200 │       20 │        4000 │
3. │       3 │           300 │        5 │        1500 │
4. │       4 │           200 │       25 │        5000 │
5. │       5 │           100 │       50 │        5000 │
   └─────────┴───────────────┴──────────┴─────────────┘
```

Исполняемые пользовательские функции

Исполняемая пользовательская функция (User-Defined Function, UDF) вызывает внешнюю программу или скрипт для обработки данных. Чтобы создать такую функцию, необходимо выполнить следующие действия:

1. Опишите конфигурацию функции в XML-файле:

```
<functions>
    <function>
        <name>...</name>
        <type>executable</type>
        <return_type>...</return_type>
        <return_name>...</return_name>
        <argument>
            <type>...</type>
            <name>...</name>
        </argument>
        <argument>
            <type>...</type>
            <name>...</name>
        </argument>
        <format>...</format>
        <command>...</command>
    </function>
</functions>
```

Параметры конфигурации исполняемой пользовательской функции

name	Имя функции
type	Вариант запуска команды. Возможные значения: - `executable` — запускается одна команда; - `executable_pool` — создается пул команд.
return_type	Тип возвращаемого значения
return_name	Имя возвращаемого значения. Необходимо указать, если имя возвращаемого значения является частью сериализации для формата пользовательской функции (например, Native или JSONEachRow). Значение по умолчанию — `result`
argument	Описание аргумента, включающее его тип (`type`) и, опционально, имя (`name`). Имена аргументов необходимо указывать, если они являются частью сериализации для формата пользовательской функции (например, Native или JSONEachRow). Значение имени аргумента по умолчанию — `c` с номером аргумента. Каждый аргумент функции описывается отдельно
format	Формат, в котором аргументы передаются в команду
command	Имя скрипта для выполнения или команда, если значение параметра `execute_direct` — `false`. Команда должна читать аргументы из stdin и выводить результат в stdout. Обработка должна выполняться в цикле, то есть после обработки группы аргументов команда должна ожидать следующую группу
lifetime	Интервал перезагрузки функции в секундах. Если значение `0` (по умолчанию), функция не перезагружается
execute_direct	Если значение `1` (по умолчанию), выполняется поиск команды, указанной в `command`, в каталоге с файлами пользовательских скриптов, который определяется параметром `user_scripts_path` в конфигурации сервера. Дополнительные аргументы скрипта можно указать, разделяя их пробелами (например: `<script_name> <arg1> <arg2>`). Если `execute_direct = 0`, содержимое `command` передается как аргумент для `bin/sh -c`

В конфигурации исполняемой пользовательской функции можно также дополнительно указать параметры выполнения скрипта, описание которых приведено ниже.

2. В параметре `user_defined_executable_functions_config` конфигурации сервера (*config.xml*) укажите путь к XML-файлу с конфигурацией функции (путь можно указать абсолютным или относительно конфигурационного файла сервера). Конфигурации исполняемых пользовательских функций могут находиться в одном или нескольких XML-файлах — в последнем случае можно использовать wildcard-символы `*` и `?`, чтобы указать путь к файлам, например:

```
<user_defined_executable_functions_config>*_function.xml</user_defined_executable_functions_config>
```

3. В параметре `user_scripts_path` конфигурации сервера укажите каталог, где будут храниться скрипты для исполняемых пользовательских функций (по умолчанию */var/lib/clickhouse/user_scripts*), и поместите в этот каталог скрипт, указанный в конфигурации функции как `command`. Убедитесь, что:

   - На сервере ADQM установлено программное обеспечение, необходимое для запуска исполняемого скрипта.
   - На выполнение скрипта выдано разрешение.

Пример

Данный пример показывает, как создать пользовательскую функцию, выполняющую Python-скрипт. Предварительно проверьте, что на сервере ADQM установлен Python и все необходимые библиотеки.

Опишите конфигурацию функции в файле */etc/clickhouse-server/UDFs.xml*:

```
<functions>
    <function>
        <type>executable</type>
        <name>total_price_function</name>
        <return_type>UInt32</return_type>
        <return_name>total_price</return_name>
        <argument>
            <type>UInt32</type>
            <name>product_price</name>
        </argument>
        <argument>
            <type>UInt8</type>
            <name>quantity</name>
        </argument>
        <format>JSONEachRow</format>
        <command>total_price.py</command>
    </function>
</functions>
```

Укажите путь к файлу с конфигурацией исполняемой пользовательской функции в параметре `user_defined_executable_functions_config` файла *config.xml*:

```
<user_defined_executable_functions_config>UDFs.xml</user_defined_executable_functions_config>
```

В файле */var/lib/clickhouse/user_scripts/total_price.py* напишите Python-скрипт:

```
#!/usr/bin/python3

import sys
import json

if __name__ == '__main__':
    for line in sys.stdin:
        data = json.loads(line)
        price = data['product_price']
        quantity = data['quantity']
        result = {'total_price': price * quantity}
        print(json.dumps(result), end='\n')
        sys.stdout.flush()
```

Дайте разрешение на выполнение скрипта:

```
$ sudo chmod +x /var/lib/clickhouse/user_scripts/total_price.py
```

Убедитесь, что функция добавлена в ADQM:

```
SELECT name FROM system.functions WHERE origin = 'ExecutableUserDefined';
```

```
   ┌─name─────────────────────┐
1. │ total_price_function     │
   └──────────────────────────┘
```

Выполните запрос на выборку данных из таблицы `sales`, используя функцию `total_price_function` для вычисления стоимости каждой продажи:

```
SELECT sale_id, product_price, quantity, total_price_function(product_price, quantity) AS total_price FROM sales;
```

```
   ┌─sale_id─┬─product_price─┬─quantity─┬─total_price─┐
1. │       1 │           100 │       15 │        1500 │
2. │       2 │           200 │       20 │        4000 │
3. │       3 │           300 │        5 │        1500 │
4. │       4 │           200 │       25 │        5000 │
5. │       5 │           100 │       50 │        5000 │
   └─────────┴───────────────┴──────────┴─────────────┘
```

Табличная функция executable

Табличная функция `executable` создает таблицу на основе выходных данных функции, определенной в пользовательском скрипте, который выводит строки в stdout.

Синтаксис табличной функции `executable` в общем виде:

```
executable(<script_name>, <format>, <table_structure>, [<input_query> ...] [,SETTINGS <parameter_name>=<value>, ...])
```

где:

- `<script_name>` — имя файла исполняемого скрипта, сохраненного в каталоге, который определяется параметром `user_scripts_path` в конфигурации сервера (по умолчанию — */var/lib/clickhouse/user_scripts/*). Скрипт может считывать данные из любого источника.
- `<format>` — формат, в котором функция будет принимать данные.
- `<table_structure>` — структура генерируемой таблицы (указывается в виде `<column_name1> <data_type1>, <column_name2> <data_type2>, …`).
- `<input_query>` — запрос (или набор запросов), результаты которого будут передаваться в stdin для чтения скриптом.
- `SETTINGS` — дополнительные настройки выполнение скрипта.

Перед вызовом табличной функции `executable` убедитесь, что на сервере ADQM установлены все необходимые пакеты для запуска исполняемого скрипта, а также что на выполнения скрипта выдано разрешение.

ПРИМЕЧАНИЕ

- Если необходимо многократно вызывать один и тот же скрипт с одними и теми же входными запросами, рассмотрите возможность использования табличного движка Executable или ExecutablePool.
- Ключевым отличием между обычными исполняемыми пользовательскими функциями и табличной функцией `executable` или табличным движком Executable является то, что обычные исполняемые пользовательские функции не могут изменять количество строк (например, если входные данные содержат 100 строк, в результате вернется 100 строк), а при использовании в табличной функции `executable` или табличном движке Executable скрипт может выполнять любые преобразования данных, включая сложные агрегации.

Пример

Заполните файл */var/lib/clickhouse/user_scripts/add_tax.py* текстом Python-скрипта, который будет добавлять налог к цене продукта:

```
#!/usr/bin/python3

import sys

if __name__ == '__main__':
     for line in sys.stdin:

            split_line = line.split()

            product = split_line[0]
            price = int(split_line[1])
            price_with_tax = price*1.2

            print(product + '\t' + str(price) + '\t' + str(price_with_tax) + '\n', end='')
            sys.stdout.flush()
```

Дайте разрешение на выполнение скрипта:

```
$ sudo chmod +x /var/lib/clickhouse/user_scripts/add_tax.py
```

Используйте табличную функцию `executable`, чтобы получить таблицу с ценами уникальных продуктов из таблицы продаж `sales` и вычислить цены с учетом налога с помощью скрипта *add_tax.py*:

```
SELECT * FROM executable(
    'add_tax.py',
    TabSeparated,
    'product  String, price UInt32, price_with_tax Float32',
    (SELECT DISTINCT ON (product_name) product_name, product_price FROM sales));
```

Результат:

```
   ┌─product───┬─price─┬─price_with_tax─┐
1. │ product_1 │   100 │            120 │
2. │ product_2 │   200 │            240 │
3. │ product_3 │   300 │            360 │
   └───────────┴───────┴────────────────┘
```

Табличные движки Executable и ExecutablePool

Табличные движки Executable и ExecutablePool позволяют создать таблицу, строки которой генерируются из указанного скрипта:

- Executable — запускает скрипт при каждом запросе данных;
- ExecutablePool — поддерживает пул постоянных процессов и берет процессы из пула для чтения данных.

Синтаксис запроса для создания таблицы на основе движка Executable или ExecutablePool в общем виде:

```
CREATE TABLE <table_name> (<column_name1> <data_type1>, <column_name2> <data_type2>, ...)
ENGINE = Executable|ExecutablePool(<script_name>, <format>, [<input_query> ...])
[SETTINGS <parameter_name>=<value>, ...];
```

где:

- `<script_name>` — исполняемый скрипт, на основе выходных данных которого будут строиться данные таблицы `<table_name>` каждый раз, когда в нее будет направляться запрос `SELECT`. Скрипт может считывать данные из любого источника. Файл скрипта должен быть сохранен в каталоге, определяемом параметром конфигурации сервера `user_scripts_path` (по умолчанию — */var/lib/clickhouse/user_scripts/*).
- `<format>` — формат данных, в котором таблица будет принимать данные.
- `<input_query>` — запрос (или набор запросов), результаты которого будут передаваться в stdin для чтения скриптом.
- `SETTINGS` — дополнительные параметры выполнения скрипта.

Пример

Создайте таблицу типа Executable, которая будет заполняться данными, генерируемыми скриптом *add_tax.py* из примера выше:

```
CREATE TABLE prices_with_tax (product  String, price UInt32, price_with_tax Float32)
ENGINE = Executable('add_tax.py', TabSeparated, (SELECT DISTINCT ON (product_name) product_name, product_price FROM sales));
```

Запрос `CREATE TABLE` создает таблицу, но не вызывает скрипт. Скрипт вызывается запросом на выборку данных из таблицы:

```
SELECT * FROM prices_with_tax;
```

```
   ┌─product───┬─price─┬─price_with_tax─┐
1. │ product_1 │   100 │            120 │
2. │ product_2 │   200 │            240 │
3. │ product_3 │   300 │            360 │
   └───────────┴───────┴────────────────┘
```

Параметры выполнения скрипта

При создании исполняемой пользовательской функции, вызове табличной функции `executable` и создании таблиц на основе движков Executable/ExecutablePool (базовый синтаксис запросов приводится в соответствующих разделах выше) можно указать следующие параметры, регулирующие выполнение скрипта.

send_chunk_header	Контролирует, нужно ли отправлять количество строк в каждом блоке данных перед отправкой блока на обработку. Значение по умолчанию — `false`
command_termination_timeout	Максимальное время (в секундах), в течение которого команда должна завершиться после закрытия конвейера. Если команда не завершается, то процессу отправляется сигнал `SIGTERM`. Значение по умолчанию — `10`
command_read_timeout	Время ожидания чтения данных из stdout команды (в миллисекундах). Значение по умолчанию — `10000`
command_write_timeout	Время ожидания записи данных в stdin команды (в миллисекундах). Значение по умолчанию — `10000`
pool_size	Размер пула процессов. Если значение `0`, то ограничений по размеру пула нет. Значение по умолчанию — `16`. Параметр релевантен для: - исполняемой пользовательской функции типа `executable_pool`; - табличной функции executable; - табличного движка ExecutablePool.
max_command_execution_time	Максимальное время выполнения команды исполняемого скрипта для обработки блока данных (в секундах). Значение по умолчанию — `10` Параметр релевантен для: - исполняемой пользовательской функции типа `executable_pool`; - табличной функции executable; - табличного движка ExecutablePool.

Пример работы с таблицами в clickhouse-client

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
- Шаг 1. Создание базы данных
- Шаг 2. Создание таблицы
- Шаг 3. Получение информации о таблице
- Шаг 4. Добавление данных в таблицу
- Шаг 5. Чтение данных из таблицы
  - Чтение всех данных
  - Фильтрация данных
  - Сортировка результатов
  - Группировка результатов
  - Комбинация SELECT с CREATE и INSERT INTO
- Шаг 6. Обновление данных в таблице
- Шаг 7. Удаление данных из таблицы
- Шаг 8. Изменение структуры таблицы
- Шаг 9. Отключение и подключение таблицы
- Шаг 10. Удаление таблицы

Обзор

Наиболее простой способ работы с таблицами ADQM/ClickHouse — использовать консольный клиент clickhouse-client, который доступен на каждом хосте кластера сразу после установки ADQM. Этот клиент позволяет вводить запросы, передавать их в ClickHouse и отображать возвращаемые результаты. Параметры в `clickhouse-client` можно передавать из командной строки или через конфигурационные файлы.

Чтобы начать работу с `clickhouse-client`, выполните команду:

```
$ clickhouse-client -m
```

Эта команда устанавливает соединение с локальным сервером ClickHouse (`localhost:9000`) от имени пользователя `default` без пароля и подключается к базе данных по умолчанию. Терминал переключается в интерактивный режим работы с сервером ClickHouse и отображает приглашение `:)`, после которого можно вводить запросы. Параметр `-m` (или `--multiline`) включает многострочный режим ввода запроса — чтобы выполнить запрос, необходимо завершить его точкой с запятой и нажать `Enter`.

```
$ clickhouse-client -m
ClickHouse client version 24.3.11.7.
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 24.3.11.

:)
```

Чтобы выйти из `clickhouse-client`, можно нажать `Ctrl+D` или ввести вместо запроса `exit`, `quit`, `q` или `logout`.

Основные операции по работе с данными в ADQM/ClickHouse описаны ниже (примеры основаны на простой базе данных, хранящей информацию о продажах). Полный список доступных команд можно найти в разделе ClickHouse SQL Statements документации ClickHouse. Перед тем как начать выполнять приведенные в данной статье команды, рекомендуется также ознакомиться с разделом ***Концепции → Модель данных*** текущей документации ADQM.

Шаг 1. Создание базы данных

База данных, к которой подключается `clickhouse-client` по умолчанию, указывается с помощью параметра ***Default database*** на странице конфигурации сервиса ADQMDB в интерфейсе ADCM (***ADQM cluster → Services → ADQMDB → Primary Configuration → Other → Default database***). Значение параметра по умолчанию — база данных `default`, которая создается автоматически при установке ADQM.

Чтобы создать новую базу данных, используйте запрос CREATE DATABASE.

Следующий запрос создает на всех хостах кластера `default_cluster` базу данных `test_db` на основе движка Atomic:

```
CREATE DATABASE test_db ON CLUSTER default_cluster;
```

Изменить текущую базу данных сессии можно с помощью команды `USE`, например:

```
USE test_db;
```

ПРИМЕЧАНИЕ

- Чтобы при подключении к `clickhouse-client` сразу выбрать текущую базу данных, отличную от установленной в ***Default database***, используйте параметр `-d` или `--database` (например, `clickhouse-client -d test_db`).
- Если в запросе требуется обратиться к таблице, которая находится не в текущей базе данных сессии, нужную базу данных необходимо указать явно через точку перед именем таблицы.

Шаг 2. Создание таблицы

Для создания новой таблицы используйте запрос `CREATE TABLE` с указанием имени таблицы и табличного движка, описанием всех столбцов таблицы (путем определения их имен, типов данных, модификаторов, кодеков сжатия данных), определением индексов, ограничений (constraints) и прочих настроек. Обзор синтаксиса с описанием параметров и выражений запроса `CREATE TABLE` можно посмотреть в разделе Создание таблиц MergeTree статьи об основных операциях с таблицами ADQM.

Следующий запрос создает таблицу `sales` со столбцами различных типов для хранения данных о продажах:

```
CREATE TABLE sales (
    id UInt32,
    product_name String,
    category String,
    sale_date Date DEFAULT toDate(now()),
    units_sold UInt32,
    price_per_unit Float32,
    region Enum('North America' = 1, 'Europe' = 2, 'Asia' = 3)
)
ENGINE = MergeTree()
ORDER BY id;
```

Шаг 3. Получение информации о таблице

Чтобы проверить существование таблицы и получить информацию о ней, можно использовать следующие команды:

- `EXISTS <table_name>` — проверяет, существует ли таблица в базе данных (возвращает столбец с одним значением — `0`, если таблицы не существует, или `1`, если таблица существует).

```
EXISTS sales;
```

Таблица `sales` существует в текущей базе данных `test_db`:

```
   ┌─result─┐
1. │      1 │
   └────────┘
```

- `DESCRIBE TABLE <table_name>` — возвращает информацию о каждом столбце таблицы (название, тип данных, выражение для значения по умолчанию, комментарий, кодек сжатия данных и выражение TTL).

```
DESCRIBE TABLE sales;
```

```
┌─name───────────┬─type──────────────────────────────────────────┬─default_type─┬─default_expression─┬─comment─┬─codec_exp
ression─┬─ttl_expression─┐
1. │ id             │ UInt32                                        │              │                    │         │
│                │
2. │ product_name   │ String                                        │              │                    │         │
│                │
3. │ category       │ String                                        │              │                    │         │
│                │
4. │ sale_date      │ Date                                          │ DEFAULT      │ toDate(now())      │         │
│                │
5. │ units_sold     │ UInt32                                        │              │                    │         │
│                │
6. │ price_per_unit │ Float32                                       │              │                    │         │
│                │
7. │ region         │ Enum8('North America' = 1, 'Europe' = 2, 'Asia' = 3) │              │                    │         │
│                │

└────────────────┴───────────────────────────────────────────────┴──────────────┴────────────────────┴─────────┴──────────
────────┴────────────────┘
```

Если вывод результата команды не умещается в окне консоли по ширине, можно использовать формат Vertical для выходных данных — в этом случае каждое значение выводится в отдельной строке. Для этого укажите `\G` вместо или после точки с запятой или `FORMAT Vertical` в конце запроса:

```
DESCRIBE TABLE sales;\G
```

```
DESCRIBE TABLE sales FORMAT Vertical;
```

Результат команды в формате Vertical

```
Row 1:
──────
name:               id
type:               UInt32
default_type:
default_expression:
comment:
codec_expression:
ttl_expression:

Row 2:
──────
name:               product_name
type:               String
default_type:
default_expression:
comment:
codec_expression:
ttl_expression:

Row 3:
──────
name:               category
type:               String
default_type:
default_expression:
comment:
codec_expression:
ttl_expression:

Row 4:
──────
name:               sale_date
type:               Date
default_type:       DEFAULT
default_expression: toDate(now())
comment:
codec_expression:
ttl_expression:

Row 5:
──────
name:               units_sold
type:               UInt32
default_type:
default_expression:
comment:
codec_expression:
ttl_expression:

Row 6:
──────
name:               price_per_unit
type:               Float32
default_type:
default_expression:
comment:
codec_expression:
ttl_expression:

Row 7:
──────
name:               region
type:               Enum8('North America' = 1, 'Europe' = 2, 'Asia' = 3)
default_type:
default_expression:
comment:
codec_expression:
ttl_expression:
```

Полный список форматов, доступных в ADQM/ClickHouse для входных и выходных данных, можно посмотреть в статье Formats for Input and Output Data документации ClickHouse.

- `SELECT * FROM system.tables WHERE name = '<table_name>'` — возвращает метаданные таблицы с указанным именем из системной таблицы `system.tables`.

```
SELECT * FROM system.tables WHERE name = 'sales';\G
```

Результат команды

```
Row 1:
──────
database:                      test_db
name:                          sales
uuid:                          4a8690b9-e048-4b55-acea-27586cd2380f
engine:                        MergeTree
is_temporary:                  0
data_paths:                    ['/var/lib/clickhouse/store/4a8/4a8690b9-e048-4b55-acea-27586cd2380f/']
metadata_path:                 /var/lib/clickhouse/store/a8e/a8e6a2c0-ea2c-4c0b-8669-de931f53178d/sales.sql
metadata_modification_time:    2025-01-16 15:17:05
metadata_version:              0
dependencies_database:         []
dependencies_table:            []
create_table_query:            CREATE TABLE test_db.sales (`id` UInt32, `product_name` String, `category` String,
`sale_date` Date DEFAULT toDate(now()), `units_sold` UInt32, `price_per_unit` Float32, `region` Enum8('North America' = 1,
'Europe' = 2, 'Asia' = 3)) ENGINE = MergeTree ORDER BY id SETTINGS index_granularity = 8192
engine_full:                   MergeTree ORDER BY id SETTINGS index_granularity = 8192
as_select:
partition_key:
sorting_key:                   id
primary_key:                   id
sampling_key:
storage_policy:                default
total_rows:                    0
total_bytes:                   0
total_bytes_uncompressed:      0
parts:                         0
active_parts:                  0
total_marks:                   0
lifetime_rows:                 NULL
lifetime_bytes:                NULL
comment:
has_own_data:                  1
loading_dependencies_database: []
loading_dependencies_table:    []
loading_dependent_database:    []
loading_dependent_table:       []
```

- `SHOW CREATE TABLE <table_name>` — возвращает один столбец с текстом запроса CREATE TABLE, с помощью которого была создана указанная таблица.

```
SHOW CREATE TABLE sales;\G
```

```
Row 1:
──────
statement: CREATE TABLE test_db.sales
(
    `id` UInt32,
    `product_name` String,
    `category` String,
    `sale_date` Date DEFAULT toDate(now()),
    `units_sold` UInt32,
    `price_per_unit` Float32,
    `region` Enum8('North America' = 1, 'Europe' = 2, 'Asia' = 3)
)
ENGINE = MergeTree
ORDER BY id
SETTINGS index_granularity = 8192
```

- `SHOW TABLES` — выводит список таблиц, существующих в базе данных.

```
SHOW TABLES;
```

В текущей базе данных `test_db` находится только одна таблица `sales`:

```
   ┌─name──┐
1. │ sales │
   └───────┘
```

Шаг 4. Добавление данных в таблицу

Чтобы добавить новые данные в таблицу, используйте запрос INSERT INTO:

```
INSERT INTO <table_name> [(<c1>, <c2>, <c3>)] VALUES (v11, v12, v13), (v21, v22, v23), ...;
```

Опционально сразу после названия таблицы можно указать список столбцов для вставки данных — перечислить имена столбцов через запятую или использовать выражение с символом `*` и модификаторами, такими как `APPLY`, `EXCEPT`, `REPLACE`. После ключевого слова `VALUES` через запятую перечисляются значения столбцов в том же порядке, что и их названия в списке `(<c1>, <c2>, <c3>)`. Если список столбцов, в которые будут вставляться данные, не определен явно, необходимо указать значения соответствующих типов для всех столбцов таблицы — приведенные значения появятся в столбцах согласно порядку их объявления при создании таблицы.

Если указанный список столбцов не включает все существующие в таблице столбцы, то все остальные столбцы заполняются:

- значениями, вычисляемыми из выражений `DEFAULT`, указанных для столбцов в определении таблицы;
- нулями и пустыми строками, если выражения `DEFAULT` для столбцов не определены.

Следующие пять отдельных запросов добавляют по одной строке в таблицу `sales`:

```
INSERT INTO sales VALUES (1, 'Laptop', 'Electronics', '2025-01-03', 5, 999.99, 1);
```

```
INSERT INTO sales VALUES (2, 'Smartphone', 'Electronics', '2025-01-04', 10, 699.99, 2);
```

```
INSERT INTO sales VALUES (3, 'Headphones', 'Electronics', '2025-01-04', 25, 49.99, 3);
```

```
INSERT INTO sales VALUES (4, 'Desk Chair', 'Furniture', '2025-01-05', 3, 149.99, 1);
```

```
INSERT INTO sales VALUES (5, 'Coffee Table', 'Furniture', '2025-01-05', 2, 249.99, 2);
```

Для добавления более одной строки в рамках одного запроса `INSERT INTO` необходимо перечислить все кортежи данных через запятую после ключевого слова `VALUES`:

```
INSERT INTO sales VALUES
(6, 'Notebook', 'Stationery', '2025-01-06', 100, 1.99, 3),
(7, 'Pen', 'Stationery', '2025-01-06', 200, 0.49, 1),
(8, 'Water Bottle', 'Accessories', '2025-01-07', 15, 9.99, 2),
(9, 'Backpack', 'Accessories', '2025-01-07', 8, 59.99, 3),
(10, 'Monitor', 'Electronics', '2025-01-08', 6, 199.99, 1),
(11, 'Keyboard', 'Electronics', '2025-01-08', 12, 49.99, 2);
```

В следующем запросе добавляются данные во все столбцы, кроме `sale_date`, поэтому передается столько значений, сколько указано столбцов, а столбец `sale_date` будет заполнен значениями по умолчанию (текущей датой):

```
INSERT INTO sales (* EXCEPT(sale_date)) VALUES (12, 'Office Chair', 'Furniture',  4, 199.99, 3);
```

Для вставки значений по умолчанию также можно использовать ключевое слово `DEFAULT`:

```
INSERT INTO sales VALUES (13, 'Printer', 'Electronics', DEFAULT, 2, 299.99, 1);
```

В запрос `INSERT INTO` можно также передавать данные любого формата, который поддерживает ClickHouse (см. столбец ***Input*** в таблице Formats for Input and Output Data). Для этого формат необходимо явно указать в запросе с помощью ключевого слова `FORMAT`, например:

```
INSERT INTO sales FORMAT JSONEachRow
{"id":14,"product_name":"Cap","category":"Accessories","sale_date":"2025-01-09","units_sold":30,"price_per_unit":5.99,"region":2}
{"id":15,"product_name":"Binder","category":"Stationery","sale_date":"2025-01-09","units_sold":50,"price_per_unit":2.
99,"region":3};
```

ПРИМЕЧАНИЕ

Данные в таблицу ADQM/ClickHouse также могут быть импортированы из файлов различных форматов — см. Импорт данных из файлов.

Шаг 5. Чтение данных из таблицы

Для чтения данных из таблицы используйте запрос SELECT. Основные особенности построения запроса:

- Сразу после выражения `SELECT` необходимо указать список столбцов для извлечения данных (перечислить имена столбцов или указать символ `*`, чтобы получить данные из всех столбцов) и/или выражений, которые нужно вычислить по набору строк (например, с помощью агрегатных или оконных функций). При формировании списка столбцов/выражений можно использовать модификаторы `APPLY`, `EXCEPT`, `REPLACE` и их комбинации.
- После ключевого слова `FROM` указывается имя таблицы, из которой нужно получить данные. Чтобы считать данные из двух таблиц и объединить их в один результирующий набор, используйте оператор JOIN.
  Источником для чтения данных также может быть подзапрос или табличная функция.
- Условия фильтрации строк для чтения описываются в блоке `WHERE`.
- Для дополнительной обработки считываемых данных (сортировки, группировки) можно применять выражения `ORDER BY`, `GROUP BY`, `HAVING` и другие.

Примеры различных `SELECT`-запросов приведены ниже.

Чтение всех данных

Следующий запрос считывает все данные из таблицы `sales`:

```
SELECT * FROM sales;
```

В этом примере используется символ `*` вместо названий столбцов и не определено условие `WHERE`, поэтому в выборку попадают все столбцы и строки таблицы:

```
    ┌─id─┬─product_name─┬─category────┬──sale_date─┬─units_sold─┬─price_per_unit─┬─region────────┐
 1. │  1 │ Laptop       │ Electronics │ 2025-01-03 │          5 │         999.99 │ North America │
 2. │  2 │ Smartphone   │ Electronics │ 2025-01-04 │         10 │         699.99 │ Europe        │
 3. │  3 │ Headphones   │ Electronics │ 2025-01-04 │         25 │          49.99 │ Asia          │
 4. │  4 │ Desk Chair   │ Furniture   │ 2025-01-05 │          3 │         149.99 │ North America │
 5. │  5 │ Coffee Table │ Furniture   │ 2025-01-05 │          2 │         249.99 │ Europe        │
 6. │  6 │ Notebook     │ Stationery  │ 2025-01-06 │        100 │           1.99 │ Asia          │
 7. │  7 │ Pen          │ Stationery  │ 2025-01-06 │        200 │           0.49 │ North America │
 8. │  8 │ Water Bottle │ Accessories │ 2025-01-07 │         15 │           9.99 │ Europe        │
 9. │  9 │ Backpack     │ Accessories │ 2025-01-07 │          8 │          59.99 │ Asia          │
10. │ 10 │ Monitor      │ Electronics │ 2025-01-08 │          6 │         199.99 │ North America │
11. │ 11 │ Keyboard     │ Electronics │ 2025-01-08 │         12 │          49.99 │ Europe        │
12. │ 12 │ Office Chair │ Furniture   │ 2025-01-16 │          4 │         199.99 │ Asia          │
13. │ 13 │ Printer      │ Electronics │ 2025-01-16 │          2 │         299.99 │ North America │
14. │ 14 │ Cap          │ Accessories │ 2025-01-09 │         30 │           5.99 │ Europe        │
15. │ 15 │ Binder       │ Stationery  │ 2025-01-09 │         50 │           2.99 │ Asia          │
    └────┴──────────────┴─────────────┴────────────┴────────────┴────────────────┴───────────────┘
```

Фильтрация данных

Следующий запрос возвращает данные только о продажах за 7 января 2025 года:

```
SELECT
    product_name,
    sale_date,
    units_sold
FROM sales
WHERE sale_date = '2025-01-07';
```

```
   ┌─product_name─┬──sale_date─┬─units_sold─┐
1. │ Water Bottle │ 2025-01-07 │         15 │
2. │ Backpack     │ 2025-01-07 │          8 │
   └──────────────┴────────────┴────────────┘
```

Сортировка результатов

Следующий запрос возвращает из таблицы `sales` 10 самых крупных продаж в порядке убывания:

```
SELECT
    sale_date,
    product_name,
    units_sold,
    price_per_unit,
    round(units_sold * price_per_unit, 2) AS revenue
FROM sales
ORDER BY revenue DESC
LIMIT 10;
```

```
    ┌──sale_date─┬─product_name─┬─units_sold─┬─price_per_unit─┬─revenue─┐
 1. │ 2025-01-04 │ Smartphone   │         10 │         699.99 │  6999.9 │
 2. │ 2025-01-03 │ Laptop       │          5 │         999.99 │ 4999.95 │
 3. │ 2025-01-04 │ Headphones   │         25 │          49.99 │ 1249.75 │
 4. │ 2025-01-08 │ Monitor      │          6 │         199.99 │ 1199.94 │
 5. │ 2025-01-16 │ Office Chair │          4 │         199.99 │  799.96 │
 6. │ 2025-01-16 │ Printer      │          2 │         299.99 │  599.98 │
 7. │ 2025-01-08 │ Keyboard     │         12 │          49.99 │  599.88 │
 8. │ 2025-01-05 │ Coffee Table │          2 │         249.99 │  499.98 │
 9. │ 2025-01-07 │ Backpack     │          8 │          59.99 │  479.92 │
10. │ 2025-01-05 │ Desk Chair   │          3 │         149.99 │  449.97 │
    └────────────┴──────────────┴────────────┴────────────────┴─────────┘
```

Группировка результатов

Следующий запрос группирует записи о продажах в таблице `sales` по категории товаров и находит товар с максимальной ценой за единицу в каждой категории:

```
SELECT
    category,
    argMax(product_name, price_per_unit) AS most_expensive_product,
    max(price_per_unit) AS highest_price
FROM sales
GROUP BY category
ORDER BY highest_price DESC;
```

```
   ┌─category────┬─most_expensive_product─┬─highest_price─┐
1. │ Electronics │ Laptop                 │        999.99 │
2. │ Furniture   │ Coffee Table           │        249.99 │
3. │ Accessories │ Backpack               │         59.99 │
4. │ Stationery  │ Binder                 │          2.99 │
   └─────────────┴────────────────────────┴───────────────┘
```

Следующий запрос возвращает ежедневную выручку в каждом регионе:

```
SELECT
    sale_date,
    region,
    round(SUM(units_sold * price_per_unit), 2) AS daily_revenue
FROM sales
GROUP BY sale_date, region
ORDER BY sale_date, region;
```

```
    ┌──sale_date─┬─region────────┬─daily_revenue─┐
 1. │ 2025-01-03 │ North America │       4999.95 │
 2. │ 2025-01-04 │ Europe        │        6999.9 │
 3. │ 2025-01-04 │ Asia          │       1249.75 │
 4. │ 2025-01-05 │ North America │        449.97 │
 5. │ 2025-01-05 │ Europe        │        499.98 │
 6. │ 2025-01-06 │ North America │            98 │
 7. │ 2025-01-06 │ Asia          │           199 │
 8. │ 2025-01-07 │ Europe        │        149.85 │
 9. │ 2025-01-07 │ Asia          │        479.92 │
10. │ 2025-01-08 │ North America │       1199.94 │
11. │ 2025-01-08 │ Europe        │        599.88 │
12. │ 2025-01-09 │ Europe        │         179.7 │
13. │ 2025-01-09 │ Asia          │         149.5 │
14. │ 2025-01-14 │ North America │        599.98 │
15. │ 2025-01-14 │ Asia          │        799.96 │
    └────────────┴───────────────┴───────────────┘
```

Комбинация SELECT с CREATE и INSERT INTO

Команду `SELECT` можно использовать для копирования данных из одной таблицы в другую — в комбинации с командами `CREATE` и `INSERT INTO`.

Приведенный ниже запрос создает новую таблицу `sales_copy` с такой же структурой, как у таблицы `sales`, а также копирует строки, соответствующие продажам до 7 января 2025 года включительно, из исходной таблицы в новую, используя команду `SELECT`:

```
CREATE TABLE sales_copy
ENGINE = MergeTree
ORDER BY id
AS SELECT * FROM sales
WHERE sale_date <= '2025-01-07';
```

Следующий запрос копирует оставшиеся строки (продажи после 7 января 2025 года) из таблицы `sales` в `sales_copy` с помощью команды `INSERT INTO`:

```
INSERT INTO sales_copy SELECT * FROM sales WHERE sale_date > '2025-01-07';
```

Чтобы убедиться, что содержимое двух таблиц идентично, сделайте выборку из обеих таблиц:

```
SELECT * FROM sales;
```

```
    ┌─id─┬─product_name─┬─category────┬──sale_date─┬─units_sold─┬─price_per_unit─┬─region────────┐
 1. │  1 │ Laptop       │ Electronics │ 2025-01-03 │          5 │         999.99 │ North America │
 2. │  2 │ Smartphone   │ Electronics │ 2025-01-04 │         10 │         699.99 │ Europe        │
 3. │  3 │ Headphones   │ Electronics │ 2025-01-04 │         25 │          49.99 │ Asia          │
 4. │  4 │ Desk Chair   │ Furniture   │ 2025-01-05 │          3 │         149.99 │ North America │
 5. │  5 │ Coffee Table │ Furniture   │ 2025-01-05 │          2 │         249.99 │ Europe        │
 6. │  6 │ Notebook     │ Stationery  │ 2025-01-06 │        100 │           1.99 │ Asia          │
 7. │  7 │ Pen          │ Stationery  │ 2025-01-06 │        200 │           0.49 │ North America │
 8. │  8 │ Water Bottle │ Accessories │ 2025-01-07 │         15 │           9.99 │ Europe        │
 9. │  9 │ Backpack     │ Accessories │ 2025-01-07 │          8 │          59.99 │ Asia          │
10. │ 10 │ Monitor      │ Electronics │ 2025-01-08 │          6 │         199.99 │ North America │
11. │ 11 │ Keyboard     │ Electronics │ 2025-01-08 │         12 │          49.99 │ Europe        │
12. │ 12 │ Office Chair │ Furniture   │ 2025-01-16 │          4 │         199.99 │ Asia          │
13. │ 13 │ Printer      │ Electronics │ 2025-01-16 │          2 │         299.99 │ North America │
14. │ 14 │ Cap          │ Accessories │ 2025-01-09 │         30 │           5.99 │ Europe        │
15. │ 15 │ Binder       │ Stationery  │ 2025-01-09 │         50 │           2.99 │ Asia          │
    └────┴──────────────┴─────────────┴────────────┴────────────┴────────────────┴───────────────┘
```

```
SELECT * FROM sales_copy;
```

```
    ┌─id─┬─product_name─┬─category────┬──sale_date─┬─units_sold─┬─price_per_unit─┬─region────────┐
 1. │  1 │ Laptop       │ Electronics │ 2025-01-03 │          5 │         999.99 │ North America │
 2. │  2 │ Smartphone   │ Electronics │ 2025-01-04 │         10 │         699.99 │ Europe        │
 3. │  3 │ Headphones   │ Electronics │ 2025-01-04 │         25 │          49.99 │ Asia          │
 4. │  4 │ Desk Chair   │ Furniture   │ 2025-01-05 │          3 │         149.99 │ North America │
 5. │  5 │ Coffee Table │ Furniture   │ 2025-01-05 │          2 │         249.99 │ Europe        │
 6. │  6 │ Notebook     │ Stationery  │ 2025-01-06 │        100 │           1.99 │ Asia          │
 7. │  7 │ Pen          │ Stationery  │ 2025-01-06 │        200 │           0.49 │ North America │
 8. │  8 │ Water Bottle │ Accessories │ 2025-01-07 │         15 │           9.99 │ Europe        │
 9. │  9 │ Backpack     │ Accessories │ 2025-01-07 │          8 │          59.99 │ Asia          │
10. │ 10 │ Monitor      │ Electronics │ 2025-01-08 │          6 │         199.99 │ North America │
11. │ 11 │ Keyboard     │ Electronics │ 2025-01-08 │         12 │          49.99 │ Europe        │
12. │ 12 │ Office Chair │ Furniture   │ 2025-01-16 │          4 │         199.99 │ Asia          │
13. │ 13 │ Printer      │ Electronics │ 2025-01-16 │          2 │         299.99 │ North America │
14. │ 14 │ Cap          │ Accessories │ 2025-01-09 │         30 │           5.99 │ Europe        │
15. │ 15 │ Binder       │ Stationery  │ 2025-01-09 │         50 │           2.99 │ Asia          │
    └────┴──────────────┴─────────────┴────────────┴────────────┴────────────────┴───────────────┘
```

Шаг 6. Обновление данных в таблице

Обновить данные в таблице можно с помощью запроса ALTER TABLE…UPDATE, где после ключевого слова `UPDATE` указывается список новых значений столбцов (или выражений для вычисления значений), а после `WHERE` — условие отбора строк, в которых нужно заменить значения:

```
ALTER TABLE <table_name> UPDATE <column_name> = <new_value>[, ...] WHERE <filter_expr>;
```

Например, следующий запрос увеличивает на 2 процента цену каждого товара категории `Accessories` в таблице `sales_copy`:

```
ALTER TABLE sales_copy
UPDATE price_per_unit = round(price_per_unit*1.02, 2)
WHERE category = 'Accessories';
```

Чтобы проверить результат выполнения запроса, можно вывести данные по товарам (название и цену) в категории `Accessories`, используя команду `SELECT`:

```
SELECT product_name, price_per_unit FROM sales_copy WHERE category='Accessories';
```

```
   ┌─product_name─┬─price_per_unit─┐
1. │ Water Bottle │          10.19 │
2. │ Backpack     │          61.19 │
3. │ Cap          │           6.11 │
   └──────────────┴────────────────┘
```

Шаг 7. Удаление данных из таблицы

Для удаления строк данных из таблицы по указанному условию можно использовать запрос `ALTER TABLE…DELETE` (для любых таблиц) или `DELETE FROM` (для таблиц MergeTree) — подробнее в разделе Удаление данных.

Следующий запрос удаляет данные о продажах до 5 января 2025 года включительно из таблицы `sales_copy`:

```
ALTER TABLE sales_copy DELETE WHERE sale_date <= '2025-01-05';
```

Выполните запрос `SELECT`, чтобы убедиться, что в таблице остались только данные о продажах после 5 января 2025 года:

```
    ┌─id─┬─product_name─┬─category────┬──sale_date─┬─units_sold─┬─price_per_unit─┬─region────────┐
 1. │  6 │ Notebook     │ Stationery  │ 2025-01-06 │        100 │           1.99 │ Asia          │
 2. │  7 │ Pen          │ Stationery  │ 2025-01-06 │        200 │           0.49 │ North America │
 3. │  8 │ Water Bottle │ Accessories │ 2025-01-07 │         15 │          10.19 │ Europe        │
 4. │  9 │ Backpack     │ Accessories │ 2025-01-07 │          8 │          61.19 │ Asia          │
 5. │ 10 │ Monitor      │ Electronics │ 2025-01-08 │          6 │         199.99 │ North America │
 6. │ 11 │ Keyboard     │ Electronics │ 2025-01-08 │         12 │          49.99 │ Europe        │
 7. │ 12 │ Office Chair │ Furniture   │ 2025-01-16 │          4 │         199.99 │ Asia          │
 8. │ 13 │ Printer      │ Electronics │ 2025-01-16 │          2 │         299.99 │ North America │
 9. │ 14 │ Cap          │ Accessories │ 2025-01-09 │         30 │           6.11 │ Europe        │
10. │ 15 │ Binder       │ Stationery  │ 2025-01-09 │         50 │           2.99 │ Asia          │
    └────┴──────────────┴─────────────┴────────────┴────────────┴────────────────┴───────────────┘
```

Следующий запрос удаляет из таблицы `sales_copy` строку о продаже продукта `Keyboard` в регионе `Europe`:

```
DELETE FROM sales_copy WHERE product_name = 'Keyboard' AND region = 'Europe';
```

В результате выполнения запроса из таблицы успешно удалена соответствующая строка:

```
   ┌─id─┬─product_name─┬─category────┬──sale_date─┬─units_sold─┬─price_per_unit─┬─region────────┐
1. │  6 │ Notebook     │ Stationery  │ 2025-01-06 │        100 │           1.99 │ Asia          │
2. │  7 │ Pen          │ Stationery  │ 2025-01-06 │        200 │           0.49 │ North America │
3. │  8 │ Water Bottle │ Accessories │ 2025-01-07 │         15 │          10.19 │ Europe        │
4. │  9 │ Backpack     │ Accessories │ 2025-01-07 │          8 │          61.19 │ Asia          │
5. │ 10 │ Monitor      │ Electronics │ 2025-01-08 │          6 │         199.99 │ North America │
6. │ 12 │ Office Chair │ Furniture   │ 2025-01-16 │          4 │         199.99 │ Asia          │
7. │ 13 │ Printer      │ Electronics │ 2025-01-16 │          2 │         299.99 │ North America │
8. │ 14 │ Cap          │ Accessories │ 2025-01-09 │         30 │           6.11 │ Europe        │
9. │ 15 │ Binder       │ Stationery  │ 2025-01-09 │         50 │           2.99 │ Asia          │
   └────┴──────────────┴─────────────┴────────────┴────────────┴────────────────┴───────────────┘
```

Шаг 8. Изменение структуры таблицы

Для изменения структуры (столбцов) таблицы используйте запрос `ALTER TABLE` (поддерживается табличными движками MergeTree, а также Merge и Distributed):

```
ALTER TABLE <table_name> ADD|DROP|RENAME|CLEAR|COMMENT|{MODIFY|ALTER}|MATERIALIZE COLUMN <column_name> ...
```

В рамках одного запроса можно указать сразу несколько действий над столбцами таблицы через запятую.

Следующий запрос изменяет имя столбца `product_name` на `product` и добавляет новый столбец `store_id`:

```
ALTER TABLE sales_copy RENAME COLUMN product_name TO product, ADD COLUMN store_id UInt8;
```

Чтобы убедиться, что столбец переименован и новый столбец `store_id` добавлен, можно выполнить запрос `DESCRIBE TABLE`:

```
DESCRIBE TABLE sales_copy;
```

```
   ┌─name───────────┬─type──────────────────────────────────────────────────┐
1. │ id             │ UInt32                                                │
2. │ product        │ String                                                │
3. │ category       │ String                                                │
4. │ sale_date      │ Date                                                  │
5. │ units_sold     │ UInt32                                                │
6. │ price_per_unit │ Float32                                               │
7. │ region         │ Enum8('North America' = 1, 'Europe' = 2, 'Asia' = 3) │
8. │ store_id       │ UInt8                                                 │
   └────────────────┴───────────────────────────────────────────────────────┘
```

ПРИМЕЧАНИЕ

С помощью запроса `ALTER TABLE` можно также выполнять различные манипуляции с партициями, кусками данных, ограничениями (constraints), индексами, проекциями таблиц, добавлять к таблицам комментарии, устанавливать правила TTL — см. подробное описание в соответствующих статьях секции ALTER документации ClickHouse.

Шаг 9. Отключение и подключение таблицы

С помощью запроса `DETACH TABLE` таблицу можно отключить, то есть сделать "невидимой" для сервера (при этом ее данные и метаданные не удаляются).

Следующий запрос отключает таблицу `sales_copy`:

```
DETACH TABLE sales_copy;
```

Убедитесь, что таблицы `sales_copy` нет в списке таблиц базы данных:

```
SHOW tables;
```

```
   ┌─name──┐
1. │ sales │
   └───────┘
```

При следующем запуске сервер прочитает метаданные таблицы и снова ее "увидит". Чтобы сервер при перезапуске не подключал таблицу автоматически, отключите ее перманентно — для этого используйте в конце запроса `DETACH TABLE` ключевое слово `PERMANENTLY`.

Чтобы подключить таблицу обратно (независимо от того, каким способом она была отключена — перманентно или нет), выполните запрос `ATTACH TABLE`:

```
ATTACH TABLE sales_copy;
```

Проверьте, что сервер ClickHouse снова "видит" таблицу `sales_copy`:

```
SHOW tables;
```

```
   ┌─name───────┐
1. │ sales      │
2. │ sales_copy │
   └────────────┘
```

Шаг 10. Удаление таблицы

Чтобы удалить таблицу, используйте запрос `DROP TABLE`:

```
DROP TABLE sales_copy;
```

Таблица успешно удалена из базы данных:

```
SHOW tables;
```

```
   ┌─name──┐
1. │ sales │
   └───────┘
```

Оператор JOIN

Елена Дворядкина

Содержание

Назад к содержанию

Обзор

Типы JOIN

- INNER JOIN
- OUTER JOIN
- CROSS JOIN
- LEFT/RIGHT SEMI JOIN
- LEFT/RIGHT ANTI JOIN
- LEFT/RIGHT/INNER ANY JOIN
- ASOF JOIN

Условие JOIN

- ON
- USING

Обзор

Оператор `JOIN` обеспечивает выборку данных из двух таблиц и объединяет эти данные в один результирующий набор. То, какие строки исходных таблиц будут включены в результат и в каких комбинациях, зависит от типа `JOIN` и явно указанного условия соединения.

В общем виде синтаксис `JOIN` можно записать следующим образом:

```
FROM <left_table> <JOIN_type> <right_table> <join_condition>
```

где:

- `<JOIN_type>` — тип соединения. ADQM поддерживает все стандартные типы JOIN языка SQL, а также дополнительные специализированные типы соединения, которые могут быть полезны для выполнения различных аналитических запросов и анализа временных рядов.

  Стандартные типы JOIN языка SQL:

  - INNER JOIN
  - LEFT/RIGHT/FULL OUTER JOIN
  - CROSS JOIN

  Дополнительные типы JOIN, доступные в ADQM:

  - LEFT/RIGHT SEMI JOIN
  - LEFT/RIGHT ANTI JOIN
  - LEFT/RIGHT/INNER ANY JOIN
  - ASOF JOIN

  Если тип JOIN явно не указан, применяется `INNER JOIN`.

- `<join_condition>` — условие, по которому определяется, какие строки двух исходных таблиц соответствуют друг другу. Условие соединения (или несколько условий) должно быть указано с помощью выражения `ON <expr_list>` или `USING <column_list>` для всех типов `JOIN` кроме `CROSS JOIN` — подробнее в разделе Условие JOIN.

Настройка join_algorithm позволяет установить алгоритм выполнения запроса `JOIN`. Можно указать несколько алгоритмов (тогда один из доступных алгоритмов будет выбираться для каждого конкретного запроса на основе его типа/строгости и движка таблицы) или разрешить системе выбирать и динамически изменять алгоритм во время выполнения запроса в зависимости от доступности ресурсов.

Создание таблиц для тестовых примеров

Чтобы воспроизвести приведенные ниже примеры и посмотреть, как различные типы `JOIN` работают с данными, создайте предварительно две таблицы:

- `users`

```
CREATE TABLE users (user_id Int32, name String, role_id Int32) ENGINE = MergeTree ORDER BY user_id;
```

```
INSERT INTO users VALUES (1, 'john', 10), (1, 'john', 20), (1, 'john', 30), (2, 'mary', 30), (2, 'mary', 40), (3, 'andrew',
40), (4, 'harry', 70), (5, 'ann', 50);
```

```
┌─user_id─┬─name───┬─role_id─┐
│       1 │ john   │      10 │
│       2 │ mike   │      20 │
│       3 │ tom    │      30 │
│       4 │ mary   │      30 │
│       5 │ ada    │      40 │
│       6 │ andrew │      40 │
│       7 │ harry  │      70 │
│       8 │ ann    │      50 │
└─────────┴────────┴─────────┘
```

- `roles`

```
CREATE TABLE roles (id Int32, title String) ENGINE = MergeTree ORDER BY id;
```

```
INSERT INTO roles VALUES (10, 'admin'), (20, 'owner'), (30, 'author'), (40, 'reviewer'), (50, 'editor'), (60, 'view only');
```

```
┌─id─┬─title─────┐
│ 10 │ admin     │
│ 20 │ owner     │
│ 30 │ author    │
│ 40 │ reviewer  │
│ 50 │ editor    │
│ 60 │ view only │
└────┴───────────┘
```

Таблица `users` содержит поле `role_id`. Это поле будет ключом, который ссылается на поле `id` в таблице `roles` и служит для определения связей между таблицами.

Типы JOIN

INNER JOIN

Для каждой строки в левой таблице проверяется, содержит ли правая таблица строку, которая удовлетворяет условию `JOIN`. Если соответствующая строка найдена, создается новая строка, содержащая столбцы из обеих таблиц, и добавляется в набор результатов. Если для строки найдено более одного совпадения по указанному условию, то в результирующую таблицу записываются все совпадения (создается декартово произведение для строк, совпадающих по условию соединения).

Ключевое слово `INNER` в запросе можно опустить.

Пример

```
SELECT users.name AS user, roles.title AS role FROM users INNER JOIN roles ON users.role_id=roles.id;
```

Запрос соединяет таблицы `users` и `roles` и возвращает строки, для которых выполняется условие соединения по идентификатору роли, назначенной пользователю. Пользователь `harry` не включен в результат, так как для идентификатора назначенной ему роли не найдена соответствующая строка в таблице `roles`. В результат также не входит роль `view only`, так как она не назначена ни одному пользователю.

```
┌─user───┬─role─────┐
│ john   │ admin    │
│ mike   │ owner    │
│ tom    │ author   │
│ mary   │ author   │
│ ada    │ reviewer │
│ andrew │ reviewer │
│ ann    │ editor   │
└────────┴──────────┘
```

OUTER JOIN

Запросы `LEFT/RIGHT/FULL OUTER JOIN` позволяют расширить `INNER JOIN`, добавляя в результаты выборки строки из исходных таблиц, не имеющие соответствий по указанному условию соединения. Ключевое слово `OUTER` в запросах можно опускать.

- `LEFT OUTER JOIN` сначала выполняет операцию `INNER JOIN`. Затем каждая строка левой таблицы, которая не вошла в результат `INNER JOIN`, добавляется к результату `LEFT OUTER JOIN` со значениями по умолчанию для соответствующих типов данных в столбцах правой таблицы. Таким образом, в объединенной таблице всегда есть по крайней мере одна строка для каждой строки левой таблицы.

  Пример

```
SELECT users.name AS user, roles.title AS role FROM users LEFT OUTER JOIN roles ON users.role_id=roles.id;
```

  Результат содержит названия ролей для всех пользователей из таблицы `users`, в том числе включает строку с пользователем `harry`, роль которого не была найдена в таблице `roles` по идентификатору роли. Столбец `roles.title` с псевдонимом `role` для этой строки содержит пустую строку — значение по умолчанию для типа `String`.

```
┌─user───┬─role─────┐
│ john   │ admin    │
│ mike   │ owner    │
│ tom    │ author   │
│ mary   │ author   │
│ ada    │ reviewer │
│ andrew │ reviewer │
│ harry  │          │
│ ann    │ editor   │
└────────┴──────────┘
```

- `RIGHT OUTER JOIN` работает аналогичным образом, но добавляет к результату `INNER JOIN` строки из правой таблицы, которые не вошли в результат `INNER JOIN`, со значениями по умолчанию в столбцах левой таблицы. В объединенной таблице всегда есть по крайней мере одна строка для каждой строки правой таблицы.

  Пример

```
SELECT users.name AS user, roles.title AS role FROM users RIGHT OUTER JOIN roles ON users.role_id=roles.id;
```

  Запрос возвращает все совпадения строк таблиц `users` и `roles` по идентификатору роли, а также строку для роли `view only`, которая не назначена ни одному пользователю (в столбце `user` пустая строка):

```
┌─user───┬─role─────┐
│ john   │ admin    │
│ mike   │ owner    │
│ tom    │ author   │
│ mary   │ author   │
│ ada    │ reviewer │
│ andrew │ reviewer │
│ ann    │ editor   │
└────────┴──────────┘
┌─user─┬─role──────┐
│      │ view only │
└──────┴───────────┘
```

- `FULL OUTER JOIN` комбинирует `LEFT OUTER JOIN` и `RIGHT OUTER JOIN` — возвращает результат `INNER JOIN`, а также значения несовпадающих строк обеих таблиц, заполняя пустые ячейки значениями по умолчанию.

  Пример

```
SELECT users.name AS user, roles.title AS role FROM users FULL OUTER JOIN roles ON users.role_id=roles.id;
```

  В результат включены все пользователи из таблицы `users` (в том числе пользователь `harry`, роль которого не определена в таблице `roles`), а также роль `view only` из таблицы `roles`, которая не назначена ни одному пользователю:

```
┌─user───┬─role─────┐
│ john   │ admin    │
│ mike   │ owner    │
│ tom    │ author   │
│ mary   │ author   │
│ ada    │ reviewer │
│ andrew │ reviewer │
│ harry  │          │
│ ann    │ editor   │
└────────┴──────────┘
┌─user─┬─role──────┐
│      │ view only │
└──────┴───────────┘
```

Чтобы вместо значений по умолчанию в пустых ячейках использовать значения NULL (как в стандартном SQL), нужно изменить значение настройки join_use_nulls.

CROSS JOIN

Запрос `CROSS JOIN` комбинирует каждую строку левой таблицы с каждой строкой правой таблицы (создает декартово произведение). Набор столбцов объединенной таблицы включает все столбцы левой таблицы, за которыми следуют все столбцы правой таблицы. Если в таблицах есть `N` и `M` строк соответственно, то в объединенной таблице будет `N * M` строк.

`CROSS JOIN` эквивалентен `FROM <left_table> INNER JOIN <right_table> ON TRUE` или `FROM <left_table>, <right_table>`.

Пример

```
SELECT * FROM users CROSS JOIN roles;
```

Результирующая таблица содержит все возможные комбинации строк из таблиц `users` и `roles`:

```
┌─user_id─┬─name───┬─role_id─┬─id─┬─title─────┐
│       1 │ john   │      10 │ 10 │ admin     │
│       1 │ john   │      10 │ 20 │ owner     │
│       1 │ john   │      10 │ 30 │ author    │
│       1 │ john   │      10 │ 40 │ reviewer  │
│       1 │ john   │      10 │ 50 │ editor    │
│       1 │ john   │      10 │ 60 │ view only │
│       2 │ mike   │      20 │ 10 │ admin     │
│       2 │ mike   │      20 │ 20 │ owner     │
│       2 │ mike   │      20 │ 30 │ author    │
│       2 │ mike   │      20 │ 40 │ reviewer  │
│       2 │ mike   │      20 │ 50 │ editor    │
│       2 │ mike   │      20 │ 60 │ view only │
│       3 │ tom    │      30 │ 10 │ admin     │
│       3 │ tom    │      30 │ 20 │ owner     │
│       3 │ tom    │      30 │ 30 │ author    │
│       3 │ tom    │      30 │ 40 │ reviewer  │
│       3 │ tom    │      30 │ 50 │ editor    │
│       3 │ tom    │      30 │ 60 │ view only │
│       4 │ mary   │      30 │ 10 │ admin     │
│       4 │ mary   │      30 │ 20 │ owner     │
│       4 │ mary   │      30 │ 30 │ author    │
│       4 │ mary   │      30 │ 40 │ reviewer  │
│       4 │ mary   │      30 │ 50 │ editor    │
│       4 │ mary   │      30 │ 60 │ view only │
│       5 │ ada    │      40 │ 10 │ admin     │
│       5 │ ada    │      40 │ 20 │ owner     │
│       5 │ ada    │      40 │ 30 │ author    │
│       5 │ ada    │      40 │ 40 │ reviewer  │
│       5 │ ada    │      40 │ 50 │ editor    │
│       5 │ ada    │      40 │ 60 │ view only │
│       6 │ andrew │      40 │ 10 │ admin     │
│       6 │ andrew │      40 │ 20 │ owner     │
│       6 │ andrew │      40 │ 30 │ author    │
│       6 │ andrew │      40 │ 40 │ reviewer  │
│       6 │ andrew │      40 │ 50 │ editor    │
│       6 │ andrew │      40 │ 60 │ view only │
│       7 │ harry  │      70 │ 10 │ admin     │
│       7 │ harry  │      70 │ 20 │ owner     │
│       7 │ harry  │      70 │ 30 │ author    │
│       7 │ harry  │      70 │ 40 │ reviewer  │
│       7 │ harry  │      70 │ 50 │ editor    │
│       7 │ harry  │      70 │ 60 │ view only │
│       8 │ ann    │      50 │ 10 │ admin     │
│       8 │ ann    │      50 │ 20 │ owner     │
│       8 │ ann    │      50 │ 30 │ author    │
│       8 │ ann    │      50 │ 40 │ reviewer  │
│       8 │ ann    │      50 │ 50 │ editor    │
│       8 │ ann    │      50 │ 60 │ view only │
└─────────┴────────┴─────────┴────┴───────────┘
```

LEFT/RIGHT SEMI JOIN

Запрос `LEFT SEMI JOIN` возвращает значения столбцов для каждой строки из левой таблицы, у которой есть хотя бы одно совпадение по условию соединения в правой таблице. Если для строки найдено более одного совпадения, возвращается только первое найденное совпадение (декартово произведение отключено).

`RIGHT SEMI JOIN` работает аналогично — возвращает значения столбцов для всех строк из правой таблицы, которые имеют хотя бы одно совпадение по условию соединения с левой таблицей. Если для строки найдено несколько совпадений, в результат добавляется только первое найденное совпадение.

Пример RIGHT SEMI JOIN

```
SELECT roles.title AS role, users.name AS user FROM users RIGHT SEMI JOIN roles ON users.role_id=roles.id;
```

В результате выполнения запроса `RIGHT SEMI JOIN` для каждой роли из правой таблицы `roles` возвращается только один пользователь, которому эта роль назначена. Роль `view only`, для которой не найдено ни одного совпадения в таблице `users`, не включается в результирующую таблицу:

```
┌─role─────┬─user─┐
│ admin    │ john │
│ owner    │ mike │
│ author   │ tom  │
│ reviewer │ ada  │
│ editor   │ ann  │
└──────────┴──────┘
```

LEFT/RIGHT ANTI JOIN

`LEFT ANTI JOIN` возвращает значения столбцов для всех строк левой таблицы, не имеющих совпадений со строками правой таблицы по условию соединения. Аналогично работает `RIGHT ANTI JOIN` — возвращает значения столбцов для строк правой таблицы, для которых не найдены соответствия в левой таблице.

Пример LEFT ANTI JOIN

```
SELECT users.name AS user, roles.title AS role FROM users LEFT ANTI JOIN roles ON users.role_id=roles.id;
```

В результате выполнения запроса `LEFT ANTI JOIN` возвращаются данные только для пользователя `harry`, роль которого не найдена в таблице `roles` по указанному идентификатору:

```
┌─user──┬─role─┐
│ harry │      │
└───────┴──────┘
```

LEFT/RIGHT/INNER ANY JOIN

Запрос `LEFT ANY JOIN` возвращает значения столбцов каждой строки левой таблицы, соединяя их либо со значениями столбцов соответствующей по условию соединения строки правой таблицы, либо со значениями по умолчанию в столбцах правой таблицы (если совпадения по условию нет). Если строка левой таблицы имеет более одного совпадения с правой таблицей, возвращаются значения столбцов только для первого найденного совпадения. Запрос `RIGHT ANY JOIN` работает по аналогии для строк правой таблицы.

Иными словами, ключевое слово `ANY` указывает на строгость операции `JOIN` — декартово произведение для совпадающих по условию соединения строк исходных таблиц не создается, то есть в результат попадает только одно из найденных совпадений для каждой строки. По умолчанию применяется строгость `ALL` — все совпадающие строки исходных таблиц соединяются и записываются в результирующую таблицу (см. примеры запросов LEFT/RIGHT OUTER JOIN и INNER JOIN выше). То есть `LEFT/RIGHT ANY JOIN` соединяет таблицы как `LEFT/RIGHT OUTER JOIN`, а `INNER ANY JOIN` как `INNER JOIN`, но с отключенным декартовым произведением.

Значение строгости по умолчанию можно регулировать с помощью настройки join_default_strictness.

Пример RIGHT ANY JOIN

```
SELECT users.name AS user, roles.title AS role FROM users RIGHT ANY JOIN roles ON users.role_id=roles.id;
```

Запрос `RIGHT ANY JOIN` возвращает для каждой роли из правой таблицы `roles` только одного пользователя (первого найденного), которому эта роль назначена, а также роль `view only`, которая не назначена ни одному пользователю:

```
┌─user─┬─role─────┐
│ john │ admin    │
│ mike │ owner    │
│ tom  │ author   │
│ ada  │ reviewer │
│ ann  │ editor   │
└──────┴──────────┘
┌─user─┬─role──────┐
│      │ view only │
└──────┴───────────┘
```

ASOF JOIN

Запросы `ASOF JOIN` и `LEFT ASOF JOIN` применяются, когда необходимо объединять записи, которые не имеют точного совпадения. Если строка из левой таблицы не имеет точного соответствия с какой-либо строкой в правой таблице, то в качестве соответствующей принимается строка правой таблицы, наиболее близкая по дополнительному условию — условию ближайшего совпадения, которое указывается на основе специального столбца (типа Int, UInt, Float, Date, DateTime или Decimal) в таблицах.

Синтаксис запроса в общем виде:

- `ASOF JOIN … ON`:

```
<left_table> [LEFT] ASOF JOIN <right_table> ON <exact_match_condition> AND <closest_match_condition>
```

  где:

  - `<exact_match_condition>` — условие для соединения по равенству (может быть задано несколько условий);
  - `<closest_match_condition>` — условие для соединения по ближайшему совпадению (должно быть задано одно такое условие). Условия для проверки на ближайшее совпадение поддерживают операторы `>`, `>=`, `<`, `<=`.

- `ASOF JOIN … USING`:

```
<left_table> [LEFT] ASOF JOIN <right_table> USING <exact_match_column1>, ... <exact_match_columnN>, <closest_match_column>
```

  где

  - `<exact_match_column1>, … <exact_match_columnN>` — столбцы для соединения по равенству значений;
  - `<closest_match_column>` — столбец (последний в списке `USING`) для соединения по ближайшему совпадению. Условие соединения: `<left_table>.<closest_match_column> >= <right_table>.<closest_match_column>`.

Пример LEFT ASOF JOIN

Создайте две таблицы для тестовых примеров:

- `table1`

```
CREATE TABLE table1 (id Int32, time DateTime, text String) ENGINE = MergeTree ORDER BY id;
```

```
INSERT INTO table1 VALUES
    (50, '2023-03-10 14:55:00', 'text1_0'),
    (50, '2023-03-10 15:00:00', 'text1_1'),
    (50, '2023-03-10 15:03:00', 'text1_2'),
    (50, '2023-03-10 15:10:00', 'text1_3'),
    (50, '2023-03-10 15:14:00', 'text1_4');
```

```
┌─id─┬────────────────time─┬─text────┐
│ 50 │ 2023-03-10 14:55:00 │ text1_0 │
│ 50 │ 2023-03-10 15:00:00 │ text1_1 │
│ 50 │ 2023-03-10 15:03:00 │ text1_2 │
│ 50 │ 2023-03-10 15:10:00 │ text1_3 │
│ 50 │ 2023-03-10 15:14:00 │ text1_4 │
└────┴─────────────────────┴─────────┘
```

- `table2`

```
CREATE TABLE table2 (id Int32, time DateTime, text String) ENGINE = MergeTree ORDER BY id;
```

```
INSERT INTO table2 VALUES
    (50, '2023-03-10 15:00:00', 'text2_1'),
    (50, '2023-03-10 15:07:00', 'text2_2'),
    (50, '2023-03-10 15:11:00', 'text2_3'),
    (50, '2023-03-10 15:20:00', 'text2_4');
```

```
┌─id─┬────────────────time─┬─text────┐
│ 50 │ 2023-03-10 15:00:00 │ text2_1 │
│ 50 │ 2023-03-10 15:07:00 │ text2_2 │
│ 50 │ 2023-03-10 15:11:00 │ text2_3 │
│ 50 │ 2023-03-10 15:20:00 │ text2_4 │
└────┴─────────────────────┴─────────┘
```

Выполните запрос `ASOF LEFT JOIN`:

```
SELECT
    table1.text AS table1_text,
    table1.time AS table1_time,
    table2.text AS table2_text,
    table2.time AS table2_time
FROM table1
ASOF LEFT JOIN table2 ON (table1.id = table2.id) AND (table1.time >= table2.time);
```

В результирующей выборке каждой строке таблицы `table1` соответствует строка таблицы `table2` с наиболее близким значением в столбце `time` по условию `table1.time >= table2.time`. Для первой строки (`text1_0`) подходящей строки в таблице `table2` не найдено, поэтому в столбцах `table2_text` и `table2_time` содержатся значения по умолчанию для соответствующих типов:

```
┌─table1_text─┬─────────table1_time─┬─table2_text─┬─────────table2_time─┐
│ text1_0     │ 2023-03-10 14:55:00 │             │ 1970-01-01 00:00:00 │
│ text1_1     │ 2023-03-10 15:00:00 │ text2_1     │ 2023-03-10 15:00:00 │
│ text1_2     │ 2023-03-10 15:03:00 │ text2_1     │ 2023-03-10 15:00:00 │
│ text1_3     │ 2023-03-10 15:10:00 │ text2_2     │ 2023-03-10 15:07:00 │
│ text1_4     │ 2023-03-10 15:14:00 │ text2_3     │ 2023-03-10 15:11:00 │
└─────────────┴─────────────────────┴─────────────┴─────────────────────┘
```

Выполните тот же `ASOF LEFT JOIN` запрос с выражением `USING`:

```
SELECT
    table1.text AS table1_text,
    table1.time AS table1_time,
    table2.text AS table2_text,
    table2.time AS table2_time
FROM table1
ASOF LEFT JOIN table2 USING (id, time);
```

В результате получается та же объединенная таблица:

```
┌─table1_text─┬─────────table1_time─┬─table2_text─┬─────────table2_time─┐
│ text1_0     │ 2023-03-10 14:55:00 │             │ 1970-01-01 00:00:00 │
│ text1_1     │ 2023-03-10 15:00:00 │ text2_1     │ 2023-03-10 15:00:00 │
│ text1_2     │ 2023-03-10 15:03:00 │ text2_1     │ 2023-03-10 15:00:00 │
│ text1_3     │ 2023-03-10 15:10:00 │ text2_2     │ 2023-03-10 15:07:00 │
│ text1_4     │ 2023-03-10 15:14:00 │ text2_3     │ 2023-03-10 15:11:00 │
└─────────────┴─────────────────────┴─────────────┴─────────────────────┘
```

Условие JOIN

`JOIN` объединяет строки двух таблиц на основе связующих столбцов между ними — **ключей соединения**. Ключи, а также дополнительные условия для соединения, указываются для оператора `JOIN` с помощью выражений `ON` или `USING`.

В запросах CROSS JOIN условие не указывается.

ON

`ON` возвращает значение boolean. Строка из одной таблицы соответствует строке из другой таблицы, если выражение `ON` возвращает `true`.

Можно указать несколько условий, связанных операторами `AND` и/или `OR` — в этом случае, строки будут объединяться, когда выполнено все составное условие. Если условие не выполняется, то строки все равно могут попасть в результат в зависимости от типа `JOIN`.

Условие, определяющее ключ соединения, задается как сравнение столбцов левой и правой таблиц через оператор равенства (`=`). Прочие условия могут использовать другие логические операторы, и каждое отдельное условие задается только для столбца одной из таблиц.

Пример

В следующем запросе указаны два условия `JOIN`:

- `users.role_id=roles.id` — ключ соединения;
- `roles.id>20` — дополнительное условие для таблицы `roles`.

```
SELECT
    users.name AS user,
    roles.title AS role,
    roles.id AS role_id
FROM users
LEFT JOIN roles ON users.role_id = roles.id AND roles.id > 20;
```

Результат:

```
┌─user───┬─role─────┬─role_id─┐
│ john   │          │       0 │
│ mike   │          │       0 │
│ tom    │ author   │      30 │
│ mary   │ author   │      30 │
│ ada    │ reviewer │      40 │
│ andrew │ reviewer │      40 │
│ harry  │          │       0 │
│ ann    │ editor   │      50 │
└────────┴──────────┴─────────┘
```

USING

Если столбцы, по которым сопоставляются таблицы, имеют одинаковые имена в обеих таблицах, то вместо выражения `ON` можно использовать `USING`. В качестве параметра выражения `USING` можно указать один или несколько столбцов (список имен столбцов, разделенных запятыми), равенство которых будет считаться условием для соединения.

Результат выражения `JOIN` с условием `USING` будет содержать один столбец для каждой из перечисленных пар столбцов в указанном порядке, за которым следуют все оставшиеся столбцы из левой и правой таблиц.

Пример

Создайте таблицы для примера:

- `test_table1`

```
CREATE TABLE test_table1 (num Int32, name String) ENGINE = MergeTree ORDER BY num;
```

```
INSERT INTO test_table1 VALUES (1, 'a'), (2, 'b'), (3, 'c');
```

- `test_table2`

```
CREATE TABLE test_table2 (num Int32, value String) ENGINE = MergeTree ORDER BY num;
```

```
INSERT INTO test_table2 VALUES (0, 'value1'), (1, 'value2'), (2, 'value3');
```

```
  test_table1        test_table2

┌─num─┬─name─┐      ┌─num─┬─value──┐
│   1 │ a    │      │   0 │ value1 │
│   2 │ b    │      │   1 │ value2 │
│   3 │ c    │      │   2 │ value3 │
└─────┴──────┘      └─────┴────────┘
```

Выполните запрос, используя выражение `USING` для оператора `JOIN`, чтобы объединить таблицы `test_table2` и `test_table2` по столбцу `num`:

```
SELECT * FROM test_table1 INNER JOIN test_table2 USING num;
```

Результат:

```
┌─num─┬─name─┬─value──┐
│   1 │ a    │ value2 │
│   2 │ b    │ value3 │
└─────┴──────┴────────┘
```

Подзапросы

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
- Подзапросы в FROM
- Подзапросы в JOIN
- Подзапросы в WHERE
  - IN
  - NOT IN
  - ANY
  - ALL
  - EXISTS

Обзор

Подзапрос — это SQL-запрос, вложенный в другой (родительский) запрос или подзапрос. Сначала выполняется подзапрос, затем полученный результат передается родительскому запросу. Таким образом, подзапросы могут быть полезны, когда для выполнения запроса требуется результат другого запроса. Например, подзапросы можно использовать в запросах `SELECT`: в выражениях `FROM` и `JOIN` для определения источника, из которого должны читаться данные, или в выражении `WHERE` для фильтрации данных.

От типа возвращаемого подзапросом значения зависит то, для каких операций можно использовать подзапрос. Результатом выполнения подзапроса может быть:

- Скалярное значение, то есть значение одного столбца из одной строки или одно вычисленное значение (например, значение агрегатной функции — сумма, среднее/минимальное/максимальное значение столбца). Такой подзапрос можно использовать с операторами сравнения `>`, `<`, `=` и др. (см. Comparison Operators в документации ClickHouse), передавать как аргумент функции или как значение столбца в операторе `SELECT`.
- Табличное значение, то есть несколько строк (возможно ноль строк). Такой подзапрос можно использовать с операторами для работы с множествами (`IN`, `NOT IN`, `ANY`, `ALL`, `EXISTS`) — в этом случае будет проверяться вхождение строки родительского запроса в результат, возвращаемый подзапросом. Также табличное значение подзапроса можно использовать в выражении `FROM` или `JOIN`, чтобы указать таблицу-источник для чтения данных.

ПРИМЕЧАНИЕ

- Синтаксически подзапрос представляет собой запрос `SELECT`, заключенный в круглые скобки.
- Синонимы не передаются в подзапросы и между подзапросами.

Ниже в статье приведены различные варианты использования подзапросов в запросах `SELECT`.

Создание таблиц для тестовых примеров

Чтобы воспроизвести приведенные в данной статье примеры, создайте предварительно таблицы:

- `employees`

```
CREATE TABLE employees (
    id UInt16,
    full_name String,
    hire_date Date,
    position Enum('c developer' = 1,
                  'python developer' = 2,
                  'java developer' = 3,
                  'frontend developer' = 4,
                  'test automation engineer' = 5,
                  'manual testing engineer' = 6,
                  'load testing engineer' = 7,
                  'support engineer 1st line' = 8,
                  'support engineer 2nd line' = 9),
    department Enum('development' = 10, 'qa' = 20, 'tech support' = 30),
    location String,
    email String)
ENGINE = MergeTree
ORDER BY id;
```

```
INSERT INTO employees VALUES
    (1, 'James Wilson', '2020-11-23', 1, 10, 'New York', 'jwilson@example.com'),
    (2, 'Mary Burton', '2023-05-06', 1, 10, 'Chicago', 'mburton@example.com'),
    (3, 'Patricia Robinson', '2021-10-11', 2, 10, 'Los Angeles', 'probinson@example.com'),
    (4, 'Robert Gray', '2019-07-03', 2, 10, 'Seattle', 'rgray@example.com'),
    (5, 'Elizabeth Tucker', '2024-02-03', 4, 10, 'Chicago', 'etucker@example.com'),
    (6, 'Joseph Lewis', '2023-04-05', 4, 10, 'New York', 'jlewis@example.com'),
    (7, 'Paul Brown', '2024-02-03', 3, 10, 'Chicago', 'pbrown@example.com'),
    (8, 'Andrew Clarke', '2024-11-23', 3, 10, 'Los Angeles', 'aclarke@example.com')
    (9, 'William Ferguson', '2023-11-12', 5, 20, 'Chicago', 'wferguson@example.com'),
    (10, 'Linda Black', '2021-07-20', 6, 20, 'New York', 'lblack@example.com'),
    (11, 'David Green', '2019-10-10', 7, 20, 'Chicago', 'dgreen@example.com'),
    (12, 'Daniel Gray', '2020-08-03', 7, 20, 'New York', 'dgray@example.com'),
    (13, 'Mark Armstrong', '2020-04-05', 8, 30, 'New York', 'marmstrong@example.com'),
    (14, 'Donald Carter', '2021-07-07', 8, 30, 'Seattle', 'dcarter@example.com'),
    (15, 'Elizabeth Collins', '2022-01-22', 9, 30, 'Chicago', 'ecollins@example.com');
```

```
    ┌─id─┬─full_name─────────┬──hire_date─┬─position──────────────────┬─department───┬─location────┬─email──────────────────┐
 1. │  1 │ James Wilson      │ 2020-11-23 │ c developer               │ development  │ New York    │ jwilson@example.com    │
 2. │  2 │ Mary Burton       │ 2023-05-06 │ c developer               │ development  │ Chicago     │ mburton@example.com    │
 3. │  3 │ Patricia Robinson │ 2021-10-11 │ python developer          │ development  │ Los Angeles │ probinson@example.com  │
 4. │  4 │ Robert Gray       │ 2019-07-03 │ python developer          │ development  │ Seattle     │ rgray@example.com      │
 5. │  5 │ Elizabeth Tucker  │ 2024-02-03 │ frontend developer        │ development  │ Chicago     │ etucker@example.com    │
 6. │  6 │ Joseph Lewis      │ 2023-04-05 │ frontend developer        │ development  │ New York    │ jlewis@example.com     │
 7. │  7 │ Paul Brown        │ 2024-02-03 │ java developer            │ development  │ Chicago     │ pbrown@example.com     │
 8. │  8 │ Andrew Clarke     │ 2024-11-23 │ java developer            │ development  │ Los Angeles │ aclarke@example.com    │
 9. │  9 │ William Ferguson  │ 2023-11-12 │ test automation engineer  │ qa           │ Chicago     │ wferguson@example.com  │
10. │ 10 │ Linda Black       │ 2021-07-20 │ manual testing engineer   │ qa           │ New York    │ lblack@example.com     │
11. │ 11 │ David Green       │ 2019-10-10 │ load testing engineer     │ qa           │ Chicago     │ dgreen@example.com     │
12. │ 12 │ Daniel Gray       │ 2020-08-03 │ load testing engineer     │ qa           │ New York    │ dgray@example.com      │
13. │ 13 │ Mark Armstrong    │ 2020-04-05 │ support engineer 1st line │ tech support │ New York    │ marmstrong@example.com │
14. │ 14 │ Donald Carter     │ 2021-07-07 │ support engineer 1st line │ tech support │ Seattle     │ dcarter@example.com    │
15. │ 15 │ Elizabeth Collins │ 2022-01-22 │ support engineer 2nd line │ tech support │ Chicago     │ ecollins@example.com   │
    └────┴───────────────────┴────────────┴───────────────────────────┴──────────────┴─────────────┴────────────────────────┘
```

- `employee_bonus_oct_24`

```
CREATE TABLE employee_bonus_oct_24 (employee_id UInt16, bonus UInt32) ENGINE = MergeTree ORDER BY employee_id;
```

```
INSERT INTO employee_bonus_oct_24
VALUES (1, 700), (2, 450), (3, 500), (4, 950), (5, 370), (6, 600),(7, 1000), (8, 950),
       (9, 770), (10, 350), (11, 500), (12, 670), (13, 600), (14, 600), (15, 480);
```

```
    ┌─employee_id─┬─bonus─┐
 1. │           1 │   700 │
 2. │           2 │   450 │
 3. │           3 │   500 │
 4. │           4 │   950 │
 5. │           5 │   370 │
 6. │           6 │   600 │
 7. │           7 │  1000 │
 8. │           8 │   950 │
 9. │           9 │   770 │
10. │          10 │   350 │
11. │          11 │   500 │
12. │          12 │   670 │
13. │          13 │   600 │
14. │          14 │   600 │
15. │          15 │   480 │
    └─────────────┴───────┘
```

- `employee_bonus_oct_24_full`

```
CREATE TABLE employee_bonus_oct_24_full (
    employee_id UInt16,
    full_name String,
    position String,
    department String,
    bonus UInt32)
ENGINE = MergeTree
ORDER BY employee_id;
```

```
INSERT INTO employee_bonus_oct_24_full SELECT
    employee_bonus_oct_24.employee_id,
    employees.full_name,
    employees.position,
    employees.department,
    employee_bonus_oct_24.bonus
FROM employee_bonus_oct_24
INNER JOIN employees ON employee_bonus_oct_24.employee_id = employees.id;
```

```
    ┌─employee_id─┬─full_name─────────┬─position──────────────────┬─department───┬─bonus─┐
 1. │           1 │ James Wilson      │ c developer               │ development  │   700 │
 2. │           2 │ Mary Burton       │ c developer               │ development  │   450 │
 3. │           3 │ Patricia Robinson │ python developer          │ development  │   500 │
 4. │           4 │ Robert Gray       │ python developer          │ development  │   950 │
 5. │           5 │ Elizabeth Tucker  │ frontend developer        │ development  │   370 │
 6. │           6 │ Joseph Lewis      │ frontend developer        │ development  │   600 │
 7. │           7 │ Paul Brown        │ java developer            │ development  │  1000 │
 8. │           8 │ Andrew Clarke     │ java developer            │ development  │   950 │
 9. │           9 │ William Ferguson  │ test automation engineer  │ qa           │   770 │
10. │          10 │ Linda Black       │ manual testing engineer   │ qa           │   350 │
11. │          11 │ David Green       │ load testing engineer     │ qa           │   500 │
12. │          12 │ Daniel Gray       │ load testing engineer     │ qa           │   670 │
13. │          13 │ Mark Armstrong    │ support engineer 1st line │ tech support │   600 │
14. │          14 │ Donald Carter     │ support engineer 1st line │ tech support │   600 │
15. │          15 │ Elizabeth Collins │ support engineer 2nd line │ tech support │   480 │
    └─────────────┴───────────────────┴───────────────────────────┴──────────────┴───────┘
```

Подзапросы в FROM

В качестве источника данных для чтения запросом `SELECT` в выражении `FROM` можно указывать не только имя таблицы или табличную функцию, но и подзапрос:

```
SELECT <expression> FROM (<subquery>) [AS <alias>]
```

В отличие от стандартного SQL, после подзапроса не обязательно указывать синоним (`AS <alias>`). Это можно сделать для совместимости, но указанное имя нигде не будет использоваться.

В следующем запросе подзапрос в выражении `FROM` возвращает премии сотрудников отдела `development` за октябрь и оценку премии каждого сотрудника относительно размера максимальной премии в компании:

```
SELECT employee, bonus, relative_to_max
FROM
(
    SELECT
        full_name AS employee,
        bonus,
        round((bonus/(SELECT max(bonus) FROM employee_bonus_oct_24_full)), 2) AS relative_to_max
    FROM employee_bonus_oct_24_full
    WHERE department = 'development'
)
WHERE relative_to_max < 0.5;
```

```
   ┌─employee─────────┬─bonus─┬─relative_to_max─┐
1. │ Mary Burton      │   450 │            0.45 │
2. │ Elizabeth Tucker │   370 │            0.37 │
   └──────────────────┴───────┴─────────────────┘
```

Подзапросы в JOIN

В запросе `SELECT` с оператором JOIN не оптимизируется порядок выполнения соединения по отношению к другим стадиям запроса. Сначала выполняется соединение (поиск в правой таблице), а затем — фильтрация `WHERE` и агрегация. Чтобы явно указать порядок вычислений и повысить производительность запроса, рекомендуется применять фильтр как можно раньше — для этого можно соединять подзапросы.

Например, в следующем запросе сначала вычисляются подзапросы в выражениях `FROM` (количество сотрудников в каждом отделе) и `LEFT ANY JOIN` (количество сотрудников в каждом отделе, получивших премию меньше 500) с группировкой и фильтрацией, а затем результаты этих подзапросов соединяются:

```
SELECT department, number_of_employees, bonus_less_500
FROM
(
    SELECT department, count() AS number_of_employees
    FROM employees
    GROUP BY department
)
LEFT ANY JOIN
(
    SELECT department, count() AS bonus_less_500
    FROM employee_bonus_oct_24_full
    WHERE bonus < 500
    GROUP BY department
) USING (department)
SETTINGS joined_subquery_requires_alias = 0;
```

```
   ┌─department───┬─number_of_employees─┬─bonus_less_500─┐
1. │ development  │                   8 │              2 │
2. │ qa           │                   4 │              1 │
3. │ tech support │                   3 │              1 │
   └──────────────┴─────────────────────┴────────────────┘
```

Подзапросы в WHERE

Подзапросы часто применяются в выражении `WHERE` запросов `SELECT` для фильтрации данных. Как упоминалось выше, для определения условий фильтрации данных подзапросы можно использовать с различными операторами сравнения:

- подзапросы, возвращающие скалярные значения — до или после операторов сравнения `=` (`==`), `!=` (`<>`), `>`, `>=`, `<`, `<=`. Например, следующий запрос находит всех сотрудников, получивших в октябре премию выше средней:

  ```
  SELECT full_name AS employee, position, department
  FROM employee_bonus_oct_24_full
  WHERE bonus > (SELECT avg(bonus) FROM employee_bonus_oct_24_full);
  ```

  ```
     ┌─employee─────────┬─position─────────────────┬─department──┐
  1. │ James Wilson     │ c developer              │ development │
  2. │ Robert Gray      │ python developer         │ development │
  3. │ Paul Brown       │ java developer           │ development │
  4. │ Andrew Clarke    │ java developer           │ development │
  5. │ William Ferguson │ test automation engineer │ qa          │
  6. │ Daniel Gray      │ load testing engineer    │ qa          │
     └──────────────────┴──────────────────────────┴─────────────┘
  ```

- подзапросы, возвращающие наборы строк данных — с операторами `IN`, `NOT IN`, `ANY`, `ALL`, `EXISTS`, которые позволяют проверять наличие конкретной строки в наборе строк. Примеры использования подзапросов с этими операторами приведены в разделах ниже.

IN

Синтаксис оператора `IN` с использованием подзапроса:

```
<IN_expression> IN (<IN_subquery>)
```

Оператор `IN` вычисляет выражение `IN_expression`, сравнивает его со значениями во всех строках результата подзапроса `IN_subquery` и возвращает:

- `1` — если в результатах подзапроса `IN_subquery` найдена строка, равная `IN_expression`;
- `0` — в следующих случаях:
  - в результатах подзапроса `IN_subquery` не найдено ни одной строки, равной `IN_expression`;
  - `IN_subquery` не возвращает ни одной строки;
  - в правой или левой части оператора `IN` присутствует `NULL` при условии, что параметр transform_null_in установлен в `0` (по умолчанию).

В следующем запросе используется оператор `IN`, чтобы вывести информацию о выплаченных в октябре премиях только для сотрудников, проживающих в Нью-Йорке:

```
SELECT full_name AS employee, bonus FROM employee_bonus_oct_24_full
WHERE employee_id IN (SELECT id FROM employees WHERE location='New York');
```

```
   ┌─employee───────┬─bonus─┐
1. │ James Wilson   │   700 │
2. │ Joseph Lewis   │   600 │
3. │ Linda Black    │   350 │
4. │ Daniel Gray    │   670 │
5. │ Mark Armstrong │   600 │
   └────────────────┴───────┘
```

ПРИМЕЧАНИЕ

- В подзапросе может быть указано более одного столбца для фильтрации кортежей (типы столбцов слева и справа от оператора `IN` должны совпадать):

  ```
  (<column1>, <column2>) IN (SELECT <IN_subquery_result_column1>, <IN_subquery_result_column2> FROM ...)
  ```

- Оператор `IN` с подзапросом можно использовать в любой части родительского запроса, в том числе в агрегатной функции или лямбда-функции.
- При распределенной обработке запроса вместо `IN` с подзапросом часто более оптимально использовать оператор `GLOBAL IN`. Более детальную информацию можно посмотреть в разделе Distributed Subqueries документации ClickHouse.

NOT IN

Синтаксис оператора `NOT IN` с подзапросом:

```
<NOT_IN_expression> NOT IN (<NOT_IN_subquery>)
```

Выражение `NOT_IN_expression` вычисляется и сравнивается с каждой строкой результата подзапроса `NOT_IN_subquery`. Возвращаемое значение:

- `1` — если в результатах `NOT_IN_expression` и `NOT_IN_subquery` не найдено совпадений, или подзапрос не возвращает ни одной строки;
- `0` — если в результатах подзапроса `NOT_IN_subquery` есть хоть одна строка, равная `NOT_IN_expression`.

Например, следующий запрос выводит данные о премиях сотрудников, проживающих не в Нью-Йорке:

```
SELECT full_name AS employee, bonus FROM employee_bonus_oct_24_full
WHERE employee_id NOT IN (SELECT id FROM employees WHERE location='New York');
```

```
    ┌─employee──────────┬─bonus─┐
 1. │ Mary Burton       │   450 │
 2. │ Patricia Robinson │   500 │
 3. │ Robert Gray       │   950 │
 4. │ Elizabeth Tucker  │   370 │
 5. │ Paul Brown        │  1000 │
 6. │ Andrew Clarke     │   950 │
 7. │ William Ferguson  │   770 │
 8. │ David Green       │   500 │
 9. │ Donald Carter     │   600 │
10. │ Elizabeth Collins │   480 │
    └───────────────────┴───────┘
```

ANY

Синтаксис оператора `ANY` с подзапросом:

```
<ANY_expression> <operator> ANY (<ANY_subquery>)
```

где `operator` может быть `=`, `<>`, `<`, `<=`, `>`, `>=`.

Оператор `ANY` выполняет сравнение результатов выражения `ANY_expression` и подзапроса `ANY_subquery` с помощью оператора `operator` и возвращает `1`, если хотя бы один элемент из результата подзапроса удовлетворяет указанному условию.

Например, следующий запрос находит сотрудников не из отдела `development`, премия которых в октябре больше чем хотя бы у одного сотрудника отдела `development` в этом же месяце:

```
SELECT full_name AS employee, bonus, department FROM employee_bonus_oct_24_full
WHERE bonus > ANY (SELECT bonus FROM employee_bonus_oct_24_full WHERE department='development') AND department!='development';
```

```
   ┌─employee──────────┬─bonus─┬─department───┐
1. │ William Ferguson  │   770 │ qa           │
2. │ David Green       │   500 │ qa           │
3. │ Daniel Gray       │   670 │ qa           │
4. │ Mark Armstrong    │   600 │ tech support │
5. │ Donald Carter     │   600 │ tech support │
6. │ Elizabeth Collins │   480 │ tech support │
   └───────────────────┴───────┴──────────────┘
```

Сравнение `= ANY` эквивалентно оператору IN. Если приведенный выше пример с `IN` переписать следующим образом, результат запроса будет тем же:

```
SELECT full_name AS employee, bonus FROM employee_bonus_oct_24_full
WHERE employee_id = ANY (SELECT id FROM employees WHERE location='New York');
```

```
   ┌─employee───────┬─bonus─┐
1. │ James Wilson   │   700 │
2. │ Joseph Lewis   │   600 │
3. │ Linda Black    │   350 │
4. │ Daniel Gray    │   670 │
5. │ Mark Armstrong │   600 │
   └────────────────┴───────┘
```

ALL

Синтаксис оператора `ALL` с подзапросом:

```
<ALL_expression> <operator> ALL (<ALL_subquery>)
```

где `operator` может быть `=`, `<>`, `<`, `<=`, `>`, `>=`.

Оператор `ALL` сравнивает значение выражения `ALL_expression` с каждым значением в наборе, полученным подзапросом `ALL_subquery`, и возвращает `1`, если условие сравнения верно для всех значений в наборе.

Например, следующий запрос возвращает список сотрудников, размер премий которых меньше чем у любого сотрудника отдела `tech support`:

```
SELECT full_name AS employee, bonus, department FROM employee_bonus_oct_24_full
WHERE bonus < ALL (SELECT bonus FROM employee_bonus_oct_24_full WHERE department='tech support');
```

```
   ┌─employee─────────┬─bonus─┬─department──┐
1. │ Mary Burton      │   450 │ development │
2. │ Elizabeth Tucker │   370 │ development │
3. │ Linda Black      │   350 │ qa          │
   └──────────────────┴───────┴─────────────┘
```

Оператор `!= ALL` эквивалентен NOT IN. Приведенный выше пример с `NOT IN` можно переписать следующим образом:

```
SELECT full_name AS employee, bonus FROM employee_bonus_oct_24_full
WHERE employee_id != ALL (SELECT id FROM employees WHERE location='New York');
```

Результат будет такой же:

```
    ┌─employee──────────┬─bonus─┐
 1. │ Mary Burton       │   450 │
 2. │ Patricia Robinson │   500 │
 3. │ Robert Gray       │   950 │
 4. │ Elizabeth Tucker  │   370 │
 5. │ Paul Brown        │  1000 │
 6. │ Andrew Clarke     │   950 │
 7. │ William Ferguson  │   770 │
 8. │ David Green       │   500 │
 9. │ Donald Carter     │   600 │
10. │ Elizabeth Collins │   480 │
    └───────────────────┴───────┘
```

EXISTS

Синтаксис оператора `EXISTS`:

```
EXISTS (<subquery>)
```

Оператор `EXISTS` проверяет, сколько строк содержит результат выполнения подзапроса. Если результат пустой, то оператор возвращает `0`. В остальных случаях оператор возвращает `1`.

ВНИМАНИЕ

Ссылки на таблицы или столбцы родительского запроса не поддерживаются в подзапросе.

Например, следующий запрос выведет данные таблицы `employees`, только если в ней есть хотя бы один сотрудник, проживающий в Чикаго:

```
SELECT full_name AS employee, location AS city
FROM employees
WHERE EXISTS (SELECT * FROM employees WHERE location='Chicago');
```

```
    ┌─employee──────────┬─city────────┐
 1. │ James Wilson      │ New York    │
 2. │ Mary Burton       │ Chicago     │
 3. │ Patricia Robinson │ Los Angeles │
 4. │ Robert Gray       │ Seattle     │
 5. │ Elizabeth Tucker  │ Chicago     │
 6. │ Joseph Lewis      │ New York    │
 7. │ Paul Brown        │ Chicago     │
 8. │ Andrew Clarke     │ Los Angeles │
 9. │ William Ferguson  │ Chicago     │
10. │ Linda Black       │ New York    │
11. │ David Green       │ Chicago     │
12. │ Daniel Gray       │ New York    │
13. │ Mark Armstrong    │ New York    │
14. │ Donald Carter     │ Seattle     │
15. │ Elizabeth Collins │ Chicago     │
    └───────────────────┴─────────────┘
```

Комбинирование запросов (UNION, INTERSECT, EXCEPT)

Елена Дворядкина

Содержание

Назад к содержанию

UNION
- Примеры

INTERSECT
- Примеры

EXCEPT
- Примеры

Для объединения результатов двух и более запросов можно использовать операторы множеств UNION, INTERSECT и EXCEPT.

Создание таблиц для тестовых примеров

Чтобы воспроизвести приведенные в данной статье примеры и посмотреть, как в ClickHouse работают различные операторы объединения множеств, создайте предварительно таблицы:

- `products_2023` — каталог продуктов за 2023 год:

```
CREATE TABLE products_2023
(
    id UInt32,
    name String,
    category String,
    price Float32
) ENGINE = MergeTree()
ORDER BY id;
```

```
INSERT INTO products_2023 VALUES
(1, 'Office Chair', 'Furniture', 170),
(2, 'Backpack', 'Accessories', 60),
(3, 'Monitor', 'Electronics', 200),
(4, 'Tablet', 'Electronics', 750),
(5, 'Water Bottle', 'Accessories', 10),
(6, 'Office Desk', 'Furniture', 250);
```

```
   ┌─id─┬─name─────────┬─category────┬─price─┐
1. │  1 │ Office Chair │ Furniture   │   170 │
2. │  2 │ Backpack     │ Accessories │    60 │
3. │  3 │ Monitor      │ Electronics │   200 │
4. │  4 │ Tablet       │ Electronics │   750 │
5. │  5 │ Water Bottle │ Accessories │    10 │
6. │  6 │ Office Desk  │ Furniture   │   250 │
   └────┴──────────────┴─────────────┴───────┘
```

- `products_2024` — каталог продуктов за 2024 год:

```
CREATE TABLE products_2024
(
    id UInt32,
    name String,
    category String,
    price Float32
) ENGINE = MergeTree()
ORDER BY id;
```

```
INSERT INTO products_2024 VALUES
(1, 'Office Chair', 'Furniture', 175),
(2, 'Backpack', 'Accessories', 60),
(3, 'Monitor', 'Electronics', 220),
(6, 'Office Desk', 'Furniture', 250),
(7, 'Laptop', 'Electronics', 1000),
(8, 'Smartphone', 'Electronics', 700),
(9, 'Cap', 'Accessories', 6),
(10, 'Coffee Table', 'Furniture', 180);
```

```
   ┌─id─┬─name─────────┬─category────┬─price─┐
1. │  1 │ Office Chair │ Furniture   │   175 │
2. │  2 │ Backpack     │ Accessories │    60 │
3. │  3 │ Monitor      │ Electronics │   220 │
4. │  6 │ Office Desk  │ Furniture   │   250 │
5. │  7 │ Laptop       │ Electronics │  1000 │
6. │  8 │ Smartphone   │ Electronics │   700 │
7. │  9 │ Cap          │ Accessories │     6 │
8. │ 10 │ Coffee Table │ Furniture   │   180 │
   └────┴──────────────┴─────────────┴───────┘
```

- `sales_2024_may` — данные о продажах в мае 2024 года:

```
CREATE TABLE sales_2024_may (
    sale_id UInt32,
    product_name String,
    sale_date Date DEFAULT toDate(now())
)
ENGINE = MergeTree()
ORDER BY sale_id;
```

```
INSERT INTO sales_2024_may VALUES
(1, 'Smartphone', '2024-05-06'),
(2, 'Smartphone', '2024-05-07'),
(3, 'Monitor', '2024-05-07'),
(4, 'Office Desk', '2024-05-08'),
(5, 'Monitor', '2024-05-09'),
(6, 'Office Chair', '2024-05-09'),
(7, 'Backpack', '2024-05-10'),
(8, 'Backpack', '2024-05-11'),
(9, 'Cap', '2024-05-11'),
(10, 'Cap', '2024-05-12');
```

```
    ┌─sale_id─┬─product_name─┬──sale_date─┐
 1. │       1 │ Smartphone   │ 2024-05-06 │
 2. │       2 │ Smartphone   │ 2024-05-07 │
 3. │       3 │ Monitor      │ 2024-05-07 │
 4. │       4 │ Office Desk  │ 2024-05-08 │
 5. │       5 │ Monitor      │ 2024-05-09 │
 6. │       6 │ Office Chair │ 2024-05-09 │
 7. │       7 │ Backpack     │ 2024-05-10 │
 8. │       8 │ Backpack     │ 2024-05-11 │
 9. │       9 │ Cap          │ 2024-05-11 │
10. │      10 │ Cap          │ 2024-05-12 │
    └─────────┴──────────────┴────────────┘
```

UNION

Выражение `UNION` добавляет к результату первого запроса результат второго запроса. Может использоваться в двух режимах:

- `UNION ALL` — в результат объединения включаются все строки из результатов обоих запросов, в том числе повторяющиеся;
- `UNION DISTINCT` — из результата объединения исключаются повторяющиеся строки.

Если ключевое слово `ALL` или `DISTINCT` явно не указано, поведение `UNION` определяется настройкой `union_default_mode`, значением которой может быть `ALL`, `DISTINCT` или пустая строка (в этом случае при использовании `UNION` генерируется исключение).

Синтаксис в общем виде:

```
SELECT <column1> [, <column2> ]
FROM <table1>
[WHERE <condition>]

UNION [ALL | DISTINCT]

SELECT <column1> [, <column2> ]
FROM <table2>
[WHERE <condition>]
```

Результирующие столбцы сопоставляются по их индексу (порядку внутри `SELECT`). Если имена столбцов не совпадают, то имена для конечного результата берутся из первого запроса.

При объединении выполняется приведение типов. Например, если два запроса имеют одно и то же поле с совместимыми типами non-Nullable и Nullable, результат объединения будет иметь поле типа Nullable.

Запросы, которые являются частью `UNION`, могут быть заключены в круглые скобки. `ORDER BY` и `LIMIT` применяются к отдельным запросам, а не к конечному результату. Если нужно применить преобразование к конечному результату, поместите все запросы с `UNION` в подзапрос в выражении `FROM`.

Примеры

UNION ALL

Следующий запрос объединяет каталоги товаров за 2023 и 2024 годы:

```
SELECT id, name FROM products_2023
UNION ALL
SELECT id, name FROM products_2024;
```

```
   ┌─id─┬─name─────────┐
1. │  1 │ Office Chair │
2. │  2 │ Backpack     │
3. │  3 │ Monitor      │
4. │  4 │ Tablet       │
5. │  5 │ Water Bottle │
6. │  6 │ Office Desk  │
   └────┴──────────────┘
    ┌─id─┬─name─────────┐
 7. │  1 │ Office Chair │
 8. │  2 │ Backpack     │
 9. │  3 │ Monitor      │
10. │  6 │ Office Desk  │
11. │  7 │ Laptop       │
12. │  8 │ Smartphone   │
13. │  9 │ Cap          │
14. │ 10 │ Coffee Table │
    └────┴──────────────┘
```

UNION DISTINCT

Следующий запрос объединяет каталоги товаров за 2023 и 2024 годы, исключая дублирование:

```
SELECT id, name FROM products_2023
UNION DISTINCT
SELECT id, name FROM products_2024;
```

```
   ┌─id─┬─name─────────┐
1. │  1 │ Office Chair │
2. │  2 │ Backpack     │
3. │  3 │ Monitor      │
4. │  4 │ Tablet       │
5. │  5 │ Water Bottle │
6. │  6 │ Office Desk  │
   └────┴──────────────┘
    ┌─id─┬─name─────────┐
 7. │  7 │ Laptop       │
 8. │  8 │ Smartphone   │
 9. │  9 │ Cap          │
10. │ 10 │ Coffee Table │
    └────┴──────────────┘
```

INTERSECT

Выражение `INTERSECT` возвращает все строки, которые есть как в результате первого запроса, так и в результате второго запроса. Запросы должны совпадать по количеству столбцов, их порядку и типам данных. Результат `INTERSECT` может содержать дубликаты строк. Чтобы исключить дубликаты, используйте `INTERSECT DISTINCT`.

Если используется несколько выражений `INTERSECT` и скобки не указаны, вычисление пересечений выполняется слева направо. `INTERSECT` имеет более высокий приоритет выполнения, чем `UNION` и `EXCEPT`.

Синтаксис в общем виде:

```
SELECT <column1> [, <column2> ]
FROM <table1>
[WHERE <condition>]

INTERSECT [DISTINCT]

SELECT <column1> [, <column2> ]
FROM <table2>
[WHERE <condition>]
```

Примеры

Следующий запрос возвращает названия продуктов, проданных в мае 2024 года, которые входят в категорию `Electronics` по данным таблицы `products_2024`:

```
SELECT product_name FROM sales_2024_may
INTERSECT
SELECT name FROM products_2024 WHERE category='Electronics';
```

В результате видно, что было по две продажи продуктов `Smartphone` и `Monitor`:

```
   ┌─product_name─┐
1. │ Smartphone   │
2. │ Smartphone   │
3. │ Monitor      │
4. │ Monitor      │
   └──────────────┘
```

Ключевое слово `DISTINCT` исключает дубликаты строк из результата предыдущего запроса:

```
SELECT product_name FROM sales_2024_may
INTERSECT DISTINCT
SELECT name FROM products_2024 WHERE category='Electronics';
```

```
   ┌─product_name─┐
1. │ Smartphone   │
2. │ Monitor      │
   └──────────────┘
```

EXCEPT

Выражение `EXCEPT` возвращает результат первого запроса, исключая строки, которые также входят в результат второго запроса. Запросы должны совпадать по количеству столбцов, их порядку и типам данных. Результат `EXCEPT` может содержать дубликаты строк. Чтобы исключить дубликаты, используйте `EXCEPT DISTINCT`.

Если используется несколько выражений `EXCEPT` и скобки не указаны, `EXCEPT` выполняется по порядку слева направо. У `EXCEPT` такой же приоритет выполнения, как у `UNION` — более низкий, чем у `INTERSECT`.

Синтаксис в общем виде:

```
SELECT <column1> [, <column2> ]
FROM <table1>
[WHERE <condition>]

EXCEPT [DISTINCT]

SELECT <column1> [, <column2> ]
FROM <table2>
[WHERE <condition>]
```

Примеры

EXCEPT

Следующий запрос возвращает названия проданных в мае 2024 года продуктов, которые не входят в категорию `Electronics`:

```
SELECT product_name FROM sales_2024_may
EXCEPT
SELECT name FROM products_2024 WHERE category='Electronics';
```

```
   ┌─product_name─┐
1. │ Office Desk  │
2. │ Office Chair │
3. │ Backpack     │
4. │ Backpack     │
5. │ Cap          │
6. │ Cap          │
   └──────────────┘
```

Используйте ключевое слово `DISTINCT`, чтобы убрать дублирование из результата предыдущего запроса:

```
SELECT product_name FROM sales_2024_may
EXCEPT DISTINCT
SELECT name FROM products_2024 WHERE category='Electronics';
```

```
   ┌─product_name─┐
1. │ Office Desk  │
2. │ Office Chair │
3. │ Backpack     │
4. │ Cap          │
   └──────────────┘
```

Общие табличные выражения (CTE)

Елена Дворядкина

Содержание

Назад к содержанию

- Синтаксис CTE
- Примеры
  - Запрос с CTE
  - Несколько CTE в запросе
  - Вложенные CTE
- Рекурсивные CTE
  - Обзор
  - Порядок поиска
  - Выявление бесконечных циклов

Общее табличное выражение (Common Table Expression, CTE) — именованный набор данных, определяемый в области SQL-запроса и существующий только во время его выполнения (CTE не материализуется и не сохраняется где-либо в базе данных на постоянной основе), к которому можно неоднократно обращаться в рамках основного запроса.

CTE позволяет упростить и сделать более читаемым сложный запрос (разделить код на простые составные части, уменьшить его размер, избежать дублирования), а также оптимизировать выполнение запроса, который предполагает многократное обращение к одним и тем же наборам данных (необходимые данные временно хранятся в кеше, их не требуется искать каждый раз).

Синтаксис CTE

CTE создается с помощью выражения `WITH` перед основным запросом:

```
WITH <cte_name> AS (<cte_subquery>)
<primary_query>;
```

где:

- `<cte_name>` — имя CTE, по которому на него можно ссылаться в других частях основного SQL-запроса;
- `<cte_subquery>` — SQL-запрос, результирующий набор данных которого заполняет CTE;
- `<primary_query>` — основной SQL-запрос, использующий CTE.

Синтаксис CTE для использования результатов скалярного подзапроса немного отличается:

```
WITH (<cte_scalar_subquery>) AS <cte_name>
<primary_query>;
```

В одном выражении `WITH` можно определить несколько CTE, разделяя их запятыми:

```
WITH
    <cte1>,
    <cte2>,
    <cte3>
<primary_query>;
```

Каждое CTE может ссылаться на любое предыдущее CTE в рамках блока `WITH`.

Примеры

В этом разделе приведены примеры использования CTE в запросах для анализа данных о премиях, выданных сотрудникам сети магазинов в двух городах за сентябрь, которые содержатся в таблице `employee_bonus_sept_24`:

```
CREATE TABLE employee_bonus_sept_24 (
    empl_id UInt16,
    full_name String,
    position Enum('store manager' = 1, 'cashier' = 2, 'sales assistant' = 3),
    store_id UInt8,
    city Enum('New York' = 1, 'Chicago' = 2),
    bonus UInt32)
ENGINE = MergeTree
ORDER BY empl_id;
```

```
INSERT INTO employee_bonus_sept_24 VALUES
    (1, 'James Wilson', 1, 101, 1, 1000),
    (2, 'Mary Burton', 2, 101, 1, 450),
    (3, 'Patricia Robinson', 2, 101, 1, 500),
    (4, 'Robert Gray', 1, 102, 1, 1200),
    (5, 'Elizabeth Tucker', 2, 102, 1, 470),
    (6, 'Joseph Lewis', 3, 102, 1, 700),
    (7, 'William Ferguson', 1, 201, 2, 970),
    (8, 'Linda Black', 2, 201, 2, 350),
    (9, 'David Green', 2, 201, 2, 370),
    (10, 'Daniel Gray', 3, 201, 2, 770),
    (11, 'Mark Armstrong', 1, 202, 2, 1200),
    (12, 'Donald Carter', 2, 202, 2, 420),
    (13, 'Elizabeth Collins', 2, 202, 2, 400),
    (14, 'Paul Brown', 3, 202, 2, 660),
    (15, 'Andrew Clarke', 3, 202, 2, 680);
```

```
    ┌─empl_id─┬─full_name─────────┬─position────────┬─store_id─┬─city─────┬─bonus─┐
 1. │       1 │ James Wilson      │ store manager   │      101 │ New York │  1000 │
 2. │       2 │ Mary Burton       │ cashier         │      101 │ New York │   450 │
 3. │       3 │ Patricia Robinson │ cashier         │      101 │ New York │   500 │
 4. │       4 │ Robert Gray       │ store manager   │      102 │ New York │  1200 │
 5. │       5 │ Elizabeth Tucker  │ cashier         │      102 │ New York │   470 │
 6. │       6 │ Joseph Lewis      │ sales assistant │      102 │ New York │   700 │
 7. │       7 │ William Ferguson  │ store manager   │      201 │ Chicago  │   970 │
 8. │       8 │ Linda Black       │ cashier         │      201 │ Chicago  │   350 │
 9. │       9 │ David Green       │ cashier         │      201 │ Chicago  │   370 │
10. │      10 │ Daniel Gray       │ sales assistant │      201 │ Chicago  │   770 │
11. │      11 │ Mark Armstrong    │ store manager   │      202 │ Chicago  │  1200 │
12. │      12 │ Donald Carter     │ cashier         │      202 │ Chicago  │   420 │
13. │      13 │ Elizabeth Collins │ cashier         │      202 │ Chicago  │   400 │
14. │      14 │ Paul Brown        │ sales assistant │      202 │ Chicago  │   660 │
15. │      15 │ Andrew Clarke     │ sales assistant │      202 │ Chicago  │   680 │
    └─────────┴───────────────────┴─────────────────┴──────────┴──────────┴───────┘
```

Запрос с CTE

В приведенном ниже запросе:

- с помощью CTE (`avg_bonus`) вычисляется средний размер премии для каждой должности по всем магазинам в данном месяце;
- выводятся только те сотрудники, которые получили премию выше средней.

```
WITH avg_bonus AS
    (
        SELECT position, round(avg(bonus)) AS position_avg_bonus
        FROM employee_bonus_sept_24
        GROUP BY position
    )
SELECT
    b.full_name AS employee,
    b.position,
    b.store_id AS store,
    b.bonus,
    ab.position_avg_bonus
FROM employee_bonus_sept_24 AS b
INNER JOIN avg_bonus AS ab ON b.position = ab.position
WHERE b.bonus > ab.position_avg_bonus
ORDER BY b.position, b.bonus DESC;
```

```
   ┌─employee──────────┬─position────────┬─store─┬─bonus─┬─position_avg_bonus─┐
1. │ Robert Gray       │ store manager   │   102 │  1200 │               1092 │
2. │ Mark Armstrong    │ store manager   │   202 │  1200 │               1092 │
3. │ Patricia Robinson │ cashier         │   101 │   500 │                423 │
4. │ Elizabeth Tucker  │ cashier         │   102 │   470 │                423 │
5. │ Mary Burton       │ cashier         │   101 │   450 │                423 │
6. │ Daniel Gray       │ sales assistant │   201 │   770 │                702 │
   └───────────────────┴─────────────────┴───────┴───────┴────────────────────┘
```

Оконные функции в CTE

Если в запросе нужно отфильтровать данные по значениям оконных функций, вычисление этих значений необходимо выполнять в рамках CTE (или подзапроса).

Например, в следующем запросе сначала в CTE `bonus_rank_by_city` премии сотрудников ранжируются с помощью оконной функции `dense_rank()` с окном по городу, затем в выражении `WHERE` все строки фильтруются по вычисленному рангу, чтобы вывести 3 самых высоких премии сотрудников в каждом городе:

```
WITH bonus_rank_by_city AS
    (
        SELECT *, dense_rank() OVER (PARTITION BY city ORDER BY bonus DESC) AS rnk
        FROM employee_bonus_sept_24
        ORDER BY city, rnk
    )
SELECT *
FROM bonus_rank_by_city
WHERE rnk BETWEEN 1 AND 3;
```

```
   ┌─empl_id─┬─full_name────────┬─position────────┬─store_id─┬─city─────┬─bonus─┬─rnk─┐
1. │       4 │ Robert Gray      │ store manager   │      102 │ New York │  1200 │   1 │
2. │       1 │ James Wilson     │ store manager   │      101 │ New York │  1000 │   2 │
3. │       6 │ Joseph Lewis     │ sales assistant │      102 │ New York │   700 │   3 │
4. │      11 │ Mark Armstrong   │ store manager   │      202 │ Chicago  │  1200 │   1 │
5. │       7 │ William Ferguson │ store manager   │      201 │ Chicago  │   970 │   2 │
6. │      10 │ Daniel Gray      │ sales assistant │      201 │ Chicago  │   770 │   3 │
   └─────────┴──────────────────┴─────────────────┴──────────┴──────────┴───────┴─────┘
```

Несколько CTE в запросе

В следующем запросе используются два CTE:

- `avg_bonus_position` — средний размер месячной премии по позиции;
- `avg_bonus_city` — средний размер месячной премии по городу.

```
WITH
    avg_bonus_position AS
    (
        SELECT position, round(avg(bonus)) AS position_avg_bonus
        FROM employee_bonus_sept_24
        GROUP BY position
    ),
    avg_bonus_city AS
    (
        SELECT city, round(avg(bonus)) AS city_avg_bonus
        FROM employee_bonus_sept_24
        GROUP BY city
    )
SELECT
    b.full_name AS employee,
    b.position AS position,
    b.store_id AS store,
    b.city AS city,
    b.bonus,
    abp.position_avg_bonus,
    abc.city_avg_bonus
FROM employee_bonus_sept_24 AS b
INNER JOIN avg_bonus_position AS abp ON b.position = abp.position
INNER JOIN avg_bonus_city AS abc ON b.city = abc.city
ORDER BY b.position, b.bonus DESC;
```

```
    ┌─employee──────────┬─position────────┬─store─┬─city─────┬─bonus─┬─position_avg_bonus─┬─city_avg_bonus─┐
 1. │ Robert Gray       │ store manager   │   102 │ New York │  1200 │               1092 │            720 │
 2. │ Mark Armstrong    │ store manager   │   202 │ Chicago  │  1200 │               1092 │            647 │
 3. │ James Wilson      │ store manager   │   101 │ New York │  1000 │               1092 │            720 │
 4. │ William Ferguson  │ store manager   │   201 │ Chicago  │   970 │               1092 │            647 │
 5. │ Patricia Robinson │ cashier         │   101 │ New York │   500 │                423 │            720 │
 6. │ Elizabeth Tucker  │ cashier         │   102 │ New York │   470 │                423 │            720 │
 7. │ Mary Burton       │ cashier         │   101 │ New York │   450 │                423 │            720 │
 8. │ Donald Carter     │ cashier         │   202 │ Chicago  │   420 │                423 │            647 │
 9. │ Elizabeth Collins │ cashier         │   202 │ Chicago  │   400 │                423 │            647 │
10. │ David Green       │ cashier         │   201 │ Chicago  │   370 │                423 │            647 │
11. │ Linda Black       │ cashier         │   201 │ Chicago  │   350 │                423 │            647 │
12. │ Daniel Gray       │ sales assistant │   201 │ Chicago  │   770 │                702 │            647 │
13. │ Joseph Lewis      │ sales assistant │   102 │ New York │   700 │                702 │            720 │
14. │ Andrew Clarke     │ sales assistant │   202 │ Chicago  │   680 │                702 │            647 │
15. │ Paul Brown        │ sales assistant │   202 │ Chicago  │   660 │                702 │            647 │
    └───────────────────┴─────────────────┴───────┴──────────┴───────┴────────────────────┴────────────────┘
```

Вложенные CTE

Следующий запрос позволяет оценить объем средств, выделенных на премирование сотрудников в разных магазинах — посмотреть, какой процент сумма премий, выплаченных сотрудникам в каждом магазине, занимает от общей суммы расходов на премии сотрудникам всей сети. Для этого в запросе используются вложенные CTE:

- `bonus_all_stores` — общая сумма расходов на премии сотрудникам всех магазинов;
- `store_bonus_totals_and_rates` — сумма премий, выплаченных сотрудникам в каждом магазине, и ее доля относительно общих расходов на премирование сотрудников в рассматриваемом месяце, которая вычисляется с использованием выходных данных CTE `bonus_all_stores`.

```
WITH
    (SELECT sum(bonus) AS all_stores_bonus FROM employee_bonus_sept_24) AS bonus_all_stores,
    store_bonus_totals_and_rates AS
    (
        SELECT
            store_id,
            count() AS employee_count,
            sum(bonus) AS store_bonus_total,
            store_bonus_total / bonus_all_stores AS store_bonus_rate
        FROM employee_bonus_sept_24
        GROUP BY store_id
    )
SELECT
    store_id,
    employee_count,
    store_bonus_total AS total_bonus,
    round(store_bonus_rate * 100, 2) AS bonus_percentage
FROM store_bonus_totals_and_rates;
```

```
   ┌─store_id─┬─employee_count─┬─total_bonus─┬─bonus_percentage─┐
1. │      101 │              3 │        1950 │            19.23 │
2. │      102 │              3 │        2370 │            23.37 │
3. │      201 │              4 │        2460 │            24.26 │
4. │      202 │              5 │        3360 │            33.14 │
   └──────────┴────────────────┴─────────────┴──────────────────┘
```

Рекурсивные CTE

Обзор

Для работы с иерархическими и древовидными данными ADQM/ClickHouse поддерживает возможность создания рекурсивных CTE, запросы в которых могут выполняться многократно, обращаясь при каждом вызове к собственным результатам, полученным на предыдущей итерации.

Для создания рекурсивного CTE используйте модификатор `RECURSIVE` после `WITH`. Синтаксис рекурсивного CTE в общем виде:

```
WITH RECURSIVE <cte_name> AS (
    <cte_non_recursive_term>
    UNION ALL
    <cte_recursive_term>
)
SELECT <list_of_columns> FROM <cte_name>;
```

где:

- `<cte_non_recursive_term>` — нерекурсивная часть запроса CTE, которая выполняется в первую очередь. Полученные данные записываются в результирующую выборку CTE, а также используются как входные значения для первого вызова рекурсивной части.
- `<cte_recursive_term>` — рекурсивная часть запроса CTE, которая обращается к результату своего выполнения на предыдущей итерации (доступен под именем CTE — `<cte_name>`) и выполняется до тех пор, пока она возвращает какие-либо строки. Результат выполнения рекурсивной части, полученный на каждой итерации, также записывается в результирующую выборку CTE, которая будет доступна в основном запросе под тем же именем `<cte_name>` после того, как отработает весь запрос CTE.

Например, создайте таблицу, в которой будет храниться информация о позициях в компании:

```
CREATE TABLE positions (
    id UInt32,
    position String,
    manager_id Nullable(UInt32)
) ENGINE = Memory;
```

Заполните таблицу тестовыми данными:

```
INSERT INTO positions VALUES
    (1, 'CEO', NULL),
    (2, 'CTO', 1),
    (3, 'Backend Team Lead', 2),
    (4, 'Frontend Team Lead', 2),
    (5, 'Backend Developer Senior', 3),
    (6, 'Backend Developer Middle', 3),
    (7, 'Frontend Developer Senior', 4),
    (8, 'Frontend Developer Middle', 4),
    (9, 'Frontend Developer Junior', 4),
    (10, 'UX/UI Designer', 2);
```

Выполните запрос с использованием рекурсивного CTE, чтобы посмотреть, кому подчиняются сотрудники на различных позициях в организационной структуре компании:

```
WITH RECURSIVE employee_hierarchy AS (
    -- Нерекурсивная часть: начать с CEO
    SELECT id, position, manager_id, toUInt16(0) AS level, [position] AS path
    FROM positions
    WHERE manager_id IS NULL

    UNION ALL

    -- Рекурсивная часть: добавить подчиненных
    SELECT p.id, p.position, p.manager_id, eh.level + 1, arrayConcat(eh.path, [p.position]) AS path
    FROM positions p
    JOIN employee_hierarchy eh ON p.manager_id = eh.id
)
SELECT position, level, arrayStringConcat(path, ' -> ') AS full_path
FROM employee_hierarchy;
```

Этот запрос выполняется в следующей последовательности:

1. Нерекурсивная часть CTE возвращает стартовые данные из таблицы `positions` — позицию, для которой не указан менеджер (по условию `manager_id IS NULL`). В данном примере это `CEO` — верхняя позиция в иерархии компании, которой назначается уровень `0`.
2. Полученные в результате выполнения нерекурсивной части данные используются для вычисления первой итерации рекурсивной части CTE — осуществляется поиск позиций, которые находятся в подчинении `CEO`, путем выполнения JOIN для таблицы `positions` и CTE `employee_hierarchy` по ключу соединения `positions.manager_id = employee_hierarchy.id`. Таким образом, первая итерация рекурсивной части CTE возвращает позицию `CTO` (для которой значение `manager_id` равно идентификатору позиции `CEO`) и присваивает ей уровень `1` в иерархии компании. Результат этой итерации добавляется в результирующую выборку CTE и будет использоваться в качестве входных значений для следующей итерации.
3. Выполняются еще две итерации рекурсивной части, которые возвращают позиции следующих двух уровней в иерархии компании — `2` и `3`.
4. Четвертая итерация возвращает пустой результат, так как у позиций уровня `3` в таблице `positions` нет подчиненных — выполнение CTE на этом завершается.
5. Основной запрос обращается к CTE, чтобы получить выборку из его результирующего набора данных, который представляет собой объединение результатов выполнения нерекурсивной части и всех итераций рекурсивной части.

```
   ┌─position─┬─level─┬─full_path─┐
1. │ CEO      │     0 │ CEO       │
   └──────────┴───────┴───────────┘
   ┌─position─┬─level─┬─full_path──┐
2. │ CTO      │     1 │ CEO -> CTO │
   └──────────┴───────┴────────────┘
   ┌─position───────────┬─level─┬─full_path─────────────────────────┐
3. │ Backend Team Lead  │     2 │ CEO -> CTO -> Backend Team Lead  │
4. │ Frontend Team Lead │     2 │ CEO -> CTO -> Frontend Team Lead │
5. │ UX/UI Designer     │     2 │ CEO -> CTO -> UX/UI Designer     │
   └────────────────────┴───────┴──────────────────────────────────┘
    ┌─position──────────────────┬─level─┬─full_path─────────────────────────────────────────────────────┐
 6. │ Backend Developer Senior  │     3 │ CEO -> CTO -> Backend Team Lead -> Backend Developer Senior   │
 7. │ Backend Developer Middle  │     3 │ CEO -> CTO -> Backend Team Lead -> Backend Developer Middle   │
 8. │ Frontend Developer Senior │     3 │ CEO -> CTO -> Frontend Team Lead -> Frontend Developer Senior │
 9. │ Frontend Developer Middle │     3 │ CEO -> CTO -> Frontend Team Lead -> Frontend Developer Middle │
10. │ Frontend Developer Junior │     3 │ CEO -> CTO -> Frontend Team Lead -> Frontend Developer Junior │
    └───────────────────────────┴───────┴───────────────────────────────────────────────────────────────┘
```

Порядок поиска

Приведенный выше запрос выводит данные в том порядке, в каком они были добавлены в результирующий набор рекурсивного CTE — отсортированными "сначала в ширину" (breadth-first order) по уровню вложенности рекурсивного запроса, который в данном случае соответствует уровню позиции в иерархии компании (`level`).

Чтобы отсортировать результаты "сначала в глубину" (depth-first order), примените сортировку по полю `path`, в котором хранится путь до каждой строки в виде массива уже просмотренных строк (в данном примере это последовательность всех вышестоящих позиций в иерархии компании для каждой позиции).

```
WITH RECURSIVE employee_hierarchy AS (
    -- Нерекурсивная часть: начать с CEO
    SELECT id, position, manager_id, toUInt16(0) AS level, [position] AS path
    FROM positions
    WHERE manager_id IS NULL

    UNION ALL

    -- Рекурсивная часть: добавить подчиненных
    SELECT p.id, p.position, p.manager_id, eh.level + 1, arrayConcat(eh.path, [p.position]) AS path
    FROM positions p
    JOIN employee_hierarchy eh ON p.manager_id = eh.id
)
SELECT repeat('    ', level) || position AS position, level, arrayStringConcat(path, ' -> ') AS full_path
FROM employee_hierarchy
ORDER BY path;
```

```
    ┌─position─────────────────────────────┬─level─┬─full_path─────────────────────────────────────────────────────┐
 1. │ CEO                                  │     0 │ CEO                                                           │
 2. │     CTO                              │     1 │ CEO -> CTO                                                    │
 3. │         Backend Team Lead            │     2 │ CEO -> CTO -> Backend Team Lead                               │
 4. │             Backend Developer Middle │     3 │ CEO -> CTO -> Backend Team Lead -> Backend Developer Middle   │
 5. │             Backend Developer Senior │     3 │ CEO -> CTO -> Backend Team Lead -> Backend Developer Senior   │
 6. │         Frontend Team Lead           │     2 │ CEO -> CTO -> Frontend Team Lead                              │
 7. │             Frontend Developer Junior │     3 │ CEO -> CTO -> Frontend Team Lead -> Frontend Developer Junior │
 8. │             Frontend Developer Middle │     3 │ CEO -> CTO -> Frontend Team Lead -> Frontend Developer Middle │
 9. │             Frontend Developer Senior │     3 │ CEO -> CTO -> Frontend Team Lead -> Frontend Developer Senior │
10. │         UX/UI Designer               │     2 │ CEO -> CTO -> UX/UI Designer                                  │
    └──────────────────────────────────────┴───────┴───────────────────────────────────────────────────────────────┘
```

Выявление бесконечных циклов

При работе с рекурсивным CTE важно проверить, не приводит ли рекурсивная часть запроса к бесконечному циклу, и при необходимости установить условия остановки рекурсии. В данном разделе приведены примеры различных механизмов защиты запросов от зацикливания:

- Отслеживание уже обработанных значений
- Ограничение глубины рекурсии
- Использование LIMIT
- Ограничение по значению

Отслеживание уже обработанных значений

Стандартный способ выявления бесконечного цикла — отслеживать одно или несколько полей и проверять, не было ли текущее значение обработано ранее (иными словами, не была ли текущая точка уже достигнута выполнением предыдущей последовательности рекурсий).

Следующий пример показывает как можно обойти граф с циклом, исключив риск бесконечной рекурсии.

Создайте таблицу `graph`:

```
CREATE TABLE graph (
    from_node UInt32,
    to_node UInt32
) ENGINE = MergeTree()
ORDER BY (from_node, to_node);
```

Вставьте в таблицу тестовые данные, соответствующие графу с циклом `1 → 2 → 4 → 7 → 1`:

```
INSERT INTO graph VALUES
    (1, 2), (1, 3),
    (2, 4), (2, 5),
    (3, 6),
    (4, 7), (4, 8),
    (5, 9),
    (7, 1),
    (6, 10);
```

Запрос на обход графа без проверки на наличие циклов уйдет в бесконечную рекурсию и завершится ошибкой `Maximum recursive CTE evaluation depth (1000) exceeded`:

```
WITH RECURSIVE graph_traversal AS (
    SELECT from_node, to_node,
    FROM graph
    WHERE from_node = 1 -- Начать с узла 1

    UNION ALL

    SELECT g.from_node, g.to_node,
    FROM graph_traversal AS gt
    JOIN graph AS g ON gt.to_node = g.from_node
)
SELECT from_node, to_node,
FROM graph_traversal;
```

Чтобы избежать зацикливания запроса, сохраняйте пройденные пути до узлов графа в виде массивов уже просмотренных узлов в поле `path` и используйте их для отслеживания повторного прохождения узлов, записывая признак обнаружения цикла в поле `has_cycle`:

```
WITH RECURSIVE graph_traversal AS (
    SELECT from_node, to_node, 1 AS depth, [from_node, to_node] AS path, false AS has_cycle
    FROM graph
    WHERE from_node = 1 -- Начать с узла 1

    UNION ALL

    SELECT g.from_node, g.to_node, gt.depth + 1, arrayPushBack(gt.path, g.to_node), has(gt.path, g.to_node) AS has_cycle
    FROM graph_traversal AS gt
    JOIN graph AS g ON gt.to_node = g.from_node
    WHERE NOT gt.has_cycle -- Остановить рекурсию при обнаружении цикла
)
SELECT from_node, to_node, depth, path, has_cycle, arrayStringConcat(path, ' -> ') AS path_str
FROM graph_traversal;
```

Результат запроса

```
   ┌─from_node─┬─to_node─┬─depth─┬─path──┬─has_cycle─┬─path_str─┐
1. │         1 │       2 │     1 │ [1,2] │ false     │ 1 -> 2   │
2. │         1 │       3 │     1 │ [1,3] │ false     │ 1 -> 3   │
   └───────────┴─────────┴───────┴───────┴───────────┴──────────┘
   ┌─from_node─┬─to_node─┬─depth─┬─path────┬─has_cycle─┬─path_str────┐
3. │         2 │       4 │     2 │ [1,2,4] │ false     │ 1 -> 2 -> 4 │
4. │         2 │       5 │     2 │ [1,2,5] │ false     │ 1 -> 2 -> 5 │
5. │         3 │       6 │     2 │ [1,3,6] │ false     │ 1 -> 3 -> 6 │
   └───────────┴─────────┴───────┴─────────┴───────────┴─────────────┘
   ┌─from_node─┬─to_node─┬─depth─┬─path───────┬─has_cycle─┬─path_str───────────┐
6. │         4 │       7 │     3 │ [1,2,4,7]  │ false     │ 1 -> 2 -> 4 -> 7   │
7. │         4 │       8 │     3 │ [1,2,4,8]  │ false     │ 1 -> 2 -> 4 -> 8   │
8. │         5 │       9 │     3 │ [1,2,5,9]  │ false     │ 1 -> 2 -> 5 -> 9   │
9. │         6 │      10 │     3 │ [1,3,6,10] │ false     │ 1 -> 3 -> 6 -> 10  │
   └───────────┴─────────┴───────┴────────────┴───────────┴────────────────────┘
    ┌─from_node─┬─to_node─┬─depth─┬─path────────┬─has_cycle─┬─path_str───────────────┐
10. │         7 │       1 │     4 │ [1,2,4,7,1] │ true      │ 1 -> 2 -> 4 -> 7 -> 1 │
    └───────────┴─────────┴───────┴─────────────┴───────────┴───────────────────────┘
```

Ограничение глубины рекурсии

При выполнении рекурсивного CTE можно сохранять уровень вложенности рекурсий в отдельном столбце и использовать его для ограничения глубины поиска. В приведенном ниже примере это столбец `depth`:

```
WITH RECURSIVE graph_traversal AS (
    SELECT from_node, to_node, 1 AS depth, [from_node, to_node] AS path
    FROM graph
    WHERE from_node = 1 -- Начать с узла 1

    UNION ALL

    SELECT g.from_node, g.to_node, gt.depth + 1, arrayPushBack(gt.path, g.to_node)
    FROM graph_traversal AS gt
    JOIN graph AS g ON gt.to_node = g.from_node
    WHERE gt.depth < 3 -- Ограничение глубины рекурсии
)
SELECT from_node, to_node, depth, path, arrayStringConcat(path, ' -> ') AS path_str
FROM graph_traversal;
```

Максимальную глубину рекурсивных CTE в ADQM можно также настроить на уровне сессии с помощью параметра `max_recursive_cte_evaluation_depth` (значение по умолчанию — `1000`), например:

```
SET max_recursive_cte_evaluation_depth = 3;
```

Использование LIMIT

Еще один способ остановить бесконечную рекурсию CTE — добавить `LIMIT` в основной запрос. Например, следующий запрос вычисляет факториал для чисел от `1` до `10`:

```
WITH RECURSIVE r AS (
    SELECT 1 AS i, toUInt32(1) AS factorial
    UNION ALL
    SELECT i+1 AS i, factorial*i as factorial
    FROM r
)
SELECT * FROM r LIMIT 10;
```

Результат запроса

```
   ┌─i─┬─factorial─┐
1. │ 1 │         1 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
2. │ 2 │         2 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
3. │ 3 │         6 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
4. │ 4 │        24 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
5. │ 5 │       120 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
6. │ 6 │       720 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
7. │ 7 │      5040 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
8. │ 8 │     40320 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
9. │ 9 │    362880 │
   └───┴───────────┘
    ┌──i─┬─factorial─┐
10. │ 10 │   3628800 │
    └────┴───────────┘
```

ПРИМЕЧАНИЕ

В этом примере начальное значение `1` в поле `factorial` приводится к типу UInt32, чтобы при вычислении факториала не было переполнения типа данных, так как тип по умолчанию — UInt8.

Ограничение по значению

Условием остановки рекурсии также может быть значение в каком-либо поле. Например:

```
WITH RECURSIVE r AS (
    SELECT 1 AS i, toUInt32(1) AS factorial
    UNION ALL
    SELECT i+1 AS i, factorial*i as factorial
    FROM r
    WHERE factorial < 5000
)
SELECT * FROM r;
```

Результат запроса

```
   ┌─i─┬─factorial─┐
1. │ 1 │         1 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
2. │ 2 │         2 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
3. │ 3 │         6 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
4. │ 4 │        24 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
5. │ 5 │       120 │
   └───┴───────────┘
   ┌─i─┬─factorial─┐
6. │ 6 │       720 │
   └───┴───────────┘
```

Агрегатные функции

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
- Комбинаторы
  - Дополнительное условие для агрегирования
  - Агрегирование по уникальным значениям
  - Агрегированные значения при пустых входных данных
  - Агрегирование массивов
  - Агрегирование значений типа Map
  - Работа с агрегатными состояниями
  - Разделение данных на группы перед агрегированием
- Обработка NULL

Обзор

Агрегатная функция вычисляет результат по набору входных значений. Например, можно вычислить сумму, среднее, максимальное или минимальное значение по набору строк.

Полный список предопределенных агрегатных функций с описаниями можно найти в разделе List of Aggregate Functions документации ClickHouse.

Создание таблицы для выполнения тестовых примеров

Чтобы воспроизвести приведенные в статье примеры и посмотреть, как работают различные агрегатные функции, создайте предварительно таблицу `users`:

```
CREATE TABLE users (user_id Int32, name String, age Int32, role String) ENGINE = MergeTree ORDER BY user_id;
```

```
INSERT INTO users VALUES
(1, 'john', 37, 'developer'),
(2, 'alex', 25, 'analyst'),
(3, 'mary', 30, 'manager'),
(4, 'david', 41, 'developer'),
(5, 'andrew', 41, 'analyst'),
(6, 'harry', 29, 'developer'),
(7, 'ann', 35, 'QA engineer'),
(8, 'jack', 27, 'designer');
```

```
┌─user_id─┬─name───┬─age─┬─role────────┐
│       1 │ john   │  37 │ developer   │
│       2 │ alex   │  25 │ analyst     │
│       3 │ mary   │  30 │ manager     │
│       4 │ david  │  41 │ developer   │
│       5 │ andrew │  41 │ analyst     │
│       6 │ harry  │  29 │ developer   │
│       7 │ ann    │  35 │ QA engineer │
│       8 │ jack   │  27 │ designer    │
└─────────┴────────┴─────┴─────────────┘
```

Например, в следующем запросе используется агрегатная функция, которая возвращает возраст самого младшего пользователя из таблицы `users`:

```
SELECT min(age) FROM users;
```

Результат:

```
┌─min(age)─┐
│       25 │
└──────────┘
```

Чтобы использовать агрегатную функцию в выражении `WHERE`, используйте подзапрос. Следующий пример возвращает имя самого младшего пользователя:

```
SELECT name FROM users WHERE age = (SELECT min(age) FROM users);
```

Результат:

```
┌─name─┐
│ alex │
└──────┘
```

Агрегатные функции можно комбинировать с `GROUP BY`. Например, следующий запрос возвращает для каждой роли количество пользователей:

```
SELECT role, count() AS number_of_users FROM users GROUP BY role;
```

Результат:

```
┌─role────────┬─number_of_users─┐
│ manager     │               1 │
│ designer    │               1 │
│ analyst     │               2 │
│ developer   │               3 │
│ QA engineer │               1 │
└─────────────┴─────────────────┘
```

Чтобы фильтровать полученные значения, используйте выражение `HAVING`. Например, следующий запрос возвращает роли, которые назначены более чем одному пользователю:

```
SELECT role, count() AS number_of_users FROM users GROUP BY role HAVING count()>1;
```

Результат:

```
┌─role──────┬─number_of_users─┐
│ analyst   │               2 │
│ developer │               3 │
└───────────┴─────────────────┘
```

ПРИМЕЧАНИЕ

Существует фундаментальное различие между выражениями `WHERE` и `HAVING`. `WHERE` выбирает входные строки перед группировкой и вычислением агрегатных функций, то есть определяет, какие строки включаются в агрегатные вычисления. `HAVING` выбирает строки после группировки и расчета агрегатных функций. `WHERE` не может содержать агрегатные функции, а `HAVING` всегда их содержит. Если вам нужно использовать дополнительные условия, добавьте их в `WHERE`, а не `HAVING`. Этот подход позволяет избежать агрегатных вычислений для строк, которые не соответствуют дополнительным условиям.

Комбинаторы

Чтобы расширить возможности агрегатных функций и адаптировать запросы под различные специализированные задачи без необходимости изменять структуру данных, ADQM поддерживает различные комбинаторы. **Комбинатор** — специальный суффикс, который добавляется к названию агрегатной функции и модифицирует логику работы этой функции. Для одной функции можно использовать несколько комбинаторов одновременно.

Дополнительное условие для агрегирования

Агрегатная функция с комбинатором `If` принимает дополнительный аргумент — условие типа UInt8, по которому определяется, какие строки агрегатная функция будет обрабатывать.

Синтаксис агрегатного выражения: `<aggr_func_name>If(<expr>, <if_condition>)`.

Пример

Следующий запрос возвращает количество пользователей старше 30 лет:

```
SELECT countIf(age>30) AS over_30 FROM users;
```

Результат:

```
┌─over_30─┐
│       4 │
└─────────┘
```

Агрегирование по уникальным значениям

При наличии комбинатора `Distinct`, агрегатная функция учитывает каждое уникальное значение аргумента (или набор уникальных значений для нескольких аргументов) один раз.

Синтаксис агрегатного выражения: `<aggr_func_name>Distinct(<expr>)` или `<aggr_func_name>(DISTINCT <expr>)`.

Пример

Следующий запрос возвращает количество уникальных ролей в таблице:

```
SELECT count(DISTINCT role) AS unique_roles FROM users;
```

Результат:

```
┌─unique_roles─┐
│            5 │
└──────────────┘
```

Агрегированные значения при пустых входных данных

С помощью следующих комбинаторов можно установить, какое значение должна возвращать агрегатная функция в случае, когда она не получает данные на вход:

- `OrDefault` — возвращается значение по умолчанию для соответствующего типа агрегатной функции;
- `OrNull` — результат агрегатной функции преобразуется к типу Nullable. Если на вход агрегатной функции передан пустой набор данных, то возвращается NULL.

Пример:

```
SELECT avg(number), avgOrDefault(number), avgOrNull(number) FROM numbers(0);
```

Результат:

```
┌─avg(number)─┬─avgOrDefault(number)─┬─avgOrNull(number)─┐
│         nan │                    0 │              NULL │
└─────────────┴──────────────────────┴───────────────────┘
```

Агрегирование массивов

Array

Агрегатные функции с комбинатором `Array` принимают аргументы типа Array(T) (массивы) вместо аргументов типа T. При обработке массивов агрегатная функция работает как исходная агрегатная функция по всем элементам массивов.

Синтаксис агрегатного выражения: `<aggr_func_name>Array(<arr>)`, где `<arr>` — массивы элементов (например, столбец с массивами). Если функция принимает несколько аргументов, массивы должны быть одинаковых размеров.

Пример

1. Создайте таблицу, один из столбцов которой содержит массивы:

```
CREATE TABLE test_arrays (id Int32, array Array(UInt32)) ENGINE = MergeTree ORDER BY id;
```

```
INSERT INTO test_arrays VALUES (1, [1, 2, 3]), (2, [2,3,7]), (3, [1,1,1]);
```

```
┌─id─┬─array───┐
│  1 │ [1,2,3] │
│  2 │ [2,3,7] │
│  3 │ [1,1,1] │
└────┴─────────┘
```

2. Используйте функцию `sumArray`, чтобы вычислить сумму элементов всех массивов в столбце типа Array(UInt32):

```
SELECT sumArray(array) AS sum FROM test_arrays;
```

```
┌─sum─┐
│  21 │
└─────┘
```

ForEach

Комбинатор `ForEach` преобразует агрегатную функцию для таблиц в агрегатную функцию для массивов, которая агрегирует соответствующие элементы массивов и возвращает массив результатов.

Синтаксис агрегатного выражения: `<aggr_func_name>ForEach(<arr>)`, где `<arr>` — набор массивов.

Пример

Используйте функцию `maxForEach`, чтобы получить максимальные значения среди соответствующих элементов массивов:

```
SELECT maxForEach(array) FROM test_arrays;
```

Агрегатная функция найдет максимальное значение среди первых элементов массивов и запишет его на первое место результирующего массива. Затем повторит то же самое для вторых и третьих элементов массивов:

```
┌─maxForEach(array)─┐
│ [2,3,7]           │
└───────────────────┘
```

Агрегирование значений типа Map

Чтобы агрегировать значения типа Map, используйте комбинатор `Map` для агрегатных функций. Агрегатная функция с этим комбинатором принимает тип Map как аргумент, агрегирует значения каждого ключа отдельно, используя соответствующую агрегатную функцию, и возвращает результат типа Map.

Пример

1. Создайте таблицу, один из столбцов которой содержит значения типа Map:

```
CREATE TABLE test_map_table (a Int32, b Map(String, UInt64)) ENGINE = MergeTree ORDER BY a;
```

```
INSERT INTO test_map_table VALUES (1, {'key1':1, 'key2':10}), (2, {'key1':2,'key2':20}), (3, {'key1':3,'key2':30});
```

```
┌─a─┬─b───────────────────┐
│ 1 │ {'key1':1,'key2':10} │
│ 2 │ {'key1':2,'key2':20} │
│ 3 │ {'key1':3,'key2':30} │
└───┴─────────────────────┘
```

2. Используйте комбинатор `Map` для агрегатной функции `sum`, чтобы получить сумму значений по ключу `key2`:

```
select sumMap(b)['key2'] from test_map_table;
```

Результат:

```
┌─arrayElement(sumMap(b), 'key2')─┐
│                              60 │
└─────────────────────────────────┘
```

Работа с агрегатными состояниями

ADQM позволяет работать с промежуточными агрегатными состояниями вместо результирующих агрегатных значений, которые занимают больше места. Используйте комбинатор `State` для агрегатных функций, чтобы сохранять промежуточные состояния агрегаций типа AggregateFunction. Чтобы в дальнейшем получить агрегированные данные, необходимо передать промежуточные состояния в качестве аргументов в те же агрегатные функции с комбинаторами:

- `Merge` — доагрегирует данные и возвращает готовое значение;
- `MergeState` — выполняет слияние промежуточных состояний агрегации аналогично комбинатору `Merge`, но возвращает не готовое значение, а промежуточное состояние агрегации аналогично комбинатору `State`.

Для некоторых агрегатных функций (например, `sum`, `max`, `min`, `any`), можно использовать комбинатор `SimpleState`. Агрегатная функция с этим комбинатором возвращает то же значение, что и исходная функция, но типа SimpleAggregateFunction. `SimpleAggregateFunction` имеет лучшую производительность, чем `AggregateFunction` для той же агрегатной функции.

Для хранения состояний агрегации и их автоматического слияния по первичному ключу можно использовать движок таблицы AggregatingMergeTree.

Разделение данных на группы перед агрегированием

Комбинатор `Resample` позволяет поделить данные на группы (например, разбить значения столбца на интервалы) и выполнить агрегацию для каждой группы отдельно.

Синтаксис агрегатного выражения: `<aggr_func_name>Resample(<start>, <end>, <step>)(<aggr_func_params>, <resample_key>)`, где:

- `<resample_key>` — столбец, значения которого используются для разделения данных на интервалы;
- `<start>` — начальное значение полного интервала значений `<resample_key>`;
- `<stop>` — конечное значение полного интервала значений `<resample_key>` (интервал не включает конечное значение — `[<start>, <stop>)`);
- `<step>` — шаг деления полного интервала на подинтервалы. Функция `aggr_func_name` выполняется для каждого из подинтервалов независимо;
- `<aggr_func_params>` — параметры агрегатной функции.

Агрегатная функция с комбинатором `Resample` возвращает массив результатов функции `aggr_func_name` для каждой группы данных.

Пример

Исходная таблица `users`:

```
┌─name───┬─age─┐
│ john   │  37 │
│ alex   │  25 │
│ mary   │  30 │
│ david  │  41 │
│ andrew │  41 │
│ harry  │  29 │
│ ann    │  35 │
│ jack   │  27 │
└────────┴─────┘
```

Следующий запрос возвращает имена пользователей, чей возраст находится в интервалах `[20,35)` и `[35,50)`:

```
SELECT groupArrayResample(20, 50, 15)(name, age) from users;
```

где:

- `name` — столбец с именами пользователей (аргумент агрегатной функции groupArray);
- `age` — столбец для агрегирования имен пользователей по возрасту.

Результат:

```
┌─groupArrayResample(20, 50, 15)(name, age)─────────────┐
│ [['alex','mary','harry','jack'],['john','david','andrew','ann']] │
└───────────────────────────────────────────────────────┘
```

Обработка NULL

Значения NULL в агрегатных функциях игнорируются. Исключением является функция `count()`, которая возвращает общее количество строк в выборке.

Пример

Создайте таблицу со столбцом типа Nullable:

```
CREATE TABLE table_nulls (x Int8, y Nullable(Int8)) ENGINE = MergeTree ORDER BY x;
```

```
INSERT INTO table_nulls VALUES (1, NULL), (2, 1), (3, 2), (4, 2), (5, 3);
```

```
┌─x─┬────y─┐
│ 1 │ NULL │
│ 2 │    1 │
│ 3 │    2 │
│ 4 │    2 │
│ 5 │    3 │
└───┴──────┘
```

Функция `groupArray(y)` формирует массив из значений столбца `y`, не учитывая значения NULL:

```
SELECT groupArray(y) FROM table_nulls;
```

```
┌─groupArray(y)─┐
│ [1,2,2,3]     │
└───────────────┘
```

Функция `count()` возвращает общее количество строк в таблице:

```
SELECT count() FROM table_nulls;
```

```
┌─count()─┐
│       5 │
└─────────┘
```

Функция `count(y)` возвращает количество строк, где значение `y` не NULL:

```
SELECT count(y) FROM table_nulls;
```

```
┌─count(y)─┐
│        4 │
└──────────┘
```

Функция `count(DISTINCT y)` возвращает количество уникальных строк со значениями `y`, не равными NULL:

```
SELECT count(DISTINCT y) FROM table_nulls;
```

```
┌─uniqExact(y)─┐
│            3 │
└──────────────┘
```

Чтобы заменить значения NULL на какое-либо подходящее значение, которое будет учитываться при вычислении агрегатной функции, можно использовать функцию `coalesce`. Например, функция `avg(coalesce(y, 0))` заменяет все значения NULL в столбце `y` на `0` при вычислении среднего значения по столбцу:

```
SELECT avg(y), avg(coalesce(y, 0)) FROM table_nulls;
```

```
┌─avg(y)─┬─avg(coalesce(y, 0))─┐
│      2 │                 1.6 │
└────────┴─────────────────────┘
```

Чтобы не пропускать NULL в агрегатных функциях, можно также использовать функцию `tuple`, например:

```
SELECT groupArray(y), groupArray(tuple(y)).1 FROM table_nulls;
```

```
┌─groupArray(y)─┬─tupleElement(groupArray(tuple(y)), 1)─┐
│ [1,2,2,3]     │ [NULL,1,2,2,3]                        │
└───────────────┴───────────────────────────────────────┘
```

Использование flameGraph

Елена Дворядкина

Содержание

Назад к содержанию

Синтаксис функции flameGraph
Построение flame-графика
Пример

Для глубокого анализа производительности и отладки запросов ADQM/ClickHouse предоставляет системную таблицу system.trace_log, в которую сохраняются трассировки стека, собираемые профилировщиком запросов. Например, с помощью данных этой таблицы можно понять почему запрос выполняется слишком медленно или не так, как ожидалось — изучить последовательность операций в трассировке и найти конкретные функции запроса, которые вызывают проблемы (требуют больше всего времени или ресурсов на выполнение). Эта информация может быть полезна для оптимизации запроса и повышения его производительности.

Для визуального представления трассировок стека в виде flame-графика можно использовать агрегатную функцию `flameGraph`. Эта функция возвращает массив строк по данным таблицы `system.trace_log`, который затем можно передать в утилиту flamegraph.pl (или аналогичный инструмент) для построения flame-графика.

Синтаксис функции flameGraph

Синтаксис функции `flameGraph` в общем виде:

```
flameGraph(<trace>, [<size>], [<ptr>])
```

где:

- `<trace>` — трассировка стека;
- `<size>` — количество выделенной памяти при профилировании по памяти (тип трассировки `Memory`, `MemorySample` или `MemoryPeak`), по умолчанию — `1`;
- `<ptr>` — адрес выделенной памяти, по умолчанию — `0`.

Построение flame-графика

Чтобы построить профиль запроса в виде flame-графика на основе информации из таблицы `system.trace_log`, следуйте приведенным ниже инструкциям.

Предварительно на хостах ADQM необходимо настроить профилировщик запросов ADQM и подготовить инструменты для отрисовки flame-графиков:

1. Включите профилировщик запросов и настройте его с помощью параметров `query_profiler_*` и `memory_profiler_*` (см. Session Settings в документации ClickHouse). Эти параметры можно устанавливать на уровне пользователя ADQM, сессии clickhouse-client или конкретного запроса, который будет профилироваться (подробнее в статье Управление настройками пользователя). Например, рекомендуется назначить параметрам `query_profiler_real_time_period_ns` и/или `query_profiler_cpu_time_period_ns` значение `10000000` для трассировки выполняемых функций C++.
2. Установите пакет `adqm-clickhouse-debuginfo`. Убедитесь, что настройка `allow_introspection_functions` активирована на уровне пользователя, который будет подключаться к ADQM и использовать функцию `flameGraph`. Эту опцию также можно включать на уровне сессии `clickhouse-client`:
   - в пакетном режиме:

     ```
     $ clickhouse-client --allow_introspection_functions=1 -q "..." ...
     ```

   - в интерактивном режиме:

     ```
     SET allow_introspection_functions = 1;
     ```

3. Скачайте скрипт flamegraph.pl на хосты ADQM и дайте разрешение на его выполнение (команда `chmod +x`). Предварительно проверьте, что на хостах установлен Perl.

Теперь можно использовать функцию `flameGraph` и утилиту `flamegraph.pl`, чтобы получить профиль конкретного запроса в виде flame-графика. Например, следующая команда создает flame-график (интерактивный SVG-файл с именем `<flame_cpu>`), визуализирующий трассировки стека по времени CPU для запроса с идентификатором `<query_id>` (`<path_to>` в данном примере — путь к скрипту `flamegraph.pl` на хосте ADQM):

```
$ clickhouse-client \
    -q "SELECT arrayJoin(flameGraph(arrayReverse(trace))) \
        FROM system.trace_log \
        WHERE trace_type = 'CPU' AND query_id = '<query_id>'" \
    | <path_to>/flamegraph.pl  > <flame_cpu>.svg
```

Пример

Данный пример показывает, как с помощью flame-графика можно посмотреть, какие функции и в каком порядке вызываются при выполнении запроса.

1. Создайте таблицу без первичного ключа с двумя столбцами — данные в первом столбце будут сжиматься по алгоритму LZ4, второй столбец будет хранить данные без сжатия:

   ```
   CREATE TABLE test_table (a Int64 CODEC(LZ4), b Int64 CODEC(NONE)) ENGINE = MergeTree ORDER BY tuple();
   ```

2. Вставьте в таблицу 100 миллионов строк с тестовыми значениями:

   ```
   INSERT INTO test_table SELECT 1, 1 FROM numbers(100000000);
   ```

   С помощью следующего запроса можно посмотреть размер хранимых данных и увидеть разницу между столбцами:

   ```
   SELECT
       name,
       formatReadableSize(data_uncompressed_bytes) AS uncompressed_size,
       formatReadableSize(data_compressed_bytes) AS compressed_size,
       round(data_uncompressed_bytes / data_compressed_bytes, 2) AS ratio
   FROM system.columns
   WHERE table = 'test_table';
   ```

   ```
      ┌─name─┬─uncompressed_size─┬─compressed_size─┬─ratio─┐
   1. │ a    │ 762.94 MiB        │ 3.45 MiB        │ 221.4 │
   2. │ b    │ 762.94 MiB        │ 763.23 MiB      │     1 │
      └──────┴───────────────────┴─────────────────┴───────┘
   ```

3. Выполните два отдельных запроса для вычисления суммы значений в каждом столбце:

   ```
   SELECT sum(a) FROM test_table;
   ```

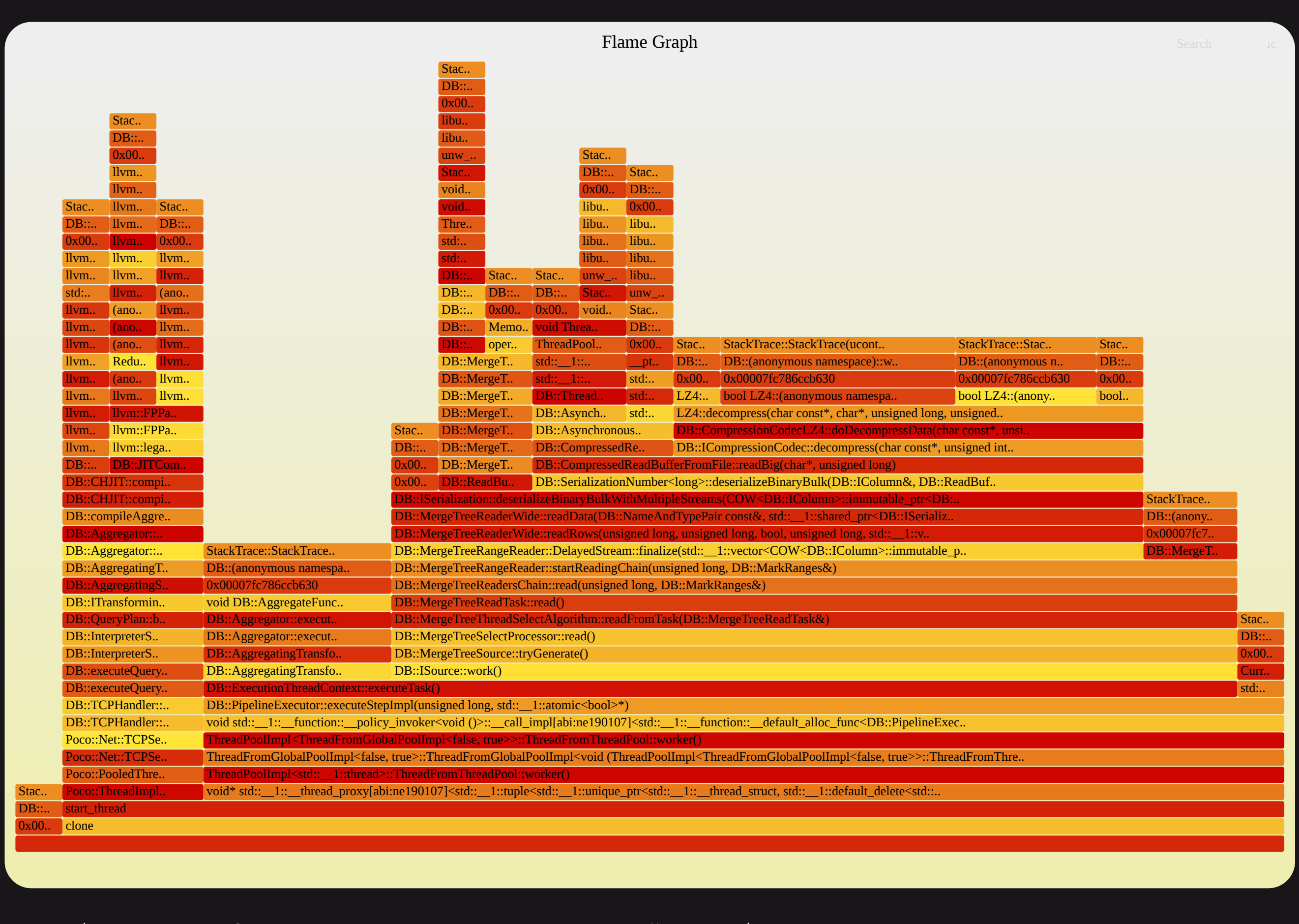
   ```
   Query id: adde6caf-60c5-4820-8bb3-5f75a8edb98e

      ┌────sum(a)─┐
   1. │ 100000000 │ -- 100.00 million
      └───────────┘

   1 row in set. Elapsed: 0.072 sec. Processed 100.00 million rows, 800.00 MB (1.39 billion rows/s., 11.14 GB/s.)
   Peak memory usage: 7.92 MiB.
   ```

   ```
   SELECT sum(b) FROM test_table;
   ```

   ```
   Query id: 8491a039-be0d-4a17-9130-ec286dfee039

      ┌────sum(b)─┐
   1. │ 100000000 │ -- 100.00 million
      └───────────┘

   1 row in set. Elapsed: 0.363 sec. Processed 100.00 million rows, 800.00 MB (275.73 million rows/s., 2.21 GB/s.)
   Peak memory usage: 14.51 MiB.
   ```

4. Постройте flame-график по загрузке CPU для первого запроса:

   ```
   $ clickhouse-client \
       -q "SELECT arrayJoin(flameGraph(arrayReverse(trace))) \
           FROM system.trace_log \
           WHERE trace_type = 'CPU' AND query_id = 'adde6caf-60c5-4820-8bb3-5f75a8edb98e'" \
       | flamegraph.pl  > flamegraph_cpu_1.svg
   ```



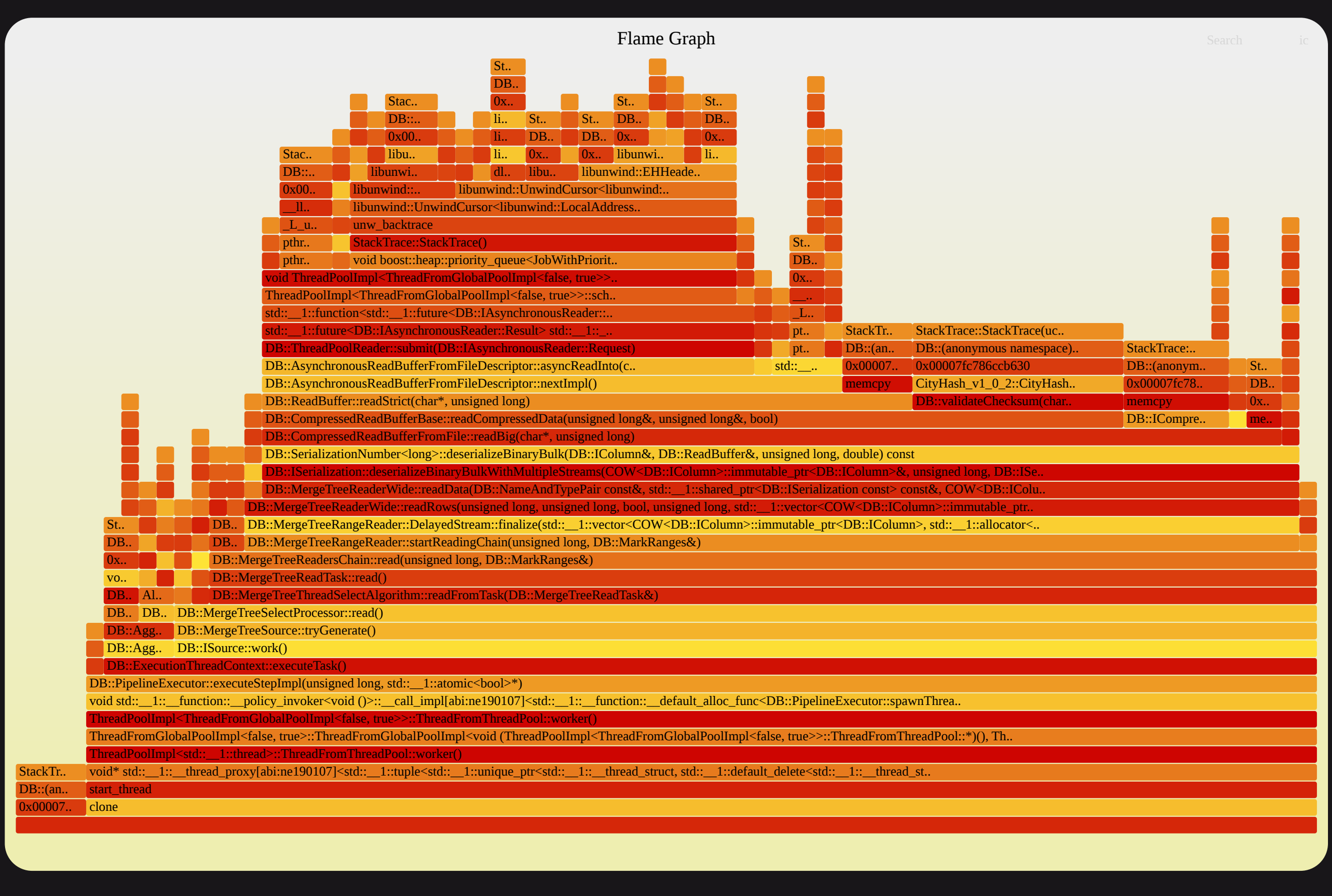
Flame-график загрузки CPU при вычислении суммы значений в столбце, для которого применяется кодек сжатия данных

   На flame-графике показан полный стек вызываемых функций C++. Например, можно увидеть, что в процессе чтения данных выполняется декомпрессия (вызываются функции `decompress`).

5. Постройте flame-график для второго запроса:

   ```
   $ clickhouse-client \
       -q "SELECT arrayJoin(flameGraph(arrayReverse(trace))) \
           FROM system.trace_log \
           WHERE trace_type = 'CPU' AND query_id = '8491a039-be0d-4a17-9130-ec286dfee039'" \
       | flamegraph.pl  > flamegraph_cpu_2.svg
   ```

   По графику видно, что в данном случае также используется класс `CompressedReadBufferBase`, но он уже не вызывает функции декомпрессии (так как данные считываются из столбца, где не применяется сжатие).



Flame-график загрузки CPU для запроса чтения данных из столбца без декомпрессии

Оконные функции

AD.QM

Елена Дворядкина

Содержание

Назад к содержанию

Обзор синтаксиса
  - Описание окна
  - ROWS
  - RANGE

Виды функций

Оконная функция выполняет вычисления по набору строк: все строки в запросе делятся на части (окна), и для каждой части считаются свои агрегаты. В отличие от агрегатной функции, оконная функция не возвращает одну результирующую строку, а добавляет агрегированное значение к каждой строке выборки в отдельном столбце.

Создание таблицы для выполнения тестовых примеров

Чтобы выполнить приведенные в статье примеры и посмотреть, как работают оконные функции, создайте предварительно таблицу `wnd_func_table`:

```
CREATE TABLE wnd_func_table (group_id Int32, sort_id Int32, value Int32) ENGINE = MergeTree ORDER BY group_id;
```

```
INSERT INTO wnd_func_table FORMAT Values
(1,1,10), (1,2,20), (1,3,30), (1,4,40), (1,5,50),
(2,1,1), (2,2,2), (2,3,3), (2,4,4), (2,4,5), (2,4,6), (2,5,7), (2,6,8);
```

```
┌─group_id─┬─sort_id─┬─value─┐
│        1 │       1 │    10 │
│        1 │       2 │    20 │
│        1 │       3 │    30 │
│        1 │       4 │    40 │
│        1 │       5 │    50 │
│        2 │       1 │     1 │
│        2 │       2 │     2 │
│        2 │       3 │     3 │
│        2 │       4 │     4 │
│        2 │       4 │     5 │
│        2 │       4 │     6 │
│        2 │       5 │     7 │
│        2 │       6 │     8 │
└──────────┴─────────┴───────┘
```

Обзор синтаксиса

Синтаксис запроса с использованием оконной функции в общем виде можно записать следующим образом:

```
SELECT <window_function_name>([<argument_list>]) OVER {() | <window_name> | (<window_spec>)}
FROM <table_name>
WINDOW <window_name> AS (<window_spec>);
```

С помощью обязательного выражения `OVER` определяется **окно** — набор записей, по которому вычисляется функция.

Чтобы задать окно для функции, после выражения `OVER` можно указать:

- `()` — пустые круглые скобки, если нужно передать все строки из набора данных в оконную функцию;

  Пример

  Следующий запрос считывает все строки из таблицы `wnd_func_table` и добавляет к результату в дополнительные столбцы `number` и `sum` номера строк и сумму всех значений столбца `value`:

```
SELECT
    group_id,
    sort_id,
    value,
    row_number() OVER() AS number,
    sum(value) OVER() AS sum
FROM wnd_func_table;
```

```
┌─group_id─┬─sort_id─┬─value─┬─number─┬─sum─┐
│        1 │       1 │    10 │      1 │ 186 │
│        1 │       2 │    20 │      2 │ 186 │
│        1 │       3 │    30 │      3 │ 186 │
│        1 │       4 │    40 │      4 │ 186 │
│        1 │       5 │    50 │      5 │ 186 │
│        2 │       1 │     1 │      6 │ 186 │
│        2 │       2 │     2 │      7 │ 186 │
│        2 │       3 │     3 │      8 │ 186 │
│        2 │       4 │     4 │      9 │ 186 │
│        2 │       4 │     5 │     10 │ 186 │
│        2 │       4 │     6 │     11 │ 186 │
│        2 │       5 │     7 │     12 │ 186 │
│        2 │       6 │     8 │     13 │ 186 │
└──────────┴─────────┴───────┴────────┴─────┘
```

- `<window_name>` — имя окна, которое указывается при описании окна с помощью выражения `WINDOW` в запросе (то есть выражение `WINDOW` позволяет описать окно один раз и использовать его для нескольких оконных функций в запросе `SELECT`);
- `<window_spec>` — описание окна в круглых скобках в виде набора опциональных выражений (см. синтаксис ниже).

РЕКОМЕНДАЦИЯ

Если данные запроса нужно отфильтровать по значениям оконных функций, вычисление оконных функций необходимо выполнять в рамках CTE или подзапроса.

Описание окна

Описание окна (`<window_spec>` в приведенном выше синтаксисе запроса) имеет следующий синтаксис:

```
[<window_name>]
[PARTITION BY <partition_column> [, ...]]
[ORDER BY <order_column> [ASC|DESC] [, ...]]
[{RANGE | ROWS} {<frame_start> | BETWEEN <frame_start> AND <frame_end>}]
```

где:

- `PARTITION BY` — столбец, по значениям которого строки запроса разбиваются на окна. Результат выполнения оконной функции для определенной строки будет зависеть от строк окна, содержащего эту строку. Если выражение `PARTITION BY` не указано, функция вычисляется для одного окна, которое содержит все строки.

  Пример

  В следующем запросе с помощью выражения `PARTITION BY` устанавливается столбец, по которому будет производиться группировка данных, то есть разделение набора строк на окна:

```
SELECT
    group_id,
    sort_id,
    value,
    row_number() OVER (w) AS number,
    sum(value) OVER (w) AS sum
FROM wnd_func_table
WINDOW w AS (PARTITION BY group_id);
```

  Функции `row_number` и `sum` вычислились по данным каждого окна отдельно (в этом примере окно — набор строк с одинаковыми значниями в столбце `group_id`):

```
┌─group_id─┬─sort_id─┬─value─┬─number─┬─sum─┐
│        1 │       1 │    10 │      1 │ 150 │
│        1 │       2 │    20 │      2 │ 150 │
│        1 │       3 │    30 │      3 │ 150 │
│        1 │       4 │    40 │      4 │ 150 │
│        1 │       5 │    50 │      5 │ 150 │
│        2 │       1 │     1 │      1 │  36 │
│        2 │       2 │     2 │      2 │  36 │
│        2 │       3 │     3 │      3 │  36 │
│        2 │       4 │     4 │      4 │  36 │
│        2 │       4 │     5 │      5 │  36 │
│        2 │       4 │     6 │      6 │  36 │
│        2 │       5 │     7 │      7 │  36 │
│        2 │       6 │     8 │      8 │  36 │
└──────────┴─────────┴───────┴────────┴─────┘
```

- `ORDER BY` — столбец, по значениям которого строки внутри каждого окна сортируются при вычислении оконной функции.
  Если в описании окна указан `ORDER BY`, функция вычисляется для строк от начала окна до текущей строки (включая все строки с одинаковыми значениями в поле `ORDER BY`) в соответствии с установленным порядком сортировки. То есть для каждой строки выбирается **фрейм** (frame) — набор строк, по которому будет вычисляться функция (например, агрегатные функции, которые используются как оконные, вычисляются на основании фрейма, а не всего окна). Указать другое правило для определения фрейма можно с помощью выражения RANGE или ROWS.

  Пример

  В следующем запросе устанавливается сортировка по столбцу `sort_id`:

```
SELECT
    group_id,
    sort_id,
    value,
    row_number() OVER (w) AS number,
    groupArray(value) OVER (w) AS frame_values,
    sum(value) OVER (w) AS sum
FROM wnd_func_table
WINDOW w AS (PARTITION BY group_id ORDER BY sort_id ASC);
```

  В результирующую таблицу добавляются столбцы с результатами вычисления оконных функций:

  - `number` — порядковый номер строки в окне;
  - `frame_values` — значения столбца `value` в строках, которые входят во фрейм для текущей строки;
  - `sum` — сумма значений столбца `value` в тех строках, которые входят во фрейм для текущей строки (сумма каждого значения `value` со всеми предыдущими).

```
┌─group_id─┬─sort_id─┬─value─┬─number─┬─frame_values──────┬─sum─┐
│        1 │       1 │    10 │      1 │ [10]              │  10 │
│        1 │       2 │    20 │      2 │ [10,20]           │  30 │
│        1 │       3 │    30 │      3 │ [10,20,30]        │  60 │
│        1 │       4 │    40 │      4 │ [10,20,30,40]     │ 100 │
│        1 │       5 │    50 │      5 │ [10,20,30,40,50]  │ 150 │
│        2 │       1 │     1 │      1 │ [1]               │   1 │
│        2 │       2 │     2 │      2 │ [1,2]             │   3 │
│        2 │       3 │     3 │      3 │ [1,2,3]           │   6 │
│        2 │       4 │     4 │      4 │ [1,2,3,4,5,6]     │  21 │
│        2 │       4 │     5 │      5 │ [1,2,3,4,5,6]     │  21 │
│        2 │       4 │     6 │      6 │ [1,2,3,4,5,6]     │  21 │
│        2 │       5 │     7 │      7 │ [1,2,3,4,5,6,7]   │  28 │
│        2 │       6 │     8 │      8 │ [1,2,3,4,5,6,7,8] │  36 │
└──────────┴─────────┴───────┴────────┴───────────────────┴─────┘
```

  Порядок сортировки строк в окне не связан с порядком сортировки на уровне запроса.

ROWS

Выражение `ROWS` позволяет определить фрейм, указав фиксированное количество строк, предшествующих или следующих за текущей строкой, одним из следующих способов:

```
ROWS <frame_start>
ROWS BETWEEN <frame_start> AND <frame_end>
```

Чтобы указать границы фрейма `<frame_start>` и `<frame_end>`, можно использовать следующие выражения.

CURRENT ROW	Фрейм начинается или заканчивается на текущей строке
UNBOUNDED PRECEDING	Фрейм начинается с первой строки окна
UNBOUNDED FOLLOWING	Фрейм заканчивается последней строкой окна
<N> PRECEDING	Фрейм включает `<N>` строк перед текущей строкой
<M> FOLLOWING	Фрейм включает `<M>` строк после текущей строки

Если `<frame_end>` не указан, конец фрейма по умолчанию `CURRENT ROW`.

Пример — ROWS <frame_start>

Фрейм включает две предыдущие и текущую запись:

```
SELECT
    group_id,
    sort_id,
    value,
    groupArray(value) OVER (PARTITION BY group_id ORDER BY sort_id ASC
            ROWS 2 PRECEDING) AS frame_values
FROM wnd_func_table;
```

В столбце `frame_values` показаны значения столбца `value` всех строк, попадающих во фрейм, который формируется для каждой строки отдельно:

```
┌─group_id─┬─sort_id─┬─value─┬─frame_values─┐
│        1 │       1 │    10 │ [10]         │
│        1 │       2 │    20 │ [10,20]      │
│        1 │       3 │    30 │ [10,20,30]   │
│        1 │       4 │    40 │ [20,30,40]   │
│        1 │       5 │    50 │ [30,40,50]   │
│        2 │       1 │     1 │ [1]          │
│        2 │       2 │     2 │ [1,2]        │
│        2 │       3 │     3 │ [1,2,3]      │
│        2 │       4 │     4 │ [2,3,4]      │
│        2 │       4 │     5 │ [3,4,5]      │
│        2 │       4 │     6 │ [4,5,6]      │
│        2 │       5 │     7 │ [5,6,7]      │
│        2 │       6 │     8 │ [6,7,8]      │
└──────────┴─────────┴───────┴──────────────┘
```

Пример — ROWS BETWEEN <frame_start> AND <frame_end>

Фрейм включает текущую строку и все последующие строки до конца окна:

```
SELECT
    group_id,
    sort_id,
    value,
    groupArray(value) OVER (PARTITION BY group_id ORDER BY sort_id ASC
        ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS frame_values
FROM wnd_func_table;
```

```
┌─group_id─┬─sort_id─┬─value─┬─frame_values──────┐
│        1 │       1 │    10 │ [10,20,30,40,50]  │
│        1 │       2 │    20 │ [20,30,40,50]     │
│        1 │       3 │    30 │ [30,40,50]        │
│        1 │       4 │    40 │ [40,50]           │
│        1 │       5 │    50 │ [50]              │
│        2 │       1 │     1 │ [1,2,3,4,5,6,7,8] │
│        2 │       2 │     2 │ [2,3,4,5,6,7,8]   │
│        2 │       3 │     3 │ [3,4,5,6,7,8]     │
│        2 │       4 │     4 │ [4,5,6,7,8]       │
│        2 │       4 │     5 │ [5,6,7,8]         │
│        2 │       4 │     6 │ [6,7,8]           │
│        2 │       5 │     7 │ [7,8]             │
│        2 │       6 │     8 │ [8]               │
└──────────┴─────────┴───────┴───────────────────┘
```

RANGE

Выражение `RANGE` также предназначено для указания фрейма окна. В отличие от `ROWS`, `RANGE` работает не с физическими строками, а с диапазонами строк, определяемых выражением `ORDER BY`. Это означает, что одинаковые по рангу строки (строки с одинаковыми значениями сортировки `ORDER BY`) рассматриваются как одна строка. В выражении `ROWS` каждая физическая запись в наборе данных считается отдельной строкой.

Есть два способа использовать выражение `RANGE`:

```
RANGE <frame_start>
RANGE BETWEEN <frame_start> AND <frame_end>
```

Границы фрейма `<frame_start>` и `<frame_end>` обозначаются с помощью следующих выражений.

CURRENT ROW	Фрейм начинается или заканчивается диапазоном строк, совпадающих с текущей строкой по значениям в столбцах, указанных в выражении `ORDER BY` (все строки диапазона включаются во фрейм)
UNBOUNDED PRECEDING	Фрейм начинается с первой строки окна
UNBOUNDED FOLLOWING	Фрейм заканчивается последней строкой окна
<offset> PRECEDING	Фрейм включает строки со значениями в столбце `ORDER BY`, которые меньше значения столбца `ORDER BY` в текущей строке максимум на значение `<offset>`. Выражение `<offset> PRECEDING` можно использовать, только если в `ORDER BY` указан один столбец
<offset> FOLLOWING	Фрейм включает строки со значениями в столбце `ORDER BY`, которые больше значения столбца `ORDER BY` в текущей строке максимум на значение `<offset>`. Выражение `<offset> FOLLOWING` можно использовать, только если в `ORDER BY` указан один столбец

Если `<frame_end>` не указан, конец фрейма по умолчанию `CURRENT ROW`.

Если в запросе указан `ORDER BY`, но фрейм не описан явно, применяется фрейм по умолчанию: `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`.

Пример — CURRENT ROW

Фрейм включает только текущую строку:

```
SELECT
    group_id,
    sort_id,
    value,
    groupArray(value) OVER (PARTITION BY group_id ORDER BY sort_id ASC
        RANGE BETWEEN CURRENT ROW AND CURRENT ROW) AS frame_values
FROM wnd_func_table;
```

Во втором окне (`group_id=2`) диапазон строк с одинаковым значением в поле `sort_id` рассматривается как одна строка:

```
┌─group_id─┬─sort_id─┬─value─┬─frame_values─┐
│        1 │       1 │    10 │ [10]         │
│        1 │       2 │    20 │ [20]         │
│        1 │       3 │    30 │ [30]         │
│        1 │       4 │    40 │ [40]         │
│        1 │       5 │    50 │ [50]         │
│        2 │       1 │     1 │ [1]          │
│        2 │       2 │     2 │ [2]          │
│        2 │       3 │     3 │ [3]          │
│        2 │       4 │     4 │ [4,5,6]      │
│        2 │       4 │     5 │ [4,5,6]      │
│        2 │       4 │     6 │ [4,5,6]      │
│        2 │       5 │     7 │ [7]          │
│        2 │       6 │     8 │ [8]          │
└──────────┴─────────┴───────┴──────────────┘
```

Пример — <offset> PRECEDING/FOLLOWING

Фрейм для каждой строки включает строки, в которых значение `number` (поле сортировки `ORDER BY`) максимум на `10` меньше и на `5` больше, чем значение `number` в текущей строке:

```
SELECT
    number,
    groupArray(number) OVER (ORDER BY number ASC
        RANGE BETWEEN 10 PRECEDING AND 5 FOLLOWING) AS frame_values
FROM values('number Int8', 10, 20, 25, 27, 30, 40, 15, 50, 60, 7, 5, 2);
```

В результирующей таблице столбец `frame_values` содержит массивы значений `number` для фрейма каждой строки:

```
┌─number─┬─frame_values──────┐
│      2 │ [2,5,7]           │
│      5 │ [2,5,7,10]        │
│      7 │ [2,5,7,10]        │
│     10 │ [2,5,7,10,15]     │
│     15 │ [5,7,10,15,20]    │
│     20 │ [10,15,20,25]     │
│     25 │ [15,20,25,27,30]  │
│     27 │ [20,25,27,30]     │
│     30 │ [20,25,27,30]     │
│     40 │ [30,40]           │
│     50 │ [40,50]           │
│     60 │ [50,60]           │
└────────┴───────────────────┘
```

Виды функций

Оконные функции можно разделить на следующие группы:

- **Агрегатные** — агрегатные функции (например, `sum`, `avg`, `count`, `min`, `max`) могут быть использованы как оконные.
- **Ранжирующие** — функции, которые ранжируют строки в окне. Например, такие функции можно использовать, чтобы присвоить порядковый номер строке или составить рейтинг строк.

Функция	Возвращаемое значение
row_number()	Номер строки внутри окна. Нумерация начинается с `1` и не зависит от того, есть ли в строках повторяющиеся значения в столбце `ORDER BY`
rank()	Ранг строки. Для строк с одинаковыми значениями в столбце `ORDER BY` возвращается одинаковый ранг, при этом для следующих строк пропускается `N-1` значений ранга, где `N` — число строк в предыдущей группе строк с одинаковыми значениями в столбце, по которому выполняется упорядочивание
dense_rank()	Ранг строки. Для строк с одинаковыми значениями в столбце `ORDER BY` возвращается одинаковый ранг, при этом числовое значение ранга для следующих строк не пропускается, то есть ранжирование идет последовательно

- **Функции смещения** — функции, которые позволяют обращаться к разным строкам фрейма относительно текущей строки, а также к значениям строк в начале или в конце фрейма.

Функция	Возвращаемое значение
lagInFrame(value, [N]) leadInFrame(value, [N])	Значение столбца `value` в строке, которая находится на смещении `N` перед или после текущей строки внутри фрейма. Значение `N` по умолчанию равно `1`
first_value(value) last_value(value)	Значение столбца `value` в первой или последней строке фрейма
nth_value(value, N)	Значение столбца `value` в строке, которая находится на позиции `N` внутри фрейма

Табличные функции

Елена Дворядкина

Содержание

Назад к содержанию

Функции

Примеры использования

Табличные функции создают объекты таблиц в ADQM на основе данных из внешних источников и позволяют выполнять операции чтения и записи данных с помощью следующих запросов:

- `SELECT … FROM <table_function>` — создание временной таблицы, доступной только в текущем запросе (таблица удаляется после завершения выполнения запроса);
- `CREATE TABLE [<db_name>.]<table_name> AS <table_function>` — создание таблицы с той же структурой и данными, как результат выполнения соответствующей табличной функции (созданная таблица будет работать так же, как указанная табличная функция);
- `INSERT INTO TABLE FUNCTION <table_function> …` — вставка данных в таблицу, на которую ссылается табличная функция.

> **ВАЖНО**
>
> Чтобы использовать табличные функции, включите настройку allow_ddl.

Функции

ADQM поддерживает множество табличных функций для интеграции с различными внешними системами. Примеры приведены ниже.

Функция	Возвращаемое значение
file	Таблица для чтения/записи данных локального файла
url	Таблица для чтения/записи данных на удаленном HTTP/HTTPS сервере по указанному адресу
hdfs, hdfsCluster	Таблица для чтения/записи данных файла в HDFS
mysql	Таблица для чтения/записи данных, хранящихся на удаленном сервере MySQL
postgresql	Таблица для чтения/записи данных таблицы удаленной базы данных PostgreSQL
mongodb	Таблица для чтения данных с удаленного сервера MongoDB
s3, s3Cluster	Таблица для чтения/записи данных файла в Amazon S3
jdbc	Таблица, подключенная к источнику данных через JDBC-драйвер
odbc	Таблица, подключенная к источнику данных через ODBC
cluster	Набор данных с кластера, сконфигурированного в секции `remote_servers` (*config.xml*)
remote	Набор данных с удаленных серверов ClickHouse

Полный список поддерживаемых табличных функций с описаниями можно найти в разделе Table Functions документации ClickHouse.

Примеры использования

Ниже приведены варианты использования табличной функции на примере функции `file`, с помощью которой можно считывать или записывать данные указанного файла. Перед выполнением тестовых запросов подготовьте файл-источник:

1. В конфигурационном файле *config.xml* укажите каталог */var/lib/clickhouse/user_files* в качестве значения параметра `user_files_path`. Выполните проверку:

```
$ sudo grep user_files_path /etc/clickhouse-server/config.xml
    <user_files_path>/var/lib/clickhouse/user_files/</user_files_path>
```

2. Создайте файл с данными в формате CSV в папке */var/lib/clickhouse/user_files*:

```
$ sudo cat > /var/lib/clickhouse/user_files/file_func_source.csv
1,2,3
4,5,6
7,8,9
```

Выполните следующие запросы, чтобы посмотреть, как работает функция `file`:

1. `CREATE TABLE`

```
CREATE TABLE file_table AS file('file_func_source.csv', 'CSV', 'column_1 UInt32, column_2 UInt32, column_3 UInt32');
```

   В результате таблица с указанной структурой и данными из файла будет добавлена в текущую базу данных.

2. `SELECT … FROM`
   Оба следующих запроса возвращают таблицу с данными из CSV-файла:

```
SELECT * FROM file_table;
```

```
SELECT * FROM file('file_func_source.csv', 'CSV', 'column_1 UInt32, column_2 UInt32, column_3 UInt32');
```

   Вывод на экран:

```
┌─column_1─┬─column_2─┬─column_3─┐
│        1 │        2 │        3 │
│        4 │        5 │        6 │
│        7 │        8 │        9 │
└──────────┴──────────┴──────────┘
```

3. `INSERT INTO TABLE FUNCTION`
   Следующий запрос выполняет вставку данных в источник:

```
INSERT INTO FUNCTION file('file_func_source.csv', 'CSV', 'column_1 UInt32, column_2 UInt32, column_3 UInt32')
    VALUES (10, 20, 30), (40, 50, 60);
```

   Убедитесь, что данные записались в файл:

```
$ sudo cat /var/lib/clickhouse/user_files/file_func_source.csv
1,2,3
4,5,6
7,8,9
10,20,30
40,50,60
```

   При вызове табличной функции `file` новые строки также попадут в выборку:

```
SELECT * FROM file_table;
```

```
┌─column_1─┬─column_2─┬─column_3─┐
│        1 │        2 │        3 │
│        4 │        5 │        6 │
│        7 │        8 │        9 │
│       10 │       20 │       30 │
│       40 │       50 │       60 │
└──────────┴──────────┴──────────┘
```

> **ПРИМЕЧАНИЕ**
>
> Чтобы получить информацию по основным запросам для работы с таблицами данных в ADQM, можно обратиться к статье Обзор таблиц.

Тип данных JSON

ADQM

Елена Дворядкина

Содержание

Назад к содержанию

- Тип данных JSON
  - Выборка данных из динамических столбцов
  - Изменение данных
  - Неструктурированные данные
  - Добавление первичных ключей
- Сопоставление столбцов таблицы с полями JSON
- Функции для работы с JSON
- Импорт данных JSON
  - Форматы импорта данных JSON
  - Импорт объектов JSON в строковом формате
  - Пропуск неизвестных столбцов
  - Импорт JSON с вложенными объектами
  - Автоматическое определение типов данных при импорте JSON
- Экспорт данных JSON
  - Форматы вывода данных JSON
  - Экспорт данных JSON в файл

Для работы с данными JSON в ADQM можно:

- использовать тип данных JSON (доступен с версии 22.3);
- явно определить столбцы таблицы, сопоставляя их с полями JSON;
- сохранить объекты JSON как строки и использовать специальные функции для извлечения значений.

Данная статья описывает примеры использования этих подходов с помощью тестового набора данных о людях (`persons`), который включает поля разных типов (`name`, `age`, `hire_date`, `address`) и может быть по-разному структурирован в формате JSON.

Тип данных JSON

Начиная с версии 22.3, ADQM поддерживает тип данных **Object(JSON)** (сокращенное название типа — **JSON**), который позволяет автоматически создавать столбцы таблицы подходящих типов на основе входящих данных в формате JSON. То есть схема таблицы может динамически изменяться в зависимости от изменения/обновления данных.

> ВАЖНО
>
> В настоящее время тип данных JSON находится в стадии разработки, но его можно использовать в экспериментальном режиме. Для этого необходимо перед созданием таблицы включить опцию: `SET allow_experimental_object_type = 1`.

Пример

Создайте файл *persons.json* с данными в формате JSON в папке */var/lib/clickhouse/user_files*:

```
[
  {"name":"john", "age":42, "hire_date":"2021-01-01", "address": {"country":"USA", "city":"New York"}},
  {"name":"mary", "age":25, "hire_date":"2022-02-02", "address": {"country":"UK", "city":"London"}},
  {"name":"andrew", "age":33, "hire_date":"2023-03-03", "address": {"country":"Germany", "city":"Berlin"}}
]
```

При импорте данных в ADQM используйте формат `JSONAsObject`, чтобы указать, что каждый объект JSON будет хранится как отдельная строка таблицы в столбце типа JSON.

Например, в следующем запросе указывается, что объекты JSON из файла *persons.json* запишутся в столбец `person` типа JSON (если столбец не указать явно, будет использоваться столбец с именем по умолчанию — `json`):

```
SELECT * FROM file('persons.json', 'JSONAsObject', 'person JSON');
```

Результат выполнения запроса:

```
┌─person──────────────────────────────────────────────────────────────────────────────────────────────────┐
│ {"address.city":"New York","address.country":"USA","age":42,"hire_date":"2021-01-01","name":"john"}     │
│ {"address.city":"London","address.country":"UK","age":25,"hire_date":"2022-02-02","name":"mary"}        │
│ {"address.city":"Berlin","address.country":"Germany","age":33,"hire_date":"2023-03-03","name":"andrew"} │
└─────────────────────────────────────────────────────────────────────────────────────────────────────────┘
```

Чтобы эффективно оперировать этими данными, рекомендуется создать таблицу MergeTree и сохранить в нее данные через запрос `INSERT INTO`.

Выполните последовательно следующие команды:

```
SET allow_experimental_object_type = 1;
```

```
CREATE TABLE json_table (person JSON) ENGINE = MergeTree ORDER BY tuple();
```

```
INSERT INTO json_table SELECT * FROM file('persons.json', 'JSONAsObject', 'person JSON');
```

Выполните запрос `DESCRIBE`, чтобы убедиться, что таблица содержит один столбец типа JSON:

```
DESCRIBE json_table;
```

```
┌─name───┬─type───────────┐
│ person │ Object('json') │
└────────┴────────────────┘
```

С помощью настройки `describe_extend_object_types` можно посмотреть, какие столбцы динамически создадутся в таблице при выполнении к ней запросов — эти столбцы будут соответствовать полям объектов JSON:

```
DESCRIBE json_table SETTINGS describe_extend_object_types=1;
```

```
┌─name───┬─type───────────────────────────────────────────────────────────────────────────────────────┐
│ person │ Tuple(address Tuple(city String, country String), age Int8, hire_date String, name String) │
└────────┴────────────────────────────────────────────────────────────────────────────────────────────┘
```

> ПРИМЕЧАНИЕ
>
> Обратите внимание, динамически создаваемые столбцы не могут быть использованы в качестве первичных ключей или ключей сортировки, так как ADQM не знает об этих столбцах на этапе создания таблицы. Поэтому в запросе `CREATE TABLE` используется выражение `ORDER BY tuple()`, то есть таблица не имеет первичного ключа. В разделе Добавление первичных ключей описано, как обойти это ограничение.

Выборка данных из динамических столбцов

К динамическим столбцам таблицы можно обращаться в запросах. Например:

```
SELECT person.name, person.address.country FROM json_table;
```

```
┌─person.name─┬─person.address.country─┐
│ john        │ USA                    │
│ mary        │ UK                     │
│ andrew      │ Germany                │
└─────────────┴────────────────────────┘
```

При выборке данных из динамических столбцов их можно использовать для сортировки и агрегирования данных.

Например, следующий запрос выводит список строк, отсортированный по именам людей:

```
SELECT person.name, person.address.city FROM json_table ORDER BY person.name;
```

```
┌─person.name─┬─person.address.city─┐
│ andrew      │ Berlin              │
│ john        │ New York            │
│ mary        │ London              │
└─────────────┴─────────────────────┘
```

Изменение данных

Если в исходном файле изменяются поля объектов JSON (например, добавляются новые поля или меняется тип существующих), ADQM учтет это при вставке данных и автоматически настроит столбцы таблицы в соответствии с новой структурой объектов JSON.

Добавление нового поля

1. Добавьте в исходный файл *persons.json* столбец `id`:

```
[
  {"id":1, "name":"john", "age":42, "hire_date":"2021-01-01", "address": {"country":"USA", "city":"New York"}},
  {"id":2, "name":"mary", "age":25, "hire_date":"2022-02-02", "address": {"country":"UK", "city":"London"}},
  {"id":3, "name":"andrew", "age":33, "hire_date":"2023-03-03", "address": {"country":"Germany", "city":"Berlin"}}
]
```

2. Вставьте данные в таблицу:

```
INSERT INTO json_table SELECT * FROM file('persons.json', 'JSONAsObject', 'person JSON');
```

3. Убедитесь, что новый столбец `id` типа Int8 добавлен в схему таблицы:

```
DESCRIBE json_table SETTINGS describe_extend_object_types=1;
```

```
┌─name───┬─type────────────────────────────────────────────────────────────────────────────────────────────────┐
│ person │ Tuple(address Tuple(city String, country String), age Int8, hire_date String, id Int8, name String) │
└────────┴─────────────────────────────────────────────────────────────────────────────────────────────────────┘
```

Изменения типа поля

1. Измените в исходном файле JSON значения поля `id` с числовых на строковые, например:

```
[
  {"id":"user_1", "name":"john", "age":42, "hire_date":"2021-01-01", "address": {"country":"USA", "city":"New York"}},
  {"id":"user_2", "name":"mary", "age":25, "hire_date":"2022-02-02", "address": {"country":"UK", "city":"London"}},
  {"id":"user_3", "name":"andrew", "age":33, "hire_date":"2023-03-03", "address": {"country":"Germany", "city":"Berlin"}}
]
```

2. Вставьте данные в таблицу:

```
INSERT INTO json_table SELECT * FROM file('persons.json', 'JSONAsObject', 'person JSON');
```

3. Убедитесь, что тип столбца `id` автоматически изменился на String:

```
DESCRIBE json_table SETTINGS describe_extend_object_types=1;
```

```
┌─name───┬─type──────────────────────────────────────────────────────────────────────────────────────────────────┐
│ person │ Tuple(address Tuple(city String, country String), age Int8, hire_date String, id String, name String) │
└────────┴───────────────────────────────────────────────────────────────────────────────────────────────────────┘
```

Неструктурированные данные

Если данные JSON неструктурированы (например, объекты JSON имеют разные наборы полей), при вставке данных ADQM учтет все поля и заполнит поля, для которых не определены значения, значениями по умолчанию для соответствующих типов данных (например, `0` для чисел, пустые строки для строк).

Пример

В исходном файле *persons.json* измените структуру первого объекта JSON — удалите поле `address.city` и добавьте новое поле `position`:

```
[
  {"name":"john", "age":42, "hire_date":"2021-01-01", "address": {"country":"USA"}, "position":"manager"},
  {"name":"mary", "age":25, "hire_date":"2022-02-02", "address": {"country":"UK", "city":"London"}},
  {"name":"andrew", "age":33, "hire_date":"2023-03-03", "address": {"country":"Germany", "city":"Berlin"}}
]
```

Создайте новую таблицу и вставьте в нее данные из файла:

```
SET allow_experimental_object_type = 1;
```

```
CREATE TABLE unstruct_json_table (person JSON) ENGINE = MergeTree ORDER BY tuple();
```

```
INSERT INTO unstruct_json_table SELECT * FROM file('persons.json', 'JSONAsObject', 'person JSON');
```

Проверьте значения столбцов таблицы:

```
SELECT person.name, person.position, person.address.city FROM unstruct_json_table;
```

```
┌─person.name─┬─person.position─┬─person.address.city─┐
│ john        │ manager         │                     │
│ mary        │                 │ London              │
│ andrew      │                 │ Berlin              │
└─────────────┴─────────────────┴─────────────────────┘
```

Значения по умолчанию в столбцах таблицы учитываются при вычислении агрегатных функций. Например, следующий запрос с функцией `groupArray` возвращает массив значений столбца `person.address.city`, где первое значение — пустая строка, так как первый объект JSON не содержит поле `city`:

```
SELECT groupArray(person.address.city) FROM  unstruct_json_table;
```

```
┌─groupArray(person.address.city)─┐
│ ['','London','Berlin']          │
└─────────────────────────────────┘
```

Добавление первичных ключей

Как отмечалось выше, динамически создаваемые столбцы нельзя использовать в качестве первичных ключей или ключей сортировки. При создании таблицы рекомендуется использовать тип JSON для полуструктурированных частей данных, которые потенциально могут быть изменены в источнике JSON, но явно объявить отдельные столбцы для данных, тип и структуру которых можно однозначно определить, и использовать эти столбцы как первичные ключи.

Пример

В исходном файле данных *persons.json* добавьте поле `person` для хранения объектов JSON:

```
[
  {"person":{"address":{"city":"New York","country":"USA"},"age":42,"hire_date":"2021-01-01","name":"john"}},
  {"person":{"address":{"city":"London","country":"UK"},"age":25,"hire_date":"2022-02-02","name":"mary"}},
  {"person":{"address":{"city":"Berlin","country":"Germany"},"age":33,"hire_date":"2023-03-03","name":"andrew"}}
]
```

Создайте таблицу со столбцами:

- `name` — строковые значения JSON-поля `name`, которые извлекаются с помощью функции `JSONExtractString` (этот столбец используется как ключ сортировки);
- `person` — объекты JSON.

```
SET allow_experimental_object_type = 1;
```

```
CREATE TABLE sort_key_json_table (name String DEFAULT JSONExtractString(toJSONString(person), 'name'), person JSON)
ENGINE = MergeTree
ORDER BY name;
```

Вставьте данные из файла *persons.json*:

```
INSERT INTO sort_key_json_table (name, person) SELECT
    JSONExtractString(toJSONString(person), 'person.name'),
    person
FROM file('persons.json', 'JSONAsObject', 'person JSON');
```

В результирующей таблице строки отсортированы по столбцу `name`:

```
SELECT * FROM sort_key_json_table;
```

```
┌─name───┬─person────────────────────────────────────────────┐
│ andrew │ ((('Berlin','Germany'),33,'2023-03-03','andrew')) │
│ john   │ ((('New York','USA'),42,'2021-01-01','john'))     │
│ mary   │ ((('London','UK'),25,'2022-02-02','mary'))        │
└────────┴───────────────────────────────────────────────────┘
```

Сопоставление столбцов таблицы с полями JSON

ADQM может работать с данными JSON без использования типа данных Object(JSON), что актуально для всех версий ADQM (в том числе более ранних, чем 22.3). Для этого рекомендуется создать таблицу с явно определенными столбцами, сопоставляя их со всеми полями JSON. В этом случае необходимо заранее знать структуру данных JSON и типы полей, а также следить за тем, чтобы схема таблицы оставалась актуальной, если добавляются/изменяются поля в источнике данных JSON. Для вложенных структур JSON можно использовать типы данных Tuple, Map и Nested при определении столбцов таблицы.

Такой подход является наиболее оптимальным, если данные JSON имеют известную фиксированную структуру, так как может обеспечить лучшую производительность, а также позволяет управлять кодеками сжатия и гибко настраивать первичные ключи и ключи сортировки.

Пример

Для примера подготовьте файл */var/lib/clickhouse/user_files/persons.json* с данными JSON без вложенных структур:

```
[
  {"name":"john", "age":42, "hire_date":"2021-01-01"},
  {"name":"mary", "age":25, "hire_date":"2022-02-02"},
  {"name":"andrew", "age":33, "hire_date":"2023-03-03"}
]
```

Создайте таблицу со столбцами соответствующих типов для каждого поля JSON:

```
CREATE TABLE struct_json_table (name String, age Int32, hire_date Date) ENGINE = MergeTree ORDER BY name;
```

Вставьте в таблицу данные из файла, используя табличную функцию `file`:

```
INSERT INTO struct_json_table SELECT * FROM file('persons.json', 'JSONEachRow');
```

В результате в столбцы таблицы запишутся данные из соответствующих JSON-полей:

```
SELECT * FROM struct_json_table;
```

```
┌─name───┬─age─┬──hire_date─┐
│ andrew │  33 │ 2023-03-03 │
│ john   │  42 │ 2021-01-01 │
│ mary   │  25 │ 2022-02-02 │
└────────┴─────┴────────────┘
```

В этом примере для импорта данных JSON используется один из поддерживаемых форматов — `JSONEachRow`. В разделе Форматы импорта данных JSON приведены и описаны другие форматы, предоставляемые ADQM.

Примеры создания столбцов типа Nested, Tuple и Map для работы с вложенными структурами JSON приведены в разделе Handle as Structured Data документации ClickHouse.

Функции для работы с JSON

ADQM позволяет сохранять объекты JSON как строки и предоставляет специальные функции для парсинга JSON-based строк и извлечения значений полей.

Например, следующий запрос интерпретирует объекты JSON из указанного файла как строки (используется формат `JSONAsString`), извлекает значения полей `name` и `age` и записывает их в соответствующие столбцы:

```
SELECT
    JSONExtractString(json, 'name') AS name,
    JSONExtractUInt(json, 'age') AS age
FROM (SELECT * FROM file('persons.json', JSONAsString) AS json);
```

Результат:

```
┌─name───┬─age─┐
│ john   │  42 │
│ mary   │  25 │
│ andrew │  33 │
└────────┴─────┘
```

Полный список поддерживаемых функций представлен в статье документации ClickHouse json-functions.

Импорт данных JSON

ADQM предоставляет различные форматы и настройки для импорта данных JSON. Чтобы выбрать подходящие опции, необходимо определить, как входные данные структурированы.

Форматы импорта данных JSON

В таблице ниже приведены примеры, как при импорте JSON выбрать формат для парсинга входящих данных в зависимости от их структуры.

Для выполнения примеров создайте предварительно таблицу:

```
CREATE TABLE import_json_table(name String, age Int32, hire_date Date) ENGINE = MergeTree ORDER BY hire_date;
```

В приведенных примерах запросы из столбца ***Импорт данных*** вставляют данные в таблицу `import_json_table` из локального файла *persons.json* (расположен в папке, из которой запущен clickhouse-client), содержимое которого описано в столбце ***Входные данные JSON***.

Форматы импорта JSON

Входные данные JSON | **Импорт данных**

Массив объектов JSON:

```
[
  {"name":"john", "age":42, "hire_date":"2021-01-01"},
  {"name":"mary", "age":25, "hire_date":"2022-02-02"},
  {"name":"andrew", "age":33, "hire_date":"2023-03-03"}
]
```

Используйте формат `JSONEachRow`:

```
INSERT INTO import_json_table
FROM INFILE 'persons.json' FORMAT JSONEachRow;
```

NDJSON (newline-delimited JSON):

```
{"name":"john","age":42,"hire_date":"2021-01-01"}
{"name":"mary","age":25,"hire_date":"2022-02-02"}
{"name":"andrew","age":33,"hire_date":"2023-03-03"}
```

Используйте формат `JSONEachRow`:

```
INSERT INTO import_json_table
FROM INFILE 'persons.json' FORMAT JSONEachRow;
```

Массивы JSON:

```
["john", 42, "2021-01-01"],
["mary", 25, "2022-02-02"],
["andrew", 33, "2023-03-03"]
```

Используйте формат `JSONCompactEachRow`:

```
INSERT INTO import_json_table
FROM INFILE 'persons.json' FORMAT JSONCompactEachRow;
```

Один объект JSON, который содержит поля с массивами значений:

```
{
  "name": ["john", "mary", "andrew"],
  "age": ["42", "25", "33"],
  "hire_date": ["2021-01-01", "2022-02-02", "2023-03-03"]
}
```

Используйте формат `JSONColumns`:

```
INSERT INTO import_json_table
FROM INFILE 'persons.json' FORMAT JSONColumns;
```

Массив с массивами JSON:

```
[
  ["john", "mary", "andrew"],
  ["42", "25", "33"],
  ["2021-01-01", "2022-02-02", "2023-03-03"]
]
```

Используйте формат `JSONCompactColumns`:

```
INSERT INTO import_json_table
FROM INFILE 'persons.json' FORMAT JSONCompactColumns;
```

В результате каждого из перечисленных выше примеров данные импортируются в таблицу как:

```
┌─name───┬─age─┬──hire_date─┐
│ john   │  42 │ 2021-01-01 │
│ mary   │  25 │ 2022-02-02 │
│ andrew │  33 │ 2023-03-03 │
└────────┴─────┴────────────┘
```

Полный список форматов для импорта данных JSON доступен в статье Formats for Input and Output Data (см. форматы с префиксом `JSON`).

Импорт объектов JSON в строковом формате

Используйте формат `JSONAsString`, чтобы импортировать объекты JSON в таблицу ADQM как строки (без парсинга). Данные запишутся в единственный столбец таблицы типа String.

Если в исходном файле несколько разделенных запятой объектов JSON, то каждый из них будет считаться отдельной строкой таблицы. Входные данные в квадратных скобках считаются массивом объектов JSON.

Пример

1. Подготовьте данные в исходном локальном файле *persons.json*:

```
[
  {"name":"john", "age":42, "hire_date":"2021-01-01"},
  {"name":"mary", "age":25, "hire_date":"2022-02-02"},
  {"name":"andrew", "age":33, "hire_date":"2023-03-03"}
]
```

2. Создайте таблицу со столбцом типа String:

```
CREATE TABLE json_as_str_table(data String) ENGINE = MergeTree ORDER BY tuple();
```

3. Вставьте данные, используя формат `JSONAsString`:

```
INSERT INTO json_as_str_table FROM INFILE 'persons.json' FORMAT JSONAsString;
```

4. Результат:

```
SELECT * FROM json_as_str_table;
```

```
┌─data──────────────────────────────────────────────────┐
│ {"name":"john", "age":42, "hire_date":"2021-01-01"}   │
│ {"name":"mary", "age":25, "hire_date":"2022-02-02"}   │
│ {"name":"andrew", "age":33, "hire_date":"2023-03-03"} │
└───────────────────────────────────────────────────────┘
```

Для дальнейшей обработки объектов JSON, записанных в виде строк, используйте функции для работы с JSON.

Например, в следующем запросе функция `JSONExtractString` анализирует строковые записи объектов JSON в столбце `data` и возвращает строковые значения поля `name`:

```
SELECT JSONExtractString(data, 'name') AS person_name, data FROM json_as_str_table;
```

```
┌─person_name─┬─data──────────────────────────────────────────────────┐
│ john        │ {"name":"john", "age":42, "hire_date":"2021-01-01"}   │
│ mary        │ {"name":"mary", "age":25, "hire_date":"2022-02-02"}   │
│ andrew      │ {"name":"andrew", "age":33, "hire_date":"2023-03-03"} │
└─────────────┴───────────────────────────────────────────────────────┘
```

Пропуск неизвестных столбцов

По умолчанию ADQM игнорирует неизвестные столбцы при импорте данных JSON.

Например, если создать таблицу с двумя столбцами `name` и `hire_date` и вставить в нее объекты JSON с тремя полями `name`, `hire_date` и `age` (данные JSON из раздела выше), поле `age` проигнорируется.

```
CREATE TABLE short_table(name String, hire_date Date) ENGINE = MergeTree ORDER BY hire_date;
```

```
INSERT INTO short_table FROM INFILE 'persons.json' FORMAT JSONEachRow;
```

```
SELECT * FROM short_table;
```

```
┌─name───┬──hire_date─┐
│ john   │ 2021-01-01 │
│ mary   │ 2022-02-02 │
│ andrew │ 2023-03-03 │
└────────┴────────────┘
```

Отключить это поведение можно с помощью настройки `input_format_skip_unknown_fields`:

```
SET input_format_skip_unknown_fields = 0;
```

В этом случае ADQM сгенерирует исключение, если структуры таблицы и JSON не согласованы:

```
INSERT INTO short_table FROM INFILE 'persons.json' FORMAT JSONEachRow;
```

```
Exception on client:
Code: 117. DB::Exception: Unknown field found while parsing JSONEachRow format: age: (at row 1)
```

Импорт JSON с вложенными объектами

При импорте объектов JSON с вложенными структурами используйте сложные типы данных (JSON, Array или Tuple) для столбцов с вложенными объектами.

Пример

1. Подготовьте файл */var/lib/clickhouse/user_files/persons.json*:

```
[
  {"name":"john", "age":42, "hire_date":"2021-01-01", "address": {"country":"USA", "city":"New York"}},
  {"name":"mary", "age":25, "hire_date":"2022-02-02", "address": {"country":"UK", "city":"London"}},
  {"name":"andrew", "age":33, "hire_date":"2023-03-03", "address": {"country":"Germany", "city":"Berlin"}}
]
```

2. Выполните выборку данных с помощью табличной функции `file`, указав в параметрах структуру таблицы следующим образом:

```
SELECT * FROM file('persons.json', JSONEachRow, 'name String, age Int32, hire_date Date, address JSON');
```

3. В результирующей таблице столбец `address` содержит вложенные объекты JSON:

```
┌─name───┬─age─┬──hire_date─┬─address───────────────────────────────┐
│ john   │  42 │ 2021-01-01 │ {"city":"New York","country":"USA"}   │
│ mary   │  25 │ 2022-02-02 │ {"city":"London","country":"UK"}      │
│ andrew │  33 │ 2023-03-03 │ {"city":"Berlin","country":"Germany"} │
└────────┴─────┴────────────┴───────────────────────────────────────┘
```

Поля вложенных объектов

Чтобы ссылаться на поля вложенных объектов, нужно включить опцию `input_format_import_nested_json`:

```
SET input_format_import_nested_json = 1
```

После этого в запросах можно указывать поля вложенных объектов JSON через точку, оборачивая их в символы обратных кавычек, например, чтобы вывести соответствующие значения в отдельные столбцы.

Пример запроса:

```
SELECT * FROM file('persons.json', JSONEachRow, 'name String, `address.city` String');
```

```
┌─name───┬─address.city─┐
│ john   │ New York     │
│ mary   │ London       │
│ andrew │ Berlin       │
└────────┴──────────────┘
```

Автоматическое определение типов данных при импорте JSON

ADQM может автоматически определять типы данных при импорте данных JSON. Эта функциональность дает возможность быстро создать таблицу со столбцами, типы данных которых соответствуют типам полей в источнике JSON.

Пример

1. Подготовьте файл-источник */var/lib/clickhouse/user_files/persons.json*:

```
{"name":"john","age":42,"hire_date":"2021-01-01"}
{"name":"mary","age":25,"hire_date":"2022-02-02"}
{"name":"andrew","age":33,"hire_date":"2023-03-03"}
```

2. Выполните запрос `DESCRIBE`, чтобы проверить, какие типы данных определены для значений в файле *persons.json*:

```
DESCRIBE TABLE file('persons.json', JSONEachRow);
```

```
┌─name──────┬─type─────────────┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ name      │ Nullable(String) │              │                    │         │                  │                │
│ age       │ Nullable(Int64)  │              │                    │         │                  │                │
│ hire_date │ Nullable(Date)   │              │                    │         │                  │                │
└───────────┴──────────────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

3. Создайте таблицу, используя табличную функцию `file`:

```
CREATE TABLE persons ENGINE = MergeTree ORDER BY tuple() AS SELECT * FROM file('persons.json', JSONEachRow);
```

4. Убедитесь, что столбцы таблицы соответствуют типам полей импортируемого JSON:

```
DESCRIBE TABLE persons;
```

```
┌─name──────┬─type─────────────┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ name      │ Nullable(String) │              │                    │         │                  │                │
│ age       │ Nullable(Int64)  │              │                    │         │                  │                │
│ hire_date │ Nullable(Date)   │              │                    │         │                  │                │
└───────────┴──────────────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

Экспорт данных JSON

Большинство форматов для работы с JSON, которые поддерживает ADQM, можно использовать для вывода данных в запросах `SELECT`.

Форматы вывода данных JSON

Ниже приведены примеры использования различных форматов для экспорта данных в формате JSON из таблицы `export_json_table`:

```
┌─name───┬─age─┬──hire_date─┐
│ john   │  42 │ 2021-01-01 │
│ mary   │  25 │ 2022-02-02 │
│ andrew │  33 │ 2023-03-03 │
└────────┴─────┴────────────┘
```

Создание таблицы

```
CREATE TABLE export_json_table (`name` String, `age` Int32, `hire_date` Date) ENGINE = MergeTree ORDER BY hire_date;
```

```
INSERT INTO export_json_table VALUES ('john', 42, '2021-01-01'), ('mary', 25, '2022-02-02'), ('andrew', 33, '2023-03-03');
```

- **JSONEachRow**
  Выводит каждую строку таблицы как отдельный JSON-объект на новой строке.

```
SELECT * FROM export_json_table FORMAT JSONEachRow;
```

```
{"name":"john","age":42,"hire_date":"2021-01-01"}
{"name":"mary","age":25,"hire_date":"2022-02-02"}
{"name":"andrew","age":33,"hire_date":"2023-03-03"}
```

- **JSONStringsEachRow**
  Отличается от `JSONEachRow` только тем, что выводит значения полей в виде строк, а не типизированных значений объектов JSON.

  Например, в результате выполнения следующего запроса значения поля `age` выводятся как строки, а не числа:

```
SELECT * FROM export_json_table FORMAT JSONStringsEachRow;
```

```
{"name":"john","age":"42","hire_date":"2021-01-01"}
{"name":"mary","age":"25","hire_date":"2022-02-02"}
{"name":"andrew","age":"33","hire_date":"2023-03-03"}
```

- **JSONCompactEachRow**
  Выводит строки таблицы как массивы JSON — массивы значений без имен столбцов.

```
SELECT * FROM export_json_table FORMAT JSONCompactEachRow;
```

```
["john", 42, "2021-01-01"]
["mary", 25, "2022-02-02"]
["andrew", 33, "2023-03-03"]
```

- **JSON**
  Выводит данные в формате JSON. Кроме данных таблицы, также выводятся имена и типы столбцов, и некоторая дополнительная информация (например, количество выведенных строк и общее количество строк, которое могло бы быть выведено, если бы в запросе не было выражения `LIMIT`).

```
SELECT * FROM export_json_table LIMIT 2 FORMAT JSON;
```

```
{
        "meta":
        [
                {
                        "name": "name",
                        "type": "String"
                },
                {
                        "name": "age",
                        "type": "Int32"
                },
                {
                        "name": "hire_date",
                        "type": "Date"
                }
        ],

        "data":
        [
                {
                        "name": "john",
                        "age": 42,
                        "hire_date": "2021-01-01"
                },
                {
                        "name": "mary",
                        "age": 25,
                        "hire_date": "2022-02-02"
                }
        ],

        "rows": 2,

        "rows_before_limit_at_least": 3,

        "statistics":
        {
                "elapsed": 0.000559985,
                "rows_read": 3,
                "bytes_read": 59
        }
}
```

- **JSONCompact**
  Выводит те же метаданные, но использует сжатую форму для самих данных.

```
SELECT * FROM export_json_table LIMIT 2 FORMAT JSONCompact;
```

```
{
        "meta":
        [
                {
                        "name": "name",
                        "type": "String"
                },
                {
                        "name": "age",
                        "type": "Int32"
                },
                {
                        "name": "hire_date",
                        "type": "Date"
                }
        ],

        "data":
        [
                ["john", 42, "2021-01-01"],
                ["mary", 25, "2022-02-02"]
        ],

        "rows": 2,

        "rows_before_limit_at_least": 3,

        "statistics":
        {
                "elapsed": 0.000571299,
                "rows_read": 3,
                "bytes_read": 59
        }
}
```

Полный список форматов для экспорта в JSON доступен в статье Formats for Input and Output Data документации ClickHouse.

Экспорт данных JSON в файл

Чтобы сохранить данные JSON в файл, можно использовать выражение `INTO OUTFILE` в запросе `SELECT`, например:

```
SELECT * FROM export_json_table INTO OUTFILE 'out.json' FORMAT JSONEachRow;
```

Проверьте, что файл создан (в папке, из которой запущен clickhouse-client) и данные в него записаны:

```
$ cat out.json
{"name":"john","age":42,"hire_date":"2021-01-01"}
{"name":"mary","age":25,"hire_date":"2022-02-02"}
{"name":"andrew","age":33,"hire_date":"2023-03-03"}
```

Вложенные структуры данных

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
  - flatten_nested = 1
  - flatten_nested = 0
- Пример
  - Создание таблицы
  - Вставка данных
  - Чтение данных

ADQM/ClickHouse предоставляет тип данных Nested для хранения и обработки вложенных структур данных, который может служить альтернативой типам Tuple и Array(Tuple). Он позволяет моделировать сложные структуры данных без необходимости создания/хранения дополнительных таблиц и использования сложных запросов.

Вложенная структура данных Nested представляет собой таблицу внутри ячейки основной таблицы. Каждой строке основной таблицы может соответствовать любое количество строк во вложенной структуре данных.

Обзор

Чтобы описать в таблице столбец типа Nested, необходимо указать параметры вложенной структуры — имена столбцов и соответствующие типы данных.

Ниже приведен пример синтаксиса запроса CREATE TABLE для создания таблицы с вложенной структурой данных в столбце (с целью упрощения в примере используются краткие названия столбцов и конкретные типы данных):

```
CREATE TABLE test_table (
    a String,
    b UInt32,
    ...
    n Nested (
        c String,
        d Int64,
        ...
    )
)
ENGINE = MergeTree ORDER BY tuple();
```

К столбцам вложенной структуры данных можно обращаться в запросах, используя формат `<nested_structure>.<nested_column>` (где `<nested_structure>` — название столбца таблицы типа Nested, `<nested_column>` — название столбца вложенной структуры).

Формат данных столбцов вложенной структуры определяется настройкой `flatten_nested`, значение которой может быть равным `1` (по умолчанию) или `0`.

flatten_nested = 1

Столбцы вложенной структуры имеют тип Array(Type), то есть хранят массивы значений соответствующих типов данных одинаковой длины. В этом случае при чтении и обработке значений вложенных столбцов можно использовать функции для работы с массивами, в том числе выражение ARRAY JOIN.

Посмотреть, какие типы данных определены для столбцов созданной таблицы, можно с помощью запроса `SHOW CREATE TABLE`:

```
SHOW CREATE TABLE test_table;
```

```
   ┌─statement─────────────────────────┐
1. │ CREATE TABLE default.test_table   │
   │ (                                 │
   │     `a` String,                   │
   │     `b` UInt32,                   │
   │     `n.c` Array(String),          │
   │     `n.d` Array(Int64)            │
   │ )                                 │
   │ ENGINE = MergeTree                │
   │ ORDER BY tuple()                  │
   │ SETTINGS index_granularity = 8192 │
   └───────────────────────────────────┘
```

Вставка данных

Для вставки данных в столбец типа Nested необходимо отдельно передать массив значений для каждого столбца вложенной структуры (система будет автоматически проверять, что все предаваемые массивы значений имеют одинаковую длину). Например:

```
INSERT INTO test_table VALUES ('a', 1, ['c1', 'c2', 'c3'], [100, 200, 300]);
```

В таблицу со столбцами, которые содержат вложенные структуры, можно также вставлять данные JSON с той же структурой. Например, используйте для этого в запросах формат JSONEachRow:

```
INSERT INTO test_table FORMAT JSONEachRow {"a":"a1", "b":2, "n.c":["c4", "c5"], "n.d":[101, 202]};
```

Чтобы передать данные как иерархический объект JSON, предварительно установите `input_format_import_nested_json=1`:

```
SET input_format_import_nested_json=1;
```

```
INSERT INTO test_table FORMAT JSONEachRow
{
    "a":"a2",
    "b":"3",
    "n": {
        "c": ["c6", "c7"],
        "d": [111, 222]
    }
};
```

Чтение данных

Запрос всех данных из таблицы выводит отдельно каждый столбец вложенной структуры:

```
SELECT * FROM test_table;
```

```
   ┌─a──┬─b─┬─n.c──────────────┬─n.d───────────┐
1. │ a  │ 1 │ ['c1','c2','c3'] │ [100,200,300] │
2. │ a1 │ 2 │ ['c4','c5']      │ [101,202]     │
3. │ a2 │ 3 │ ['c6','c7']      │ [111,222]     │
   └────┴───┴──────────────────┴───────────────┘
```

flatten_nested = 0

Данные всех столбцов вложенной структуры хранятся как один массив кортежей, то есть фактически столбец Nested имеет тип Array(Tuple). В этом случае поддерживается произвольный уровень вложенности.

Если пересоздать таблицу test_table, установив перед этим `flatten_nested = 0`, в выводе запроса `SHOW CREATE TABLE` для столбца `n` будет показан тип Nested, хотя фактически это Array(Tuple(...)):

```
   ┌─statement──────────────────────────┐
1. │ CREATE TABLE default.test_table    │
   │ (                                  │
   │     `a` String,                    │
   │     `b` UInt32,                    │
   │     `n` Nested(c String, d Int64)  │
   │ )                                  │
   │ ENGINE = MergeTree                 │
   │ ORDER BY tuple()                   │
   │ SETTINGS index_granularity = 8192  │
   └────────────────────────────────────┘
```

Вставка данных

Соответственно данные во вложенную структуру необходимо вставлять как массив кортежей. Например:

```
INSERT INTO test_table VALUES ('a', 1, [('c1', 100), ('c2', 200), ('c3', 300)]);
```

Для вставки в таблицу с Nested-столбцом иерархического объекта JSON при `flatten_nested = 0` не обязательно активировать опцию `input_format_import_nested_json`, при этом данные для вложенных столбцов нужно передавать как массив:

```
INSERT INTO test_table FORMAT JSONEachRow
{
    "a":"a1",
    "b":"2",
    "n": [
        {"c": "c4", "d": 101},
        {"c": "c5", "d": 202}
        ]
};
```

Чтение данных

Запрос всех данных из таблицы выводит столбец типа Nested, где каждая строка — массив кортежей значений столбцов вложенной структуры:

```
SELECT * FROM test_table;
```

```
   ┌─a──┬─b─┬─n─────────────────────────────────┐
1. │ a  │ 1 │ [('c1',100),('c2',200),('c3',300)] │
2. │ a1 │ 2 │ [('c4',101),('c5',202)]            │
   └────┴───┴────────────────────────────────────┘
```

Данные отдельных вложенных столбцов можно также запрашивать, используя нотацию с точкой:

```
SELECT n.c, n.d FROM test_table;
```

```
   ┌─n.c──────────────┬─n.d───────────┐
1. │ ['c1','c2','c3'] │ [100,200,300] │
2. │ ['c4','c5']      │ [101,202]     │
   └──────────────────┴───────────────┘
```

Пример

Создание таблицы

Создайте таблицу `orders` для хранения данных о заказах в столбцах соответствующих типов:

- `order_id` (UInt64) — идентификатор заказа;
- `customer_id` (UInt32) — идентификатор покупателя;
- `order_date` (Date) — дата заказа.

Добавьте также в таблицу столбец для хранения информации о продуктах, которые включает каждый заказ — столбец `items` типа Nested с описанием следующих полей вложенной структуры:

- `product_id` (UInt32) — идентификатор продукта;
- `product_name` (String) — название продукта;
- `quantity` (UInt16) — количество единиц продукта в заказе;
- `price_per_unit` (Decimal) — стоимость единицы продукта.

Значение параметра `flatten_nested` — `1` (по умолчанию).

```
CREATE TABLE orders (
    order_id UInt64,
    customer_id UInt32,
    order_date Date,
    items Nested (
        product_id UInt32,
        product_name String,
        quantity UInt16,
        price_per_unit Decimal(10, 2)
    )
) ENGINE = MergeTree()
ORDER BY (order_date, order_id);
```

Вставка данных

Вставьте в таблицу тестовые данные (например, несколько заказов за два дня):

```
INSERT INTO orders VALUES
(1001, 42, '2025-08-07',
    [101, 102, 103],
    ['Laptop', 'Mouse', 'Keyboard'],
    [1, 2, 1],
    [999.99, 19.99, 49.99]),
(1002, 77, '2025-08-07',
    [104, 105],
    ['Monitor', 'Headphones'],
    [2, 1],
    [199.99, 89.99]),
(1003, 55, '2025-08-07',
    [101, 104, 102],
    ['Laptop', 'Monitor', 'Mouse'],
    [1, 1, 1],
    [999.99, 199.99, 19.99]),
(1004, 24, '2025-08-08',
    [102, 105],
    ['Mouse', 'Headphones'],
    [3, 4],
    [19.99, 89.99]),
(1005, 38, '2025-08-08',
    [101, 103, 104],
    ['Laptop', 'Keyboard', 'Monitor'],
    [1, 1, 1],
    [999.99, 49.99, 199.99]);
```

Чтение данных

Запросите все данные таблицы:

```
SELECT * FROM orders;
```

```
   ┌─order_id─┬─customer_id─┬─order_date─┬─items.product_id─┬─items.product_name────────────┬─items.quantity─┬─items.
price_per_unit─┐
1. │     1001 │          42 │ 2025-08-07 │ [101,102,103]    │ ['Laptop','Mouse','Keyboard']   │ [1,2,1]        │ [999.99,19.99,49.
99]  │
2. │     1002 │          77 │ 2025-08-07 │ [104,105]        │ ['Monitor','Headphones']        │ [2,1]          │ [199.99,89.99]
│
3. │     1003 │          55 │ 2025-08-07 │ [101,104,102]    │ ['Laptop','Monitor','Mouse']    │ [1,1,1]        │ [999.99,199.
99,19.99] │
4. │     1004 │          24 │ 2025-08-08 │ [102,105]        │ ['Mouse','Headphones']          │ [3,4]          │ [19.99,89.99]
│
5. │     1005 │          38 │ 2025-08-08 │ [101,103,104]    │ ['Laptop','Keyboard','Monitor'] │ [1,1,1]        │ [999.99,49.
99,199.99] │

 └──────────┴─────────────┴────────────┴──────────────────┴─────────────────────────────────┴────────────────┴──────────────────
───┘
```

Чтобы вывести данные в обычном табличном виде, развернув Nested-структуру в несколько строк, используйте в запросе `ARRAY JOIN`:

```
SELECT * FROM orders ARRAY JOIN items;
```

```
    ┌─order_id─┬─customer_id─┬─order_date─┬─items.product_id─┬─items.product_name─┬─items.quantity─┬─items.price_per_unit─┐
 1. │     1001 │          42 │ 2025-08-07 │              101 │ Laptop             │              1 │               999.99 │
 2. │     1001 │          42 │ 2025-08-07 │              102 │ Mouse              │              2 │                19.99 │
 3. │     1001 │          42 │ 2025-08-07 │              103 │ Keyboard           │              1 │                49.99 │
 4. │     1002 │          77 │ 2025-08-07 │              104 │ Monitor            │              2 │               199.99 │
 5. │     1002 │          77 │ 2025-08-07 │              105 │ Headphones         │              1 │                89.99 │
 6. │     1003 │          55 │ 2025-08-07 │              101 │ Laptop             │              1 │               999.99 │
 7. │     1003 │          55 │ 2025-08-07 │              104 │ Monitor            │              1 │               199.99 │
 8. │     1003 │          55 │ 2025-08-07 │              102 │ Mouse              │              1 │                19.99 │
 9. │     1004 │          24 │ 2025-08-08 │              102 │ Mouse              │              3 │                19.99 │
10. │     1004 │          24 │ 2025-08-08 │              105 │ Headphones         │              4 │                89.99 │
11. │     1005 │          38 │ 2025-08-08 │              101 │ Laptop             │              1 │               999.99 │
12. │     1005 │          38 │ 2025-08-08 │              103 │ Keyboard           │              1 │                49.99 │
13. │     1005 │          38 │ 2025-08-08 │              104 │ Monitor            │              1 │               199.99 │
    └──────────┴─────────────┴────────────┴──────────────────┴────────────────────┴────────────────┴──────────────────────┘
```

По столбцам вложенной структуры можно применять агрегирование. Например, следующий запрос вычисляет сколько единиц каждого продукта было продано за месяц (количество по всем заказам):

```
SELECT
    formatDateTime(order_date, '%M') as month,
    items.product_name as product,
    SUM(items.quantity) as units_sold
FROM orders
ARRAY JOIN items
GROUP BY
    formatDateTime(order_date, '%M'),
    items.product_name;
```

```
   ┌─month──┬─product────┬─units_sold─┐
1. │ August │ Keyboard   │          2 │
2. │ August │ Headphones │          5 │
3. │ August │ Laptop     │          3 │
4. │ August │ Mouse      │          6 │
5. │ August │ Monitor    │          4 │
   └────────┴────────────┴────────────┘
```

По полю Nested-структуры можно также фильтровать данные. Например, следующий запрос выводит продажи продукта `Laptop` за каждый день:

```
SELECT
    order_date,
    items.product_name AS product ,
    SUM(items.quantity) AS quantity
FROM orders
ARRAY JOIN items
WHERE items.product_name = 'Laptop'
GROUP BY
    order_date,
    items.product_name;
```

```
   ┌─order_date─┬─product─┬─quantity─┐
1. │ 2025-08-07 │ Laptop  │        2 │
2. │ 2025-08-08 │ Laptop  │        1 │
   └────────────┴─────────┴──────────┘
```

Чтобы вернуть только одну строку с количеством проданных единиц каждого продукта для каждой даты, можно использовать подзапрос и функцию `groupArray`:

```
SELECT
    order_date,
    groupArray((product, units_sold))
FROM
(
    SELECT
        order_date,
        product,
        sum(product_sales) AS units_sold
    FROM orders
    ARRAY JOIN
        items.product_name AS product,
        items.quantity AS product_sales
    GROUP BY
        order_date,
        product
)
GROUP BY order_date;
```

```
   ┌─order_date─┬─groupArray((product, units_sold))───────────────────────────────────────┐
1. │ 2025-08-07 │ [('Laptop',2),('Mouse',3),('Monitor',3),('Keyboard',1),('Headphones',1)] │
2. │ 2025-08-08 │ [('Keyboard',1),('Mouse',3),('Monitor',1),('Headphones',4),('Laptop',1)] │
   └────────────┴──────────────────────────────────────────────────────────────────────────┘
```

Полнотекстовый поиск

Елена Дворядкина

Содержание

Назад к содержанию

Обзор
Управление индексами
Пример

Обзор

Для эффективного полнотекстового поиска по столбцам типа String или FixedString (также поддерживаются типы Array(String), Array(FixedString), Map(String) и Map(String)) в таблицах с очень большим количеством записей можно использовать **инвертированные индексы** — специальный тип индекса пропуска данных.

В ADQM/ClickHouse инвертированный индекс строится следующим образом:

- Текст каждой ячейки в столбце, по которому составляется индекс, разбивается на токены — **термы (terms)**. По умолчанию разбиение осуществляется по пробелам, но также возможно разбить текст на **n-граммы**. Все термы записываются в отсортированный список — **словарь (dictionary)**.
- Для каждого терма составляется отсортированный список идентификаторов строк, содержащих этот терм (**posting list**).

Иными словами, инвертированный индекс реализует хранение сопоставления всех термов с указателями на их расположение в таблице (номера соответствующих строк). Если в таблице для текстового столбца создан инвертированный индекс, ADQM автоматически его применяет при выполнении запроса с поиском терма в этом столбце (например, запрос типа `SELECT … WHERE <column_name> == | IN | LIKE '<term>'`) — быстро находит терм в словаре и возвращает нужные строки таблицы, используя posting list.

Например, рассмотрим таблицу со столбцом `text` типа String:

```
┌─row_id─┬─text────────────┐
│      0 │ Lower East Side │
│      1 │ Upper West Side │
│      2 │ West Village    │
│      3 │ East Village    │
└────────┴─────────────────┘
```

Инвертированный индекс по столбцу `text` в общем виде будет составлен следующим образом.

Dictionary (terms)	Posting lists
East	0, 3
Lower	0
Side	0, 1
Upper	1
Village	2, 3
West	1, 2

ВАЖНО

В настоящее время инвертированные индексы поддерживаются в экспериментальном режиме — не рекомендуется их использовать в production-системах.

Управление индексами

Перед созданием инвертированного индекса необходимо включить следующую опцию:

```
SET allow_experimental_inverted_index = true;
```

Инвертированный индекс описывается так же как и другие типы индекса пропуска данных — с помощью выражения `INDEX` в секции описания столбцов запроса `CREATE TABLE`:

```
CREATE TABLE <table_name>
(   <column_name> <column_type>, ...
    INDEX <index_name> <index_expr> TYPE inverted([<n>], [<max_rows_per_posting_list>]), ...)
ENGINE = MergeTree
...;
```

где:

- `<index_name>` — имя индекса.
- `<index_expr>` — выражение, по которому будет строиться индекс (часто это имя текстового столбца).
- `<n>` — токенизатор (tokenizer), который определяет, как строка в ячейке текстового столбца разбивается на термы:
    - `inverted(0)` (сокращенно `inverted()` или `inverted`) — строка разбивается на слова по пробелам;
    - `inverted(n)` (где `n` — число в диапазоне `[2, 8]`) — строка разбивается на n-граммы (подстроки из `n` символов).
- `<max_rows_per_posting_list>` — максимальное количество идентификаторов строк, содержащих терм, которое может быть записано в список posting list (если `0` — размер списка posting list для терма не ограничивается, если параметр не указан — устанавливается ограничение по умолчанию, равное `64K`).

Так как инвертированные индексы сопоставляют термы с идентификаторами (номерами) строк, а не с идентификаторами гранул, при описании инвертированного индекса не указывается параметр `GRANULARITY`, который возможно применять для других типов индекса пропуска данных.

В существующую таблицу инвертированный индекс можно добавить с помощью следующего запроса:

```
ALTER TABLE <table_name> ADD INDEX <index_name> <index_expr> TYPE inverted([<n>], [<max_rows_per_posting_list>]);
```

Когда индекс добавляется в существующую таблицу, он не обновляется автоматически. `ALTER TABLE` изменяет метаданные, поэтому индекс будет рассчитываться только для новых данных, вставляемых в таблицу. Чтобы применить индекс к существующим данным, выполните дополнительную команду:

```
ALTER TABLE <table_name> MATERIALIZE INDEX <index_name>;
```

Удалить индексы можно запросами:

- `ALTER TABLE <table_name> DROP INDEX <index_name>` — удаляет описание индекса из метаданных таблицы и удаляет индексные файлы с диска;
- `ALTER TABLE <table_name> CLEAR INDEX <index_name>` — удаляет файлы индекса с диска без удаления описания индекса из метаданных.

Пример

Создайте таблицу `trips` с данными такси Нью-Йорка, как описано в примере документации ClickHouse:

```
CREATE TABLE trips (
    trip_id             UInt32,
    pickup_datetime     DateTime,
    dropoff_datetime    DateTime,
    pickup_longitude    Nullable(Float64),
    pickup_latitude     Nullable(Float64),
    dropoff_longitude   Nullable(Float64),
    dropoff_latitude    Nullable(Float64),
    passenger_count     UInt8,
    trip_distance       Float32,
    fare_amount         Float32,
    extra               Float32,
    tip_amount          Float32,
    tolls_amount        Float32,
    total_amount        Float32,
    payment_type        Enum('CSH' = 1, 'CRE' = 2, 'NOC' = 3, 'DIS' = 4, 'UNK' = 5),
    pickup_ntaname      LowCardinality(String),
    dropoff_ntaname     LowCardinality(String)
)
ENGINE = MergeTree
PRIMARY KEY (pickup_datetime, dropoff_datetime);
```

```
INSERT INTO trips
SELECT
    trip_id,
    pickup_datetime,
    dropoff_datetime,
    pickup_longitude,
    pickup_latitude,
    dropoff_longitude,
    dropoff_latitude,
    passenger_count,
    trip_distance,
    fare_amount,
    extra,
    tip_amount,
    tolls_amount,
    total_amount,
    payment_type,
    pickup_ntaname,
    dropoff_ntaname
FROM s3(
    'https://datasets-documentation.s3.eu-west-3.amazonaws.com/nyc-taxi/trips_{0..2}.gz',
    'TabSeparatedWithNames');
```

Выполните тестовый запрос на выборку данных с условием по столбцу `pickup_ntaname`:

```
SELECT pickup_datetime,pickup_datetime,pickup_ntaname,dropoff_ntaname
FROM trips
WHERE (pickup_ntaname LIKE '%Village%') OR (pickup_ntaname LIKE '%Square%');
```

```
┌─────pickup_datetime─┬─────pickup_datetime─┬─pickup_ntaname─────────────────────────────┬─dropoff_ntaname───────┐
│ 2015-07-01 00:02:24 │ 2015-07-01 00:02:24 │ East Village                               │ Gramercy              │
│ 2015-07-01 00:03:15 │ 2015-07-01 00:03:15 │ Hudson Yards-Chelsea-Flatiron-Union Square │ West Village          │

...

│ 2015-07-09 00:32:13 │ 2015-07-09 00:32:13 │ West Village                               │ East Village          │
└─────────────────────┴─────────────────────┴────────────────────────────────────────────┴───────────────────────┘
  Showed first 10000.

714664 rows in set. Elapsed: 0.831 sec. Processed 3.00 million rows, 18.00 MB (3.61 million rows/s., 21.66 MB/s.)
Peak memory usage: 13.04 MiB.
```

Добавьте в таблицу инвертированный индекс:

```
SET allow_experimental_inverted_index = true;
```

```
ALTER TABLE trips ADD INDEX inv_idx(pickup_ntaname) TYPE inverted;
```

```
ALTER TABLE trips MATERIALIZE INDEX inv_idx;
```

Повторите запрос — обратите внимание, что скорость его выполнения увеличилась:

```
SELECT pickup_datetime,pickup_datetime,pickup_ntaname,dropoff_ntaname
FROM trips
WHERE (pickup_ntaname LIKE '%Village%') OR (pickup_ntaname LIKE '%Square%');
```

```
┌─────pickup_datetime─┬─────pickup_datetime─┬─pickup_ntaname─────────────────────────────┬─dropoff_ntaname───────┐
│ 2015-07-01 00:02:24 │ 2015-07-01 00:02:24 │ East Village                               │ Gramercy              │
│ 2015-07-01 00:03:15 │ 2015-07-01 00:03:15 │ Hudson Yards-Chelsea-Flatiron-Union Square │ West Village          │

...

│ 2015-07-09 00:32:13 │ 2015-07-09 00:32:13 │ West Village                               │ East Village          │
└─────────────────────┴─────────────────────┴────────────────────────────────────────────┴───────────────────────┘
  Showed first 10000.

714664 rows in set. Elapsed: 0.129 sec. Processed 3.00 million rows, 18.00 MB (23.27 million rows/s., 139.63 MB/s.)
Peak memory usage: 13.04 MiB.
```

С помощью выражения EXPLAIN с параметром `indexes` можно убедиться, что при выполнении запроса использовался инвертированный индекс:

```
EXPLAIN indexes = 1 SELECT pickup_datetime,pickup_datetime,pickup_ntaname,dropoff_ntaname
FROM trips
WHERE (pickup_ntaname LIKE '%Village%') OR (pickup_ntaname LIKE '%Square%');
```

```
┌─explain─────────────────────────────────────┐
│ Expression ((Project names + Projection))   │
│   Expression                                │
│     ReadFromMergeTree (default.trips)       │
│     Indexes:                                │
│       PrimaryKey                            │
│         Condition: true                     │
│         Parts: 2/2                          │
│         Granules: 368/368                   │
│       Skip                                  │
│         Name: inv_idx                       │
│         Description: inverted GRANULARITY 1 │
│         Parts: 2/2                          │
│         Granules: 368/368                   │
└─────────────────────────────────────────────┘
```

Повышение производительности запросов

Елена Дворядкина

Содержание

Назад к содержанию

Индексы
- Первичный индекс
- Составной ключ первичного индекса
- Индекс пропуска данных (Set)

Строгая типизация данных
- Пример

Сжатие и шифрование данных
- Пример

Кеширование запросов
- Управление кешированием запросов
- Настройка кеша запросов
- Просмотр информации о кеше запросов
- Пример

PREWHERE
- Пример

Оптимизация операций JOIN

В этой статье описываются инструменты, которые предоставляет ADQM для повышения производительности запросов.

Индексы

Высокая производительность аналитических запросов в ADQM, как правило, достигается за счет индексирования таблиц — правильное определение табличных индексов может существенно сократить объем данных, которые необходимо будет считывать с диска для выполнения запросов.

Для таблиц семейства MergeTree ADQM поддерживает следующие типы индексов:

- **Первичный индекс** — позволяет быстро находить диапазоны строк, которые потенциально содержат нужные значения. Ключ, по которому строится первичный индекс, может быть составным, то есть включать несколько столбцов.
- **Индексы пропуска данных** — хранят некоторую дополнительную информацию по указанному столбцу в блоках данных и при выполнении запросов на чтение позволяют пропускать блоки данных, в которых точно нет запрашиваемых значений. Тип информации о содержимом блока данных, которую агрегирует и хранит индекс пропуска данных, определяется типом индекса: MinMax, Set, Bloom Filter или Inverted.

РЕКОМЕНДАЦИЯ

Подробную информацию о том, как работают индексы ADQM и как ими управлять, можно получить в статье Индексы.

Ниже приведены примеры, демонстрирующие как использование первичного индекса и индекса пропуска данных (например, типа Set) влияет на скорость чтения данных из таблицы MergeTree.

Первичный индекс

1. Создайте таблицу без первичного ключа (с одним числовым столбцом и одним строковым) и заполните ее тестовыми данными:

```sql
CREATE TABLE table_no_primary_key (a Int64, b String) ENGINE = MergeTree ORDER BY tuple();
```

```sql
INSERT INTO table_no_primary_key
SELECT toInt64(randUniform(0, 1000)), ['a', 'b', 'c', 'd', 'e'][(rand() % 5) + 1]
FROM numbers(100000000);
```

2. Выполните запрос, который вычисляет количество строк со значением `555` в столбце `a`:

```sql
SELECT count() FROM table_no_primary_key WHERE a = 555;
```

В результате видно, что выполнилось полное сканирование таблицы — ADQM обработал 100 миллионов строк, на что потребовалось примерно 0.5 секунды:

```
┌─count()─┐
│  100162 │
└─────────┘

1 row in set. Elapsed: 0.512 sec. Processed 100.00 million rows, 800.00 MB (195.30 million rows/s., 1.56 GB/s.)
```

3. Чтобы улучшить производительность этого запроса, нужно переопределить таблицу так, чтобы первый столбец был указан в качестве первичного ключа:

```sql
CREATE TABLE table_with_primary_key (a Int64, b String) ENGINE = MergeTree ORDER BY (a);
```

Заполните новую таблицы данными из первой таблицы:

```sql
INSERT INTO table_with_primary_key SELECT * FROM table_no_primary_key;
```

4. Повторите запрос с фильтрацией по столбцу `a`:

```sql
SELECT count() FROM table_with_primary_key WHERE a = 555;
```

В этом случае ADQM обработает 172 тысячи строк вместо 100 миллионов и выполнит запрос значительно быстрее:

```
┌─count()─┐
│  100162 │
└─────────┘

1 row in set. Elapsed: 0.004 sec. Processed 172.03 thousand rows, 1.38 MB (43.90 million rows/s., 351.17 MB/s.)
```

Если для запроса использовать выражение EXPLAIN с параметром `indexes`, то можно увидеть, что в процессе выполнения запроса из 12209 гранул была выбрана только 21, где, возможно, содержатся строки со значением `555` в столбце `a`:

```sql
EXPLAIN indexes = 1 SELECT count() FROM table_with_primary_key WHERE a = 555;
```

```
┌─explain────────────────────────────────────────────────────┐
│ Expression ((Projection + Before ORDER BY))                │
│   Aggregating                                              │
│     Expression (Before GROUP BY)                           │
│       Filter (WHERE)                                       │
│         ReadFromMergeTree (default.table_with_primary_key) │
│         Indexes:                                           │
│           PrimaryKey                                       │
│             Keys:                                          │
│               a                                            │
│             Condition: (a in [555, 555])                   │
│             Parts: 8/8                                     │
│             Granules: 21/12209                             │
└────────────────────────────────────────────────────────────┘
```

Составной ключ первичного индекса

1. Создайте таблицу с составным первичным ключом:

```sql
CREATE TABLE table_compound_primary_key (a Int64, b String) ENGINE = MergeTree ORDER BY (a, b);
```

Заполните таблицу данными:

```sql
INSERT INTO table_compound_primary_key
SELECT toInt64(randUniform(0, 1000)), ['a', 'b', 'c', 'd', 'e'][(rand() % 5) + 1]
FROM numbers(100000000);
```

Обратите внимание, в этом примере столбцы таблицы сильно различаются по кардинальности — первый столбец заполняется целыми числами из диапазона `[0, 1000)`, а во втором содержится всего пять строковых значений.

2. Посмотрите скорость выполнения отдельных запросов с фильтрацией по первому и по второму столбцу первичного ключа:

Фильтрация по столбцу a

```sql
SELECT count()
FROM table_compound_primary_key
WHERE a = 555;
```

```
┌─count()─┐
│  100162 │
└─────────┘

1 row in set. Elapsed: 0.004 sec.
Processed 155.65 thousand rows, 1.25 MB
(38.72 million rows/s., 309.78 MB/s.)
```

Фильтрация по столбцу b

```sql
SELECT count()
FROM table_compound_primary_key
WHERE b = 'c';
```

```
┌──count()─┐
│ 19997249 │
└──────────┘

1 row in set. Elapsed: 1.050 sec.
Processed 44.17 million rows, 441.66 MB
(42.07 million rows/s., 420.70 MB/s.)
```

3. Поменяйте столбцы в первичном ключе местами, упорядочив их по кардинальности от низкой к более высокой, согласно рекомендациям:

```sql
CREATE TABLE table_compound_primary_key1 (a Int64, b String) ENGINE = MergeTree ORDER BY (b, a);
```

```sql
INSERT INTO table_compound_primary_key1 SELECT * from table_compound_primary_key;
```

4. Посмотрите, как изменилась скорость выполнения тех же запросов с фильтрацией по каждому столбцу. Запрос, фильтрующий данные по столбцу `b`, выполняется теперь в два раза быстрее, при этом время выполнения запроса, фильтрующего по столбцу `a`, не увеличилось:

Фильтрация по столбцу a

```sql
SELECT count()
FROM table_compound_primary_key1
WHERE a = 555;
```

```
┌─count()─┐
│  100162 │
└─────────┘

1 row in set. Elapsed: 0.004 sec.
Processed 540.67 thousand rows, 4.33 MB
(123.51 million rows/s., 988.06 MB/s.)
```

Фильтрация по столбцу b

```sql
SELECT count()
FROM table_compound_primary_key1
WHERE b = 'c';
```

```
┌──count()─┐
│ 19997249 │
└──────────┘

1 row in set. Elapsed: 0.566 sec.
Processed 20.01 million rows, 200.13 MB
(35.38 million rows/s., 353.83 MB/s.)
```

Индекс пропуска данных (Set)

1. Создайте тестовую таблицу с 10000 строк в каждой грануле и добавьте в нее 100 миллионов строк данных:

```sql
CREATE TABLE table_skip_index (a UInt64, b UInt64)
ENGINE MergeTree
PRIMARY KEY a
SETTINGS index_granularity=10000;
```

```sql
INSERT INTO table_skip_index SELECT number, intDiv(number,5000) FROM numbers(100000000);
```

2. При выполнении запроса с фильтрацией по столбцу `b`, который не входит в первичный ключ, сканируются все 100 миллионов записей:

```sql
SELECT * FROM table_skip_index WHERE b=100 OR b=555;
```

```
┌──────a─┬───b─┐
│ 500000 │ 100 │
│ 500001 │ 100 │
│ 500002 │ 100 │
│    ... │ ... │
└────────┴─────┘
10000 rows in set. Elapsed: 0.042 sec. Processed 100.00 million rows, 800.12 MB (2.38 billion rows/s., 19.01 GB/s.)
```

3. Добавьте индекс пропуска данных:

```sql
ALTER TABLE table_skip_index ADD INDEX my_index b TYPE set(100) GRANULARITY 2;
```

На этом этапе индекс будет применяться только к новым данным, добавляемым в таблицу. Чтобы проиндексировать уже существующие в таблице данные, выполните следующий запрос:

```sql
ALTER TABLE table_skip_index MATERIALIZE INDEX my_index;
```

4. Повторно выполните запрос на выборку данных с тем же фильтром:

```sql
SELECT * FROM table_skip_index WHERE b=100 OR b=555;
```

Вместо обработки 100 миллионов строк (800 МБ) ADQM прочитает и проанализирует только 40 тысяч строк (440 КБ) — четыре гранулы по 10000 строк каждая.

```
┌──────a─┬───b─┐
│ 500000 │ 100 │
│ 500001 │ 100 │
│ 500002 │ 100 │
│    ... │ ... │
└────────┴─────┘
10000 rows in set. Elapsed: 0.018 sec. Processed 40.00 thousand rows, 440.00 KB (2.19 million rows/s., 24.06 MB/s.)
```

Строгая типизация данных

Чтобы обеспечить высокую производительность запросов и минимизировать объем хранилища, важно проанализировать данные, которые будут храниться в таблице, и назначить каждому столбцу наиболее подходящий тип. Например, если столбец имеет низкую кардинальность (low cardinality), то есть содержит небольшой набор повторяющихся строковых значений, хранение этих значений можно оптимизировать, используя для столбца тип LowCardinality вместо String.

Пример

1. Создайте таблицу с двумя столбцами типа Int64 и String:

```sql
CREATE TABLE table_types (a Int64, b String) ENGINE = MergeTree ORDER BY a;
```

2. Вставьте в таблицу 100 миллионов строк с тестовыми данными (в каждой строке первого столбца хранится целое число из диапазона `[0,1000)`, второго столбца — одна из пяти предопределенных строк):

```sql
INSERT INTO table_types
SELECT toInt64(randUniform(0, 1000)), ['a', 'b', 'c', 'd', 'e'][(rand() % 5) + 1]
FROM numbers(100000000);
```

3. Посмотрите объем несжатых данных:

```sql
SELECT
    name,
    type,
    formatReadableSize(data_uncompressed_bytes) AS uncompressed_size
FROM system.columns
WHERE table = 'table_types';
```

```
┌─name─┬─type───┬─uncompressed_size─┐
│ a    │ Int64  │ 759.91 MiB        │
│ b    │ String │ 189.98 MiB        │
└──────┴────────┴───────────────────┘
```

4. Зная диапазон значений в столбце `a`, можно использовать в схеме таблицы более подходящий тип для целочисленных данных — UInt16 вместо Int64 (см. диапазоны значений, поддерживаемые различными типами данных, в статье UInt*). Хранение строк в столбце `b` также можно оптимизировать, заменив тип String на LowCardinality. Создайте таблицу, аналогичную созданной ранее, но со строгими типами данных для столбцов:

```sql
CREATE TABLE table_strict_types (a UInt16, b LowCardinality(String)) ENGINE = MergeTree ORDER BY a;
```

```sql
INSERT INTO table_strict_types SELECT a, toLowCardinality(b) from table_types;
```

5. Проверьте, что размер данных в столбцах уменьшился:

```sql
SELECT
    name,
    type,
    formatReadableSize(data_uncompressed_bytes) AS uncompressed_size
FROM system.columns
WHERE table = 'table_strict_types';
```

```
┌─name─┬─type───────────────────┬─uncompressed_size─┐
│ a    │ UInt16                 │ 189.98 MiB        │
│ b    │ LowCardinality(String) │ 95.32 MiB         │
└──────┴────────────────────────┴───────────────────┘
```

6. Сравните выполнение одного и того же тестового запроса для первоначальной таблицы `table_types` и таблицы с оптимальными типами столбцов `table_strict_types`:

```sql
SELECT count() FROM table_types WHERE b = 'a';
```

```
┌──count()─┐
│ 20003609 │
└──────────┘

1 row in set. Elapsed: 5.330 sec. Processed 100.00 million rows, 1.00 GB (18.76 million rows/s., 187.61 MB/s.)
```

```sql
SELECT count() FROM table_strict_types WHERE b = 'a';
```

```
┌──count()─┐
│ 20003609 │
└──────────┘

1 row in set. Elapsed: 0.390 sec. Processed 100.00 million rows, 100.12 MB (256.40 million rows/s., 256.71 MB/s.)
```

В результате видно, что оба запроса сканируют одинаковое количество строк, но запрос, обращающийся к первой таблице, считывает объем данных почти в 10 раз больше.

Сжатие и шифрование данных

При выполнении запроса необходимо считывать данные с диска, и чем больше данных считывается, тем медленнее запрос. Объем считываемых данных можно уменьшить с помощью сжатия и шифрования. ADQM поддерживает кодеки сжатия общего назначения, которые позволяют обеспечить оптимальное соотношение между потреблением дискового пространства и ресурсов CPU, набор специализированных кодеков сжатия для определенных типов данных, а также кодеки шифрования. Полный список доступных кодеков сжатия и шифрования приведен в разделе Column Compression Codecs документации ClickHouse.

По умолчанию, ADQM применяет к столбцу метод сжатия, определенный в конфигурации сервера. Кроме того, можно указать метод сжатия для каждого отдельного столбца в запросе `CREATE TABLE` с помощью выражения `CODEC`:

```sql
CREATE TABLE <table_name>
(   <column_name1> <data_type1> CODEC(<compression_codec1>) [...],
    <column_name2> <data_type2> CODEC(<compression_codec2>) [...],
    ...)
ENGINE = MergeTree
...;
```

Кодеки можно последовательно комбинировать (например, `CODEC(Delta, Default)`).

Пример

1. Создайте тестовую таблицу с тремя столбцами — первый и второй столбцы сжимаются по алгоритмам LZ4 и ZSTD соответственно, третий хранит данные без сжатия:

```sql
CREATE TABLE table_sizes (a Int32 CODEC(LZ4), b Int32 CODEC(ZSTD), c Int32 CODEC(NONE))
ENGINE = MergeTree
ORDER BY a;
```

2. Вставьте 100 миллионов строк с одинаковыми значениями во всех столбцах (например, `1`):

```sql
INSERT INTO table_sizes SELECT 1, 1, 1 FROM numbers(100000000);
```

3. В системной таблице `system.columns` можно посмотреть размер хранимых данных и увидеть разницу между столбцами. Первые два сжатых разными кодеками столбца хранят 1.7 МБ и 279 КБ данных соответственно, а несжатый — около 380 МБ:

```sql
SELECT
    name,
    formatReadableSize(data_uncompressed_bytes) AS uncompressed_size,
    formatReadableSize(data_compressed_bytes) AS compressed_size,
    round(data_uncompressed_bytes / data_compressed_bytes, 2) AS ratio
FROM system.columns
WHERE table = 'table_sizes';
```

```
┌─name─┬─uncompressed_size─┬─compressed_size─┬───ratio─┐
│ a    │ 379.95 MiB        │ 1.71 MiB        │  222.15 │
│ b    │ 379.95 MiB        │ 279.06 KiB      │ 1394.21 │
│ c    │ 379.95 MiB        │ 380.10 MiB      │       1 │
└──────┴───────────────────┴─────────────────┴─────────┘
```

4. Запустите запрос, который вычисляет сумму по каждому столбцу, чтобы увидеть разницу в работе со сжатыми и несжатыми столбцами:

```sql
SELECT sum(a) FROM table_sizes;
```

```
┌────sum(a)─┐
│ 100000000 │
└───────────┘

1 row in set. Elapsed: 0.197 sec. Processed 100.00 million rows, 400.00 MB (507.58 million rows/s., 2.03 GB/s.)
```

```sql
SELECT sum(b) FROM table_sizes;
```

```
┌────sum(b)─┐
│ 100000000 │
└───────────┘

1 row in set. Elapsed: 0.091 sec. Processed 100.00 million rows, 400.00 MB (1.09 billion rows/s., 4.38 GB/s.)
```

```sql
SELECT sum(c) FROM table_sizes;
```

```
┌────sum(c)─┐
│ 100000000 │
└───────────┘

1 row in set. Elapsed: 0.389 sec. Processed 100.00 million rows, 400.00 MB (257.00 million rows/s., 1.03 GB/s.)
```

Все три запроса сканируют по 100 миллионов строк, но сканирование сжатых данных выполняется быстрее: 0.197 и 0.091 секунды против 0.389 секунды для несжатых данных.

Кеширование запросов

Еще один способ сократить время выполнения запросов `SELECT`, а также снизить нагрузку на сервер ADQM и оптимизировать потребление ресурсов — кеширование запросов. В этом случае запрос вычисляется один раз, результат сохраняется в кеше и используется в дальнейшем при повторном выполнении того же запроса. Если запрос не выполнился из-за ошибки или отмены пользователем, запись в кеш запросов не добавляется.

Управление кешированием запросов

1. Включите поддержку кеширования запросов в ADQM на уровне сессии:

```sql
SET allow_experimental_query_cache = 1;
```

2. Установите значение настройки `use_query_cache` в `1`, чтобы сохранять в кеш результаты конкретного запроса или всех запросов текущей сессии:

   - `SELECT … SETTINGS use_query_cache = 1;` — кеширование на уровне отдельного запроса;
   - `SET use_query_cache = 1;` — кеширование запросов на уровне сессии.

Последующие выполнения запросов (также с параметром `use_query_cache = 1`) будут считывать вычисленный ранее результат из кеша и немедленно возвращать его.

Для более детальной настройки кеширования можно также использовать параметры (по умолчанию, оба значения `1`):

   - `enable_writes_to_query_cache` — определяет, должны ли результаты запроса `SELECT` сохраняться в кеш запросов;
   - `enable_reads_from_query_cache` — определяет, должны ли результаты запроса `SELECT` извлекаться из кеша запросов.

ПРИМЕЧАНИЕ

Настройка `use_query_cache` и другие параметры, связанные с кешированием запросов, влияют только на автономные запросы `SELECT`.

Очистить кеш запросов можно с помощью команды `SYSTEM DROP QUERY CACHE`.

Настройка кеша запросов

Чтобы обеспечить возможность более детально управлять поведением кеша запросов, ADQM предоставляет набор конфигурационных параметров.

Настройки на уровне сервера

Размер кеша запросов (в байтах), максимальное количество записей в кеше и максимальный размер отдельных записей кеша (в байтах и в строках) можно настроить с помощью соответствующих параметров в секции `query_cache` конфигурационного файла сервера *config.xml*.

```xml
<query_cache>
    <max_size>1073741824</max_size>
    <max_entries>1024</max_entries>
    <max_entry_size>1048576</max_entry_size>
    <max_entry_rows>30000000</max_entry_rows>
</query_cache>
```

Настройки на уровне запроса/сеанса

- Чтобы кешировать только самые частые или затратные с точки зрения времени запросы, можно настроить следующие параметры:
  - `query_cache_min_query_runs` — сколько раз (минимум) должен быть выполнен запрос `SELECT`, прежде чем его результат будет сохранен в кеше запросов;
  - `query_cache_min_query_duration` — минимальная продолжительность запроса (в миллисекундах), результаты которого будут записываться в кеш запросов.
- Записи в кеше запросов устаревают по истечении определенного периода времени (по умолчанию, `60` секунд). Этот период можно настроить с помощью параметра `query_cache_ttl`.
- Записи кеша запросов по умолчанию не передаются между пользователями из соображений безопасности. Однако при необходимости записи кеша можно отметить как доступные для других пользователей (то есть как общие), установив значение параметра `query_cache_share_between_users` в `1`.
- Результаты запросов с недетерминированными функциями (например, `rand()`, `now()`) по умолчанию не кешируются. Чтобы это изменить, используйте параметр `query_cache_store_results_of_queries_with_nondeterministic_functions`.

Просмотр информации о кеше запросов

Содержимое кеша запросов можно посмотреть в системной таблице system.query_cache:

```sql
SELECT * FROM system.query_cache;
```

```
┌─query──────────────────────────────────────────────────┬────────────key_hash─┬──────────expires_at─┬─stale─┬─shared─┬─result_s
ize─┐
│ SELECT count() FROM table_types WHERE b = 'a' SETTINGS  │ 3145443392323842376 │ 2023-10-18 12:58:43 │     1 │      0 │
256 │
└────────────────────────────────────────────────────────┴─────────────────────┴─────────────────────┴───────┴────────┴─────────
────┘
```

В примере выше запись в кеше помечена как устаревшая (stale), так как время, прошедшее с момента кеширования результата запроса, превышает время жизни записи кеша. Это означает, что при следующем запуске запроса не будет использоваться результат кешированного запроса, а вместо этого обновится запись кеша.

Выражение `SETTINGS` в тексте запроса (поле `query`) показывается частично, так как осуществляется внутренняя очистка всех параметров, связанных с кешем запросов, прежде чем запрос будет использоваться в качестве ключа для кеша запросов.

Количество попаданий и промахов кеша запросов с момента запуска сервера учитываются как события `QueryCacheHits` и `QueryCacheMisses` в системной таблице `system.events`. Оба счетчика обновляются только для запросов `SELECT`, которые выполняются с настройкой `use_query_cache = 1`.

```sql
SELECT * FROM system.events WHERE (event = 'QueryCacheHits' OR event = 'QueryCacheMisses');
```

```
┌─event────────────┬─value─┬─description──────────────────────────────────────────────────────────────────────────────────────
───┐
│ QueryCacheHits   │     2 │ Number of times a query result has been found in the query cache (and query computation was
avoided).  │
│ QueryCacheMisses │     4 │ Number of times a query result has not been found in the query cache (and required query
computation). │
└──────────────────┴───────┴──────────────────────────────────────────────────────────────────────────────────────────────────
───┘
```

Пример

1. Выполните запрос `SELECT`, обращающийся к неоптимизированной таблице `table_types`, созданой выше:

```sql
SELECT count() FROM table_types WHERE b = 'a';
```

На выполнение запроса требуется примерно 2.5 секунды:

```
┌──count()─┐
│ 19999789 │
└──────────┘

1 row in set. Elapsed: 2.584 sec. Processed 100.00 million rows, 1.00 GB (38.70 million rows/s., 387.00 MB/s.)
```

2. Подключите возможность кеширования запросов и повторите выполнение запроса с активированной настройкой `use_query_cache`:

```sql
SET allow_experimental_query_cache = 1;
```

```sql
SELECT count() FROM table_types WHERE b = 'a' SETTINGS use_query_cache = 1;
```

```
┌──count()─┐
│ 19999789 │
└──────────┘

1 row in set. Elapsed: 2.924 sec. Processed 100.00 million rows, 1.00 GB (34.20 million rows/s., 341.98 MB/s.)
```

Как видно, на выполнение запроса потребовалось примерно столько же времени (даже чуть больше), поскольку при первом запуске запроса с настройкой `use_query_cache = 1` результат его выполнения запишется в кеш. Последующие выполнения этого же запроса (также с настройкой `use_query_cache = 1`) в пределах интервала времени, указанного в `query_cache_ttl`, будут считывать результат из кеша и возвращать его немедленно:

```sql
SELECT count() FROM table_types WHERE b = 'a' SETTINGS use_query_cache = 1;
```

```
┌──count()─┐
│ 19999789 │
└──────────┘

1 row in set. Elapsed: 0.003 sec.
```

PREWHERE

`PREWHERE` — способ оптимизации запроса `SELECT … WHERE`, позволяющий выполнить фильтрацию данных до операции чтения, тем самым уменьшить объем данных, которые будут считываться из таблицы, и ускорить выполнение запроса. В ADQM оптимизация фильтрации `PREWHERE` применяется автоматически (при включенной по умолчанию настройке `optimize_move_to_prewhere`) — даже если выражение `PREWHERE` явно не указано в запросе, часть условий фильтрации `WHERE` перемещается в `PREWHERE` и проверяется в первую очередь.

Вы можете явно указать в запросе выражение `PREWHERE`, чтобы скорректировать оптимизацию фильтрации вручную. Сначала из запрашиваемой таблицы считываются столбцы, указанные в `PREWHERE`. Затем считываются другие столбцы, необходимые для выполнения остальной части запроса, но только те блоки, в которых условие `PREWHERE` выполняется хотя бы для некоторых строк. Если в большом количестве блоков условие `PREWHERE` не выполняется для всех строк и выражение `PREWHERE` использует меньше столбцов, чем другие части запроса, оптимизация через `PREWHERE` может существенно сократить количество данных, считываемых с диска для выполнения запроса.

В запросе могут быть одновременно указаны выражения `PREWHERE` и `WHERE` (сначала выполнится `PREWHERE`, затем — `WHERE`).

Если в запросе есть модификатор `FINAL`, оптимизация `PREWHERE` не всегда выполняется корректно. Она действует только если включены обе настройки `optimize_move_to_prewhere` и `optimize_move_to_prewhere_if_final`.

Пример

1. Создайте тестовую таблицу с тремя столбцами и добавьте в нее 10 миллионов строк (в первом столбце — целые числа по порядку, во втором — случайные числа в диапазоне `[0, 1000)`, в третьем — значение `test string` в 100 строках, значения для остальных строк сконвертированы из соответствующих числовых идентификаторов в первом столбце):

```sql
CREATE TABLE table_prewhere (id UInt64, int_value Int64, string_value String) ENGINE = MergeTree ORDER BY id;
```

```sql
INSERT INTO table_prewhere SELECT
    number,
    toInt64(randUniform(0, 1000)),
    if(number between 11 and 110, 'test string', toString(number))
FROM numbers(10000000);
```

2. Выполните запрос, фильтрующий данные по столбцам, которые не входят в первичный ключ:

```sql
SELECT count()
FROM table_prewhere
WHERE (int_value > 550) AND (string_value = 'test string');
```

```
┌─count()─┐
│      43 │
└─────────┘
1 row in set. Elapsed: 0.638 sec. Processed 10.00 million rows, 238.89 MB (15.68 million rows/s., 374.58 MB/s.)
```

3. Если перед выполнением запроса включить трассировку командой `SET send_logs_level='debug'`, то можно увидеть, что ADQM автоматически переместил первое условие фильтрации `int_value > 550` в `PREWHERE`:

```
InterpreterSelectQuery: MergeTreeWhereOptimizer: condition "int_value > 550" moved to PREWHERE
```

Таким образом, запрос сначала получает много строк по условию `int_value > 550`, из которых большая часть потом отфильтровывается по условию `string_value = 'test string'`. Чтобы оптимизировать запрос, измените порядок фильтрации, явно передав условие фильтрации по столбцу `string_value` в `PREWHERE`:

```sql
SELECT count()
FROM table_prewhere
PREWHERE string_value = 'test string'
WHERE int_value > 550;
```

В этом случае запрос сначала по условию `string_value = 'test string'` получает небольшой набор строк, который затем проверяется на второе условие `int_value > 550`. В результате обрабатывается меньше данных (158.89 МБ вместо 238.89 МБ) и запрос выполняется быстрее:

```
┌─count()─┐
│      43 │
└─────────┘
1 row in set. Elapsed: 0.249 sec. Processed 10.00 million rows, 158.89 MB (40.12 million rows/s., 637.45 MB/s.)
```

Оптимизация операций JOIN

Операции JOIN в ADQM можно оптимизировать несколькими способами для повышения производительности запросов. Ниже приведены некоторые рекомендации:

- ADQM поддерживает несколько алгоритмов выполнения соединения — выберите алгоритм, наиболее оптимальный для вашего запроса с учетом применяемого типа соединения, строгости и движка объединяемых таблиц. Устанавливается алгоритм соединения с помощью настройки join_algorithm — можно назначить один или несколько алгоритмов (тогда один из доступных алгоритмов будет выбираться для каждого конкретного запроса на основе его типа/строгости и движка таблицы) или разрешить планировщику запросов выбирать и динамически изменять алгоритм во время выполнения запроса в зависимости от доступности ресурсов. Сравнение алгоритмов соединения и рекомендации по выбору алгоритмов, подходящих для разных сценариев, можно найти в блог-посте ClickHouse Choosing the Right Join Algorithm.
- Используйте оптимальные ключи соединения (столбцы, по которым сопоставляются строки между двумя таблицами). В идеале ключи соединения должны быть проиндексированы и иметь низкую кардинальность (небольшое количество уникальных значений). Тип данных ключа также может влиять на производительность (например, использование строкового ключа соединения может быть медленнее, чем использование целочисленного ключа).
- При использовании `JOIN` в запросе отсутствует оптимизация порядка выполнения соединения по отношению к другим стадиям запроса. Соединение (поиск в правой таблице) выполняется до фильтрации `WHERE` и до агрегации. Чтобы явно указать порядок вычислений и повысить производительность запроса, рекомендуется применять фильтр как можно раньше, то есть соединять подзапросы.
- Результаты соединений не кешируются — каждый раз для выполнения запроса с одинаковым `JOIN`, подзапрос соединения выполняется заново. Чтобы этого избежать, можно использовать специальный табличный движок Join, представляющий собой подготовленное множество для соединения, которое всегда находится в оперативной памяти.
- Для небольших объемов данных (до ~200 ГБ в сжатом виде) рекомендуется использовать не меньший объем памяти.
- Иногда вместо `JOIN` более эффективно использовать `IN` или словари. При этом не рекомендуется использовать `JOIN` со словарями, поскольку функции dictGet работают эффективнее.
- В некоторых сценариях эффективнее денормализовать данные путем объединения нескольких таблиц в одну, чтобы полностью исключить использование операций `JOIN` и повысить производительность запросов.

Анализ запросов

Елена Дворядкина

Содержание

Назад к содержанию

- EXPLAIN AST
  - Пример
  - Параметры
- EXPLAIN SYNTAX
  - Пример
- EXPLAIN QUERY TREE
  - Пример
  - Параметры
- EXPLAIN PLAN
  - Пример
  - Параметры
- EXPLAIN PIPELINE
  - Пример
  - Параметры
- EXPLAIN ESTIMATE
  - Пример
- EXPLAIN TABLE OVERRIDE
  - Пример

Чтобы проанализировать, как ADQM преобразует и выполняет запросы, используйте выражение `EXPLAIN`. Базовый синтаксис:

```
EXPLAIN [AST | SYNTAX | QUERY TREE | PLAN | PIPELINE | ESTIMATE | TABLE OVERRIDE] [<setting_name> = <setting_value>, ...]
<query_to_analyze>
[FORMAT <output_format>]
```

где:

- `EXPLAIN [<explain_type>] [<explain_settings>]` — тип анализатора `EXPLAIN` (см. таблицу ниже) и соответствующие ему дополнительные настройки;
- `<query_to_analyze>` — анализируемый запрос;
- `FORMAT <output_format>` — формат вывода результата анализа.

Типы анализатора EXPLAIN

Тип EXPLAIN	Результат выполнения EXPLAIN
AST	Абстрактное синтаксическое дерево запроса
SYNTAX	Оптимизированная версия синтаксиса запроса, на которую ADQM заменит исходный текст запроса при его выполнении
QUERY TREE	Дерево запроса после его оптимизации
PLAN	План выполнения запроса. Этот тип `EXPLAIN` используется по умолчанию
PIPELINE	Конвейер выполнения запроса
ESTIMATE	Оценка числа строк, засечек (marks) и кусков данных, которые будут обработаны при выполнении запроса
TABLE OVERRIDE	Предварительная проверка настроек переопределения схемы таблицы для движка баз данных MaterializedMySQL

EXPLAIN AST

Запрос `EXPLAIN AST` выводит абстрактное синтаксическое дерево (Abstract Syntax Tree, AST) анализируемого запроса. Поддерживается для всех типов запросов, не только `SELECT`.

Пример

```
EXPLAIN AST SELECT sum(number) AS test_sums FROM numbers(5) GROUP BY number % 2;
```

```
┌─explain─────────────────────────────────────────┐
│ SelectWithUnionQuery (children 1)               │
│  ExpressionList (children 1)                    │
│   SelectQuery (children 3)                      │
│    ExpressionList (children 1)                  │
│     Function sum (alias test_sums) (children 1) │
│      ExpressionList (children 1)                │
│       Identifier number                         │
│    TablesInSelectQuery (children 1)             │
│     TablesInSelectQueryElement (children 1)     │
│      TableExpression (children 1)               │
│       Function numbers (children 1)             │
│        ExpressionList (children 1)              │
│         Literal UInt64_5                        │
│    ExpressionList (children 1)                  │
│     Function modulo (children 1)                │
│      ExpressionList (children 2)                │
│       Identifier number                         │
│       Literal UInt64_2                          │
└─────────────────────────────────────────────────┘
```

Параметры

Для запроса `EXPLAIN AST` можно использовать опцию `graph`, чтобы получить абстрактное синтаксическое дерево в виде графа, описанного на языке DOT. Полученный синтаксис можно загрузить в специальное приложение для отрисовки графов (например, Graphviz — пакет утилит с открытым исходным кодом для автоматической визуализации графов).

Пример построения абстрактного синтаксического дерева в виде графа:

```
EXPLAIN AST graph=1 SELECT sum(number) AS test_sums FROM numbers(5) GROUP BY number % 2;
```

```
┌─explain────────────────────────────────────────────────────────────────────┐
│ digraph {                                                                  │
│     rankdir="UD";                                                          │
│     n140208872268568[label="SelectWithUnionQuery (children 1)"];           │
│     n140208877065784[label="ExpressionList (children 1)"];                 │
│     n140205632168472[label="SelectQuery (children 3)"];                    │
│     n140206404082776[label="ExpressionList (children 1)"];                 │
│     n140206025898776[label="Function sum (alias test_sums) (children 1)"]; │
│     n140206026032952[label="ExpressionList (children 1)"];                 │
│     n140205801742360[label="Identifier number"];                           │
│     ...                                                                   │
│ }                                                                          │
└────────────────────────────────────────────────────────────────────────────┘
```

EXPLAIN SYNTAX

При выполнении запросов ADQM может автоматически переписывать их синтаксис для улучшения производительности. Запрос `EXPLAIN SYNTAX` позволяет посмотреть применяемые ADQM синтаксические оптимизации — в дальнейшем их можно использовать, чтобы вручную оптимизировать сложные запросы для повышения скорости их выполнения.

Пример

1. Создайте таблицу типа MergeTree и добавьте в нее 1 миллион строк, указав ключ партиционирования так, чтобы данные поделились на 10 партиций:

   ```
   CREATE TABLE test_table (value UInt64)
   ENGINE = MergeTree
   PARTITION BY value % 10
   ORDER BY value AS
   SELECT number
   FROM numbers(1000000);
   ```

2. Выполните запрос, который вычисляет среднее арифметическое по столбцу `value` для первой партиции:

   ```
   SELECT avg(value)
   FROM (SELECT value FROM test_table)
   WHERE (value % 10) = 1;
   ```

3. В результате выполнения запроса видно, что ADQM обработал только 100 тысяч строк, а не 1 миллион:

   ```
   ┌─avg(value)─┐
   │     499996 │
   └────────────┘

   1 row in set. Elapsed: 0.006 sec. Processed 100.00 thousand rows, 800.00 KB (16.40 million rows/s., 131.16 MB/s.)
   ```

4. Выполните анализ `EXPLAIN SYNTAX` для приведенного выше запроса:

   ```
   EXPLAIN SYNTAX
   SELECT avg(value)
   FROM (SELECT value FROM test_table)
   WHERE (value % 10) = 1;
   ```

   В результате этого анализа видно, что ADQM автоматически оптимизировал запрос, переместив фильтр `WHERE` внутрь подзапроса — поэтому достаточно обработать только 1/10 часть данных таблицы:

   ```
   ┌─explain────────────────────┐
   │ SELECT avg(value)          │
   │ FROM                       │
   │ (                          │
   │     SELECT value           │
   │     FROM test_table        │
   │     WHERE (value % 10) = 1 │
   │ )                          │
   │ WHERE (value % 10) = 1     │
   └────────────────────────────┘
   ```

   Таким образом можно использовать этот способ оптимизации в более сложных запросах, сразу указывая необходимые фильтры в подзапросах.

EXPLAIN QUERY TREE

Запрос `EXPLAIN QUERY TREE` создает дамп дерева запроса. Этот тип анализатора находится в стадии разработки, но его можно использовать в экспериментальном режиме. Для этого перед выполнением запроса необходимо включить опцию: `SET allow_experimental_analyzer = 1`.

Создание таблицы для выполнения приведенных ниже примеров

В примерах запросов `EXPLAIN QUERY TREE`, приведенных ниже, используется тестовая таблица с данными о просмотрах страниц, сгенерированными случайным образом. Чтобы воспроизвести эти примеры, создайте предварительно таблицу `page_views`:

```
CREATE TABLE page_views (time DateTime, page_name String, time_on_page UInt64)
ENGINE = MergeTree
ORDER BY time;
```

```
INSERT INTO page_views
SELECT
    toDate('2023-01-01 00:00:00') + (rand() % 182) AS time,
    ['landing', 'products', 'docs', 'about'][toInt32(randBinomial(4, 0.01)) + 1],
    rand() % 15
FROM numbers(1000000);
```

Пример

```
SET allow_experimental_analyzer = 1;
```

```
EXPLAIN QUERY TREE
SELECT page_name, toDate(toStartOfMonth(time)) AS month, count() AS page_view_count
FROM page_views
WHERE page_name = 'landing'
GROUP BY page_name, month
ORDER BY page_name, month
LIMIT 3;
```

```
┌─explain───────────────────────────────────────────────────────────────────────────────────────────────────┐
│ QUERY id: 0                                                                                               │
│   PROJECTION COLUMNS                                                                                      │
│     page_name String                                                                                      │
│     month Date                                                                                            │
│     page_view_count UInt64                                                                                │
│   PROJECTION                                                                                              │
│     LIST id: 1, nodes: 3                                                                                  │
│       COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3                             │
│       FUNCTION id: 4, function_name: toDate, function_type: ordinary, result_type: Date                   │
│         ARGUMENTS                                                                                         │
│           LIST id: 5, nodes: 1                                                                            │
│             FUNCTION id: 6, function_name: toStartOfMonth, function_type: ordinary, result_type: Date     │
│               ARGUMENTS                                                                                   │
│                 LIST id: 7, nodes: 1                                                                      │
│                   COLUMN id: 8, column_name: time, result_type: DateTime, source_id: 3                    │
│       FUNCTION id: 9, function_name: count, function_type: aggregate, result_type: UInt64                 │
│   JOIN TREE                                                                                               │
│     TABLE id: 3, table_name: default.page_views                                                           │
│   WHERE                                                                                                   │
│     FUNCTION id: 10, function_name: equals, function_type: ordinary, result_type: UInt8                   │
│       ARGUMENTS                                                                                           │
│         LIST id: 11, nodes: 2                                                                             │
│           COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3                         │
│           CONSTANT id: 12, constant_value: 'landing', constant_value_type: String                         │
│   GROUP BY                                                                                                │
│     LIST id: 13, nodes: 2                                                                                 │
│       COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3                             │
│       FUNCTION id: 4, function_name: toDate, function_type: ordinary, result_type: Date                   │
│         ARGUMENTS                                                                                         │
│           LIST id: 5, nodes: 1                                                                            │
│             FUNCTION id: 6, function_name: toStartOfMonth, function_type: ordinary, result_type: Date     │
│               ARGUMENTS                                                                                   │
│                 LIST id: 7, nodes: 1                                                                      │
│                   COLUMN id: 8, column_name: time, result_type: DateTime, source_id: 3                    │
│   ORDER BY                                                                                                │
│     LIST id: 14, nodes: 2                                                                                 │
│       SORT id: 15, sort_direction: ASCENDING, with_fill: 0                                                │
│         EXPRESSION                                                                                        │
│           COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3                         │
│       SORT id: 16, sort_direction: ASCENDING, with_fill: 0                                                │
│         EXPRESSION                                                                                        │
│           FUNCTION id: 4, function_name: toDate, function_type: ordinary, result_type: Date               │
│             ARGUMENTS                                                                                     │
│               LIST id: 5, nodes: 1                                                                        │
│                 FUNCTION id: 6, function_name: toStartOfMonth, function_type: ordinary, result_type: Date │
│                   ARGUMENTS                                                                               │
│                     LIST id: 7, nodes: 1                                                                  │
│                       COLUMN id: 8, column_name: time, result_type: DateTime, source_id: 3                │
│   LIMIT                                                                                                   │
│     CONSTANT id: 17, constant_value: UInt64_3, constant_value_type: UInt64                                │
└───────────────────────────────────────────────────────────────────────────────────────────────────────────┘
```

РЕКОМЕНДАЦИЯ

Чтобы ускорить выполнение запросов, в которых вычисляются агрегатные значения (как, например, в запросе выше вычисляется количество просмотров страниц по месяцам), можно использовать материализованные представления, автоматически вычисляющие промежуточные агрегатные состояния данных — см. пример в статье Материализованные представления.

Параметры

Вывод результата запроса `EXPLAIN QUERY TREE` можно настроить с помощью следующих параметров (если значение параметра `1` — соответствующая настройка включена, если `0` — выключена):

- `run_passes` — указывает, выполнять ли все проходы дерева запроса перед созданием дампа дерева запроса (значение по умолчанию — `1`);
- `passes` — определяет, сколько проходов дерева запроса выполнять (значение по умолчанию — `-1`, выполняются все проходы);
- `dump_passes` — указывает, выводить ли информацию о выполненных проходах перед созданием дампа дерева запроса (значение по умолчанию — `0`).

Пример

Следующий анализ выполняет три прохода дерева запроса и выводит информацию о них перед дампом дерева запроса:

```
EXPLAIN QUERY TREE passes=3, dump_passes=1
SELECT page_name, toDate(toStartOfMonth(time)) AS month, count() AS page_view_count
FROM page_views
WHERE page_name = 'landing'
GROUP BY page_name, month
ORDER BY page_name, month
LIMIT 3
FORMAT TSVRaw;
```

```
Pass 1 QueryAnalysis - Resolve type for each query expression. Replace identifiers, matchers with query expressions. Perform constant folding. Evaluate scalar subqueries.
Pass 2 FunctionToSubcolumns - Rewrite function to subcolumns, for example tupleElement(column, subcolumn) into column.subcolumn
Pass 3 CountDistinct - Optimize single countDistinct into count over subquery
QUERY id: 0
  PROJECTION COLUMNS
    page_name String
    month Date
    page_view_count UInt64
  PROJECTION
    LIST id: 1, nodes: 3
      COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3
      FUNCTION id: 4, function_name: toDate, function_type: ordinary, result_type: Date
        ARGUMENTS
          LIST id: 5, nodes: 1
            FUNCTION id: 6, function_name: toStartOfMonth, function_type: ordinary, result_type: Date
              ARGUMENTS
                LIST id: 7, nodes: 1
                  COLUMN id: 8, column_name: time, result_type: DateTime, source_id: 3
      FUNCTION id: 9, function_name: count, function_type: aggregate, result_type: UInt64
  JOIN TREE
    TABLE id: 3, table_name: default.page_views
  WHERE
    FUNCTION id: 10, function_name: equals, function_type: ordinary, result_type: UInt8
      ARGUMENTS
        LIST id: 11, nodes: 2
          COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3
          CONSTANT id: 12, constant_value: 'landing', constant_value_type: String
  GROUP BY
    LIST id: 13, nodes: 2
      COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3
      FUNCTION id: 4, function_name: toDate, function_type: ordinary, result_type: Date
        ARGUMENTS
          LIST id: 5, nodes: 1
            FUNCTION id: 6, function_name: toStartOfMonth, function_type: ordinary, result_type: Date
              ARGUMENTS
                LIST id: 7, nodes: 1
                  COLUMN id: 8, column_name: time, result_type: DateTime, source_id: 3
  ORDER BY
    LIST id: 14, nodes: 2
      SORT id: 15, sort_direction: ASCENDING, with_fill: 0
        EXPRESSION
          COLUMN id: 2, column_name: page_name, result_type: String, source_id: 3
      SORT id: 16, sort_direction: ASCENDING, with_fill: 0
        EXPRESSION
          FUNCTION id: 4, function_name: toDate, function_type: ordinary, result_type: Date
            ARGUMENTS
              LIST id: 5, nodes: 1
                FUNCTION id: 6, function_name: toStartOfMonth, function_type: ordinary, result_type: Date
                  ARGUMENTS
                    LIST id: 7, nodes: 1
                      COLUMN id: 8, column_name: time, result_type: DateTime, source_id: 3
  LIMIT
    CONSTANT id: 17, constant_value: UInt64_3, constant_value_type: UInt64
```

EXPLAIN PLAN

Запрос `EXPLAIN PLAN` выводит план выполнения анализируемого запроса в виде дерева, которое показывает в каком порядке будут выполняться выражения запроса. Этапы плана выполнения запроса читаются снизу вверх. Этот тип анализатора используется по умолчанию, если не указан другой тип `EXPLAIN`.

Пример

Следующий анализ показывает, что для выполнения запроса `SELECT` ADQM сначала считывает данные из таблицы (`ReadFromStorage`), затем фильтрует их (`Filter`) и после этого агрегирует (`Aggregating`).

```
EXPLAIN SELECT sum(number) AS test_sums FROM numbers(10000) WHERE number > 100 GROUP BY number % 2;
```

```
┌─explain─────────────────────────────────────┐
│ Expression ((Projection + Before ORDER BY)) │
│   Aggregating                               │
│     Expression (Before GROUP BY)            │
│       Filter (WHERE)                        │
│         ReadFromStorage (SystemNumbers)     │
└─────────────────────────────────────────────┘
```

Параметры

В запросах `EXPLAIN PLAN` можно использовать следующие параметры, чтобы указать какая информация должна включаться в результат выполняемого анализа (если значение параметра `1` — соответствующая настройка включена, если `0` — выключена):

- `header` — в план выполнения запроса добавляется ключ `Header` в виде массива столбцов (выводится название и тип столбца), значения которых вычисляются на каждом шаге.

  Пример

  EXPLAIN header = 0 (по умолчанию)

  ```
  EXPLAIN
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain─────────────────────────────────────┐
  │ Expression ((Projection + Before ORDER BY)) │
  │   Aggregating                               │
  │     Expression (Before GROUP BY)            │
  │       ReadFromStorage (SystemNumbers)       │
  └─────────────────────────────────────────────┘
  ```

  EXPLAIN header = 1

  ```
  EXPLAIN header = 1
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain─────────────────────────────────────┐
  │ Expression ((Projection + Before ORDER BY)) │
  │ Header: test_sums UInt64                    │
  │   Aggregating                               │
  │   Header: modulo(number, 2) UInt8           │
  │           sum(number) UInt64                │
  │     Expression (Before GROUP BY)            │
  │     Header: number UInt64                   │
  │             modulo(number, 2) UInt8         │
  │       ReadFromStorage (SystemNumbers)       │
  │       Header: number UInt64                 │
  └─────────────────────────────────────────────┘
  ```

- `description` — план выполнения запроса включает описание каждого шага.

  Пример

  EXPLAIN description = 0

  ```
  EXPLAIN description = 0
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain───────────────┐
  │ Expression            │
  │   Aggregating         │
  │     Expression        │
  │       ReadFromStorage │
  └───────────────────────┘
  ```

  EXPLAIN description = 1 (по умолчанию)

  ```
  EXPLAIN
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain─────────────────────────────────────┐
  │ Expression ((Projection + Before ORDER BY)) │
  │   Aggregating                               │
  │     Expression (Before GROUP BY)            │
  │       ReadFromStorage (SystemNumbers)       │
  └─────────────────────────────────────────────┘
  ```

- `indexes` — в план выполнения запроса добавляется ключ `Indexes`, который содержит массив используемых индексов. Для каждого индекса выводится ключ `Type` (возможные значения: `MinMax`, `Partition`, `PrimaryKey`, `Skip`) и дополнительные ключи:

  - `Name` — имя индекса (выводится только для индекса `Skip`);
  - `Keys` — массив столбцов, используемых индексом;
  - `Condition` — используемое условие;
  - `Description` — описание индекса (выводится только для индекса `Skip`);
  - `Parts` — количество кусков данных, отфильтрованных применяемым индексом;
  - `Granules` — количество гранул, отфильтрованных применяемым индексом.

  Этот параметр поддерживается только для анализа запросов, обращающихся к таблицам семейства MergeTree.

  Пример

  EXPLAIN actions = 0 (default)

  ```
  EXPLAIN indexes = 0
  SELECT sum(value)
  FROM test_table
  WHERE (value % 10) = 1;
  ```

  ```
  ┌─explain──────────────────────────────────────────
  ┐
  │ Expression ((Projection + Before ORDER BY))
  │
  │   Aggregating
  │
  │     Expression (Before GROUP BY)
  │
  │       Filter (WHERE)
  │
  │         ReadFromMergeTree (default.test_table)
  │
  └──────────────────────────────────────────────────
  ┘
  ```

  EXPLAIN actions = 1

  ```
  EXPLAIN indexes = 1
  SELECT sum(value)
  FROM test_table
  WHERE (value % 10) = 1;
  ```

  ```
  ┌─explain──────────────────────────────────────────
  ──────────┐
  │ Expression ((Projection + Before ORDER BY))
  │
  │   Aggregating
  │
  │     Expression (Before GROUP BY)
  │
  │       Filter (WHERE)
  │
  │         ReadFromMergeTree (default.test_table)
  │
  │         Indexes:
  │
  │           MinMax
  │
  │             Condition: true
  │
  │             Parts: 10/10
  │
  │             Granules: 120/120
  │
  │           Partition
  │
  │             Keys:
  │
  │               moduloLegacy(value, 10)
  │
  │             Condition: (moduloLegacy(value, 10)
  in [1, 1]) │
  │             Parts: 1/10
  │
  │             Granules: 12/120
  │
  │           PrimaryKey
  │
  │             Condition: true
  │
  │             Parts: 1/1
  │
  │             Granules: 12/12
  │
  └──────────────────────────────────────────────────
  ──────────┘
  ```

- `actions` — в план выполнения запроса добавляется подробная информация о действиях, выполняемых на каждом шаге (ключ `Actions`).

  Пример

  EXPLAIN actions = 0 (по умолчанию)

  ```
  EXPLAIN
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain─────────────────────────────────────┐
  │ Expression ((Projection + Before ORDER BY)) │
  │   Aggregating                               │
  │     Expression (Before GROUP BY)            │
  │       ReadFromStorage (SystemNumbers)       │
  └─────────────────────────────────────────────┘
  ```

  EXPLAIN actions = 1

  ```
  EXPLAIN actions = 1
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain──────────────────────────────────────────
  ──────────────────────────────┐
  │ Expression ((Projection + Before ORDER BY))
  │
  │ Actions: INPUT : 0 -> sum(number) UInt64 : 0
  │
  │          ALIAS sum(number) :: 0 -> test_sums
  UInt64 : 1                              │
  │ Positions: 1
  │
  │   Aggregating
  │
  │   Keys: modulo(number, 2)
  │
  │   Aggregates:
  │
  │       sum(number)
  │
  │         Function: sum(UInt64) → UInt64
  │
  │         Arguments: number
  │
  │   Skip merging: 0
  │
  │     Expression (Before GROUP BY)
  │
  │     Actions: INPUT : 0 -> number UInt64 : 0
  │
  │              COLUMN Const(UInt8) -> 2 UInt8 : 1
  │
  │              FUNCTION modulo(number : 0, 2 ::
  1) -> modulo(number, 2) UInt8 : 2 │
  │     Positions: 0 2
  │
  │       ReadFromStorage (SystemNumbers)
  │
  └──────────────────────────────────────────────────
  ──────────────────────────────┘
  ```

- `json` — план выполнения запроса выводится в формате JSON. Каждый узел — это словарь, в котором всегда есть ключи `Node Type` (название шага) и `Plans` (массив с описаниями дочерних шагов). Другие дополнительные ключи могут быть добавлены в зависимости от типа узла и настроек.
  Чтобы избежать ненужного экранирования, рекомендуется использовать формат TSVRaw для вывода результата анализа.

  Пример

  EXPLAIN json = 0 (по умолчанию)

  ```
  EXPLAIN
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain─────────────────────────────────────┐
  │ Expression ((Projection + Before ORDER BY)) │
  │   Aggregating                               │
  │     Expression (Before GROUP BY)            │
  │       ReadFromStorage (SystemNumbers)       │
  └─────────────────────────────────────────────┘
  ```

  EXPLAIN json = 1

  ```
  EXPLAIN json = 1
  SELECT sum(number) AS test_sums
  FROM numbers(5) GROUP BY number % 2 FORMAT TSVRaw;
  ```

  ```
  [
    {
      "Plan": {
        "Node Type": "Expression",
        "Description": "(Projection + Before ORDER BY)",
        "Plans": [
          {
            "Node Type": "Aggregating",
            "Plans": [
              {
                "Node Type": "Expression",
                "Description": "Before GROUP BY",
                "Plans": [
                  {
                    "Node Type": "ReadFromStorage",
                    "Description": "SystemNumbers"
                  }
                ]
              }
            ]
          }
        ]
      }
    }
  ]
  ```

EXPLAIN PIPELINE

Запрос `EXPLAIN PIPELINE` выводит конвейер выполнения запроса, который показывает сколько параллельных потоков используется для выполнения каждого шага плана запроса.

Пример

```
EXPLAIN PIPELINE SELECT sum(number) AS test_sums FROM numbers(5) GROUP BY number % 2;
```

```
┌─explain───────────────────┐
│ (Expression)              │
│ ExpressionTransform × 8   │
│   (Aggregating)           │
│   Resize 1 → 8            │
│     AggregatingTransform  │
│       (Expression)        │
│       ExpressionTransform │
│         (ReadFromStorage) │
│         Limit             │
│           Numbers 0 → 1   │
└───────────────────────────┘
```

Параметры

Вывод результата выполнения запроса `EXPLAIN PIPELINE` можно настраивать с помощью следующих параметров (если значение параметра `1` — настройка включена, если `0` — выключена):

- `header` — выводится заголовок для каждого выходного порта (значение по умолчанию — `0`).

  Пример

  ```
  EXPLAIN PIPELINE header = 1
  SELECT sum(number) AS test_sums FROM numbers(5) GROUP BY number % 2;
  ```

  ```
  ┌─explain─────────────────────────────────────────────────────────────────────────┐
  │ (Expression)                                                                    │
  │ ExpressionTransform × 8                                                         │
  │ Header: test_sums UInt64: test_sums UInt64 UInt64(size = 0)                     │
  │   (Aggregating)                                                                 │
  │   Resize 1 → 8                                                                  │
  │   Header × 8 : modulo(number, 2) UInt8: modulo(number, 2) UInt8 UInt8(size = 0) │
  │                sum(number) UInt64: sum(number) UInt64 UInt64(size = 0)          │
  │     AggregatingTransform                                                        │
  │     Header: modulo(number, 2) UInt8: modulo(number, 2) UInt8 UInt8(size = 0)    │
  │             sum(number) UInt64: sum(number) UInt64 UInt64(size = 0)             │
  │       (Expression)                                                              │
  │       ExpressionTransform                                                       │
  │       Header: number UInt64: number UInt64 UInt64(size = 0)                     │
  │               modulo(number, 2) UInt8: modulo(number, 2) UInt8 UInt8(size = 0)  │
  │         (ReadFromStorage)                                                       │
  │         Limit                                                                   │
  │         Header: number UInt64: number UInt64 UInt64(size = 0)                   │
  │           Numbers 0 → 1                                                         │
  │           Header: number UInt64: number UInt64 UInt64(size = 0)                 │
  └─────────────────────────────────────────────────────────────────────────────────┘
  ```

- `graph` — конвейер выполнения запроса выводится в виде описанного на языке DOT графа (значение по умолчанию — `0`). Полученный синтаксис можно загрузить в приложение для визуализации графов (например, Graphviz).

  Пример

  ```
  EXPLAIN PIPELINE graph = 1
  SELECT sum(number) AS test_sums FROM numbers(5) GROUP BY number % 2 FORMAT TSV;
  ```

  ```
  digraph
  {
    rankdir="LR";
    { node [shape = rect]
      subgraph cluster_0 {
        label ="Expression";
        style=filled;
        color=lightgrey;
        node [style=filled,color=white];
        { rank = same;
          n6 [label="ExpressionTransform × 8"];
        }
      }
      subgraph cluster_1 {
        label ="ReadFromStorage";
        style=filled;
        color=lightgrey;
        node [style=filled,color=white];
        { rank = same;
          n2 [label="Limit"];
          n1 [label="Numbers"];
        }
      }
      subgraph cluster_2 {
        label ="Expression";
        style=filled;
        color=lightgrey;
        node [style=filled,color=white];
        { rank = same;
          n3 [label="ExpressionTransform"];
        }
      }
      subgraph cluster_3 {
        label ="Aggregating";
        style=filled;
        color=lightgrey;
        node [style=filled,color=white];
        { rank = same;
          n4 [label="AggregatingTransform"];
          n5 [label="Resize"];
        }
      }
    }
    n2 -> n3 [label=""];
    n1 -> n2 [label=""];
    n3 -> n4 [label=""];
    n4 -> n5 [label=""];
    n5 -> n6 [label="× 8"];
  }
  ```

- `compact` — если включена настройка `graph`, граф выводится в компактном режиме (значение по умолчанию — `1`).

  Пример

  ```
  EXPLAIN PIPELINE graph = 1, compact = 0
  SELECT sum(number) AS test_sums FROM numbers(5) GROUP BY number % 2 FORMAT TSV;
  ```

  ```
  digraph
  {
    rankdir="LR";
    { node [shape = rect]
      n140206024061464[label="Numbers"];
      n140205465170584[label="Limit"];
      n140205620507928[label="ExpressionTransform"];
      n140205801688600[label="AggregatingTransform"];
      n140205649781464[label="Resize"];
      n140205630520728[label="ExpressionTransform"];
      n140205788048536[label="ExpressionTransform"];
      n140205620512152[label="ExpressionTransform"];
      n140206023882392[label="ExpressionTransform"];
      n140206024060696[label="ExpressionTransform"];
      n140205620509080[label="ExpressionTransform"];
      n140205717371544[label="ExpressionTransform"];
      n140206023882008[label="ExpressionTransform"];
    }
    n140206024061464 -> n140205465170584;
    n140205465170584 -> n140205620507928;
    n140205620507928 -> n140205801688600;
    n140205801688600 -> n140205649781464;
    n140205649781464 -> n140205630520728;
    n140205649781464 -> n140205788048536;
    n140205649781464 -> n140205620512152;
    n140205649781464 -> n140206023882392;
    n140205649781464 -> n140206024060696;
    n140205649781464 -> n140205620509080;
    n140205649781464 -> n140205717371544;
    n140205649781464 -> n140206023882008;
  }
  ```

EXPLAIN ESTIMATE

Анализ с помощью запроса `EXPLAIN ESTIMATE` показывает предполагаемое количество строк, засечек и кусков данных, которые необходимо считать из таблицы при обработке запроса. Этот тип анализа можно применять только к запросам, обращающимся к таблицам семейства MergeTree.

Пример

1. Создайте таблицу MergeTree, используя специальные опции, чтобы настроить деление кусков данных на гранулы:

   - `index_granularity` — задает максимальное количество строк данных между засечками индекса;
   - `write_final_mark` — указывает, записывать ли последнюю засечку индекса в конце куска данных, указывающую на последний байт.

   ```
   CREATE TABLE test_table_estimate (i Int64) ENGINE = MergeTree ORDER BY i SETTINGS index_granularity = 16, write_final_mark = 0;
   ```

2. Вставьте данные в таблицу, повторив следующую команду три раза:

   ```
   INSERT INTO test_table_estimate SELECT number FROM numbers(128);
   ```

3. Выполните анализ запроса для чтения данных из таблицы:

   ```
   EXPLAIN ESTIMATE SELECT * FROM test_table_estimate;
   ```

   Результат анализа:

   ```
   ┌─database─┬─table───────────────┬─parts─┬─rows─┬─marks─┐
   │ default  │ test_table_estimate │     3 │  384 │    24 │
   └──────────┴─────────────────────┴───────┴──────┴───────┘
   ```

EXPLAIN TABLE OVERRIDE

Запрос `EXPLAIN TABLE OVERRIDE` позволяет предварительно проверить корректность конфигураций для переопределения схемы таблицы, которые можно указать при создании базы данных типа MaterializedMySQL (см. Table Overrides). Если переопределение схемы таблицы может привести к какой-либо ошибке, будет выведено соответствующее сообщение (например, при использовании столбцов типа Nullable для выражений `PARTITION BY` и `ORDER BY`, для которых это не рекомендуется).

Для выполнения запроса `EXPLAIN TABLE OVERRIDE` подключение к таблице на удаленном сервере MySQL осуществляется через табличную функцию mysql.

Пример

1. На удаленном сервере MySQL создайте таблицу в базе данных `mysql_db`:

   ```
   CREATE TABLE mysql_table (id INT PRIMARY KEY, date DATETIME DEFAULT now());
   ```

2. В ADQM выполните следующий запрос, чтобы проверить корректность партиционирование таблицы по столбцу `date` (замените `<host>:<port>` на адрес сервера MySQL, а `<user_name>` и `<user_password>` на имя пользователя и пароль для доступа к базе данных MySQL):

   ```
   EXPLAIN TABLE OVERRIDE mysql('<host>:<port>', 'mysql_db', 'mysql_table', '<user_name>', '<user_password>')
   PARTITION BY toYYYYMM(date);
   ```

   В результате выполнения запроса будет выведено исключение с рекомендацией, как можно исправить переопределение ключа партиционирования:

   ```
   Exception: PARTITION BY override refers to nullable column `date` (use assumeNotNull() if the column does not in fact contain NULL values). (INVALID_TABLE_OVERRIDE)
   ....`
   ```

3. Измените переопределение `PARTITION BY` в соответствии с указанной рекомендацией:

   ```
   EXPLAIN TABLE OVERRIDE mysql('<host>:<port>', 'mysql_db', 'mysql_table', 'root', 'my-secret-pw')
   PARTITION BY toYYYYMM(assumeNotNull(date));
   ```

   Вывод:

   ```
   ┌─explain──────────────────────────────────────────────┐
   │ PARTITION BY uses columns: `date` Nullable(DateTime) │
   └──────────────────────────────────────────────────────┘
   ```

Семплирующий профилировщик запросов

Елена Дворядкина

Содержание

Назад к содержанию

Включение профилировщика запросов

Анализ отчетов профилировщика

Функции интроспекции

- addressToLine
- addressToLineWithInlines
- addressToSymbol
- demangle
- tid
- logTrace

Семплирующий профилировщик запросов (sampling query profiler) предоставляет возможность интроспекции производительности запросов в ADQM. Например, с его помощью в исходном коде можно найти функции, которые наиболее часто вызываются во время выполнения запроса, или отследить затрачиваемое процессорное и реальное время, включая время простоя.

Включение профилировщика запросов

1. В интерфейсе ADCM на странице конфигурации сервиса ADQMDB активируйте опцию ***Show advanced*** и в секции ***Log settings*** включите настройку ***trace_log***, чтобы профилировщик запросов начал собирать трассировки стека и сохранять их в таблицу `trace_log`. В этой же секции настройте параметры таблицы `trace_log`:

   - ***Database*** — имя базы данных, в которой будет храниться таблица;
   - ***Flush interval, milliseconds*** — интервал сброса данных из буфера памяти в таблицу (в миллисекундах);
   - ***TTL, days*** — время хранения строк в таблице (в днях).

   Нажмите ***Save*** и выполните действие ***Reconfig*** с включенной опцией ***Restart*** для сервиса ADQMDB, чтобы сохранить информацию об указанных параметрах конфигурации и перезапустить сервис.

   > ПРИМЕЧАНИЕ
   >
   > - Ключом партиционирования таблицы `trace_log` автоматически устанавливается столбец `event_date` — дата в момент получения экземпляра трассировки стека. Подробнее столбцы таблицы описаны в статье trace_log.
   > - Данные в таблице `trace_log` актуальны только для работающего сервера. После перезапуска сервера ADQM таблица не очищается и все сохраненные адреса виртуальной памяти могут стать недействительными.

2. С помощью параметров `query_profiler_real_time_period_ns` и `query_profiler_cpu_time_period_ns` настройте таймер реального времени и таймер CPU, чтобы определить периодичность, с которой профилировщик запросов будет собирать трассировки стека и записывать их в таблицу `trace_log`.

   По умолчанию оба таймера настроены на выполнение одной выборки в секунду. Такая частота выборки позволяет собрать достаточно информации о кластере ADQM, при этом профилирование не влияет на производительность серверов ADQM. Если необходимо профилировать каждый запрос отдельно, можно попробовать использовать более высокую частоту выборки, например, 100 раз в секунду:

   ```
   SET query_profiler_real_time_period_ns=10000000;
   ```

   ```
   SET query_profiler_cpu_time_period_ns=10000000;
   ```

   Собирать трассировки стека с частотой выше нескольких тысяч выборок в секунду обычно невозможно.

Анализ отчетов профилировщика

1. Установите пакет adqm-clickhouse-debuginfo.
2. Включите функции интроспекции для профилирования запросов (в целях безопасности эти функции отключены по умолчанию):

   ```
   SET allow_introspection_functions = 1;
   ```

3. Используйте функции интроспекции для анализа данных таблицы `trace_log` (см. следующий раздел).

Функции интроспекции

В этом разделе описаны функции, которые можно использовать для анализа форматов ELF и DWARF при профилировании запросов.

Предварительные настройки для выполнения примеров

1. Перед выполнением примеров использования функций интроспекции, убедитесь, что пакет adqm-clickhouse-debuginfo установлен и активирована настройка `allow_introspection_functions = 1`.
2. В приведенных примерах функции интроспекции применяются для анализа первого экземпляра трассировки стека из таблицы `trace_log`. Используйте следующий запрос, чтобы получить первую строку этой таблицы:

   ```
   SELECT * FROM system.trace_log LIMIT 1 FORMAT Vertical;
   ```

   ```
   Row 1:
   ──────
   event_date:              2023-09-04
   event_time:              2023-09-04 10:18:01
   event_time_microseconds: 2023-09-04 10:18:01.858935
   timestamp_ns:            1693822681858935963
   revision:                54472
   trace_type:              Memory
   thread_id:               11216
   query_id:                7b53b488-6e38-495a-ab63-e0ed985bf2de
   trace:
   [282096402,282019952,282019801,282176985,373506363,387689543,282729773,282744526,140712295374501,140712292395789]
   size:                    8760
   event:
   increment:               0
   ```

   Поле `trace` содержит трассировку стека в момент выборки, где каждый элемент массива — это адрес виртуальной памяти внутри процесса сервера ADQM. Функции интроспекции можно применять к одному адресу или ко всей трассировке стека.

addressToLine

Функция `addressToLine` преобразует адрес виртуальной памяти внутри процесса сервера ADQM в имя файла и номер строки в исходном коде.

Синтаксис вызова функции:

```
addressToLine(<address_of_binary_instruction>)
```

где `<address_of_binary_instruction>` — адрес инструкции в запущенном процессе (значение типа UInt64).

Функция возвращает значение типа String:

- Имя файла исходного кода и номер строки в этом файле, разделяемые двоеточием.
- Имя бинарного файла, если функция не может найти отладочную информацию.
- Пустую строку, если передаваемый в качестве аргумента адрес не является допустимым.

Пример

1. Получите имя файла исходного кода и номер строки для одного адреса:

   ```
   SELECT addressToLine(282096402::UInt64);
   ```

   В выводе результата выполнения функции `./build/src/Common/StackTrace.cpp` — имя файла, а `287` — номер строки:

   ```
   ┌─addressToLine(CAST('282096402', 'UInt64'))─┐
   │ ./build/src/Common/StackTrace.cpp:287      │
   └────────────────────────────────────────────┘
   ```

2. Примените функцию ко всей трассировке стека. Используйте функцию arrayMap, чтобы обработать каждый отдельный элемент массива `trace` с помощью функции `addressToLine`:

   ```
   SELECT arrayStringConcat(arrayMap(x -> addressToLine(x), trace), '\n') AS trace_source_code_lines
   FROM system.trace_log
   LIMIT 1
   FORMAT Vertical;
   ```

   ```
   Row 1:
   ──────
   trace_source_code_lines: ./build/src/Common/StackTrace.cpp:287
   ./build/src/Common/MemoryTracker.cpp:202
   ./build/src/Common/MemoryTracker.cpp:338
   ./build/src/Common/ThreadStatus.cpp:185
   ./build/contrib/llvm-project/libcxx/include/__memory/unique_ptr.h:302
   ./build/contrib/llvm-project/libcxx/include/__memory/shared_ptr.h:701
   ./build/base/base/wide_integer_impl.h:789
   ./build/contrib/llvm-project/libcxx/include/__memory/unique_ptr.h:302
   /usr/lib64/libpthread-2.17.so
   /usr/lib64/libc-2.17.so
   ```

addressToLineWithInlines

Функция `addressToLineWithInlines` похожа на `addressToLine`, но возвращает массив со всеми встроенными функциями (поэтому она вычисляется медленнее).

Синтаксис вызова функции:

```
addressToLineWithInlines(<address_of_binary_instruction>)
```

где `<address_of_binary_instruction>` — адрес инструкции в запущенном процессе (значение типа UInt64).

Функция возвращает массив строк:

- Массив, в котором первый элемент содержит имя файла исходного кода и номер строки в этом файле (разделенные двоеточием), остальные элементы включают информацию о встроенных функциях — имя файла исходного кода с встроенной функцией, номер строки и имя функции.
- Массив с одним элементом — именем бинарного файла, если функция не смогла найти отладочную информацию.
- Пустой массив, если передаваемый в качестве аргумента адрес не является допустимым.

Пример

1. Примените функцию к одному адресу:

   ```
   SELECT addressToLineWithInlines(373506363::UInt64) FORMAT Vertical;
   ```

   ```
   Row 1:
   ──────
   addressToLineWithInlines(CAST('373506363', 'UInt64')): ['./build/src/Interpreters/ThreadStatusExt.
   cpp:185','./build/contrib/llvm-project/libcxx/include/__memory/unique_ptr.h:302:std::__1::unique_ptr<DB::QueryProfilerReal,
   std::__1::default_delete<DB::QueryProfilerReal>>::reset[abi:v15000]
   (DB::QueryProfilerReal*)','./build/src/Interpreters/ThreadStatusExt.cpp:415:DB::ThreadStatus::finalizeQueryProfiler()']
   ```

2. Примените функцию ко всей трассировке стека. Используйте функцию arrayJoin, чтобы разделить массив на строки:

   ```
   SELECT address, addressToLineWithInlines(arrayJoin(trace) AS address)
   FROM system.trace_log
   WHERE query_id = '7b53b488-6e38-495a-ab63-e0ed985bf2de'
   FORMAT Vertical;
   ```

   ```
   Row 1:
   ──────
   address:                                    282096402
   addressToLineWithInlines(arrayJoin(trace)): ['./build/src/Common/StackTrace.cpp:287']

   Row 2:
   ──────
   address:                                    282019952
   addressToLineWithInlines(arrayJoin(trace)): ['./build/src/Common/MemoryTracker.cpp:202']

   Row 3:
   ──────
   address:                                    282019801
   addressToLineWithInlines(arrayJoin(trace)): ['./build/src/Common/MemoryTracker.cpp:338']

   Row 4:
   ──────
   address:                                    282176985
   addressToLineWithInlines(arrayJoin(trace)): ['./build/src/Common/ThreadStatus.cpp:185']

   Row 5:
   ──────
   address:                                    373506363
   addressToLineWithInlines(arrayJoin(trace)): ['./build/src/Interpreters/ThreadStatusExt.cpp:185','./build/contrib/llvm-
   project/libcxx/include/__memory/unique_ptr.h:302:std::__1::unique_ptr<DB::QueryProfilerReal,
   std::__1::default_delete<DB::QueryProfilerReal>>::reset[abi:v15000]
   (DB::QueryProfilerReal*)','./build/src/Interpreters/ThreadStatusExt.cpp:415:DB::ThreadStatus::finalizeQueryProfiler()']

   ...
   ```

addressToSymbol

Функция `addressToSymbol` преобразует адрес виртуальной памяти внутри процесса сервера ADQM в символ из объектного файла.

Синтаксис вызова функции:

```
addressToSymbol(<address_of_binary_instruction>)
```

где `<address_of_binary_instruction>` — адрес инструкции в запущенном процессе (значение типа UInt64).

Функция возвращает значение типа String:

- Символ из объектного файла.
- Пустую строку, если передаваемый в качестве аргумента адрес не является допустимым.

Пример

1. Получите символ для одного адреса:

   ```
   SELECT addressToSymbol(282096402::UInt64);
   ```

   ```
   ┌─addressToSymbol(CAST('282096402', 'UInt64'))─┐
   │ _ZN10StackTrace10tryCaptureEv                │
   └──────────────────────────────────────────────┘
   ```

2. Примените функцию ко всей трассировке стека. Используйте функцию arrayMap, чтобы обработать каждый отдельный элемент массива `trace` с помощью функции `addressToSymbol`:

   ```
   SELECT arrayStringConcat(arrayMap(x -> addressToSymbol(x), trace), '\n') AS trace_symbols
   FROM system.trace_log
   LIMIT 1
   FORMAT Vertical;
   ```

   ```
   Row 1:
   ──────
   trace_symbols: _ZN10StackTrace10tryCaptureEv
   _ZN13MemoryTracker9allocImplElbPS_
   _ZN13MemoryTracker9allocImplElbPS_
   _ZN2DB12ThreadStatus20flushUntrackedMemoryEv
   _ZN2DB12ThreadStatus15detachFromGroupEv
   _ZNSt3__110__function16__policy_invokerIFvvEE11__call_implINS0_20__default_alloc_funcIZN24ThreadFromGlobalPoolImplILb1EEC1I
   ZN2DB28PullingAsyncPipelineExecutor4pullERNS9_5ChunkEmE3$_0JEEEOT_DpOT0_EUlvE_S2_EEEEvPKNS0_16__policy_storageE
   _ZN14ThreadPoolImplINSt3__16threadEE6workerENS0_15__list_iteratorIS1_PvEE
   _ZNSt3__114__thread_proxyB6v15000INS_5tupleIJNS_10unique_ptrINS_15__thread_structENS_14default_deleteIS3_EEEEZN14ThreadPool
   ImplINS_6threadEE12scheduleImplIvEET_NS_8functionIFvvEEElNS_8optionalImEEbEUlvE0_EEEEEPvSJ_
   start_thread
   __clone
   ```

demangle

Функция `demangle` преобразует символ, который можно получить с помощью функции `addressToSymbol`, в имя функции C++.

Синтаксис вызова функции:

```
demangle(<symbol>)
```

где `<symbol>` — символ из объектного файла (значение типа String).

Функция возвращает значение типа String:

- Имя функции C++.
- Пустую строку, если передаваемый в качестве аргумента символ не является допустимым.

Пример

1. Получите имя функции для одного адреса:

   ```
   SELECT demangle(addressToSymbol(282096402::UInt64));
   ```

   ```
   ┌─demangle(addressToSymbol(CAST('282096402', 'UInt64')))─┐
   │ StackTrace::tryCapture()                               │
   └────────────────────────────────────────────────────────┘
   ```

2. Примените функцию ко всей трассировке стека. Используйте функцию arrayMap, чтобы обработать каждый отдельный элемент массива `trace` с помощью функции `demangle`:

   ```
   SELECT arrayStringConcat(arrayMap(x -> demangle(addressToSymbol(x)), trace), '\n') AS trace_functions
   FROM system.trace_log
   LIMIT 1
   FORMAT Vertical;
   ```

   ```
   Row 1:
   ──────
   trace_functions: StackTrace::tryCapture()
   MemoryTracker::allocImpl(long, bool, MemoryTracker*)
   MemoryTracker::allocImpl(long, bool, MemoryTracker*)
   DB::ThreadStatus::flushUntrackedMemory()
   DB::ThreadStatus::detachFromGroup()
   void std::__1::__function::__policy_invoker<void
   ()>::__call_impl<std::__1::__function::__default_alloc_func<ThreadFromGlobalPoolImpl<true>::ThreadFromGlobalPoolImpl<DB::Pu
   llingAsyncPipelineExecutor::pull(DB::Chunk&, unsigned long)::$_0>(DB::PullingAsyncPipelineExecutor::pull(DB::Chunk&,
   unsigned long)::$_0&&)::'lambda'(), void ()>>(std::__1::__function::__policy_storage const*)
   ThreadPoolImpl<std::__1::thread>::worker(std::__1::__list_iterator<std::__1::thread, void*>)
   void* std::__1::__thread_proxy[abi:v15000]<std::__1::tuple<std::__1::unique_ptr<std::__1::__thread_struct,
   std::__1::default_delete<std::__1::__thread_struct>>, void ThreadPoolImpl<std::__1::thread>::scheduleImpl<void>
   (std::__1::function<void ()>, long, std::__1::optional<unsigned long>, bool)::'lambda0'()>>(void*)
   start_thread
   __clone
   ```

tid

Функция `tid` возвращает идентификатор потока, в котором обрабатывается текущий Block.

Синтаксис вызова функции:

```
tid()
```

Функция возвращает значение типа Uint64.

Пример

```
SELECT tid();
```

```
┌─tid()─┐
│ 10959 │
└───────┘
```

logTrace

Функция `logTrace` выводит сообщение в лог сервера для каждого Block.

Синтаксис вызова функции:

```
logTrace('<message>')
```

где `<message>` — сообщение, которое отправляется в серверный лог.

Функция всегда возвращает `0`.

Пример

```
SELECT logTrace('logTrace message');
```

```
┌─logTrace('logTrace message')─┐
│                            0 │
└──────────────────────────────┘
```

Принудительное завершение запроса

Елена Дворядкина

Содержание

Назад к содержанию

KILL QUERY
  └ Пример
KILL MUTATION
  └ Пример
Известная проблема

Для принудительной остановки выполнения запросов и прерывания мутаций ADQM/ClickHouse предоставляет команды KILL QUERY и KILL MUTATION.

KILL QUERY

Хотя ADQM/ClickHouse является высокопроизводительной СУБД, плохо оптимизированные запросы (например, полное сканирование больших наборов данных, применение неэффективных индексов или неоптимизированные операции JOIN) могут выполняться достаточно долго, потреблять много ресурсов и блокировать критически важные процессы. В таких случаях может возникнуть необходимость остановить выполнение запроса — используйте для этого команду `KILL QUERY`.

> **РЕКОМЕНДАЦИЯ**
>
> Возможные способы ускорения выполнения запросов можно посмотреть в статье Повышение производительности запросов.

Базовый синтаксис запроса `KILL QUERY`:

```
KILL QUERY [ON CLUSTER <cluster_name>] WHERE <where_expr> [ASYNC|SYNC|TEST] [FORMAT <output_format>];
```

где:

- `[ON CLUSTER <cluster_name>]` — необязательное выражение, которое используется, чтобы указать имя кластера, когда нужно выполнить прерывание запроса на всех узлах кластера.
- `WHERE <where_expr>` — обязательное выражение, определяющее критерии отбора запросов, которые необходимо завершить.
  Выявить выполняющиеся слишком долго и/или потребляющие много памяти запросы можно с помощью системной таблицы system.processes, которая содержит информацию о выполняющихся в настоящее время запросах и их состояниях. Например, вы можете использовать поля `query_id`, `user`, `elapsed`, `memory_usage` и другие релевантные поля этой таблицы для идентификации запроса, выполнение которого необходимо остановить:

  ```
  SELECT query_id, user, elapsed, memory_usage FROM system.processes ORDER BY elapsed DESC;
  ```

  В качестве альтернативы `SELECT * FROM system.processes` можно использовать команду:

  ```
  SHOW PROCESSLIST;
  ```

- `[ASYNC|SYNC|TEST]` — ключевое слово, указывающее должна ли команда `KILL QUERY` выполняться асинхронно, синхронно или в тестовом режиме:
  - `ASYNC` (по умолчанию) — команда не ждет подтверждения о том, что выполнение указанных запросов остановлено.
  - `SYNC` — команда ожидает завершения всех указанных запросов и отображает информацию о каждом процессе по мере его завершения.
  - `TEST` — тестовый запуск команды `KILL QUERY`, который позволяет проверить права пользователя и посмотреть, какие запросы будут остановлены командой, без их фактического завершения.
- `[FORMAT <output_format>]` — необязательное выражение, указывающее формат вывода результата операции завершения запроса.

Результат выполнения команды `KILL QUERY` содержит столбец `kill_status`, возможные значения которого включают:

- `finished` — выполнение запроса было успешно остановлено;
- `waiting` — ожидание завершения запроса после отправки ему сигнала на завершение;
- `cant_cancel` — сигнал на завершение не может быть отправлен;
- `pending` — запрос еще не инициализирован;
- `unknown_status` — статус операции прерывания запроса неизвестен.

Чтобы пользователь мог прерывать выполнение запросов других пользователей, у него должна быть привилегия `KILL QUERY`. Пользователи с правами только на чтение могут прерывать исключительно свои запросы.

Пример

> **ПРИМЕЧАНИЕ**
>
> В данном примере используются два пользовательских аккаунта ADQM:
>
> - `default` — пользователь по умолчанию, который имеет максимальные привилегии;
> - `adqm_user` — пользователь с правами на чтение данных (`SELECT`) и отмену запросов других пользователей (`KILL QUERY`).

1. В двух отдельных сессиях clickhouse-client от имени пользователя default запустите запрос для тестирования, который будет выполняться бесконечно, например:

   ```
   SELECT count() FROM system.numbers;
   ```

   Во время выполнения запроса в клиенте будет показан идентификатор запроса и прогресс выполнения.

   - Сессия 1:

     ```
     SELECT count()
     FROM system.numbers

     Query id: 6d522fb4-97db-45e4-9565-3158edb5a281

     ↖ Progress: 398.17 billion rows, 3.19 TB (3.40 billion rows/s., 27.22 GB/s.)   (1.0 CPU)
     Press the space key to toggle the display of the progress table.
     ```

   - Сессия 2:

     ```
     SELECT count()
     FROM system.numbers

     Query id: f38506fd-4923-4bd2-8668-e6affdaa61c4

     ↖ Progress: 403.64 billion rows, 3.23 TB (3.41 billion rows/s., 27.24 GB/s.)   (1.0 CPU)
     Press the space key to toggle the display of the progress table.
     ```

2. Откройте еще одну сессию клиента `clickhouse-client` от имени пользователя `adqm_user`. Посмотрите список незавершенных запросов, запущенных пользователем `default`, и длительность их выполнения на текущий момент:

   ```
   SELECT query_id, user, elapsed, query
   FROM system.processes
   WHERE user='default'
   ORDER BY elapsed DESC;
   ```

   ```
      ┌─query_id─────────────────────────────┬─user────┬────elapsed─┬─query───────────────────────────────┐
   1. │ 6d522fb4-97db-45e4-9565-3158edb5a281 │ default │ 133.361736 │ SELECT count() FROM system.numbers; │
   2. │ f38506fd-4923-4bd2-8668-e6affdaa61c4 │ default │ 125.132542 │ SELECT count() FROM system.numbers; │
      └──────────────────────────────────────┴─────────┴────────────┴─────────────────────────────────────┘
   ```

3. Также от имени пользователя `adqm_user` выполните принудительное завершение запросов одним из способов, перечисленных ниже.

   - Завершение запроса по его идентификатору:

     ```
     KILL QUERY WHERE query_id = '6d522fb4-97db-45e4-9565-3158edb5a281';
     ```

     Результат выполнения команды `KILL QUERY`:

     ```
        ┌─kill_status─┬─query_id─────────────────────────────┬─user────┬─query───────────────────────────────┐
     1. │ waiting     │ 6d522fb4-97db-45e4-9565-3158edb5a281│ default │ SELECT count() FROM system.numbers; │
        └─────────────┴──────────────────────────────────────┴─────────┴─────────────────────────────────────┘
     ```

     Столбец `kill_status` содержит значение `waiting`, так как запрос `KILL QUERY` по умолчанию выполняется в асинхронном режиме, то есть возвращает результат сразу, не дожидаясь фактического завершения указанного запроса.

   - Завершение всех запросов, запущенных пользователем `default`:

     ```
     KILL QUERY WHERE user = 'default';
     ```

     ```
        ┌─kill_status─┬─query_id─────────────────────────────┬─user────┬─query───────────────────────────────┐
     1. │ waiting     │ 6d522fb4-97db-45e4-9565-3158edb5a281 │ default │ SELECT count() FROM system.numbers; │
     2. │ waiting     │ f38506fd-4923-4bd2-8668-e6affdaa61c4 │ default │ SELECT count() FROM system.numbers; │
        └─────────────┴──────────────────────────────────────┴─────────┴─────────────────────────────────────┘
     ```

     Перед тем как вызывать команду для прерывания запросов, можно выполнить `KILL QUERY` с модификатором `TEST`, чтобы убедиться, что у пользователя `adqm_user` есть права на остановку запросов, запущенных другим пользователем, и посмотреть, какие именно запросы будет отменять эта команда:

     ```
     KILL QUERY WHERE user = 'default' TEST;
     ```

     ```
        ┌─kill_status────┬─query_id─────────────────────────────┬─user────┬─query───────────────────────────────┐
     1. │ unknown_status │ 6d522fb4-97db-45e4-9565-3158edb5a281 │ default │ SELECT count() FROM system.numbers; │
     2. │ unknown_status │ f38506fd-4923-4bd2-8668-e6affdaa61c4 │ default │ SELECT count() FROM system.numbers; │
        └────────────────┴──────────────────────────────────────┴─────────┴─────────────────────────────────────┘
     ```

     Если у пользователя недостаточно прав на прерывание запросов другого пользователя, клиент выведет соответствующее сообщение, например: `DB::Exception: User adqm_user attempts to kill query created by default. (ACCESS_DENIED)`.

   - Завершение всех выполняющихся в данный момент запросов, длительность которых превышает 90 секунд:

     ```
     KILL QUERY WHERE elapsed > 90;
     ```

KILL MUTATION

Мутации в ADQM/ClickHouse (операции изменения/удаления данных `ALTER TABLE … UPDATE/DELETE`) подразумевают перезапись больших фрагментов данных и могут быть довольно ресурсозатратными. После запуска мутации ее нельзя откатить, выполнение мутации будет продолжаться даже после перезапуска сервера, но можно явно прервать операцию мутации с помощью команды `KILL MUTATION` (например, если мутация выполняется очень долго или в ней больше нет необходимости). Прерывание мутации отменяет ее дальнейшую обработку, но все изменения, уже примененные к данным, остаются — поэтому рекомендуется использовать `KILL MUTATION` осторожно, особенно в production-системах, так как это может повлиять на целостность данных.

Синтаксис запроса `KILL MUTATION`:

```
KILL MUTATION [ON CLUSTER <cluster_name>] WHERE <where_expr> [ASYNC|SYNC|TEST] [FORMAT <output_format>];
```

где:

- `[ON CLUSTER <cluster_name>]` — имя кластера, на всех узлах которого нужно остановить выполнение мутации.
- `WHERE <where_expr>` — обязательное выражение для выбора мутаций, выполнение которых необходимо прервать. Мутации, которые не завершились на текущий момент и потенциально могут быть проблемными, можно посмотреть в таблице system.mutations (строки с условием `is_done = 0`). В выражении `<where_expr>` можно использовать значения столбцов `database`, `table`, `mutation_id` и других столбцов этой таблицы, чтобы указать мутации для прерывания командой `KILL MUTATION`.
- `[ASYNC|SYNC|TEST]` — ключевое слово, указывающее должна ли команда `KILL MUTATION` выполняться асинхронно, синхронно или в тестовом режиме (аналогично KILL QUERY).
- `[FORMAT <output_format>]` — необязательное выражение, указывающее формат вывода результата операции прерывания мутации.

Так же как для `KILL QUERY`, в результате команды `KILL MUTATION` выводится столбец kill_status, значение которого показывает статус выполнения операции прерывания мутации.

Для прерывания мутации через команду `KILL MUTATION` необходимо иметь те же права, что требуются для запуска этой мутации. Например, чтобы остановить мутацию `ALTER DELETE`, необходимо иметь права `ALTER DELETE`, `ALTER TABLE` или `ALTER`. Для выполнения `KILL MUTATION ON CLUSTER` необходимо иметь права на выполнение всех типов мутаций (`ALTER UPDATE`, `ALTER DELETE`, `ALTER MATERIALIZE COLUMN`, `ALTER MATERIALIZE INDEX`, `ALTER MATERIALIZE TTL`) с областью действия `ON *.*`, поскольку при запуске запроса в кластере система сначала проверяет права доступа и не знает какие конкретно мутации будет завершать команда `KILL MUTATION` до начала ее выполнения.

Пример

1. Получите список мутаций, которые находятся в процессе выполнения:

   ```
   SELECT * FROM system.mutations WHERE is_done = 0;
   ```

   Следующий запрос возвращает список активных мутаций в кластере (замените `<cluster_name>` на имя своего кластера):

   ```
   SELECT * FROM clusterAllReplicas('<cluster_name>', system.mutations) WHERE is_done = 0;
   ```

2. Выполните прерывание мутаций, фильтруя их в запросе `KILL MUTATION` по значениям из таблицы `system.mutations`.

   - Прерывание всех незаконченных мутаций в таблице `my_table` базы данных `my_database` в кластере `my_cluster`:

     ```
     KILL MUTATION ON CLUSTER my_cluster WHERE database = 'my_database' AND table = 'my_table' AND is_done = 0;
     ```

   - Завершение конкретной мутации с идентификатором `mutation_2.txt` в таблице `my_table` базы данных `my_database`:

     ```
     KILL MUTATION WHERE database = 'my_database' AND table = 'my_table' AND mutation_id = 'mutation_2.txt' AND is_done = 0;
     ```

     Пример вывода результата запроса на прерывание мутации (где `<mutation_query_text>` — текст команды на изменение/удаление данных, `UPDATE…` или `DELETE…`):

     ```
        ┌─kill_status─┬─database────┬─table────┬─mutation_id────┬─command─────────────────┐
     1. │ waiting     │ my_database │ my_table │ mutation_2.txt │ (<mutation_query_text>) │
        └─────────────┴─────────────┴──────────┴────────────────┴─────────────────────────┘
     ```

Известная проблема

Не все запросы могут быть прерваны командой `KILL QUERY`. Эта команда не останавливает выполнение запроса фактически, а только устанавливает флаг на прерывание, который проверяется самим запросом в определенные моменты его выполнения (обычно после завершения обработки одного блока конвейера выполнения запроса и перед началом другого). Запрос не может быть остановлен, если он "застрял" на этапе, где этот флаг не проверяется (например, при вычислении скалярного значения подзапроса, которое выполняется на этапе анализа запроса, вне основного конвейера выполнения запроса). В случаях, когда запрос невозможно прервать с помощью `KILL QUERY`, единственным способом его завершить может быть перезапуск сервера ClickHouse.

Кодеки шифрования

Елена Дворядкина

Содержание

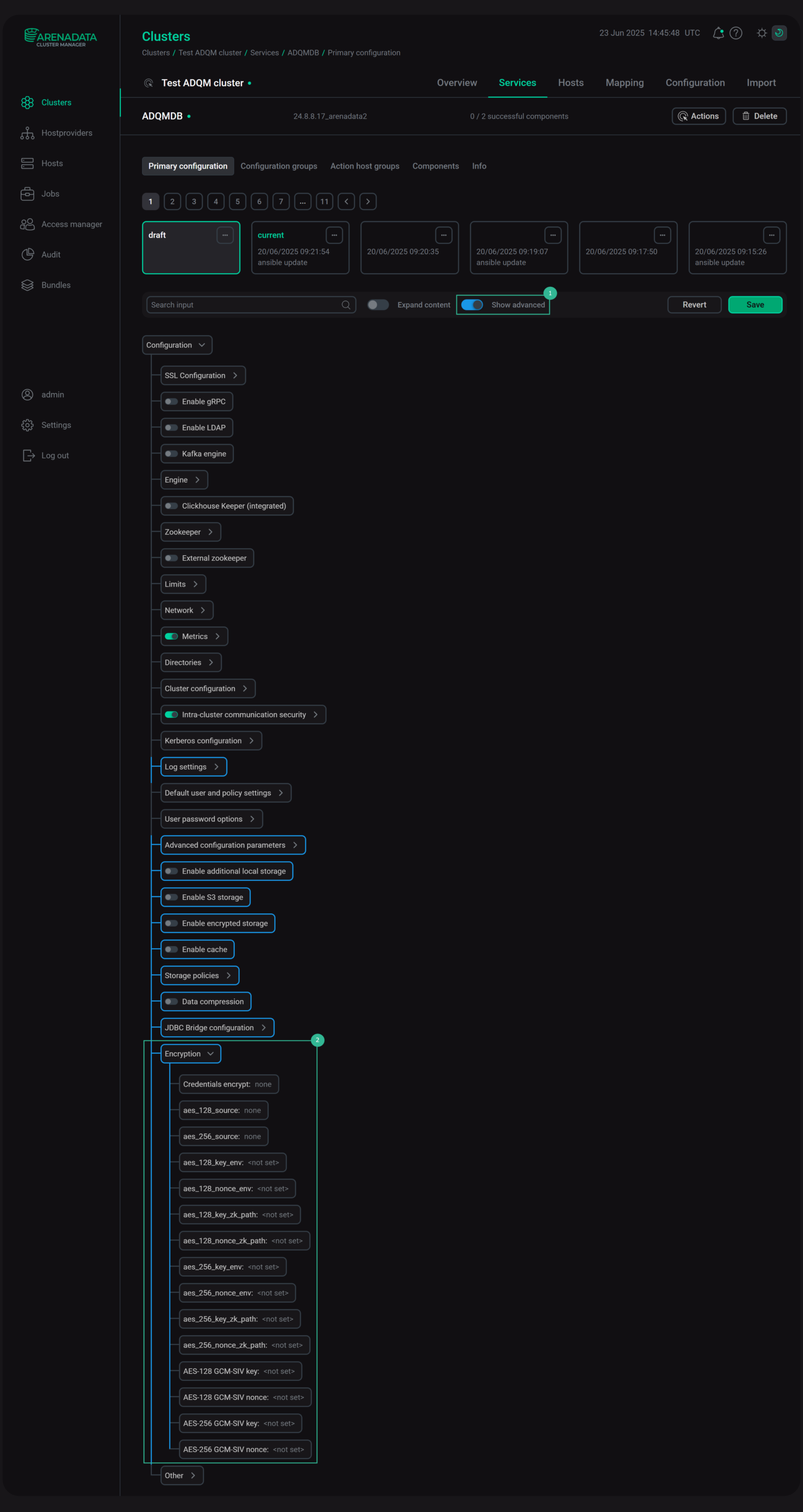
Назад к содержанию

- Настройка ключей шифрования
- Применение алгоритмов шифрования данных
  - Шифрование паролей
  - Шифрование данных столбцов таблиц
- Пример

ADQM поддерживает возможность шифрования конфиденциальных данных (паролей, которые хранятся в конфигурационных файлах ClickHouse) и данных столбцов таблиц на диске с помощью алгоритма AES-128 или AES-256 в режиме GCM-SIV.

Настройка ключей шифрования

Параметры алгоритмов шифрования AES-128 и AES-256 настраиваются на странице конфигурации сервиса ADQMDB в секции ***Encryption***, которая становится видимой при включенной опции ***Show advanced***.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Primary configuration
23 Jun 2025 14:45:48 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
24.8.8.17_arenadata2
0 / 2 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
1 2 3 4 5 6 7 ... 11
draft
current
20/06/2025 09:21:54
ansible update
20/06/2025 09:20:35
20/06/2025 09:19:07
ansible update
20/06/2025 09:17:50
20/06/2025 09:15:26
ansible update
Search input
Expand content
Show advanced
1
Revert
Save
Configuration
SSL Configuration
Enable gRPC
Enable LDAP
Kafka engine
Engine
Clickhouse Keeper (integrated)
Zookeeper
External zookeeper
Limits
Network
Metrics
Directories
Cluster configuration
Intra-cluster communication security
Kerberos configuration
Log settings
Default user and policy settings
User password options
Advanced configuration parameters
Enable additional local storage
Enable S3 storage
Enable encrypted storage
Enable cache
Storage policies
Data compression
JDBC Bridge configuration
2
Encryption
Other

Credentials encrypt: none
aes_128_source: none
aes_256_source: none
aes_128_key_env: <not set>
aes_128_nonce_env: <not set>
aes_128_key_zk_path: <not set>
aes_128_nonce_zk_path: <not set>
aes_256_key_env: <not set>
aes_256_nonce_env: <not set>
aes_256_key_zk_path: <not set>
aes_256_nonce_zk_path: <not set>
AES-128 GCM-SIV key: <not set>
AES-128 GCM-SIV nonce: <not set>
AES-256 GCM-SIV key: <not set>
AES-256 GCM-SIV nonce: <not set>

Настройка алгоритмов шифрования

В параметрах этой секции для нужного алгоритма шифрования укажите ключ и nonce или откуда их загрузить (рекомендуется хранить ключи шифрования в переменных окружения или узлах ZooKeeper, не в конфигурационных файлах). Описание параметров приведено в соответствующем разделе статьи Конфигурационные параметры.

AES-128

- Загрузка ключей из переменных окружения:
  - В параметрах ***aes_128_key_env*** и ***aes_128_nonce_env*** укажите переменные окружения, которые содержат ключ и nonce для алгоритма шифрования соответственно.
  - Для параметра ***aes_128_source*** установите значение `from_env`.
- Загрузка ключей из узлов ZooKeeper:
  - В параметрах ***aes_128_key_zk_path*** и ***aes_128_nonce_zk_path*** укажите узлы ZooKeeper, которые содержат ключ и nonce для алгоритма шифрования соответственно.
  - Для параметра ***aes_128_source*** установите значение `from_zk`.
- Хранение ключей в конфигурационном файле (не рекомендуется):
  - В параметрах ***AES-128 GCM-SIV key*** и ***AES-128 GCM-SIV nonce*** укажите ключ и nonce для алгоритма шифрования соответственно.
  - Для параметра ***aes_128_source*** установите значение `none`.

AES-256

- Загрузка ключей из переменных окружения:
  - В параметрах ***aes_256_key_env*** и ***aes_256_nonce_env*** укажите переменные окружения, которые содержат ключ и nonce для алгоритма шифрования соответственно.
  - Для параметра ***aes_256_source*** установите значение `from_env`.
- Загрузка ключей из узлов ZooKeeper:
  - В параметрах ***aes_256_key_zk_path*** и ***aes_256_nonce_zk_path*** укажите узлы ZooKeeper, которые содержат ключ и nonce для алгоритма шифрования соответственно.
  - Для параметра ***aes_256_source*** установите значение `from_zk`.
- Хранение ключей в конфигурационном файле (не рекомендуется):
  - В параметрах ***AES-256 GCM-SIV key*** и ***AES-256 GCM-SIV nonce*** укажите ключ и nonce для алгоритма шифрования соответственно.
  - Для параметра ***aes_256_source*** установите значение `none`.

После назначения параметров нажмите ***Save*** и выполните действие ***Reconfig*** для сервиса ADQMDB.

Настройки алгоритмов шифрования сохраняются в секции `encryption_codecs` конфигурационного файла */etc/clickhouse-server/config.d/encryption.xml*.

Применение алгоритмов шифрования данных

После того как для алгоритмов AES-128/AES-256 указаны ключи, можно включить шифрование:

- паролей, записанных в конфигурационном файле;
- данных столбцов таблиц на диске.

Шифрование паролей

В секции ***Encryption*** конфигурации сервиса ADQMDB установите значение параметра ***Credentials encrypt*** равным `AES-128` или `AES-128`. Сохраните конфигурацию и выполните сервисное действие ***Reconfig***.

После этого в конфигурационном файле */etc/clickhouse-server/credentials.xml* значения следующих параметров (если они настроены) будут храниться в зашифрованном виде:

- `interserver_http_password` — пароль для аутентификации между репликами;
- `shard_secret` — ключ кластера для проверки распределенных запросов на шардах;
- `default_user_password` — пароль пользователя default.

Пример файла credentials.xml

```xml
<clickhouse>
    <interserver_http_password>
        <password hide_in_preprocessed="true" encrypted_by="AES_256_GCM_SIV">...encrypted value...</password>
    </interserver_http_password>
    <shard_secret>
        <secret hide_in_preprocessed="true" encrypted_by="AES_256_GCM_SIV">...encrypted value...</secret>
    </shard_secret>
    <default_user_password>
        <password hide_in_preprocessed="true" encrypted_by="AES_256_GCM_SIV">...encrypted value...</password>
    </default_user_password>
</clickhouse>
```

ВНИМАНИЕ

Для шифрования паролей в настоящее время не поддерживается загрузка ключей из узлов ZooKeeper, если для соединений между ClickHouse и ZooKeeper используется SSL.

Шифрование данных столбцов таблиц

В ClickHouse алгоритм шифрования данных, применяемый к столбцу таблицы, определяется **кодеком шифрования**. Назначить кодек шифрования столбцу можно в запросе CREATE TABLE с помощью выражения `CODEC`:

```sql
CREATE TABLE <table_name>
(   <column_name1> <data_type1> CODEC(AES_128_GCM_SIV | AES_256_GCM_SIV) [...],
    <column_name2> <data_type2> CODEC(AES_128_GCM_SIV | AES_256_GCM_SIV) [...],
    ...)
ENGINE = MergeTree()
...;
```

Кодеки шифрования фактически только шифруют данные столбца на диске, но не сжимают их. Если требуется применить сжатие, это нужно явно указать для столбца — для этого в выражении `CODEC` перед кодеком шифрования перечислите через запятую нужные кодеки сжатия данных (сначала специализированные, затем кодеки общего назначения). Например:

```sql
CREATE TABLE <table_name>
(   <column_name> <data_type> CODEC(Delta, LZ4, AES_256_GCM_SIV) [...],
    ...)
ENGINE = MergeTree()
...;
```

ВНИМАНИЕ

- Большинство табличных движков, включая движки семейства MergeTree, создают индексные файлы на диске без применения кодеков шифрования. Это значит, что на диске могут появиться данные в открытом виде, если зашифрованный столбец индексируется.
- Если в запросе `SELECT` содержится определенное значение из зашифрованного столбца (например, в выражении `WHERE`), это значение может появиться в лог-таблице `system.query_log`. Возможно, в этом случае стоит отключить логирование.

Пример

Следующий пример показывает, как указать сохраненный в ZooKeeper ключ для алгоритма шифрования данных AES-256 в ADQM, а также как применить этот алгоритм для шифрования пароля пользователя `default` (должен быть установлен — см. Настройка авторизации для нового кластера) и данных столбца тестовой таблицы.

1. Создайте два узла ZooKeeper, в которых сохраните ключ и nonce для алгоритма шифрования AES-256 (для этого можно использовать консоль ZooKeeper CLI доступную на каждом хосте кластера, где установлен ZooKeeper):

   ```
   $ sudo /usr/lib/zookeeper/bin/zkCli.sh
   ```

   ```
   create /clickhouse/clickhouse/aes256_key_hex '00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF'
   create /clickhouse/clickhouse/aes256_nonce_hex 'ABCDEFABCDEFABCDEFABCDEF'
   ```

2. В секции ***Encryption*** конфигурации сервиса ADQMDB установите следующие значения параметров:

   - ***aes_256_key_zk_path***: `/clickhouse/aes256_key_hex`;
   - ***es_256_nonce_zk_path***: `/clickhouse/aes256_nonce_hex`;
   - ***aes_256_source***: `from_zk`;
   - ***Credentials encrypt***: `AES-256`.

   Сохраните конфигурацию и запустите выполнение действия ***Reconfig*** для сервиса ADQMDB.

   Убедитесь, что настройки кодека шифрования сохранились в файле */etc/clickhouse-server/config.d/encryption.xml*:

   ```xml
   <clickhouse>
       <encryption_codecs>
           <aes_256_gcm_siv>
               <key_hex hide_in_preprocessed="true" from_zk="/clickhouse/aes256_key_hex"/>
               <nonce_hex hide_in_preprocessed="true" from_zk="/clickhouse/aes256_nonce_hex"/>
           </aes_256_gcm_siv>
       </encryption_codecs>
   </clickhouse>
   ```

3. Проверьте, что пароль пользователя `default` в конфигурационном файле */etc/clickhouse-server/credentials.xml* теперь хранится в зашифрованном виде:

   ```xml
   <clickhouse>
       <default_user_password>
           <password hide_in_preprocessed="true" encrypted_by="AES_256_GCM_SIV">...encrypted password...</password>
       </default_user_password>
   </clickhouse>
   ```

4. Создайте таблицу ADQM и вставьте в нее тестовые данные:

   ```sql
   CREATE TABLE encryption_test (x String CODEC(AES_256_GCM_SIV), y String) ENGINE = MergeTree ORDER BY x;
   INSERT INTO encryption_test VALUES ('1', '10'), ('2', '20'), ('3', '30');
   ```

   Посмотрите содержимое файла */var/lib/clickhouse/data/default/encryption_test/all_1_1_0/data.bin*, чтобы убедиться, что данные первого столбца таблицы хранятся на диске в зашифрованном виде.

Функции ADQM для машинного обучения

Елена Дворядкина

Содержание

Назад к содержанию

- Основные понятия
- Алгоритмы машинного обучения
  - Простая линейная регрессия
  - Стохастическая регрессия
- Использование функций машинного обучения
  - Построение модели
  - Прогнозирование
- Пример
  - Подготовка данных
  - Построение модели машинного обучения
  - Оценка модели

ADQM/ClickHouse предоставляет набор функций для машинного обучения — агрегатные функции, реализующие алгоритмы машинного обучения, которые можно обучить на известных наборах данных, а затем использовать для прогнозирования неизвестных значений.

Основные понятия

В этой статье алгоритмы машинного обучения и особенности применения соответствующих функций ADQM описаны с использованием следующей терминологии:

- **Независимые и зависимые переменные.** Независимые переменные являются входными данными для алгоритма машинного обучения. Зависимые переменные — это выходные (прогнозируемые, целевые) данные, которые зависят от независимых переменных. Другими словами, независимые переменные — это признаки (features), описывающие характеристики объекта или явления, по значениям которых можно прогнозировать значение целевой переменной (target) с помощью обученной модели машинного обучения.
- **Обучающий и тестовый наборы данных.** Обучающий набор данных (training data) — данные, которые используются для обучения модели машинного обучения. Обучающие данные должны включать столбцы со значениями независимых переменных и столбец с целевым значением, которое необходимо научиться предсказывать. Тестовый набор данных (test data) — другой набор данных, который не зависит от обучающих данных и используется для оценки модели, построенной с помощью обучающего набора данных.
- **Обучение модели.** Алгоритм машинного обучения находит закономерности в предоставленных обучающих данных, сопоставляя значения независимых переменных с целевым значением, и выводит модель машинного обучения, которая фиксирует эти закономерности. Затем модель можно использовать для получения прогнозов на основе новых данных, для которых значения зависимой переменной неизвестны.
- **Параметры модели или веса (коэффициенты) регрессии.** Параметры модели (или веса регрессии) определяют, как независимые переменные влияют на зависимую переменную. Например, в функциональном уравнении линейной регрессии $y = a_1 \cdot x_1 + a_2 \cdot x_2 + ... + b$ параметрами модели являются константы $a_1, a_2, ..., b$ (b — свободный коэффициент или смещение). Параметры модели автоматически вычисляются в процессе обучения модели на основе обучающих данных и возвращаются функцией машинного обучения ADQM.
- **Гиперпараметры модели.** Гиперпараметры — характеристики модели, которые вручную настраиваются до начала процесса обучения и определяют структуру модели и способ ее обучения. Правильный подбор гиперпараметров может значительно повлиять на эффективность модели и точность результатов обучения.

Алгоритмы машинного обучения

В ADQM функции машинного обучения реализуют следующие алгоритмы регрессии.

Простая линейная регрессия

Функция `simpleLinearRegression(<feature>, <target>)` выполняет простую линейную регрессию, то есть вычисляет линейную зависимость между двумя переменными — одной независимой и целевой. В качестве параметров `<feature>` и `<target>` в функцию передаются столбцы обучающего набора данных со значениями независимой и целевой переменных соответственно.

Стохастическая регрессия

Для прогнозирования значений целевой переменной на основе нескольких независимых переменных ADQM предоставляет следующие функции множественной регрессии, использующие **стохастический градиентный спуск** в качестве способа обучения модели:

- `stochasticLinearRegression([<hyperparameters>])(<target>, <feature1>, <feature2>, …)` — вычисляет линейную зависимость между целевой переменной (`<target>`) и набором независимых переменных (`<feature1>, <feature2>, …`) и позволяет получить прогноз непрерывных числовых значений зависимой переменной на основе наблюдаемых значений независимых переменных.
- `stochasticLogisticRegression([<hyperparameters>])(<target>, <feature1>, <feature2>, …)` — позволяет оценить вероятность наступления некоторого события по значениям множества признаков. Зависимая переменная может принимать одно из двух значений — обычно это `0` (событие не произошло) и `1` (событие произошло). Используется для задач бинарной классификации.

При использовании функции стохастической регрессии для построения модели машинного обучения можно указать четыре гиперпараметра (`<hyperparameters>` в приведенных выше описаниях синтаксиса функций). В следующей таблице гиперпараметры перечислены в том порядке, в котором они должны последовательно передаваться в функцию.

Гиперпараметр	Описание	Значение по умолчанию
learning rate	Скорость обучения, которая указывается как коэффициент длины шага при выполнении градиентного спуска. Слишком большая скорость обучения может привести к бесконечным весам модели	0.00001
l2 regularization coefficient	Коэффициент регуляризации L2, который позволяет избежать переобучения (мешает модели подгоняться под обучающие данные)	0.1
mini-batch size	Количество элементов обучающей выборки, градиенты которых будут вычисляться и суммироваться при выполнении одного шага градиентного спуска. Чистый стохастический спуск использует один элемент, однако использование мини-пакета из фиксированного небольшого числа элементов (около 10 элементов) делает градиентные шаги более стабильными	15
method for updating weights	Метод обновления весов. Возможные значения: `Adam` (по умолчанию), `SGD`, `Momentum` и `Nesterov`. Последние два метода более требовательны к вычислительным ресурсам и памяти, однако они имеют высокую скорость сходимости и устойчивость методов стохастического градиента	Adam

Использование функций машинного обучения

Функции машинного обучения используются на двух этапах: построение модели и прогнозирование новых данных.

Построение модели

Для построения модели необходимо выполнить следующие действия:

1. Создайте таблицу на основе набора данных для обучения. Пример запроса:

   ```
   CREATE TABLE train_data
   (
       feature1 Float64,
       feature2 Float64,
       target Float64
   )
   ENGINE = MergeTree;
   ```

   Для линейной регрессии независимые (`feature1`, `feature2` ...) и целевая (`target`) переменные должны быть числовыми. Для логистической регрессии целевая переменная должна быть бинарной (то есть принимает только два значения — например, `0` и `1`), а прогнозируемые значения (вероятности) будут находиться в диапазоне от `0` до `1`.

2. Используйте функцию машинного обучения с комбинатором `State`, чтобы построить модель и сохранить ее состояние для дальнейшего использования. Пример запроса с функцией `stochasticLinearRegression`:

   ```
   CREATE TABLE my_model ENGINE = Memory AS SELECT
   stochasticLinearRegressionState(0.001, 0.1, 10, 'SGD')(target, feature1, feature2)
   AS state FROM train_data;
   ```

   Если не применять комбинатор `State` к функции машинного обучения, можно получить параметры созданной модели без ее сохранения. Например, следующий запрос построит модель и вернет ее параметры (столбец, в котором первые значения соответствуют весам регрессии, а последнее — смещению):

   ```
   stochasticLinearRegression(0.001, 0.1, 10, 'SGD')(target, feature1, feature2) FROM train_data;
   ```

Прогнозирование

Сохраненное в таблице состояние модели можно использовать несколько раз для прогнозирования или объединить с другими состояниями и создать новые (улучшенные) модели.

1. Создайте таблицу на основе набора данных для тестирования (`test_data` в приведенном ниже примере синтаксиса). Эта таблица по структуре должна быть аналогична `train_data`, но может не содержать целевое (прогнозируемое) значение.
2. Для прогнозирования используйте функцию `evalMLMethod`, которая принимает в качестве аргументов состояние модели и признаки для прогнозирования. Например, следующий запрос возвращает столбец значений целевой переменной, прогнозируемых по значениям признаков `feature1` и `feature2` в таблице `test_data`:

   ```
   WITH (SELECT state FROM my_model) AS model SELECT
   evalMLMethod(model, feature1, feature2) FROM test_data;
   ```

Пример

Подготовка данных

1. Создайте таблицы с данными для обучения (`trips_train`) и тестирования (`trips_test`), как описано в примере New York Taxi Data документации ClickHouse:

   ```
   CREATE TABLE trips_train (
       trip_id             UInt32,
       pickup_datetime     DateTime,
       dropoff_datetime    DateTime,
       pickup_longitude    Nullable(Float64),
       pickup_latitude     Nullable(Float64),
       dropoff_longitude   Nullable(Float64),
       dropoff_latitude    Nullable(Float64),
       passenger_count     UInt8,
       trip_distance       Float32,
       fare_amount         Float32,
       extra               Float32,
       tip_amount          Float32,
       tolls_amount        Float32,
       total_amount        Float32,
       payment_type        Enum('CSH' = 1, 'CRE' = 2, 'NOC' = 3, 'DIS' = 4, 'UNK' = 5),
       pickup_ntaname      LowCardinality(String),
       dropoff_ntaname     LowCardinality(String)
   )
   ENGINE = MergeTree
   PRIMARY KEY (pickup_datetime, dropoff_datetime);
   ```

   ```
   CREATE TABLE trips_test (
       trip_id             UInt32,
       pickup_datetime     DateTime,
       dropoff_datetime    DateTime,
       pickup_longitude    Nullable(Float64),
       pickup_latitude     Nullable(Float64),
       dropoff_longitude   Nullable(Float64),
       dropoff_latitude    Nullable(Float64),
       passenger_count     UInt8,
       trip_distance       Float32,
       fare_amount         Float32,
       extra               Float32,
       tip_amount          Float32,
       tolls_amount        Float32,
       total_amount        Float32,
       payment_type        Enum('CSH' = 1, 'CRE' = 2, 'NOC' = 3, 'DIS' = 4, 'UNK' = 5),
       pickup_ntaname      LowCardinality(String),
       dropoff_ntaname     LowCardinality(String)
   )
   ENGINE = MergeTree
   PRIMARY KEY (pickup_datetime, dropoff_datetime);
   ```

2. Заполните таблицы `trips_train` и `trips_test` данными из файлов *trips_0.gz* и *trips_1.gz* хранилища S3 соответственно:

   ```
   INSERT INTO trips_train
   SELECT
       trip_id,
       pickup_datetime,
       dropoff_datetime,
       pickup_longitude,
       pickup_latitude,
       dropoff_longitude,
       dropoff_latitude,
       passenger_count,
       trip_distance,
       fare_amount,
       extra,
       tip_amount,
       tolls_amount,
       total_amount,
       payment_type,
       pickup_ntaname,
       dropoff_ntaname
   FROM s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/nyc-taxi/trips_0.gz', 'TabSeparatedWithNames');
   ```

   ```
   INSERT INTO trips_test
   SELECT
       trip_id,
       pickup_datetime,
       dropoff_datetime,
       pickup_longitude,
       pickup_latitude,
       dropoff_longitude,
       dropoff_latitude,
       passenger_count,
       trip_distance,
       fare_amount,
       extra,
       tip_amount,
       tolls_amount,
       total_amount,
       payment_type,
       pickup_ntaname,
       dropoff_ntaname
   FROM s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/nyc-taxi/trips_1.gz', 'TabSeparatedWithNames');
   ```

Построение модели машинного обучения

1. Создайте модель машинного обучения для прогнозирования стоимости проезда в такси (целевая переменная — `fare_amount`) в зависимости от количества пассажиров и дальности поездки (независимые переменные — `passenger_count` и `trip_distance`):

   ```
   SELECT stochasticLinearRegression(0.001, 0.1, 5)(fare_amount, passenger_count, trip_distance)
   FROM trips_train;
   ```

   В результате выполнения этого запроса выводятся параметры модели, которые определяют коэффициенты уравнения линейной регрессии ($\text{fare\_amount} = 0.08 \cdot \text{passenger\_count} + 2.7 \cdot \text{trip\_distance} + 4.9$):

   ```
   ┌─stochasticLinearRegression(0.001, 0.1, 5)(fare_amount, passenger_count, trip_distance)─┐
   │ [0.08194997204590103,2.706812913667359,4.897836872083942]                              │
   └────────────────────────────────────────────────────────────────────────────────────────┘

   1 row in set. Elapsed: 0.036 sec. Processed 1.00 million rows, 9.01 MB (27.71 million rows/s., 249.35 MB/s.)
   Peak memory usage: 2.38 MiB.
   ```

   Как видно, стоимость проезда не сильно зависит от количества пассажиров.

2. Сохраните состояние модели в таблицу `taxi_trips_model` на базе движка Memory:

   ```
   CREATE TABLE taxi_trips_model ENGINE = Memory AS
   SELECT stochasticLinearRegressionState(0.001, 0.1, 5)(fare_amount, passenger_count, trip_distance)
   AS state FROM trips_train;
   ```

Оценка модели

Используйте обученную модель для получения прогнозов по тестовому набору данных и сравните эти прогнозы с фактическими значениями (обратите внимание, в функцию `evalMLMethod` передается состояние модели и тот же набор независимых переменных, который использовался для обучения — `passenger_count` и `trip_distance`):

```
WITH (SELECT state FROM taxi_trips_model) AS model
SELECT
    fare_amount,
    evalMLMethod(model, passenger_count, trip_distance) AS prediction
FROM trips_test LIMIT 10;
```

В результирующей таблице видно, что полученные прогнозы довольно близки к фактическим данным:

```
┌─fare_amount─┬─────────prediction─┐
│         8.5 │   9.20394916259993 │
│        13.5 │ 13.714548983540645 │
│        41.5 │  41.78716293769729 │
│          24 │ 22.230999964014817 │
│         8.5 │  9.018605114531145 │
│         6.5 │ 7.6682400731869595 │
│        19.5 │ 21.287184808552087 │
│          19 │ 20.377558220769647 │
│         7.5 │ 7.8100114849842495 │
│           9 │  8.954756513157905 │
└─────────────┴────────────────────┘
```

Для более точных прогнозов можно подобрать другие гиперпараметры модели.

JDBC Bridge

Елена Дворядкина

Содержание

Назад к содержанию

- Установка
- Настройка
- Использование
  - Табличная функция jdbc
  - Движок таблиц JDBC
- Пример

ClickHouse JDBC Bridge позволяет обращаться из ADQM к внешним источникам данных, которые поддерживают работу через интерфейс JDBC.

Установка

ClickHouse JDBC Bridge в ADQM — это компонент сервиса ADQMDB, который устанавливается так же, как другие компоненты. В статье Добавление компонентов подробно описаны действия, которые необходимо выполнить, чтобы развернуть компоненты ADQM на хостах кластера через интерфейс ADCM.

ВАЖНО

При установке компонента ClickHouse JDBC Bridge его необходимо добавить на все хосты кластера, где установлен компонент Clickhouse Server.

Настройка

Для подключения ADQM к внешнему источнику данных (data source) через JDBC Bridge нужно добавить описание этого источника под некоторым именем в поле ***datasource.json***, которое находится на странице конфигурации сервиса ADQMDB в секции ***JDBC Bridge configuration*** (секция доступна при включенной опции ***Show advanced***) интерфейса ADCM.

По умолчанию в конфигурации JDBC Bridge прописаны два источника данных (`postgres13` для PostgreSQL и `ch-server` для ClickHouse):

```
[
    {
        "$schema": "./datasource.jschema",
        "postgres13": {
            "converter": {
                "mappings": [
                    {
                        "nativeType": "bool",
                        "toType": "String"
                    }
                ]
            },
            "driverUrls": [
                "drivers/postgresql/postgresql-jdbc.jar"
            ],
            "driverClassName": "org.postgresql.Driver",
            "jdbcUrl": "jdbc:postgresql://localhost/test",
            "username": "",
            "password": "",
            "initializationFailTimeout": 0,
            "minimumIdle": 0,
            "maximumPoolSize": 10
        }
    },
    {
        "$schema": "./datasource.jschema",
        "ch-server": {
            "driverUrls": [
                "drivers/clickhouse/clickhouse-jdbc.jar"
            ],
            "driverClassName": "ru.yandex.clickhouse.ClickHouseDriver",
            "jdbcUrl": "jdbc:clickhouse://ch-server:8123/system",
            "username": "default",
            "password": "",
            "initializationFailTimeout": 0,
            "minimumIdle": 0,
            "maximumPoolSize": 10
        }
    }
]
```

Эти конфигурации можно изменить или использовать как шаблоны для описания новых источников данных, соблюдая формат JSON. Например, для PostgreSQL необходимо указать реальные значения `username` и `password`, а также в значении ключа `jdbcUrl` заменить `localhost` на хост, на котором установлен PostgreSQL, и `test` — на нужную базу данных.

Для удобства ниже приведен шаблон конфигурации источника данных в общем виде — вы можете его скопировать и вставить в поле ***datasource.json***, заполнив реальными данными для внешней СУБД:

```
{
  "<datasource_name>": {
  "driverUrls": [
    "<datasource_jdbc_driver_url>"
  ],
  "jdbcUrl": "jdbc:<driver_name>://<host>:<port>/<database>",
  "username": "<username>",
  "password": "<password>"
  }
}
```

где:

- `<datasource_name>` — произвольное имя источника данных, которое будет использоваться при указании источника в табличной функции jdbc или при создании таблицы на базе движка JDBC;
- `<datasource_jdbc_driver_url>` — URL-адрес, с которого можно загрузить JDBC-драйвер для соответствующей СУБД;
- `<driver_name>` — название драйвера;
- `<host>:<port>` — хост и порт, где установлена СУБД;
- `<database>` — имя базы данных в СУБД (данный элемент не является обязательным — если его пропустить, базу данных нужно будет явно указывать при каждом подключении ADQM к внешнему источнику данных);
- `<username>` и `<password>` — учетные данные для доступа в СУБД.

ВАЖНО

После завершения редактирования конфигурации не забудьте нажать ***Save*** и выполнить действие ***Reconfig*** с включенной опцией ***Restart*** для сервиса ADQMDB, чтобы сохранить изменения и перезапустить сервис.

Получить список именованных источников данных, настроенных для JDBC Bridge, можно с помощью следующего запроса:

```
SELECT name FROM jdbc('', 'show datasources');
```

Пример вывода на экран:

```
┌─name────────┐
│ ch-server   │
│ self_server │
│ postgres13  │
│ script      │
└─────────────┘
```

Использование

Взаимодействие ADQM с внешней СУБД по JDBC-интерфейсу осуществляется через табличную функцию jdbc или табличный движок JDBC.

Табличная функция jdbc

Табличная функция `jdbc` возвращает таблицу, подключенную к внешнему источнику данных через JDBC-драйвер. Базовый синтаксис запроса с вызовом функции:

```
SELECT ... FROM jdbc('<datasource>', ['<external_database_name | schema>'], '<external_table_name | query>');
```

где:

- `<datasource>` — имя внешнего источника данных, указанное в конфигурации JDBC Bridge;
- `<external_database_name | schema>` — имя базы данных во внешнем источнике или схема таблицы (необязательный параметр);
- `<external_table_name | query>` — имя таблицы во внешней базе данных или запрос на выборку данных из внешней таблицы в нативном формате внешнего источника данных.

Движок таблиц JDBC

Для интеграции с внешним источниками данных через JDBC-интерфейс можно также использовать специальный табличный движок — JDBC. Синтаксис запроса для создания таблицы на базе этого движка в общем виде:

```
CREATE TABLE [IF NOT EXISTS] [<db_name>.]<jdbc_table_name> (<column_name> <column_type>, ...)
ENGINE = JDBC('<datasource>', '<external_database_name>', '<external_table_name | query>');
```

где:

- `<column_name> <column_type>, …` — описание столбцов таблицы (схема таблицы должна быть согласована со схемой внешней таблицы `<external_table_name>`, к которой осуществляется подключение, или с таблицей, которую возвращает запрос `<query>` — то есть имена и порядок столбцов должны быть одинаковыми, а типы столбцов должны быть совместимыми);
- `<datasource>` — имя внешнего источника данных, указанное в конфигурации JDBC Bridge;
- `<external_db_name>` — имя базы данных во внешнем источнике (оставьте этот параметр пустым — `''`, если имя базы данных прописано в конфигурации источника данных или явно указывается в запросе `<query>`, передаваемом в качестве третьего параметра движка JDBC);
- `<external_table_name | query>` — имя таблицы в базе данных `external_db_name` или запрос на выборку данных из внешней таблицы (например, `SELECT * FROM test_table WHERE column_1 = 1`).

Пример

1. На сервере MariaDB создайте таблицу в базе данных `mariadb_db`:

   ```
   CREATE TABLE mariadb_table (id INT PRIMARY KEY, date DATETIME);
   ```

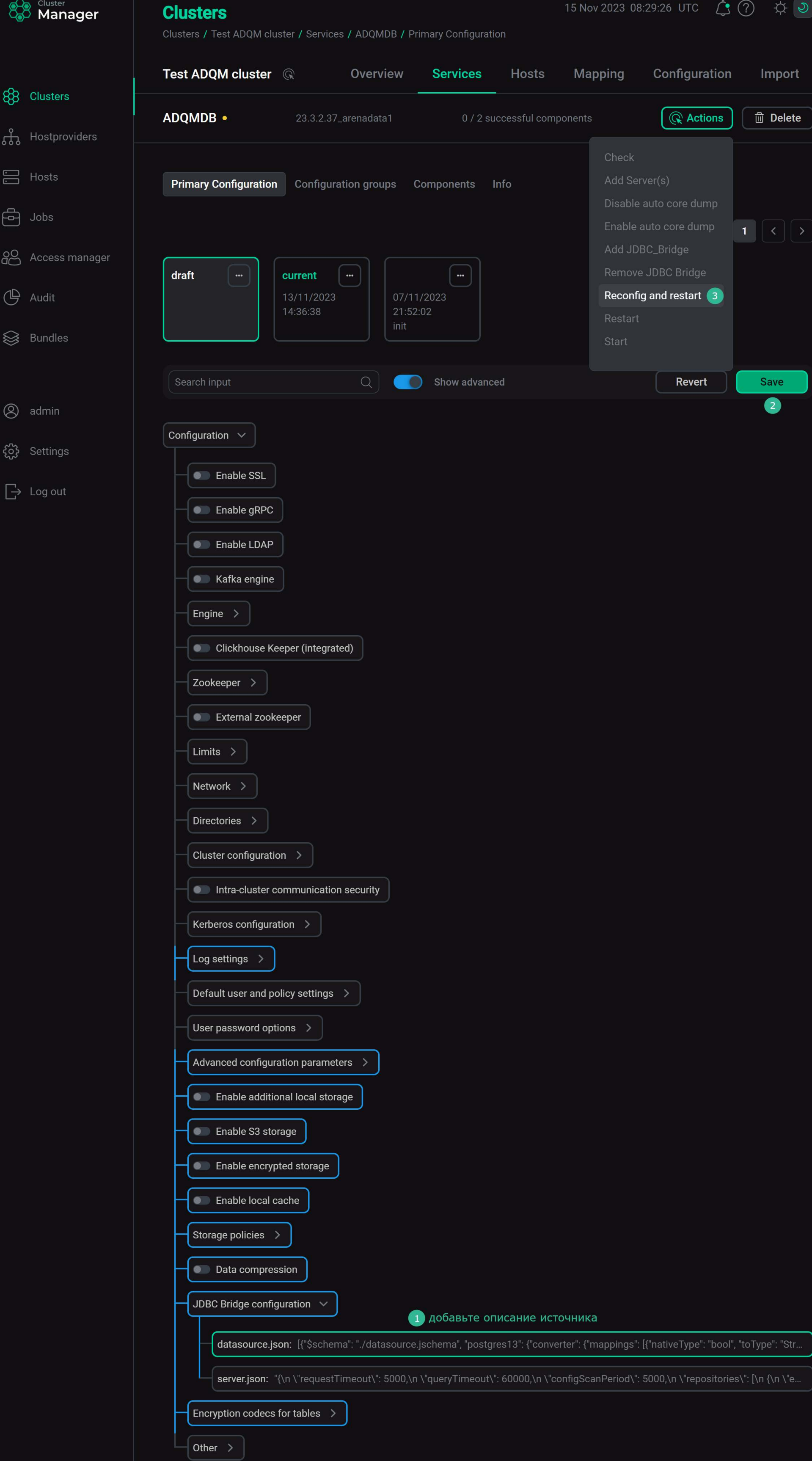
   Вставьте в таблицу тестовые данные:

   ```
   INSERT INTO mariadb_table VALUES (1, "2023-09-25 11:12:13"), (2, "2023-09-26 12:13:14"), (3, "2023-09-27 13:14:15");
   ```

2. В текст параметра ***datasource.json*** конфигурации JDBC Bridge добавьте следующее описание источника данных MariaDB (замените `<host>:<port>` на адрес сервера MariaDB, а `<user_name>` и `<user_password>` на имя пользователя и пароль для доступа к базе данных MariaDB):

   ```
   {
       "$schema": "./datasource.jschema",
       "mariadb10": {
           "driverUrls": [
           "https://repo1.maven.org/maven2/org/mariadb/jdbc/mariadb-java-client/2.7.4/mariadb-java-client-2.7.4.jar"
           ],
           "driverClassName": "org.mariadb.jdbc.Driver",
           "jdbcUrl": "jdbc:mariadb://<host>:<port>",
           "dataSource": {
           "user": "<user_name>",
           "password": "<user_password>"
           },
           "initializationFailTimeout": 0,
           "minimumIdle": 0,
           "maximumPoolSize": 10
       }
   }
   ```

   Нажмите ***Save*** и выполните действие ***Reconfig*** с опцией ***Restart*** для сервиса ADQMDB, чтобы сохранить информацию об источнике данных MariaDB в конфигурации JDBC Bridge и перезапустить сервис.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
15 Nov 2023 08:29:26 UTC
Clusters / Test ADQM cluster / Services / ADQMDB / Primary Configuration
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
23.3.2.37_arenadata1
0 / 2 successful components
Actions
Delete
Check
Add Server(s)
Disable auto core dump
Enable auto core dump
Add JDBC_Bridge
Remove JDBC Bridge
Reconfig and restart
3
Restart
Start
Primary Configuration
Configuration groups
Components
Info
draft
current
13/11/2023 14:36:38
07/11/2023 21:52:02 init
1
Search input
Show advanced
Revert
Save
2
Configuration
Enable SSL
Enable gRPC
Enable LDAP
Kafka engine
Engine
Clickhouse Keeper (integrated)
Zookeeper
External zookeeper
Limits
Network
Directories
Cluster configuration
Intra-cluster communication security
Kerberos configuration
Log settings
Default user and policy settings
User password options
Advanced configuration parameters
Enable additional local storage
Enable S3 storage
Enable encrypted storage
Enable local cache
Storage policies
Data compression
JDBC Bridge configuration
1 добавьте описание источника
Encryption codecs for tables
Other

datasource.json: [{"$schema": "./datasource.jschema", "postgres13": {"converter": {"mappings": [{"nativeType": "bool", "toType": "Str...

server.json: "{\n \"requestTimeout\": 5000,\n \"queryTimeout\": 60000,\n \"configScanPeriod\": 5000,\n \"repositories\": [\n {\n \"e...

Описание источника данных для JDBC Bridge

3. Подключитесь из ADQM к таблице `mariadb_table` внешней базы данных `mariadb_db` с помощью табличной функции или табличного движка JDBC, используя в обоих случаях имя сконфигурированного выше источника данных в качестве значения первого параметра.

   - Табличная функция jdbc:

     ```
     SELECT * FROM jdbc('mariadb10', 'mariadb_db', 'mariadb_table');
     ```

     ```
     ┌─id─┬────────────────date─┐
     │  1 │ 2023-09-25 11:12:13 │
     │  2 │ 2023-09-26 12:13:14 │
     │  3 │ 2023-09-27 13:14:15 │
     └────┴─────────────────────┘
     ```

     ```
     SELECT * FROM jdbc('mariadb10', 'SELECT * FROM mariadb_db.mariadb_table WHERE id >= 2');
     ```

     ```
     ┌─id─┬────────────────date─┐
     │  2 │ 2023-09-26 12:13:14 │
     │  3 │ 2023-09-27 13:14:15 │
     └────┴─────────────────────┘
     ```

     ```
     SELECT * FROM jdbc('mariadb10', 'test_dates DateTime', 'SELECT date as test_dates FROM mariadb_db.mariadb_table');
     ```

     ```
     ┌──────────test_dates─┐
     │ 2023-09-25 11:12:13 │
     │ 2023-09-26 12:13:14 │
     │ 2023-09-27 13:14:15 │
     └─────────────────────┘
     ```

   - Табличный движок JDBC:

     ```
     CREATE TABLE jdbc_table (id Int32, date DateTime) ENGINE JDBC('mariadb10', 'mariadb_db', 'mariadb_table');
     ```

     ```
     SELECT * FROM jdbc_table;
     ```

     ```
     ┌─id─┬────────────────date─┐
     │  1 │ 2023-09-25 11:12:13 │
     │  2 │ 2023-09-26 12:13:14 │
     │  3 │ 2023-09-27 13:14:15 │
     └────┴─────────────────────┘
     ```

     ```
     CREATE TABLE jdbc_table_1 (test_IDs Int32, test_dates DateTime)
     ENGINE JDBC('mariadb10', '', 'SELECT id as test_IDs, date as test_dates FROM mariadb_db.mariadb_table');
     ```

     ```
     SELECT * FROM jdbc_table_1;
     ```

     ```
     ┌─test_IDs─┬──────────test_dates─┐
     │        1 │ 2023-09-25 11:12:13 │
     │        2 │ 2023-09-26 12:13:14 │
     │        3 │ 2023-09-27 13:14:15 │
     └──────────┴─────────────────────┘
     ```

   Убедитесь также, что через JDBC Bridge можно записывать данные из ADQM во внешнюю базу данных. Для этого добавьте новую запись в таблицу ADQM:

   ```
   INSERT INTO jdbc_table VALUES (5, '2023-09-28 14:15:16');
   ```

   Затем проверьте содержимое таблицы на сервере MariaDB:

   ```
   SELECT * FROM mariadb_table;
   ```

   ```
   +----+---------------------+
   | id | date                |
   +----+---------------------+
   |  1 | 2023-09-25 11:12:13 |
   |  2 | 2023-09-26 12:13:14 |
   |  3 | 2023-09-27 13:14:15 |
   |  5 | 2023-09-28 14:15:16 |
   +----+---------------------+
   ```

Интеграция ADQM и HDFS

Елена Дворядкина

Содержание

Назад к содержанию

Табличный движок HDFS
  Пример
Табличные функции
  Пример
Символы подстановки в пути к файлу HDFS
Режим высокой доступности (HA) HDFS
Пример доступа ADQM к HDFS керберизованного кластера ADH
  Проверка аутентификации и создание тестовых данных в HDFS
  Создание принципала пользователя для подключения ADQM к HDFS
  Настройка ADQM
  Создание таблицы в ADQM

Для интеграции с распределенной файловой системой HDFS (Hadoop Distributed File System) в ClickHouse реализованы табличный движок HDFS и специальные табличные функции, которые позволяют осуществлять чтение и запись данных HDFS из ClickHouse. Данная статья описывает особенности и способы использования этих инструментов на примере ADQM и ADH (Arenadata Hadoop), который предоставляет сервис HDFS.

Для интеграции с HDFS запросы из ADQM направляются к хосту кластера ADH с NameNode в состоянии Active (в приведенных ниже примерах замените `<namenode_ip>` в URI файлов HDFS на IP-адрес соответствующего хоста ADH). При подключении к NameNode в статусе Standby ADQM может вернуть сообщение об ошибке следующего содержания:

```
Exception: Unable to connect to HDFS: Operation category READ is not supported in state standby.
```

Табличный движок HDFS

Табличный движок HDFS позволяет управлять данными, хранящимися в файлах HDFS, из ADQM. Этот тип таблиц поддерживает многопоточное чтение и запись данных HDFS, но имеет ряд ограничений (репликация, индексы, операции `ALTER` и `SELECT…SAMPLE` не поддерживаются).

Базовый синтаксис запроса для создания таблицы HDFS:

```
CREATE TABLE <table_name> (<column_name> <column_type>, ...) ENGINE = HDFS(<URI>, <format>);
```

Параметры табличного движка HDFS:

- `<URI>` — полный URI файла в HDFS. Часть URI, содержащая путь к файлу в HDFS (имя каталога, имя файла), может содержать символы подстановки, чтобы получить данные из нескольких файлов — в этом случае таблица может быть использована только для чтения.
- `<format>` — формат, в котором ADQM будет принимать при выполнении запросов `SELECT` и отдавать при выполнении запросов `INSERT` табличные данные (см. форматы в столбцах ***Input*** и ***Output*** таблицы Formats for Input and Output Data соответственно).

При экспорте данных из таблицы ADQM в файловую систему HDFS используйте параметры `hdfs_create_new_file_on_insert` и `hdfs_truncate_on_insert`, которые позволяют определить, как будут сохраняться данные в файлы HDFS — будут создаваться новые файлы или перезаписываться существующие. По умолчанию обе опции выставлены в `0`, и при вставке данных в таблицу ADQM возвращается следующее сообщение:

```
Exception: File with path <file_path> already exists. If you want to overwrite it, enable setting hdfs_truncate_on_insert,
if you want to create new file on each insert, enable setting hdfs_create_new_file_on_insert.
```

где `<file_path>` — путь к файлу в HDFS, на основе которого была создана таблица HDFS в ADQM.

В файле конфигурации ADQM можно указать дополнительные настройки для таблиц HDFS: глобальные (указываются в секции настроек `<hdfs>`) и настройки на уровне пользователя (указываются в секции `hdfs_<user_name>`, где `<user_name>` — имя пользователя). Список параметров, которые можно настроить для движка HDFS, приведен в соответствующем разделе статьи HDFS документации ClickHouse.

Пример

Создание таблицы HDFS

1. В HDFS в каталоге *adqm_data_sources* создайте файл *test_1.txt* с тестовыми данными для последующей вставки в таблицу ADQM:

   ```
   "one",1
   "two",2
   "three",3
   ```

   URI файла для обращения из ADQM — *hdfs://<namenode_ip>:8020/adqm_data_sources/test_1.txt*, где `<namenode_ip>` — IP-адрес хоста ADH с NameNode в состоянии Active.

2. Запустите консольный клиент clickhouse-client и создайте таблицу на базе движка HDFS со структурой, соответствующей данным, которые будут импортироваться из HDFS:

   ```
   CREATE TABLE hdfs_table (name String, value UInt32) ENGINE=HDFS('hdfs://<namenode_ip>:8020/adqm_data_sources/test_1.txt',
   'CSV');
   ```

3. Запросите данные из таблицы:

   ```
   SELECT * FROM hdfs_table;
   ```

   ```
   ┌─name──┬─value─┐
   │ one   │     1 │
   │ two   │     2 │
   │ three │     3 │
   └───────┴───────┘
   ```

Вставка данных в таблицу HDFS

1. Включите настройку `hdfs_create_new_file_on_insert`, чтобы при каждой вставке данных создавался новый файл в HDFS (например, если таблица была создана на основе файла *file_name.txt*, новым файлам названия будут назначаться по шаблону — *file_name.1.txt*, *file_name.2.txt* и так далее):

   ```
   SET hdfs_create_new_file_on_insert = 1;
   ```

2. Выполните вставку данных, повторив два запроса `INSERT INTO`:

   ```
   INSERT INTO hdfs_table VALUES('four', 4);
   ```

   ```
   INSERT INTO hdfs_table VALUES('five', 5);
   ```

3. Проверьте содержимое каталога *adqm_data_sources* в HDFS:

   ```
   $ hdfs dfs -ls /adqm_data_sources
   ```

   В каталоге создано два новых файла *test_1.1.txt* и *test_1.2.txt*:

   ```
   Found 3 items
   -rwxrwxrwx   3 clickhouse hadoop          9 2023-10-25 13:40 /adqm_data_sources/test_1.1.txt
   -rwxrwxrwx   3 clickhouse hadoop          9 2023-10-25 13:40 /adqm_data_sources/test_1.2.txt
   -rwxrwxrwx   3 clickhouse hadoop         23 2023-10-25 13:25 /adqm_data_sources/test_1.txt
   ```

   Посмотрите содержимое этих файлов:

   ```
   $ hdfs dfs -cat /adqm_data_sources/test_1.1.txt
   ```

   ```
   "four",4
   ```

   ```
   $ hdfs dfs -cat /adqm_data_sources/test_1.2.txt
   ```

   ```
   "five",5
   ```

   При последующем чтении таблицы можно убедиться, что в выборку будут включены данные из всех файлов:

   ```
   SELECT * FROM hdfs_table;
   ```

   ```
   ┌─name──┬─value─┐
   │ one   │     1 │
   │ two   │     2 │
   │ three │     3 │
   └───────┴───────┘
   ┌─name─┬─value─┐
   │ five │     5 │
   └──────┴───────┘
   ┌─name─┬─value─┐
   │ four │     4 │
   └──────┴───────┘
   ```

4. Активируйте опцию `hdfs_truncate_on_insert` и выполните еще одну вставку данных в таблицу — в этом случае последний файл с данными будет перезаписан:

   ```
   SET hdfs_truncate_on_insert = 1;
   ```

   ```
   INSERT INTO hdfs_table VALUES('six', 6);
   ```

5. Проверьте содержимое каталога *adqm_data_sources* в HDFS:

   ```
   $ hdfs dfs -ls /adqm_data_sources
   ```

   Новый файл не был добавлен, но файл *test_1.2.txt* обновлен:

   ```
   Found 3 items
   -rwxrwxrwx   3 clickhouse hadoop          9 2023-10-25 13:40 /adqm_data_sources/test_1.1.txt
   -rwxrwxrwx   3 clickhouse hadoop          9 2023-10-25 13:45 /adqm_data_sources/test_1.2.txt
   -rwxrwxrwx   3 clickhouse hadoop         23 2023-10-25 13:25 /adqm_data_sources/test_1.txt
   ```

   ```
   $ hdfs dfs -cat /adqm_data_sources/test_1.2.txt
   ```

   ```
   "six",6
   ```

   Если настройка `hdfs_truncate_on_insert` включена, новые данные будут заменять текущее содержимое существующего файла в любом случае, независимо от значения `hdfs_create_new_file_on_insert`.

Табличные функции

ADQM также предоставляет две табличные функции для интеграции с HDFS:

- **hdfs** — создает таблицу для чтения/записи данных в HDFS.
  Синтаксис функции:

  ```
  hdfs(<URI>, <format>, <structure>)
  ```

  где:

  - `<URI>` — URI файла в HDFS (в режиме использования функции для чтения можно указать путь к нескольким файлам с помощью символов подстановки);
  - `<format>` — формат, в котором функция будет принимать или отдавать данные (подробная информация о форматах для импорта и экспорта данных ADQM/ClickHouse представлена в статье Formats for Input and Output Data документации ClickHouse);
  - `<structure>` — структура таблицы в формате `column_name1 data_type1, column_name2 data_type2, …`.

- **hdfsCluster** — позволяет обрабатывать файлы HDFS параллельно из нескольких узлов в указанном кластере. На узле-инициаторе функция создает соединение со всеми узлами в кластере, заменяет символы `*` в пути к файлу HDFS и динамически отправляет каждый файл. На рабочем узле функция запрашивает у инициатора следующую задачу и обрабатывает ее. Это повторяется до тех пор, пока все задачи не будут завершены.
  Синтаксис функции:

  ```
  hdfs(<cluster_name>, <URI>, <format>, <structure>)
  ```

  где `<cluster_name>` — имя кластера, используемое для создания набора адресов и параметров подключения к удаленным и локальным серверам; остальные параметры аналогичны параметрам функции `hdfs`.

Пример

Чтение данных из файла HDFS

Выполните следующий запрос, чтобы получить в ADQM данные из файла HDFS *test_1.txt* с помощью функции `hdfs`:

```
SELECT * FROM hdfs('hdfs://<namenode_ip>:8020/adqm_data_sources/test_1.txt', 'CSV', 'column1 String, column2 UInt32');
```

```
┌─column1─┬─column2─┐
│ one     │       1 │
│ two     │       2 │
│ three   │       3 │
└─────────┴─────────┘
```

Запись данных в HDFS

Если настройка `hdfs_truncate_on_insert` включена, при вставке данных через функцию `hdfs` существующие данные в файле, на которую ссылается функция, будут заменяться новыми данными. В этом случае необходимо, чтобы у пользователя `clickhouse` в HDFS были права на изменение этого файла.

Например, выполните следующий запрос (значение `hdfs_truncate_on_insert` было установлено в `1` в приведенном выше примере для табличного движка HDFS):

```
INSERT INTO FUNCTION hdfs('hdfs://<namenode_ip>:8020/adqm_data_sources/test_1.txt', 'CSV', 'column1 String, column2 UInt32')
VALUES ('hdfs_func_test_string_1', 100);
```

В HDFS проверьте, что содержимое файла *test_1.txt* обновилось:

```
$ hdfs dfs -cat /adqm_data_sources/test_1.txt
```

```
"hdfs_func_test_string_1",100
```

Если значение параметра `hdfs_truncate_on_insert` установлено в `0` и при этом активирована настройка `hdfs_create_new_file_on_insert`, при вставке данных в HDFS через функцию `hdfs`, которая ссылается на файл *file_name.txt*, будет создан (или перезаписан, если уже существует) файл с именем *file_name.1.txt*.

Например, выполните запрос на запись данных в файл *test_1.txt*:

```
SET hdfs_truncate_on_insert = 0, hdfs_create_new_file_on_insert = 1;
```

```
INSERT INTO FUNCTION hdfs('hdfs://<namenode_ip>:8020/adqm_data_sources/test_1.txt', 'CSV', 'column1 String, column2 UInt32')
VALUES ('hdfs_func_test_string_2', 200);
```

В результате новые данные запишутся в файл *test_1.1.txt*:

```
$ hdfs dfs -cat /adqm_data_sources/test_1.1.txt
```

```
"hdfs_func_test_string_2",200
```

Символы подстановки в пути к файлу HDFS

Если таблица ADQM будет принимать данные из нескольких файлов HDFS и использоваться только для чтения, путь к файлам HDFS в параметре `URI` табличного движка HDFS или табличной функции `hdfs`/`hdfsCluster` можно указать с использованием символов подстановки:

- `*` — заменяет любое количество любых символов кроме разделителя пути `/`, включая отсутствие символов;
- `?` — заменяет любой одиночный символ;
- `{first_string,second_string,third_one}` — заменяет любую из строк `'first_string'`, `'second_string'`, `'third_one'` (в этом шаблоне также можно использовать числа — например, `{1,3,5}`);
- `{n..m}` — заменяет любое число в диапазоне `[n, m]`.

Символы подстановки могут содержаться в нескольких компонентах пути (например, в названии каталога и в названии файла). Обрабатываются только файлы, существующие в файловой системе, пути и названия которых полностью соответствуют шаблону. Список файлов определяется во время выполнения операции `SELECT` (не `CREATE`).

Пример

1. Создайте несколько файлов со следующими URI в HDFS:

   - *hdfs://<namenode_ip>:8020/first_dir/file_1.txt*
   - *hdfs://<namenode_ip>:8020/first_dir/file_2.txt*
   - *hdfs://<namenode_ip>:8020/second_dir/file_3.txt*
   - *hdfs://<namenode_ip>:8020/second_dir/file_4.txt*

   Данные во всех файлах должны согласовываться с форматом и схемой, которые будут указываться в запросах при подключении из ADQM.

2. С помощью функции `hdfs` импортируйте в ADQM данные из всех приведенных выше файлов одним из способов:

   ```
   SELECT * FROM hdfs('hdfs://<namenode_ip>:8020/{first,second}_dir/file_{1..4}.txt', 'CSV', 'column1 String, column2 UInt32');
   ```

   ```
   SELECT * FROM hdfs('hdfs://<namenode_ip>:8020/{first,second}_dir/file_?.txt', 'CSV', 'column1 String, column2 UInt32');
   ```

   Следующий запрос считывает данные из всех файлов в каталогах *first_dir* и *second_dir*:

   ```
   SELECT * FROM hdfs('hdfs://<namenode_ip>:8020/{first,second}_dir/*', 'CSV', 'column1 String, column2 UInt32');
   ```

3. Если список файлов содержит числовые интервалы с ведущими нулями в названиях файлов, можно использовать общий шаблон `{0n..0m}` или отдельный шаблон `{n..m}` (либо знак `?`) для каждой цифры в названии файла. Например, следующие запросы создают таблицу с данными из файлов *file_000.txt*, *file_001.txt*, …, *file_999.txt*:

   ```
   CREATE TABLE hdfs_table_1 (column1 String, column2 UInt32)
   ENGINE=HDFS('hdfs://<namenode_ip>:8020/<dir_name>/file_{001..999}.txt', 'CSV');
   ```

   ```
   CREATE TABLE hdfs_table_2 (column1 String, column2 UInt32)
   ENGINE=HDFS('hdfs://<namenode_ip>:8020/<dir_name>/file_{0..9}{0..9}{0..9}.txt', 'CSV');
   ```

   ```
   CREATE TABLE hdfs_table_3 (column1 String, column2 UInt32)
   ENGINE=HDFS('hdfs://<namenode_ip>:8020/<dir_name>/file_???.txt', 'CSV');
   ```

Режим высокой доступности (HA) HDFS

Если в кластере ADH для HDFS включен режим высокой доступности (High Availability, HA), выполните настройку:

1. Скопируйте файл /etc/hadoop/conf/hdfs-site.xml с хоста ADH на хост ADQM в папку */etc/clickhouse-server/*.

2. Добавьте следующую секцию в файл конфигурации ClickHouse */etc/clickhouse-server/config.xml*:

   ```
   <hdfs>
       <libhdfs3_conf>/etc/clickhouse-server/hdfs-site.xml</libhdfs3_conf>
   </hdfs>
   ```

3. Перезапустите сервер ClickHouse:

   ```
   $ sudo systemctl restart clickhouse-server
   ```

После этого при подключении к HDFS через табличный движок HDFS или табличные функции можно использовать значение тега `dfs.nameservices` в файле *hdfs-site.xml* вместо `<namenode_ip>:8020` в URI файла HDFS. Например, если значение тега `dfs.nameservices` — `adhcluster`, то запрос на создание таблицы hdfs_table и вызов функции hdfs, приведенные в примерах выше, можно изменить следующим образом:

```
CREATE TABLE hdfs_table (name String, value UInt32) ENGINE=HDFS('hdfs://adhcluster/adqm_data_sources/test_1.txt', 'CSV');
```

```
SELECT * FROM hdfs('hdfs://adhcluster/adqm_data_sources/test_1.txt', 'CSV', 'column1 String, column2 UInt32');
```

Пример доступа ADQM к HDFS керберизованного кластера ADH

Проверка аутентификации и создание тестовых данных в HDFS

1. Выполните настройку MIT Kerberos KDC и проведите керберизацию кластера ADH, следуя инструкциям в разделе MIT Kerberos документации ADH. В качестве имени realm используйте `ADREALM.IO`.

2. Убедитесь, что аутентификация в HDFS через Kerberos выполняется успешно. Для этого на хосте ADH создайте тикет на предоставление тикетов (Ticket-Granting Ticket, TGT) для принципала `hdfs/<adh_node_hostname>@ADREALM.IO`:

   ```
   $ sudo -u hdfs kinit -k -t /etc/security/keytabs/hdfs.service.keytab hdfs/<adh_node_hostname>@ADREALM.IO
   ```

   Затем выполните команду, например, для просмотра содержимого корневого каталога HDFS:

   ```
   $ sudo -u hdfs hdfs dfs -ls /
   ```

   При успешной аутентификации через Kerberos на экран будет выведено текущее содержимое корневого каталога.

3. Создайте в HDFS новый каталог */adqm_test_kerb* и добавьте в него файл *test_2.txt* с несколькими строками для тестирования подключения ADQM через Kerberos, например:

   ```
   "ten",10
   "twenty",20
   "thirty",30
   ```

4. Чтобы разрешить добавление данных на стороне ADQM, назначьте владельцем нового каталога пользователя `adqm_user`, от имени которого ADQM будет обращаться к HDFS (принципал этого пользователя будет создан на следующем шаге):

   ```
   $ sudo -u hdfs hdfs dfs -chown -R adqm_user /adqm_test_kerb
   ```

> **ВАЖНО**
>
> Из-за ограничений библиотеки libhdfs3 для доступа ADQM к каталогам HDFS керберизованного кластера ADH нужно настроить следующие параметры сервиса HDFS:
>
> - для параметра `hadoop.rpc.protection` в файле *core-site.xml* не должно быть установлено значение `privacy`;
> - значение параметра `dfs.encrypt.data.transfer` в файле *hdfs-site.xml* должно быть `false`.
>
> Изменить значения этих параметров можно на странице конфигурации сервиса HDFS в интерфейсе ADCM, после чего необходимо запустить для сервиса действие ***Restart***.

Создание принципала пользователя для подключения ADQM к HDFS

1. На хосте, где развернут MIT Kerberos KDC, создайте принципал для пользователя `adqm_user`, который будет использоваться для подключения ADQM к HDFS:

   ```
   $ sudo kadmin.local -q "add_principal -randkey adqm_user@ADREALM.IO"
   ```

2. Создайте для этого принципала keytab-файл:

   ```
   $ sudo kadmin.local -q "xst -kt /tmp/adqm_user.keytab adqm_user@ADREALM.IO"
   ```

3. Скопируйте keytab-файл */tmp/adqm_user.keytab* на хосты ADQM (например, в директорию */tmp*).
   На хосте ADQM назначьте `clickhouse` владельцем keytab-файла и ограничьте доступ к файлу в целях безопасности:

   ```
   $ sudo chown clickhouse:clickhouse /tmp/adqm_user.keytab
   $ sudo chmod 0600 /tmp/adqm_user.keytab
   ```

Настройка ADQM

1. В каталоге */etc/clickhouse-server/config.d* создайте файл (например, *hdfs_kerberos.xml*) с настройками:

   ```
   <?xml version="1.0"?>
   <clickhouse>
       <hdfs>
           <hadoop_security_authentication>kerberos</hadoop_security_authentication>
           <hadoop_kerberos_keytab>/tmp/adqm_user.keytab</hadoop_kerberos_keytab>
           <hadoop_kerberos_principal>adqm_user@ADREALM.IO</hadoop_kerberos_principal>
           ...
       </hdfs>
   </clickhouse>
   ```

   Выполните действие ***Reconfig*** с опцией ***Restart*** для сервиса ADQMDB.

2. Установите на хосте кластера ADQM пакеты:

   ```
   $ sudo yum install krb5-libs krb5-workstation
   ```

3. Отредактируйте файл */etc/krb5.conf* следующим образом (замените `<kdc_host>` на имя хоста с установленным MIT Kerberos KDC):

   ```
   # Configuration snippets may be placed in this directory as well
   includedir /etc/krb5.conf.d/

   [logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

   [libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
    forwardable = true
    rdns = false
    pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
    default_realm = ADREALM.IO

   [realms]
    ADREALM.IO = {
     admin_server = <kdc_host>
     kdc = <kdc_host>
    }

   [domain_realm]
    .adrealm.io = ADREALM.IO
    adrealm.io = ADREALM.IO
   ```

Создание таблицы в ADQM

1. На хосте ADQM в консольном клиенте clickhouse-client создайте таблицу на базе движка HDFS:

   ```
   CREATE TABLE hdfs_table_kerb (name String, value UInt32)
   ENGINE=HDFS('hdfs://<namenode_ip>:8020/adqm_test_kerb/test_2.txt', 'CSV');
   ```

2. Считайте данные из таблицы:

   ```
   SELECT * FROM hdfs_table_kerb;
   ```

   ```
      ┌─name───┬─value─┐
   1. │ ten    │    10 │
   2. │ twenty │    20 │
   3. │ thirty │    30 │
      └────────┴───────┘
   ```

3. Вставьте в таблицу новую строку:

   ```
   SET hdfs_create_new_file_on_insert = 1;
   ```

   ```
   INSERT INTO hdfs_table_kerb VALUES('forty', 40);
   ```

4. Проверьте содержимое каталога *adqm_test_kerb* в HDFS:

   ```
   $ sudo -u hdfs hdfs dfs -ls /adqm_test_kerb
   ```

   ```
   -rwxrwxrwx   3 adqm_user hadoop         12 2025-02-24 09:04 /adqm_test_kerb/test_2.1.txt
   -rw-r--r--   3 adqm_user hadoop         36 2025-02-24 08:53 /adqm_test_kerb/test_2.txt
   ```

   Убедитесь, что в новый файл *test_2.1.txt* добавлена строка, вставленная в таблицу HDFS на стороне ADQM:

   ```
   $ sudo -u hdfs hdfs dfs -cat /adqm_test_kerb/test_2.1.txt
   ```

   ```
   "forty",40
   ```

Чтение данных Hive из ADQM

Елена Дворядкина

Содержание

Назад к содержанию

Создание таблицы Hive

Режим высокой доступности (HA) HDFS

Пример

Для получения данных из таблиц Hive через запросы `SELECT` ClickHouse предоставляет табличный движок Hive. Форматы принимаемых данных в настоящее время поддерживаются следующим образом:

- Text — поддерживаются только простые скалярные типы столбцов кроме `binary`;
- ORC — поддерживаются простые скалярные типы столбцов кроме `char` и сложный тип `array`;
- Parquet — поддерживаются все простые скалярные типы столбцов и сложный тип `array`.

В данной статье описываются особенности работы с табличным движком Hive на примере ADQM и ADH (Arenadata Hadoop), который предоставляет сервис Hive.

Создание таблицы Hive

Базовый синтаксис запроса для создания таблицы Hive в ADQM:

```
CREATE TABLE <table_name> (<column_name> <column_type> [ALIAS <expr>], ...)
ENGINE = Hive('thrift://<host>:<port>', '<hive_database>', '<hive_table>')
PARTITION BY <expr>;
```

Параметры табличного движка Hive:

- `thrift://<host>:<port>` — адрес хоста ADH, где установлен компонент Hive Metastore;
- `<hive_database>` — имя удаленной базы данных Hive;
- `<hive_table>` — имя удаленной таблицы Hive.

Структура таблицы ADQM может отличаться от структуры удаленной таблицы Hive, к которой осуществляется подключение:

- Имена столбцов должны быть такими же, как в исходной таблице Hive, но количество и порядок столбцов не обязательно должны совпадать. Также в таблице ADQM с помощью выражения `ALIAS` можно указать столбцы, значения которых вычисляются на основе значений других столбцов.
- Типы столбцов в таблице ADQM должны быть такими же, как в исходной таблице Hive.
- Ключ партиционирования `PARTITION BY` таблицы ADQM должен соответствовать ключу `PARTITIONED BY` исходной таблицы Hive, а столбцы, указанные в выражении `PARTITION BY`, должны быть определены в структуре таблицы.

Режим высокой доступности (HA) HDFS

Если в кластере ADH активирован режим высокой доступности (High Availability, HA) для HDFS, перед подключением ADQM к таблицам Hive необходимо выполнить следующую настройку:

1. Скопируйте файл /etc/hadoop/conf/hdfs-site.xml с хоста ADH на хост ADQM в папку */etc/clickhouse-server/*.
2. Добавьте следующую настройку в файл конфигурации ClickHouse */etc/clickhouse-server/config.xml*:

```
<hdfs>
    <libhdfs3_conf>/etc/clickhouse-server/hdfs-site.xml</libhdfs3_conf>
</hdfs>
```

3. Перезапустите сервер ClickHouse:

```
$ sudo systemctl restart clickhouse-server
```

Внутреннее имя кластера HDFS, указанное как internal nameservice, должно быть в нижнем регистре. Иначе при попытке получить данные из удаленной таблицы Hive ADQM может вернуть сообщение об ошибке следующего содержания:

```
Unable to connect to HDFS: InvalidParameter: Cannot parse URI: hdfs://<internal_nameservice_in_lower_case>, missing port or
invalid HA configuration
Caused by: HdfsConfigNotFound: Config key: dfs.ha.namenodes.<internal_nameservice_in_lower_case> not found. (NETWORK_ERROR)
```

где `<internal_nameservice_in_lower_case>` — internal nameservice, указанный в параметре `dfs.internal.nameservices` файла конфигурации *hdfs-site.xml*, но в нижнем регистре.

Пример

В примере ниже показано, как в ADQM получить данные из удаленной таблицы Hive кластера ADH (режим высокой доступности для HDFS не включен).

1. На хосте ADH, где установлен компонент Hive HiveServer2, создайте таблицу Hive `test_table` в базе данных `default`:

```
CREATE TABLE test_table(`id` int, `value` int, `name` string) PARTITIONED BY (`date` date);
```

2. Вставьте в таблицу тестовые данные:

```
INSERT INTO test_table VALUES (0, 1, 'one', '2023-11-23');
```

3. На хосте ADQM запустите консольный клиент clickhouse-client и создайте таблицу на базе движка Hive (замените `<adh_host>` на IP-адрес хоста ADH с компонентом Hive Metastore):

```
CREATE TABLE hive_table(id Int8, value Int16, name String, date Date)
ENGINE = Hive('thrift://<adh_host>:9083', 'default', 'test_table')
PARTITION BY date;
```

4. Запросите данные из таблицы:

```
SELECT * FROM hive_table;
```

```
┌─id─┬─value─┬─name─┬─date────────┐
│  0 │     1 │ one  │ 2023-11-23  │
└────┴───────┴──────┴─────────────┘
```

Интеграция ADQM и Kafka

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
- Табличный движок Kafka
  - Создание таблицы Kafka
  - Расширенная настройка
  - Поддержка Kerberos
  - Виртуальные столбцы
- Общие операции
  - Остановка и перезапуск потребления сообщений
  - Изменение целевой таблицы
  - Изменение таблицы Kafka
- Пример
  - Создание топика Kafka в ADS
  - Создание таблиц в ADQM для интеграции с Kafka
  - Чтение данных из Kafka
  - Вставка данных из ADQM в Kafka
  - Изменение целевой таблицы
- Пример подключения ADQM к Kafka с установленным Kerberos SASL
  - Настройка ADS
  - Создание топика в ADS
  - Создание принципала для подключения ADQM к Kafka
  - Настройка ADQM
  - Создание таблиц в ADQM
  - Проверка подключения ADQM к Kafka

ClickHouse поддерживает возможность интеграции с Apache Kafka через специальный табличный движок — Kafka. Этот движок позволяет:

- подписываться на топики Kafka, чтобы организовать потоковую загрузку данных из Kafka в ClickHouse (наиболее частый сценарий использования интеграции) или публиковать данные из ClickHouse в Kafka;
- создавать отказоустойчивые хранилища;
- обрабатывать потоки по мере их появления.

В данной статье описываются особенности работы с табличным движком Kafka на примере ADQM и ADS (Arenadata Streaming), который предоставляет сервис Kafka.

Обзор

Движок Kafka позволяет ClickHouse читать данные напрямую из топика Kafka и отслеживать поток сообщений, но каждое сообщение из таблицы Kafka можно получить только один раз. Как только данные запрашиваются из таблицы Kafka, они считаются уже полученными (consumed) из очереди и перед возвратом результата увеличивается смещение для потребителя (offset). Данные невозможно повторно считать без сброса этих смещений, поэтому не следует делать выборку данных из таблицы Kafka напрямую (через запрос `SELECT`) — вместо этого рекомендуется использовать материализованное представление. Такой подход предполагает создание следующих таблиц на стороне ClickHouse:

- Таблица на базе движка Kafka — потребитель Kafka, который подписывается на топик и читает поток данных.
- Целевая таблица с нужной структурой для постоянного хранения данных, полученных из Kafka. Обычно это таблица на базе движка семейства MergeTree.
- Материализованное представление, которое в фоновом режиме собирает данные из таблицы Kafka (получает новые данные блоками при срабатывании триггера на вставку данных в таблицу Kafka), преобразует их в необходимый формат и помещает в заранее созданную целевую таблицу.
  Иногда необходимо применять различные преобразования к данным, поступающим из Kafka — например, для хранения необработанных и агрегированных данных. В этом случае к одной таблице Kafka можно привязать несколько материализованных представлений, чтобы сохранять данные с разным уровнем детализации в несколько таблиц.

Чтобы записывать данные из ClickHouse в Kafka, можно вставлять их напрямую в таблицу на базе движка Kafka.

Табличный движок Kafka

Создание таблицы Kafka

Базовый синтаксис запроса для создания таблицы Kafka в ADQM:

```
CREATE TABLE <table_name> (<column_name> <column_type> [ALIAS <expr>], ...)
ENGINE = Kafka()
SETTINGS
    kafka_broker_list = '<host_name>:9092,...',
    kafka_topic_list = '<topic_name>,...',
    kafka_group_name = '<consumer_group_name>',
    kafka_format = '<data_format>'[,]
    [kafka_schema = '',]
    [kafka_num_consumers = <N>,]
    [kafka_max_block_size = 0,]
    [kafka_skip_broken_messages = <N>,]
    [kafka_commit_every_batch = 0,]
    [kafka_client_id = '',]
    [kafka_poll_timeout_ms = 0,]
    [kafka_poll_max_batch_size = 0,]
    [kafka_flush_interval_ms = 0,]
    [kafka_thread_per_consumer = 0,]
    [kafka_handle_error_mode = 'default',]
    [kafka_commit_on_select = false,]
    [kafka_max_rows_per_message = 1];
```

Табличный движок Kafka имеет следующие обязательные параметры. Эти параметры можно указывать в круглых скобках при создании таблицы (см. пример).

kafka_broker_list	Список брокеров (разделяются запятыми)
kafka_topic_list	Список топиков Kafka (разделяются запятыми)
kafka_group_name	Группа потребителей Kafka. Смещения для чтения отслеживаются для каждой группы отдельно. Если нужно, чтобы сообщения в кластере не дублировались, используйте одно имя группы потребителей для всех таблиц Kafka
kafka_format	Формат сообщений Kafka. Для определения этого параметра используются такие же названия форматов, как для выражения `FORMAT` в запросах `INSERT` и `SELECT` — подробнее в статье документации ClickHouse Formats for Input and Output Data

Опциональные параметры

kafka_schema	Параметр, который необходимо использовать, если формат требует определения схемы. Например, CapnProto требует путь к файлу со схемой и название корневого объекта `schema.capnp:Message`
kafka_num_consumers	Количество потребителей (consumer) на таблицу. По умолчанию: `1`. Укажите больше потребителей, если пропускная способность одного потребителя недостаточна. Общее число потребителей не должно превышать количество партиций в топике (так как на одну партицию может быть назначено не более одного потребителя) и количество ядер на сервере, где развернут ClickHouse
kafka_max_block_size	Максимальный размер блока (в сообщениях) для poll. Значение по умолчанию равно значению параметра max_insert_block_size
kafka_skip_broken_messages	Максимальное количество допустимых некорректных (несовместимых со схемой) сообщений в блоке. Если `kafka_skip_broken_messages = N`, то движок пропускает `N` сообщений Kafka, которые не получилось обработать (одно сообщение соответствует одной строке данных). Значение по умолчанию: `0`
kafka_commit_every_batch	Включает или отключает режим коммита (commit) каждого принятого и обработанного пакета по отдельности вместо одного коммита после записи целого блока. Значение по умолчанию: `0`
kafka_client_id	Идентификатор клиента. По умолчанию не указывается
kafka_poll_timeout_ms	Тайм-аут для одного опроса (poll) Kafka. Значение по умолчанию равно значению параметра stream_poll_timeout_ms
kafka_poll_max_batch_size	Максимальное количество сообщений, опрашиваемых в одном вызове опроса (poll) Kafka. Значение по умолчанию равно значению параметра max_block_size
kafka_flush_interval_ms	Тайм-аут для сброса данных из Kafka. Значение по умолчанию равно значению параметра stream_flush_interval_ms
kafka_thread_per_consumer	Включает или отключает предоставление отдельного потока каждому потребителю. При включенном режиме каждый потребитель сбрасывает данные независимо и параллельно, при отключенном — строки данных от нескольких потребителей собираются в один блок. Значение по умолчанию: `0`
kafka_handle_error_mode	Способ обработки ошибок для движка Kafka. Возможные значения: - `default` — выдается исключение, если сообщение не удалось обработать (parse); - `stream` — текст ошибки и сообщение, которое не удалось обработать, будут сохранены в виртуальных столбцах `_error` и `_raw_message`
kafka_commit_on_select	Включает или отключает коммит (commit) сообщений при выполнении запроса `SELECT`. Значение по умолчанию: `false`
kafka_max_rows_per_message	Максимальное количество строк в одном сообщении Kafka для форматов row-based. Значение по умолчанию: `1`

РЕКОМЕНДАЦИЯ

В реальных системах рекомендуется передавать параметры через именованные коллекции (named collections).

Расширенная настройка

В файле конфигурации ClickHouse можно указать дополнительные настройки для таблиц Kafka:

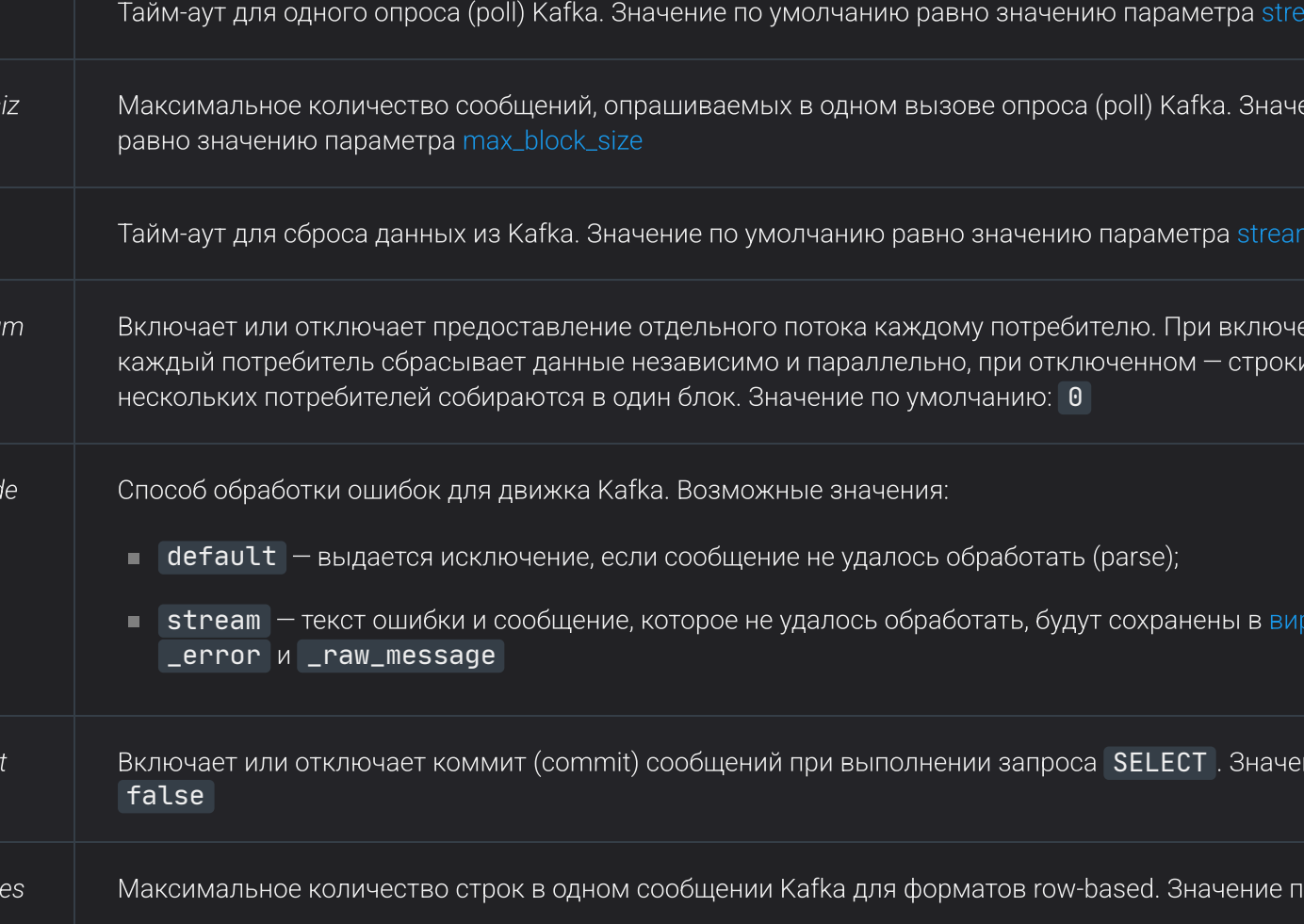
- глобальные настройки в секции `<kafka>`;
- настройки на уровне топика в секции `<kafka_topic>` внутри тега `<kafka>` (имя топика указывается через параметр `<name>`).

Например, ведение журнала отладки (debug) для библиотеки Kafka (librdkafka) активируется с помощью следующего параметра в файле *config.xml*:

```
<kafka>
   <debug>all</debug>
</kafka>
```

Список всех возможных параметров конфигурации можно посмотреть в статье Configuration properties документации библиотеки librdkafka. В ADQM используйте нижнее подчеркивание вместо точки в названии параметра — например, `<auto_offset_reset>latest</auto_offset_reset>` вместо `auto.offset.reset = latest`.

Настройки для таблиц Kafka в ADQM можно указать через интерфейс ADCM. Для этого на странице конфигурации сервиса ADQMDB активируйте опцию ***Kafka engine*** и в поле ***Kafka Properties*** перечислите нужные параметры движка Kafka (соответствующие теги со значениями, без тега `<kafka>`).



Kafka Properties: <debug>all</debug> <auto_offset_reset>latest</auto_offset_reset>

Дополнительные настройки движка Kafka

Нажмите ***Save*** и выполните действие ***Reconfig*** с опцией ***Restart*** для сервиса ADQMDB — после того, как выполнение действия завершится, секция `<kafka>` с указанными параметрами будет добавлена в файл *config.xml*.

Поддержка Kerberos

Для работы с Kafka с поддержкой Kerberos назначьте параметру `security_protocol` значение `sasl_plaintext`. Этого достаточно, если тикет выдачи тикетов Kerberos (ticket-granting ticket) получен и кешируется средствами операционной системы. ClickHouse может поддерживать учетные данные Kerberos с помощью файла keytab. Используйте также параметры `sasl_kerberos_service_name`, `sasl_kerberos_keytab` и `sasl_kerberos_principal` для настройки аутентификации через Kerberos — см. пример ниже.

Виртуальные столбцы

Иногда может быть полезно иметь метаданные сообщений Kafka при их загрузке в ClickHouse. Например, может потребоваться знать координаты потребленного сообщения в топике или сколько сообщений получено из определенного топика или партиции. Для этой цели движок Kafka предоставляет следующие виртуальные столбцы.

Название столбца	Тип данных	Описание
_topic	LowCardinality(String)	Топик Kafka
_key	String	Ключ сообщения
_offset	UInt64	Смещение (offset) сообщения
_timestamp	Nullable(DateTime)	Временная метка сообщения
_timestamp_ms	Nullable(DateTime64(3))	Временная метка сообщения в миллисекундах
_partition	UInt64	Партиция топика Kafka
_headers.name	Array(String)	Массив ключей заголовков сообщений
_headers.value	Array(String)	Массив значений заголовков сообщений
_error	String	Текст исключения, которое было выдано, если сообщение не удалось обработать. Столбец заполняется, если параметр `kafka_handle_error_mode` установлен в `stream` и произошла ошибка во время синтаксического анализа сообщения (если сообщение было проанализировано успешно, столбец остается пустым)
_raw_message	String	Сообщение, которое не удалось успешно обработать. Столбец заполняется, если параметр `kafka_handle_error_mode` установлен в `stream` и произошла ошибка во время синтаксического анализа сообщения (если сообщение было проанализировано успешно, столбец остается пустым)

Чтобы получать значения виртуальных столбцов, необходимо добавить соответствующие столбцы в структуру целевой таблицы и указать их в выражении `SELECT` материализованного представления — см. пример ниже. Виртуальные столбцы в таблице Kafka создавать не нужно, так как они доступны автоматически.

Общие операции

Остановка и перезапуск потребления сообщений

Чтобы остановить получение сообщений из топика, отсоедините таблицу Kafka:

```
DETACH TABLE <kafka_engine_table>;
```

Отсоединение таблицы Kafka не влияет на смещение для группы потребителей. Чтобы возобновить чтение данных и продолжить с предыдущего смещения, снова присоедините таблицу:

```
ATTACH TABLE <kafka_engine_table>;
```

Изменение целевой таблицы

Если необходимо изменить целевую таблицу, рекомендуется следующая последовательность действий:

1. Отключите таблицу Kafka (`DETACH TABLE`).
2. Измените целевую таблицу (`ALTER TABLE`).
3. Удалите материализованное представление, чтобы избежать несоответствия между измененной целевой таблицей и данными из представления (`DROP VIEW`).
4. Снова присоедините таблицу Kafka (`ATTACH TABLE`).
5. Пересоздайте материализованное представление (`CREATE MATERIALIZED VIEW`).

Изменение таблицы Kafka

Чтобы изменить настройки таблицы Kafka, удалите ее и заново создайте с новыми настройками. При этом материализованное представление изменять не нужно — потребление сообщений возобновится после пересоздания таблицы Kafka.

Пример

Создание топика Kafka в ADS

Подключитесь к любому хосту кластера ADS, на котором установлен сервис Kafka. Перейдите в корневую директорию со скриптами для выполнения команд в среде Kafka (файлы с расширением *.sh*) и следуйте инструкциям ниже.

1. Создайте топик `topic-kafka-to-adqm` с помощью следующей команды (замените `<kafka_host>` на имя хоста ADS, где создается топик):

   ```
   $ bin/kafka-topics.sh --create --topic topic-kafka-to-adqm --bootstrap-server <kafka_host>:9092
   ```

   Результат:

   ```
   Created topic topic-kafka-to-adqm.
   ```

   ПРИМЕЧАНИЕ

   Дополнительную информацию о создании и чтении топиков в Kafka можно получить в статье документации ADS Начало работы с Kafka.

2. Запишите тестовые сообщения в топик следующим образом.
   Выполните команду, которая запускает режим записи сообщений:

   ```
   $ bin/kafka-console-producer.sh --topic topic-kafka-to-adqm --bootstrap-server <kafka_host>:9092
   ```

   На следующей строке после ввода команды введите сообщения, каждое на новой строке (используйте `Enter` для перехода на новую строку):

   ```
   >1,"one"
   >2,"two"
   >3,"three"
   ```

   Чтобы выйти из режима записи сообщений, перейдите на следующую строку после записи последнего сообщения и нажмите `Ctrl+C`.

Создание таблиц в ADQM для интеграции с Kafka

На хосте ADQM запустите консольный клиент clickhouse-client и выполните следующие шаги.

1. Создайте таблицу на базе движка Kafka:

   ```
   CREATE TABLE kafka_table (id Int32, name String) ENGINE = Kafka ('<kafka_host>:9092', 'topic-kafka-to-adqm', 'clickhouse',
   'CSV');
   ```

2. Создайте целевую таблицу, в которую будут сохраняться данные:

   ```
   CREATE TABLE kafka_data (id Int32, name String) ENGINE = MergeTree() ORDER BY id;
   ```

3. Создайте материализованное представление, которое будет получать данные из таблицы Kafka и помещать их в целевую таблицу MergeTree:

   ```
   CREATE MATERIALIZED VIEW kafka_data_mv TO kafka_data AS SELECT id, name FROM kafka_table;
   ```

   В момент создания материализованное представление подключается к таблице Kafka, начинает чтение данных и вставку строк в целевую таблицу. Этот процесс продолжаться постоянно — все последующие вставки сообщений в топик Kafka будут считываться.

Чтение данных из Kafka

1. Убедитесь, что данные Kafka вставлены в целевую таблицу:

   ```
   SELECT * FROM kafka_data;
   ```

   ```
   ┌─id─┬─name──┐
   │  1 │ one   │
   │  2 │ two   │
   │  3 │ three │
   └────┴───────┘
   ```

2. Вставьте новые сообщения в топик Kafka (на хосте ADS):

   ```
   $ bin/kafka-console-producer.sh --topic topic-kafka-to-adqm --bootstrap-server <kafka_host>:9092
   ```

   ```
   >4,"four"
   >5,"five"
   ```

3. Проверьте, что новые данные попали в таблицу ADQM:

   ```
   SELECT * FROM kafka_data;
   ```

   ```
   ┌─id─┬─name──┐
   │  1 │ one   │
   │  2 │ two   │
   │  3 │ three │
   │  4 │ four  │
   │  5 │ five  │
   └────┴───────┘
   ```

Вставка данных из ADQM в Kafka

1. На хосте ADQM вставьте данные в таблицу на базе движка Kafka:

   ```
   INSERT INTO kafka_table VALUES (6, 'six');
   ```

2. На хосте ADS проверьте, что соответствующее сообщение записалось в топик Kafka:

   ```
   $ bin/kafka-console-consumer.sh --topic topic-kafka-to-adqm --from-beginning --bootstrap-server <kafka_host>:9092
   ```

   ```
   1,"one"
   2,"two"
   3,"three"
   4,"four"
   5,"five"
   6,"six"
   ```

3. Убедитесь также, что новые данные записались в целевую таблицу ADQM с данными Kafka:

   ```
   SELECT * FROM kafka_data;
   ```

   ```
   ┌─id─┬─name──┐
   │  1 │ one   │
   │  2 │ two   │
   │  3 │ three │
   │  4 │ four  │
   │  5 │ five  │
   │  6 │ six   │
   └────┴───────┘
   ```

Изменение целевой таблицы

Чтобы целевая таблица включала метаинформацию о сохраняемых в нее сообщениях Kafka (например, название топика и смещение сообщения), сделайте следующие изменения.

1. Отключите таблицу Kafka:

   ```
   DETACH TABLE kafka_table;
   ```

2. Добавьте в целевую таблицу столбцы `topic` и `offset`:

   ```
   ALTER TABLE kafka_data
       ADD COLUMN topic String,
       ADD COLUMN offset UInt64;
   ```

3. Удалите материализованное представление:

   ```
   DROP VIEW kafka_data_mv;
   ```

4. Снова подключите таблицу Kafka:

   ```
   ATTACH TABLE kafka_table;
   ```

5. Пересоздайте материализованное представление:

   ```
   CREATE MATERIALIZED VIEW kafka_data_mv TO kafka_data AS SELECT id, name, _topic as topic, _offset as offset FROM kafka_table;
   ```

Теперь для новых сообщений, поступающих в целевую таблицу ADQM, в столбцах `topic` и `offset` будет показываться название топика Kafka и смещение сообщения в партиции. Чтобы это проверить:

1. Вставьте новое сообщение в топик Kafka на хосте ADS:

   ```
   $ bin/kafka-console-producer.sh --topic topic-kafka-to-adqm --bootstrap-server <kafka_host>:9092
   ```

   ```
   >7,"seven"
   ```

2. Запросите данные из таблицы `kafka_data` в ADQM:

   ```
   SELECT * FROM kafka_data;
   ```

   ```
   ┌─id─┬─name──┐
   │  1 │ one   │
   │  2 │ two   │
   │  3 │ three │
   │  4 │ four  │
   │  5 │ five  │
   │  6 │ six   │
   └────┴───────┘
   ┌─id─┬─name──┬─topic───────────────┬─offset─┐
   │  7 │ seven │ topic-kafka-to-adqm │      6 │
   └────┴───────┴─────────────────────┴────────┘
   ```

Пример подключения ADQM к Kafka с установленным Kerberos SASL

Настройка ADS

1. Выполните настройку MIT Kerberos KDC на отдельном хосте и проведите керберизацию кластера ADS, следуя инструкции MIT Kerberos. В качестве имени realm используйте `ADS-KAFKA.LOCAL`.
2. Убедитесь в корректной настройке аутентификации для сервиса Kafka, выполнив шаги из раздела Проверка установленного Kerberos SASL.
3. Создайте принципал для аутентификации в Kafka (например, `ads_user`). Для этого выполните следующую команду на хосте, где развернут MIT Kerberos KDC:

   ```
   $ sudo kadmin.local -q "add_principal -pw PASSWORD ads_user@ADS-KAFKA.LOCAL"
   ```

4. На стороне ADS настройте для пользователя JAAS-файл */tmp/client.jaas* и файл конфигурации */tmp/client.properties* — см. описание в соответствующих разделах статьи Использование MIT Kerberos в Kafka.

Создание топика в ADS

Создайте тестовый топик, к которому будет подключаться ADQM для чтения/записи сообщений. Для этого на хосте ADS последовательно выполните описанные ниже шаги:

1. Создайте тикет на предоставление тикетов (Ticket-Granting Ticket, TGT) для принципала `ads_user`:

   ```
   $ kinit -p ads_user@ADS-KAFKA.LOCAL
   ```

   Введите пароль, указанный при создании принципала — в данном примере `PASSWORD`.

2. Экспортируйте созданный файл *client.jaas* как параметр JVM для данного пользователя с помощью переменной среды `KAFKA_OPTS`:

   ```
   $ export KAFKA_OPTS="-Djava.security.auth.login.config=/tmp/client.jaas"
   ```

3. Создайте топик (например, `topic-adqm-to-kafka-kerberos`), указав путь к созданному файлу *client.properties*:

   ```
   $ /usr/lib/kafka/bin/kafka-topics.sh \
     --create --topic topic-adqm-to-kafka-kerberos  \
     --bootstrap-server <kafka_host1>:9092,<kafka_host2>:9092,<kafka_host3>:9092  \
     --command-config /tmp/client.properties
   ```

   Замените `<kafka_host1>`, `<kafka_host2>`, `<kafka_host3>` на имена хостов ADS, где создается топик.

4. Запишите несколько сообщений в топик, также указав путь к файлу *client.properties*:

   ```
   $ /usr/lib/kafka/bin/kafka-console-producer.sh  \
     --topic topic-adqm-to-kafka-kerberos   \
     --bootstrap-server <kafka_host1>:9092,<kafka_host2>:9092,<kafka_host3>:9092  \
     --producer.config /tmp/client.properties
   ```

   ```
   >1,"one"
   >2,"two"
   >3,"three"
   ```

   Для выхода из режима записи сообщений перейдите на следующую строку после записи последнего сообщения и нажмите `Ctrl+C`.

Создание принципала для подключения ADQM к Kafka

1. На хосте, где развернут MIT Kerberos KDC, создайте новый принципал `adqm_user`, который будет использовать ADQM для подключения к Kafka:

   ```
   $ sudo kadmin.local -q "add_principal -randkey adqm_user@ADS-KAFKA.LOCAL"
   ```

2. Создайте для этого принципала keytab-файл:

   ```
   $ sudo kadmin.local -q "xst -kt /tmp/adqm_user.keytab adqm_user@ADS-KAFKA.LOCAL"
   ```

3. Скопируйте keytab-файл */tmp/adqm_user.keytab* на хосты ADQM (например, в директорию */tmp*).
   На хосте ADQM назначьте `clickhouse` владельцем keytab-файла и ограничьте доступ к файлу в целях безопасности:

   ```
   $ sudo chown clickhouse:clickhouse /tmp/adqm_user.keytab
   $ sudo chmod 0600 /tmp/adqm_user.keytab
   ```

Настройка ADQM

1. На странице конфигурации сервиса ADQMDB включите ***Kafka engine*** и введите в поле ***Kafka Properties*** следующие параметры:

   ```
   <security_protocol>SASL_PLAINTEXT</security_protocol>
   <sasl_mechanism>GSSAPI</sasl_mechanism>
   <sasl_kerberos_service_name>kafka</sasl_kerberos_service_name>
   <sasl_kerberos_keytab>/tmp/adqm_user.keytab</sasl_kerberos_keytab>
   <sasl_kerberos_principal>adqm_user@ADS-KAFKA.LOCAL</sasl_kerberos_principal>
   ```

   Нажмите ***Save*** и выполните действие ***Reconfig*** с опцией ***Restart*** для сервиса ADQMDB.

2. Установите необходимые пакеты на хосте кластера ADQM:

   ```
   $ sudo yum install krb5-libs krb5-workstation
   ```

3. Также на хосте ADQM отредактируйте файл */etc/krb5.conf*, изменив его содержимое следующим образом (замените `<kdc_host>` на имя хоста с установленным MIT Kerberos KDC):

   ```
   # Configuration snippets may be placed in this directory as well
   includedir /etc/krb5.conf.d/

   [logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

   [libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
    forwardable = true
    rdns = false
    pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
    default_realm = ADS-KAFKA.LOCAL

   [realms]
    ADS-KAFKA.LOCAL = {
    admin_server = <kdc_host>
    kdc = <kdc_host>
    }

   [domain_realm]
    .ads-kafka.local = ADS-KAFKA.LOCAL
    ads-kafka.local = ADS-KAFKA.LOCAL
   ```

Создание таблиц в ADQM

На хосте ADQM в консольном клиенте clickhouse-client создайте таблицы для интеграции с Kafka:

1. Таблица на базе движка Kafka:

   ```
   CREATE TABLE adqm_to_ads_kerberos_kafka (id Int32, name String)
   ENGINE = Kafka ('<kafka_host1>:9092,<kafka_host2>:9092,<kafka_host3>:9092', 'topic-adqm-to-kafka-kerberos', 'clickhouse_group',
   'CSV');
   ```

2. Целевая таблица, в которую будут сохраняться данные:

   ```
   CREATE TABLE adqm_to_ads_kerberos_data (id Int32, name String) ENGINE = MergeTree() ORDER BY id;
   ```

3. Материализованное представление, которое будет получать данные из таблицы Kafka и помещать их в целевую таблицу MergeTree:

   ```
   CREATE MATERIALIZED VIEW adqm_to_ads_kerberos_data_mv TO adqm_to_ads_kerberos_data
   AS SELECT id, name FROM adqm_to_ads_kerberos_kafka;
   ```

Проверка подключения ADQM к Kafka

1. Проверьте, что данные из топика Kafka вставлены в целевую таблицу ADQM:

   ```
   SELECT * FROM adqm_to_ads_kerberos_data;
   ```

   ```
      ┌─id─┬─name──┐
   1. │  1 │ one   │
   2. │  2 │ two   │
   3. │  3 │ three │
      └────┴───────┘
   ```

2. Вставьте строку данных в таблицу ADQM на движке Kafka:

   ```
   INSERT INTO adqm_to_ads_kerberos_kafka VALUES (4, 'four');
   ```

3. Убедитесь, что соответствующее сообщение успешно получено на стороне кластера ADS:

   ```
   $ /usr/lib/kafka/bin/kafka-console-consumer.sh \
     --topic topic-adqm-to-kafka-kerberos \
     --from-beginning  \
     --bootstrap-server <kafka_host1>:9092,<kafka_host2>:9092,<kafka_host3>:9092  \
     --consumer.config /tmp/client.properties
   ```

   ```
   1,"one"
   2,"two"
   3,"three"
   4,"four"
   ```

Интеграция ADQM и ADPG

Елена Дворядкина

Содержание

Назад к содержанию

- Обзор
- Основные параметры интеграции
- Табличная функция postgresql
  - Пример
- Табличный движок PostgreSQL
  - Создание таблицы
  - Пример
- Движок баз данных PostgreSQL
  - Создание базы данных
  - Пример
- Табличный движок MaterializedPostgreSQL
  - Требования
  - Создание таблицы
  - Виртуальные столбцы
  - Пример
- Движок баз данных MaterializedPostgreSQL
  - Требования
  - Создание базы данных
  - Настройки
  - Динамическое добавление и удаление таблиц из репликации
  - Пример
- Источник данных для словаря типа POSTGRESQL
  - Пример

Обзор

ClickHouse предоставляет несколько способов интеграции с PostgreSQL:

- табличная функция postgresql | табличный движок PostgreSQL | движок баз данных PostgreSQL — обеспечивают возможность отправлять запросы из ClickHouse к таблицам/базам данных на сервере PostgreSQL для обмена данными в реальном времени. Поддерживаются операции чтения и записи данных PostgreSQL. При выполнении запроса `SELECT` или `INSERT` ClickHouse подключается к таблице/базе данных PostgreSQL и получает актуальные данные PostgreSQL или экспортирует новые данные. Иными словами, данные хранятся в PostgreSQL, а из ClickHouse можно к ним обращаться.
- табличный движок MaterializedPostgreSQL | движок баз данных MaterializedPostgreSQL — позволяют использовать ClickHouse в качестве реплики PostgreSQL. При создании таблицы/базы данных MaterializedPostgreSQL в ClickHouse создается исходный дамп данных таблицы или таблиц базы данных PostgreSQL и запускается процесс репликации — в фоновом режиме к таблице/базе данных MaterializedPostgreSQL применяются изменения данных, выполненные на стороне PostgreSQL. Таким образом, при использовании движков MaterializedPostgreSQL в отличие от движков PostgreSQL данные хранятся и в PostgreSQL, и в ClickHouse.
- тип источника данных POSTGRESQL для словарей — позволяет использовать таблицы PostgreSQL в качестве источника данных для словарей в ClickHouse.

В этой статье описываются особенности и способы использования этих инструментов на примере ADQM и ADPG (Arenadata Postgres — реляционная система управления базами данных, основанная на PostgreSQL). Приводятся запросы для вызова табличной функции и создания таблиц/баз данных на основе перечисленных интеграционных движков, а также примеры запросов для чтения/записи данных ADPG из ADQM.

Подготовка ADPG для тестовых примеров

На хосте ADPG запустите терминальный клиент `psql` от имени пользователя по умолчанию `postgres` с помощью команд:

```
$ sudo su - postgres
```

```
$ psql
```

Теперь в `psql` можно вводить команды для работы с базами данных и таблицами PostgreSQL:

```
psql (14.1)
Type "help" for help.

postgres=#
```

ПРИМЕЧАНИЕ

Обзор по работе с PostgreSQL-таблицами в `psql` можно посмотреть в статье документации ADPG Пример работы с таблицами через psql.

Создание пользователя PostgreSQL

Создайте пользователя (с правами суперпользователя для демонстрационных целей), от имени которого будет выполняться подключение к ADPG из ADQM:

```
CREATE ROLE adqm_user SUPERUSER LOGIN PASSWORD '<adqm_user_secret_password>';
```

Создание базы данных и таблицы ADPG для примеров с табличной функцией и движками PostgreSQL

1. Создайте в ADPG базу данных `adpg_db1`:

   ```
   CREATE DATABASE adpg_db1;
   ```

   Используйте `psql`-команду `\c` для подключения к этой базе данных:

   ```
   \c adpg_db1
   ```

   Результат выполнения команды:

   ```
   You are now connected to database "adpg_db1" as user "postgres".
   adpg_db1=#
   ```

2. Создайте таблицу:

   ```
   CREATE TABLE table1 (id INTEGER PRIMARY KEY, name VARCHAR(10));
   ```

3. Добавьте в таблицу тестовые данные:

   ```
   INSERT INTO table1 (id, name) VALUES (1, 'one'), (2, 'two');
   ```

Создание базы данных с таблицами ADPG для примеров с движками MaterializedPostgreSQL

1. Создайте в ADPG новую базу данных `adpg_db2` и подключитесь к ней:

   ```
   CREATE DATABASE adpg_db2;
   ```

   ```
   \c adpg_db2
   ```

2. Создайте в базе данных `adpg_db2` три таблицы с одинаковой структурой:

   ```
   CREATE TABLE table1 (id INTEGER PRIMARY KEY, name VARCHAR(10));
   ```

   ```
   CREATE TABLE table2 (id INTEGER PRIMARY KEY, name VARCHAR(10));
   ```

   ```
   CREATE TABLE table3 (id INTEGER PRIMARY KEY, name VARCHAR(10));
   ```

3. Добавьте в эти таблицы тестовые данные:

   ```
   INSERT INTO table1 (id, name) VALUES (1, 'one'), (2, 'two');
   ```

   ```
   INSERT INTO table2 (id, name) VALUES (3, 'three'), (4, 'four');
   ```

   ```
   INSERT INTO table3 (id, name) VALUES (5, 'five'), (6, 'six');
   ```

Создание таблицы ADPG, источника данных для словаря

Создайте в базе данных `adpg_db2` таблицу `roles_dict`, которая будет источником данных для словаря:

```
CREATE TABLE roles_dict (id INTEGER PRIMARY KEY, role VARCHAR(10));
```

```
INSERT INTO roles_dict (id, role) VALUES (10, 'admin'), (20, 'owner'), (30, 'author'), (40, 'reviewer'), (50, 'editor'), (60,
'view only');
```

Настройка аутентификации для соединения с ADPG

Чтобы разрешить подключение к базам данных `adpg_db1` и `adpg_db2` с хоста ADQM от имени пользователя `adqm_user` и настроить аутентификацию по паролю, необходимо добавить следующие записи в файл *pg_hba.conf* (для ADPG это можно сделать через интерфейс ADCM — подробнее в статье Аутентификация по паролю):

```
host adpg_db1 adqm_user <adqm_host_address> password
host adpg_db2 adqm_user <adqm_host_address> password
```

где `<adqm_host_address>` — адрес хоста ADQM (например, если IPv4-адрес хоста ADQM — `10.92.40.25`, замените `<adqm_host_address>` на `10.92.40.25/32`). Подробное описание поля `address` и других полей записей файла *pg_hba.conf* приведены в статье The pg_hba.conf file.

Основные параметры интеграции

При вызове табличной функции `postgresql` и создании таблиц/баз данных на основе движков для интеграции с PostgreSQL (базовый синтаксис запросов приводится в соответствующих разделах ниже) необходимо указывать следующие параметры.

<postgres_host>	Адрес сервера PostgreSQL (IP или имя хоста)
<postgres_database>	Имя базы данных PostgreSQL
<postgres_table>	Имя таблицы в базе данных PostgreSQL (указывается для табличной функции `postgresql`, а также для табличных движков PostgreSQL и MaterializedPostgreSQL)
<postgres_user>	Имя пользователя PostgreSQL, от имени которого ClickHouse подключается к PostgreSQL
<user_password>	Пароль пользователя PostgreSQL
<schema>	Схема PostgreSQL (опционально может быть указана для табличной функции `postgresql`, табличного движка PostgreSQL, движка баз данных PostgreSQL)

РЕКОМЕНДАЦИЯ

В реальных системах рекомендуется передавать параметры через именованные коллекции (named collections).

Табличная функция postgresql

Табличная функция `postgresql` позволяет направлять запросы `SELECT` и `INSERT` к таблицам в базах данных PostgreSQL.

Синтаксис функции:

```
postgresql('<postgres_host>:5432', '<postgres_database>', '<postgres_table>', '<postgres_user>', '<user_password>'[, '<schema>'])
```

Описание параметров приведено выше.

Пример

Чтение данных из ADPG

Выполните следующий запрос, чтобы получить в ADQM данные из таблицы ADPG `adpg_db1.table1` с помощью функции `postgresql`:

```
SELECT id, name FROM postgresql('<adpg_host>:5432', 'adpg_db1', 'table1', 'adqm_user', '<adqm_user_secret_password>');
```

где `<adpg_host>` — адрес хоста ADPG (IP или имя хоста).

Результат:

```
┌─id─┬─name─┐
│  1 │ one  │
│  2 │ two  │
└────┴──────┘
```

Запись данных в ADPG

Выполните следующий запрос, чтобы добавить новую запись в таблицу ADPG:

```
INSERT INTO TABLE FUNCTION postgresql('<adpg_host>:5432', 'adpg_db1', 'table1', 'adqm_user', '<adqm_user_secret_password>') (id,
name)
VALUES (3, 'three');
```

Убедитесь, что данные записались в таблицу назначения. Для этого на хосте ADPG выполните следующую команду, предварительно подключившись к базе данных `adpg_db1`:

```
SELECT * FROM table1;
```

```
 id | name
----+-------
  1 | one
  2 | two
  3 | three
(3 rows)
```

Табличный движок PostgreSQL

Табличный движок PostgreSQL позволяет направлять запросы `SELECT` и `INSERT` к таблицам на удаленном сервере PostgreSQL.

Создание таблицы

Базовый синтаксис запроса для создания таблицы PostgreSQL (описание параметров приведено выше):

```
CREATE TABLE <table_name> (<column_name> <column_type> [DEFAULT|MATERIALIZED|ALIAS <expr>] [TTL <expr>], ...)
ENGINE = PostgreSQL('<postgres_host>:5432', '<postgres_database>', '<postgres_table>', '<postgres_user>', '<user_password>'[,
'<schema>']);
```

Структура таблицы ADQM может отличаться от структуры таблицы PostgreSQL, к которой осуществляется подключение:

- Имена столбцов должны быть такими же, как в исходной таблице PostgreSQL, но количество и порядок столбцов не обязательно должны совпадать.
- Типы столбцов в таблице ADQM могут отличаться от типов в исходной таблице PostgreSQL — ADQM пытается приводить значения к типам данных ClickHouse.
- Настройка external_table_functions_use_nulls определяет, как обрабатываются Nullable-столбцы. Значение по умолчанию: `1`. Если значение `0`, столбцы не преобразуются в Nullable — вместо NULL в столбце выставляются значения по умолчанию для соответствующего типа данных. Это также применимо для значений NULL в массивах.

Пример

В ADQM создайте таблицу на базе движка PostgreSQL и запросите из нее данные:

```
CREATE TABLE psql_table (id Int32, name String)
ENGINE = PostgreSQL('<adpg_host>:5432', 'adpg_db1', 'table1', 'adqm_user', '<adqm_user_secret_password>');
```

```
SELECT * FROM psql_table;
```

```
┌─id─┬─name──┐
│  1 │ one   │
│  2 │ two   │
│  3 │ three │
└────┴───────┘
```

Вставьте данные в таблицу ADQM:

```
INSERT INTO psql_table VALUES (4, 'four');
```

На хосте ADPG запросите данные из таблицы `adpg_db1.table1`, чтобы проверить, что в нее успешно были импортированы данные из ADQM:

```
SELECT * FROM table1;
```

```
 id | name
----+-------
  1 | one
  2 | two
  3 | three
  4 | four
(4 rows)
```

Движок баз данных PostgreSQL

Движок баз данных PostgreSQL позволяет подключаться к базе данных на сервере PostgreSQL и выполнять операции чтения и записи (запросы `SELECT` и `INSERT`) для обмена данными между ClickHouse и PostgreSQL, а также в реальном времени получать от сервера PostgreSQL информацию о таблицах базы данных и их структуре с помощью запросов `SHOW TABLES` и `DESCRIBE TABLE`.

Создание базы данных

Базовый синтаксис запроса для создания базы данных PostgreSQL в ADQM:

```
CREATE DATABASE <database_name>
ENGINE = PostgreSQL('<postgres_host>:5432', '<postgres_database>', '<postgres_user>', '<user_password>'[, `<schema>`,
`use_table_cache`]);
```

Описание основных параметров приведено выше. Необязательный параметр `use_table_cache` определяет, кешируются ли структуры таблиц. Если значение параметра `1`, структура таблицы кешируется и изменения в ней не отслеживаются. Чтобы обновить структуру таблицы, нужно последовательно выполнить команды `DETACH TABLE` и `ATTACH TABLE`.

Пример

Создайте в ADQM базу данных `psql_database` на основе движка PostgreSQL:

```
CREATE DATABASE psql_database
ENGINE = PostgreSQL('<adpg_host>:5432', 'adpg_db1', 'adqm_user', '<adqm_user_secret_password>', 1);
```

Проверьте, что новая база данных добавлена в список баз данных ADQM:

```
SHOW DATABASES;
```

```
┌─name───────────────┐
│ INFORMATION_SCHEMA │
│ default            │
│ information_schema │
│ psql_database      │
│ system             │
└────────────────────┘
```

Посмотрите, какие таблицы есть в базе данных `psql_database`:

```
SHOW TABLES FROM psql_database;
```

```
┌─name───┐
│ table1 │
└────────┘
```

Запросите данные из таблицы:

```
SELECT * FROM psql_database.table1;
```

```
┌─id─┬─name──┐
│  1 │ one   │
│  2 │ two   │
│  3 │ three │
│  4 │ four  │
└────┴───────┘
```

Добавьте новые данные в таблицу на стороне ADPG:

```
INSERT INTO table1 (id, name) VALUES (5, 'five');
```

Проверьте, что эти данные доступны через таблицу ADQM:

```
SELECT * FROM psql_database.table1;
```

```
┌─id─┬─name──┐
│  1 │ one   │
│  2 │ two   │
│  3 │ three │
│  4 │ four  │
│  5 │ five  │
└────┴───────┘
```

На хосте ADPG измените структуру таблицы `adpg_db1.table1`, добавив в нее новый столбец:

```
ALTER TABLE table1 ADD COLUMN value SMALLINT;
```

Структура таблицы на стороне ADQM не поменяется, так как при создании базы данных параметр `use_table_cache` был установлен в `1` (значит, структура таблицы в ADQM была закеширована):

```
DESCRIBE TABLE psql_database.table1;
```

```
┌─name─┬─type─────────────┬─
│ id   │ Int32            │
│ name │ Nullable(String) │
└──────┴──────────────────┴─
```

Если таблицу в ADQM отключить, а затем снова подключить, структура таблицы обновится:

```
DETACH TABLE psql_database.table1;
```

```
ATTACH TABLE psql_database.table1;
```

```
DESCRIBE TABLE psql_database.table1;
```

```
┌─name──┬─type─────────────┬─
│ id    │ Int32            │
│ name  │ Nullable(String) │
│ value │ Nullable(Int16)  │
└───────┴──────────────────┴─
```

Табличный движок MaterializedPostgreSQL

Табличный движок MaterializedPostgreSQL создает таблицу ClickHouse с исходным дампом данных таблицы PostgreSQL и запускает процесс репликации, то есть в фоновом режиме к таблице ADQM применяются изменения, произведенные в таблице PostgreSQL.

Если требуется реплицировать в ADQM более одной таблицы PostgreSQL, вместо табличного движка рекомендуется использовать движок баз данных MaterializedPostgreSQL и с помощью настройки `materialized_postgresql_tables_list` указать таблицы базы данных PostgreSQL, которые нужно реплицировать. Это позволит оптимизировать нагрузку на процессор, уменьшить количество подключений и количество слотов репликации в базе данных PostgreSQL.

ВНИМАНИЕ

Репликация TOAST-значений не поддерживается. Для типа данных используется значение по умолчанию.

Требования

Для использования табличного движка MaterializedPostgreSQL необходимо выполнение следующих условий:

- В конфигурационном файле PostgreSQL настройка `wal_level` должна иметь значение `logical`, значение параметра `max_replication_slots` должно быть не меньше `2` (подробнее об этих параметрах и о том, как их настроить в ADPG, написано в статье Логическая репликация).
- Таблица, созданная на основе движка MaterializedPostgreSQL, должна иметь такой же первичный ключ, как репликационный идентификатор (обычно это первичный ключ) таблицы PostgreSQL.
- Таблица MaterializedPostgreSQL может быть создана только в базе данных Atomic.

Создание таблицы

Базовый синтаксис запроса для создания таблицы на основе движка MaterializedPostgreSQL:

```
CREATE TABLE <table_name> (<key_column_name> <key_column_type>, <column_name> <column_type>, ...)
ENGINE = MaterializedPostgreSQL('<postgres_host>:5432', '<postgres_database>', '<postgres_table>', '<postgres_user>',
'<user_password>')
PRIMARY KEY <key_column_name>;
```

Описание параметров приведено выше.

Виртуальные столбцы

В таблицу MaterializedPostgreSQL автоматически добавляются следующие виртуальные столбцы, которые доступны в запросах `SELECT`:

- `_version` (UInt64) — счетчик транзакций (значение LSN в WAL);
- `_sign` (Int8) — метка удаления. Возможные значения:
  - `1` — строка не удалена;
  - `-1` — строка удалена.

Пример

Настройка ADPG

В ADPG настройте следующие параметры конфигурации, необходимые для логической репликации, с помощью которой реализуется репликация через табличный движок MaterializedPostgreSQL:

- значение параметра `wal_level` — `logical`;
- значение параметра `max_replication_slots` — не меньше `2`.

Создание таблицы MaterializedPostgreSQL

В ADQM создайте таблицу MaterializedPostgreSQL:

```
CREATE TABLE psql_table_replica (id Int32, name String)
ENGINE = MaterializedPostgreSQL('<adpg_host>:5432', 'adpg_db2', 'table1', 'adqm_user', '<adqm_user_secret_password>')
PRIMARY KEY id;
```

В созданную таблицу ADQM автоматически вставятся данные из исходной таблицы ADPG:

```
SELECT * FROM psql_table_replica;
```

```
┌─id─┬─name─┐
│  1 │ one  │
│  2 │ two  │
└────┴──────┘
```

Тестирование репликации

В ADPG добавьте новые строки в таблицу `adpg_db2.table1`:

```
INSERT INTO table1 (id, name) VALUES (11, 'eleven'), (22, 'twenty two');
```

Убедитесь, что новые строки реплицированы в таблицу ADQM:

```
SELECT * FROM psql_table_replica;
```

```
┌─id─┬─name─┐
│  1 │ one  │
│  2 │ two  │
└────┴──────┘
┌─id─┬─name───────┐
│ 11 │ eleven     │
│ 22 │ twenty two │
└────┴────────────┘
```

Движок баз данных MaterializedPostgreSQL

Движок баз данных MaterializedPostgreSQL позволяет создать базу данных ADQM с таблицами из базы данных PostgreSQL. При создании базы данных на основе движка MaterializedPostgreSQL создается снепшот базы данных PostgreSQL и в базу данных MaterializedPostgreSQL загружаются необходимые таблицы (это может быть любой набор таблиц из любого набора схем базы данных PostgreSQL). База данных MaterializedPostgreSQL получает исходный дамп данных таблиц PostgreSQL и запускает процесс репликации — читает WAL и выполняет запросы DML, применяя изменения данных, которые были произведены в базе данных PostgreSQL.

Репликация реализована с помощью протокола логической репликации PostgreSQL, который не поддерживает репликацию запросов DDL, но позволяет определять изменения, нарушающие репликацию (например, изменение типа столбца, добавление/удаление столбцов). При таких изменениях соответствующие таблицы перестают получать обновления. В этом случае следует выполнить запросы `ATTACH TABLE` / `DETACH TABLE` для перезагрузки таблицы. Если DDL-запрос не нарушает репликацию (например, переименование столбца), таблица будет продолжать получать обновления (вставка производится по позиции, а не по имени столбца).

ВАЖНО

- В настоящее время движок баз данных MaterializedPostgreSQL находится в статусе экспериментального. Чтобы его использовать, необходимо перед созданием базы данных включить опцию: `SET allow_experimental_database_materialized_postgresql = 1`.
- Репликация TOAST-значений не поддерживается. Для типа данных будет использоваться значение по умолчанию.

Требования

Чтобы создавать и использовать базы данных на движке MaterializedPostgreSQL необходимо выполнение следующих условий:

- В конфигурационном файле PostgreSQL настройка `wal_level` должна иметь значение `logical`, значение параметра `max_replication_slots` должно быть не меньше `2` (подробнее об этих параметрах и о том, как их настроить в ADPG, написано в статье Логическая репликация).
- Каждая реплицируемая таблица должна иметь репликационный идентификатор — первичный ключ (по умолчанию) или индекс, определенный как REPLICA IDENTITY.

Создание базы данных

Синтаксис запроса для создания базы данных на основе движка MaterializedPostgreSQL в общем виде:

```
CREATE DATABASE <database_name>
ENGINE = MaterializedPostgreSQL('<postgres_host>:5432', '<postgres_database>', '<postgres_user>', '<user_password>')
[SETTINGS ...];
```

Описание параметров приведено выше.

Настройки

Через выражение `SETTINGS` для базы данных MaterializedPostgreSQL можно указать следующие настройки.

materialized_postgresql_tables_list	Список таблиц базы данных PostgreSQL (разделяются запятыми), которые будут реплицироваться через движок баз данных MaterializedPostgreSQL. - В списке можно указать таблицы из разных схем базы данных. Для этого нужно указать имя каждой таблицы вместе с именем схемы (через точку), а также установить настройку `materialized_postgresql_tables_list_with_schema = 1`. В этом случае при обращении к таблице базы данных MaterializedPostgreSQL в ADQM также необходимо указывать имя схемы вместе с именем таблицы. Например: `` SELECT * FROM <database_name>.`<schema_name>.<table_name>`; `` - Если в базу данных MaterializedPostgreSQL реплицируются таблицы из одной схемы (схемы, назначенной через параметр `materialized_postgresql_schema`, или схемы по умолчанию), имена таблиц в списке `materialized_postgresql_tables_list` указываются без имени схемы. Обращение к таблице базы данных MaterializedPostgreSQL в этом случае также осуществляется только по имени таблицы. Например: `SELECT * FROM <database_name>.<table_name>;` - Если список таблиц не указан, реплицируются все таблицы из схемы `materialized_postgresql_schema` или схемы по умолчанию.
materialized_postgresql_schema	Схема базы данных PostgreSQL, таблицы которой будут реплицироваться
materialized_postgresql_schema_list	Список схем базы данных PostgreSQL (разделяются запятыми), все таблицы которых будут реплицироваться. Если для базы данных MaterializedPostgreSQL через параметр `materialized_postgresql_schema_list` указано несколько схем PostgreSQL, таблицы которых будут реплицироваться, для обращения к таблице в ADQM нужно указывать имя схемы и имя таблицы через точку. Например: `` SELECT * FROM <database_name>.`<schema_name>.<table_name>`; ``
materialized_postgresql_max_block_size	Максимальное количество строк, собранных в памяти перед вставкой данных в таблицу базы данных PostgreSQL
materialized_postgresql_replication_slot *materialized_postgresql_snapshot*	Идентификатор созданного пользователем слота репликации и идентификатор снепшота, из которого будет выполняться исходный дамп таблиц PostgreSQL. Обе эти настройки должны быть указаны вместе. Использовать их рекомендуется только в случае реальной необходимости — например, если нужно определить постоянный слот репликации, чтобы избежать нарушения репликации данных PostgreSQL в случае сбоя основного сервера (за более подробной информацией и примером можно обратиться к разделу Failover of the logical replication slot документации ClickHouse). Обычно самостоятельно конфигурировать слот репликации не нужно — он создается таблицей

Динамическое добавление и удаление таблиц из репликации

После того, как база данных MaterializedPostgreSQL создана, новые таблицы, добавляемые в базу данных PostgreSQL, не попадают автоматически в репликацию — их необходимо добавить вручную с помощью запроса `ATTACH TABLE`. Этот запрос можно также использовать, чтобы добавить таблицу в список `materialized_postgresql_tables_list`, указанный при создании базы данных.

Удалить из репликации отдельные таблицы можно с помощью запроса `DETACH TABLE`.

Пример

Настройка ADPG

Движок баз данных MaterializedPostgreSQL требует те же настройки логической репликации в ADPG, что и табличный движок MaterializedPostgreSQL. В ADPG убедитесь, что:

- параметр `wal_level` установлен в `logical`;
- значение параметра `max_replication_slots` не меньше `2`.

Создание базы данных MaterializedPostgreSQL

В ADQM включите возможность работы с движком баз данных MaterializedPostgreSQL и создайте базу данных:

```
SET allow_experimental_database_materialized_postgresql=1;
```

```
CREATE DATABASE psql_db_replica
ENGINE = MaterializedPostgreSQL('<adpg_host>:5432', 'adpg_db2', 'adqm_user', '<adqm_user_secret_password>')
SETTINGS materialized_postgresql_tables_list = 'table1,table2';
```

Посмотрите, какие таблицы содержатся в базе данных:

```
SHOW TABLES FROM psql_db_replica;
```

```
┌─name───┐
│ table1 │
│ table2 │
└────────┘
```

Получите данные одной из таблиц (например, `table2`):

```
SELECT * FROM psql_db_replica.table2;
```

```
┌─id─┬─name──┐
│  3 │ three │
│  4 │ four  │
└────┴───────┘
```

Тестирование репликации

В ADPG добавьте новую запись в таблицу `adpg_db2.table2`:

```
INSERT INTO table2 (id, name) VALUES (30, 'thirty');
```

Убедитесь, что эта запись попала в таблицу-реплику в ADQM:

```
SELECT * FROM psql_db_replica.table2;
```

```
┌─id─┬─name───┐
│  3 │ three  │
│  4 │ four   │
│ 30 │ thirty │
└────┴────────┘
```

Добавление новой таблицы в репликацию

Чтобы указать дополнительную таблицу базы данных ADPG для репликации в базу данных ADQM (иными словами, изменить список `materialized_postgresql_tables_list` базы данных MaterializedPostgreSQL), выполните команду `ATTACH TABLE`:

```
ATTACH TABLE psql_db_replica.table3;
```

Теперь таблица `table3` также реплицируется в базу данных `psql_db_replica`:

```
SHOW TABLES FROM psql_db_replica;
```

```
┌─name───┐
│ table1 │
│ table2 │
│ table3 │
└────────┘
```

```
SELECT * FROM psql_db_replica.table3;
```

```
┌─id─┬─name─┐
│  5 │ five │
│  6 │ six  │
└────┴──────┘
```

Источник данных для словаря типа POSTGRESQL

Таблицу PostgreSQL можно назначить источником данных для словаря в ClickHouse/ADQM. Для этого при создании словаря в запросе `CREATE DICTIONARY` укажите тип источника `POSTGRESQL` в выражении `SOURCE` и настройте параметры подключения к таблице PostgreSQL следующим образом:

```
CREATE DICTIONARY <dictionary_name> (...)
...
SOURCE(POSTGRESQL(
    port 5432
    host '<postgres_host>'
    user '<postgres_user>'
    password '<postgres_user_password>'
    replica(host 'postgres_host_1' port 5432 priority 1)
    replica(host 'postgres_host_2' port 5432 priority 2)
    db '<postgres_database>'
    table '<postgres_table>'
    where '<filter_expr>'
    invalidate_query '<query_to_check_dictionary_status>'
    query '<custom_query>'))
...
```

Конфигурация источника данных словаря типа POSTGRESQL включает следующие поля:

- `port` — порт PostgreSQL.
- `host` — хост PostgreSQL.
- `user` — имя пользователя PostgreSQL, от имени которого ClickHouse будет подключаться к PostgreSQL.
- `password` — пароль пользователя PostgreSQL.
- `replica` — настройки подключения к реплике PostgreSQL.
  При определении источника данных для словаря можно сконфигурировать общие настройки подключения к PostgreSQL (`host`, `port`, `user` и `password`) или указать несколько реплик и каждую настроить отдельно — в поле `replica` указать хост и порт для подключения к реплике, а также назначить реплике приоритет, который ClickHouse будет учитывать при попытке соединения с PostgreSQL (чем меньше значение `priority`, тем выше приоритет реплики).
- `db` — имя базы данных PostgreSQL.
- `table` — имя таблицы в базе данных PostgreSQL.
- `where` — условие выборки (необязательный параметр).
- `invalidate_query` — запрос для проверки условия загрузки словаря (необязательный параметр).
- `query` — пользовательский запрос (необязательный параметр).

Источник данных обязательно должен быть объявлен через одно из полей: `table` или `query`. Поля `table` и `where` не могут быть использованы одновременно с `query`.

Пример

В ADQM создайте словарь, который будет считывать данные из таблицы ADPG `adpg_db2.roles_dict`:

```
CREATE DICTIONARY roles_dict_psql (id Int32, role String DEFAULT 'no role assigned')
PRIMARY KEY id
SOURCE(POSTGRESQL(port 5432 host '<adpg_host_name>' user 'adqm_user' password '<adqm_user_secret_password>' db 'adpg_db2' table
'roles_dict'))
LIFETIME(MIN 10 MAX 20)
LAYOUT(FLAT());
```

Создайте таблицу `users` с именами пользователей и идентификаторами назначенных им ролей:

```
CREATE TABLE users (user_id Int32, name String, role_id Int32) ENGINE = MergeTree ORDER BY user_id;
```

```
INSERT INTO users VALUES (1, 'john', 10), (2, 'mary', 30), (3, 'andrew', 40), (4, 'harry', 70), (5, 'ann', 50);
```

```
--user_id---name-----role_id--
│       1 │ john   │      10 │
│       2 │ mary   │      30 │
│       3 │ andrew │      40 │
│       4 │ harry  │      70 │
│       5 │ ann    │      50 │
------------------------------
```

Выполните следующий запрос, чтобы считать имена пользователей из таблицы `users` и найти названия их ролей в словаре `roles_dict_psql` по идентификаторам `role_id`:

```
SELECT  name AS user, dictGet('roles_dict_psql', 'role', toUInt64(role_id)) AS role FROM users;
```

Результат:

```
┌─user───┬─role─────────────┐
│ john   │ admin            │
│ mary   │ author           │
│ andrew │ reviewer         │
│ harry  │ no role assigned │
│ ann    │ editor           │
└────────┴──────────────────┘
```

Интеграция ADQM и S3

Елена Дворядкина

Содержание

Назад к содержанию

- Параметры интеграции
  - Параметры движка S3 и табличных функций
  - Дополнительные настройки
- Табличный движок S3
  - Создание таблицы
  - Пример
- Табличные функции
  - Пример
- Использование именованных коллекций
  - Ключи параметров
  - Пример
- Символы подстановки в пути к файлам S3
  - Примеры
- Партиционирование данных при записи
- Виртуальные столбцы

Для интеграции с хранилищем S3 в ClickHouse поддерживаются табличный движок S3 и специальные табличные функции, которые позволяют осуществлять чтение и запись данных S3 из ClickHouse.

Файлы S3 для тестовых примеров

В качестве хранилища S3 в примерах, приведенных в данной статье, используется бакет *test-bucket* в Yandex Object Storage. В бакет загружены следующие объекты (файлы) для тестирования интеграции ADQM и S3:

- *test_1.txt*:

```
"one",1
"two",2
"three",3
```

- *test_2.txt*:

```
"ten",10
"twenty",20
"thirty",30
```

Чтобы логически структурировать данные и сгруппировать файлы внутри бакета, используется префикс (каталог) *adqm*. Таким образом, обращаться к файлам из ADQM можно по следующим URL-адресам:

- *https://storage.yandexcloud.net/test-bucket/adqm/test_1.txt*
- *https://storage.yandexcloud.net/test-bucket/adqm/test_2.txt*

Параметры интеграции

Параметры движка S3 и табличных функций

В следующей таблице описаны параметры табличного движка и табличных функций для интеграции с S3 (базовый синтаксис запросов приводится в соответствующих разделах ниже).

<path>	URL бакета с путем к файлу. В режиме использования таблицы или функции для чтения можно указать путь к нескольким файлам с помощью символов подстановки
<access_key_id>, *<secret_access_key>*	Идентификатор ключа доступа и секретный ключ для аутентификации запросов к хранилищу S3. Ключ доступа не обязательно указывать с помощью этих параметров — можно указать учетные данные в конфигурационном файле или с помощью настройки use_environment_credentials включить возможность извлекать их из переменных окружения. Если для бакета определен публичный доступ, вместо параметров ключа можно указать ключевое слово `NOSIGN` — в этом случае запросы не будут подписываться
<session_token>	Токен сессии для временного ключа доступа, который можно опционально указать вместе с `<access_key_id>` и `<secret_access_key>`
<format>	Формат файла. Полный список форматов, доступных в ADQM/ClickHouse для входных и выходных данных, можно посмотреть в статье Formats for input and output data документации ClickHouse
<compression>	Метод сжатия. Возможные значения: - `none`; - `gzip` или `gz`; - `brotli` или `br`; - `xz` или `LZMA`; - `zstd` или `zst`. По умолчанию метод сжатия автоматически определяется по расширению файла
<headers>	Заголовки запроса S3 в формате `headers('key1'='value1', 'key2'='value2', …)`

РЕКОМЕНДАЦИЯ

В реальных системах рекомендуется передавать параметры через именованные коллекции (named collections).

Дополнительные настройки

Параметры на уровне запроса/сессии

ADQM также поддерживает следующие дополнительные параметры интеграции с S3, которые можно настроить перед выполнением запроса или указать необходимые значения в конфигурационном файле на уровне пользователя:

```
SELECT name, default FROM system.settings WHERE startsWith(name, 's3_');
```

```
    ┌─name───────────────────────────────────────────────┬─default────┐
 1. │ s3_strict_upload_part_size                         │ 0          │
 2. │ s3_min_upload_part_size                            │ 16777216   │
 3. │ s3_max_upload_part_size                            │ 5368709120 │
 4. │ s3_upload_part_size_multiply_factor                │ 2          │
 5. │ s3_upload_part_size_multiply_parts_count_threshold │ 500        │
 6. │ s3_max_part_number                                 │ 10000      │
 7. │ s3_max_single_operation_copy_size                  │ 33554432   │
 8. │ s3_max_inflight_parts_for_one_file                 │ 20         │
 9. │ s3_max_single_part_upload_size                     │ 33554432   │
10. │ s3_max_single_read_retries                         │ 4          │
11. │ s3_max_unexpected_write_error_retries              │ 4          │
12. │ s3_max_redirects                                   │ 10         │
13. │ s3_max_connections                                 │ 1024       │
14. │ s3_max_get_rps                                     │ 0          │
15. │ s3_max_get_burst                                   │ 0          │
16. │ s3_max_put_rps                                     │ 0          │
17. │ s3_max_put_burst                                   │ 0          │
18. │ s3_list_object_keys_size                           │ 1000       │
19. │ s3_use_adaptive_timeouts                           │ 1          │
20. │ s3_truncate_on_insert                              │ 0          │
21. │ s3_create_new_file_on_insert                       │ 0          │
22. │ s3_skip_empty_files                                │ 0          │
23. │ s3_check_objects_after_upload                      │ 0          │
24. │ s3_allow_parallel_part_upload                      │ 1          │
25. │ s3_throw_on_zero_files_match                       │ 0          │
26. │ s3_ignore_file_doesnt_exist                        │ 0          │
27. │ s3_validate_request_settings                       │ 1          │
28. │ s3_disable_checksum                                │ 0          │
29. │ s3_retry_attempts                                  │ 100        │
30. │ s3_request_timeout_ms                              │ 30000      │
31. │ s3_connect_timeout_ms                              │ 1000       │
    └────────────────────────────────────────────────────┴────────────┘
```

Описание параметров можно посмотреть в статье Session Settings документации ClickHouse.

Параметры подключения к конечной точке

В конфигурационном файле (*/etc/clickhouse-server/config.xml* или в отдельном XML-файле директории */etc/clickhouse-server/config.d/*) внутри тега `<s3>` можно настроить параметры для конкретной конечной точки (endpoint), которые будут использоваться в запросах, где префикс указанного URL точно совпадает с этой конечной точкой.

Конфигурация в общем виде:

```
<clickhouse>
    <s3>
        <endpoint_config>
            <endpoint>ENDPOINT_URL</endpoint>
            <access_key_id>ACCESS_KEY_ID</access_key_id>
            <secret_access_key>SECRET_ACCESS_KEY</secret_access_key>
            <!-- Other endpoint-related parameters -->
        </endpoint_config>
    </s3>
</clickhouse>
```

где:

- `endpoint_config` — название конфигурации.
- `ENDPOINT_URL` — URL точки приема запроса на стороне S3, включающий URL бакета и путь к каталогу, где хранятся данные (префикс). Обязательный параметр.
- `ACCESS_KEY_ID`, `SECRET_ACCESS_KEY` — учетные данные для доступа к данной конечной точке. Эти параметры не являются обязательными — например, они не указываются, если бакет публичный или включена опция для считывания учетных данных из переменных окружения (см. следующий раздел).

Полный список настроек подключения к конечной точке S3 приведен в разделе Endpoint-based Settings документации ClickHouse.

Получение учетных данных для доступа к S3 из переменных окружения

Чтобы явно не указывать учетные данные доступа к бакету S3 в запросах или конфигурационных файлах, в ADQM можно включить опцию `use_environment_credentials` для считывания данных из переменных окружения `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, `AWS_SESSION_TOKEN` и метаданных Amazon EC2:

- на уровне конечной точки S3:

```
<clickhouse>
    <s3>
        <endpoint_config>
            <use_environment_credentials>true</use_environment_credentials>
            <!-- Other endpoint-related parameters -->
        </endpoint_config>
    </s3>
</clickhouse>
```

- на глобальном уровне:

```
<clickhouse>
    <s3>
        <use_environment_credentials>true</use_environment_credentials>
    </s3>
</clickhouse>
```

Табличный движок S3

Табличный движок S3 позволяет направлять запросы `SELECT` и `INSERT` к данным в хранилище S3.

Создание таблицы

Базовый синтаксис запроса для создания таблицы S3:

```
CREATE TABLE <table_name> (<column_name> <column_type>, ...)
ENGINE = S3('<path>' [, NOSIGN | '<access_key_id>', '<secret_access_key>' [, '<session_token>']]
            [, '<format>'] [, '<compression>'] [, <headers>])
[PARTITION BY <expr>]
[SETTINGS <setting_name> = <setting_value>, ...];
```

Описание параметров приведено выше.

Движок не поддерживает запросы `ALTER` и `SELECT…SAMPLE`, а также индексирование.

Пример

Создание таблицы S3

1. В консольном клиенте clickhouse-client создайте таблицу на базе движка S3 со структурой, соответствующей данным, которые будут импортироваться из S3:

```
CREATE TABLE s3_table (name String, value UInt32)
ENGINE=S3('https://storage.yandexcloud.net/test-bucket/adqm/test_1.txt',
          '<access_key_id>',
          '<secret_access_key>',
          'CSV');
```

   где `<access_key_id>` и `<secret_access_key>` — ключ доступа к бакету *test-bucket*.

2. Запросите данные из таблицы:

```
SELECT * FROM s3_table;
```

```
   ┌─name──┬─value─┐
1. │ one   │     1 │
2. │ two   │     2 │
3. │ three │     3 │
   └───────┴───────┘
```

Вставка данных в таблицу S3

1. Включите настройку `s3_create_new_file_on_insert`, чтобы при каждой вставке данных создавался новый файл в S3 (например, если таблица была создана на основе файла *file_name.txt*, новым файлам названия будут назначаться по шаблону — *file_name.1.txt*, *file_name.2.txt* и так далее):

```
SET s3_create_new_file_on_insert = 1;
```

2. Выполните вставку данных, повторив дважды запрос `INSERT INTO`:

```
INSERT INTO s3_table VALUES('four', 4);
```

```
INSERT INTO s3_table VALUES('five', 5);
```

3. Проверьте содержимое бакета — в каталог *adqm* добавлено два новых файла *test_1.1.txt* и *test_1.2.txt*, которые содержат по одной строке (`"four",4` и `"five",5` соответственно).
   При последующем чтении таблицы можно убедиться, что в выборку будут включены данные из всех файлов:

```
SELECT * FROM s3_table;
```

```
┌─name──┬─value─┐
│ one   │     1 │
│ two   │     2 │
│ three │     3 │
└───────┴───────┘
┌─name─┬─value─┐
│ five │     5 │
└──────┴───────┘
┌─name─┬─value─┐
│ four │     4 │
└──────┴───────┘
```

4. Активируйте опцию `s3_truncate_on_insert` и выполните еще одну вставку данных в таблицу — в этом случае последний файл с данными будет перезаписан:

```
SET s3_truncate_on_insert = 1;
```

```
INSERT INTO s3_table VALUES('six', 6);
```

5. Проверьте содержимое бакета. Список файлов в каталоге *adqm* не изменился, но файл *test_1.2.txt* был обновлен — теперь он содержит строку `"six",6`.
   Если настройка `s3_truncate_on_insert` включена, новые данные будут заменять текущее содержимое существующего файла в любом случае, независимо от значения `s3_create_new_file_on_insert`.

Табличные функции

ADQM также предоставляет две табличные функции для интеграции с S3:

- `s3` — создает таблицу для чтения/записи данных в S3.
  Синтаксис функции:

```
s3('<path>' [, NOSIGN | '<access_key_id>', '<secret_access_key>' [, '<session_token>']]
   [, '<format>'] [, '<structure>'] [, '<compression>'] [, <headers>])
```

  где `<structure>` — структура таблицы в формате `column_name1 data_type1, column_name2 data_type2, …`. Остальные параметры описаны выше.

- `s3Cluster` — позволяет обрабатывать файлы S3 параллельно из нескольких узлов указанного кластера. На узле-инициаторе функция создает соединение со всеми узлами кластера, заменяет символы `*` в пути к файлу S3 и динамически отправляет каждый файл. На рабочем узле функция запрашивает у инициатора следующую задачу и обрабатывает ее. Это повторяется до тех пор, пока все задачи не будут завершены.
  Синтаксис функции:

```
s3Cluster('<cluster_name>',
          '<path>' [, NOSIGN | '<access_key_id>', '<secret_access_key>' [, '<session_token>']]
          [, '<format>'] , '<structure>' [, '<compression>'] [, <headers>])
```

  где `<cluster_name>` — имя кластера, используемое для создания набора адресов и параметров подключения к удаленным и локальным серверам. Остальные параметры аналогичны параметрам функции `s3`, при этом указание структуры таблицы является обязательным.

Пример

Чтение данных из S3

Выполните следующий запрос, чтобы получить в ADQM данные из файла S3 *test_2.txt* с помощью функции `s3`:

```
SELECT * FROM s3('https://storage.yandexcloud.net/test-bucket/adqm/test_2.txt',
                 '<access_key_id>',
                 '<secret_access_key>',
                 'CSV',
                 'name String, value UInt32');
```

```
   ┌─name───┬─value─┐
1. │ ten    │    10 │
2. │ twenty │    20 │
3. │ thirty │    30 │
   └────────┴───────┘
```

Запись данных в S3

Если настройка `s3_truncate_on_insert` включена, при вставке данных через функцию `s3` существующие данные в файле, на которую ссылается функция, будут заменяться новыми данными.

Например, выполните следующий запрос (значение `s3_truncate_on_insert` было установлено в `1` в приведенном выше примере для табличного движка S3):

```
INSERT INTO FUNCTION s3('https://storage.yandexcloud.net/test-bucket/adqm/test_2.txt',
                        '<access_key_id>', '<secret_access_key>',
                        'CSV')
VALUES ('s3_func_test_string_1', 100);
```

Проверьте, что содержимое файла *test_2.txt* обновилось:

```
SELECT * FROM s3('https://storage.yandexcloud.net/test-bucket/adqm/test_2.txt',
                 '<access_key_id>',
                 '<secret_access_key>',
                 'CSV');
```

```
   ┌─c1────────────────────┬──c2─┐
1. │ s3_func_test_string_1 │ 100 │
   └───────────────────────┴─────┘
```

Если значение параметра `s3_truncate_on_insert` установлено в `0` и при этом активирована настройка `s3_create_new_file_on_insert`, при вставке данных в S3 через функцию `s3`, которая ссылается на файл *file_name.txt*, будет создан (или перезаписан, если уже существует) файл с именем *file_name.1.txt*.

Обновите настройки вставки данных и вызовите функцию `s3` для записи данных в файл *test_2.txt*:

```
SET s3_truncate_on_insert = 0, s3_create_new_file_on_insert = 1;
```

```
INSERT INTO FUNCTION s3('https://storage.yandexcloud.net/test-bucket/adqm/test_2.txt',
                        '<access_key_id>', '<secret_access_key>',
                        'CSV')
VALUES ('s3_func_test_string_2', 200);
```

В результате данные запишутся в новый файл *test_2.1.txt*:

```
SELECT * FROM s3('https://storage.yandexcloud.net/test-bucket/adqm/test_2.1.txt',
                 '<access_key_id>',
                 '<secret_access_key>',
                 'CSV');
```

```
   ┌─c1────────────────────┬──c2─┐
1. │ s3_func_test_string_2 │ 200 │
   └───────────────────────┴─────┘
```

Использование именованных коллекций

Чтобы не передавать все параметры подключения к хранилищу S3 каждый раз при создании таблицы S3 или вызове табличной функции, можно указать их один раз в именованной коллекции (named collection) в виде списка пар ключ/значение. Кроме того, этот подход позволяет скрыть конфиденциальные учетные данные для интеграции с S3 от пользователей без прав доступа администратора.

Ключи параметров

Основные ключи в именованной коллекции для хранения настроек подключения к хранилищу S3 соответствуют именам описанных выше параметров: `url`, `access_key_id`, `secret_access_key`, `session_token`, `format`, `compression` (или `compression_method`), `structure`, `use_environment_credentials`. Дополнительно поддерживаются следующие параметры.

filename	Имя файла, которое добавляется к URL
no_sign_request	Указывает, нужно ли игнорировать учетные данные, чтобы запросы не подписывались (например, при обращении к публичным бакетам)
expiration_window_seconds	Период для проверки истечения срока действия учетных данных
max_connections	Максимальное количество подключений к серверу
max_single_read_retries	Максимальное количество попыток запроса во время одного чтения S3
min_upload_part_size	Минимальный размер части объекта для загрузки при выполнении multipart-загрузки в S3
upload_part_size_multiply_parts_count_threshold	Количество частей объекта, при загрузке которого в S3 значение `min_upload_part_size` умножается на `upload_part_size_multiply_factor`
upload_part_size_multiply_factor	Коэффициент, на который умножается значение `min_upload_part_size` каждый раз, когда количество частей объекта, равное значению `upload_part_size_multiply_parts_count_threshold`, загрузилось одной операцией записи в S3
max_single_part_upload_size	Максимальный размер объекта для загрузки с использованием singlepart-загрузки в S3

Пример

1. Создайте именованную коллекцию с параметрами подключения к бакету S3:

```
CREATE NAMED COLLECTION test_bucket_creds AS
url = 'https://storage.yandexcloud.net/test-bucket/adqm/',
access_key_id = '<access_key_id>',
secret_access_key = '<secret_access_key>';
```

2. Используйте именованную коллекцию при вызове табличной функции `s3` (файл данных не определен в именованной коллекции, поэтому его необходимо указать отдельно в параметрах функции):

```
SELECT * FROM s3(test_bucket_creds, filename = 'test_1.txt');
```

```
   ┌─c1────┬─c2─┐
1. │ one   │  1 │
2. │ two   │  2 │
3. │ three │  3 │
   └───────┴────┘
```

3. Создайте таблицу S3 с помощью именованной коллекции и считайте из нее данные:

```
CREATE TABLE s3_table_nc (name String, value UInt32)
ENGINE=S3(test_bucket_creds, filename = 'test_1.txt');
```

```
SELECT * FROM s3_table_nc;
```

```
   ┌─name──┬─value─┐
1. │ one   │     1 │
2. │ two   │     2 │
3. │ three │     3 │
   └───────┴───────┘
```

Символы подстановки в пути к файлам S3

Если таблица ADQM будет принимать данные из нескольких файлов S3 и использоваться только для чтения, путь к файлам S3 в параметре `path` табличного движка S3 или табличной функции `s3`/`s3Cluster` можно указать с использованием символов подстановки:

- `*` — заменяет любое количество любых символов кроме разделителя пути `/`, включая отсутствие символов;
- `**` — заменяет любое количество любых символов, включая разделитель пути `/` и отсутствие символов (этот шаблон можно использовать для рекурсивного обхода каталога);
- `?` — заменяет любой одиночный символ;
- `{first_string,second_string,third_one}` — заменяет любую из строк `'first_string'`, `'second_string'`, `'third_one'` (в этом шаблоне также можно использовать числа — например, `{1,3,5}`);
- `{n..m}` — заменяет любое число в диапазоне `[n, m]`. Если список файлов содержит диапазон чисел с ведущими нулями, можно использовать общий шаблон `{0n..0m}` или отдельный шаблон `{n..m}` (либо знак `?`) для каждой цифры в названии файла.

Символы подстановки могут содержаться в нескольких компонентах пути (например, в названии каталога и в названии файла). Обрабатываются только существующие файлы, пути и названия которых полностью соответствуют шаблону. Список файлов определяется во время выполнения операции `SELECT` (не `CREATE`).

Примеры

Импортировать в ADQM данные из приведенных выше файлов *test_1.txt* и *test_2.txt* можно одним из способов:

```
SELECT * FROM s3(test_bucket_creds, filename = 'test_{1,2}.txt');
```

```
SELECT * FROM s3(test_bucket_creds, filename = 'test_?.txt');
```

Получить данных из всех файлов в каталоге *adqm* можно с помощью следующего запроса:

```
SELECT * FROM s3(test_bucket_creds, filename = '*');
```

Партиционирование данных при записи

Если при создании таблицы S3 указать выражение `PARTITION BY`, то для каждого значения ключа партиционирования в хранилище S3 будет создаваться отдельный файл при вставке данных в таблицу. Например, может быть полезно разделять данные на отдельные файлы для их дальнейшего переноса в другую систему — так как наборы данных ClickHouse часто очень велики, а надежность сети иногда недостаточна, имеет смысл передавать наборы данных относительно небольшими частями.

Ключ партиционирования также можно указать при экспорте данных из ADQM в S3 через табличную функцию: `INSERT INTO FUNCTION s3(…) PARTITION BY <expr> VALUES …`.

ПРИМЕЧАНИЕ

- Не следует применять слишком детализированное партиционирование данных (например, более детализированное, чем по месяцам).
- Партиционирование данных не ускоряет выполнение запросов (в отличие от выражения `ORDER BY`). Например, не стоит разделять данные по идентификаторам или именам пользователей — вместо этого укажите столбец с идентификаторами или именами первым в выражении `ORDER BY`.
- В настоящее время не поддерживается выборка данных напрямую из партиционированных таблиц S3 — данные могут быть получены из отдельных партиций с помощью табличной функции `s3`.

Ниже приведен пример деления вставляемых в таблицу S3 данных по значениям столбца, который имеет низкую кардинальность (небольшое количество уникальных значений).

1. Создайте таблицу S3, указав столбец `value` в качестве ключа партиционирования и параметр `{_partition_id}` в имени файла:

```
CREATE TABLE s3_table_partition (name String, value UInt32)
ENGINE=S3('https://storage.yandexcloud.net/test-bucket/adqm/test_partition_{_partition_id}.txt',
          '<access_key_id>',
          '<secret_access_key>',
          'CSV')
PARTITION BY value;
```

2. Вставьте в таблицу данные:

```
INSERT INTO s3_table_partition
VALUES ('one_1', 1), ('two_1', 2), ('one_2', 1), ('two_2', 2), ('three_1', 3);
```

3. В результате данные будут записаны в три файла, в названиях которых параметр `{_partition_id}` заменен на соответствующие значения ключа партиционирования: *test_partition_1.txt*, *test_partition_2.txt* и *test_partition_3.txt*. Параметр можно также указать в названии каталога (в префиксе).

4. С помощью табличной функции `s3` считайте данные из отдельных файлов, например:

```
SELECT * FROM s3('https://storage.yandexcloud.net/test-bucket/adqm/test_partition_1.txt',
                 '<access_key_id>',
                 '<secret_access_key>',
                 'CSV');
```

```
   ┌─c1────┬─c2─┐
1. │ one_1 │  1 │
2. │ one_2 │  1 │
   └───────┴────┘
```

```
SELECT * FROM s3('https://storage.yandexcloud.net/test-bucket/adqm/test_partition_2.txt',
                 '<access_key_id>',
                 '<secret_access_key>',
                 'CSV');
```

```
   ┌─c1────┬─c2─┐
1. │ two_1 │  2 │
2. │ two_2 │  2 │
   └───────┴────┘
```

   При этом, если попробовать получить данные напрямую из таблицы `s3_table_partition`, будет выведено сообщение об ошибке:

```
Exception: Reading from a partitioned S3 storage is not implemented yet. (NOT_IMPLEMENTED)
```

Виртуальные столбцы

В таблицы на базе движка S3 и таблицы, возвращаемые табличными функциями `s3` и `s3Cluster`, автоматически добавляются следующие виртуальные столбцы.

Название столбца	Тип данных	Описание
_path	LowCardinality(String)	Путь к файлу
_file	LowCardinality(String)	Имя файла
_size	Nullable(UInt64)	Размер файла в байтах
_time	Nullable(DateTime)	Время последнего изменения файла
_etag	LowCardinality(String)	ETag файла

Виртуальные столбцы можно использовать в запросах `SELECT`, чтобы получить информацию о файлах в бакете S3, к которым обращается таблица ADQM. Например, следующий запрос возвращает список всех *.txt*-файлов в каталоге *adqm* бакета *test-bucket* и вычисляет количество строк в каждом файле:

```
SELECT _file, count() AS count
FROM s3('https://storage.yandexcloud.net/test-bucket/adqm/*.txt', '<access_key_id>', '<secret_access_key>', 'CSV')
GROUP BY _file
ORDER BY _file;
```

```
   ┌─_file────────────────┬─count─┐
1. │ test_1.1.txt         │     1 │
2. │ test_1.2.txt         │     1 │
3. │ test_1.txt           │     3 │
4. │ test_2.1.txt         │     1 │
5. │ test_2.txt           │     1 │
6. │ test_partition_1.txt │     2 │
7. │ test_partition_2.txt │     2 │
8. │ test_partition_3.txt │     1 │
   └──────────────────────┴───────┘
```

Импорт и экспорт в файлы

Елена Дворядкина

Содержание

Назад к содержанию

Импорт данных из файлов
- FROM INFILE
- Табличный движок File
- Табличная функция file
- Перенаправление данных в командой строке

Экспорт данных в файлы
- INTO OUTFILE
- Табличный движок File
- Табличная функция file
- Перенаправление данных в командой строке

Импорт данных из файлов

ADQM поддерживает несколько способов импорта данных из файлов в таблицы.

FROM INFILE

Чтобы вставить в таблицу данные из локального файла или нескольких файлов сразу (поддерживаются в том числе сжатые файлы), можно использовать выражение `FROM INFILE` при выполнении запроса `INSERT INTO` в клиенте командной строки clickhouse-client. При этом файлы должны быть расположены в папке, из которой запущен `clickhouse-client`. Базовый синтаксис запроса:

```
INSERT INTO <table_name> [(<column_list>)] FROM INFILE '<file_name.ext>' [COMPRESSION '<compr_type>'] [FORMAT <input_format>];
```

В запросе с выражением `FROM INFILE` используются следующие аргументы.

<file_name.ext>	Имя файла с расширением, из которого будут загружаться данные в таблицу ADQM
<compr_type>	Метод сжатия файла. Возможные значения: `none`, `gzip`, `deflate`, `br`, `xz`, `zstd`, `lz4`, `bz2`. По умолчанию метод сжатия определяется по расширению файла
<input_format>	Формат импортируемых данных. Доступные для импорта данных форматы перечислены в статье Formats for Input and Output Data документации ClickHouse. По умолчанию формат автоматически определяется по расширению файла. Примеры использования различных форматов для импорта данных JSON в зависимости от структуры входящих данных приведены в статье Тип данных JSON

Пример

1. Создайте два локальных тестовых файла *input_1.csv* и *input_2.csv* с данными для последующей вставки в таблицы ADQM:

   ```
   1,"aaa",100
   2,"bbb",200
   3,"ccc",300
   ```

   ```
   4,"ddd",400
   5,"eee",500
   6,"fff",600
   ```

   Запустите консольный клиент clickhouse-client из папки, где находятся эти файлы.

2. Создайте таблицу с соответствующей структурой:

   ```
   CREATE TABLE test_table (a Int32, b String, c Int32) ENGINE = MergeTree ORDER BY a;
   ```

3. Выполните запрос для импорта данных одного из файлов в созданную таблицу:

   ```
   INSERT INTO test_table FROM INFILE 'input_1.csv';
   ```

4. Проверьте, что данные загрузились в таблицу:

   ```
   SELECT * FROM test_table;
   ```

   ```
   ┌─a─┬─b───┬───c─┐
   │ 1 │ aaa │ 100 │
   │ 2 │ bbb │ 200 │
   │ 3 │ ccc │ 300 │
   └───┴─────┴─────┘
   ```

5. Вставьте в таблицу данные из двух файлов сразу, указав `input_*.csv` в качестве имени файла:

   ```
   INSERT INTO test_table FROM INFILE 'input_*.csv';
   ```

6. Проверьте результат импорта:

   ```
   SELECT * FROM test_table;
   ```

   ```
   ┌─a─┬─b───┬───c─┐
   │ 1 │ aaa │ 100 │
   │ 2 │ bbb │ 200 │
   │ 3 │ ccc │ 300 │
   │ 4 │ ddd │ 400 │
   │ 5 │ eee │ 500 │
   │ 6 │ fff │ 600 │
   └───┴─────┴─────┘
   ┌─a─┬─b───┬───c─┐
   │ 1 │ aaa │ 100 │
   │ 2 │ bbb │ 200 │
   │ 3 │ ccc │ 300 │
   └───┴─────┴─────┘
   ```

Табличный движок File

Загружать данные в ADQM из файла можно используя табличный движок File, который хранит данные в файле. Данные, отредактированные непосредственно в файле на диске, автоматически обновляются в таблице типа File при выполнении запроса `SELECT`. При создании таблицы на основе этого движка для импорта данных из файла в качестве аргумента должен быть указан формат, поддерживаемый для входных данных — см. форматы в колонке ***Input*** таблицы Formats for Input and Output Data. Особенностью этого движка также является то, что путь к файлу для хранения данных указать явно нельзя — используется каталог, установленный через параметр ***Data path*** на странице конфигурации сервиса ADQMDB (по умолчанию */var/lib/clickhouse/*).

Пример

1. Создайте таблицу `file_table` с использованием движка File:

   ```
   CREATE TABLE file_table (name String, value UInt32) ENGINE=File(TabSeparated);
   ```

   ADQM создаст каталог */var/lib/clickhouse/data/default/file_table*.

2. Создайте вручную файл */var/lib/clickhouse/data/default/file_table/data.TabSeparated* и заполните его тестовыми данными:

   ```
   $ sudo cat > /var/lib/clickhouse/data/default/file_table/data.TabSeparated
   a     1
   b     2
   ```

   Чтобы в дальнейшем иметь возможность экспортировать данные из таблицы в этот файл, назначьте ему владельца `clickhouse` следующей командой:

   ```
   $ sudo chown clickhouse:clickhouse /var/lib/clickhouse/data/default/file_table/data.TabSeparated
   ```

3. Запросите данные из таблицы `file_table`:

   ```
   SELECT * FROM test_table;
   ```

   ```
   ┌─name─┬─value─┐
   │ a    │     1 │
   │ b    │     2 │
   └──────┴───────┘
   ```

Табличная функция file

Для вставки данных из файла в таблицу можно также использовать табличную функцию `file` — см. пример в статье Табличные функции.

Перенаправление данных в командой строке

Импортировать данные из файла в таблицу можно напрямую в командой строке clickhouse-client, например:

```
$ clickhouse-client -q "INSERT INTO test_table FORMAT CSV" < input_2.csv
```

Экспорт данных в файлы

Экспортировать данные из ADQM в файлы можно перечисленными ниже способами.

INTO OUTFILE

Чтобы сохранить результаты запроса `SELECT` в файл, при выполнении этого запроса в клиенте командной строки clickhouse-client можно использовать выражение `INTO OUTFILE`. Базовый синтаксис:

```
SELECT <expr_list>
[FROM <table_name>]
...
INTO OUTFILE '<file_name.ext>' [AND STDOUT] [COMPRESSION '<compr_type>' [LEVEL <compr_level>]]
[FORMAT <output_format>];
```

В запросе с выражением `INTO OUTFILE` используются следующие аргументы.

<file_name.ext>	Имя файла с расширением, в который будут сохраняться результаты запроса `SELECT`. Файл будет создан в папке, из которой запущен clickhouse-client. Если файл с указанным именем уже существует, выполнение запроса завершится ошибкой
<compr_type>	Метод сжатия. Возможные значения: `none`, `gzip`, `deflate`, `br`, `xz`, `zstd`, `lz4`, `bz2`. По умолчанию метод сжатия определяется по расширению файла
<compr_level>	Уровень сжатия для указанного метода. Возможные значения — положительные целые числа в следующих диапазонах: ▪ `[1, 12]` — для метода `lz4`; ▪ `[1, 22]` — для метода `zstd`; ▪ `[1, 9]` — для остальных методов сжатия.
<output_format>	Формат экспортируемых данных. Доступные для экспорта данных форматы перечислены в статье Formats for Input and Output Data документации ClickHouse. По умолчанию формат определяется по расширению файла, или используется формат `TabSeparated`, если формат не удалось установить по расширению файла. Примеры использования различных форматов для вывода данных JSON представлены в статье Тип данных JSON

При использовании в запросе выражения `AND STDOUT` сохраняемые в файл данные также будут выведены на экран. Если применяется сжатие, данные на экране отобразятся в виде простого текста.

Пример запроса `SELECT … INTO OUTFILE`:

```
SELECT * FROM test_table INTO OUTFILE 'output.csv';
```

Табличный движок File

Еще один способ выгрузить данные из ADQM в файл — использовать табличный движок File. При создании таблицы с использованием этого движка для экспорта данных в файл (осуществляется при выполнении запросов `INSERT`) в качестве аргумента должен быть указан формат, поддерживаемый для выходных данных — см. форматы в колонке ***Output*** таблицы в статье Formats for Input and Output Data.

Пример

1. Создайте таблицу `file_table_out` на движке File:

   ```
   CREATE TABLE file_table_out (name String, value UInt32) ENGINE=File(TabSeparated);
   ```

   ADQM создаст каталог */var/lib/clickhouse/data/default/file_table_out*, где `/var/lib/clickhouse/` — каталог для хранения данных ADQM, который можно изменить через параметр ***Data path*** на странице конфигурации сервиса ADQMDB.

2. Заполните таблицу тестовыми данными:

   ```
   INSERT INTO file_table_out VALUES('a', 1), ('b', 2), ('c', 3);
   ```

3. Убедитесь, что в каталоге */var/lib/clickhouse/data/default/file_table_out* создан файл *data.TabSeparated*, который содержит данные из таблицы `file_table_out`:

   ```
   a     1
   b     2
   c     3
   ```

Табличная функция file

Для экспорта данных из ADQM в файл можно также использовать табличную функцию `file` — см. пример в статье Табличные функции.

Перенаправление данных в командой строке

Следующий запрос создаст файл *redirected_output.csv* в папке, из которой запущен clickhouse-client, и запишет в него результаты выполнения указанного запроса `SELECT` (если такой файл уже существует, данные будут перезаписаны):

```
$ clickhouse-client --query "SELECT * from test_table" --format CSV > redirected_output.csv
```

Конфигурационные параметры

Елена Дворядкина

Содержание

Назад к содержанию

- ADQMDB
- Chproxy
- Zookeeper
- Clickhousekeeper
- Monitoring

В этой статье описаны параметры, которые могут быть настроены для сервисов ADQM через ADCM. Процесс настройки сервисов приведен в статьях Настройка сервисов и Настройка сервисов.

ПРИМЕЧАНИЕ

Некоторые из перечисленных параметров становятся доступны в UI ADCM после установки флага ***Show advanced***.

ADQMDB

SSL Configuration

ПРИМЕЧАНИЕ

В статье Шифрование соединений с базой данных приведен пример, как настроить проверку SSL-сертификатов для соединений с ADQM.

Параметр	Описание	Значение по умолчанию
Enable SSL	Включает прослушивание портов для SSL-защищенных соединений по протоколам HTTPS и TCP	Неактивен
HTTPS port	Порт для безопасных соединений по протоколу HTTPS	8443
TCP secure port	TCP-порт для безопасных соединений с клиентами и межсерверной коммуникации по бинарному протоколу	9440
Disable protocols	Протоколы, которые не разрешено использовать для подключения	—
Server certificate file	Путь к файлу серверного SSL-сертификата в формате PEM	/etc/clickhouse-server/server.crt
Server private key file	Путь к файлу закрытого ключа серверного SSL-сертификата	/etc/clickhouse-server/server.key
Certificate authority file	Путь к файлу корневого сертификата центра сертификации (Certification Authority, CA) для проверки клиентских сертификатов при включенной TLS-аутентификации	—
Client certificate verification mode	Режим проверки клиентских SSL-сертификатов, подписанных корневым сертификатом (путь к файлу корневого сертификата должен быть указан в ***Certificate authority file***). Возможные значения: `none`, `relaxed`, `strict`, `once`	none
Client certificate file	Путь к файлу SSL-сертификата клиента в формате PEM (используется для защищенного соединения между ZooKeeper и ClickHouse)	—
Client private key file	Путь к файлу закрытого ключа SSL-сертификата клиента (используется для защищенного соединения между ZooKeeper и ClickHouse)	—
Client certificate authority file	Путь к файлу корневого сертификата центра сертификации для проверки клиентских сертификатов (используется для защищенного соединения между ZooKeeper и ClickHouse)	—

gRPC

ПРИМЕЧАНИЕ

Чтобы получить доступ к параметрам конфигурации, необходимо активировать ***Enable gRPC***.

Параметр	Описание	Значение по умолчанию
gRPC port	Порт для клиентских соединений через интерфейс gRPC	9100
Enable SSL	Включает SSL для защищенного взаимодействия с клиентами через gRPC	Неактивен
SSL certificate file	Путь к файлу SSL-сертификата (в формате PEM) для gRPC-подключений. Используется при включенном SSL	/etc/clickhouse-server/grpc_ssl_cert_file.crt
SSL private key file	Путь к файлу закрытого ключа SSL-сертификата для gRPC-подключений. Используется при включенном SSL	/etc/clickhouse-server/grpc_ssl_key_file.key
Client auth required	Указывает, будет ли сервер запрашивать у клиента сертификат. Если опция включена, необходимо указать ***Certificate authority file***	Неактивен
Certificate authority file	Путь к файлу корневого сертификата центра сертификации (Certification Authority, CA) в формате PEM, который будет использоваться для проверки клиентских сертификатов. Указывается при включенной опции ***Client auth required***	—
Compression algorithm	Алгоритм сжатия, необходимый для минимизации трафика. Возможные значения: `none`, `gzip`, `stream_gzip`	deflate
Compression level	Уровень сжатия трафика. Возможные значения: `none`, `low`, `medium`, `high`	medium
Send message size limits	Ограничение на размер отправляемых сообщений (в байтах). При `-1` ограничение отсутствует	-1
Receive message size limits	Ограничение на размер принимаемых сообщений (в байтах). При `-1` ограничение отсутствует	-1
Verbose logs	Указывает, записываются ли детализированные логи	Неактивен

LDAP

ПРИМЕЧАНИЕ

- Чтобы получить доступ к параметрам конфигурации, необходимо активировать ***Enable LDAP***.
- В статье Аутентификация с использованием LDAP описаны подходы использования удаленного сервера LDAP для аутентификации пользователей.
- Значения по умолчанию в таблице ниже указаны для сервера LDAP, добавленного в секцию ***LDAP server*** по умолчанию. При добавлении нового сервера LDAP все поля конфигурации изначально пустые, обязательные для заполнения поля подсвечиваются красным.

Параметр	Описание	Значение по умолчанию
name	Имя конфигурации подключения к серверу LDAP	default
ldap_hostname	Имя хоста или IP сервера LDAP	localhost
ldap_port	Порт сервера LDAP. Если настройка ***enable_tls*** установлена в `yes`, по умолчанию используется порт `636`, иначе — `389`	636
bind_dn	Шаблон для построения DN подключения. При формировании DN все подстроки `{user_name}` в шаблоне будут заменяться на фактическое имя пользователя при каждой попытке аутентификации	uid={user_name},ou=users,dc=example,dc=com
base_dn	DN объекта каталога, в котором выполняется поиск записей. Пример: `OU=Peoples,DC=ad,DC=ranger-test`	—
search_filter	Шаблон для построения фильтра поиска пользователя в дереве LDAP. При формировании фильтра все подстроки `{user_name}`, `{bind_dn}` и `{base_dn}` в шаблоне будут заменяться на фактическое имя пользователя, DN подключения и базовый DN соответственно при каждом LDAP-поиске. Пример: `(&(objectClass=user)(sAMAccountName={user_name}))`	—
enable_tls	Флаг для включения использования безопасного соединения с сервером LDAP. Доступны следующие значения: ▪ `no` — для использования текстового протокола (ldap://), не рекомендуется; ▪ `yes` — для обращения к LDAP по протоколу SSL/TLS (ldaps://), рекомендуется; ▪ `starttls` — для использования устаревшего протокола StartTLS (простой текстовый протокол, обновленный до TLS).	no
tls_minimum_protocol_version	Минимальная версия протокола SSL/TLS. Принимаются следующие значения: `ssl2`, `ssl3`, `tls1.0`, `tls1.1`, `tls1.2`	—
tls_require_cert	Поведение при проверке сертификата SSL/TLS. Принимаются следующие значения: `never`, `allow`, `try`, `demand`	—
tls_cert_file	Путь к файлу сертификата	—
tls_key_file	Путь к файлу ключа сертификата	—
tls_ca_cert_file	Путь к файлу корневого сертификата (Certification Authority, CA)	—
tls_ca_cert_dir	Путь к каталогу, содержащему корневые сертификаты	—
tls_cipher_suite	Разрешенный набор шифров	—
rbac_enable	Активирует управление доступом на основе ролей	Неактивен
rbac_default_role	Роль по умолчанию для всех аутентифицированных пользователей	default
rbac_base_dn	Базовый DN для поиска групп, включающих пользователя	ou=Groups,dc=ad,dc=local
rbac_scope	Область поиска LDAP. Возможные значения: `base`, `one_level`, `children`, `subtree`	—
rbac_search_filter	Фильтр поиска LDAP для определения групп, которые необходимо выбрать для сопоставления пользователей. При формировании фильтра все подстроки `{user_name}`, `{bind_dn}`, `{user_dn}` и `{base_dn}` в шаблоне будут заменяться на фактические имя пользователя, DN подключения, DN пользователя и базовый DN соответственно при каждом LDAP-поиске	(&amp;(objectClass=groupOfNames)(member={bind_dn}))
rbac_attribute	Атрибут для поиска в дереве LDAP	cn
rbac_role_prefix	Префикс значения атрибута для поиска в дереве LDAP	—

Kafka engine

ПРИМЕЧАНИЕ

Чтобы получить доступ к параметрам конфигурации, необходимо активировать опцию ***Kafka engine***.

Параметр	Описание	Значение по умолчанию
Kafka Properties	В этом поле необходимо указать параметры табличного движка Kafka. Например: `<debug>all</debug><auto_offset_reset>latest</auto_offset_reset><compression_type>snappy</compression_type>`. Поле обязательно	—

Coordination system

Параметр	Описание	Значение по умолчанию
Coordination system	Сервис координации для репликации данных и выполнения распределенных DDL-запросов. Возможные значения: `Zookeeper`, `Clickhouse_keeper (integrated)`, `Clickhouse_keeper (allocated)`	Zookeeper

Clickhouse Keeper (integrated)

ПРИМЕЧАНИЕ

Чтобы получить доступ к параметрам конфигурации, необходимо активировать опцию ***Clickhouse Keeper (integrated)***.

Параметр	Описание	Значение по умолчанию
Number of nodes in cluster	Количество узлов в кластере, которые будут использоваться для ClickHouse Keeper	1
Enable IPv6	Позволяет отключить IPv6, если хосты не поддерживают этот протокол	Активен
TCP port	Порт для подключения клиента	2129
Log storage path	Путь к журналам координации	/var/lib/clickhouse-keeper/coordination/log
Snapshot storage path	Путь к снепшотам координации	/var/lib/clickhouse-keeper/coordination/snapshots
Operation timeout ms	Тайм-аут для одной операции клиента (в миллисекундах)	10000
Session timeout ms	Тайм-аут для сеанса клиента (в миллисекундах)	30000
Snapshot distance	Частота, с которой ClickHouse Keeper будет делать новые снепшоты (по количеству записей в журналах)	100000
Raft logs level	Уровень логирования процесса координации (`trace`, `debug` и т.д.)	error
Rotate log storage interval	Количество записей в журнале для хранения в одном файле	10000
Port	Порт, на котором сервер прослушивает соединения RAFT	9444
root_path	Узел znode, который используется в качестве корня для узлов znode, используемых сервером ClickHouse	/clickhouse
Distributed DDL path	По этому ZK-пути ClickHouse отправляет DDL-задачи. В противном случае убедитесь, что в конфигурации нет пересечений с другими кластерами	/clickhouse/task_queue/ddl

Zookeeper

Параметр	Описание	Значение по умолчанию
root_path	Узел znode, который используется в качестве корня для узлов znode, используемых сервером ClickHouse	/clickhouse
Distributed DDL zookeeper path	По этому ZK-пути ClickHouse размещает DDL-задачи. Если этот CH-кластер единственный использует ZooKeeper — настраивать это поле не нужно. В противном случае убедитесь, что в конфигурации нет пересечений с другими кластерами	/clickhouse/task_queue/ddl
SSL connection	Включает использование безопасного соединения с ZooKeeper	Неактивен

External zookeeper

ПРИМЕЧАНИЕ

Для доступа к параметрам конфигурации необходимо активировать ***External zookeeper***.

Параметр	Описание	Значение по умолчанию
zk_hosts	Расположение всех серверов в ансамбле через запятую с указанием портов, через которые они взаимодействуют. Например, `host1.mydomain.com:2181,host2.mydomain.com:2181,host3.mydomain.com:2181`. Поле является обязательным	—

Limits

Параметр	Описание	Значение по умолчанию
Max connections	Максимальное количество входящих соединений	4096
Max concurrent queries	Максимальное количество одновременно обрабатываемых запросов	100
Max concurrent threads	Максимальное количество потоков для выполнения всех запросов (исключая потоки для получения данных с удаленных серверов). Если установленный предел достигнут, для выполнения запроса все равно будет выделен хотя бы один поток. Если значение параметра `0`, ограничений нет	0
Max server memory usage	Позволяет ограничить объем оперативной памяти (в байтах), используемой сервером ClickHouse. Если значение параметра `0`, ограничений нет	0
Max thread pool size	Позволяет ограничить количество потоков в глобальном пуле потоков, которые ClickHouse использует для обработки запросов (если в пуле нет свободных потоков для обработки запроса, создается новый поток)	10000

Network

Параметр	Описание	Значение по умолчанию
Enable http	Включает порт для соединения с ADQM по протоколу HTTP	Активен
ClickHouse http port	HTTP-порт для клиентских или прокси-соединений	8123
Enable tcp	Включает порт для соединения с ADQM по протоколу Native (ClickHouse TCP)	Активен
ClickHouse tcp port	Порт для TCP-соединений	9000
Listen host	Ограничение по хостам, с которых могут приходить запросы. Учитывайте возможное отсутствие IPv6 на целевом хосте и используйте, например, `0.0.0.0` в этом случае	::

Metrics

Параметр	Описание	Значение по умолчанию
Listen port	Порт доступа к метрикам ADQMDB в формате Prometheus	9363
Endpoint	Конечная точка HTTP, в которую экспортируются метрики ADQMDB	/metrics

Directories

Параметр	Описание	Значение по умолчанию
Data path	Путь к каталогу, содержащему данные	/var/lib/clickhouse/
Log path	Путь к каталогу, содержащему лог-файлы	/var/log/clickhouse-server/
Temp path	Путь к временным данным для обработки больших запросов	/var/lib/clickhouse/tmp/
User files path	Каталог с файлами пользователя. Используется в табличной функции `file()`	/var/lib/clickhouse/user_files/
Access control path	Путь к папке, в которой сервер ClickHouse хранит конфигурации пользователей и ролей, созданные командами SQL	/var/lib/clickhouse/access/

Cluster configuration

ПРИМЕЧАНИЕ

Особенности использования параметров конфигурирования кластера описаны в статье Конфигурирование логических кластеров в интерфейсе ADCM.

Параметр	Описание	Значение по умолчанию
Replication factor	Число шардов, умноженное на значение параметра `replication factor` — это общее число узлов в физическом кластере ADQM. Топология кластера и макросы генерируются один раз и не изменяются при дальнейшей реконфигурации кластера. Подробнее в статье Distributed Table Engine	1
Cluster Configuration	Позволяет добавлять параметры конфигурации кластера	—
Default cluster topology	Этот кластер включен в раздел `remote_servers` в конфигурации CH-узла. Чтобы получить доступ к этому параметру, необходимо активировать опцию ***Show advanced*** наверху страницы настройки сервиса	{}

Intra-cluster communication security

Параметр	Описание	Значение по умолчанию
Enable secure intra-cluster communication	При включении этой опции для обмена данными между серверами ClickHouse будут использоваться только безопасные протоколы (HTTPS, защищенный TCP). Для этого также необходимо активировать ***Enable SSL*** в секции ***SSL Configuration***	Неактивен
Interserver protocol's port	Порт для обмена данными между серверами ClickHouse по межсерверному протоколу	9009
Interserver http name	Имя пользователя для аутентификации между репликами	—
Interserver http password	Пароль для аутентификации между репликами	—
Shard secret	Ключ кластера для проверки распределенных запросов на шардах	—

Kerberos configuration

Параметр	Описание	Значение по умолчанию
security	Тип используемого модуля Kerberos	—
keytab	Путь к файлу keytab	—
reinit_frequency	Частота реинициализации Kerberos	3600
principal	Принципал (principal) — уникальное имя для клиента, которому разрешается аутентификация в Kerberos	—

Log settings

ПРИМЕЧАНИЕ

В статье Логирование приведена общая информация о логах, которые ведет и записывает в файлы или системные таблицы ADQM.

Параметр	Описание	Значение по умолчанию
Logging level	Уровень детализации сообщений, записываемых в лог-файл. Возможные значения: `trace`, `debug`, `information`, `notice`, `warning`, `error`, `critical`, `fatal`, `none`	trace
Log file size, MB	Размер лог-файла (в МБ). Как только файл достигает указанного размера, он архивируется и переименовывается, а на его месте создается новый лог-файл	1000
Log files count	Количество архивных лог-файлов, которые хранит ADQM	10
Database	Имя системной базы данных, в которую записываются логи	system
Flush interval, milliseconds	Интервал перемещения данных из буфера памяти в таблицу (в миллисекундах)	7500
TTL, days	Время жизни логов (в днях)	30
query_log	Информация о выполненных запросах (например, время начала выполнения запроса, продолжительность обработки запроса, сообщения об ошибках). Это часто используемый лог. Он ведется, если установлен флаг. Чтобы отключить его, установите настройку `log_queries = 0` на уровне сеанса или запроса	Неактивен
trace_log	Трассировки стека, собранные семплирующим профилировщиком запросов, которые могут быть полезны для глубокого анализа производительности. Лог ведется, если установлен флаг	Неактивен
query_views_log	Информация о зависимых представлениях, выполняемых при обработке запросов (например, тип представления, продолжительность выполнения представления). Лог отключен по умолчанию (даже если установлен флаг) и работает только в том случае, если установлена настройка `log_query_views = 1` на уровне сеанса или запроса	Неактивен
query_thread_log	Информация о потоках выполнения запросов (например, имя потока, время начала выполнения запроса потоком, продолжительность обработки запроса). Лог отключен по умолчанию (даже если установлен флаг) и работает только в том случае, если установлена настройка `log_query_threads = 1` на уровне сеанса или запроса	Неактивен
part_log	Информация о событиях, связанных с кусками данных таблиц семейства MergeTree (например, добавление или объединение данных). Лог ведется, если установлен флаг, и не контролируется на уровне сеанса	Неактивен
crash_log	Информация о трассировках стека для фатальных ошибок. Лог ведется, если установлен флаг, и не контролируется на уровне сеанса	Неактивен
session_log	Информация обо всех успешных и неудачных событиях входа и выхода из системы. Логирование ведется, если установлен флаг, и не контролируется на уровне сеанса	Неактивен
metric_log	Указывает, записывается ли история значений метрик из таблиц `system.metrics` и `system.events` в таблицу `system.metric_log`	Неактивен
asynchronous_metric_log	Указывает, записывается ли история значений метрик из таблицы `system.asynchronous_metrics` в таблицу `system.asynchronous_metric_log`	Неактивен
asynchronous_insert_log	Информация об асинхронных вставках, включая результаты запросов в режиме fire-and-forget	Неактивен

Default user and policy settings

Параметр	Описание	Значение по умолчанию
default_profile_settings	Позволяет добавить настройки профиля по умолчанию	`max_memory_usage : 10000000000` `use_uncompressed_cache : 0`
Access Management	Контроль доступа и управление учетными записями для пользователя по умолчанию на основе SQL	Активен
Default user password	Пароль пользователя по умолчанию	—
Default user IP	Пользователь может подключаться к серверу ClickHouse только с указанного IP-адреса или подсети. Примеры: `192.168.0.0/16`, `2001:DB8::/32`	—
Default user host	Хост пользователя может быть указан как FQDN. Например, `mysite.com`	—
Default user host_regexp	Вы можете использовать регулярные выражения pcre при указании пользовательских хостов. Например, `.*\.mysite\.com`	—

User password options

Параметр	Описание	Значение по умолчанию
Allow plaintext password	Разрешает или запрещает назначать пользователям пароли в открытом текстовом формате	Активен
Allow NO_password	Разрешает или запрещает создание пользователя без пароля	Активен
Allow Implicit_NO_password	В сочетании с ***Allow NO_password*** запрещает создание пользователя без пароля, если явно не указано IDENTIFIED WITH no_password	Активен
Length of password	Минимальная длина пароля. По умолчанию `0` — длина пароля не ограничена	0
Include Upper-case	Указывает, должен ли пароль содержать хотя бы один символ в верхнем регистре	Неактивен
Include Lower-case	Указывает, должен ли пароль содержать хотя бы один символ в нижнем регистре	Неактивен
Include Number	Указывает, должен ли пароль содержать хотя бы одну цифру	Неактивен
Include special symbols	Указывает, должен ли пароль содержать хотя бы один специальный символ	Неактивен

Advanced configuration parameters

Параметр	Описание	Значение по умолчанию
Advanced parameters (name/value)	Параметры конфигурации для файла ClickHouse *advanced.xml*	—
Advanced custom XML	Настройки сервера ClickHouse в формате XML. Параметры должны быть указаны внутри тега `<clickhouse>` и соответствовать правилам XML, применяемым в конфигурационных файлах ClickHouse	`<clickhouse>` `</clickhouse>`

Additional local storage

ПРИМЕЧАНИЕ

- Чтобы получить доступ к параметрам конфигурации, необходимо активировать ***Enable additional local storage***.
- В статье Настройка хранилищ данных приведен пример, как сконфигурировать многоуровневое хранилище данных с использованием дополнительных блочных устройств (SSD- и HDD-диска).

Параметр	Описание	Значение по умолчанию
Disks	Список дисков в локальной файловой системе, на которых можно хранить табличные данные	—
Recursively create directories	Указывает, должны ли автоматически создаваться директории для локальных дисков, если они не существуют	Активен

S3 storage

ПРИМЕЧАНИЕ

- Чтобы получить доступ к параметрам конфигурации, необходимо активировать ***Enable S3 storage***.
- В статье Настройка хранилищ данных описаны детали работы с дисками разных типов для хранения данных таблиц ADQM, в том числе приводятся описания параметров конфигурирования дисков S3.

Параметр	Описание	Значение по умолчанию
S3 disks	Список внешних дисков S3, которые можно использовать для хранения табличных данных	—

Encrypted storage

ПРИМЕЧАНИЕ

- Чтобы получить доступ к параметрам конфигурации, необходимо активировать ***Enable encrypted storage***.
- В статье Настройка хранилищ данных описаны детали работы с дисками разных типов для хранения данных таблиц ADQM, в том числе приводятся описания параметров конфигурирования зашифрованных дисков.

Параметр	Описание	Значение по умолчанию
Encrypted disks	Список зашифрованных дисков, создаваемых поверх дисков других типов (локальных, S3, кеширующих)	—
Recursively create directories	Указывает, должны ли автоматически создаваться директории для зашифрованных дисков, если они не существуют	Активен

Cache

ПРИМЕЧАНИЕ

- Чтобы получить доступ к параметрам конфигурации, необходимо активировать ***Enable cache***.
- В статье Настройка хранилищ данных описаны детали работы с дисками разных типов для хранения данных таблиц ADQM, в том числе приводятся описания параметров конфигурирования кеширующих дисков.

Параметр	Описание	Значение по умолчанию
Cache disks	Список кеширующих дисков, создаваемых поверх внешних дисков с данными	—
Recursively create directories	Указывает, должны ли автоматически создаваться директории для кеширующих дисков, если они не существуют	Активен

Storage policies

ПРИМЕЧАНИЕ

Назначение и параметры конфигурирования политик хранения для таблиц ADQM описаны в статье Настройка хранилищ данных.

Параметр	Описание	Значение по умолчанию
Policies	Список политик хранения, которые могут быть назначены таблицам	—

Data compression

ПРИМЕЧАНИЕ

- Для доступа к параметрам конфигурации необходимо активировать переключатель ***Data compression***.
- Информацию о доступных в ADQM алгоритмах сжатия и способах их применения можно получить в статье Сжатие данных.

Параметр	Описание	Значение по умолчанию
Minimum part size	Минимальный размер куска данных (в байтах)	10000000000
Minimum part size ratio	Отношение минимального размера куска данных к размеру таблицы	0.01
Method	Метод сжатия. Возможные значения: `LZ4`, `LZ4HC`, `ZSTD`	LZ4
Level	Уровень сжатия. Возможные значения: ▪ для метода `LZ4HC` — [`1`, `12`]; ▪ для метода `ZSTD` — [`1`, `22`]; ▪ метод `LZ4` не имеет уровней сжатия — любое значение игнорируется.	9

JDBC Bridge configuration

ПРИМЕЧАНИЕ

Подробная информация об интеграции ADQM с внешними источниками данных через интерфейс JDBC приведена в статье JDBC Bridge.

Параметр	Описание	Значение по умолчанию
jdbc bridge log path	Путь к каталогу, содержащему лог-файлы	/var/log/clickhouse-jdbc-bridge/
datasource.json	Источник данных для JDBC Bridge	

```
[
    {
        "$schema": "./datasource.jschema",
        "postgres13": {
            "converter": {
                "mappings": [
                    {
                        "nativeType": "bool",
                        "toType": "String"
                    }
                ]
            },
            "driverUrls": [
                "drivers/postgresql/postgresql-jdbc.jar"
            ],
            "driverClassName": "org.postgresql.Driver",
            "jdbcUrl": "jdbc:postgresql://localhost/test",
            "username": "",
            "password": "",
            "initializationFailTimeout": 0,
            "minimumIdle": 0,
            "maximumPoolSize": 10
        }
    },
    {
        "$schema": "./datasource.jschema",
        "ch-server": {
            "driverUrls": [
                "drivers/clickhouse/clickhouse-jdbc.jar"
            ],
            "driverClassName": "ru.yandex.clickhouse.ClickHouseDriver",
            "jdbcUrl": "jdbc:clickhouse://ch-server:8123/system",
            "username": "default",
            "password": "",
            "initializationFailTimeout": 0,
            "minimumIdle": 0,
            "maximumPoolSize": 10
        }
    }
]
```

Параметр	Описание	Значение по умолчанию
server.json	Конфигурация сервера для JDBC Bridge	'{\n \"requestTimeout\": 5000,\n \"queryTimeout\": 60000,\n \"configScanPeriod\": 5000,\n \"repositories\": [\n {\n \"entity\": \"ru.yandex.clickhouse.jdbcbridge.core.NamedDataSource\",\n \"repository\": \"ru.yandex.clickhouse.jdbcbridge.impl.JsonFileRepository\"\n },\n {\n \"entity\": \"ru.yandex.clickhouse.jdbcbridge.core.NamedSchema\",\n \"repository\": \"ru.yandex.clickhouse.jdbcbridge.impl.JsonFileRepository\"\n },\n {\n \"entity\": \"ru.yandex.clickhouse.jdbcbridge.core.NamedQuery\",\n \"repository\": \"ru.yandex.clickhouse.jdbcbridge.impl.JsonFileRepository\"\n }\n],\n \"extensions\": [\n {\n \"class\": \"ru.yandex.clickhouse.jdbcbridge.impl.JdbcDataSource\"\n },\n {\n \"class\": \"ru.yandex.clickhouse.jdbcbridge.impl.ConfigDataSource\"\n },\n {\n \"class\": \"ru.yandex.clickhouse.jdbcbridge.impl.ScriptDataSource\"\n }\n]\n}\n'

Encryption

ПРИМЕЧАНИЕ

В статье Кодеки шифрования приведен пример, как настроить и применить шифрование данных на основе алгоритма AES-256.

Параметр	Описание	Значение по умолчанию
Credentials encrypt	Алгоритм шифрования конфиденциальных данных в файлах конфигурации Clickhouse. Возможные значения: ▪ `AES-128`; ▪ `AES-256`; ▪ `none` — шифрование не будет применено, и конфиденциальные данные останутся незашифрованными.	none
aes_128_source	Источник значений параметров шифрования для алгоритма AES-128. Возможные значения: ▪ `from_env` — переменная окружения; ▪ `from_zk` — узел ZooKeeper; ▪ `none` — значения, указанные в конфигурации.	none
aes_256_source	Источник значений параметров шифрования для алгоритма AES-256. Возможные значения: ▪ `from_env` — переменная окружения; ▪ `from_zk` — узел ZooKeeper; ▪ `none` — значения, указанные в конфигурации.	none
aes_128_key_env	Переменная окружения, которая содержит ключ для алгоритма шифрования AES-128	—
aes_128_nonce_env	Переменная окружения, которая содержит случайный код (nonce) для алгоритма шифрования AES-128	—
aes_128_key_zk_path	Путь в ZooKeeper к ключу для алгоритма шифрования AES-128	—
aes_128_nonce_zk_path	Путь в ZooKeeper к случайному коду (nonce) для алгоритма шифрования AES-128	—
aes_256_key_env	Переменная окружения, которая содержит ключ для алгоритма шифрования AES-256	—
aes_256_nonce_env	Переменная окружения, которая содержит случайный код (nonce) для алгоритма шифрования AES-256	—
aes_256_key_zk_path	Путь в ZooKeeper к ключу для алгоритма шифрования AES-256	—
aes_256_nonce_zk_path	Путь в ZooKeeper к случайному коду (nonce) для алгоритма шифрования AES-256	—
AES-128 GCM-SIV key	Ключ для алгоритма шифрования AES-128 в виде 32-символьной шестнадцатеричной строки. Например, `00112233445566778899AABBCCDDEEFF`	—
AES-128 GCM-SIV nonce	Случайный код (nonce) для алгоритма шифрования AES-128 в виде 24-символьной шестнадцатеричной строки. Например, `ABCDEFABCDEFABCDEFABCDEF`	—
AES-256 GCM-SIV key	Ключ для алгоритма шифрования AES-256 в виде 64-символьной шестнадцатеричной строки. Например, `00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF`	—
AES-256 GCM-SIV nonce	Случайный код (nonce) для алгоритма шифрования AES-256 в виде 24-символьной шестнадцатеричной строки. Например, `ABCDEFABCDEFABCDEFABCDEF`	—

ВНИМАНИЕ

ZooKeeper с включенным SSL нельзя использовать для обеспечения работы `encrypt_decrypt` на данный момент.

Other

Параметр	Описание	Значение по умолчанию
Time zone	Часовой пояс, используемый для преобразований между типами данных String и DateTime (например, когда значения полей типа DateTime выводятся в текстовом формате или при получении значений типа DateTime из строк). Если значение параметра установлено в `System`, будет использоваться значение параметра `timezone`, установленное в конфигурации сервера ADQM или в настройках операционной системы на момент запуска сервера	System
Default database	База данных по умолчанию	default
Default profile	Профиль настроек по умолчанию. Профили настроек находятся в файле, указанном в параметре `user_config`	default
Auto Core Dump Service	Указывает, включен ли автоматический дамп памяти. Включение/выключение автоматического дампа памяти управляется выполнением соответствующих действий для сервиса ADQMDB	Неактивен

Chproxy

Network configuration

Параметр	Описание	Значение по умолчанию
Enable HTTP	Включает прослушивание HTTP-запросов	Активен
HTTP listen port	Порт, на котором прослушиваются HTTP-запросы	9090
HTTP allowed networks	Список сетей, из которых разрешен доступ по HTTP. Каждый элемент может содержать IP-адрес или маску подсети. Значение `127.0.0.1` добавлено в конфигурацию по умолчанию. Chproxy пытается обнаружить самые очевидные ошибки конфигурации, такие как `allowed_networks: ["0.0.0.0/0"]`	—
Enable HTTPS	Включает прослушивание HTTPS-запросов	Неактивен
HTTPS listen port	Порт, на котором прослушиваются HTTPS-запросы	8543
HTTPS allowed networks	Список сетей, из которых разрешен доступ по HTTPS. Каждый элемент может содержать IP-адрес или маску подсети. Значение `127.0.0.1` добавлено в конфигурацию по умолчанию. Chproxy пытается обнаружить самые очевидные ошибки конфигурации, такие как `allowed_networks: ["0.0.0.0/0"]`	—
Certificate file	Путь к файлу серверного SSL-сертификата в формате PEM	/etc/chproxy/server.crt
Private key file	Путь к файлу закрытого ключа серверного SSL-сертификата	/etc/chproxy/server.key
Use HTTP to connect to ADQMDB	Указывает, какой протокол (HTTP или HTTPS) Chproxy будет использовать для подключения к ClickHouse	Активен

Proxy users configuration

ПРИМЕЧАНИЕ

Значения по умолчанию в таблице ниже указаны для пользователя, добавленного в секцию ***Clickhouse Proxy Users*** по умолчанию. При добавлении нового пользователя все поля конфигурации изначально пустые, обязательные для заполнения поля подсвечиваются красным.

Параметр	Описание	Значение по умолчанию
deny_http	Указывает, запрещены ли HTTP-подключения для пользователя	Неактивен
user_name	Имя пользователя ADQM или, если установлен флаг `wildcarded_user`, имя wildcarded-пользователя, которое представляет собой шаблон вида `<prefix>*` или `*<suffix>`	default
allow_cors	Указывает, разрешены ли CORS-запросы (требуются для таких инструментов как Tabix)	Неактивен
cache_name	Имя кеша ответов (эта настройка включает кеширование ответов, отключенное по умолчанию)	—
deny_https	Указывает, запрещены ли HTTPS-подключения для пользователя	Неактивен
max_queue_size	Максимальное количество запросов, ожидающих выполнения в очереди (`0` — запросы выполняются без ожидания в очереди)	0
max_queue_time	Максимальное время ожидания запроса в очереди	10s
wildcarded_user	Индикатор конфигурации wildcarded-пользователя. Если флаг установлен, в качестве значения параметра `user_name` должен быть указан шаблон имени пользователя вида `<prefix>*` или `*<suffix>` (например, `dba_*` или `*-analyst`). В этом случае любой пользователь с именем, соответствующим этому шаблону, будет авторизовываться в ADQM как пользователь, указанный в `clickhouse_target_user`	Неактивен
allowed_networks	Список сетей, из которых разрешен доступ (элементом списка может быть IP-адрес или маска подсети)	—
max_execution_time	Максимальная длительность выполнения запроса пользователя	120s
requests_per_minute	Максимальное количество запросов от пользователя в минуту (`0` — нет ограничения; отрицательное значение фактически заблокирует пользователя)	0
clickhouse_target_user	Пользователь ADQM, учетные данные которого будут использоваться для проксирования запросов пользователей, соответствующих шаблону wildcarded-пользователя (если флаг `wildcarded_user` не установлен, этот параметр не используется)	default
max_concurrent_queries	Максимальное количество одновременно выполняемых запросов пользователя	0
Users/password map	Пароли пользователей, указанных в секции ***Proxy users configuration***	—

Connection pool configuration

Параметр	Описание	Значение по умолчанию
Max idle connections	Максимальное количество неиспользуемых соединений в пуле	100
Max idle connections per host	Максимальное количество неиспользуемых соединений на хост	2

ADQM cluster configuration

Параметр	Описание	Значение по умолчанию
Interval	Интервал проверки всех узлов кластера на доступность	5s
Timeout	Тайм-аут ожидания ответа от узлов кластера	5s

Zookeeper

Main

Параметр	Описание	Значение по умолчанию
connect	Строка подключения ZooKeeper, которая будет использоваться другими службами или кластерами. Генерируется автоматически	—
dataDir	Место, где ZooKeeper будет хранить снепшоты резидентной базы данных (in-memory database) и, если не указано иное, логи транзакций обновлений базы данных. Указывается как параметр `dataDir` в конфигурационном файле *zoo.cfg*	/var/lib/zookeeper
admin.serverPort	Порт AdminServer. Встроенный сервер Jetty, который предоставляет интерфейс HTTP для команд из четырехбуквенных слов	58080

zoo.cfg

Параметр	Описание	Значение по умолчанию
clientPort	Порт для прослушивания клиентских подключений, т.е. порт, к которому пытаются подключиться клиенты	2181
tickTime	Продолжительность одного тика (в миллисекундах), который является основной единицей времени, используемой ZooKeeper	2000
initLimit	Тайм-ауты (в тиках), которые ZooKeeper использует для ограничения времени, в течение которого серверы-последователи должны подключиться и синхронизироваться с лидером. Увеличивайте это значение по мере необходимости, если объем данных, которыми управляет ZooKeeper, большой	5
syncLimit	Количество времени (в тиках), в течение которого сервер должен синхронизироваться с ZooKeeper	2
maxClientCnxns	Максимальное количество активных подключений с одного хоста, указанного по IP-адресу, к одному узлу ансамбля ZooKeeper	0
autopurge.snapRetainCount	Когда функция автоматической очистки ZooKeeper включена, она сохраняет указанное количество последних снепшотов и соответствующие им логи транзакций в каталоги *dataDir* и *dataLogDir* соответственно, а остальные удаляет. Минимальное значение `3`	3
autopurge.purgeInterval	Временной интервал (в часах), для которого должна быть запущена задача очистки. Установите целое положительное число, чтобы включить автоматическую очистку	24
Add key,value	Используйте это поле, чтобы задать дополнительные параметры, которые не отображаются в пользовательском интерфейсе ADCM, но разрешены в файле конфигурации *zoo.cfg*	—

SSL configuration

Параметр	Описание	Значение по умолчанию
Enable SSL	Включает использование безопасных входящих соединений в ZooKeeper	Неактивен
sslQuorum	Активирует зашифрованную кворумную коммуникацию	Неактивен
serverCnxnFactory	Имя класса, реализующего `ServerCnxnFactory`. Чтобы использовать серверную коммуникацию на основе TLS, установите значение параметра `NettyServerCnxnFactory`	org.apache.zookeeper.server.NettyServerCnxnFactory
ssl.quorum.keyStore.location	Полный путь к серверному keystore-файлу (например: */etc/zookeeper/ssl/keystore.jks*)	/etc/zookeeper/server.jks
ssl.quorum.keyStore.password	Пароль, используемый при создании keystore-файла, указанного в `ssl.quorum.keyStore.location`	—
ssl.quorum.trustStore.location	Полный путь к серверному truststore-файлу (например: */etc/zookeeper/ssl/truststore.jks*)	/etc/zookeeper/ca.jks
ssl.quorum.trustStore.password	Пароль, используемый при создании truststore-файла, указанного в `ssl.quorum.trustStore.location`	—
ssl.keyStore.location	Полный путь к серверному keystore-файлу, который будет использоваться для клиентских TLS-подключений (например: */etc/zookeeper/ssl/keystore.jks*)	/etc/zookeeper/server.jks
ssl.keyStore.password	Пароль, используемый при создании keystore-файла, указанного в `ssl.keyStore.location`	—
ssl.trustStore.location	Полный путь к серверному truststore-файлу, который будет использоваться для клиентских TLS-подключений (например: */etc/zookeeper/ssl/truststore.jks*)	/etc/zookeeper/ca.jks
ssl.trustStore.password	Пароль, используемый при создании truststore-файла, указанного в `ssl.trustStore.location`	—
ssl.protocol	Протокол, используемый при TLS-рукопожатии в клиенте	TLSv1.2
ssl.quorum.protocol	Протокол, используемый при TLS-рукопожатии в кворуме	TLSv1.2
secureClientPort	Порт для защищенных клиентских соединений	2281

Metrics

Параметр	Описание	Значение по умолчанию
Listen port	Порт доступа к метрикам сервиса Zookeeper в формате Prometheus	9020

zookeeper-env.sh

Параметр	Описание	Значение по умолчанию
ZOO_LOG_DIR	Каталог для хранения логов	/var/log/zookeeper
ZOOPIDFILE	Каталог для хранения ID процесса ZooKeeper	/var/run/zookeeper/zookeeper_server.pid
SERVER_JVMFLAGS	Используется для установки различных параметров JVM, связанных, например, со сборкой мусора	-Xmx1024m
JAVA	Путь к Java	$JAVA_HOME/bin/java
ZOO_LOG4J_PROP	Используется для установки уровня ведения журнала log4j и указания того, какие приложения журнала следует включить. Эффект от включения приложения журнала `CONSOLE` заключается в том, что журналы отправляются на стандартный вывод. Эффект от включения `ROLLINGFILE` заключается в том, что файл *zookeeper.log* создается, ротируется и истекает	INFO, CONSOLE, ROLLINGFILE

Clickhousekeeper

General settings

Параметр	Описание	Значение по умолчанию
Listen host	Ограничение по хостам, с которых могут приходить запросы. Учитывайте возможное отсутствие IPv6 на целевом хосте и используйте, например, `0.0.0.0` в этом случае	::
Enable IPv6	Позволяет отключить IPv6, если хосты не поддерживают этот протокол	Активен
TCP port	Порт для подключения клиента	2129
Log storage path	Путь к журналам координации	/var/lib/clickhouse-keeper/coordination/log
Snapshot storage path	Путь к снепшотам координации	/var/lib/clickhouse-keeper/coordination/snapshots
root_path	Узел znode, который используется в качестве корня для узлов znode, используемых сервером ClickHouse	/clickhouse
Distributed DDL path	ZK-путь, по которому ClickHouse отправляет DDL-задачи. В противном случае убедитесь, что в конфигурации нет пересечений с другими кластерами	/clickhouse/task_queue/ddl

Coordination settings

Параметр	Описание	Значение по умолчанию
Operation timeout ms	Тайм-аут для одной операции клиента (в миллисекундах)	10000
Session timeout ms	Тайм-аут для сеанса клиента (в миллисекундах)	30000
Snapshot distance	Частота, с которой ClickHouse Keeper будет делать новые снепшоты (по количеству записей в журналах)	100000
Raft logs level	Уровень логирования процесса координации (`trace`, `debug` и т.д.)	error
Rotate log storage interval	Количество записей в журнале для хранения в одном файле	10000

Raft settings

Параметр	Описание	Значение по умолчанию
Port	Порт, на котором сервер прослушивает соединения RAFT	9444

Metrics

Параметр	Описание	Значение по умолчанию
Listen port	Порт доступа к метрикам сервиса Clickhousekeeper в формате Prometheus	9010
Endpoint	Конечная точка HTTP, в которую экспортируются метрики сервиса Clickhousekeeper	/metrics

Logger options

Параметр	Описание	Значение по умолчанию
Log level	Уровень логирования (`trace`, `debug`, `information`, `warning` или `error`)	Trace
Log path	Путь к каталогу, содержащему лог-файлы	/var/log/clickhouse-keeper/
Log file size	Размер лог-файла (в МБ). Как только файл достигает указанного размера, он архивируется и переименовывается, а на его месте создается новый лог-файл	1000
Log file count	Количество архивных лог-файлов, которые хранит ClickHouse	5

Monitoring

Prometheus settings

Группа	Параметр	Описание	Значение по умолчанию
—	scrape_interval	Интервал сбора метрик	1m
—	scrape_timeout	Время ожидания данных	10s
—	Password for grafana connection	Пароль пользователя Grafana (`admprom_grafana`) для подключения к Prometheus	—
—	Prometheus users to login/logout to Prometheus	Учетные данные пользователей для входа в веб-интерфейс Prometheus	—
Service parameters	config.file	Путь к конфигурационному файлу сервера Prometheus	/etc/admprom/prometheus/prometheus.yml
	storage.tsdb.path	Путь к базе данных сервера Prometheus	/var/lib/admprom/prometheus
	web.console.libraries	Путь к библиотекам управления консолями	/usr/share/admprom/prometheus/console_libraries
	web.console.templates	Путь к шаблонам консолей сервера Prometheus	/usr/share/admprom/prometheus/consoles
	web.config.file	Указывает, какой файл веб-конфигурации следует загрузить. Файл записывается в YAML-формате	/etc/admprom/prometheus/prometheus-auth.yml
	storage.tsdb.retention.time	Время хранения данных. Поддерживаемые единицы: `y`, `w`, `d`, `h`, `m`, `s`, `ms`	15d
	web.listen-address	Адрес для доступа к веб-интерфейсу Prometheus	0.0.0.0:11200

Grafana settings

Параметр	Описание	Значение по умолчанию
Grafana administrator's password	Пароль администратора Grafana	—
Grafana listen port	Порт доступа к web-интерфейсу Grafana	11210

Pushgateway settings

Параметр	Описание	Значение по умолчанию
listen_address	Адрес для доступа к Pushgateway Web API	0.0.0.0:9091

Node Exporter settings

Параметр	Описание	Значение по умолчанию
Listen port	Порт доступа к системным метрикам хоста ADQM в формате Prometheus	11203
Metrics endpoint	Конечная точка, в которую node exporter экспортирует системные метрики в формате Prometheus	/metrics

SSL configuration

Параметр	Описание	Значение по умолчанию
[Prometheus] → Enable SSL	Включает SSL-шифрование входящего трафика для компонента Prometheus Server	Неактивен
[Prometheus] → Certificate file	Путь к файлу серверного SSL-сертификата для компонента Prometheus Server	/etc/admprom/prometheus/server.crt
[Prometheus] → Private key file	Путь к файлу закрытого ключа серверного SSL-сертификата для компонента Prometheus Server	/etc/admprom/prometheus/server.key
[Prometheus] → Certificate authority file	Путь к файлу корневого сертификата центра сертификации для проверки ресурсов (target), с которых поступают метрики для мониторинга в Prometheus	/etc/admprom/prometheus/ca.crt
[Grafana] → Enable SSL	Включает SSL-шифрование входящего трафика для компонента Grafana	Неактивен
[Grafana] → Certificate file	Путь к файлу серверного SSL-сертификата для компонента Grafana	/etc/admprom/grafana/server.crt
[Grafana] → Private key file	Путь к файлу закрытого ключа серверного SSL-сертификата для компонента Grafana	/etc/admprom/grafana/server.key
[Grafana] → Certificate authority file	Путь к файлу корневого сертификата центра сертификации для проверки сертификата Prometheus	/etc/admprom/grafana/ca.crt
[Node-exporter] → Enable SSL	Включает SSL-шифрование входящего трафика для компонента Node Exporter	Неактивен
[Node-exporter] → Certificate file	Путь к файлу серверного SSL-сертификата для компонента Node Exporter	/etc/ssl/server.crt
[Node-exporter] → Private key file	Путь к файлу закрытого ключа серверного SSL-сертификата для компонента Node Exporter	/etc/ssl/server.key
[Pushgateway] → Enable SSL	Включает SSL-шифрование входящего трафика для компонента Pushgateway	Неактивен
[Pushgateway] → Certificate file	Путь к файлу серверного SSL-сертификата для компонента Pushgateway	/etc/admprom/pushgateway/server.crt
[Pushgateway] → Private key file	Путь к файлу закрытого ключа серверного SSL-сертификата для компонента Pushgateway	/etc/admprom/pushgateway/server.key

Релизы ADQM

Елена Дворядкина

Содержание

Назад к содержанию

25.8.7.3
25.3.2.39
24.8.8.17
24.3.11.7
24.3.2.23
23.8.2.7
23.3.2.37
22.8.12.45
22.8.6.71
22.8.4.7
22.3.7.28
22.3.2.2
21.8.13.6
21.8.8.29
21.3.14.1
21.3.12.2
21.3.11.5
21.3.4.25
20.8.13.15
20.8.11.17
20.8.4.11
20.3.20.6
20.8.2.3
20.3.13.130
20.3.10.75
19.14.11.16

25.8.7.3

25.8.7.3.1.b1

Дата: 26.11.2025

Новые функции

ClickHouse обновлен до версии 25.8.7.3

В секцию ***Advanced configuration parameters*** конфигурации сервиса ADQMDB добавлено поле ***Advanced custom XML***, XML-значение которого проверяется на стадии precheck и добавляется в основной конфигурационный файл *config.xml*

Включена динамическая реконфигурация ZooKeeper для расширения/сокращения ансамблей без перезапуска сервиса

Добавлена возможность включения SSL для компонентов сервиса Monitoring через интерфейс Manage SSL

Добавлена возможность перезагрузки конфигурации без перезапуска сервиса для следующих сервисов:

- ADQMDB (в том числе, если установлен компонент Clickhouse JDBC Bridge);
- Clickhousekeeper;
- Chproxy.

Сервисное действие ***Reconfig and restart*** заменено действием ***Reconfig*** с опциональным флагом ***Restart*** в окне конфигурирования действия

Улучшения

Реализована возможность удалять сервисы в состоянии `created` или `installed`, компоненты которых добавлены на хосты, через иконку 🗑 на странице Services в интерфейсе ADCM

Добавлен параметр верхнего уровня ***Coordination system*** вместо группы ***Engine*** с одним параметром ***System*** на странице конфигурации сервиса ADQMDB

Реализована поддержка SSL, и добавлены соответствующие настройки для компонента Pushgateway сервиса Monitoring

Улучшена обработка пакетов JDBC bridge во время обновления и отмены обновления, включая остановку сервиса, очистку и автотесты понижения версии

Выполнен рефакторинг конфигурационных параметров сервиса Chproxy:

- добавлены группы параметров ***Proxy users configuration*** и ***Connection pool configuration***;
- удалена группа параметров ***Proxy wildcarded users***;
- реализована логика преобразования и валидации параметров при обновлении версии бандла;
- пароли пользователей Chproxy защищены с помощью типа данных secretmap.

Реализовано автоматическое удаление неиспользуемых источников данных из конфигурации JDBC bridge

Добавлено удаление statuschecker при выполнении операций удаления сервисных компонентов с хостов

25.3.2.39

25.3.2.39.1.b2

Дата: 08.10.2025

Улучшения

Улучшена обработка состояния сервиса Monitoring Clients за счет настройки состояния сервиса до и после переключения бандла

Добавлена проверка наличия каталога */var/lib/clickhouse* и настройка разрешений для него на этапе установки

Сервисы в режиме обслуживания (maintenance mode) теперь гарантированно остаются в targets конфигурации Prometheus только при доступности хоста

Скорректировано поведение, когда все хосты ADQMDB или ZooKeeper находятся в режиме обслуживания

Обновление кластера теперь завершается ошибкой (корректное поведение), если какой-либо хост кластера находится в режиме обслуживания

Исправленные баги

Исправлена ошибка отсутствия атрибута `ADQM_arenadata_enterprise`, возникавшая при выполнении upgrade-действий для версии Community

Исправлена ошибка установки allocated ClickHouse Keeper

Исправлена ошибка, возникавшая при выполнении действия ***Add hosts*** для Chproxy, когда все хосты ADQMDB в режиме обслуживания и недоступны

Исправлена ошибка выполнения действия ***Reconfig and restart*** сервиса ADQMDB, возникавшая когда все хосты какого-либо шарда в режиме обслуживания

25.3.2.39.1.b1

Дата: 19.08.2025

Новые функции

Реализован режим обслуживания (maintenance mode) в ADQM

Добавлена поддержка Alt Linux 10.2

Добавлена поддержка Astra Linux 1.7.6 Воронеж

ClickHouse обновлен до версии 25.3.2.39

Обновлен пакет clickhouse-copier — теперь он поддерживает все схемы ADQMDB версии 25.3.2.39

Реализовано действие ***Add/Remove components*** для всех сервисов ADQM (вместо действий ***Add Server(s)***, ***Remove Server(s)***, ***Add JDBC Bridge***, ***Remove JDBC Bridge***, ***Shrink*** и ***Expand***)

Добавлены пакеты jdbc-mysql-connector и jdbc-postgresql в бандл ADQM

На страницу ***Configuration*** сервиса ADQMDB в секцию ***Log settings*** снова добавлен параметр ***session_log*** после исправлений багов в ClickHouse upstream

Улучшения

Полные доменные имена хостов (FQDN) теперь используются при установке Zookeeper

Полные доменные имена хостов теперь используются в разделе `remote_servers` файла */etc/clickhouse-server/config.xml*

Пароль пользователя `default` теперь должен иметь длину не менее 8 символов и не должен содержать специальных символов

Добавлена поддержка HTTPS для Enterprise Tools

Прочее

Удален устаревший сервис мониторинга на основе Diamond

24.8.8.17

24.8.8.17.2.b1

Дата: 24.03.2025

Новые функции

Добавлено кластерное действие Manage SSL

В Chproxy реализована возможность безопасного подключения к ADQMDB

ClickHouse обновлен до версии 24.8.8.17

Группа параметров ***Encryption codecs for tables*** конфигурации сервиса ADQMDB расширена и переименована в ***Encryption***

Добавлено шифрование паролей

Улучшения

В конфигурации сервисов ADQMDB, Clickhousekeeper и Zookeeper добавлена активируемая группа параметров ***Metrics***

Удалены опции автоматического перезапуска сервисов ADQM из конфигурации действия ***Reconfigure and restart*** сервиса Monitoring

Реализовано автоматическое включение группы параметров ***Metrics*** для сервисов ADQM при установке Prometheus

Реализовано размещение node exporter на всех хостах после обновления

Добавлена проверка шаблона в поле ***Listen host*** конфигурации сервиса Clickhousekeeper

Пароль для пользователя по умолчанию удален из всех конфигурационных файлов `clickhouse-client`

Опции из секции ***Reliability control*** конфигурации кластера используются для повторной попытки выполнения задачи `Create Clickhouse znode`

В конфигурационном файле Chproxy используется FQDN

HTML-блок `logo-container` удален для playground (`/play`) в ADQMDB

Исправленные баги

Исправлен шаблон для ***Proxy wildcarded users*** сервиса Chproxy

Обновлены настройки по умолчанию и разрешения для SSL в ADQM

Внесены исправления в работу службы проверки состояния (statuschecker)

Выполнен рефакторинг сообщения об ошибке на вкладке ***Ansible [check]*** при проверке Chproxy

Прочее

Минимальная версия ADQM для обновления увеличена до 23.8.2.7.1.b1

24.3.11.7

24.3.11.7.1.b1

Дата: 17.10.2024

Новые функции

ClickHouse обновлен до версии 24.3.11.7-lts

Исправленные баги

Отмена обновления кластера (действие ***Abort the upgrade***) теперь включает шаг удаления пакетов ADQM, если они были обновлены

24.3.2.23

24.3.2.23.2.b1

Дата: 02.08.2024

Новые функции

Добавлена поддержка операционной системы Ubuntu

В настройки конфигурации кластера добавлена секция ***Custom JRE***, где можно устанавливать пользовательское значение переменной `JAVA_HOME`

Включена установка кастомного пакета JRE на Astra Linux

Добавлена возможность настройки нескольких серверов LDAP

Реализована функциональность rolling upgrade (последовательное обновление) для ADQMDB начиная с версии ADCM 2.3

Реализована функциональность rolling restart (последовательный перезапуск) для ADQMDB начиная с версии ADCM 2.3

Добавлен параметр ***retention_time*** в секцию ***Prometheus settings*** на странице конфигурации сервиса Monitoring в ADCM

Улучшения

Задачи Prometheus перемещены на уровень кластера

Исправленные баги

Исправлена ошибка действия ***Manage Kerberos***, возникавшая при использовании специальных символов в пароле ***Kadmin password***

Исправлена ошибка удаления сервиса Monitoring после добавления пустого хоста

Прочее

Минимальная версия ADQM для обновления увеличена до 23.3.2.37.1.b1

Минимальная версия ADCM увеличена до 2.2.0

Удалена проверка Baje

24.3.2.23.1.b1

Дата: 25.04.2024

Новые функции

ClickHouse обновлен до версии 24.3.2.23-lts

Минимальная версия ADCM теперь 2023.04.21.12

Добавлен пакет adqm-clickhouse-copier для копирования между кластерами и перешардирования данных (замена clickhouse-copier)

Вывод лога в ADCM теперь доступен для незапустившегося сервиса

Реализована поддержка SSL между ZooKeeper и ADQMDB

Улучшения

Старые действия для управления Kerberos (***Enable/Disable/Reconfig Kerberos***) заменены новым действием ***Manage Kerberos***

Добавлены параметры LDAP для указания `bind_dn` и `user_dn_detection`

Добавлена возможность удаления сервиса Zookeeper через иконку 🗑 на странице Services

Реализована функциональность для удаления legacy-мониторинга

Добавлен параметр ***asynchronous_insert_log*** в секцию ***Log settings*** на странице конфигурации сервиса ADQMDB в ADCM

Добавлена переустановка statuschecker для сервисов Chproxy и Clickhousekeeper

Исправленные баги

Исправлена ошибка, связанная с атрибутом `fqdn`, возникавшая при установке ADQM с ClickHouse Keeper на выделенном хосте

Прочее

Реализована проверка хостов в структуре топологии кластера

23.8.2.7

23.8.2.7.1.b1

Дата: 29.12.2023

Новые функции

ClickHouse обновлен до версии 23.8.2.7-lts

ZooKeeper обновлен до версии 3.8.3

Добавлен сервис мониторинга на основе Prometheus — рекомендуемый способ установки мониторинга ADQM (поддержка кластера мониторинга будет прекращена в будущих релизах ADQM)

Реализованы новые дашборды Grafana, в том числе добавлен дашборд General

Добавлен параметр ***jdbc bridge log path*** в секцию ***JDBC Bridge configuration*** на странице конфигурации сервиса ADQMDB в ADCM

Реализована поддержка сертификатов CA для межсерверного взаимодействия. Добавлен параметр ***Enable secure intra-cluster communication*** (вместо ***Interserver protocol***) в секцию ***Intra-cluster communication security*** конфигурационных параметров сервиса ADQMDB

Ошибки ClickHouse теперь отображаются в задачах (jobs) ADQM

Улучшения

Настройки хранилища, указанные в ADCM, теперь полностью синхронизируются с файлом *storage.xml*

Реализовано автоматическое создание необходимых директорий для путей к локальным дискам

Добавлена проверка пути к диску для предотвращения конфликтов директорий

Повышена надежность управления шардами и репликами

Прочее

В действие ***Reconfig and restart*** сервиса ADQMDB добавлена проверка наличия неопределенных политик хранения

Для действия ***Reconfig and restart*** сервиса ADQMDB реализована опция ***Disable validation***, позволяющая отключить проверку корректности конфигурации

Создание кеша поверх локальных дисков теперь не поддерживается

23.3.2.37

23.3.2.37.1.b1

Дата: 21.07.2023

Новые функции

ClickHouse обновлен до версии 23.3.2.37-lts

Реализована поддержка сетевого интерфейса gRPC. На страницу ***Configuration*** сервиса ADQMDB добавлена секция ***Enable gRPC***

Расширен набор опций для конфигурирования хранилища данных. Добавлены возможности:

- хранение данных на внешних дисках S3;
- создание кеширующих дисков поверх локальных и S3-дисков;
- создание зашифрованных дисков поверх локальных, S3- и кеширующих дисков.

На странице конфигурации сервиса ADQMDB вместо параметра ***Storage Configuration*** добавлены новые секции: ***Enable additional local storage***, ***Enable S3 storage***, ***Enable local cache***, ***Enable encrypted storage*** и ***Storage policies***.

Реализована возможность определения требований к сложности паролей пользователей. На странице конфигурации сервиса ADQMDB в секцию ***User password options*** добавлены параметры: ***Allow Implicit_NO_password***, ***Length of password***, ***Include Upper-case***, ***Include Lower-case***, ***Include Number***, ***Include special symbols***

В параметры конфигурации сервиса ADQMDB добавлена секция ***Encryption codecs for tables***, где можно указывать ключи для кодеков шифрования таблиц

Реализована поддержка РЕД ОС в Enterprise-версии ADQM

Улучшения

Улучшена безопасность каталогов и файлов конфигурации ClickHouse

Прочее

Добавлена проверка наличия в политике хранения хотя бы одного тома

Выполнен рефакторинг в следующих секциях на странице конфигурации сервиса ADQMDB:

- ***Cluster configuration*** — удалены параметры реплик ***port***, ***user***, ***password*** (вместо них используются параметры секции ***Intra-cluster communication security***);
- ***User password options*** — параметр ***Access Management*** переименован в ***Allow plaintext password***.

22.8.12.45

22.8.12.45.1.b1

Дата: 18.04.2023

Новые функции

ClickHouse обновлен до версии 22.8.12.45-lts

Добавлена поддержка Astra Linux для Enterprise-версии ADQM

Реализована возможность настройки аутентификации в ADQM по протоколу Kerberos через интерфейс ADCM

Реализована возможность установки ClickHouse Keeper как отдельного сервиса

Реализована поддержка логирования ClickHouse Keeper (лог-файл сохраняется в директорию */var/log/chkeeper*)

Реализована поддержка SSL в сервисе Chproxy

Добавлены новые параметры на страницу ***Configuration*** сервиса ADQMDB:

- параметр ***Enable IPv6*** в секцию ***Clickhouse Keeper (integrated)*** (ранее ***Clickhouse Keeper***);
- параметры ***Max server memory usage*** и ***Max thread pool size*** в секцию ***Limits***;
- параметры ***Interserver protocol*** и ***Interserver protocol's port*** в секцию ***Intra-cluster communication security***;
- параметры ***Logging level***, ***Log file size***, ***Log files count***, ***metric_log*** и ***asynchronous_metric_log*** в секцию ***Log settings***;
- параметр ***Time zone*** в секцию ***Other***;
- секция ***User password options***;
- секция ***Data compression***;
- секция ***Kerberos configuration***.

Улучшения

Добавлены значения `relaxed` и `once` для параметра `verificationMode` и дополнительные проверки сертификатов в зависимости от значений этого параметра

Добавлены новые параметры в конфигурационный файл клиента для успешной работы при включенных SSL/TCP

Добавлена возможность передавать в качестве значения параметра ***Max concurrent threads*** целое число, чтобы указать максимальное количество потоков обработки запроса

Прочее

На странице ***Configuration*** сервиса ADQMDB удален параметр ***session_log*** из секции ***Log settings***

22.8.6.71

22.8.6.71.1.b1

Дата: 17.01.2023

Новые функции

ClickHouse обновлен до версии 22.8.6.71-lts

Для сервиса ADQMDB в ADCM реализованы действия ***Enable auto core dump*** и ***Disable auto core dump***

Добавлены новые параметры на страницу ***Configuration*** сервиса ADQMDB:

- параметр ***Default user password*** в секцию ***Default user and policy settings***;
- параметр ***Log path*** в секцию ***Directories***;
- параметр ***Max concurrent threads*** в секцию ***Limits***;
- параметр ***Auto Core Dump Service*** в секцию ***Other***;
- секция ***Intra-cluster communication security***.

Улучшения

Выполнен ряд улучшений, повышающих безопасность в части взаимодействия хостов кластера, поддержки SSL, взаимодействия Chproxy и ClickHouse

Прочее

Выполнен рефакторинг страницы ***Configuration*** сервиса ADQMDB: улучшены описания некоторых параметров, удалены параметры ***external_dictionaries_path*** и ***external_dictionaries*** из секции ***Other***, секция ***Other*** перенесена в конец списка

Расширен набор действий, выполняемых для предварительной проверки состояния хостов, компонентов и сервисов перед установкой кластера ADQM

22.8.4.7

22.8.4.7.1.b1

Дата: 05.10.2022

Новые функции

Реализован параметр `concurrent_threads_soft_limit`, который позволяет ограничить количество потоков обработки запроса

На страницу конфигурации кластера в интерфейсе ADCM добавлены опции ***Disable SElinux*** и ***Disable Firewalld***

Прочее

Сервис Baje proxy удален из ADQM

Добавлена проверка версии Alt Linux перед установкой ADQM

22.3.7.28

22.3.7.28.1.b1

Дата: 12.08.2022

Новые функции

Добавлен сервис Chproxy

Исправленные баги

Исправлена ошибка при обновлении ADQM на Alt Linux

Исправлено некорректное переключение статуса кластера с `upgradable` на `installed` во время обновления версии кластера

Прочее

Пакеты ADQM теперь имеют названия, отличные от названий пакетов ClickHouse

22.3.2.2

22.3.2.2.1.b1

Дата: 26.05.2022

Новые функции

ClickHouse обновлен до версии 22.3.2.2-lts

Добавлена поддержка Alt Linux 8.4 SP

Добавлен компонент ClickHouse Keeper

Расширены возможности управления встроенными средствами логирования

Улучшения

Добавлена проверка сервисов мониторинга, установленных в кластер Enterprise Tools, перед импортом настроек мониторинга в кластер ADQM

21.8.13.6

21.8.13.6.1.b2

Дата: 17.01.2023

Исправленные баги

Исправлено: возможная потеря данных о конфигурации кластера после обновления версии бандла ADQM, если в поле ***Default cluster topology*** была указана некорректная топология кластера (например, шарды/реплики были указаны в неотсортированном порядке или в идентификаторе шарда/реплики присутствовал пробел)

21.8.13.6.1.b1

Дата: 28.01.2022

Новые функции

Реализовано действие ***Reinstall status-checker*** для кластера в состоянии `installed`. Запускает скрипты деплоя status-checker для сервисов, а также для контейнеров Docker

На страницу конфигурации сервиса ADQMDB добавлена секция ***Advanced configuration parameters***, где можно указывать параметры, которые будут сохраняться в файл */etc/clickhouse-server/config.d/advanced.xml*

Исправленные баги

Исправлена ошибка установки сервиса Baje proxy

21.8.8.29

21.8.8.29.1.b2

Дата: 07.12.2021

Исправленные баги

Исправлена ошибка offline-установки пакета ZooKeeper

21.8.8.29.1.b1

Дата: 28.10.2021

Новые функции

ClickHouse обновлен до версии 21.8.8.29-lts

JDBC bridge обновлен до версии 2.0.6

ODBC driver обновлен до версии 1.1.10

Добавлена возможность назначать аутентифицированным пользователям роли из каталога LDAP (см. Аутентификация с использованием LDAP)

21.3.14.1

21.3.14.1.1.b1

Дата: 20.09.2021

Новые функции

Для сервиса Monitoring Clients в ADCM реализованы действия ***Expand*** и ***Shrink***

Добавлена поддержка Alt Linux 8.2

Прочее

Добавлена повторная проверка JDBC bridge

Изменена логика добавления параметра `log_queries` в профиль настроек по умолчанию (файл *users.xml*)

21.3.12.2

21.3.12.2.1.b2

Дата: 23.07.2021

Новые функции

Добавлена поддержка RHEL

21.3.12.2.1.b1

Дата: 10.06.2021

Улучшения

Добавлена проверка версий компонентов при выполнении действия ***Check***

Исправленные баги

Исправлена проблема с импортом сервиса HTTP Mirror кластера Enterprise Tools в кластер ADQM

21.3.11.5

21.3.11.5.1.b1

Дата: 26.05.2021

Новые функции

ClickHouse обновлен до версии 21.3.11.5

21.3.4.25

21.3.4.25.1.b1

Дата: 29.04.2021

Новые функции

ClickHouse обновлен до версии 21.3.4.25-lts

Реализована возможность устанавливать ADQM без доступа в интернет (см. раздел Offline-установка)

Исправленные баги

Исправлена ошибка с применением действия ***Expand*** к сервису Zookeeper

Исправлено: при установке мониторинга не загружались дашборды для ADQMDB

Исправлено: в таблицу ClickHouse загружалось более одного сообщения в формате Avro из Kafka

20.8.13.15

20.8.13.15.1.b1

Дата: 05.04.2021

Новые функции

ClickHouse обновлен до версии 20.8.13.15-lts

JDBC bridge обновлен до версии 2.0

Реализована возможность использовать сервер LDAP для аутентификации пользователей

Улучшения

Добавлена проверка Zookeeper после установки

Исправленные баги

Исправлена ошибка установки кластера ADQM при использовании внешнего ZooKeeper

20.8.11.17

20.8.11.17.1.b2

Дата: 08.02.2021

Исправленные баги

Исправлена ошибка в файле *users.xml*: в списке `networks` повторялся один и тот же IP-адрес

20.8.11.17.1.b1

Дата: 27.01.2021

Новые функции

ClickHouse обновлен до версии 20.8.11.17-lts

Добавлены параметры для конфигурирования аккаунта пользователя и профиля настроек по умолчанию

Прочее

Реализовано создание макросов для нескольких кластерных топологий

Удалены кнопки ***Create user*** и ***Delete users***

20.8.4.11

20.8.4.11.1.b1

Дата: 20.10.2020

Новые функции

ClickHouse обновлен до версии 20.8.4.11-lts

20.3.20.6

20.3.20.6.1.b1

Дата: 20.10.2020

Новые функции

ClickHouse обновлен до версии 20.3.20.6-lts

20.8.2.3

20.8.2.3.1.b1

Дата: 09.10.2020

Новые функции

ClickHouse обновлен до версии 20.8.2.3-stable

Добавлена возможность указывать конфигурационные параметры Kafka

20.3.13.130

20.3.13.130.1.b1

Дата: 27.08.2020

Новые функции

ClickHouse обновлен до версии 20.3.13.130-lts

На страницу ***Configuration*** сервиса ADQMDB в интерфейсе ADCM добавлены параметры: ***Cluster Configuration***, ***Storage Configuration***, ***User policy***, ***query_log***

Прочее

Максимальная версия Enterprise Tools (ET) для ADQM обновлена до 2020022800

Добавлена возможность указывать имя пользователя и пароль для реплик в секции `remote_servers` конфигурационного файла *config.xml*

20.3.10.75

20.3.10.75.1.b2

Дата: 06.08.2020

Исправленные баги

Исправлено: неверная минимальная версия для обновления

20.3.10.75

Дата: 11.06.2020

Новые функции

ClickHouse обновлен до версии 20.3.10.75

Исправленные баги

Исправлена ошибка проверки сервиса Zookeeper при выполнении действия ***Check*** для кластера

19.14.11.16

19.14.11.16.1.b3

Дата: 11.08.2020

Прочее

Максимальная версия Enterprise Tools (ET) для ADQM обновлена до 2020022800

19.14.11.16.1.b2

Дата: 04.08.2020

Улучшения

Проведен рефакторинг бандла ADQM

19.14.11.16

Дата: 14.05.2020

Новые функции

ClickHouse обновлен до версии 19.14.11.16-lts

Добавлена поддержка операционной системы Alt Linux

Бандлы ADQM разделены на версии Community и Enterprise

Обновлено лицензионное соглашение для версии Community

После загрузки бандла открывается окно для подписания лицензионного соглашения

В сервис ADQMDB добавлен компонент Clickhouse JDBC Bridge

Реализована возможность использовать внешний ZooKeeper

Известные проблемы ADQM

Елена Дворядкина

Назад к содержанию

Проблема	С версии	Fixed	Обходное решение
Выполнение некоторых запросов не может быть остановлено командой `KILL QUERY` (например, если запрос содержит подзапрос, вычисляющий скалярное значение)	19.14.11.16	—	Перезапустите сервер ClickHouse (см. подробнее)