Arenadata Orchestrator

Arenadata Orchestrator (ADO) — это платформа для создания и эксплуатации конвейеров данных в производственной среде. Центральным компонентом платформы является Apache Airflow, инструмент с открытым исходным кодом, используемый для создания, планирования и мониторинга последовательностей процессов и задач (DAG).

Версия CURRENT Язык: RU

Содержание

1. Концепции
   - Архитектура Airflow
2. Подготовка окружения
   - Требования к сети
   - Программные требования
3. Начало работы
   - Установка
     - Online-установка
       - Установка ADCM
       - Подготовка хостов
       - Установка кластера ADO
         - 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 ADO
         - Создание кластера
         - Добавление сервисов
         - Добавление хостов в кластер
         - Добавление компонентов
         - Настройка сервисов
         - Настройка кластера
         - Импорт настроек ET
         - Установка кластера
       - Установка мониторинга
   - Подключение к Airflow
     - CLI
     - REST API
4. Администрирование кластера
   - Управление кластером через ADCM
     - Кластерные действия
       - Add/Remove components
       - Check
       - Manage SSL
       - Manage Kerberos
       - Reinstall status-checker
       - Start
       - Stop
     - Сервисные действия
       - ADPG
       - Airflow
       - Monitoring
       - Redis
   - Безопасность
     - Интеграция с HashiCorp Vault для хранения конфиденциальных данных
   - Логирование
   - Управление зависимостями через ADCM
   - Оптимизация производительности
5. Web-интерфейс
   - Обзор
   - Работа с DAG
     - Создание простого DAG
     - Работа с TaskFlow
     - Использование сенсоров
     - Кастомизация расписания DAG
     - Добавление кастомных операторов и хуков
     - Динамическая генерация DAG
     - Использование шаблонов
6. Справочные материалы
   - Конфигурационные параметры
   - Airflow CLI
     - Общие команды
       - cheat-sheet
       - dag-processor
       - info
       - kerberos
       - plugins
       - rotate-fernet-key
       - scheduler
       - standalone
       - sync-perm
       - triggerer
       - version
       - webserver
     - Команды Celery
       - flower
       - stop
       - worker
     - Просмотр конфигурации
       - get-value
       - list
     - Управление соединениями
       - add
       - delete
       - export
       - get
       - import
       - list
       - test
     - Управление DAG
       - backfill
       - delete
       - details
       - list
       - list-import-errors
       - list-jobs
       - list-runs
       - next-execution
       - pause
       - report
       - reserialize
       - show
       - show-dependencies
       - state
       - test
       - trigger
       - unpause
     - Управление БД
       - check
       - check-migrations
       - clean
       - downgrade
       - drop-archived
       - export-archived
       - init
       - reset
       - shell
       - upgrade
     - Управление заданиями
       - check
     - Управление KubernetesExecutor
       - cleanup-pods
       - generate-dag-yaml
     - Управление пулами
       - delete
       - export
       - get
       - import
       - list
       - set
     - Управление провайдерами
       - auth
       - behaviours
       - get
       - hooks
       - links
       - list
       - logging
       - secrets
       - widgets
     - Управление ролями
       - add-perms
       - create
       - del-perms
       - delete
       - export
       - import
       - list
     - Управление задачами
       - clear
       - failed-deps
       - list
       - render
       - run
       - state
       - states-for-dag-run
       - test
     - Управление пользователями
       - add-role
       - create
       - delete
       - export
       - import
       - list
       - remove-role
     - Управление переменными
       - delete
       - export
       - get
       - import
       - list
       - set
7. Релизы
   - Релизы ADO
   - Поддерживаемые сервисы

Архитектура Airflow

Константин Алтаевский, Дарья Барышева

AD.O

Содержание

Назад к содержанию

- Модель данных
  - DAG
  - Task
  - Operator
- Idempotence
- Архитектура
  - Дополнительные компоненты в ADO

Airflow — это платформа, позволяющая разрабатывать, планировать, запускать и отслеживать выполнение сложных процессов обработки данных (workflows). Airflow идеально подходит для описания ETL/ELT-процессов, но также может быть полезен для периодического запуска и мониторинга любых иных задач. Для описания процессов Airflow использует язык программирования Python.

Модель данных

В основе модели данных Airflow лежат следующие объекты.

DAG

Это ключевой объект модели данных Airflow, представляющий семантически связанную группу задач, которые должны выполняться в строго определенном порядке — в соответствии с указанным расписанием. Визуально DAG выглядит как ориентированный ациклический граф (directed acyclic graph), то есть граф без циклических зависимостей.

Простой DAG, изображенный ниже, состоит из трех задач (Task). Обратите внимание, что DAG определяет порядок запуска задач, а также зависимости между ними. Дополнительно Airflow позволяет настроить расписание для обработки DAG, действия в случае неудачного запуска, количество повторных попыток запуска и многое другое. При этом DAG не анализирует, что именно он запускает — за настройку обработки данных отвечают непосредственно сами задачи (Task).

Простой DAG

Task

Задача (Task) — это отдельный узел DAG, который описывает операции, применяемые к данным. Такими операциями могут быть загрузка данных из внешних источников, агрегация и индексирование, удаление дубликатов и другие типовые ETL-операции.

Operator

Оператор (Operator) — это шаблон для выполнения задач. Если задачи описывают применяемые к данным действия, то операторы определяют конкретные действия на уровне кода: с использованием Python-функций, Bash-скриптов и так далее.

Airflow поддерживает множество встроенных операторов. Кроме этого, доступны операторы из community provider packages. Также есть возможность создать свой собственный Operator путем расширения базового класса `BaseOperator`.

Отдельный вид операторов — **Сенсоры (Sensors)** — позволяют обрабатывать реакцию на определенные события: наступление заданного времени, загрузку специфических данных, запуск других DAG/Task и так далее.

Примеры полезных Airflow операторов приведены ниже:

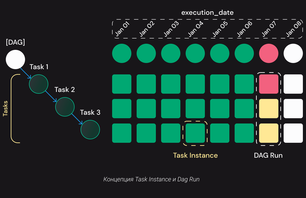
Операторы Airflow

Оператор	Описание
PythonOperator	Выполнение Python-кода
BranchPythonOperator	Выполнение программного кода в зависимости от заданных условий
BashOperator	Запуск Bash-скриптов
SimpleHttpOperator	Отправка HTTP-запросов
PostgresOperator	Отправка запросов в базу данных PostgreSQL
MySqlOperator	Отправка запросов в базу данных MySQL
SqlSensor	Проверка выполнения SQL-запроса
DockerOperator	Запуск Docker-контейнера для задачи (Task)
KubernetesPodOperator	Создание отдельного пода (Pod) Kubernetes для задачи (Task)
EmailOperator	Отправка электронной почты (Email)
DummyOperator	"Пустой" оператор, используемый для группировки задач (Task)

Idempotence

Одна из ключевых концепций, лежащих в основе Airflow — это хранение информации о каждом запуске DAG в соответствии с настроенным расписанием. Например, если указать, что DAG должен запускаться начиная с 01.01.2022 00:00:00 один раз в сутки, AirFlow будет сохранять информацию о запусках DAG для следующих временных метки: 01.01.2022 00:00:00, 02.01.2022 00:00:00, 03.01.2022 00:00:00 и так далее. Временные метки, приведенные в нашем примере, называются **execution_date**, соответствующие им экземпляры DAG — **DAG Run**, а экземпляры задач (Task), связанные с каждым конкретным DAG Run — **Task Instance**.

Эта концепция крайне важна для поддержания идемпотентности данных (idempotence): запуск или перезапуск задачи с одинаковым набором данных для определенной даты в прошлом всегда приводит к одним и тем же результатам. Дополнительно появляется возможность запускать задачи одного DAG для разных временных меток одновременно.



Концепция Task Instance и Dag Run

Архитектура

Архитектура Airflow основана на следующих компонентах:

- **Web Server**. Предоставляет удобный пользовательский интерфейс, который позволяет запускать DAG и Task, отслеживать их расписание, статус выполнения и так далее.
- **Metadata Database**. Общая база данных, которую используют Scheduler, Executor и Web Server для хранения системных метаданных: глобальных переменных, настроек подключения к источникам данных, статусов запуска Task Instance и DAG Run и так далее. Так как в основе этого компонента лежит библиотека SQLAlchemy, требуется установка совместимой с ней базы данных. Например, MySQL или PostgreSQL.
- **Планировщик (Scheduler)**. Сервис, отвечающий за планирование в Airflow. Он управляет как запуском запланированных процессов по расписанию, так и непосредственной отправкой Task в Executor для выполнения.
  Регулярно (по умолчанию раз в минуту) планировщик анализирует результаты парсинга DAG и проверяет наличие задач, готовых к выполнению. Если необходимые для запуска задач условия соблюдены, планировщик инициализирует создание Task Instance. Для обработки активных задач Scheduler использует Executor, определенный в их настройках.

  Определенные версии Metadata Database (PostgreSQL 9.6+ и MySQL 8+) поддерживают наличие нескольких планировщиков. Их применение обеспечивает максимальную отказоустойчивость системы.
- **Executor**. Механизм, с помощью которого выполняются экземпляры задач (Task Instance). По умолчанию Executor работает внутри планировщика (в рамках одного и того же процесса). Но для production-сред рекомендуется использовать типы Executor, делегирующие выполнение задач в Worker.

Типы Executor

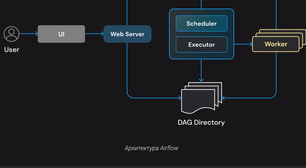
Тип	Описание
SequentialExecutor	Запускает задачи (Task) и приостанавливает планировщик (Scheduler) на время их выполнения. Этот тип Executor рекомендуется использовать только для тестирования — для production-среды он не подходит
LocalExecutor	Запускает дочерний процесс для каждой задачи (Task), что позволяет обрабатывать несколько задач одновременно. Это является отличной имитацией production-среды в тестовом окружении, но также не рекомендуется для реального использования из-за низкой отказоустойчивости: в случае сбоя на машине, на которой запущен Executor, задача не передается на другие узлы
CeleryExecutor	Поддерживает работу множества Worker на различных узлах. Использует библиотеку Celery и требует установки брокера сообщений. Например, Redis или RabbitMQ. При увеличения нагрузки достаточно добавить дополнительный Worker. При сбое любого Worker его задачи передаются работоспособным узлам
DaskExecutor	Аналогичен CeleryExecutor, но вместо Celery для параллельных вычислений использует библиотеку Dask
KubernetesExecutor	Запускает Worker на отдельном поде (Pod) в Kubernetes для каждого экземпляра задачи (Task Instance)
CeleryKubernetesExecutor	Позволяет запускать CeleryExecutor и KubernetesExecutor одновременно. Конкретный тип Executor выбирается в зависимости от текущего статуса очереди задач (Task)
DebugExecutor	Используется для запуска и отладки пайплайнов из IDE

- **Worker**. Отдельный процесс, в котором выполняются задачи (Task). Расположение Worker зависит от выбранного типа Executor.
- **DAG Directory**. Директория с DAG-файлами, доступная для Scheduler, Executor и Worker.

ВАЖНО

Все DAG в форме python-скриптов необходимо размещать в DAG Directory. Web UI предназначен только для мониторинга DAG, не для их добавления либо изменения.

Верхнеуровневая архитектурная схема Airflow изображена ниже. В зависимости от используемых типов Executor эта схема может содержать дополнительные компоненты: например, очередь сообщений (Message Queue) между Executor и Worker в случае CeleryExecutor. Но даже в таких частных случаях можно по-прежнему рассматривать Executor и Worker как единый логический компонент, управляющий выполнением задач (Task).



Архитектура Airflow

ПРИМЕЧАНИЕ

Для получения дополнительной информации по концепциям можно обратиться к документации Airflow.

Дополнительные компоненты в ADO

Архитектура Airflow в ADO также включает несколько компонентов, которые расширяют базовую функциональность и обеспечивают лучшую масштабируемость:

- **DAG Processor**. Отдельный компонент, отвечающий за парсинг DAG-файлов. Он считывает код DAG из директории и подготавливает их к планированию. В новых версиях Airflow эта логика вынесена из Scheduler, что позволяет строить более гибкие и масштабируемые конфигурации.
- **Triggerer**. Компонент, обрабатывающий отложенные задачи и асинхронные операции. Он позволяет операторам, которые ожидают внешние события, не занимать слот воркера. Это снижает потребление ресурсов и повышает эффективность кластера при выполнении длительных пайплайнов.
- **Broker (Redis)**. Брокер сообщений, который используется для взаимодействия между Scheduler и воркерами (при использовании CeleryExecutor). В ADO в качестве брокера по умолчанию используется Redis. Он отвечает за постановку задач в очередь и их доставку воркерам.
- **Airflow Flower**. Веб-инструмент мониторинга для Celery. Позволяет следить за выполнением задач, состоянием воркеров и очередей. Flower используется для диагностики и мониторинга активности кластера в режиме реального времени.

Требования к сети

Елена Костюченко

Содержание

Назад к содержанию

ADCM

Arenadata PostgreSQL

Airflow2

Redis

Мониторинг

В таблицах ниже перечислены сетевые порты компонентов и серверов сервисов ADO. Столбец ***Конфигурирование*** содержит имя файла, в котором порт может быть переопределен.

ADCM

Компонент/Сервер	Порт	Протокол	Описание
ADCM	22	TCP	Порт подключения ADCM

Arenadata PostgreSQL

Компонент/Сервер	Порт	Протокол	Описание	Параметр	Конфигурирование
Arenadata PostgreSQL	5432	TCP	Порт для подключения к PostgreSQL	port	postgresql.conf

Airflow2

Компонент/Сервер	Порт	Протокол	Описание	Параметр	Конфигурирование
Webserver	8080/8080	HTTP/HTTPS	Порт веб-интерфейса Airflow Server	web_server_port (***server_port*** в ADCM)	airflow.cfg
Worker	8793/8793	HTTP/HTTPS	Порт для доступа к Celery Worker API	worker_log_server_port (***worker_port*** в ADCM)	airflow.cfg
Flower	5555/5555	HTTP/HTTPS	Порт веб-интерфейса Flower	flower_port	airflow.cfg
Metastore	5432	TCP	Порт сервиса PostgreSQL	sql_alchemy_conn, result_backend	airflow.cfg

Redis

Компонент/Сервер	Порт	Протокол	Описание
Redis	6379	TCP	Порт для подключения к Redis

Мониторинг

Компонент/Сервер	Порт	Протокол	Описание
Graphana	11210	TCP	Порт для отправки метрик
Prometheus	11200	TCP	Порт для отправки метрик
Node Exporter	11203	TCP	Порт для отправки метрик

Программные требования

Елена Костюченко

Назад к содержанию

Компонент	Требования
Платформа	Intel x86_64
Операционная система	▪ CentOS 7.9. ▪ RHEL 7.9. ▪ AltLinux 8 SP (версия ADO Enterprise). ▪ AltLinux 10 SP (версия ADO Enterprise, начиная с 2.10.5.1). ▪ AstraLinux 1.7.6 "Орел" SE с Axiom JDK (версия ADO Enterprise). Параметру `LC_TIME` должно быть присвоено значение `en_US.utf8`. ▪ AstraLinux 1.7.6 "Воронеж" SE с Axiom JDK (версия ADO Enterprise, начиная с 2.6.3.2). ▪ Ubuntu 22.04.2 LTS. ▪ RED OS 7.3 Сертифицированная редакция (версия ADO Enterprise).
Браузер	▪ Internet Explorer ▪ Firefox ▪ Google Chrome ▪ Safari
Программное обеспечение	▪ RPM + YUM / DPKG + APT ▪ scp ▪ curl ▪ unzip ▪ tar
Python	По умолчанию ADO включает Python 3.10. Путь интерпретатора: `/opt/python3.10/bin/python3.10`
Metastore	▪ PostgreSQL: 12, 13, 14, 15, 16 ▪ MySQL: 8.0, Innovation ▪ SQLite: 3.15.0+ Мы рекомендуем использовать ADPG (Postgres). ADPG 16 доступен как сервис в составе ADO
Broker	Airflow с Celery executor поддерживают RabbitMQ, Redis и экспериментально поддерживают базу данных SQLAlchemy. Redis доступен как сервис в составе ADO
Arenadata Cluster Manager (ADCM)	Версия 2.5.0 и выше

Установка

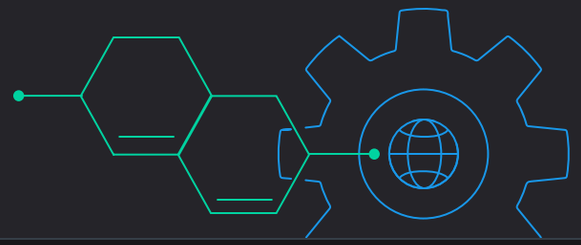
Назад к содержанию

Этот раздел содержит описание установки кластера Arenadata Orchestrator (ADO). 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 ADO 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 ADO. Данный способ установки подходит для хостов с доступом к Интернету.

Процесс online-установки через ADCM включает следующие шаги.

AD.CM

Шаг 1.

Установка ADCM

AD.CM

Шаг 2.

Подготовка хостов

AD.CM

Orchestrator

Шаг 3.

Установка кластера ADO

AD.CM

Шаг 4.

Установка мониторинга

Установка АСУТП

Содержание

Шаг 1. Предварительная подготовка
- Требования к сетевым настройкам
- Требования к аппаратным ресурсам
- Требования к системным ресурсам
- Установка Docker
- Установка Docker Compose
- Установка Git
- Установка Python
- Установка Node.js
- Установка Yarn
- Установка Nginx
- Установка Redis
- Установка PostgreSQL
- Установка Elasticsearch
- Установка Logstash
- Установка Kibana
- Установка Filebeat
- Установка Metricbeat
- Установка Packetbeat
- Установка Auditbeat
- Установка Heartbeat
- Установка Winlogbeat
- Установка Elastic Agent
- Установка Fleet Server

Шаг 2. Установка АСУТП
- Клонирование репозиториев
- Установка Python-зависимостей
- Установка Node.js-зависимостей
- Сборка фронтенда
- Сборка Elastic Agent
- Сборка Fleet Server
- Сборка Elasticsearch
- Сборка Kibana
- Сборка Logstash
- Сборка Filebeat
- Сборка Metricbeat
- Сборка Packetbeat
- Сборка Auditbeat
- Сборка Heartbeat
- Сборка Winlogbeat
- Сборка Elastic Agent
- Сборка Fleet Server
- Сборка Docker-образов

Шаг 3. Запуск АСУТП
- Запуск Elasticsearch
- Запуск Kibana
- Запуск Logstash
- Запуск Fleet Server
- Запуск Elastic Agent
- Запуск Filebeat
- Запуск Metricbeat
- Запуск Packetbeat
- Запуск Auditbeat
- Запуск Heartbeat
- Запуск Winlogbeat
- Запуск Redis
- Запуск PostgreSQL
- Запуск Nginx
- Запуск Yarn
- Запуск Docker
- Запуск Docker Compose

Шаг 4. Проверка установки
- Проверка статуса Docker-контейнеров
- Проверка доступности сервисов
- Проверка доступности Elasticsearch
- Проверка доступности Kibana
- Проверка доступности Fleet Server
- Проверка доступности Elastic Agent
- Проверка доступности Redis
- Проверка доступности PostgreSQL
- Проверка доступности Nginx
- Проверка доступности Yarn
- Проверка доступности Docker
- Проверка доступности Docker Compose

Шаг 5. Обновление АСУТП
- Обновление Docker-образов
- Обновление Elastic Agent
- Обновление Fleet Server
- Обновление Elasticsearch
- Обновление Kibana
- Обновление Logstash
- Обновление Filebeat
- Обновление Metricbeat
- Обновление Packetbeat
- Обновление Auditbeat
- Обновление Heartbeat
- Обновление Winlogbeat
- Обновление Redis
- Обновление PostgreSQL
- Обновление Nginx
- Обновление Yarn
- Обновление Docker
- Обновление Docker Compose

Шаг 1. Предварительная подготовка

Требования к сетевым настройкам

Для корректной работы необходимо обеспечить доступность следующих портов:

- 9200
- 5601
- 8220
- 5044
- 9600
- 8200
- 8201
- 8202
- 8203
- 8204
- 8205
- 8206
- 8207
- 8208
- 8209
- 8210
- 8211
- 8212
- 8213
- 8214
- 8215
- 8216
- 8217
- 8218
- 8219
-

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

ВНИМАНИЕ

При создании хостов ADO в ADCM требуется заполнять их полные доменные имена (FQDN).

Создание кластера

Елена Костюченко, Дарья Барышева

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.

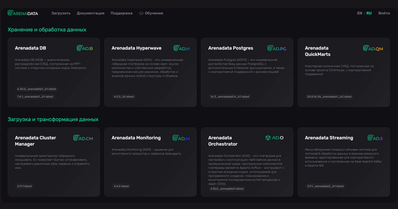
Шаги для добавления нового кластера приведены ниже.

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 Orchestrator***.



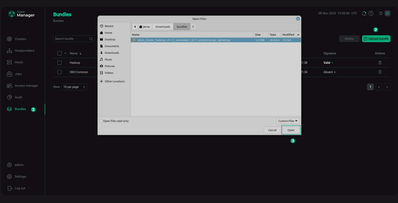
Переход к загрузке дистрибутивов

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

Шаг 2. Загрузка бандла в ADCM

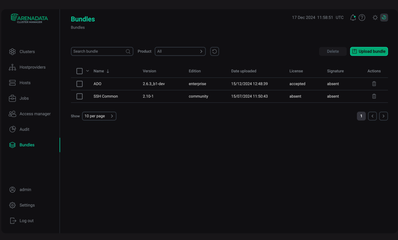
Чтобы загрузить бандл в ADCM:

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



Переход к загрузке бандла

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



Результат успешной загрузки бандла

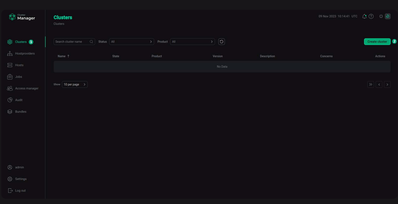
Шаг 3. Создание кластера на базе загруженного бандла

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.

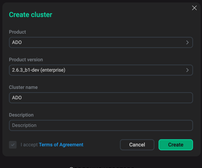
Для создания кластера необходимо выполнить шаги:

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



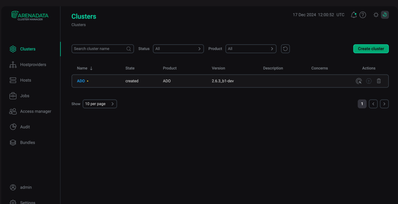
Переход к созданию кластера

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***.



Создание кластера

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



Результат успешного добавления кластера

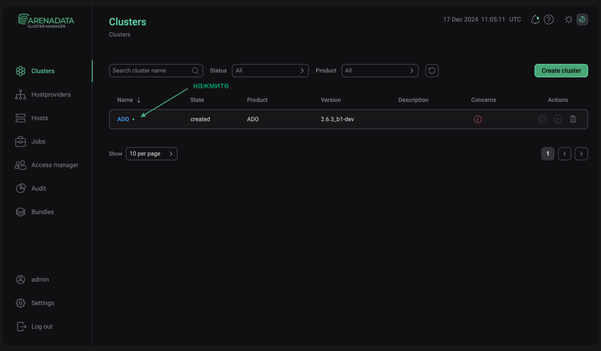
Добавление сервисов

Елена Костюченко, Дарья Барышева

Назад к содержанию

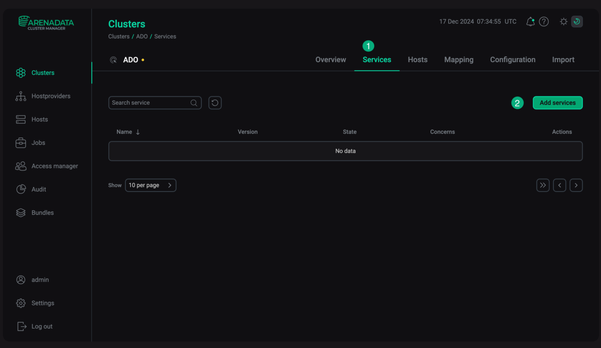
В ADCM **сервис** означает программное обеспечение, выполняющее некоторую функцию. Примеры сервисов в кластерах ADO: Airflow, ADPG и другие. Для добавления сервисов в кластер:

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



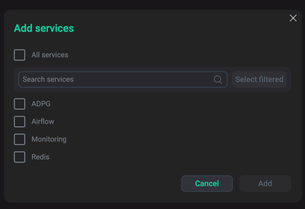
Выбор кластера

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



Переход к добавлению сервисов

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



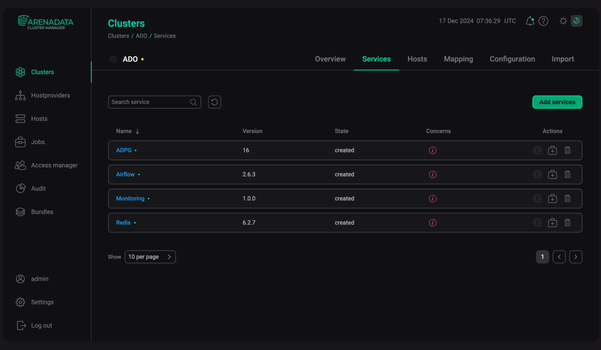
Выбор сервисов

Краткое описание доступных сервисов приведено ниже.

Сервисы, которые могут быть добавлены в кластер ADO

Сервис	Назначение
Airflow	Сервис для создания, планирования и мониторинга процессов, представленных в форме ориентированных ациклических графов (Directed Acyclic Graphs, DAG). Используется в кластерах Arenadata Orchestrator для построения процессов ETL/ELT
Monitoring	Сервис, добавляемый в случаях, когда запланирована настройка мониторинга ADO через ADCM
Redis	Redis — это хранилище данных NoSQL с открытым исходным кодом, которое размещается в памяти и может быть использовано как кеш приложений или база данных с быстрым откликом
ADPG	Arenadata Postgres (ADPG) — коммерческий дистрибутив базы данных PostgreSQL с дополнительными функциями Enterprise

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



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.

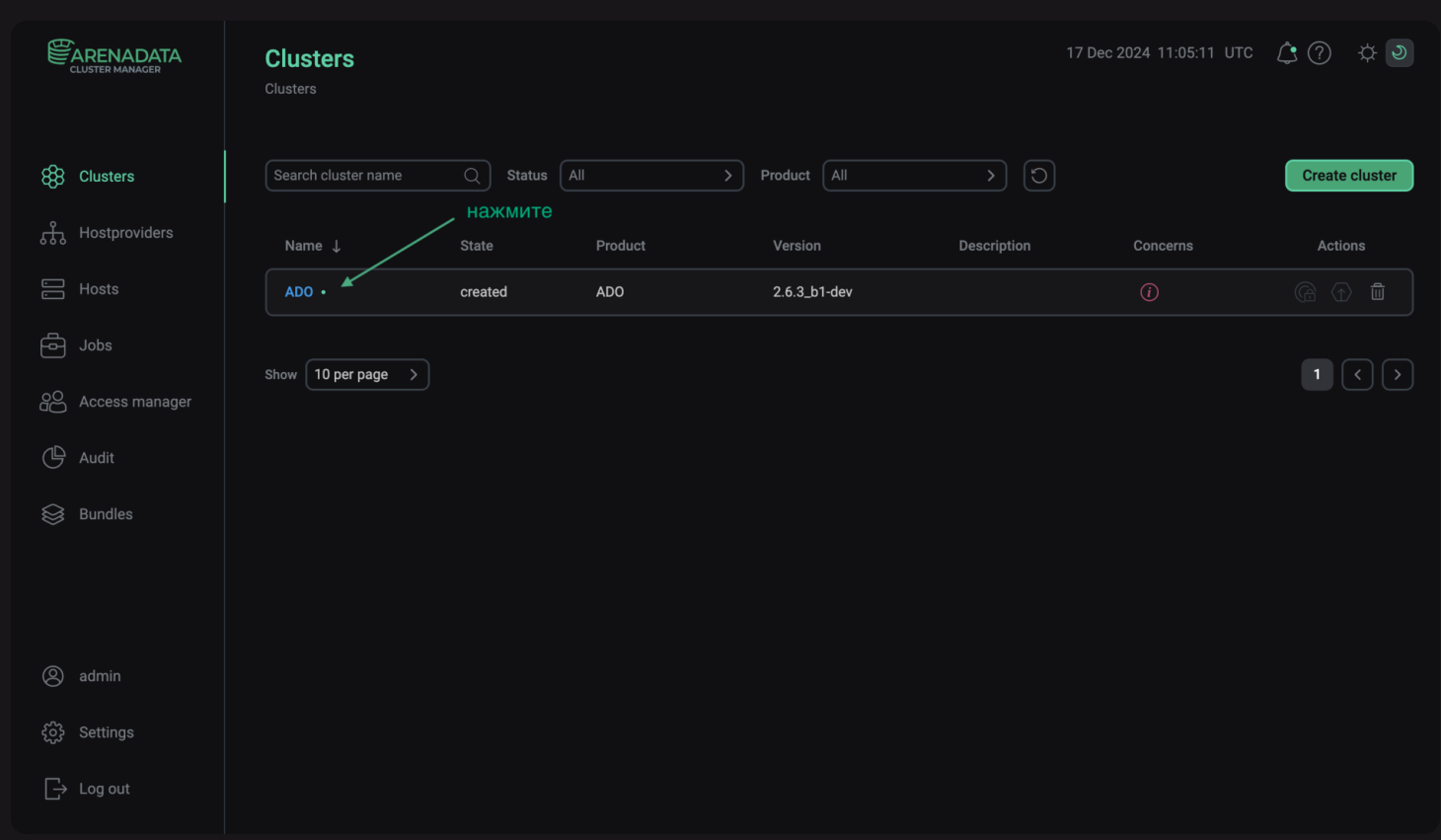
Добавление хостов в кластер

Елена Костюченко, Дарья Барышева

Назад к содержанию

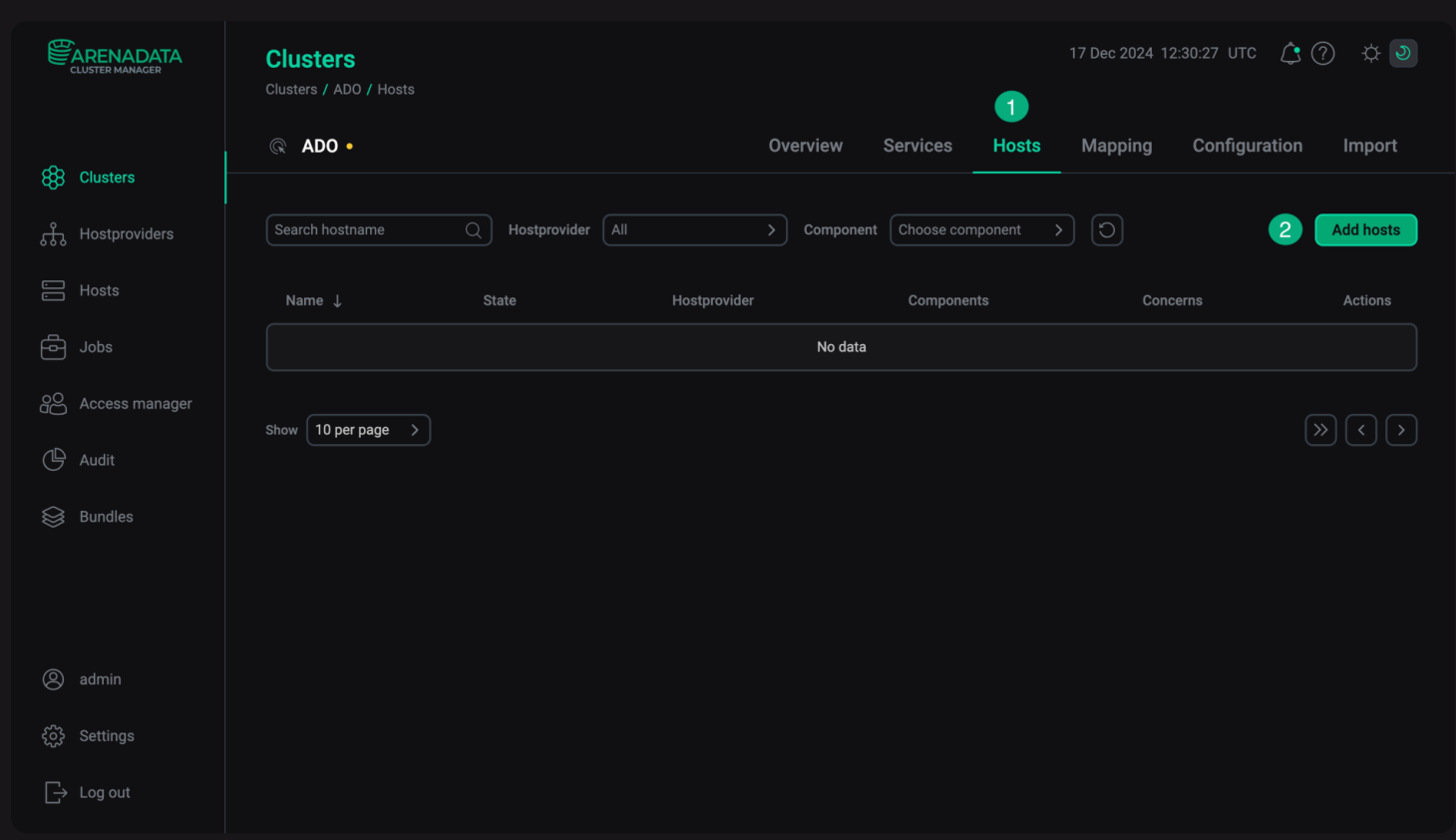
Для добавления хостов в кластер:

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



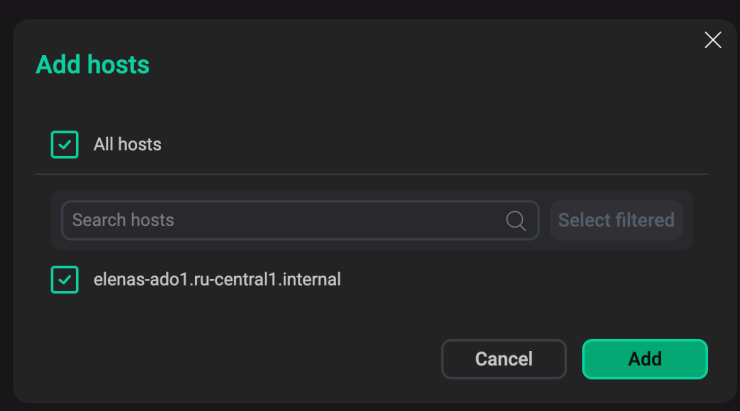
Выбор кластера

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



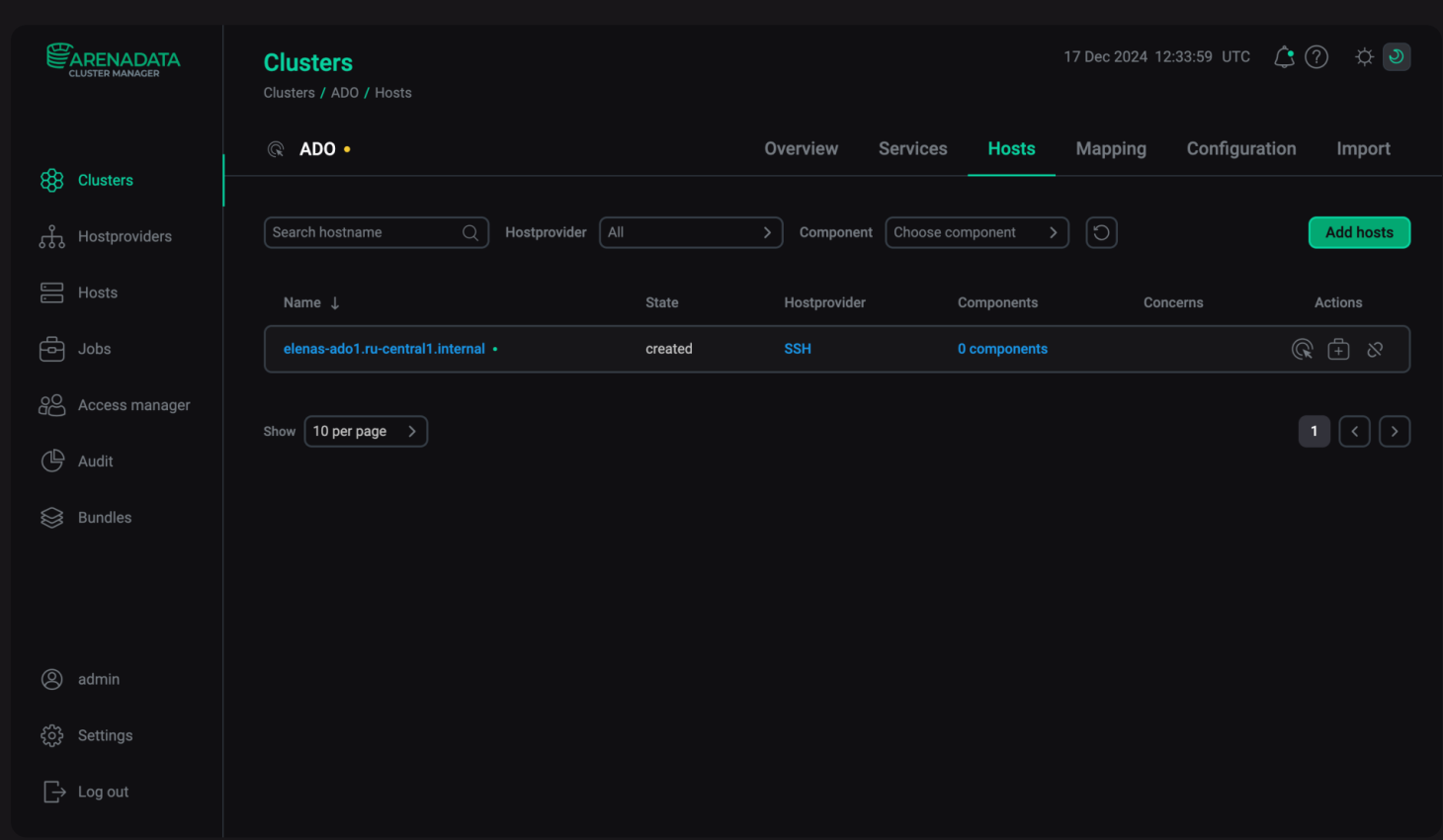
Переход к добавлению хостов

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



Выбор хостов

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



Результат успешного добавления хостов в кластер

Добавление компонентов

Елена Костюченко, Дарья Барышева

Назад к содержанию

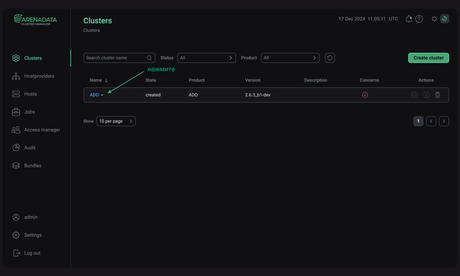
В ADCM **компонент** означает часть сервиса, которая должна быть развернута на одном или нескольких хостах кластера. Например, сервис Airflow состоит из таких компонентов, как Airflow Server, Airflow Scheduler, Airflow Worker и другие.

ВАЖНО

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

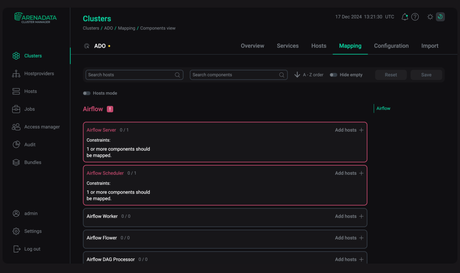
Для добавления компонентов на хосты кластера:

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



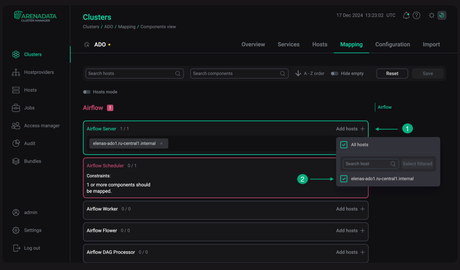
Выбор кластера

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



Переход к распределению сервисных компонентов

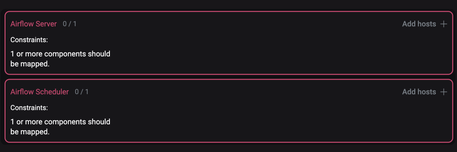
3. Распределите все компоненты между хостами кластера. Для назначения хостов компоненту нажмите ***Add hosts*** и выберите хосты в открывшемся окне.



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.



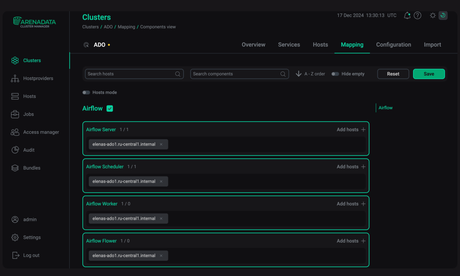
Обязательные компоненты

ВНИМАНИЕ

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

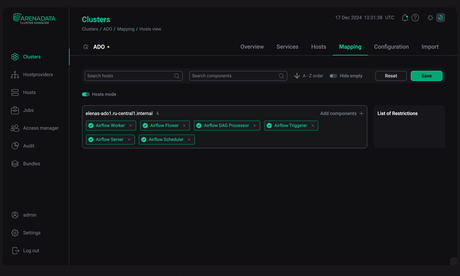
Все компоненты в кластере ADO могут быть установлены на одном хосте. Вы можете добавить несколько хостов в кластер (в зависимости от требований к производительности) и распределить компоненты сервисов между ними.

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



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.



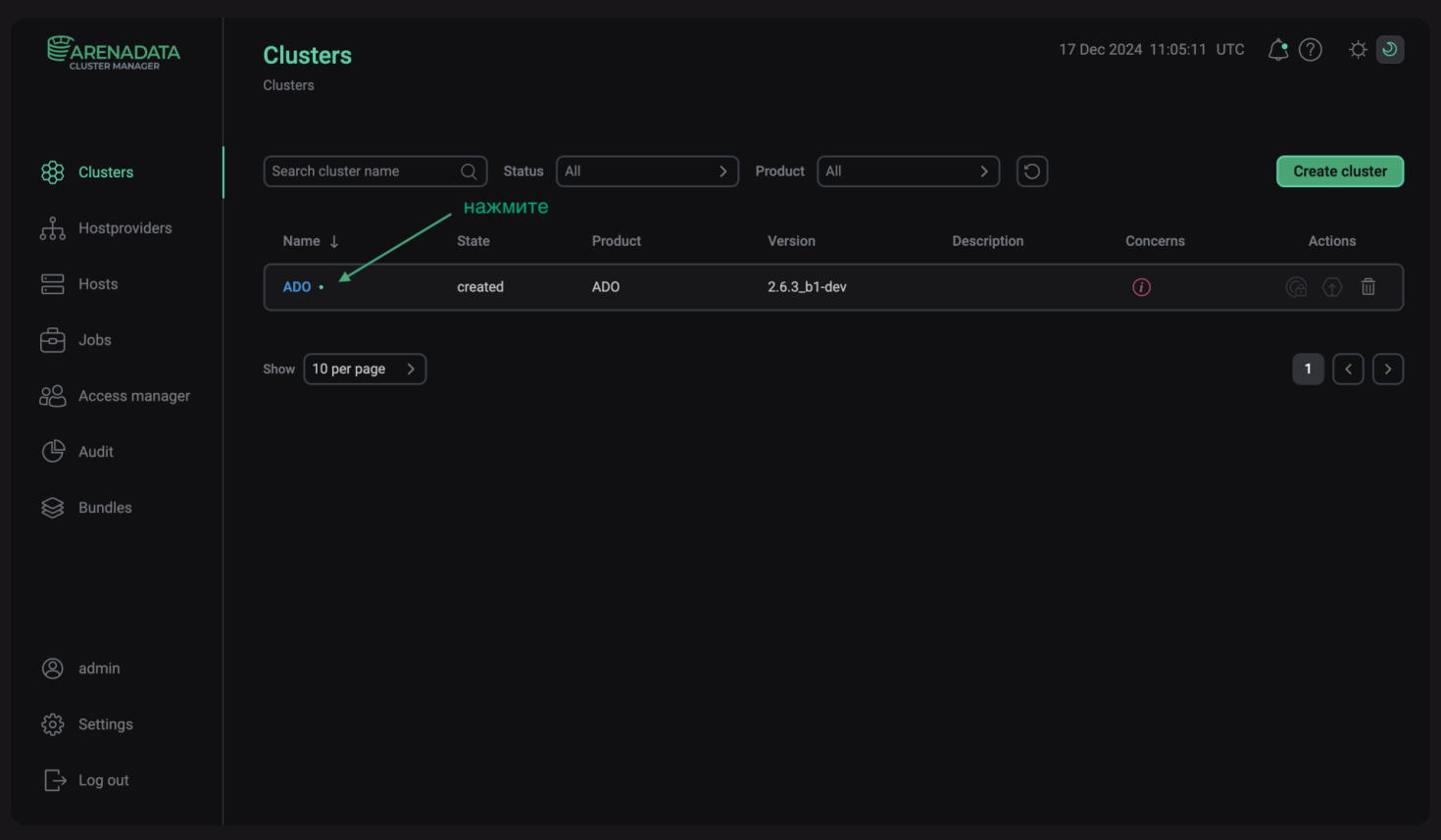
Распределение по хостам

Настройка сервисов

Елена Костюченко, Дарья Барышева

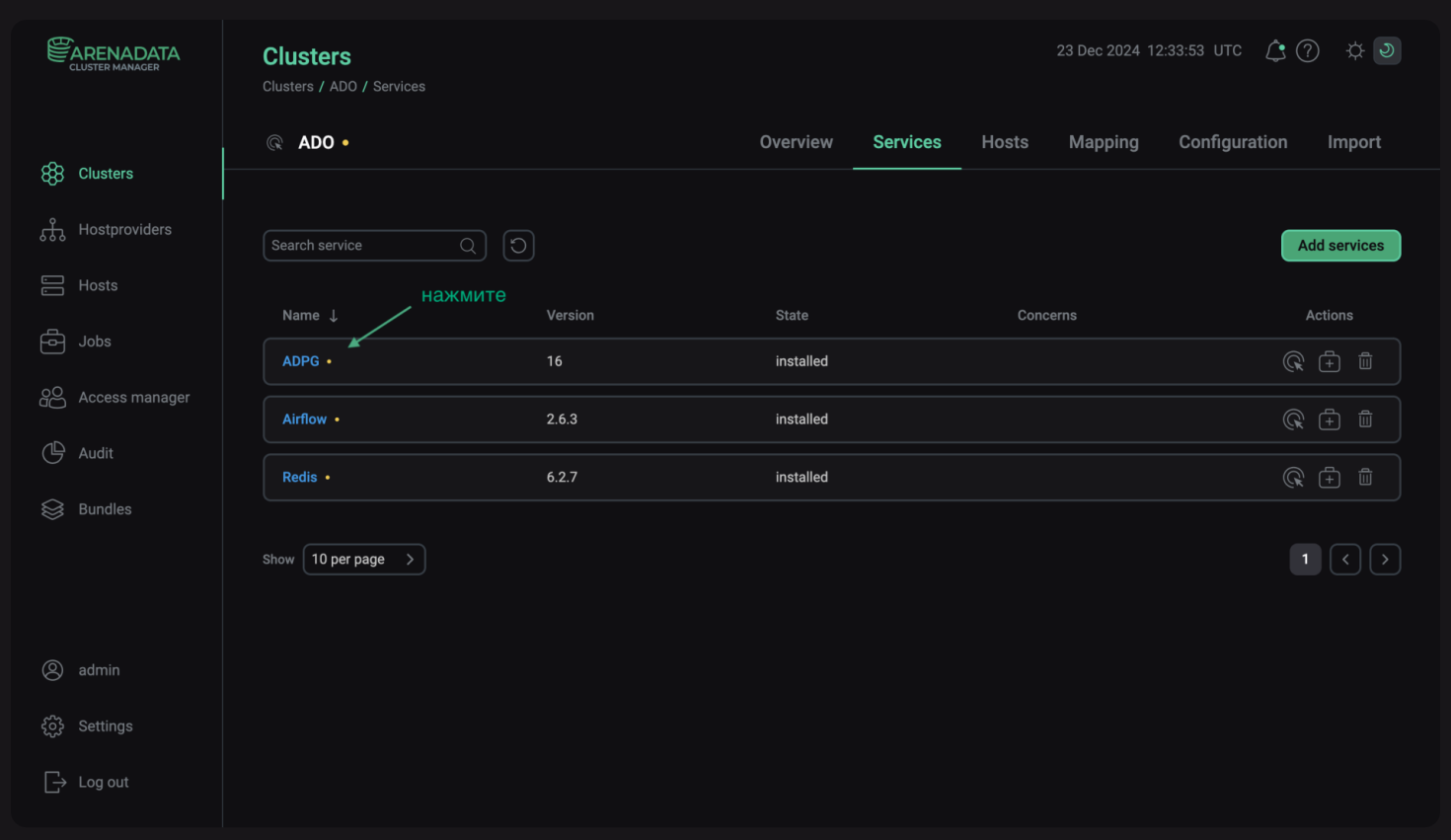
Назад к содержанию

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



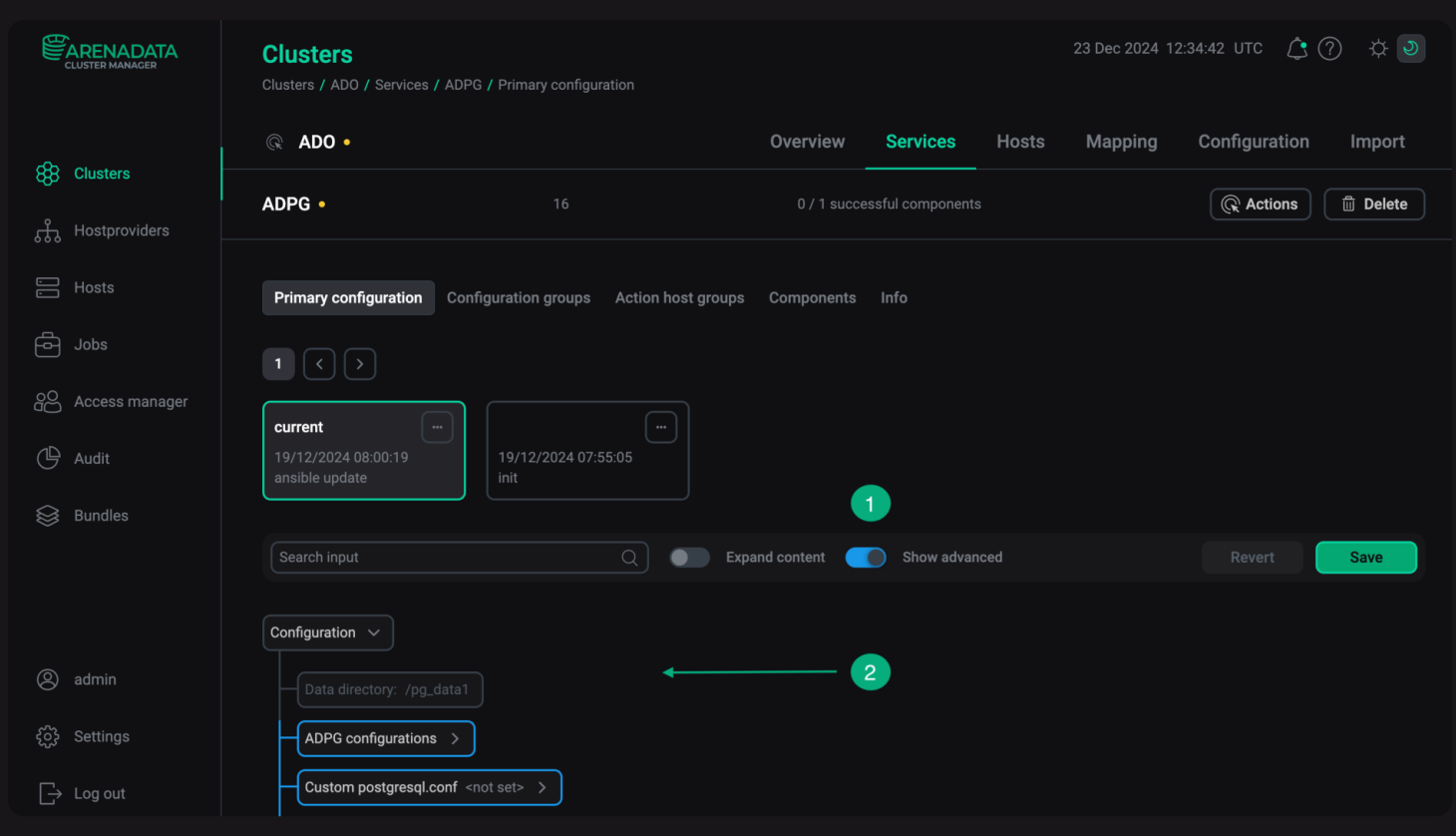
Выбор кластера

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



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***.



Настройка сервиса

ПРИМЕЧАНИЕ

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

Настройка кластера

Елена Костюченко, Дарья Барышева

Содержание

Назад к содержанию

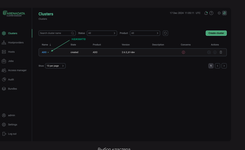
Primary configuration
Configuration groups
Ansible settings

ВАЖНО

В большинстве случаев кастомная настройка кластера не требуется – значение 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**.



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 параметры перечислены ниже:

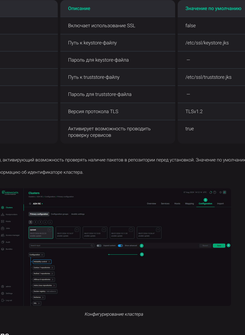
- Раздел ***Reliability control***. Отображается, если активирован переключатель **Show advanced**. Раздел содержит следующие настройки:
  - ***retries*** – количество попыток для операций в кластере (по умолчанию: 20);
  - ***delay*** – задержка между попытками в секундах (по умолчанию: 10);
  - ***timeout*** – тайм-аут проверки сетевых портов в секундах (по умолчанию: 60).
- Путь к репозиторию CentOS, Red Hat, ALT Linux & Astra Linux, которые будут использованы в процессе установки в зависимости от операционной системы кластера:
  - ***annadata*** – YUM-репозиторий Annadata;
  - ***annadata_postgres*** – YUM-репозиторий Annadata PostgreSQL;
  - ***prometheus_repo*** – YUM-репозиторий Prometheus.
- Параметр ***DockerRegistry***, указывающий адрес Docker Registry. Значение по умолчанию – *hub.adsw.io* – Docker Registry из репозиториев Annadata.
- Раздел ***Kerberos***, содержащий параметры, перечисленные в таблице ниже. Некоторые параметры становятся видимыми только при установленном флажке **Show advanced**.

Параметр	Описание	Значение по умолчанию
Enable Kerberos	Включает аутентификацию Kerberos	False
Authentication on WEB UIs	Включает аутентификацию Kerberos для веб-интерфейсов	False
SPNEGO Signature Value	Секретный ключ, используемый для генерации и проверки подписи в протоколе HTTP-аутентификации SPNEGO (в формате Base64)	—
Kerberos KDC type	Тип КОС. За дополнительной информацией обратитесь к статье Kerberos	MIT
KDC hosts	Один или несколько хостов КОС с запущенными серверами FreeIPA	—
Realm	Kerberos realm, для подключения к серверу FreeIPA	—
Domains	Один или несколько доменов, ассоциированных с FreeIPA	—
Kadmin server	Хост, на котором работает `kadmin`	—
Kadmin principal	Имя принципала, используемое для подключения через `kadmin`, например `admin@RU-CENTRAL1.INTERNAL`	—
Kadmin password	Пароль администратора IPA	—
Keytabs directory	Каталог с keytab-файлами	/etc/security/keytabs
Additional realms	Дополнительные Kerberos realms	—
Delay between kinit invocation attempts	Задержка между попытками вызова `kinit`	5
Number of retries for kinit invocation attempts	Количество попыток вызова `kinit`	10
Trusted Active Directory server	Сервер Active Directory для обеспечения односторонних доверительных отношений между realms из MIT Kerberos KDC	—
Trusted Active Directory realm	Active Directory realm для обеспечения односторонних доверительных отношений между realms из MIT Kerberos KDC	—
Custom krb5.conf	Активирует добавление кастомных параметров в файл krb5.conf	False
krb5.conf	Дополнительные параметры для записи в файл krb5.conf	—
Admin DN	Полное отличительное имя (DistinguishedName) пользователя admin с правами `create/modify/delete/pwchange` для создания/изменения учетных записей в целевом Organizational Unit	—
LDAP URL	LDAP URL, который состоит из префикса протокола `ldap://`, или `ldaps://`, имени хоста или IP-адреса и порта AD-сервера. Например `ldaps://192.168.4.2:636`	—
Container DN	Отличительное имя (DistinguishedName) контейнера	—
TLS CA certificate Path	Путь к сертификату CA в файловой системе хоста, который предварительно размещен или будет записан из поля **TLS CA certificate**	—
TLS CA certificate (optional)	Сертификат CA, который записывается на хост по пути из **TLS CA certificate Path** во время исполнения действия **Manage → Kerberos**. Сертификат не создается в ADCM	—
Custom ldap.conf	Активирует добавление кастомных параметров в файл ldap.conf	False
ldap.conf	Дополнительные параметры для записи в файл ldap.conf	—
ipaClient No NTP Autoconf	Отключает автоконфигурирование NTP во время установки клиента IPA	False
ipaClient No DNS Lookup	Отключает поиск DNS для сервера FreeIPA во время установки клиента IPA	True
Custom smb.conf	Использование кастомного файла конфигурации smb.conf	False
smb.conf	Кастомный файл конфигурации smb.conf	—
Set up Kerberos utils	Активирует возможность устанавливать и удалять клиенты и утилиты Kerberos. Данный параметр учитывается в действиях ***Expand*** и ***Install***	True
Configure Kerberos on hosts	Активирует возможность конфигурировать кластер, включая krb5.conf, ldap.conf	True
Set up principals and keytabs	Активирует возможность создавать, пересоздавать и удалять принципалов и keytab-файлы. Пароли для принципалов генерируются случайным образом перед каждым запуском. Данный параметр учитывается в действиях ***Expand*** и ***Install***. Бандл ADCM устанавит сигнал и разрешения для keytab-файлов только если этот флаг установлен. В случае отсутствия прав администратора пользователь должен предоставить подготовленный keytab-файл с правильно установленными сигнал и разрешениями	True
Configure services and clients	Активирует возможность обновлять конфигурации серверов и клиентов	True
Run service checks	Активирует возможность проводить проверку сервисов	True

- Раздел ***SSL*** с настройками, перечисленными в таблице ниже.

Параметр	Описание	Значение по умолчанию
Enable SSL	Включает использование SSL	false
Keystore path	Путь к keystore-файлу	/etc/ssl/keystore.jks
Keystore password	Пароль для keystore-файла	—
Truststore path	Путь к truststore-файлу	/etc/ssl/truststore.jks
Truststore password	Пароль для truststore-файла	—
TLS Version	Версия протокола TLS	TLSv1.2
Run service checks	Активирует возможность проводить проверку сервисов	true

- Параметр ***Precheck packages***, активирующий возможность проверять наличие пакетов в репозиториях перед установкой. Значение по умолчанию – `false`.
- ***Cluster UUID*** отображает информацию об идентификаторе кластера.

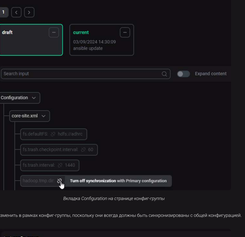


Конфигурирование кластера

Configuration groups

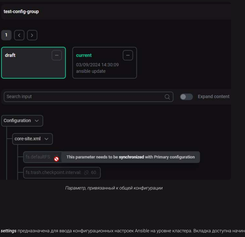
Вкладка ***Configuration → Configuration groups*** предназначена для настройки конфиг-групп кластера.

Чтобы настроить параметры в конфиг-группе, выберите параметры, которые вы хотите настроить иначе, нажмите иконку **Turn off synchronization**, отредактируйте их и нажмите **Save**.



Вкладка Configuration на странице конфиг-группы

Некоторые параметры нельзя изменить в рамках конфиг-группы, поскольку они всегда должны быть синхронизированы с общей конфигурацией.



Параметр, привязанный к общей конфигурации

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

Елена Костоломко, Дарья Борщакова

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.

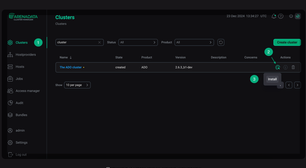
ВАЖНО

Установка должна производиться на чистые хосты без предустановленного подобного ПО.

Установка всего кластера

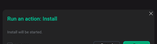
Чтобы установить все сервисы кластера за одну операцию, выполните шаги:

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



Переход к установке всего кластера

2. Подтвердите действие в открывшемся окне.



Подтверждение действия

3. Дождитесь завершения установки.

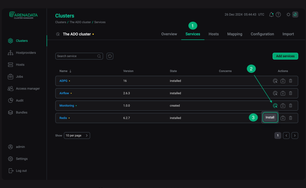
Установка отдельных сервисов

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

ВНИМАНИЕ

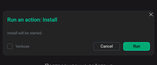
Обратите внимание, что сервисы устанавливаются по очереди друг за другом. Не пытайтесь установить следующий сервис до окончания установки предыдущего.

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.



Переход к установке отдельных сервисов

2. Подтвердите действие в открывшемся окне.

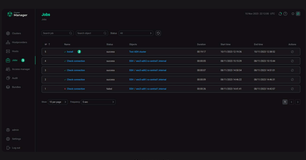


Подтверждение действия

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

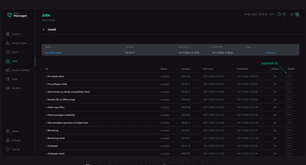
Шаг 2. Просмотр результатов установки

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



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***. Откроется окно с шагами выполнения задачи. Кликните многоточие в столбце ***Details***, чтобы получить дополнительную информацию об определенном шаге.

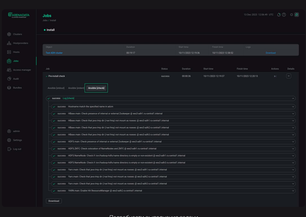


Переход на страницу с подробностями выполнения

Страница задачи содержит два основных раздела: ***ansible [stdout]*** и ***ansible [stderr]***. Это технические логи, содержащие информацию из стандартных I/O потоков stdout/stderr.

В дополнение к ним может выводиться третий опциональный раздел ***ansible [check]***. Он содержит краткие описания наиболее часто возникающих ошибок. По сравнению с предыдущими разделами, эти описания, как правило, более простые и конкретные.

При возникновении ошибок в процессе установки рекомендуется изучить логи всех трех разделов.

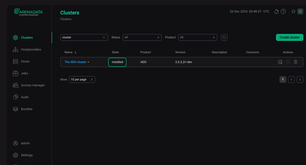


Подробности выполнения задачи

Шаг 3. Проверка статуса кластера после установки

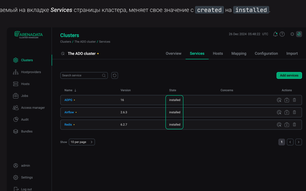
В результате успешной установки происходит обновление статуса кластера и его сервисов:

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



Статус кластера после успешной установки

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



Статус сервисов кластера после успешной установки

Установка мониторинга

Версия 6.0.0 от 15.05.2025

Содержание

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

Перед установкой

Версия мониторинга 4.60.1(x64 сервер Prometheus), которая реализует сбор и хранение метрик мониторинга платформы АОС и такие подключения клиентских мониторингов как мониторинг Grafana для визуализации и анализа метрик. В данной статье описываются шаги, необходимые для полноценной этого сервиса.

Обзор

При добавлении сервиса Monitoring в кластер АОС на всех узлах устанавливается Node Exporter — этот компонент, считывающий с хостов системные метрики, которые будет собирать Prometheus. Prometheus, сервер которого также будет развернут на одном сервере АОС (по умолчанию будет установлен в формате Prometheus as service и запускаться, управляемый в systemd systemctl, переменные окружения).

Prometheus, в зависимости от конфигурации, может использовать для получения метрик данных для сетевых экспортеров АОС, которые, в свою очередь, получают метрики с АОС.

Prometheus, которые также будет развернут на одном сервере АОС, по умолчанию будет установлен в формате Prometheus as service и запускаться, управляемый в systemd systemctl, переменные окружения). Prometheus, в зависимости от конфигурации, может использовать для получения метрик данных для сетевых экспортеров АОС, которые, в свою очередь, получают метрики с АОС.

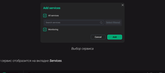
Для полноценного мониторинга платформы АОС, для визуализации и анализа метрик, в данной статье описываются шаги, необходимые для полноценной этого сервиса.

Шаг 1. Добавление сервиса

1. В интерфейсе АОС(по умолчанию Grafana) **Dashboards** в элементе по имени кластера АОС, Затем на открывшейся странице кластера перейдите на вкладку **Services** и нажмите **Add services**

Переход к добавлению сервиса

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



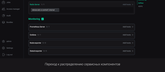
Выбор сервиса

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

Результат успешного добавления сервиса в кластер

Шаг 2. Добавление компонентов

1. На странице кластера открыта вкладка **Mapping**, чтобы перейти к распределению серверных компонентов между системами кластера.



Переход к распределению серверных компонентов

Каждый компонент необходимо установить на один любой хост кластера.

Компоненты сервиса Monitoring

Компонент	Описание
Prometheus Server	Развертывается в АОС сервер Prometheus, который выполняет следующие функции: • прием метрик для всех серверов, собираемых метриками на хосте • хранилище для метрик кластера • генератор оповещений на основе собираемых метрик
Grafana	Развертывает предварительно настроенную Grafana с дашбордами, оповещениями и рендерером
Node Exporter	Считывает с хостов системные метрики, которые будет собирать Prometheus
Statsd-exporter	Перевод метрики из формата Statsd в метрики формата Prometheus с помощью настраиваемых правил

2. Нажмите кнот каждому компоненту сервиса Monitoring — нажмите **Add hosts** и выберите нужный хост в открывшемся окне.

Выбор хоста для компонента

ВНИМАНИЕ

Компонент Prometheus Server на development установлен на хосте с АОС - используйте для него отдельный хост. Иначе при выходе из строя хоста АОС всему кластеру Monitoring будет потеряна информация о состоянии/нагрузке приложения и будет создаваться.

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

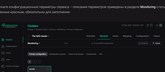
Сохранение распределения компонентов

NOTE

When changing the topology of services/configurations related to monitoring, it is necessary to restart the Prometheus component in the Monitoring service.

Шаг 3. Настройка сервиса

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



Переход к настройке сервиса

2. В открывшемся окне заполните конфигурационные параметры сервиса — описание параметров приведены в разделе **Monitoring** статьи Конфигурационные параметры. Поля, подлежащие изменению, обозначены для заполнения.

Настройка сервиса Monitoring

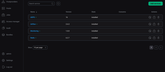
После указания всех необходимых параметров нажмите **Save**.

Шаг 4. Установка сервиса

1. На вкладке **Services** в списке сервисов Monitoring нажмите на иконку **Actions** в столбце **Actions** и запустите действие **Install**

Переход к установке сервиса

2. Подтвердите действие в открывшемся окне

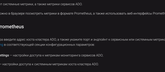


Подтверждение действия

3. Дождитесь завершения установки и убедитесь, что статус сервиса изменился на **installed** — **installed**

Процесс установки завершен

Чтобы увидеть, прошло ли выполнение установки сервиса и проконтролировать ошибки в случае их возникновения, выберите пункт **Jobs** в меню навигационного меню и ознакомьте по каждой задачи **Install** и **Jobs**.



Страница задачи Install

Шаг 5. Проверка результатов

Сервис Monitoring запускается автоматически после установки. По завершении установки необходимо убедиться в следующем:

- Метрики собираются на всех хостах кластера, а не только с хостов, на которых развернуты компоненты сервиса мониторинга
- С системы мониторинга Grafana получаются метрики с сервиса Prometheus АОС

Чтобы проверить обе пункта, зайдите в браузере посмотреть метрики в формате Prometheus, а также использовать веб-интерфейсы Prometheus и Grafana.

Данные в формате Prometheus

1. В адресной строке браузера введите адрес хоста кластера АОС и затем укажите порт в формате с перечислены или системных метриках, которые отправлены на Prometheus Server и в **Prometheus** отображаются в виде отдельных временных серий:
   - **АОС service metric settings** — настройки доступа к метрикам мониторинга сервисов АОС
   - **Node Exporter settings** — настройки доступа к системным метрикам хостов АОС

   Пример адреса для просмотра метрик Node Exporter на хосте IP: **10.10.10.101**: http://10.10.10.101:12000/metrics
2. На открывшейся странице будут показаны метрики мониторинга с указанного хоста в формате Prometheus

Метрики сервера Node в формате Prometheus

Веб-интерфейс Prometheus

1. В адресной строке браузера введите IP-адрес хоста, где установлен компонент Prometheus Server сервера Monitoring. Укажите номер порта, который определяется параметром **Node_address** в списке **Prometheus settings** в разделе сервиса Monitoring (**Services**). Пример адреса: http://10.10.10.101:9090. Для входа используйте **admin** в качестве имени пользователя и пароль, который задан в параметре **Grafana settings** — при первом **Prometheus** сервер по **Grafana**.
2. В поле **Expression** можно ввести название метрики и нажать **Execute** — в интерфейсе будут показаны значения этой метрики на всех хостах кластера АОС.

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** в разделе **Grafana settings** в настройке сервиса Monitoring (**Services**). Пример адреса: http://10.10.10.101:3000. Для входа используйте **admin** в качестве имени пользователя и пароль **Grafana administrator's password** в разделе порта в разделе настроек **Grafana settings** в разделе сервиса.
2. В открывшемся окне выберите пункт меню **Home** — **Dashboards** и развернуте панель **АОС Dashboard v.6(0_cluster_name)**. В этой панели можно выбрать одно из доступных дашбордов для просмотра графиков метрик или системных метрик, поступающих из машин кластера АОС.

Переход к интерфейсу сервиса АОС в Grafana

Offline-установка

Назад к содержанию

Этот раздел посвящен **offline**-установке ADO. 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 ADO.

AD.CM

Шаг 1.

Установка ADCM

AD.CM

Шаг 2.

Подготовка хостов

AD.CM

Шаг 3.

Установка кластера ET

AD.CM

Orchestrator

Шаг 4.

Установка кластера ADO

AD.CM

Шаг 5.

Установка мониторинга

Установка АСКМ

Содержание

Развернуть содержание

Шаг 1. Подготовительные требования
Требования к аппаратуре и ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Требования к ресурсам
Шаг 2. Установка ACKM
Установка ACKM
Установка ACKM

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

Для offline-установки рекомендуется использование SSH-хостпровайдера.

ВНИМАНИЕ

При создании хостов ADO в ADCM требуется 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.

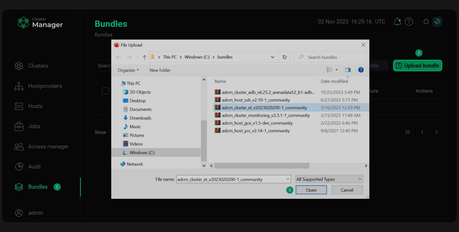
Шаг 1. Получение Enterprise-версии бандла

Пожалуйста, запросите бандл у команды поддержки Arenadata.

Шаг 2. Загрузка бандла в ADCM

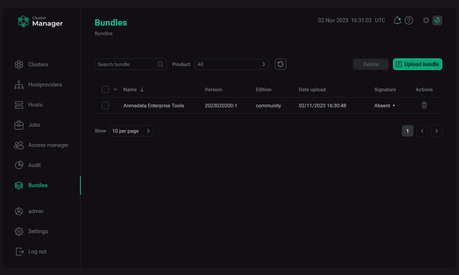
Чтобы загрузить бандл в ADCM:

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



Переход к загрузке бандла

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



Результат успешной загрузки бандла

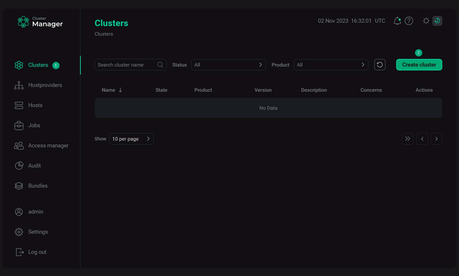
Шаг 3. Создание кластера на базе загруженного бандла

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.

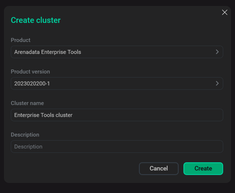
Для создания кластера необходимо выполнить шаги:

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



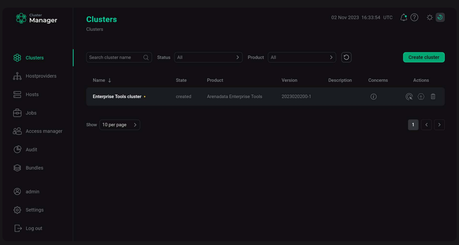
Переход к созданию кластера

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***.



Создание кластера

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



Результат успешного добавления кластера

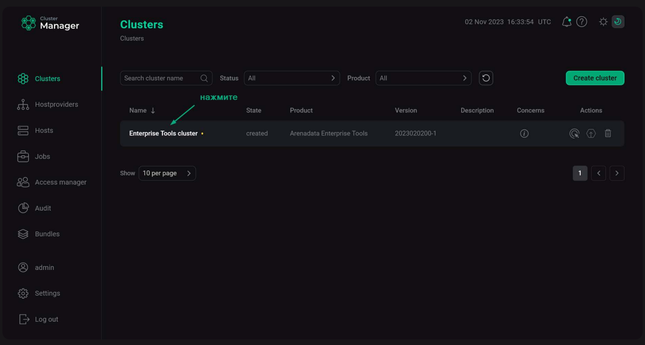
Добавление сервисов

Дарья Барышева

Назад к содержанию

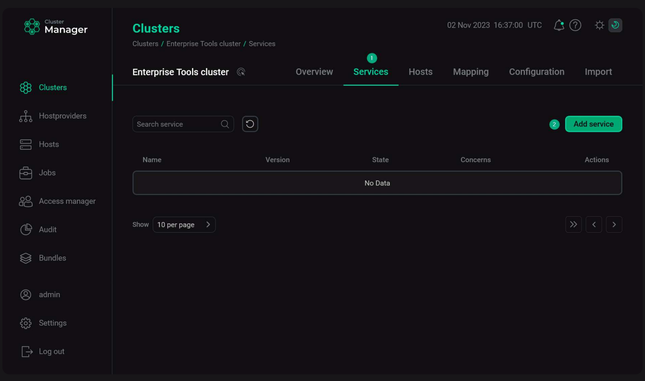
В ADCM **сервис** означает программное обеспечение, выполняющее некоторую функцию. Для добавления сервисов в кластер:

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



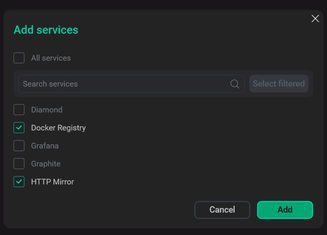
Выбор кластера

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



Переход к добавлению сервисов

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



Выбор сервисов

Краткое описание доступных сервисов приведено ниже.

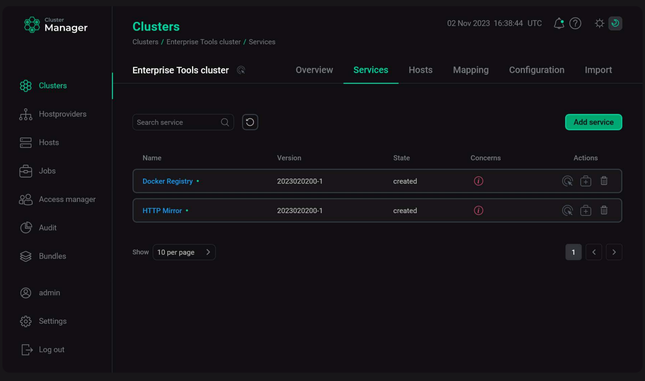
Сервисы, которые необходимо добавить в кластер Enterprise Tools

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***.



Результат успешного добавления сервисов в кластер

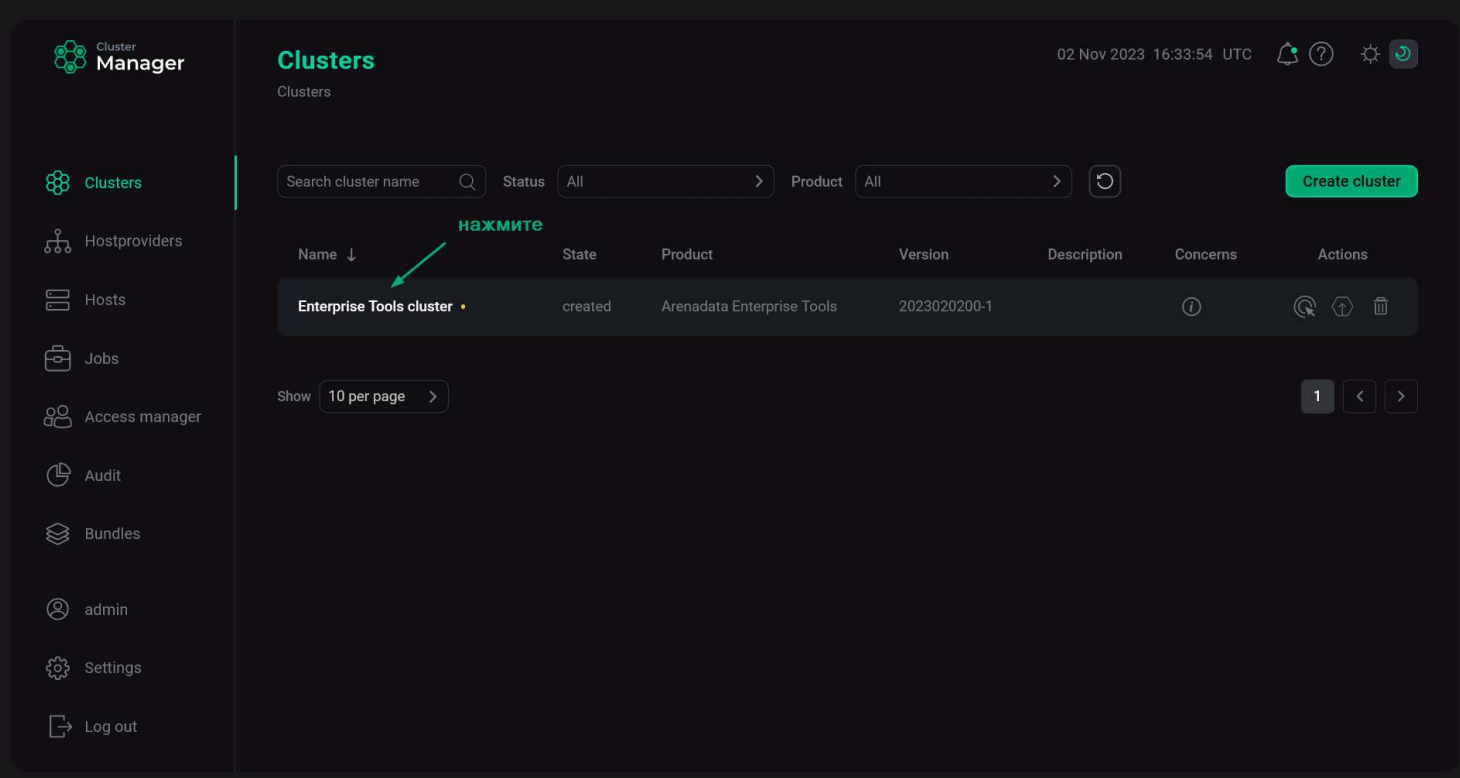
Добавление хостов в кластер

Дарья Барышева

Назад к содержанию

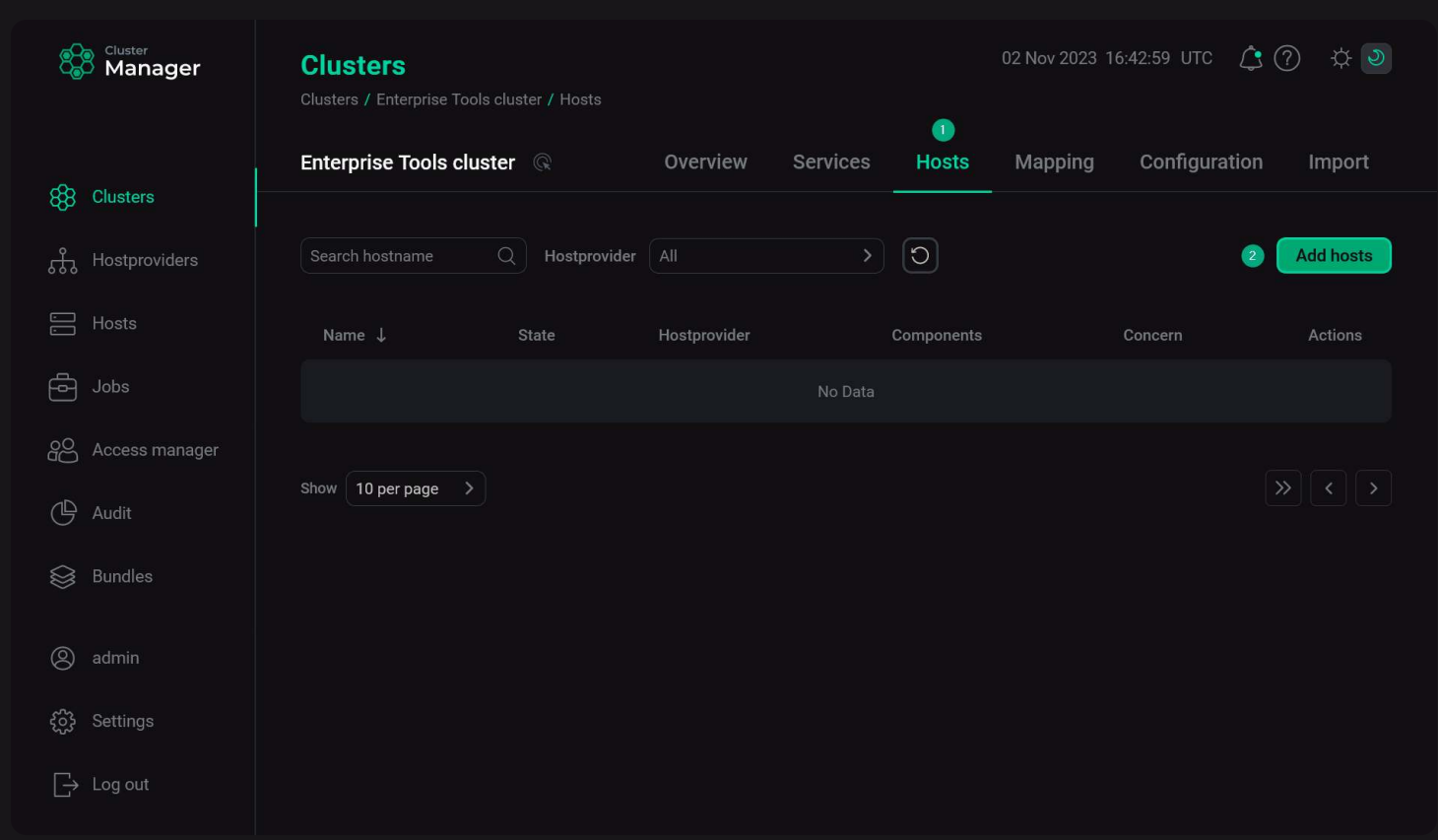
Для добавления хостов в кластер:

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



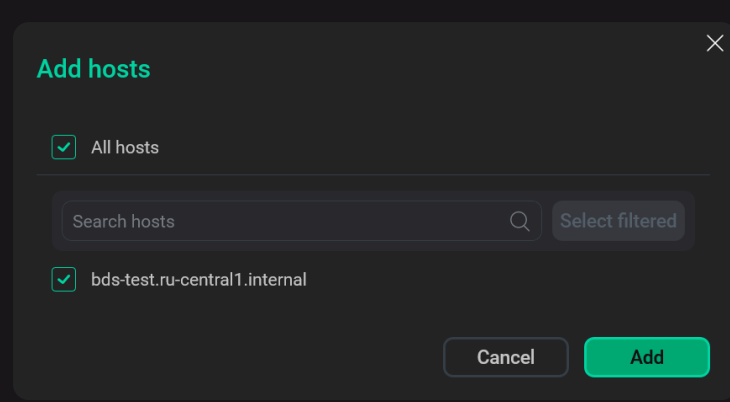
Выбор кластера

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



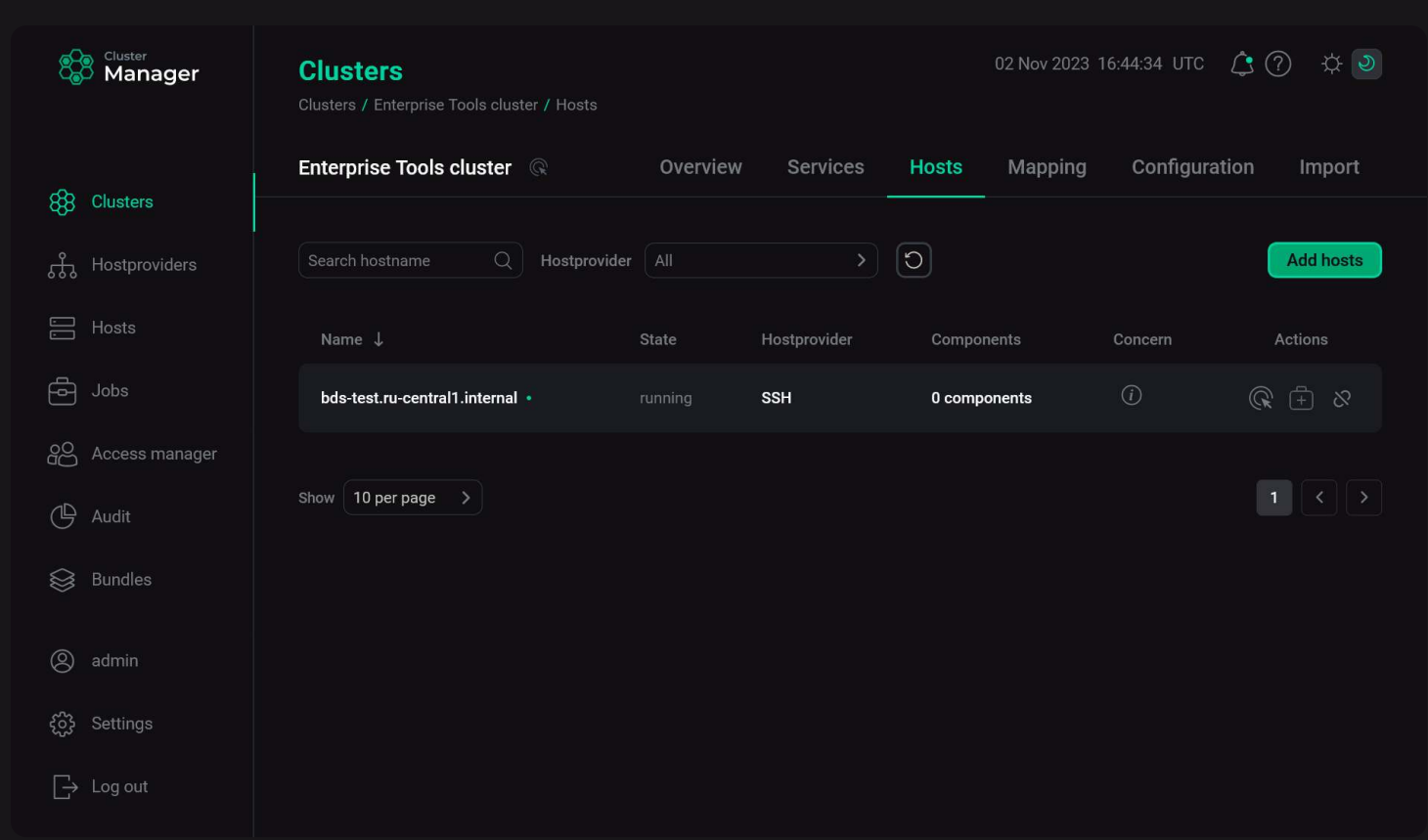
Переход к добавлению хостов

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



Выбор хостов

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



Результат успешного добавления хостов в кластер

Добавление компонентов

Дарья Барышева

Назад к содержанию

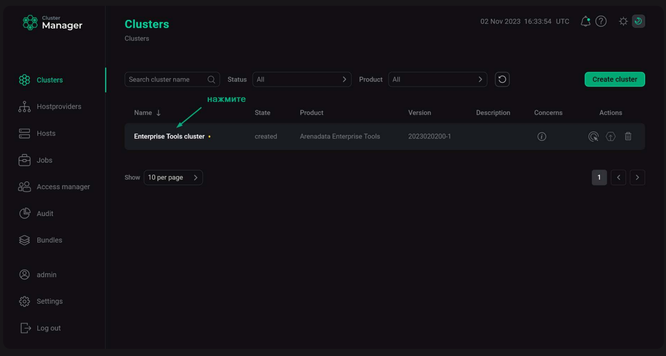
В ADCM **компонент** означает часть сервиса, которая должна быть развернута на одном или нескольких хостах кластера.

ВАЖНО

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

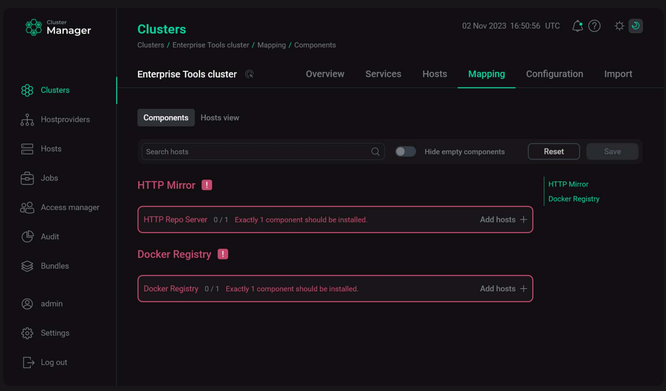
Для добавления компонентов на хосты кластера:

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



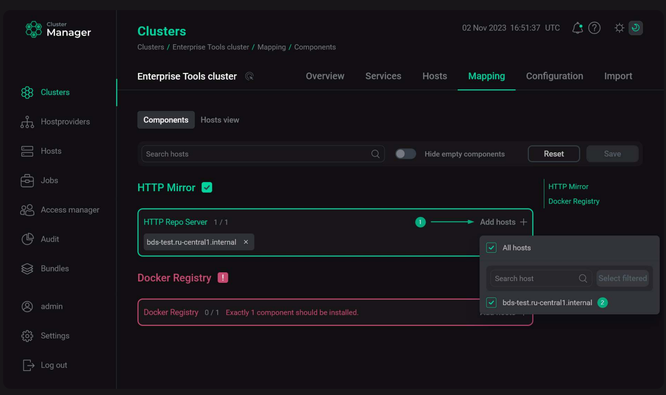
Выбор кластера

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



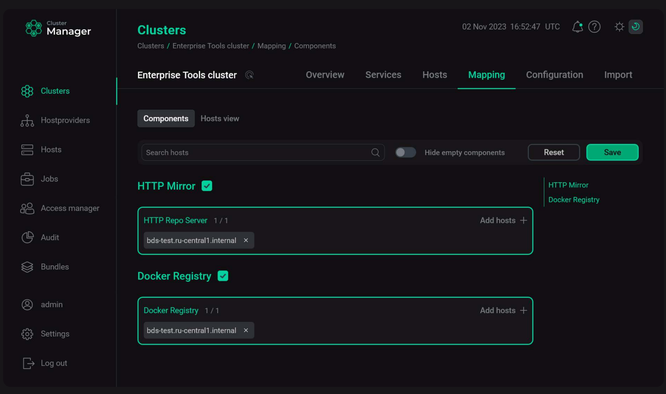
Переход к распределению сервисных компонентов

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



Выбор хостов для компонента

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



Сохранение распределения компонентов

Установка кластера

Дарья Барышева

Содержание

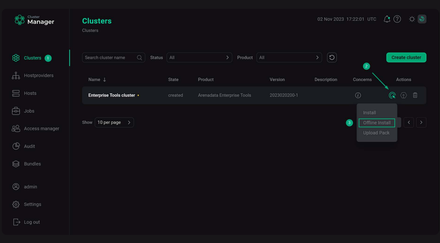
Назад к содержанию

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

Шаг 1. Запуск установки

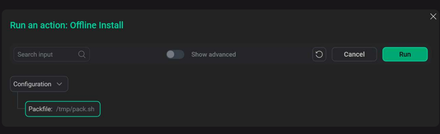
Для установки кластера Enterprise Tools выполните следующие шаги:

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



Переход к установке кластера

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

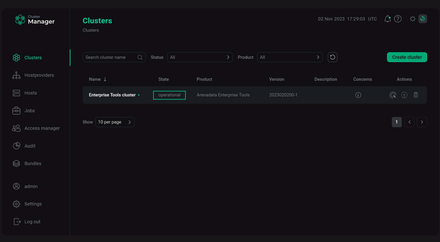


Подтверждение установки

3. Дождитесь завершения процесса установки.

ПРИМЕЧАНИЕ

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

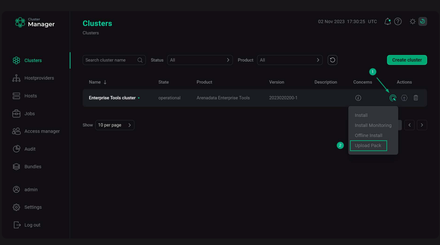


Процесс установки завершен

Шаг 2. Загрузка пакетов для offline-установки

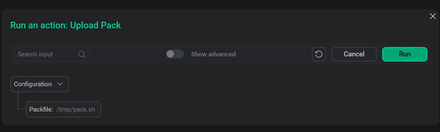
В дополнение к установке сервисов ET требуется загрузка всех установочных файлов и образов в инициализированный http-репозиторий и Docker Registry на хосте, где развернут кластер ET. Это важно для последующего offline-развертывания продуктов Arenadata. Для загрузки выполните шаги:

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



Переход к загрузке пакетов

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



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 ADO.

```
total 0
drwxr-xr-x 5 root root 45 Sep 16 13:35 .
drwxr-xr-x 3 root root 28 Sep 16 13:35 ..
drwxr-xr-x 4 root root 38 Sep 16 13:35 ADO
drwxr-xr-x 4 root root 28 Sep 16 13:35 ADM
drwxr-xr-x 3 root root 20 Sep 16 13:35 zookeeper
```

Создание кластера

Елена Костюченко, 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.

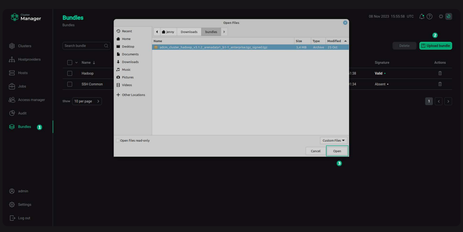
Шаг 1. Получение Enterprise-версии бандла

Пожалуйста, запросите бандл у команды поддержки Arenadata.

Шаг 2. Загрузка бандла в ADCM

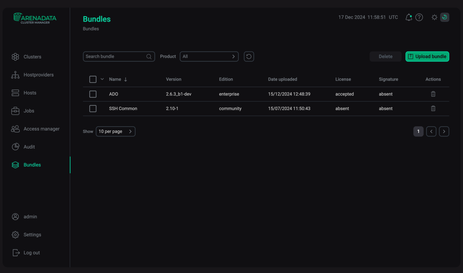
Чтобы загрузить бандл в ADCM:

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



Переход к загрузке бандла

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



Результат успешной загрузки бандла

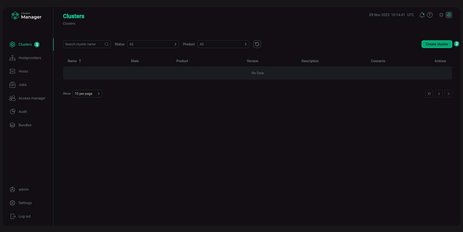
Шаг 3. Создание кластера на базе загруженного бандла

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.

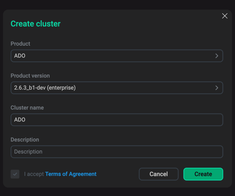
Для создания кластера необходимо выполнить шаги:

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



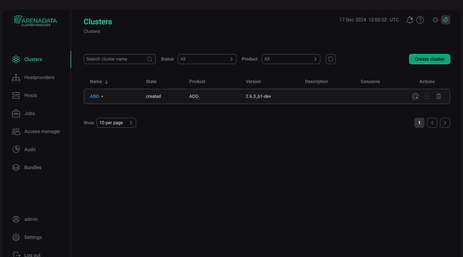
Переход к созданию кластера

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***.



Создание кластера

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



Результат успешного добавления кластера

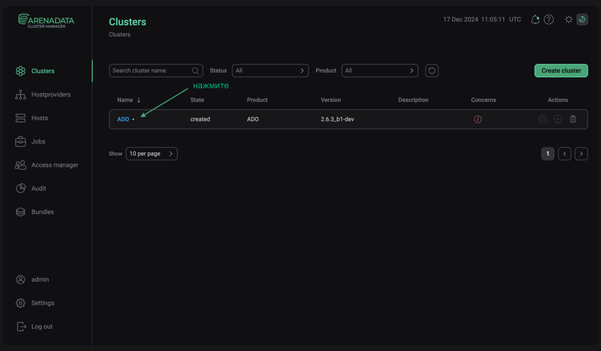
Добавление сервисов

Елена Костюченко, Дарья Барышева

Назад к содержанию

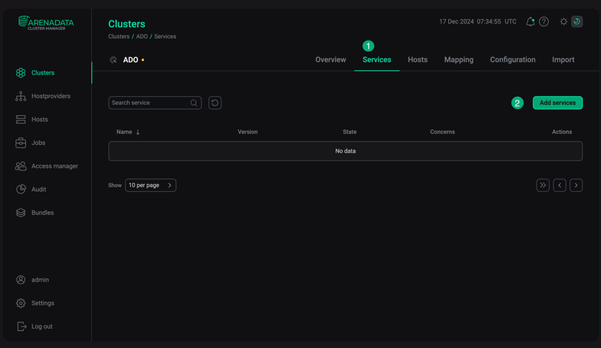
В ADCM **сервис** означает программное обеспечение, выполняющее некоторую функцию. Примеры сервисов в кластерах ADO: Airflow, ADPG и другие. Для добавления сервисов в кластер:

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



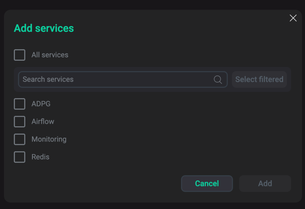
Выбор кластера

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



Переход к добавлению сервисов

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



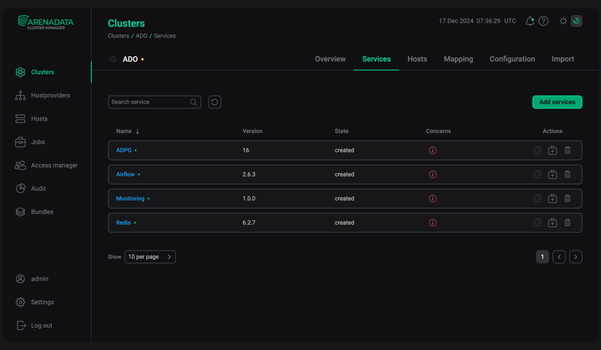
Выбор сервисов

Краткое описание доступных сервисов приведено ниже.

Сервисы, которые могут быть добавлены в кластер ADO

Сервис	Назначение
Airflow	Сервис для создания, планирования и мониторинга процессов, представленных в форме ориентированных ациклических графов (Directed Acyclic Graphs, DAG). Используется в кластерах Arenadata Orchestrator для построения процессов ETL/ELT
Monitoring	Сервис, добавляемый в случаях, когда запланирована настройка мониторинга ADO через ADCM
Redis	Redis — это хранилище данных NoSQL с открытым исходным кодом, которое размещается в памяти и может быть использовано как кеш приложений или база данных с быстрым откликом
ADPG	Arenadata Postgres (ADPG) — коммерческий дистрибутив базы данных PostgreSQL с дополнительными функциями Enterprise

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



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.

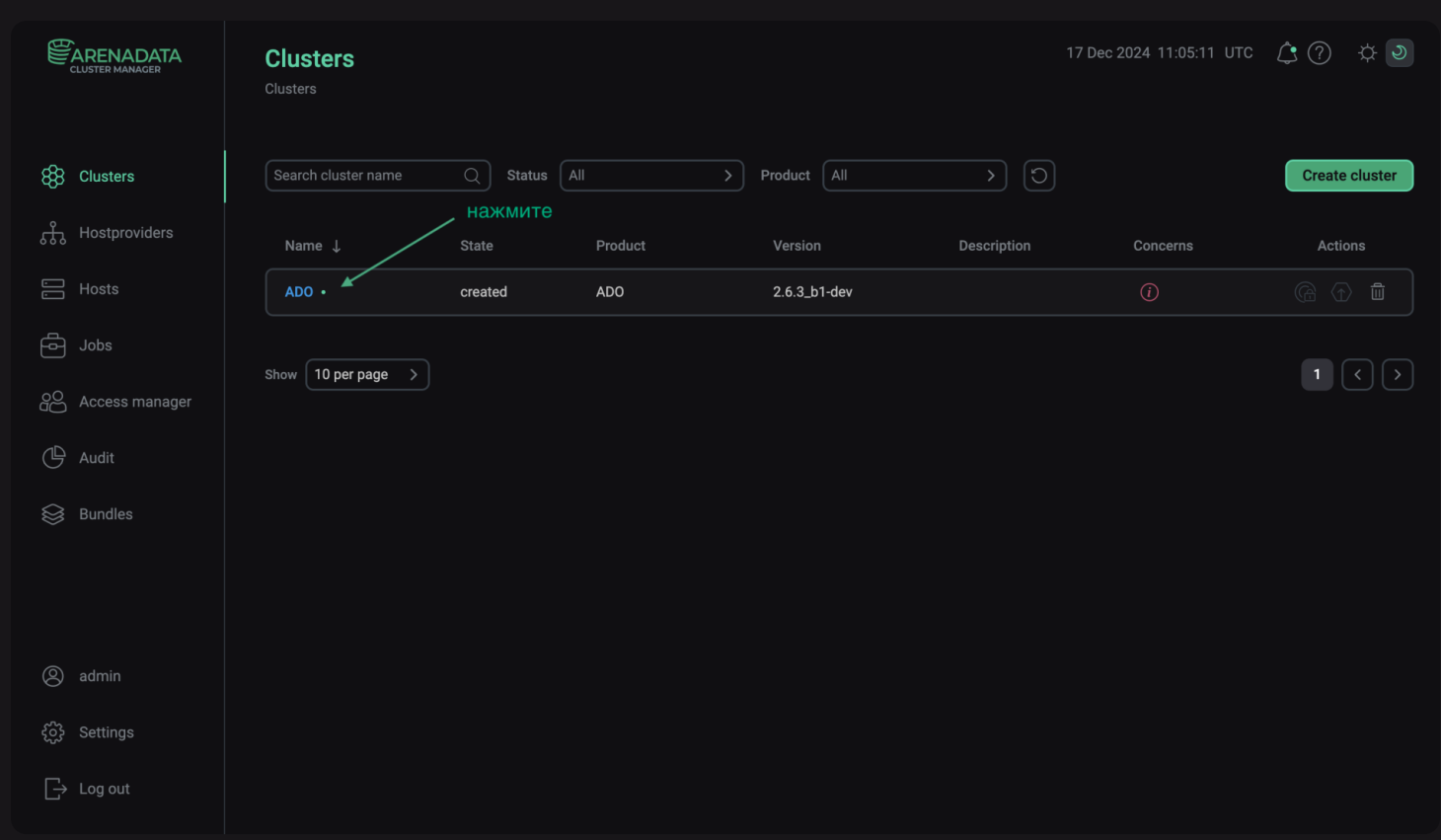
Добавление хостов в кластер

Елена Костюченко, Дарья Барышева

Назад к содержанию

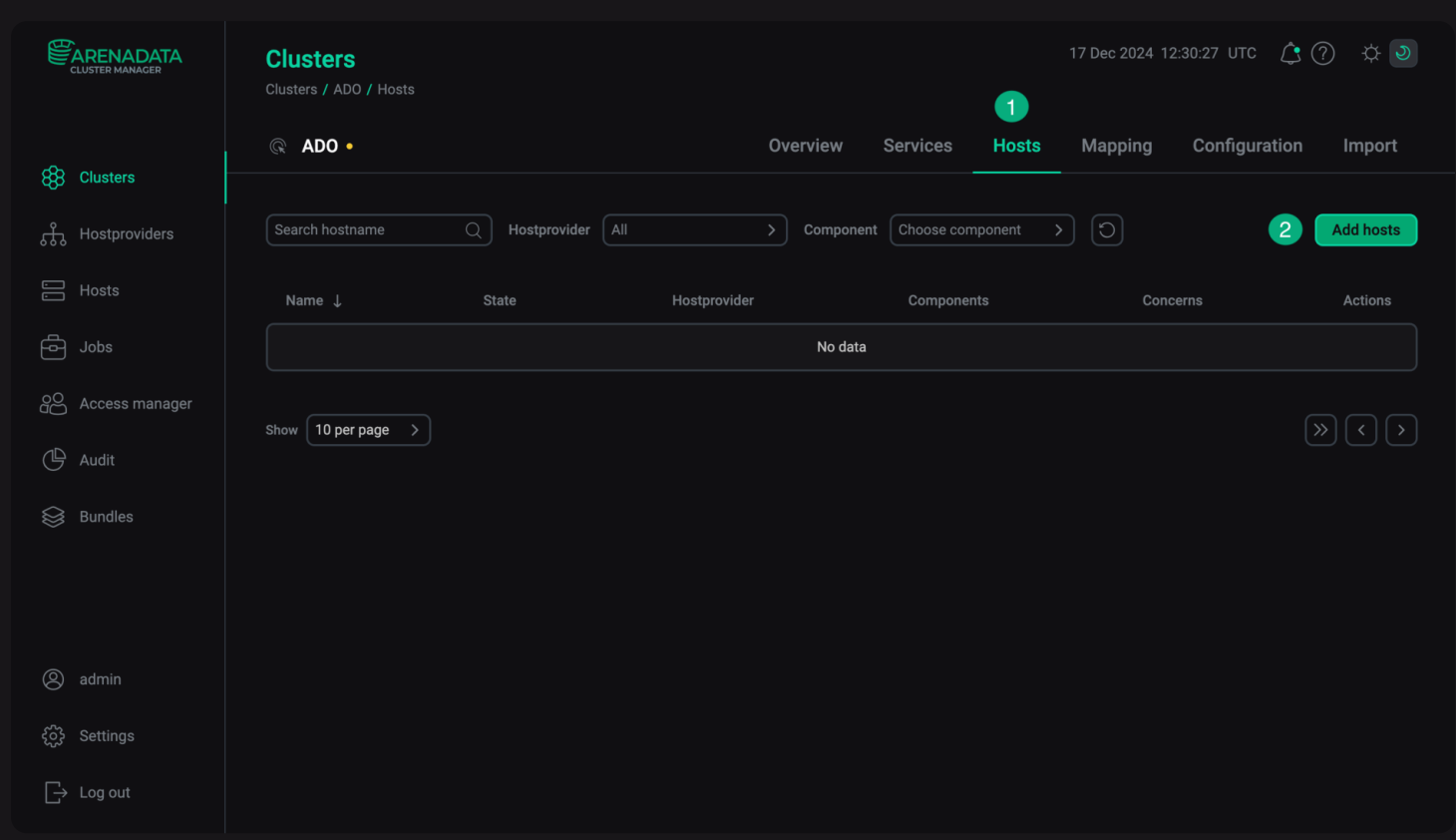
Для добавления хостов в кластер:

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



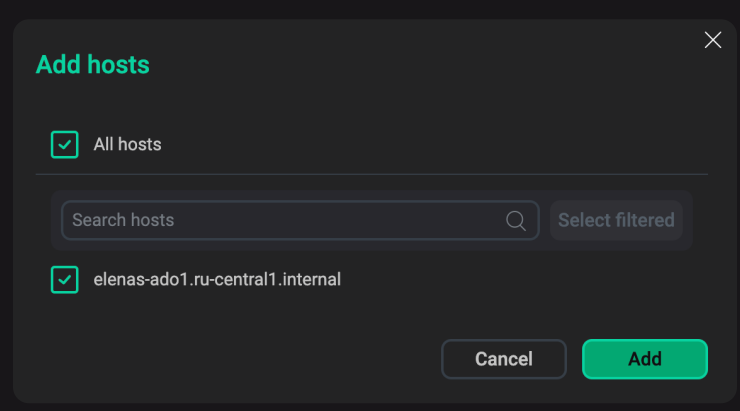
Выбор кластера

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



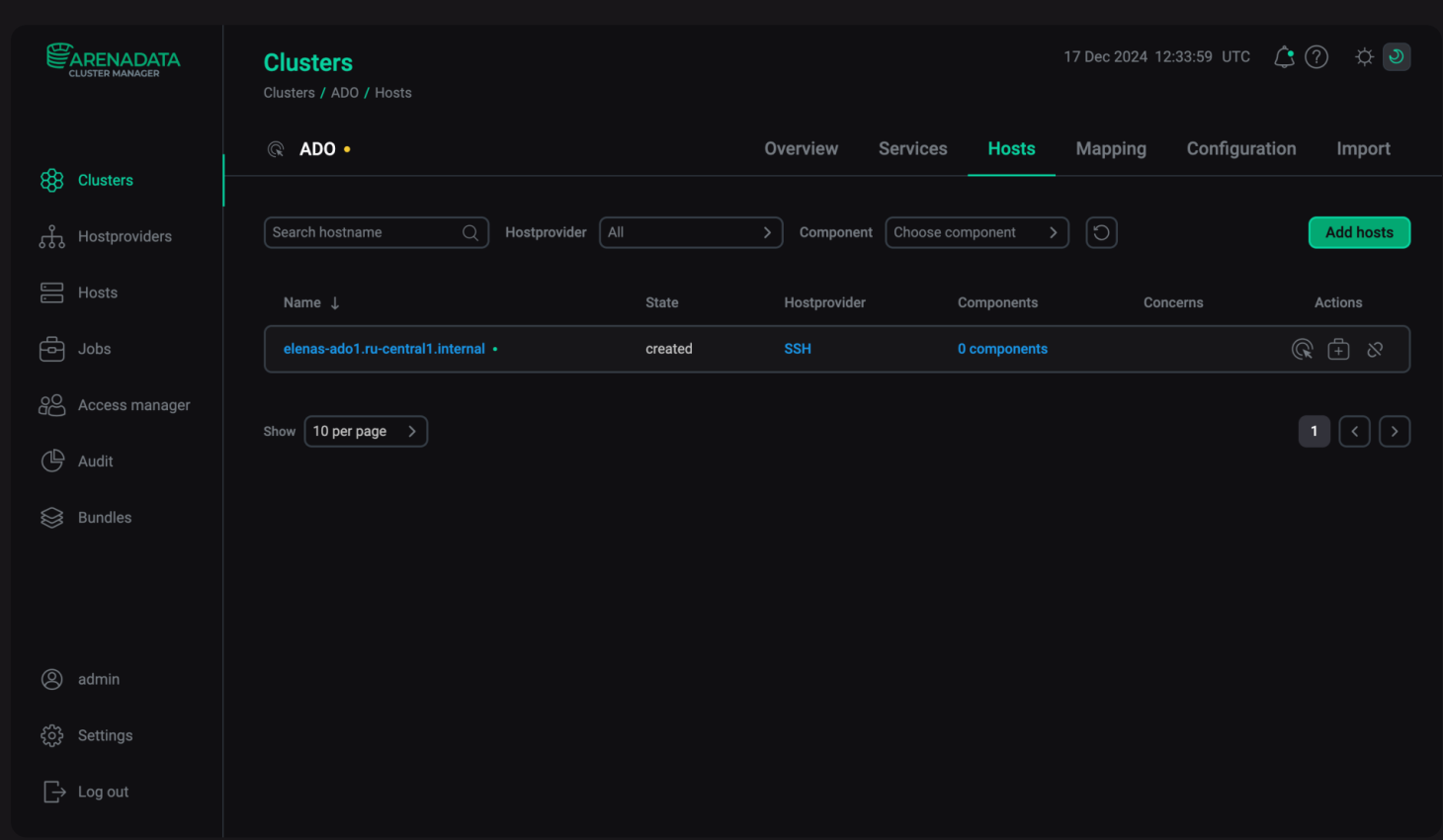
Переход к добавлению хостов

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



Выбор хостов

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



Результат успешного добавления хостов в кластер

Добавление компонентов

Елена Костюченко, Дарья Барышева

Назад к содержанию

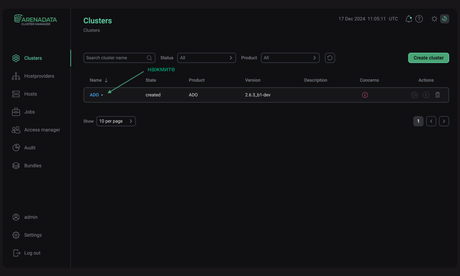
В ADCM **компонент** означает часть сервиса, которая должна быть развернута на одном или нескольких хостах кластера. Например, сервис Airflow состоит из таких компонентов, как Airflow Server, Airflow Scheduler, Airflow Worker и другие.

ВАЖНО

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

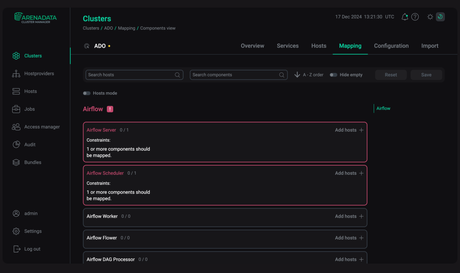
Для добавления компонентов на хосты кластера:

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



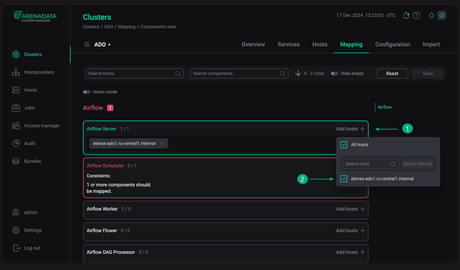
Выбор кластера

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



Переход к распределению сервисных компонентов

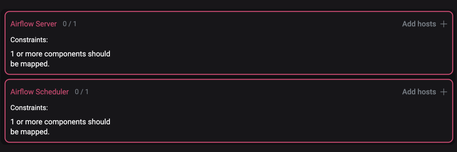
3. Распределите все компоненты между хостами кластера. Для назначения хостов компоненту нажмите ***Add hosts*** и выберите хосты в открывшемся окне.



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.



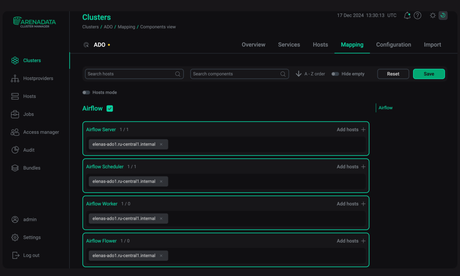
Обязательные компоненты

ВНИМАНИЕ

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

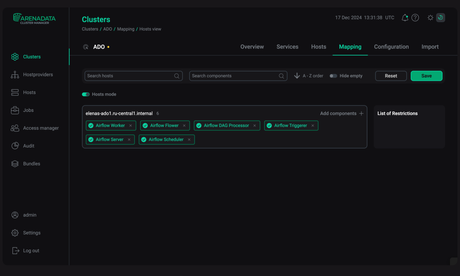
Все компоненты в кластере ADO могут быть установлены на одном хосте. Вы можете добавить несколько хостов в кластер (в зависимости от требований к производительности) и распределить компоненты сервисов между ними.

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



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.



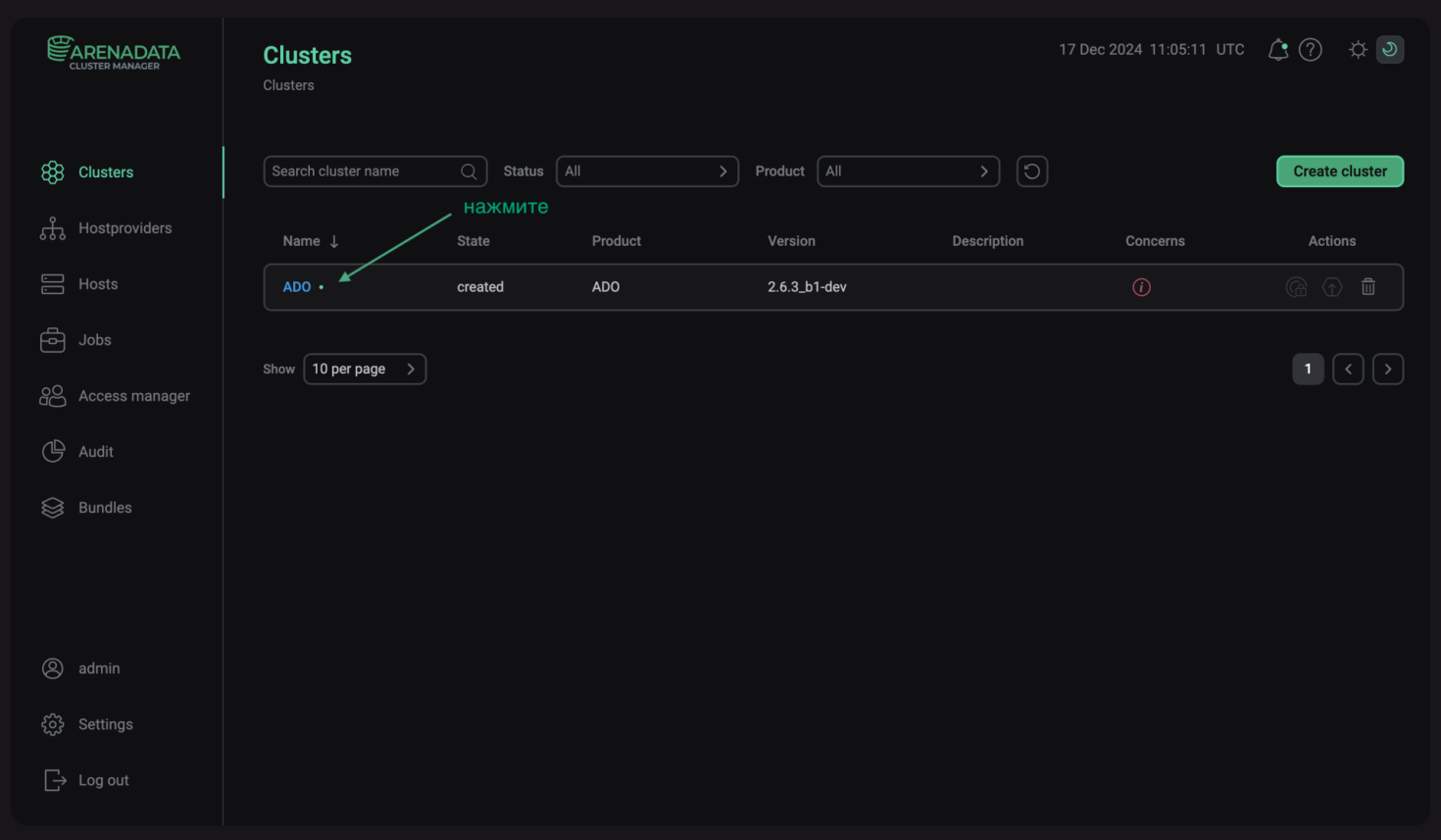
Распределение по хостам

Настройка сервисов

Елена Костюченко, Дарья Барышева

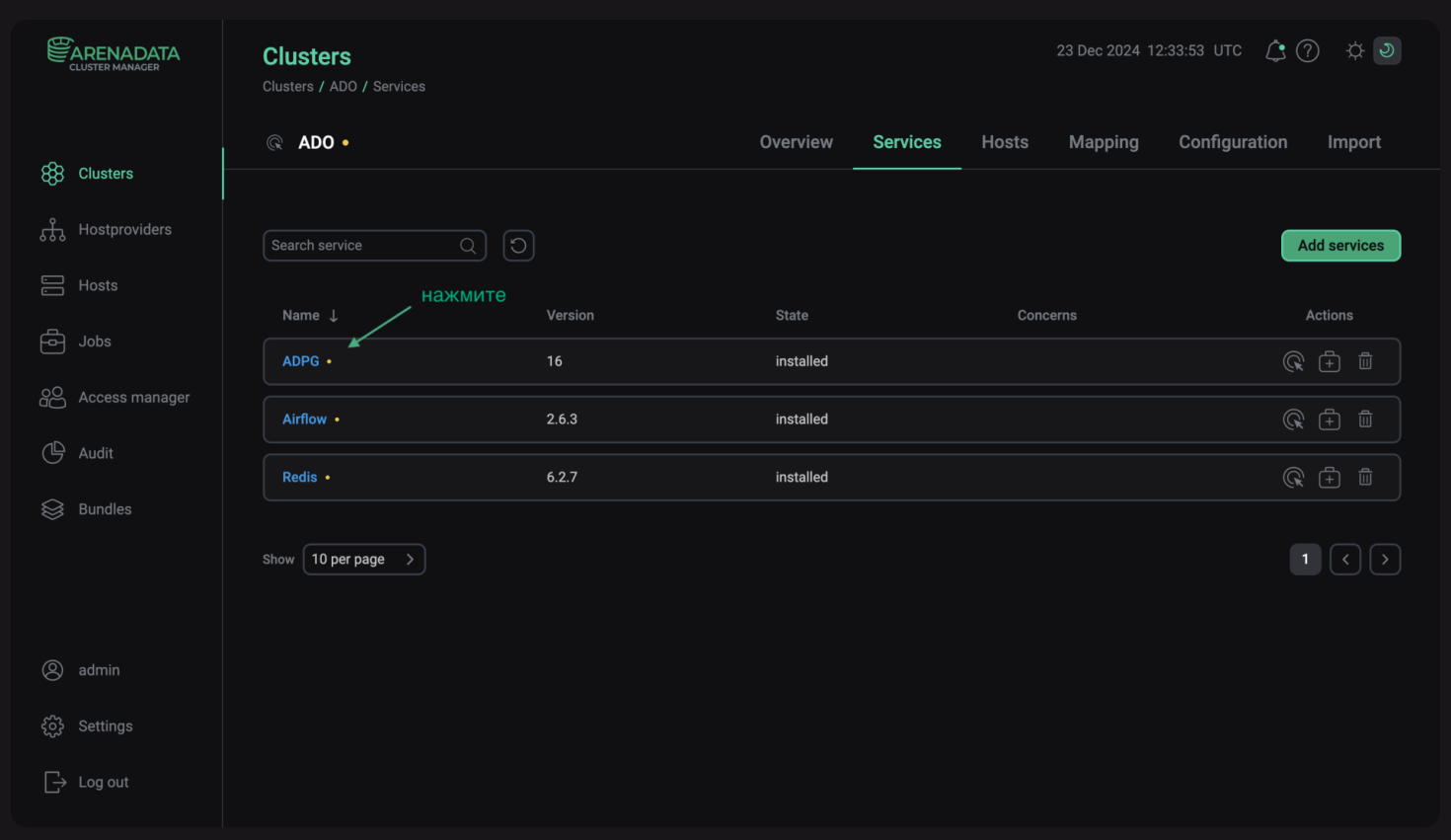
Назад к содержанию

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



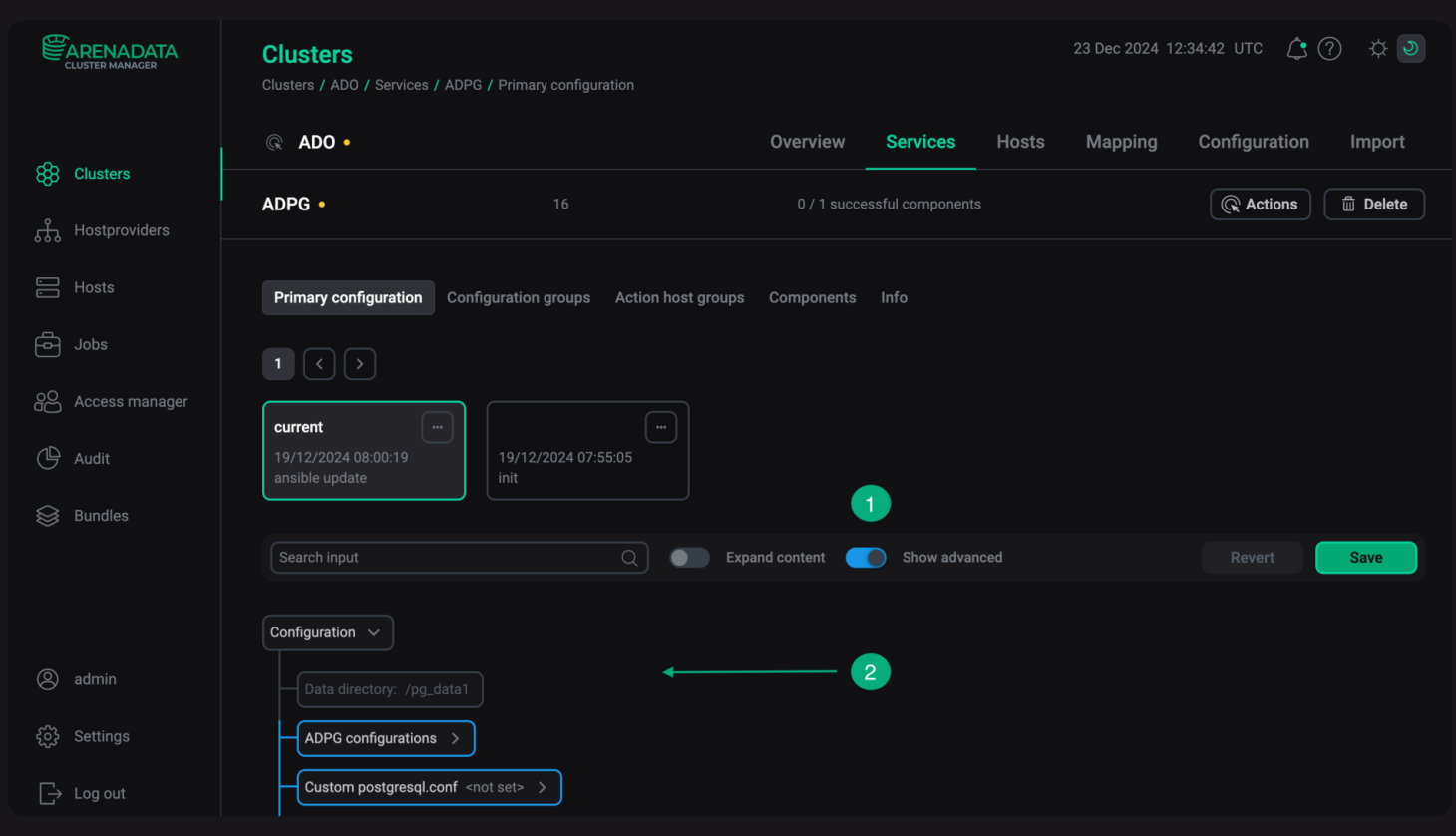
Выбор кластера

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



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***.



Настройка сервиса

ПРИМЕЧАНИЕ

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

Настройка кластера

Елена Костюченко, Дарья Барышева

Содержание

Назад к содержанию

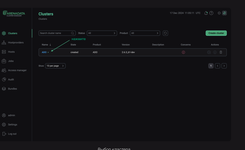
Primary configuration
Configuration groups
Ansible settings

ВАЖНО

В большинстве случаев кастомная настройка кластера не требуется – значение 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**.



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 параметры перечислены ниже:

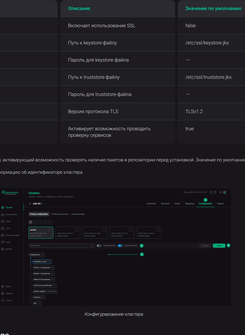
- Раздел ***Reliability control***. Отображается, если активирован переключатель **Show advanced**. Раздел содержит следующие настройки:
  - ***retries*** – количество попыток для операций в кластере (по умолчанию: 20);
  - ***delay*** – задержка между попытками в секундах (по умолчанию: 10);
  - ***timeout*** – тайм-аут проверки сетевых портов в секундах (по умолчанию: 60).
- Путь к репозиторию CentOS, Red Hat, ALT Linux & Astra Linux, которые будут использованы в процессе установки в зависимости от операционной системы кластера:
  - ***annadata*** – YUM-репозиторий Annadata;
  - ***annadata_postgres*** – YUM-репозиторий Annadata PostgreSQL;
  - ***prometheus_repo*** – YUM-репозиторий Prometheus.
- Параметр ***DockerRegistry***, указывающий адрес Docker Registry. Значение по умолчанию – *hub.adsw.io* – Docker Registry из репозиториев Annadata.
- Раздел ***Kerberos***, содержащий параметры, перечисленные в таблице ниже. Некоторые параметры становятся видимыми только при установленном флажке **Show advanced**.

Параметр	Описание	Значение по умолчанию
Enable Kerberos	Включает аутентификацию Kerberos	False
Authentication on WEB UIs	Включает аутентификацию Kerberos для веб-интерфейсов	False
SPNEGO Signature Value	Секретный ключ, используемый для генерации и проверки подписи в протоколе HTTP-аутентификации SPNEGO (в формате Base64)	—
Kerberos KDC type	Тип КОС. За дополнительной информацией обратитесь к статье Kerberos	MIT
KDC hosts	Один или несколько хостов КОС с запущенными серверами FreeIPA	—
Realm	Kerberos realm, для подключения к серверу FreeIPA	—
Domains	Один или несколько доменов, ассоциированных с FreeIPA	—
Kadmin server	Хост, на котором работает `kadmin`	—
Kadmin principal	Имя принципала, используемое для подключения через `kadmin`, например `admin@RU-CENTRAL1.INTERNAL`	—
Kadmin password	Пароль администратора IPA	—
Keytabs directory	Каталог с keytab-файлами	/etc/security/keytabs
Additional realms	Дополнительные Kerberos realms	—
Delay between kinit invocation attempts	Задержка между попытками вызова `kinit`	5
Number of retries for kinit invocation attempts	Количество попыток вызова `kinit`	10
Trusted Active Directory server	Сервер Active Directory для обеспечения односторонних доверительных отношений между realms из MIT Kerberos KDC	—
Trusted Active Directory realm	Active Directory realm для обеспечения односторонних доверительных отношений между realms из MIT Kerberos KDC	—
Custom krb5.conf	Активирует добавление кастомных параметров в файл krb5.conf	False
krb5.conf	Дополнительные параметры для записи в файл krb5.conf	—
Admin DN	Полное отличительное имя (DistinguishedName) пользователя admin с правами `create/modify/delete/pwchange` для создания/изменения учетных записей в целевом Organizational Unit	—
LDAP URL	LDAP URL, который состоит из префикса протокола `ldap://`, или `ldaps://`, имени хоста или IP-адреса и порта AD-сервера. Например `ldaps://192.168.4.2:636`	—
Container DN	Отличительное имя (DistinguishedName) контейнера	—
TLS CA certificate Path	Путь к сертификату CA в файловой системе хоста, который предварительно размещен или будет записан из поля **TLS CA certificate**	—
TLS CA certificate (optional)	Сертификат CA, который записывается на хост по пути из **TLS CA certificate Path** во время исполнения действия **Manage → Kerberos**. Сертификат не создается в ADCM	—
Custom ldap.conf	Активирует добавление кастомных параметров в файл ldap.conf	False
ldap.conf	Дополнительные параметры для записи в файл ldap.conf	—
ipaClient No NTP Autoconf	Отключает автоконфигурирование NTP во время установки клиента IPA	False
ipaClient No DNS Lookup	Отключает поиск DNS для сервера FreeIPA во время установки клиента IPA	True
Custom smb.conf	Использование кастомного файла конфигурации smb.conf	False
smb.conf	Кастомный файл конфигурации smb.conf	—
Set up Kerberos utils	Активирует возможность устанавливать и удалять клиенты и утилиты Kerberos. Данный параметр учитывается в действиях ***Expand*** и ***Install***	True
Configure Kerberos on hosts	Активирует возможность конфигурировать кластер, включая krb5.conf, ldap.conf	True
Set up principals and keytabs	Активирует возможность создавать, пересоздавать и удалять принципалов и keytab-файлы. Пароли для принципалов генерируются случайным образом перед каждым запуском. Данный параметр учитывается в действиях ***Expand*** и ***Install***. Бандл ADCM устанавит сигнал и разрешения для keytab-файлов только если этот флаг установлен. В случае отсутствия прав администратора пользователь должен предоставить подготовленный keytab-файл с правильно установленными сигнал и разрешениями	True
Configure services and clients	Активирует возможность обновлять конфигурации серверов и клиентов	True
Run service checks	Активирует возможность проводить проверку сервисов	True

- Раздел ***SSL*** с настройками, перечисленными в таблице ниже.

Параметр	Описание	Значение по умолчанию
Enable SSL	Включает использование SSL	false
Keystore path	Путь к keystore-файлу	/etc/ssl/keystore.jks
Keystore password	Пароль для keystore-файла	—
Truststore path	Путь к truststore-файлу	/etc/ssl/truststore.jks
Truststore password	Пароль для truststore-файла	—
TLS Version	Версия протокола TLS	TLSv1.2
Run service checks	Активирует возможность проводить проверку сервисов	true

- Параметр ***Precheck packages***, активирующий возможность проверять наличие пакетов в репозиториях перед установкой. Значение по умолчанию – `false`.
- ***Cluster UUID*** отображает информацию об идентификаторе кластера.

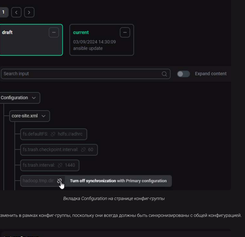


Конфигурирование кластера

Configuration groups

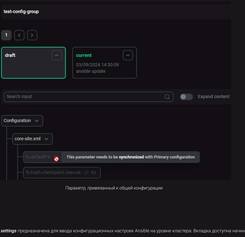
Вкладка ***Configuration → Configuration groups*** предназначена для настройки конфиг-групп кластера.

Чтобы настроить параметры в конфиг-группе, выберите параметры, которые вы хотите настроить иначе, нажмите иконку **Turn off synchronization**, отредактируйте их и нажмите **Save**.



Вкладка Configuration на странице конфиг-группы

Некоторые параметры нельзя изменить в рамках конфиг-группы, поскольку они всегда должны быть синхронизированы с общей конфигурацией.



Параметр, привязанный к общей конфигурации

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

Импорт настроек ET

Елена Костюченко, Дарья Барышева

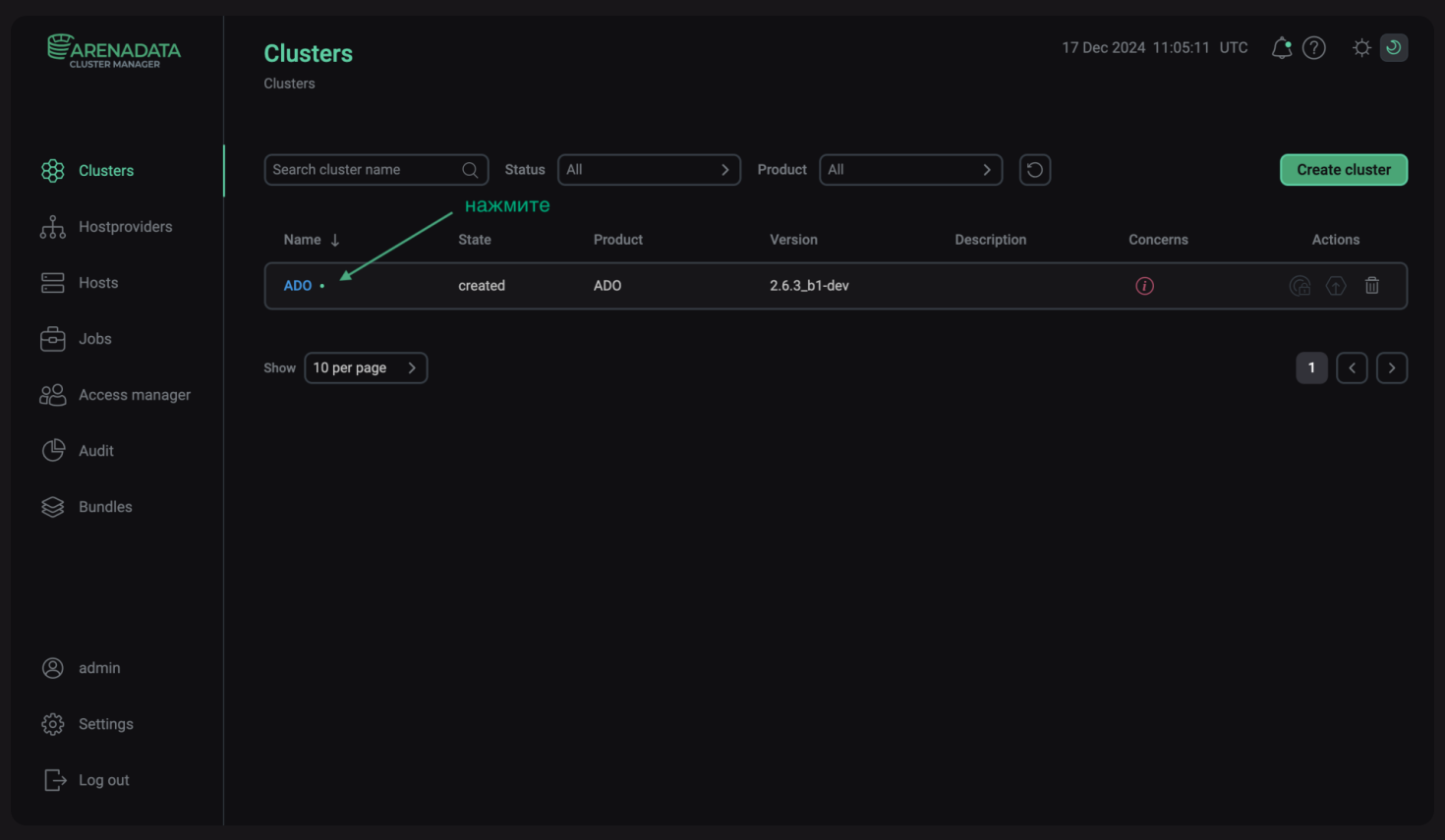
Назад к содержанию

ВАЖНО

Этот шаг необходим только в случае offline-установки.

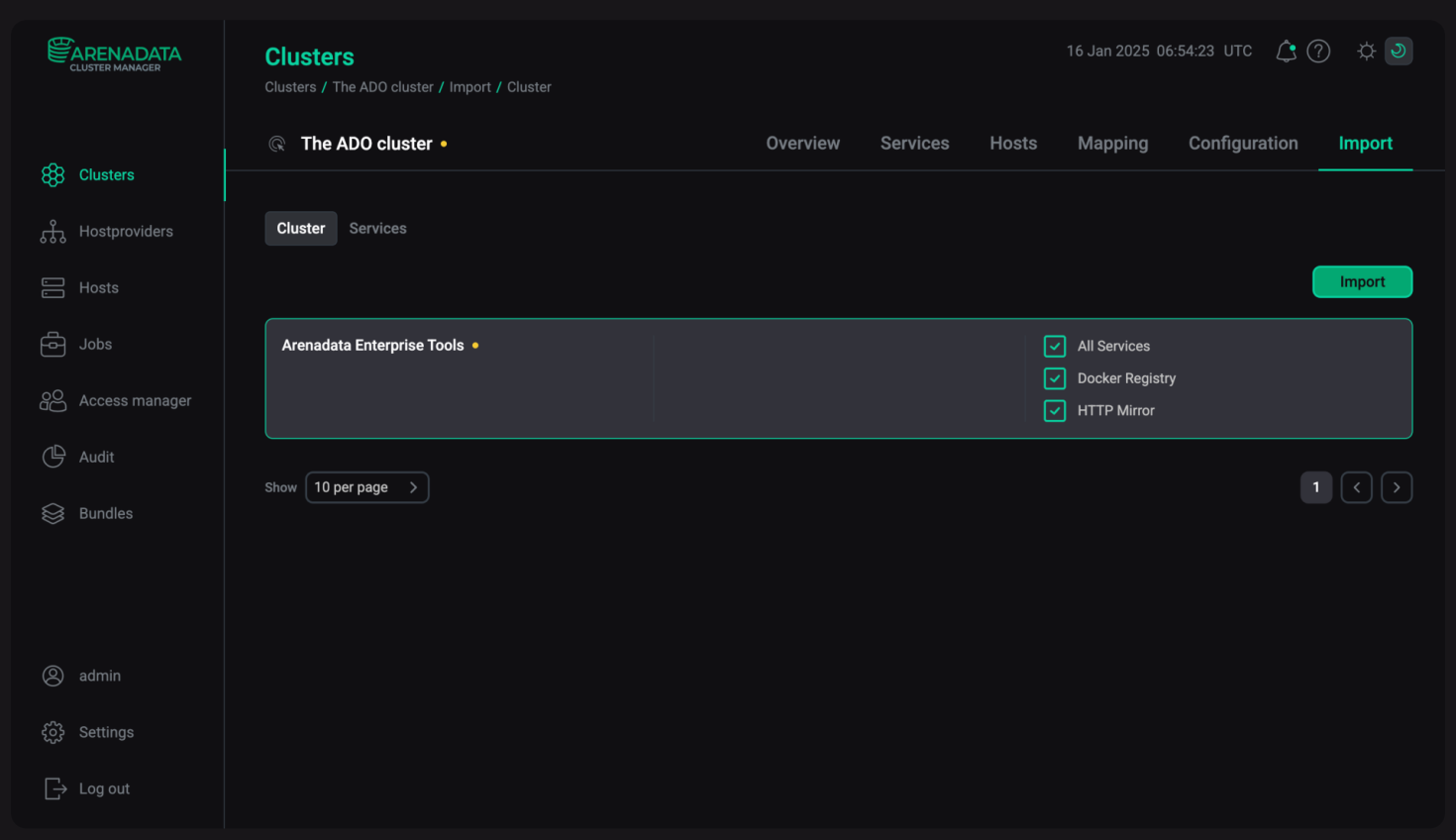
Для установки кластера в режиме offline требуется интегрировать его с ранее созданным кластером Enterprise Tools (ET):

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



Выбор кластера

2. Откройте вкладку ***Import*** на странице кластера.
3. Выберите сервисы ***Docker Registry*** и ***HTTP Mirror*** предварительно созданного кластера ET.
4. Нажмите ***Import***.



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

Елена Костоломко, Дарья Борщакова

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.

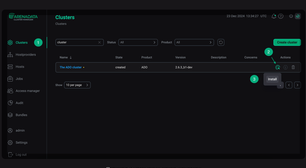
ВАЖНО

Установка должна производиться на чистые хосты без предустановленного подобного ПО.

Установка всего кластера

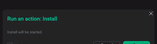
Чтобы установить все сервисы кластера за одну операцию, выполните шаги:

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



Переход к установке всего кластера

2. Подтвердите действие в открывшемся окне.



Подтверждение действия

3. Дождитесь завершения установки.

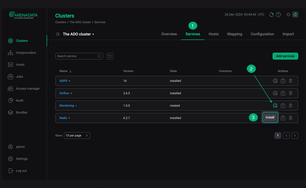
Установка отдельных сервисов

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

ВНИМАНИЕ

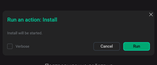
Обратите внимание, что сервисы устанавливаются по очереди друг за другом. Не пытайтесь установить следующий сервис до окончания установки предыдущего.

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.



Переход к установке отдельных сервисов

2. Подтвердите действие в открывшемся окне.

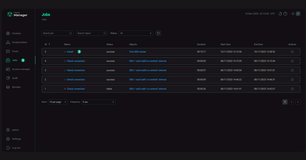


Подтверждение действия

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

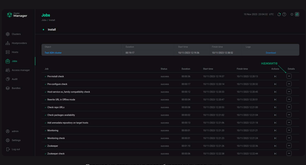
Шаг 2. Просмотр результатов установки

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



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***. Откроется окно с шагами выполнения задачи. Кликните многоточие в столбце ***Details***, чтобы получить дополнительную информацию об определенном шаге.

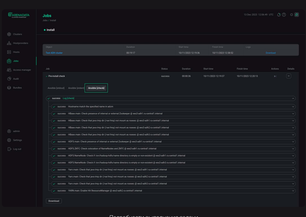


Переход на страницу с подробностями выполнения

Страница задачи содержит два основных раздела: ***ansible [stdout]*** и ***ansible [stderr]***. Это технические логи, содержащие информацию из стандартных I/O потоков stdout/stderr.

В дополнение к ним может выводиться третий опциональный раздел ***ansible [check]***. Он содержит краткие описания наиболее часто возникающих ошибок. По сравнению с предыдущими разделами, эти описания, как правило, более простые и конкретные.

При возникновении ошибок в процессе установки рекомендуется изучить логи всех трех разделов.

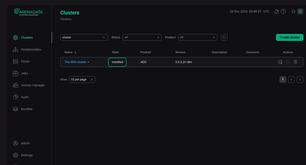


Подробности выполнения задачи

Шаг 3. Проверка статуса кластера после установки

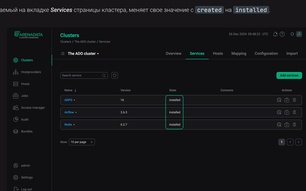
В результате успешной установки происходит обновление статуса кластера и его сервисов:

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



Статус кластера после успешной установки

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



Статус сервисов кластера после успешной установки

Установка мониторинга

Версия 6.0.0 от 15.05.2025

Содержание

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

Перед установкой

Версия мониторинга 4.60.1(x64 сервер Prometheus), которая реализует сбор и хранение метрик мониторинга платформы АОС и такие подключения клиентских мониторингов как мониторинг Grafana для визуализации и анализа метрик. В данной статье описываются шаги, необходимые для полноценной этого сервиса.

Обзор

При добавлении сервиса Monitoring в кластер АОС на всех узлах устанавливается Node Exporter — этот компонент, считывающий с хостов системные метрики, которые будет собирать Prometheus. Prometheus, сервер которого также будет развернут на одном сервере АОС (по умолчанию будет установлен в формате Prometheus as service и запускаться, управляемый в systemd systemctl, переменные окружения).

Prometheus, в зависимости от конфигурации, может использовать для получения метрик данных для сетевых экспортеров АОС, которые, в свою очередь, получают метрики с АОС.

Prometheus, которые также будет развернут на одном сервере АОС, по умолчанию будет установлен в формате Prometheus as service и запускаться, управляемый в systemd systemctl, переменные окружения). Prometheus, в зависимости от конфигурации, может использовать для получения метрик данных для сетевых экспортеров АОС, которые, в свою очередь, получают метрики с АОС.

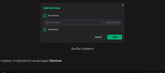
Для полноценного мониторинга платформы АОС, для визуализации и анализа метрик, в данной статье описываются шаги, необходимые для полноценной этого сервиса.

Шаг 1. Добавление сервиса

1. В интерфейсе АОС(по умолчанию Grafana) **Dashboards** в элементе по имени кластера АОС, Затем на открывшейся странице кластера перейдите на вкладку **Services** и нажмите **Add services**

Переход к добавлению сервиса

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



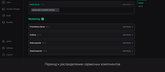
Выбор сервиса

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

Результат успешного добавления сервиса в кластер

Шаг 2. Добавление компонентов

1. На странице кластера открыта вкладка **Mapping**, чтобы перейти к распределению серверных компонентов между системами кластера.



Переход к распределению серверных компонентов

Каждый компонент необходимо установить на один любой хост кластера.

Компоненты сервиса Monitoring

Компонент	Описание
Prometheus Server	Развертывается в АОС сервер Prometheus, который выполняет следующие функции: • прием метрик для всех серверов, собираемых метриками на хосте • хранилище для метрик кластера • генератор оповещений на основе собираемых метрик
Grafana	Развертывает предварительно настроенную Grafana с дашбордами, оповещениями и рендерером
Node Exporter	Считывает с хостов системные метрики, которые будет собирать Prometheus
Statsd-exporter	Перевод метрики из формата Statsd в метрики формата Prometheus с помощью настраиваемых правил

2. Нажмите кнот каждому компоненту сервиса Monitoring — нажмите **Add hosts** и выберите нужный хост в открывшемся окне.

Выбор хоста для компонента

ВНИМАНИЕ

Компонент Prometheus Server на development установлен на хосте с АОС - используйте для него отдельный хост. Иначе при выходе из строя хоста АОС всему кластеру Monitoring будет потеряна информация о состоянии/нагрузке приложения и будет создаваться.

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

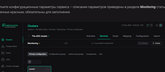
Сохранение распределения компонентов

NOTE

When changing the topology of services/configurations related to monitoring, it is necessary to restart the Prometheus component in the Monitoring service.

Шаг 3. Настройка сервиса

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



Переход к настройке сервиса

2. В открывшемся окне заполните конфигурационные параметры сервиса — описание параметров приведены в разделе **Monitoring** статьи Конфигурационные параметры. Поля, подлежащие изменению, обозначены для заполнения.

Настройка сервиса Monitoring

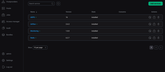
После указания всех необходимых параметров нажмите **Save**.

Шаг 4. Установка сервиса

1. На вкладке **Services** в списке сервисов Monitoring нажмите на иконку **Actions** в столбце **Actions** и запустите действие **Install**

Переход к установке сервиса

2. Подтвердите действие в открывшемся окне

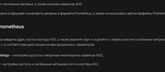


Подтверждение действия

3. Дождитесь завершения установки и убедитесь, что статус сервиса изменился на **installed** — **installed**

Процесс установки завершен

Чтобы увидеть, прошло ли выполнение установки сервиса и проконтролировать ошибки в случае их возникновения, выберите пункт **Jobs** в меню навигационного меню и ознакомьте по каждой задачи **Install** и **Jobs**.



Страница задачи Install

Шаг 5. Проверка результатов

Сервис Monitoring запускается автоматически после установки. По завершении установки необходимо убедиться в следующем:

- Метрики собираются на всех хостах кластера, а не только с хостов, на которых развернуты компоненты сервиса мониторинга
- С системы мониторинга Grafana получаются метрики с сервиса Prometheus АОС

Чтобы проверить обе пункта, зайдите в браузере посмотреть метрики в формате Prometheus, а также использовать веб-интерфейсы Prometheus и Grafana.

Данные в формате Prometheus

1. В адресной строке браузера введите адрес хоста кластера АОС и затем укажите порт в формате с перечислены или системных метриках, которые отправлены на Prometheus Server и в **Prometheus** отображаются в виде отдельных временных серий:
   - **АОС service metric settings** — настройки доступа к метрикам мониторинга сервисов АОС
   - **Node Exporter settings** — настройки доступа к системным метрикам хостов АОС

   Пример адреса для просмотра метрик Node Exporter на хосте IP: **10.10.10.101**: http://10.10.10.101:12000/metrics
2. На открывшейся странице будут показаны метрики мониторинга с указанного хоста в формате Prometheus

Метрики сервера Node в формате Prometheus

Веб-интерфейс Prometheus

1. В адресной строке браузера введите IP-адрес хоста, где установлен компонент Prometheus Server сервера Monitoring. Укажите номер порта, который определяется параметром **Node_address** в списке **Prometheus settings** в разделе сервиса Monitoring (**Services**). Пример адреса: http://10.10.10.101:9090. Для входа используйте **admin** в качестве имени пользователя и пароль, который задан в параметре **Grafana settings** — при первом **Prometheus** сервер по **Grafana**.
2. В поле **Expression** можно ввести название метрики и нажать **Execute** — в интерфейсе будут показаны значения этой метрики на всех хостах кластера АОС.

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** в разделе **Grafana settings** в настройке сервиса Monitoring (**Services**). Пример адреса: http://10.10.10.101:3000. Для входа используйте **admin** в качестве имени пользователя и пароль **Grafana administrator's password** в разделе порта в разделе настроек **Grafana settings** в разделе сервиса.
2. В открывшемся окне выберите пункт меню **Home** — **Dashboards** и развернуте панель **АОС Dashboard v.6(0_cluster_name)**. В этой панели можно выбрать одно из доступных дашбордов для просмотра графиков метрик или системных метрик, поступающих из машин кластера АОС.

Переход к интерфейсу сервиса АОС в Grafana

Подключение к Airflow через CLI

Елена Костюченко

Содержание

Назад к содержанию

Подключение

Примеры использования

Подключение

Airflow предоставляет интерфейс командной строки, реализованный в виде скрипта *bin/airflow*, который расположен в директории */opt/airflow/bin*.

Хотя рекомендуется управлять Airflow через ADCM или веб-интерфейс Airflow, CLI предоставляет доступ к большему количеству возможностей при работе с Airflow.

Чтобы взаимодействовать с Airflow через CLI, подключитесь к хосту кластера с компонентом Airflow через SSH и запустите скрипт, выполнив:

```
$ sudo /opt/airflow/bin/airflow
```

Пример вывода:

```
Usage: airflow [-h] GROUP_OR_COMMAND ...

Positional Arguments:
  GROUP_OR_COMMAND

    Groups:
      celery         Celery components
      config         View configuration
      connections    Manage connections
      dags           Manage DAGs
      db             Database operations
      jobs           Manage jobs
      kubernetes     Tools to help run the KubernetesExecutor
      pools          Manage pools
      providers      Display providers
      roles          Manage roles
      tasks          Manage tasks
      users          Manage users
      variables      Manage variables

    Commands:
      cheat-sheet    Display cheat sheet
      dag-processor  Start a standalone Dag Processor instance
      info           Show information about current Airflow and environment
      kerberos       Start a kerberos ticket renewer
      plugins        Dump information about loaded plugins
      rotate-fernet-key
                     Rotate encrypted connection credentials and variables
      scheduler      Start a scheduler instance
      standalone     Run an all-in-one copy of Airflow
      sync-perm      Update permissions for existing roles and optionally DAGs
      triggerer      Start a triggerer instance
      version        Show the version
      webserver      Start a Airflow webserver instance

Options:
  -h, --help         show this help message and exit
```

Подробное описание каждой команды доступно на странице Airflow CLI.

Чтобы протестировать работу CLI, запросите версию Airflow:

```
$ sudo /opt/airflow/bin/airflow version
```

Вывод:

```
2.6.3
```

Чтобы увидеть справку по определенной команде, используйте опцию `--help`:

```
$ sudo /opt/airflow/bin/airflow dags --help
```

Команды имеют следующий синтаксис:

```
$ airflow [GROUP_OR_COMMAND] [SUB_COMMAND] [OPTIONS]
```

Здесь:

- `GROUP_OR_COMMAND` — команда Airflow CLI или группа команд (например `dags`, `jobs` и т. д.).
- `SUB_COMMAND` — команда, относящаяся к группе команд, например, `dags list`.
- `OPTIONS` — параметры выбранной команды.

Примеры использования

Airflow предоставляет несколько инструментов, одним из которых является команда `dags`, которая позволяет управлять DAG с помощью CLI.

Например, чтобы получить список существующих DAG, выполните команду:

```
$ sudo /opt/airflow/bin/airflow dags list
```

Пример вывода:

```
dag_id                                                     | filepath
| owner   | paused
====================================================+=================================================================================
=================+=======
dataset_consumes_1                                         | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_consumes_1_and_2                                   | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_consumes_1_never_scheduled                         | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_consumes_unknown_never_scheduled                   | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_produces_1                                         | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_produces_2                                         | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
example_bash_operator                                      | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_bash_operat | airflow | None
                                                           | or.py
|         |
example_branch_datetime_operator                           | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_branch_date | airflow | None
                                                           | time_operator.py
|         |
example_branch_datetime_operator_2                         | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_branch_date | airflow | None
                                                           | time_operator.py
|         |
example_branch_datetime_operator_3                         | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_branch_date | airflow | None
                                                           | time_operator.py
|         |
example_kubernetes_executor                                | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_kubernetes_ | airflow | None
                                                           | executor.py
|         |
example_local_kubernetes_executor                          | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_local_kuber | airflow | None
                                                           | netes_executor.py
|         |
example_nested_branch_dag                                  | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_nested_bran | airflow | None
                                                           | ch_dag.py
|         |
example_params_trigger_ui                                  | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_params_trig | airflow | None
                                                           | ger_ui.py
|         |
tutorial                                                   | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/tutorial.py         | airflow | None
tutorial_dag                                               | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/tutorial_dag.py     | airflow | None
tutorial_taskflow_api                                      | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/tutorial_taskflow_a | airflow | None
                                                           | pi.py
|         |
```

Чтобы выгрузить список DAG в файл, используйте опцию `--output` и укажите один из следующих форматов в качестве аргумента: `json`, `yaml`, `plain`, `table`.

Чтобы запустить DAG, используйте команду `trigger`:

```
$ sudo /opt/airflow/bin/airflow dags trigger tutorial
```

Здесь `tutorial` — это ID того DAG, который нужно запустить.

Подключение к Airflow через REST API

Опубликовано: 2025-01-27

Содержание

Назад к содержанию

Обзор

Примеры запросов

- Получение списка пулов
- Проверка ошибок импорта
- Получение информации о DAG
- Запуск DAG
- Получение списка задач

Обзор

Кроме web-интерфейса, Airflow также предоставляет доступ к REST API. В этой статье содержатся несколько примеров того, как работать с Airflow, используя `curl`. Полный список эндпоинтов доступен в спецификации REST API Airflow.

URL для запросов к API Airflow имеет следующий формат:

```
http://<server-address>:<port>/api/v1/<endpoint>
```

Здесь:

- `<server-address>` — IP-адрес или имя хоста сервера Airflow;
- `<port>` — порт сервера Airflow;
- `<endpoint>` — эндпоинт желаемого метода.

Большинство эндпоинтов принимают JSON в качестве входных данных и возвращают JSON в ответе. Добавьте следующий заголовок при выполнении запросов:

```
Content-Type: application/json
```

Пример запроса:

```
$ curl -L 'http://localhost:8080/api/v1/dags' -H 'Content-Type: application/json' --user 'admin:admin'
```

Эта команда запрашивает информацию о существующих DAG в формате JSON.

Пример вывода:

```
{
    "dag_id": "example_params_ui_tutorial",
    "default_view": "grid",
    "description": "DAG demonstrating various options for a trigger form generated by DAG params",
    "file_token": "eJwfx1EOgCAMDRcJEZ4KEKAwCQ0mCkYSBgMzFUBTLtUmYIpVmFIJ7B_sYUAfvYlEowwF3WwQ+Fq3Mq3Y1Sb5hPcObOcjlA-N4qqqXdRCfCMCJ_PqqmGJxuGqA",
    "fileloc": "/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_params_ui_tutorial.py",
    "has_task_concurrency_limits": false,
    "is_active": true,
    "is_paused": true,
    "is_subdag": false,
    "last_expired": null,
    "last_parsed_time": "2024-08-27T13:43:21.971051+00:00",
    "last_pickled": null,
    "max_active_runs": 16,
    "max_active_tasks": 16,
    "next_dagrun": null,
    "next_dagrun_create_after": null,
    "next_dagrun_data_interval_end": null,
    "next_dagrun_data_interval_start": null,
    "owners": [
      "airflow"
    ],
    "pickle_id": null,
    "root_dag_id": null,
    "scheduler_lock": null,
    "tags": [
      {
        "name": "example_ui"
      }
    ],
    "timetable_description": "Never, external triggers only"
  },
```

Примеры запросов

Ниже приведены примеры запросов к REST API Airflow. В запросах к URL, где присутствует параметр `<dag-id>`, требуется предоставить идентификатор DAG. Большинство идентификаторов начинается DAG, имеется этого параметра.

Для авторизации используются данные по умолчанию: пользователь `admin` с паролем `admin`.

Получение списка пулов

Чтобы получить список существующих пулов Airflow, выполняйте запрос `GET` к эндпоинту `pools`.

Пример запроса:

```
$ curl -X GET 'http://localhost:8080/api/v1/pools' \
  -H 'Content-Type: application/json' --user "admin:admin"
```

Ответ сервера

```
{
  "pools": [
    {
      "description": "Default pool",
      "name": "default_pool",
      "occupied_slots": 0,
      "open_slots": 128,
      "queued_slots": 0,
      "running_slots": 0,
      "scheduled_slots": 7,
      "slots": 128
    }
  ],
  "total_entries": 1
}
```

Проверка ошибок импорта

Чтобы проверить наличие проблем с импортом DAG, отправьте запрос на эндпоинт `importErrors`.

Пример запроса:

```
$ curl --verbose 'http://localhost:8080/api/v1/importErrors' -H 'Content-Type: application/json' --user 'admin:admin'
```

Ответ сервера

```
{
  "import_errors": [
    {
      "filename": "/opt/airflow/dags/second_example.py",
      "import_error_id": 7,
      "stack_trace": "Traceback (most recent call last):\n  File \"<frozen importlib._bootstrap_external>\", line 947, in source_to_code\n  File \"<frozen importlib._bootstrap>\", line 241, in _call_with_frames_removed\n  File \"/opt/airflow/dags/second_example.py", line 14\    apart pandas as pd\n                                                       ^^^^^^^^\nSyntaxError: invalid syntax\n",
      "timestamp": "2024-09-03T15:24:12.860998+00:00"
    }
  ],
  "total_entries": 1
}
```

Получение информации о DAG

Чтобы получить информацию о конкретном DAG, отправьте запрос `GET` с идентификатором нужного DAG на эндпоинт `/details`.

Пример запроса:

```
$ curl -v -X GET http://localhost:8080/api/v1/dags/<dag-id>/details \
  -H 'Content-Type: application/json' \
  --user "admin:admin"
```

Ответ сервера

```
{
  "catchup": false,
  "concurrency": 16,
  "dag_id": "tutorial_dag",
  "dag_run_timeout": null,
  "default_view": "grid",
  "description": "DAG tutorial",
  "doc_md": "\nThe DAG tutorial documentation\nThis DAG is demonstrating an Extract -> Transform -> Load pipeline\n",
  "end_date": null,
  "file_token": "eJwfx1EQDAFRAN8_3CiVrJKUTFHwUCz_lXuH_uhqUI3WRAHmGaGxlnXQkp-mr7rOZsUEYq_BsQFCHqqqj-vHVE-RBvmZEv_RKQ_eyoo1zRwJGDEFQJwYlerGrPCBM",
  "fileloc": "/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/tutorial_dag.py",
  "is_active": true,
  "is_paused": false,
  "is_paused_upon_creation": null,
  "is_subdag": false,
  "last_parsed": "2024-09-02T12:20:29 979933+00:00",
  "max_active_runs": 16,
  "max_active_tasks": 16,
  "orientation": "LR",
  "owners": [
    "airflow"
  ],
  "owner": {},
  "pickle_id": null,
  "render_template_as_native_obj": false,
  "schedule_interval": null,
  "start_date": "2021-01-01T00:00:00+00:00",
  "tags": [
    {
      "name": "example"
    }
  ],
  "timezone": "Timezone('UTC')"
}
```

Запуск DAG

Чтобы запустить DAG, отправьте запрос `POST`, содержащий параметры запуска DAG, на эндпоинт `dagRuns`.

Пример запроса:

```
$ curl -v -X POST http://localhost:8080/api/v1/dags/<dag-id>/dagRuns \
  -H 'Content-Type: application/json' \
  -d '{conf:{}}' \
  --user "admin:admin"
```

Здесь `{conf:{}}` — параметры запуска DAG в формате JSON.

Пример параметров:

```
{
  "dag_run_id": "string",
  "logical_date": "2019-08-24T14:15:22Z",
  "execution_date": "2019-08-24T14:15:22Z",
  "conf": {},
  "note": "string"
}
```

Чтобы установить конкретную дату выполнения `logical_date` для DAG, передайте временную метку в формате `YYYY-MM-DDTHH:mm:SS`.

Ответ сервера

```
{
  "conf": {},
  "dag_id": "tutorial_dag",
  "dag_run_id": "string",
  "data_interval_end": "2019-08-24T14:15:22+00:00",
  "data_interval_start": "2019-08-24T14:15:22+00:00",
  "end_date": null,
  "execution_date": "2019-08-24T14:15:22+00:00",
  "external_trigger": true,
  "last_scheduling_decision": null,
  "logical_date": "2019-08-24T14:15:22+00:00",
  "note": null,
  "run_type": "manual",
  "start_date": null,
  "state": "queued"
}
```

Получение списка задач

Чтобы просмотреть список задач определенного DAG, отправьте запрос `GET` на эндпоинт `tasks` этого DAG.

Пример запроса:

```
$ curl -v -X GET http://localhost:8080/api/v1/dags/<dag-id>/tasks \
  -H 'Content-Type: application/json' \
  --user "admin:admin"
```

Ответ сервера


```
{
  "tasks": [
    {
      "class_ref": {
        "class_name": "PythonOperator",
        "module_path": "airflow.operators.python"
      },
      "depends_on_past": false,
      "downstream_task_ids": [
        "transform_people"
      ],
      "end_date": null,
      "execution_timeout": null,
      "extra_links": [],
      "is_mapped": false,
      "operator_name": "PythonOperator",
      "owner": "airflow",
      "params": {},
      "pool": "default_pool",
      "pool_slots": 1.0,
      "priority_weight": 1.0,
      "queue": "default",
      "retries": 0.0,
      "retry_delay": {
        "__type": "TimeDelta",
        "days": 0,
        "microseconds": 0,
        "seconds": 300
      },
      "retry_exponential_backoff": false,
      "start_date": "2021-12-17T00:00:00+00:00",
      "task_id": "extract_people",
      "template_fields": [
        "templates_dict",
        "op_args",
        "op_kwargs"
      ],
      "trigger_rule": "all_success",
      "ui_color": "#FFefff",
      "ui_fgcolor": "#000",
      "wait_for_downstream": false,
      "weight_rule": "downstream"
    },
    {
      "class_ref": {
        "class_name": "PythonOperator",
        "module_path": "airflow.operators.python"
      },
      "depends_on_past": false,
      "downstream_task_ids": [],
      "end_date": null,
      "execution_timeout": null,
      "extra_links": [],
      "is_mapped": false,
      "operator_name": "PythonOperator",
      "owner": "airflow",
      "params": {},
      "pool": "default_pool",
      "pool_slots": 1.0,
      "priority_weight": 1.0,
      "queue": "default",
      "retries": 0.0,
      "retry_delay": {
        "__type": "TimeDelta",
        "days": 0,
        "microseconds": 0,
        "seconds": 300
      },
      "retry_exponential_backoff": false,
      "start_date": "2021-12-17T00:00:00+00:00",
      "task_id": "transform_people",
      "template_fields": [
        "templates_dict",
        "op_args",
        "op_kwargs"
      ],
      "trigger_rule": "all_success",
      "ui_color": "#FFefff",
      "ui_fgcolor": "#000",
      "wait_for_downstream": false,
      "weight_rule": "downstream"
    }
  ],
  "total_entries": 3
}
```

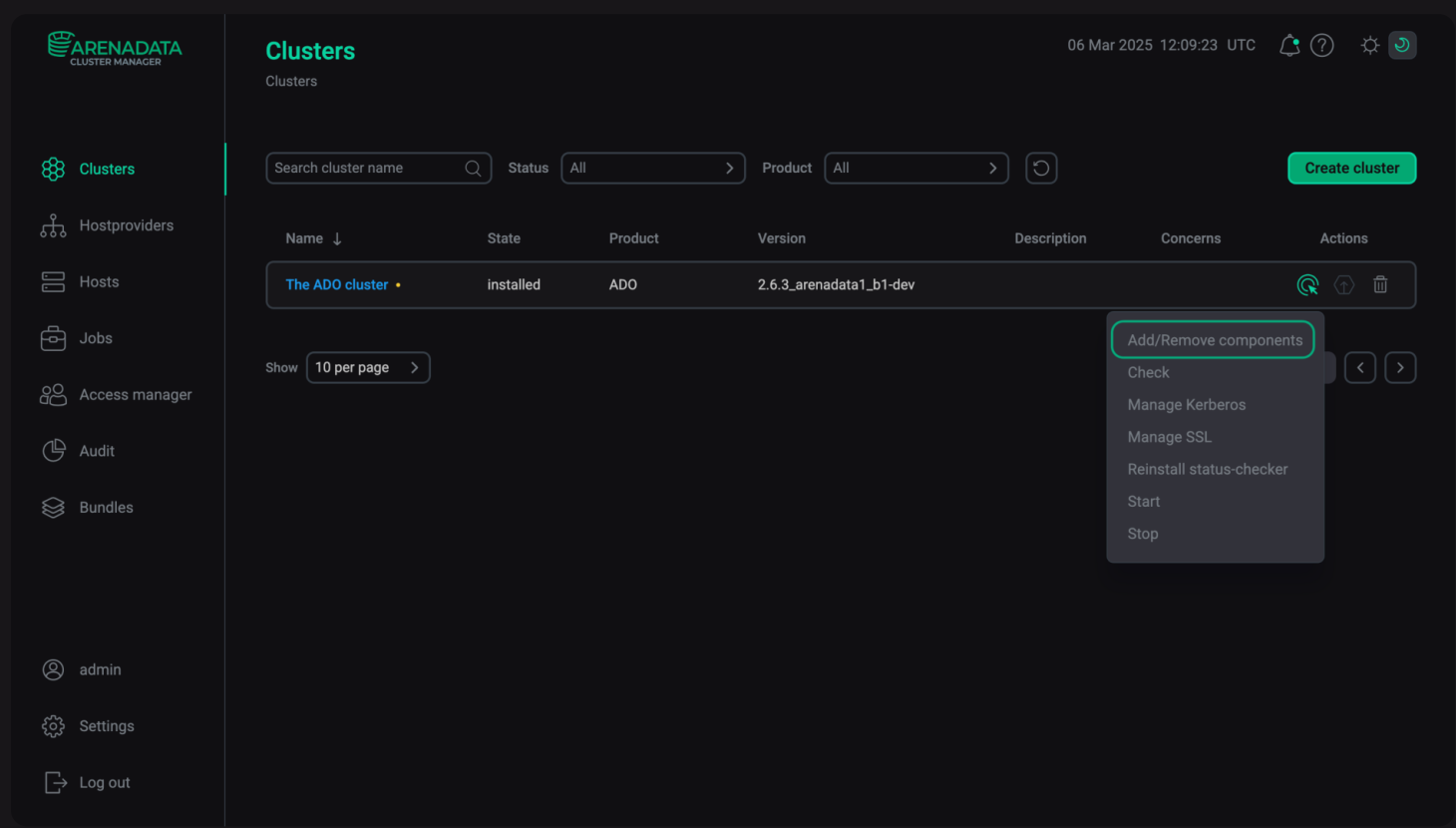
Add/Remove components

Елена Костюченко

Назад к содержанию

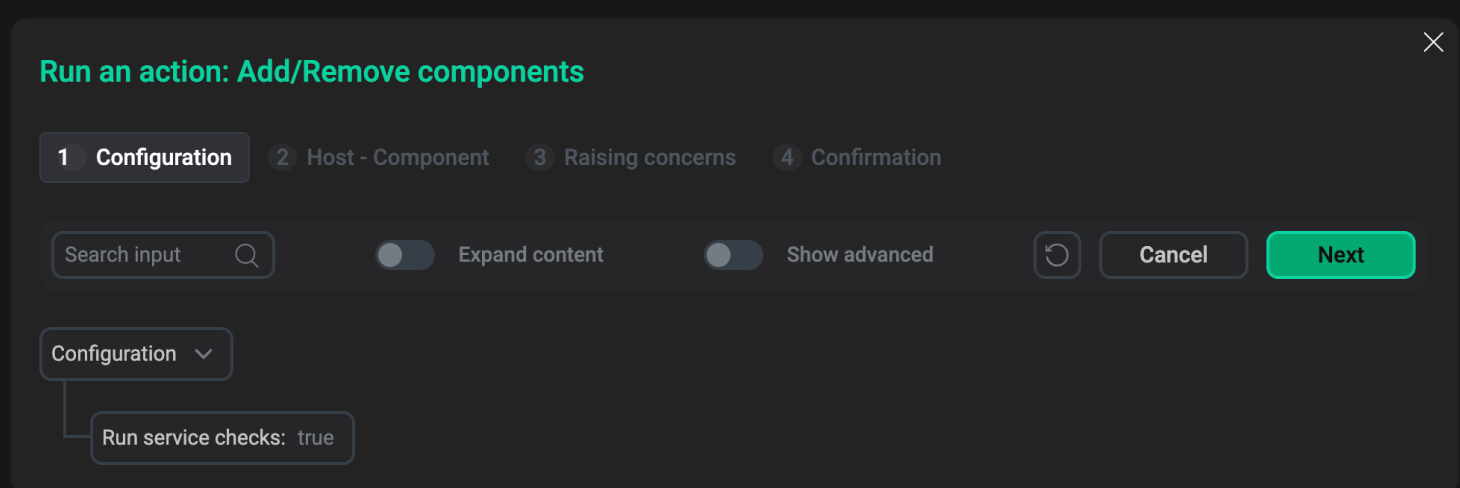
При выполнении действия ***Add/Remove components*** открывается интерфейс распределения компонентов, где вы можете добавить, удалить или перераспределить компоненты сервисов.

Чтобы выполнить действие, перейдите на страницу ***Clusters***, выберите кластер ADO, нажмите и выберите действие из появившегося списка.



Выполнение действия Add/Remove components

Диалоговое окно действия ***Add/Remove components*** содержит опцию ***Run service checks***, которая определяет, следует ли запускать действие Check для сервисов после добавления или удаления компонентов.



Окно "Run an action"

Также можно использовать действие ***Add/Remove components*** определенного сервиса, чтобы вручную добавить компоненты этого сервиса.

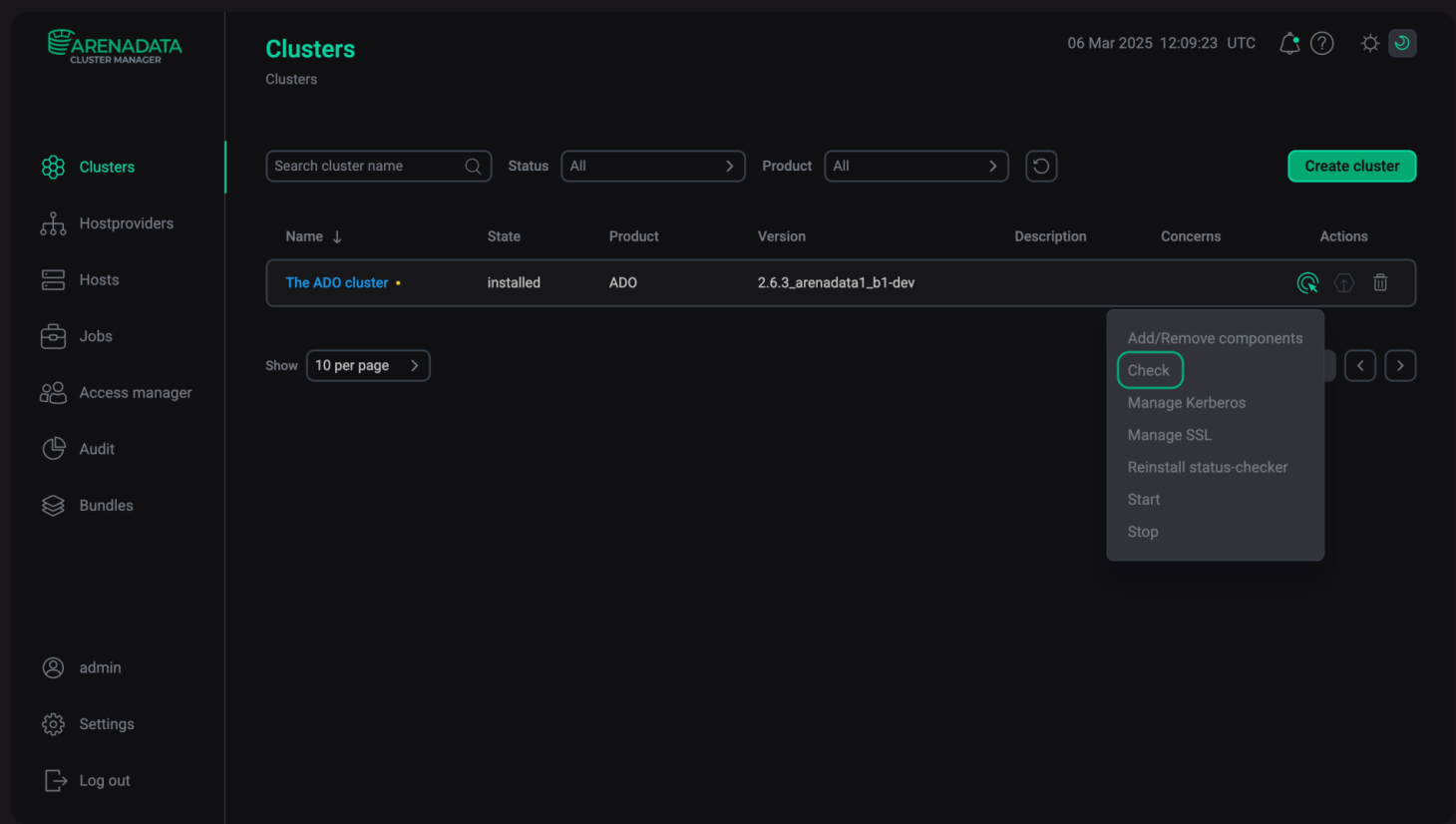
Check

Евгения Кузина, Елена Костюченко

Назад к содержанию

Действие ***Check*** проверяет правильно ли работают сервисы ADO. ***Check*** последовательно запускает тесты для каждого сервиса, чтобы проверить работоспособность сервиса и его компонентов.

Чтобы выполнить действие, перейдите на страницу ***Clusters***, выберите кластер ADO, нажмите и выберите действие из появившегося списка.



Выполнение действия "Check"

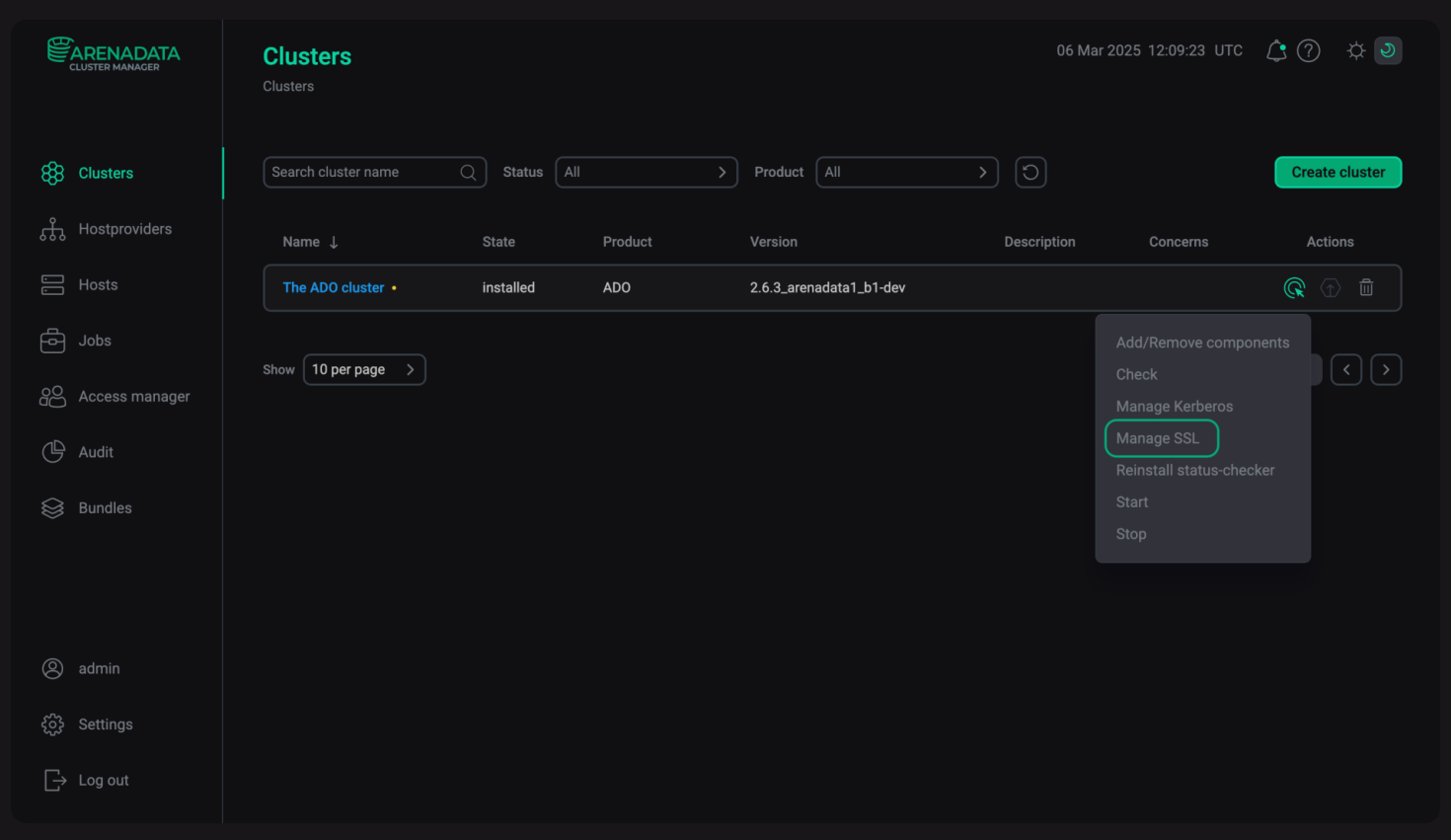
Manage SSL

Сергей Тихомиров, Евгения Кузина, Елена Костюченко

Назад к содержанию

Действие ***Manage SSL*** позволяет управлять SSL-шифрованием всех сервисов в кластере ADO. За дополнительной информацией о настройках SSL можно обратиться к статье Требования к SSL-сертификатам.

Чтобы выполнить действие, перейдите на страницу ***Clusters***, выберите кластер ADO, нажмите и выберите действие из появившегося списка.



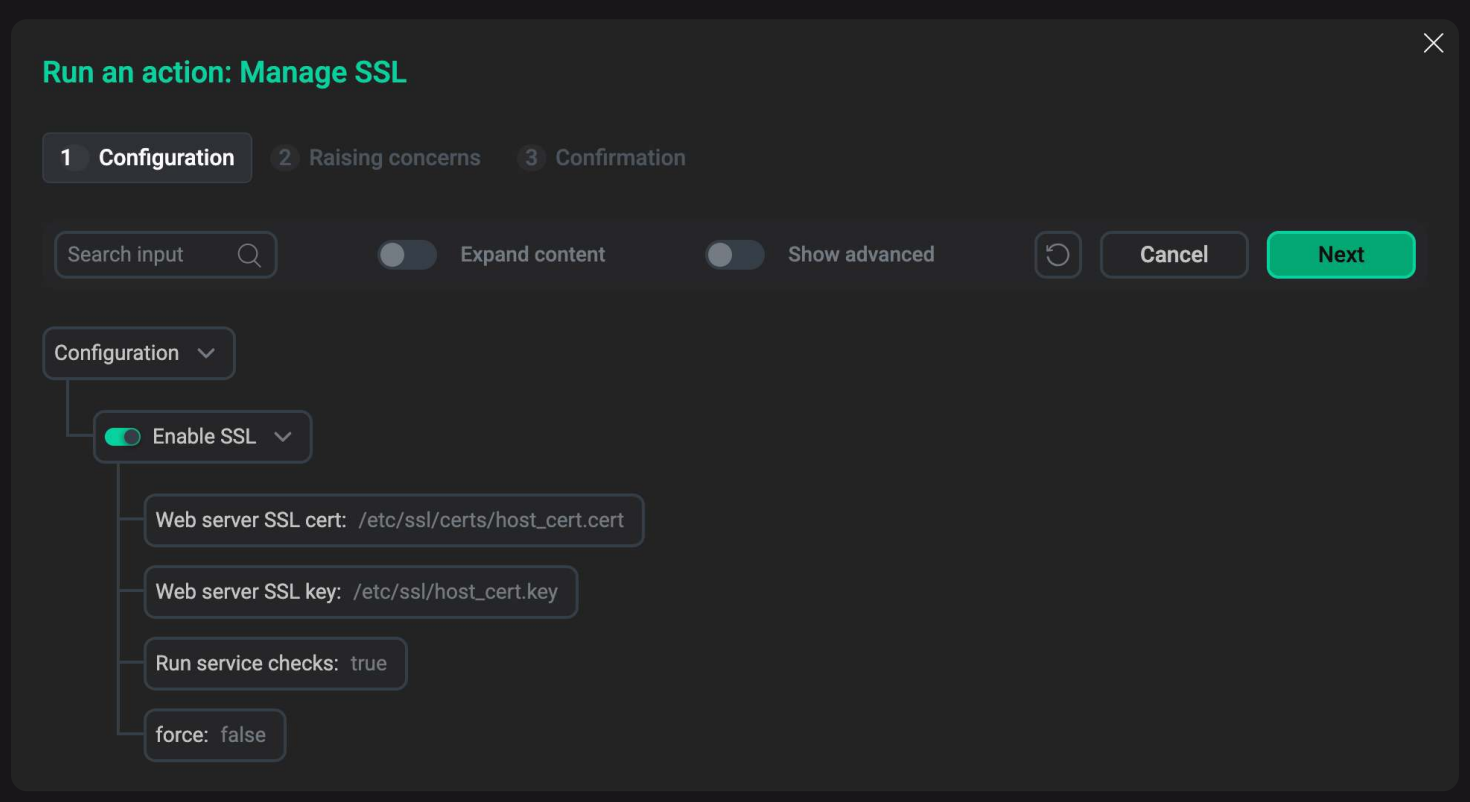
Выполнение действия "Manage SSL"

Когда вы выберете ***Manage SSL***, откроется диалоговое окно, в котором необходимо указать настройки SSL, перечисленные в таблице ниже.

Настройки SSL

Параметр	Описание	Значение по умолчанию
Web server SSL cert	Путь к SSL-сертификату веб-сервера	/etc/ssl/certs/host_cert.cert
Web server SSL key	Путь к SSL-ключу веб-сервера	/etc/ssl/host_cert.key
Run service checks	Запускает проверку работоспособности сервиса после применения настроек	True
force	Определяет, следует ли перезаписать текущие параметры SSL указанными значениями	False

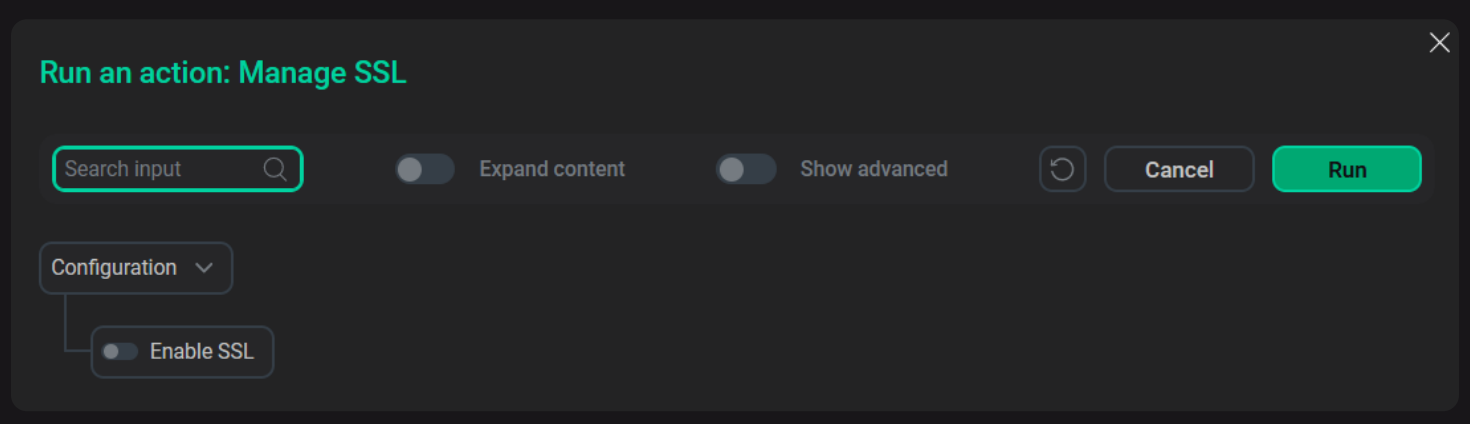
Для активации SSL запустите действие с активированной группой ***Enable SSL*** и заполненными параметрами.



Включение SSL действием "Manage SSL"

После указания параметров нажмите ***Run***. В окне с подтверждением действия, которое появится, можно установить флажок ***Verbose***, чтобы была выведена дополнительная информация о выполнении действия.

Для отключения SSL запустите действие ***Manage SSL*** с деактивированной группой ***Enable SSL***.



Отключение SSL действием "Manage SSL"

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.

Manage Kerberos

Сергей Тихомиров, Елена Костюченко

Содержание

Назад к содержанию

Действие Manage Kerberos

Опция Custom kerberization settings

Рекомендации для пользовательских keytab-файлов

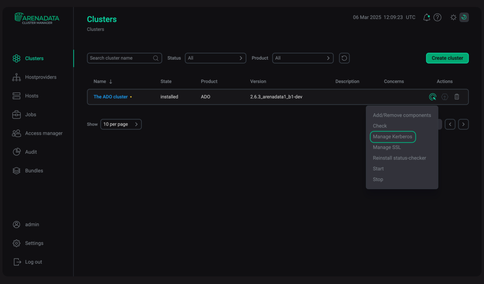
Действие Manage Kerberos

Действие ***Manage Kerberos*** позволяет включать, реконфигурировать и отключать Kerberos. Для его запуска потребуется установленный ADO-кластер (см. Установка Arenadata Orchestrator) и один или несколько Центров Распределения Ключей (Key Distribution Center, KDC). Прежде чем приступать к управлению Kerberos, рекомендуется прочитать статью Обзор Kerberos о требованиях для керберизации.

Среди прочего, ***Manage Kerberos*** создано для реализации следующих сценариев:

- Включение Kerberos без данных администратора с помощью существующего клиента, принципалов и/или keytab-файлов.
- Быстрый перевод кластера из состояния kerberized в состояние non-kerberized в случае возникновения ошибки в начале процесса керберизации (например, неверные данные kadmin) без необходимости реконфигурации сервисов, удаления принципалов и keytab-файлов.
- Отложенная реконфигурация сервисов — возможность сконфигурировать Kerberos в моменте, а реконфигурировать и перезапустить сервисы позже.

Для запуска действия в веб-интерфейсе ADCM перейдите на страницу ***Clusters***. Выберите установленный и подготовленный кластер ADO и запустите действие ***Manage Kerberos***.

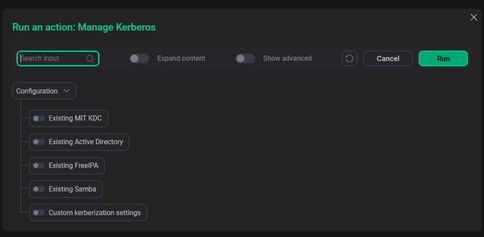


Manage Kerberos

В появившемся окне будет предложено несколько опций для запуска действия:

- С помощью существующего MIT KDC.
- С помощью существующей MS Active Directory.
- С помощью существующей FreeIPA.
- С помощью существующей Samba.
- С помощью опции Custom kerberization settings.

Каждую из этих опций можно совместить с опцией ***Custom kerberization settings***.



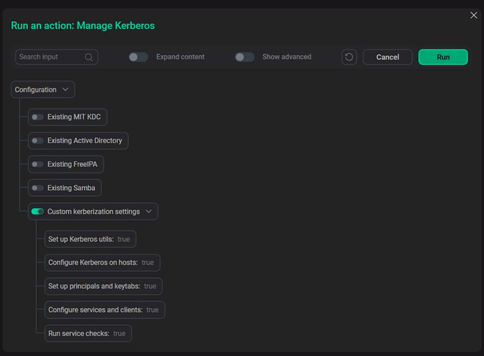
Способы управления Kerberos

ВАЖНО

Запуск действия ***Manage Kerberos*** с выбранным KDC автоматически активирует Kerberos.

Опция Custom kerberization settings

Опция ***Custom kerberization settings*** позволяет пользователю выбирать шаги керберизации, например, создание принципалов и keytab-файлов.



Параметры Custom kerberization settings

Описание параметров Custom kerberization settings

Параметр	Описание	Значение по умолчанию
Set up Kerberos utils	Активирует возможность устанавливать и удалять клиенты и утилиты Kerberos. Данный параметр учитывается в действиях ***Expand*** и ***Install***	True
Configure Kerberos on hosts	Активирует возможность конфигурировать кластер, включая *krb5.conf*, *ldap.conf*	True
Set up principals and keytabs	Активирует возможность создавать, пересоздавать и удалять принципалов и keytab-файлы. Пароли для принципалов генерируются случайным образом перед созданием keytab-файлов. Данный параметр учитывается в действиях ***Expand*** и ***Install***. Бандл ADCM установит owner и разрешения для keytab-файлов только если этот флаг установлен. В случае отсутствия прав администратора пользователь должен предоставить подготовленный keytab-файл с правильно установленными owner и разрешениями	True
Configure services and clients	Активирует возможность обновлять конфигурации сервисов и клиентов	True
Run service checks	Активирует возможность проводить проверку сервисов	True

Рекомендации для пользовательских keytab-файлов

Ниже представлены рекомендации для значений owner, group и прав для keytab-файлов.

Рекомендации для keytab-файлов

Short name компонента	Owner keytab-файла	Group keytab-файла	Права
airflow	airflow	airflow	600

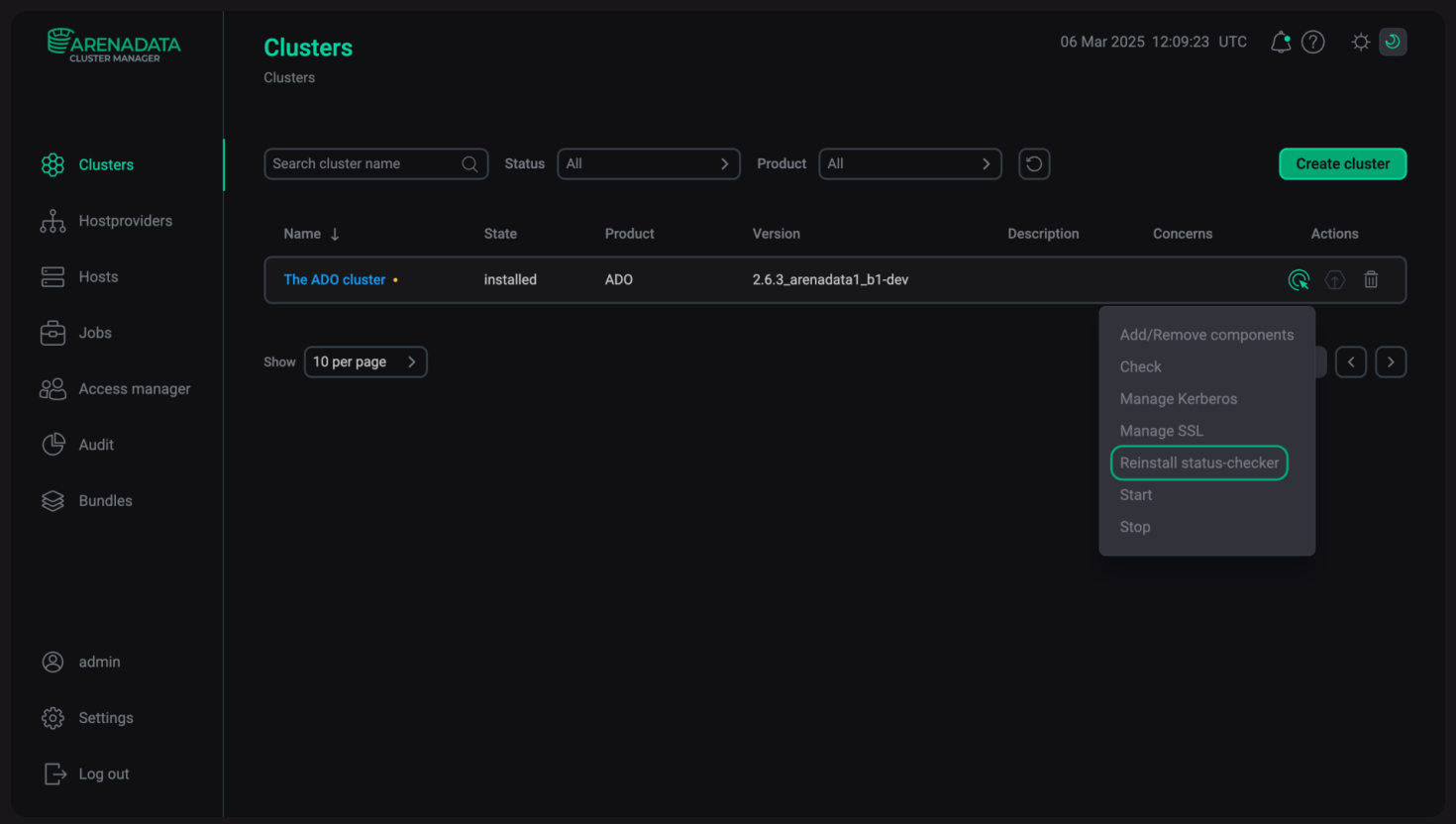
Reinstall status-checker

Евгения Кузина, Елена Костюченко

Назад к содержанию

Действие ***Reinstall status-checker*** последовательно переустанавливает status-checker для всех сервисов.

Чтобы выполнить действие, перейдите на страницу ***Clusters***, выберите кластер ADO, нажмите и выберите действие из появившегося списка.



Выполнение действия "Reinstall status-checker"

Start

Евгения Кузина, Елена Костюченко

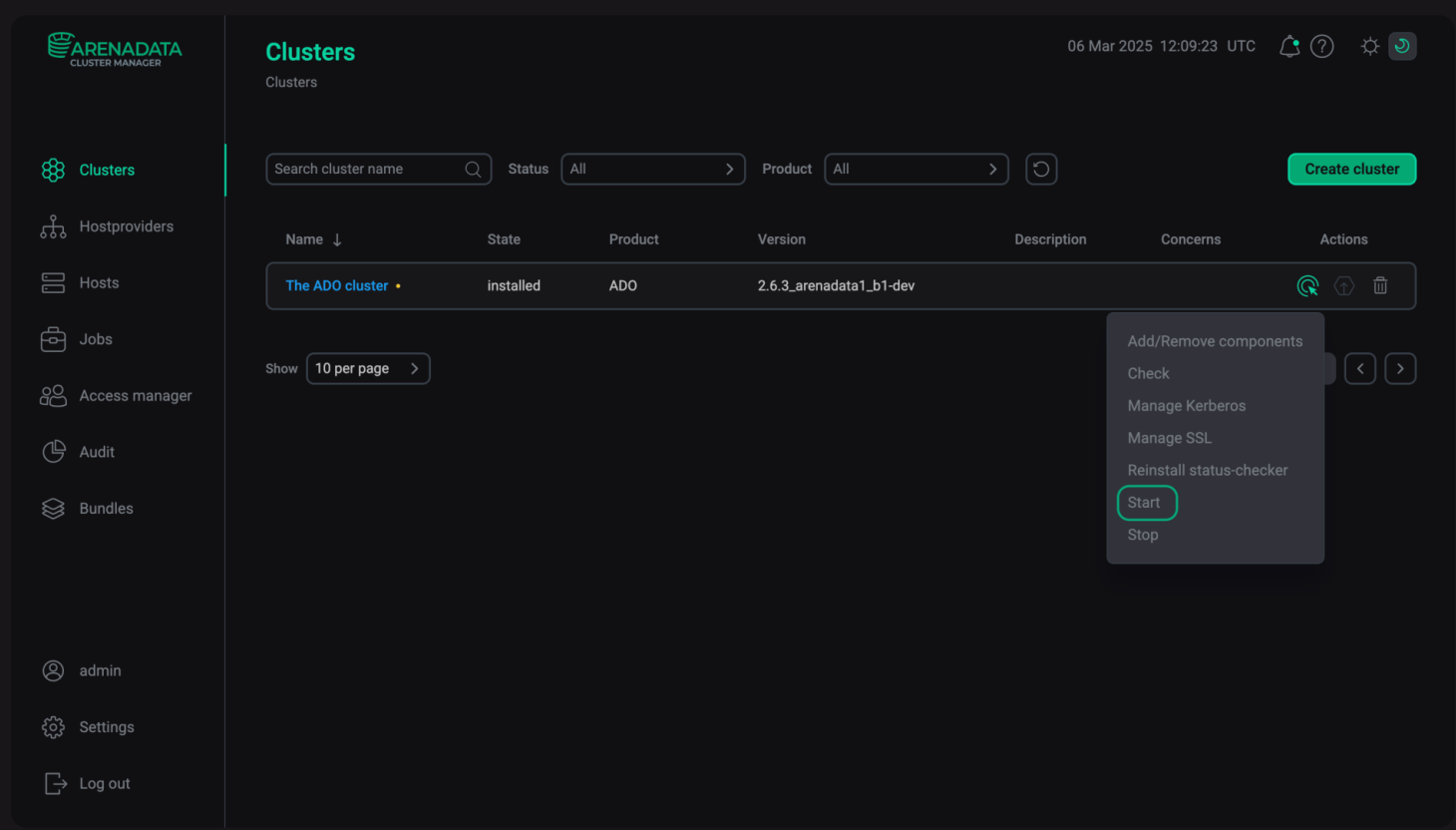
Назад к содержанию

ВАЖНО

Если вы установили кластер мониторинга и хотите использовать его с кластером ADO, сначала запустите кластер мониторинга. Для этого выполните действие ***Start*** кластера мониторинга так же, как описано для кластера ADO.

Действие ***Start*** запускает все сервисы, установленные в кластере ADO, один за другим в правильном порядке.

Чтобы выполнить действие, перейдите на страницу ***Clusters***, выберите кластер ADO, нажмите и выберите действие из появившегося списка.



Выполнение действия "Start"

Также можно использовать действие ***Start*** определенного сервиса, чтобы вручную запустить только необходимые сервисы.

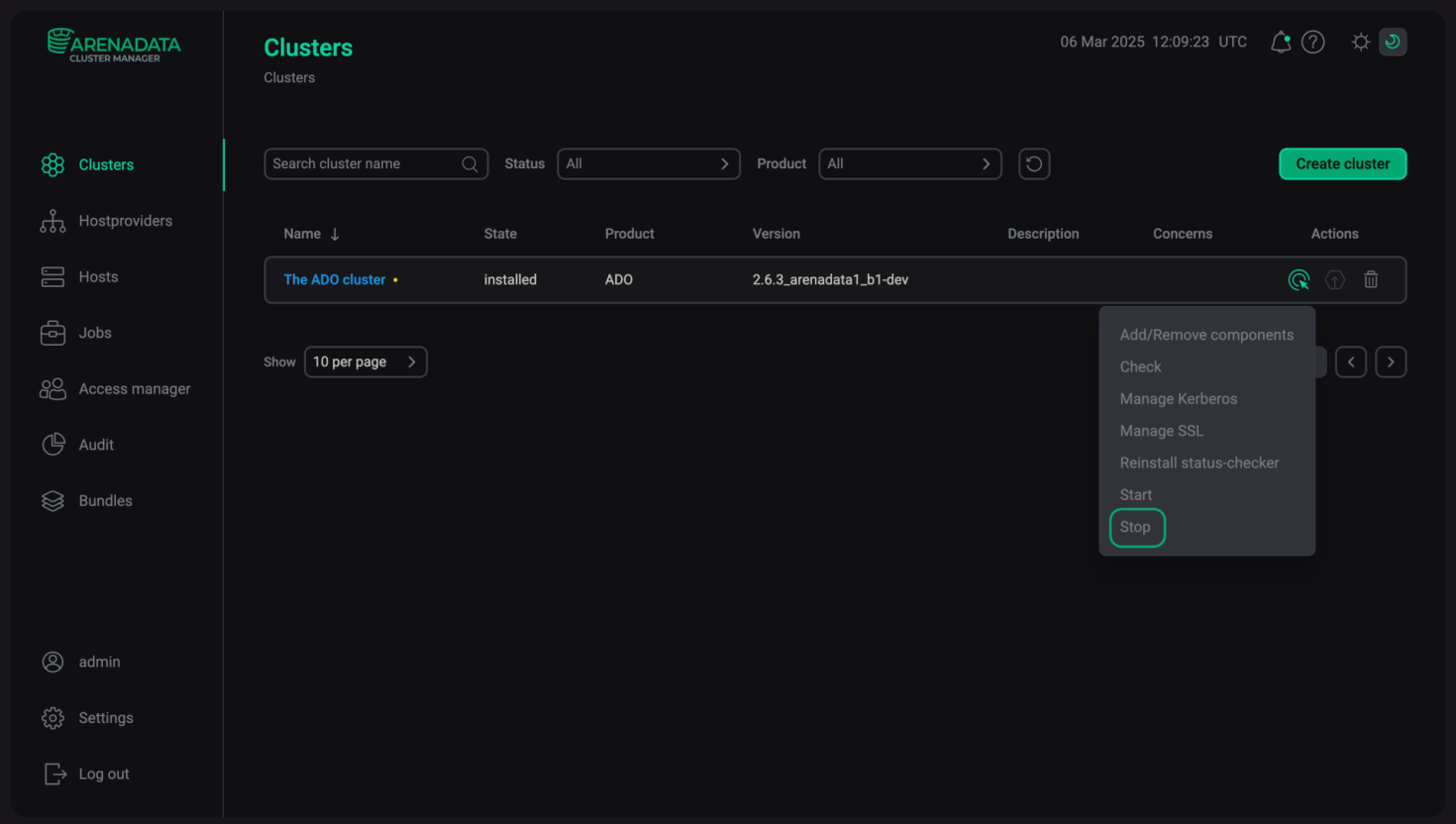
Stop

Евгения Кузина, Елена Костюченко

Назад к содержанию

Действие ***Stop*** останавливает все сервисы, установленные в кластере ADO, в правильном порядке.

Чтобы выполнить действие, перейдите на страницу ***Clusters***, выберите кластер ADO, нажмите и выберите действие из появившегося списка.



Выполнение действия "Stop"

Также можно использовать действие ***Stop*** определенного сервиса, чтобы вручную остановить необходимые сервисы или весь кластер.

Управление сервисом ADPG через ADCM

Елена Костюченко

Назад к содержанию

Пользовательский интерфейс ADCM позволяет выполнять действия, управляющие сервисом Arenadata Postgres (ADPG). Действия, доступные для сервиса ADPG, перечислены в таблице ниже.

Действие	Описание
Add/Remove components	При выполнении этого действия открывается интерфейс распределения компонентов, где вы можете добавить, удалить или перераспределить компоненты ADPG
Check ADPG	Выполняет проверку работоспособности сервиса и его отдельных компонентов
Restart ADPG	Перезапускает сервис. Для этого действия доступна опция ***Apply configs from ADCM***. Если выбрано значение `true`, то рестарт сервиса выполняется с учетом всех параметров, указанных в настройках этого сервиса в ADCM. В противном случае настройки сервиса в ADCM игнорируются
Remove	Удаляет сервис из кластера. Данное действие предназначено для удаления уже установленных сервисов. В то время как кнопка 🗑 используется для удаления сервисов, чьи компоненты еще не были распределены по узлам кластера
Start ADPG	Запускает сервис. Для этого действия доступна опция ***Apply configs from ADCM***. Если выбрано значение `true`, то запуск сервиса выполняется с учетом всех параметров, указанных в настройках этого сервиса в ADCM. В противном случае настройки сервиса в ADCM игнорируются
Stop ADPG	Останавливает сервис

Управление сервисом Airflow через ADCM

Елена Костюченко

Назад к содержанию

Пользовательский интерфейс ADCM позволяет выполнять действия, управляющие сервисом Airflow. Действия, доступные для сервиса Airflow, перечислены в таблице ниже.

Подробное описание компонентов Airflow приведено в статье Airflow.

Действие	Описание
Add/Remove components	При выполнении этого действия открывается интерфейс распределения компонентов, где вы можете добавить, удалить или перераспределить компоненты Airflow
Check	Выполняет проверку работоспособности сервиса и его отдельных компонентов
Start	Запускает сервис. Для этого действия доступна опция ***Apply configs from ADCM***. Если выбрано значение `true`, то запуск сервиса выполняется с учетом всех параметров, указанных в настройках этого сервиса в ADCM. В противном случае настройки сервиса в ADCM игнорируются
Stop	Останавливает сервис
Restart	Перезапускает сервис. Для этого действия доступна опция ***Apply configs from ADCM***. Если выбрано значение `true`, то рестарт сервиса выполняется с учетом всех параметров, указанных в настройках этого сервиса в ADCM. В противном случае настройки сервиса в ADCM игнорируются
Remove	Удаляет сервис из кластера. Данное действие предназначено для удаления уже установленных сервисов. В то время как кнопка 🗑 используется для удаления сервисов, чьи компоненты еще не были распределены по узлам кластера
Rotate fernet key	Выполняет ротацию ключа шифрования Fernet. Если опция ***Autogenerate new fernet key*** имеет значение `True`, ADCM сгенерирует новый ключ Fernet автоматически. Результат зависит от параметра ***Manage sensitive configuration data***, см. Ротация fernet-ключа. Это действие потребует несколько перезапусков сервиса Airflow
Sync permission	Обновляет разрешения для существующих ролей
Sync requirements	Обновляет зависимости Airflow на всех хостах. Для применения изменений требуется перезапуск (***Restart***)

Некоторые из вышеперечисленных действий можно активировать отдельно для каждого компонента Airflow. Доступные действия для компонентов перечислены в таблице ниже.

Компонент	Start	Stop	Restart	Check
Airflow DAG Processor	+	+	+	—
Airflow Flower	+	+	+	+
Airflow Scheduler	+	+	+	—
Airflow Server	+	+	+	+
Airflow Triggerer	+	+	+	—
Airflow Worker	+	+	+	—

Управление сервисом Monitoring через ADCM

Елена Костюченко

Назад к содержанию

Пользовательский интерфейс ADCM позволяет выполнять действия, управляющие сервисом Monitoring и его компонентами. Действия, доступные для сервиса Monitoring, перечислены в таблице ниже.

Действие	Описание
Add/Remove components	При выполнении этого действия открывается интерфейс распределения компонентов, где вы можете добавить, удалить или перераспределить компоненты Monitoring
Check	Выполняет проверку работоспособности сервиса и его отдельных компонентов
Reconfigure and restart	При запуске действия все конфигурации сервиса, определенные в настройках ADCM, будут применены после перезагрузки
Reinstall	Переустанавливает сервис с конфигурацией по умолчанию
Start	Запускает сервис
Stop	Останавливает сервис

Для каждого конкретного компонента Statsd-exporter можно выполнить следующие действия: ***Restart Statsd-exporter***, ***Start Statsd-exporter*** и ***Stop Statsd-exporter***.

Управление сервисом Redis через ADCM

Елена Костюченко

Назад к содержанию

Пользовательский интерфейс ADCM позволяет выполнять действия, управляющие сервисом Redis. Действия, доступные для сервиса Redis, перечислены в таблице ниже.

Действие	Описание
Add/Remove components	При выполнении этого действия открывается интерфейс распределения компонентов, где вы можете добавить, удалить или перераспределить компоненты Redis
Check Redis	Выполняет проверку работоспособности сервиса и его отдельных компонентов
Restart Redis	Перезапускает сервис. Для этого действия доступна опция ***Apply configs from ADCM***. Если выбрано значение `true`, то рестарт сервиса выполняется с учетом всех параметров, указанных в настройках этого сервиса в ADCM. В противном случае настройки сервиса в ADCM игнорируются
Remove	Удаляет сервис из кластера. Данное действие предназначено для удаления уже установленных сервисов. В то время как кнопка 🗑 используется для удаления сервисов, чьи компоненты еще не были распределены по узлам кластера
Start Redis	Запускает сервис. Для этого действия доступна опция ***Apply configs from ADCM***. Если выбрано значение `true`, то запуск сервиса выполняется с учетом всех параметров, указанных в настройках этого сервиса в ADCM. В противном случае настройки сервиса в ADCM игнорируются
Stop Redis	Останавливает сервис

Для каждого отдельного сервера Redis вы можете выполнить следующие действия: ***Check***, ***Restart***, ***Start*** и ***Stop***.

Интеграция с HashiCorp Vault для хранения секретов

Сергей Тихомиров

Содержание

Назад к содержанию

Обзор
Подготовка
Настройка в ADCM
Ротация fernet-ключа

Обзор

В Airflow под secret backend понимается механизм безопасного хранения и получения конфиденциальных данных, таких как пароли или API-ключи. Airflow поддерживает работу с различными видами secret backend (например, системные переменные, БД метаданных, локальная файловая система), но для безопасности рекомендуется использовать HashiCorp Vault.

Подготовка

Перед тем как проводить настройку в ADCM, убедитесь, что у вас установлен, настроен и доступен экземпляр Vault (см. Установка Vault).

Для подготовки к интеграции с ADO выполните следующие действия:

1. Создайте точку монтирования для Airflow:

```
$ vault secret enable -path=airflow -version=2 kv
```

2. Сгенерируйте данные для входа (например, токен):

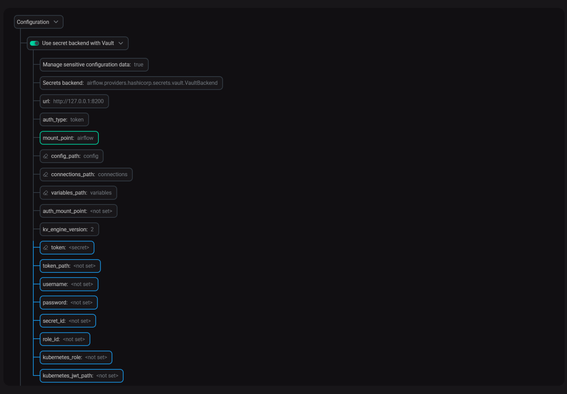
```
$ vault token create -period=30m
```

Сохраните полученный токен в файл или скопируйте его, т.к. он вам понадобится далее.

Настройка в ADCM

Для интеграции ADO с HashiCorp Vault следуйте шагам ниже:

1. В ADCM выберите ваш кластер ADO и перейдите в раздел ***Services → Airflow***.
2. Активируйте переключатель ***Show advanced***.
3. Активируйте переключатель ***Use secret backend with Vault*** и присвойте параметрам необходимые значения (см. Конфигурационные параметры Airflow):



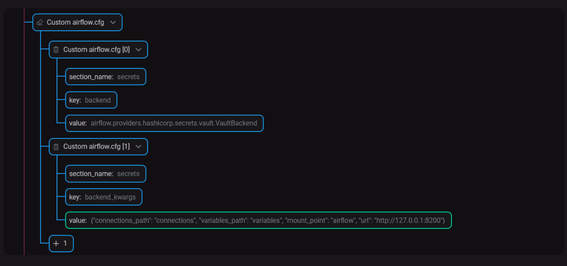
Настройки Vault в ADCM

ПРИМЕЧАНИЕ

При активированном параметре ***Manage sensitive configuration data*** ADO берет на себя как создание секретов (перемещение их из настроек в Vault), так и их обновление. Для подобных действий требуются права на создание секретов в Vault.

4. Сохраните конфигурацию и перезапустите Airflow.

Чтобы сгенерировать блок ***airflow.cfg[secrets]*** вручную, вам необходимо добавить необходимые параметры через поля параметра ***Custom airflow.cfg***.



Ручная настройка секретов в ADCM

Значения кастомных параметров переписывают значения, установленные в секции ***Use secret backend with Vault***.

Ротация fernet-ключа

При включенной интеграции ADO с HashiCorp Vault меняется поведение сервисного действия ***Rotate fernet key***.

	Autogenerate new fernet key активен	*Autogenerate new fernet key* неактивен
Manage sensitive configuration data активен	Fernet-ключ изменится в Vault, значение не будет сохранено на хосте	Если в качестве кастомного fernet-ключа предоставить пустое значение, то действие выполнится, но ни в Vault, ни на хосте никаких изменений не будет
Manage sensitive configuration data неактивен	Fernet-ключ изменится на хосте, в Vault не произойдет никаких изменений	Если в качестве кастомного fernet-ключа предоставить пустое значение, то действие завершится с ошибкой из-за невозможности ротации ключа с пустым значением

Во всех остальных случаях (например, при непустом валидном/невалидном кастомном fernet-ключе), действие ведет себя одинаково вне зависимости от упомянутых выше параметров.

Логирование в Airflow

Альберт Багдасарян, Елена Костюченко

Содержание

Назад к содержанию

Поиск по логам

Уровни логирования

Конфигурация логирования

Airflow ведет текстовые логи для анализа ошибок, которые могут возникнуть в процессе работы DAG. Эти логи расположены в директории */var/log/airflow/* хоста сервера Airflow, но они также доступны в пользовательском интерфейсе Airflow:

Пути к логам

Полный путь к лог-файлам выглядит следующим образом:

/var/log/airflow/dag_id=<DAG_ID>/run_id=<DAG_Run_ID>/task_id=<Task_ID>/<log_number>.log

Здесь:

- `<DAG_ID>` — идентификатор DAG.
- `<DAG_Run_ID>` — идентификатор запуска DAG, который объединяет в себе тип запуска и временную метку. Например, *run_id=scheduled__2024-07-14T14:18:33.254657+00:00*.
- `<Task_ID>` — идентификатор задачи.
- `<log_number>` — номер лог-файла (отсчет начинается с 1).

Полный путь к логам менеджера процессов (process manager) — */var/log/airflow/dag_processor_manager/*.

Полный путь к логам планировщика (scheduler) — */var/log/airflow/scheduler/*.

Для просмотра логов Airflow на хосте:

1. Подключитесь к серверу Airflow с помощью SSH и запустите следующую команду:

```
$ ls -la /var/log/airflow/
```

Вывод в консоль приведен ниже:

```
total 12
drwxr-xr-x.  5 airflow airflow   77 Jul 15 14:18 .
drwxr-xr-x. 16 root    root    4096 Aug  5 08:44 ..
drwxr-xr-x. 13 airflow airflow 4096 Jul 30 10:12 dag_id=adcm_check
drwxr-xr-x.  2 airflow airflow  189 Jul 29 23:54 dag_processor_manager
drwxr-xr-x. 20 airflow airflow 4096 Aug  5 07:56 scheduler
```

2. Откройте содержимое нужного лог-файла:

```
$ cat /var/log/airflow/dag_id=adcm_check/run_id=manual__2024-07-15T14:18:43.743847+00:00/task_id=runme_1/attempt=1.log
```

Вывод в консоль приведен ниже:

```
[2024-07-15T14:18:46.143+0000] {taskinstance.py:1103} INFO - Dependencies all met for dep_context=requeueable deps ti=
<TaskInstance: adcm_check.runme_1 manual__2024-07-15T14:18:43.743847+00:00 [queued]>
[2024-07-15T14:18:46.144+0000] {taskinstance.py:1308} INFO - Starting attempt 1 of 4
[2024-07-15T14:18:46.157+0000] {taskinstance.py:1327} INFO - Executing <Task(BashOperator): runme_1> on 2024-07-15 14:18:43.
743847+00:00
[2024-07-15T14:18:46.164+0000] {standard_task_runner.py:57} INFO - Started process 7082 to run task
[2024-07-15T14:18:46.168+0000] {standard_task_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'adcm_check', 'runme_1',
'manual__2024-07-15T14:18:43.743847+00:00', '--job-id', '12', '--raw', '--subdir', 'DAGS_FOLDER/adcm_check.py', '--cfg-path',
'/tmp/tmpr_3qshf4']
[2024-07-15T14:18:46.171+0000] {standard_task_runner.py:85} INFO - Job 12: Subtask runme_1
[2024-07-15T14:18:46.233+0000] {task_command.py:410} INFO - Running <TaskInstance: adcm_check.runme_1 manual__2024-07-15T14:18:43.
743847+00:00 [running]> on host elenas-ado3.ru-central1.internal
[2024-07-15T14:18:46.326+0000] {taskinstance.py:1545} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow'
AIRFLOW_CTX_DAG_ID='adcm_check' AIRFLOW_CTX_TASK_ID='runme_1' AIRFLOW_CTX_EXECUTION_DATE='2024-07-15T14:18:43.743847+00:00'
AIRFLOW_CTX_TRY_NUMBER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2024-07-15T14:18:43.743847+00:00'
```

Поиск по логам

Чтобы найти в логах интересующую вас информацию, например, сообщения об ошибках, подключитесь к хосту с логами, которые вы хотите просмотреть, и воспользуйтесь командой `grep`.

Например:

```
$ cat /var/log/airflow/dag_processor_manager/dag_processor_manager.log | grep -i -A3 -B1 error | grep -i -A3 -B1 error
```

Эта команда парсит лог менеджера процессов и ищет сообщения, содержащие слово `error`. Опция `-i` позволяет игнорировать различия в регистре, а опции `-A3 -B1` добавляют к найденной строке с ошибкой одну строчку до нее и три строчки после.

Пример вывода команды:

```
File Path                                                
```

Управление зависимостями через ADCM

Елена Костюченко

Содержание

Назад к содержанию

Обзор
Определение зависимостей в ADCM
Добавление Python-библиотеки
Добавление библиотек из приватного репозитория
Ограничение зависимостей

Обзор

Некоторые DAG Airflow требуют дополнительных Python-библиотек, которые не включены в базовое окружение. Эти библиотеки могут быть из публичного репозитория Python Package Index (PyPI), приватных репозиториев или локальных источников.

ADCM предоставляет централизованный способ управления зависимостями. Вы можете определить и обновить список необходимых Python-библиотек напрямую из интерфейса ADCM, используя конфигурационные параметры, доступные для сервиса Airflow. Все зависимости, добавленные таким образом, будут добавлены на всех хостах кластера.

Эта функция поддерживает:

- добавление библиотек из публичных репозиториев PyPI (например, `requests`, `pandas`);
- добавление библиотек из приватных или внутренних Python-пакетов, размещенных в аутентифицированных репозиториях;
- доступ к репозиториям через прокси;
- ограничение версий зависимостей.

Конфигурация Airflow в ADCM содержит редактируемые поля, которые соответствуют параметрам, используемым в `pip install`.

В этой статье описывается, как настроить управление зависимостями в Airflow через ADCM.

Определение зависимостей в ADCM

Рассмотрим DAG, имеющий следующие зависимости:

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime
import requests ①
import pandas as pd ②
import my_private_module ③
```

① Python-пакет.

② Python-пакет определенной версии.

③ Пакет из приватного репозитория.

Добавление Python-библиотеки

ADCM предоставляет поле ***Extra requirements*** в конфигурации Airflow, которое работает аналогично файлу *requirements.txt* для управления зависимостями.

Чтобы установить необходимые Python-пакеты для DAG через ADCM:

1. На странице ***Clusters*** выберите нужный кластер.
2. Перейдите на вкладку ***Services*** и нажмите на ***Airflow***.
3. Активируйте опцию ***Show advanced*** и разверните секцию ***Dependency Management***.
4. В поле ***Extra requirements*** укажите список пакетов в следующем формате: `<package-name>==<version>`. Например:

```
requests
pandas==2.2.2
```

   Если версия не указана, по умолчанию будет установлена последняя доступная. Дополнительную информацию об этом формате можно найти в статье Requirements File Format.

5. Подтвердите изменения в конфигурации, нажав ***Save***.
6. В раскрывающемся меню ***Actions*** выберите ***Sync requirements*** и нажмите ***Run***.
7. В раскрывающемся меню ***Actions*** выберите ***Restart***, убедитесь, что для параметра ***Apply configs from ADCM*** установлено значение `true`, и нажмите ***Run***.

Добавление библиотек из приватного репозитория

Если DAG используют библиотеку, размещенную в приватном репозитории, вы можете указать в ADCM хост репозитория и данные для аутентификации.

Чтобы установить необходимые Python-пакеты из приватного репозитория через ADCM:

1. На странице ***Clusters*** выберите нужный кластер.
2. Перейдите на вкладку ***Services*** и нажмите на ***Airflow***.
3. Активируйте опцию ***Show advanced*** и разверните секцию ***Dependency Management***.
4. В поле ***Extra requirements*** укажите список пакетов в следующем формате: `<package-name>==<version>`. Например:

```
my-private-pkg==0.1.0
```

5. Заполните следующие поля:
   - ***index-url*** — URL репозитория;
   - ***index-url-user*** — имя пользователя для аутентификации;
   - ***index-url-password*** — пароль для аутентификации.
6. Подтвердите изменения в конфигурации, нажав ***Save***.
7. В раскрывающемся меню ***Actions*** выберите ***Sync requirements*** и нажмите ***Run***.
8. В раскрывающемся меню ***Actions*** выберите ***Restart***, убедитесь, что для параметра ***Apply configs from ADCM*** установлено значение `true`, и нажмите ***Run***.

Дополнительно, если для доступа к публичным пакетам требуется прокси, укажите параметры прокси в следующих полях:

- ***proxy*** — адрес прокси-сервера;
- ***proxy-user*** — имя пользователя для аутентификации через прокси;
- ***proxy-password*** — пароль для аутентификации через прокси.

Если прокси не поддерживает HTTPS, укажите IP-адрес хоста или пару вида `<IP хоста>:<порт>` в поле ***trusted-host***.

Ограничение зависимостей

ADCM предоставляет возможность для управления версиями критически важных Python-пакетов с помощью файла ограничений (constraints file). Это позволяет предотвращать конфликты при добавлении новых зависимостей или случайные обновления ключевых компонентов, таких как Apache Airflow.

Файлы ограничений отличаются от файлов требований (requirements) тем, что они определяют допустимые для установки версии пакетов, но не устанавливают их автоматически.

Раздел конфигурации ***Dependency Management*** в Airflow включает следующие поля для управления ограничениями:

- ***Constraints file*** — редактируемый список ограничений версий (задается пользователем);
- ***Base constraints file*** — доступный только для чтения список ограничений, необходимых для корректной работы Airflow, который нельзя изменить с помощью ***Constraints file***.

Чтобы ограничить версии пакетов через ADCM:

1. На странице ***Clusters*** выберите нужный кластер.
2. Перейдите на вкладку ***Services*** и нажмите на ***Airflow***.
3. Активируйте опцию ***Show advanced*** и разверните секцию ***Dependency Management***.
4. В поле ***Constraints file*** перечислите пакеты в следующем формате: `<package-name>==<version>`. Например:

```
apache-airflow==2.10.5
pandas==2.2.2
requests<3.0.0
```

5. Подтвердите изменения в конфигурации, нажав ***Save***.
6. В раскрывающемся меню ***Actions*** выберите ***Sync requirements*** и нажмите ***Run***.
7. В раскрывающемся меню ***Actions*** выберите ***Restart***, убедитесь, что для параметра ***Apply configs from ADCM*** установлено значение `true`, и нажмите ***Run***.

При выполнении действия ***Sync requirements*** ADCM проверяет все зависимости на соответствие полному списку ограничений.

Если установка пакета нарушает заданные ограничения, процесс синхронизации завершается с ошибкой и в логах сохраняется запись об ошибке.

Примеры нарушений ограничений:

- попытка установить пакет, требующий более высокую версию Apache Airflow, чем разрешено;
- указание версии пакета, конфликтующей с базовыми ограничениями;
- ручная установка несовместимых пакетов в виртуальной среде Airflow.

Во всех подобных случаях синхронизация зависимостей будет остановлена для поддержания стабильности работы кластера.

Оптимизация производительности Airflow

© Елена Костыленко

Содержание

Назад к содержанию

- Настройка планировщика
  - Ключевые настройки
  - Параллелизм, высокая доступность и координация воркеров
- Рекомендации по дизайну DAG
  - Парсинг и высокоуровневый код
  - Динамический маппинг задач
  - XCom и передача данных между задачами
- Настройка воркеров и Executor
- Настройка БД метаданных (PostgreSQL)
  - Пул соединений и SQLAlchemy
- Контроль параллелизма, пулов и приоритетов
  - Пулы
  - Приоритеты и SLA
- Ресурсы Web Server
- Логирование
- Оценка результатов настройки

Эта статья описывает методы оптимизации и рекомендуемые практики, которые помогут улучшить производительность сервиса Airflow в кластере Arenadata Orchestrator. Описанные подходы основаны на последних рекомендациях Apache Airflow и применимы к версиям Airflow от 2.10.5 и выше.

Настройка планировщика

Планировщик (Scheduler) является основным ограничителем пропускной способности Airflow при большом количестве задач. Когда планировщик не успевает обрабатывать DAG, логгер сообщает задержки и увеличивается время задачи. Ниже описаны параметры для баланса. Поэтому в плане оптимизации производительности настройка планировщика дает наибольший эффект.

Если время цикла планировщика увеличивается, пробуждение числа в производительности можно выявить, проверив продолжительность парсинга DAG, проблемы в оптимизации кода DAG и влияние параметров конфигурации, которые влияют на работу планировщика. Ограничивает **max_dagruns_per_loop_per_dag** и **min_file_process_interval**.

Если задач много остается в очереди, проверьте состояние Executor, длину очереди Celery broker, значение параметров **worker_concurrency** и возможное истощение пула соединений к БД.

Ключевые настройки

Параметр	Описание	Рекомендуемое начальное значение
store_serialized_dags	Включает возможность хранения сериализованных DAG в БД, чтобы планировщику не приходилось постоянно сериализовывать DAG заново. Повторной проверкой Python на каждом цикле. Снижает нагрузку на CPU и I/O, особенно эффективно при большом количестве DAG.	True
min_file_process_interval	Минимальное количество секунд между циклами парсинга DAG. Повышает эффективность парсинга за счет более редкой обработки DAG	30s
dag_dir_list_interval	Интервал сканирования директории dags для проверки новых или удаленных файлов. Можно увеличить для снижения I/O нагрузки	60s
max_tis_per_query	Количество строк TaskInstance, получаемых за один цикл работы планировщика. Следует увеличивать, если задач очень много, чтобы сократить накладные расходы на обращение к БД и пропускной способности планировщика. Значение слишком большое может вызвать **core.parallelism**	16–128
max_dagruns_to_create_per_loop	Максимальное количество DAG, для которых планировщик запускал на один цикл (помогает распределить работу между несколькими планировщиками)	10–50

Параллелизм, высокая доступность и координация воркеров

Рекомендуется запускать планировщик на CPU-оптимизированных нодах. При высоких нагрузках на кластер используйте несколько планировщиков в режиме высокой доступности (High Availability, HA), проверьте параметры **use_row_level_locking** значение `True` и настроите опции **max_dagruns_to_create_per_loop** чтобы запускал DAG распределялись между планировщиками.

При большом числе DAG или их динамической генерации лучше увеличить CPU и память планировщика, предварительно минимизировав высокоуровневый код в DAG.

Рекомендации по дизайну DAG

Проблемы в дизайне DAG могут значительно увеличить нагрузку на планировщик и воркеры. Чтобы DAG не мешали оптимизации Airflow, используйте лучшие практики разработки DAG, описанные в этом разделе.

Парсинг и высокоуровневый код

Airflow периодически DAG заново выполняет код файлов DAG в процессе выполнения на верхнем уровне (вне функций задач или операторов). Планировщик и веб-сервер парсят DAG для построения расписания, зависимостей операций. Замедляет этот процесс.

Каждый раз при парсинге DAG выполняется весь верхний уровень файла. Если в нем есть импорты, которые занимают продолжительное время, запросы к БД или сетевые вызовы, это сильно увеличивает время парсинга и нагружает планировщик и веб-сервер. При высокой нагрузке в большом количестве DAG такие задержки накапливаются, что замедляет цикл планировщика, нарушает SLA и снижает пропускную способность кластера.

Блокирующие вызовы или большие зависимости из модульных уровней могут также вызвать нестабильность. Например, если DAG импортирует библиотеку, загрузка которой занимает несколько секунд, планировщик может зависнуть или отвечать DAG как медлительный.

Выносите операции из верхнего уровня внутрь функций задач или операторов, чтобы они выполнялись только при запуске задачи. Оптимизированный код DAG делает парсинг более быстрым и предсказуемым даже при большом количестве задач и частых обновлениях кода.

Пример проблемного DAG:

```
import pandas as pd
heavy = heavy_initialization()

with DAG(
    dag_id="bad_dag",
    start_date=datetime(2025, 1, 1),
    schedule_interval=None,
    catchup=False,
):

    @task()
    def do_work_bad():
        heavy process()

    do_work_bad()
```

1. Тяжелый импорт здесь замедляет парсинг.
2. Высокоуровневая функция выполняется при парсинге, что также замедляет работу.

Пример исправленного DAG:

```
with DAG(
    dag_id="good_dag",
    start_date=datetime(2025, 1, 1),
    schedule_interval=None,
    catchup=False,
):

    @task()
    def do_work_good():
        from walrus import heavy_initialization   1
        heavy = heavy_initialization()            2
        heavy process()

    do_work_good()
```

1. Импорт внутри задачи.
2. Вызов функции внутри задачи.

Таким образом, перенос тяжелых импортов и высокоуровневых функций в операторы задач позволяет снизить задержку парсинга и ближе использования CPU планировщика.

Динамический маппинг задач

Динамический маппинг задач (dynamic task mapping) повышает оптимизацию производительность Airflow в среде с большим количеством параметризованных или повторяющихся задач с помощью изменения кодовой базы DAG. Эта функция позволяет создавать задачи во время выполнения, а не генерировать их при парсинге. При таком создании DAG код пишется только для одной задачи, а которой создается список входных данных вместо того, чтобы вручную создавать все объекты задач. При запуске такого DAG Airflow автоматически добавляет эту задачу в несколько параллельных экземпляров, а затем создает отдельные задачи в базе метаданных. Это снижает нагрузку на планировщик, упрощает парсинг и ускоряет развертывание DAG.

XCom и передача данных между задачами

Не рекомендуется передавать большие объемы данных через XCom. Для этого лучше использовать объектное хранилище (S3, HDFS) и передавать через XCom только указатели (пути, URI).

Для небольших структурированных сообщений используйте JSON-сериализацию или удаленный XCom-бэкенд.

Настройка воркеров и Executor

Executor определяет, как Airflow выполняет задачи параллельно. В ADO используются `CeleryExecutor` и `KubernetesExecutor` в зависимости от характеристик рабочей нагрузки.

Ключевые параметры настройки Executor в ADO перечислены ниже:

Параметр	Описание	Значение по умолчанию
parallelism	Глобальный лимит на количество экземпляров задач, которые могут выполняться параллельно во всей среде Airflow	32
dag_concurrency	Максимальное количество экземпляров задач, разрешенных для одновременного выполнения в группе DAG. Предотвращает исчерпание ресурсов кластера одной группой DAG	16
max_active_runs_per_dag	Максимальное количество активных запусков для каждого DAG. Слишком большое значение может привести к перегрузке Executor и планировщика.	16
worker_concurrency	Максимальное количество задач, выполняемых одним воркером	16

Параметры параллельной работы воркеров и выделенные им ресурсы напрямую определяют эффективность выполнения задач в кластере.

Конфигурация воркеров определяет пропускную способность, скорость отклика и стабильность всей системы. Параллельное выполнение, сколько задач один воркер может выполнять одновременно. Если значение параллельности слишком низкое, кластер может недоиспользовать доступные ресурсы CPU и памяти, что приводит к медленной работе DAG и простою инфраструктуры. Если значение слишком высокое, воркер может попытаться выполнить больше задач, чем может обработать хост, что приводит к нехватке CPU, исчерпанию памяти и замедлению задач.

Например, воркер с 8 CPU и параметром параллельности **worker_concurrency** 64 может попытаться запустить слишком много процессов Python параллельно, что приведет к перегрузке из за переключения контекста и снижению производительности. Однако, если значение параллелизма равно 2 на той же машине, ресурсы будут использоваться неэффективно, что приведет к нехватке ресурсов (ожидающие задачи в очереди).

Для оптимальной производительности установите уровень параллелизма, соответствующий доступным ресурсам (задачи, памяти) и типичному профилю ресурсов ваших задач. Для простых задач (например, вызовов API) более высокий уровень параллелизма может быть эффективнее, но для более сложных задач (например, ETL или ML) более низкий уровень параллелизма обеспечивает стабильность и лучшую пропускную способность.

Рабочие процессы Airflow можно масштабировать как вертикально (увеличивая ресурсы CPU, памяти или диска), так и горизонтально, добавляя больше рабочих узлов. Рекомендуется сочетать мощность вашего оборудования с типом рабочей нагрузки.

- Задачи с высокой нагрузкой на CPU (например, преобразование данных, отправка заданий Spark) выигрывают от большого количества ядер, но требуют тщательной настройки параллелизма, чтобы задачи, связанные с CPU, не перегружали операционную.
- Задачи с высоким потреблением памяти (например, большие фреймы данных Pandas) требуют выделения большего объема памяти и более низких настроек параллелизма, чтобы предотвратить ошибки нехватки памяти.
- Смешанные рабочие нагрузки часто выигрывают от разделения рабочих процессов на пулы или очереди, чтобы разные типы задач выполнялись на рабочих процессах, оптимизированных под их потребности.

Настройка БД метаданных (PostgreSQL)

Базы метаданных часто становятся первым и наиболее критическим узким местом при масштабировании Airflow. Каждый сигнал планировщика, обновление состояния задачи, запись в лог и каждый DAG приводит к операциям чтения и записи в этой базе данных. Производительность рабочих нагрузок Airflow состоит из очень большого объема небольших, часто выполняемых вставок, обновлений и запросов. база данных должна быть специально настроена для высокой пропускной способности записи и чтения.

Рекомендации по оптимизации базы данных:

- Установите значение **max_connections** достаточно большим для поддержки компонентов Airflow (планировщика, веб-сервера, рабочих процессов, CLI), но обратите внимание на использование пула соединений, чтобы избежать исчерпания памяти.
- Увеличьте значение **wal_buffers** и установите правильное значение **checkpoint_completion_target**, чтобы очистка WAL не блокировала частые обновления.
- Убедитесь, что хранилище оптимизировано для быстрой случайной записи (например, SSD).
- Таблицы метаданных Airflow, такие как `task_instance`, `dag_run` и `log`, быстро растут. Если автоочистка недостаточно производительна, разделите таблицы, может привести к замедлению запросов. Ежегодное чистку автоочистки для или большой таблицы или запланируйте ручные операции `VACUUM ANALYZE` для обеспечения предсказуемой производительности.
- Партиционирование или периодическое архивирование истории старых задач и логов позволяет контролировать размеры таблицы.

Также важно масштабировать базу данных вертикально (процессор, память, IOPS) по мере роста рабочих процессов. Подробнее об оптимизации Amazada Postgres читайте в статье Оптимизация производительности.

Пул соединений и SQLAlchemy

Параметр	Описание	Рекомендуемое начальное значение
sql_alchemy_pool_size	Размер пула SQLAlchemy — это максимальное количество подключений к базе данных в пуле. Значение `0` означает отсутствие ограничений	20
sql_alchemy_max_overflow	Максимальный размер переполнения пула. Когда количество соединений достигает размера, заданного параметром `pool_size`, будут доступны дополнительные соединения в пределах этого лимита. Когда для дополнительных соединений возвращаются в пул, они отключаются и удаляются. Общее количество одновременных соединений с базой данных может быть `pool_size` + `max_overflow`, а общее количество ожидающих соединений, которые могут ожидать, равно `pool_size` + `max_overflow` можно установить равным `-1`, чтобы обозначить отсутствие ограничения переполнения. Ограничение на общее количество одновременных соединений не накладывается. Значение по умолчанию `10`	30
sql_alchemy_pool_recycle	Время перезапуска пула SQLAlchemy — это количество секунд, в течение которых соединение может простаивать в пулах, прежде чем оно будет переоткрыто. Если количество подключений к базе данных будет превышено, более низкое значение настройки позволит системе быстрее восстанавливаться	1800

Увеличение **sql_alchemy_pool_size** сокращает время ожидания подключений к базе данных. Убедитесь, что значение **max_connections** Postgres соответствует общему количеству подключений Airflow и других пользователей базы данных.

Контроль параллелизма, пулов и приоритетов

Пулы

По умолчанию Airflow планирует столько задач, сколько позволяет возможности Executor и параметров воркеров. Это обеспечивает максимальную пропускную способность, но может легко привести к конкуренции за ресурсы, например, к нехватке базы данных очередью большим количеством задач превышения ограничений сервисов API или исчерпыванию рабочего процесса доступными соединениями с ограниченными ведомыми ресурсами. Пулы действуют как регулятор параллелизма: они группируют задачи по категориям ресурсов и ограничивают их количество, которое может выполнять одновременно во всем кластере Airflow.

Один из лучших способов использования этр функции — создавать пулы для операций, ограниченных ресурсами (например, `hadoop_jobs`, `db_ingest`, `api_calls`), явно задавать количество слотов и назначать задачи пулам с помощью аргумента пула в операторах.

Вы можете настроить пулы с помощью Airflow CLI, или веб-интерфейса Airflow.

Приоритеты и SLA

Используйте **priority_weight** для задач, чтобы влиять на распределение слотов при заполнении пулов. Сочетайте это с пулами, чтобы обеспечить более раннее планирование высокоприоритетных задач.

Контролируйте использование на уровне DAG (например, **max_active_runs_per_dag**), вместе с пулами помогает создать лимиты выполнения задач и предотвратить исчерпание ресурсов. По умолчанию, если DAG запускается часто или имеет большой объем незавершенных запусков, Airflow может попытаться активировать несколько запусков DAG одновременно. Параметры **max_active_runs_per_dag** ограничивают количество запусков DAG, которые могут выполняться параллельно, обеспечивая равномерное выполнение и предыдущие перегрузку системных ресурсов. В сочетании с пулами настройки уровня DAG управляют тем, как сценарий рабочей процесс использует доступные ресурсы, но пулы в конечном счете контролируют параллельные рабочие процессы и типовые ресурсы.

Вы также можете реализовать ограничить скорость выполнения задач вручную уровне, которой в некоторых случаях запускает данные в промежуточную очередь (например, Kafka).

Ресурсы Web Server

Веб-сервер Airflow обслуживает запросы UI, API и метаданных DAG подач, объем которых линейно увеличивается с количеством пользователей и активностью DAG. Недостаток ресурсов приводит к медленной загрузке страниц, тайм-аутам и недоступности сервиса в периоды пиковой нагрузки.

Для оптимизации производительности важно масштабировать веб-сервер как вертикально, выделяя достаточно ресурсов CPU, памяти и пропускной способности сети, так и горизонтально, запуская несколько реплик за балансировщиком нагрузки. Дополнительно можно использовать Gunicorn для воркеров, чтобы регулировать их количество и число потоков в соответствии с доступными ресурсами, что позволяет эффективно обслуживать запросы без перегрузки CPU и памяти.

При высокой нагрузке на кластер можно использовать специальный сервер статистики, ограничение количества загружаемых DAG за сеанс пользователя и вынести тяжелые вычисления по ресурсам API запросов на отдельные экземпляры. Это позволяет повысить до производительность интерфейса Airflow при высокой активности пользователей.

Логирование

Избегайте ведения только локальных логов при масштабировании рабочих процессов. Настройте удаленное логирование (например, в S3/GCS/Elasticsearch) для централизованных логов и сокращения дискового заказа вывода в рабочих процессах. Настройте параметры **remote_base_log_folder** и **remote_log_conn_id** через ADCM.

Оценка результатов настройки

Перед внесением существенных изменений в конфигурацию (например, повышение уровня параллелизма или размера пула баз данных) рекомендуется провести нагрузочные тесты с репрезентативным DAG, чтобы оценить влияние на планировщик, базу данных и воркеры. Лучший способ оценить эффективность настройки производительности — это использовать A/B тестирование, чтобы изолировать влияние других факторов.

Обзор UI в Airflow

Версия: 2.0

Содержание

- Airflow UI
  - Grids
  - Datasets
  - Security
  - Browse
  - Admin
  - Docs
- Flower UI

Назад к содержанию

Содержание

Airflow UI
- Grids
- Datasets
- Security
- Browse
- Admin
- Docs

Flower UI

Адреса веб-интерфейсов Airflow2

Компонент	Дефолтный порт
Airflow Web 2	http://localhost:8080/
Flower	http://localhost:5555/

Чтобы перейти к веб-интерфейсу Airflow2 нужно ввести **http://localhost:8080/** в адресной строке.

Если вы не обновили конфигурацию docker-compose.yaml **8080** на порт **8081** по причине занятости 8080:

Ссылка на веб-интерфейс сервиса Airflow в Docker

Airflow UI

Grids

Страница **Grids** — это основная страница пользовательского интерфейса Airflow. Частью отображается поле настройки данных DAG-ов(фильтр). В этом плане страница разделена на несколько разделов, которые мы рассмотрим ниже:

- **DAG** — дефолтный DAG, который мы выбрали. Можно переключиться на другой DAG, если есть несколько DAG-ов. Также можно ввести название нужного DAG-а в поисковой строке.
- **Owner** — владелец DAG-а. Также можно выбрать DAG-и определенного владельца (фильтр). Чтобы отобразить DAG-и с определенным владельцем, нужно в DAG-е задать параметр owner.
- **Runs** — статусы всех выполненных DAG-ов. Если Runs не отображается, нужно в DAG-е поставить параметр dagrun_timeout.
- **Recent Tasks** — статусы последних DAG-ов. Если Recent Tasks не отображается, нужно в DAG-е поставить параметр dagrun_timeout.
- **Last Run** — дата последнего запуска DAG-а. Если параметр не задан, отображается дата последнего запуска DAG-а.
- **Schedule** — расписание DAG-а. Если параметр не задан, отображается дефолтное расписание. Обновляется после обновления расписания, независимо от выполнения DAG-а.
- **Actions** — Trigger, Refresh, Delete. Позволяет запустить выполнение DAG-а, обновить DAG-и, удалить выполнение DAG-а.
- **Links** — Trigger, Refresh, Delete. Позволяет запустить выполнение DAG-а, обновить DAG-и, удалить выполнение DAG-а.

Также на странице доступны следующие переключатели (DAG):

- **Filter DAGs by tag** — фильтрация DAG-ов по тегам. Если DAG-ов несколько, можно выбрать нужный тег, чтобы отобразить только DAG-и с выбранным тегом.
- **Filter DAGs by owner** — фильтрация DAG-ов по владельцам. Если DAG-ов несколько, можно выбрать нужного владельца, чтобы отобразить только DAG-и с выбранным владельцем.
- **All** — все DAG-и.
- **Active** — все активные DAG-и.
- **Paused** — все приостановленные DAG-и.

Airflow UI. Страница Grids

Datasets

На странице **Datasets** отображаются последние выполненные задачи, которые могут использовать данные (после выполнения задачи в статусе success). При наличии нескольких задач, которые могут использовать данные, отображается последняя выполненная задача.

Airflow UI. Страница Datasets

Security

Раздел **Security** отображает доступ к информации о пользователях, роли, группы безопасности и к каждому правилу безопасности в отдельности.

List Users

Страница **List Users** показывает всех пользователей Airflow. Каждый пользователь может следующие характеристики: имя, фамилию, имя пользователя, адрес электронной почты, роль, последний раз выполненный вход, статус (активный/неактивный).

Чтобы добавить пользователя, нужно в терминале ввести следующую команду:

```
airflow users create -r Admin -u admin -e admin@example.com -f admin -l user -p admin
```

Airflow UI. Страница List Users

List Roles

Страница **List Roles** показывает роли пользователей. Каждая роль имеет следующие атрибуты: название роли, разрешения, назначенные к роли.

Чтобы добавить роль, нужно в терминале ввести следующую команду:

```
airflow roles create Viewer
```

Airflow UI. Страница List Roles

User Statistics

Страница **User statistics** отображает визуализацию статистики выполнения пользователей. Чтобы посмотреть статистику нужно выбрать интересующего пользователя, нажать на кнопку **Go** и выбрать период времени.

Actions, Resources, Permissions

- **Actions** — все страницы отображаются в таком разделе, которые можно назначить каждому пользователю.
- **Resources** — все страницы отображаются в таком разделе, которые можно назначить каждому пользователю.
- **Permissions** — все страницы отображаются в таком разделе, которые можно назначить каждому пользователю.

User Pools

Страница **User Pools** показывает пулы пользователей.

Чтобы добавить пользователя, нужно в терминале ввести следующую команду:

```
airflow pools -s <name_pool_1> 128 "<description_pool_1>" <name_pool_2> 256 "<description_pool_2>"
```

Например, чтобы добавить пулы **Pool1** и **Pool2**, нужно ввести:

```
airflow pools -s Pool1 128 "Pool1" Pool2 256 "Pool2"
```

Airflow UI. Страница User Pools

Audit

Страница **Audit** показывает полный журнал всех действий, которые были выполнены в системе Airflow, или выполнены администратором.

Airflow UI. Страница Audit

Audit Logs

В разделе **Audit Logs** также отображается журнал всех действий выполненных в системе Airflow, но в отличие от предыдущего раздела, в таблице отображаются только те действия, которые были выполнены пользователем, или выполнены администратором.

Airflow UI. Страница Audit Logs

Task Instances и Task Reschedules

Страницы **Task Instances** показывают информацию о каждой задаче DAG, с которыми они взаимодействуют, причем каждая отображается в отдельной строке в форме таблицы.

Страницы **Task Reschedules** показывают информацию о каждой задаче DAG, которая была перезапущена, причем каждая отображается в отдельной строке в форме таблицы.

Airflow UI. Страница Task Instances

Browse

Jobs

Страница **Jobs** отображает статусы всех задач(job), запущенных на сервере. Airflow.

Airflow UI. Страница Jobs

Slots Pools

Страница **Slots Pools** отображает пулы задач(slots) и информацию о них.

Чтобы добавить пулы задач, нужно в терминале ввести следующую команду:

```
airflow pools -s <name_pool_1> 128 "<description_pool_1>" <name_pool_2> 256 "<description_pool_2>"
```

Например, чтобы добавить пулы **Pool1** и **Pool2**, нужно ввести:

```
airflow pools -s Pool1 128 "Pool1" Pool2 256 "Pool2"
```

Airflow UI. Страница Slots Pools

Logs

Страница **Logs** отображает общие логи сервера.

Airflow UI. Страница Logs

DAG Dependencies

Страница **DAG Dependencies** показывает графы зависимостей DAG, при переходе, на которую отображается следующая информация:

Airflow UI. Страница DAG Dependencies

Admin

Раздел **Admin** предназначен для ручного управления ресурсами и информацией в графическом интерфейсе Airflow.

Variables

Страница **Variables** — это ключ-значение переменные, которые задаются глобально. Страница **Variables** позволяет посмотреть, отредактировать и импортировать в JSON-формат все переменные, которые были добавлены в Airflow.

Чтобы добавить переменную, нужно в терминале ввести следующую команду:

```
airflow variables -s <name_variable_1> <value_variable_1> <name_variable_2> <value_variable_2>
```

Например, чтобы добавить переменные **var1** и **var2**, нужно ввести:

```
airflow variables -s var1 val1 var2 val2
```

Airflow UI. Страница Variables

Configurations

На странице может отображаться настройка и конфигурация всего сервиса Airflow. По умолчанию страница отключена.

Airflow UI. Страница Configurations

Plugins

На странице отображаются все плагины, установленные в системе. Чтобы добавить плагин, нужно его поместить в директорию plugins.

Airflow UI. Страница Plugins

Pools

Страница Pools отображает пулы задач(slots) и информацию о них.

Чтобы добавить пулы задач, нужно в терминале ввести следующую команду:

```
airflow pools -s <name_pool_1> 128 "<description_pool_1>" <name_pool_2> 256 "<description_pool_2>"
```

Например, чтобы добавить пулы **Pool1** и **Pool2**, нужно ввести:

```
airflow pools -s Pool1 128 "Pool1" Pool2 256 "Pool2"
```


Airflow UI. Страница Pools

Providers

Страница Providers отображает пулы провайдеров(providers) сервиса Airflow. Все провайдеры, поставляемые с сервисом **Apache Airflow**.

Airflow UI. Страница Providers

Plugins

Страница Plugins отображает пулы плагинов(plugins) сервиса Airflow. Чтобы добавить плагин, нужно его поместить в директорию plugins.

Airflow UI. Страница Plugins

Providers

Страница Providers отображает пулы провайдеров(providers) сервиса Airflow. Все провайдеры, поставляемые с сервисом **Apache Airflow**.

Airflow UI. Страница Providers

Docs

Раздел **Docs** позволяет ознакомиться с документацией:

- **Documentation** — официальная документация Airflow.
- **Airflow Website** — официальный сайт Airflow.
- **GitHub Repo** — официальный проект Airflow на GitHub.
- **REST API Reference (Swagger)** — документация REST API в формате Swagger.
- **REST API Reference (Redoc)** — документация REST API в формате Redoc.

Flower UI

Flower — это сервис, который позволяет отслеживать и управлять кластером celery. Чтобы зайти в интерфейс Flower, нужно ввести в адресную строку **http://localhost:5555/**.

- **Workers** — показывает статусы каждого из работающих в кластере celery-воркеров.
- **Tasks** — показывает статистику задач, выполненных на каждом из celery-воркеров. Можно посмотреть статус каждой задачи, время выполнения и все аргументы.
- **Broker** — показывает статистику по брокерам (в проекте Airflow — Redis).
- **Documentation** — показывает официальную документацию Flower.

Airflow UI. Страница Flower

Для использования дашбордов установите Flower через команду в docker-compose.yaml.

Создание простого DAG

Содержание

Назад к содержанию

Шаг 1. Подготовка DAG-файла

Шаг 2. Запуск через Web-интерфейс

Шаг 3. Проверка результатов

Данный раздел описывает процедуру создания и запуска вашего первого DAG с использованием файлов CSV. Этот DAG будет импортировать процесс ETL (Extract, Transform, Load) и включать следующие задачи:

- **Extract**: Эта задача загружает записи из исходного файла `people.csv` в следующем разделе. Мы будем использовать тестовый файл `people_ages_titles.csv`, содержащий списки из 1000 человек с их именами и профессиями.
- **Transform**: Эта задача выбирает записи, полученные в предыдущей задаче, по следующему критерию: возраст людей должен быть не старше 18 лет.
- **Load**: Эта задача загружает отфильтрованные записи в отдельный файл с предыдущей задачи. В новый файл CSV.

> **ПРИМЕЧАНИЕ**
>
> Мы приняли вы можете модифицировать этот пример путем замены входных файлов на соединения с внешними базами данных.
>
> Другие примеры можно найти в airflow/example_dags.
>
> Описанный в данной статье пример запущен для версии Airflow 1.10.11.

Шаг 1. Подготовка DAG-файла

Подготовьте DAG с помощью следующих шагов:

1. Скопируйте файл `elt_task.py` в папку вашего DAG, которая обычно расположена в домашнем каталоге Airflow и называется dags. Выполним содержимое файла:

```
$ nano ~/airflow/dags/elt_task.py
```

```
import pandas as pd ❶
from datetime import timedelta, datetime
from airflow import DAG
from airflow.operators.python_operator import PythonOperator

day = [0x0]*5;c,hour=0,0
for i in open('/var/log/apache2/access-5'):
    ahost,atime=i[:i.find(' ')],i[i.rfind('[')+1:i.rfind(']')]
    if ahost.find('.')!=-1:
        day[int(atime[0:2])-1]+=1
def extract_people(path=None, **kwargs):
    extracted_people = pd.read_csv(path, sep=',', header=0)
    context['ti'].xcom_push(key='extracted_people', value=extracted_people)

def transform_people(*econtext):
    extracted_people = econtext['ti'].xcom_pull(key='extracted_people', task_ids=('extract_people'))[0]
    transformed_people = []
    for person in extracted_people.values:
        if int(person[2]) <= 18:
            transformed_people.append([
                person[0],
                person[1],
                person[2]
            ])
    context['ti'].xcom_push(key='transformed_people', value=transformed_people)

def load_people(path=None, **kwargs):
    transformed_people = econtext['ti'].xcom_pull(key='transformed_people', task_ids=('transform_people'))
    loaded_people = pd.DataFrame(transformed_people)
    loaded_people.to_csv(path, index=False)

task_extract_people = PythonOperator( ❷
    task_id='extract_people',
    python_callable=extract_people,
    op_kwargs={'path': '/opt/airflow/input/people_ages_titles.csv'},
    dag=dag,
    provide_context=True
)

task_transform_people = PythonOperator( ❸
    task_id='transform_people',
    python_callable=transform_people,
    dag=dag,
    provide_context=True
)

task_load_people = PythonOperator( ❹
    task_id='load_people',
    python_callable=load_people,
    op_kwargs={'path': '/opt/airflow/output/loaded_people.csv'},
    dag=dag,
    provide_context=True
)

task_extract_people >> task_transform_people >> task_load_people ❺
```

- **Импортирование** Импортируем пакет `pandas` для обработки файлов CSV.
- **Определение DAG**. Создаем объект `ETL_test` типа DAG с планировщиком по запуску, начиная с 2021-12-11 00:00:00.
- **Первая функция Python**. Вы используете первую задачу `extract_people`. Эта функция считывает файл CSV по пути, заданному переменной `path`, с помощью пакета `pandas`. Она использует объект `context` для передачи загруженных данных через сервис трансляторов данных XCOM в следующую задачу `transform_people`.
- **Вторая функция Python**. Вы используете вторую задачу `transform_people`. Эта функция получает данные, собранные первой задачей `extract_people`, и выделяет из них записи с определенным значением полезной `Age`, меньшим или равным 18. Она выполняет нефильтрованный ранний выход и делает свои частичные результаты для следующей задачи `load_people`.
- **Третья функция Python**. Вы используете третью задачу `load_people`. Эта функция получает данные, обработанные второй задачей `transform_people`, и сохраняет их в новом файле по пути, заданному переменной `path`. Для этой цели она использует модуль `pandas`.
- **Декларирование задач**. Этот фрагмент определяет операторы, функции и входные параметры, необходимые для задач. Он также привязывает задачи к графу DAG задействованы.
- **Определение зависимостей**. Этот фрагмент задает зависимости между созданными задачами.

> **ВАЖНО!**
>
> Если вы используете Airflow, устанавливаемый в контейнере Docker (как это поставляется в составе ADS), учитывайте различия в использовании пути: например, используйте относительно вашего пользовательского.

2. Убедитесь, что файл расположен в каталоге DAG с помощью следующей команды:

```
$ ls -la /u\s/airflow/home/oepo/
```

Вывод на консоль приведен ниже:

```
total 8
drwxr-xr-x. 2 5000 5000  65 Dec 16 17:11 .
drwxr-xr-x. 5 5000 5000 135 Dec 16 17:11 ..
-rw-r-----. 1 5000 5000 799 Dec 16 07:18 admin_check.py
-rw-r-----. 1 5000 5000 1775 Dec 16 13:10 ETL_test.py
drwxr-xr-x. 2 5000 5000  74 Dec 16 13:10 __pycache__
```

3. Создайте папки input и output в домашнем каталоге Airflow:

```
$ sudo mkdir /u\s/airflow/home/input
$ sudo mkdir /u\s/airflow/home/output
```

Убедитесь, что эти папки созданы в домашнем каталоге Airflow:

```
$ ls -la /u\s/airflow/home/
```

Вывод на консоль приведен ниже:

```
total 48
drwxr-xr-x. 6 5000 5000  126 Dec 16 13:56
drwxr-xr-x. 3 root root   20 Dec 16 17:27
-rw-r-----. 1 root root 37224 Dec 16 07:18 airflow.cfg
-rw-r-----. 1 5000 5000   3 Dec 16 11:07 airflow-webserver.pid
drwxr-xr-x. 5 5000 5000  45 Dec 16 13:11 dags
drwxr-xr-x. 2 root root   6 Dec 16 17:27 input
drwxr-xr-x. 3 5000 5000 121 Dec 16 13:05 logs
drwxr-xr-x. 2 root root   6 Dec 16 17:27 output
-rw-r-----. 1 5000 5000 2528 Dec 16 07:18 unittests.cfg
```

4. Загрузите исходный файл `people_ages_titles.csv` в локальную файловую систему сервера Airflow. Убедитесь, что файл загружен успешно, используя следующую команду:

```
$ ls -la ~
```

Вывод на консоль приведен ниже:

```
total 44
drwxr----. 7 dasha dasha  194 Dec 17 15:41
drwxr-xr-x. 3 root  root   19 Aug 31 11:55
-rwxrwxrwx. 1 dasha dasha   27 Aug 31 19:24 ansible
-rw-r-----. 1 dasha dasha 23565 Dec 17 10:17 bash_history
-rw-r-----. 1 dasha dasha   18 Apr  1  2020 bash_logout
-rw-r-----. 1 dasha dasha  193 Apr  1  2020 bash_profile
-rw-r-----. 1 dasha dasha  231 Apr  1  2020 bashrc
-rwxrwxrwx. 1 dasha dasha   27 Aug 31 19:24 hosts
-rw-rw----. 1 dasha dasha  727 Nov 25 07:05 dasha
-rwxrwxrwx. 1 dasha dasha 17461 Dec 17 15:44 people_ages_titles.csv
drwxr----. 5 dasha dasha   39 Dec  4 14:06 .pki
drwxr-----. 1 dasha dasha   29 Dec 16 07:21 .cpline
drwx------. 2 dasha dasha   29 Aug 31 11:55 .ssh
```

5. Скопируйте файл people_ages_titles.csv в ранее созданную папку input:

```
$ sudo cp ~/people_ages_titles.csv /u\s/airflow/home/input
```

Убедитесь, что операция выполнилась успешно, используя следующую команду:

```
$ ls -la /u\s/airflow/home/input
```

Вывод на консоль приведен ниже:

```
total 20
drwxr-xr-x. 2 root root   36 Dec 16 14:03
drwxr-xr-x. 6 5000 5000  126 Dec 16 13:56
-rw-r-----. 1 root root 17461 Dec 16 14:03 people_ages_titles.csv
```

6. Измените владельца созданных папок input and output, включая их содержимое, на `airflow`. В противном случае возможно возникновение ошибки доступа к файлам в процессе запуска DAG. Для смены владельца используйте следующие команды:

```
$ sudo chown -R 5000:5000 /u\s/airflow/home/input
$ sudo chown -R 5000:5000 /u\s/airflow/home/output
```

Для проверки результата воспользуйтесь следующей командой:

```
$ ls -la /u\s/airflow/home/
```

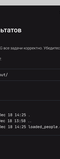
Вывод на консоль приведен ниже:

```
total 48
drwxr-xr-x. 6 5000 5000  126 Dec 16 13:56
drwxr-xr-x. 3 root root   20 Dec 16 17:27
-rw-r-----. 1 root root 37224 Dec 16 07:18 airflow.cfg
-rw-r-----. 1 5000 5000   3 Dec 16 11:07 airflow-webserver.pid
drwxr-xr-x. 5 5000 5000  45 Dec 16 13:11 dags
drwxr-xr-x. 2 5000 5000  36 Dec 16 14:03 input
drwxr-xr-x. 3 5000 5000 137 Dec 16 14:12 logs
drwxr-xr-x. 2 5000 5000  36 Dec 16 14:03 output
-rw-r-----. 1 5000 5000 2528 Dec 16 07:18 unittests.cfg
```

Шаг 2. Запуск DAG через Web-интерфейс

Для запуска DAG и проверки результатов воспользуйтесь следующими шагами:

1. Откройте домашнюю страницу Web-интерфейса Airflow. Если все настройки правильны, вы увидите состояние графа по вкладке **DAGs**. В противном случае будут отображены ошибки:



Отображение новых DAG

2. Активизируйте DAG с помощью щелчка на переключе.
   - Нажмите перезагрузить и увидите запуска согласно настроенному расписанию.
   - Откройте DAG и щелкните на иконке **Trigger Dag** для немедленного запуска DAG.

Активирование DAG

3. После запуска DAG вы можете увидеть информацию о его состоянии на домашней странице: количество успешных (successful), неудачных (failed), активных (running) и других запусков DAG, а также состояние экземпляров задач. Для просмотра подробной информации о запусках конкретного DAG (DAG runs), щелкните на названии DAG.

DAG выполняется

4. Просматривая подробную информацию, вы можете увидеть состояние всех запусков DAG и экземпляров задач. При необходимости изменим уровень просмотра на **Graph View** для анализа длительности выполнения задач или выполнения других действий, доступных на текущей странице.

Подробный обзор запусков DAG

Если в процессе исполнения DAG появятся ошибки, страница будет выглядеть наше представление ниже.

Страница с ошибками

Шаг 3. Проверка результатов

В завершение, проверим исполнение на DAG все задачи корректно. Убедитесь, что выходной файл loaded_people.csv появился в папке output.

1. Проверьте содержимое папки output:

```
$ ls -la /u\s/airflow/home/output/
```

Вывод на консоль приведен ниже:

```
total 4
drwxr-xr-x. 2 5000 5000  36 Dec 16 14:25
drwxr-xr-x. 6 5000 5000 126 Dec 16 13:56
-rw-r-----. 1 5000 5000 452 Dec 16 14:25 loaded_people.csv
```

2. Получите выходные данные:

```
$ cat /u\s/airflow/home/output/loaded_people.csv
```

Данные выглядят следующим образом:

```
Name,Age
Arnold Bottle,18
Campbell Todd,18
Thompson Barrel,18
Hernandez Ball,18
Davis Alan,18
Vaughn Bella,18
Munoz Bizarre,18
Norton Bertha,18
Nolan Isabella,18
Bailey Cameron,18
Ramos Charles,18
Hayes Kyle,18
Torres Elsie,18
Phelps Rena,18
Hill Katherine,18
Hudson Adela,18
Fitzgerald Timothy,18
Stephens Ruth,18
Ramos Catherine,18
Casey Louise,18
```

Работа с TaskFlow

Елена Костоменко

Содержание

Назад к содержанию

Использование TaskFlow с традиционными операторами
- Передача данных между задачами
- Кросс-функциональные зависимости

Структура DAG в TaskFlow

TaskFlow API, добавленный в Airflow 2, представляет новый подход к созданию DAG, который позволяет передавать параметры от функции к функции напрямую и помогает сделать код DAG более лаконичным и удобным для чтения.

В предыдущих версиях Airflow наиболее распространенным способом написания DAG было использование операторов Python (PythonOperator). В традиционном подходе к созданию DAG задачи описываались следующим образом:

```
def extract_people(path: str, **context) -> None:
    extracted_people = pd.read_csv(path,sep=',',header=1)
    context['ti'].xcom_push(key='extracted_people', value=extracted_people)

def load_people(path: str, **context) -> None:
    extracted_people = context['ti'].xcom_pull(key='extracted_people', task_ids=['extract_people'])[0]
    loaded_people = pd.DataFrame(extracted_people[0])
    loaded_people.to_csv(path, index=None)

task_extract_people = PythonOperator(
    task_id='extract_people',
    python_callable=extract_people,
    op_kwargs={'path': '/opt/airflow/input/people_ages_titles.csv'},
    dag=dag,
    provide_context=True
)

task_load_people = PythonOperator(
    task_id='load_people',
    python_callable=load_people,
    op_kwargs={'path': '/opt/airflow/output/loaded_people.csv'},
    dag=dag,
    provide_context=True
)

task_extract_people >> task_load_people
```

Это упрощенная версия традиционного DAG. Полный пример доступен в статье Создание простого DAG.

Пример выше иллюстрирует ограничения традиционного подхода, которые могут привести к раздутому коду:

- Необходимость использования объекта контекста для передачи данных в следующую задачу через сервис XCom.
- Необходимость отдельно определять операторы, функции и входные параметры, которые будут использованы в задачах.

В TaskFlow API есть специальные декораторы, которые убирают необходимость использовать традиционные операторы и обеспечивают передачу данных между задачами с помощью функций того же XCom, но без необходимости их явного вызова.

При создании декорированной функции не нужно указывать `task_id` или `python_callable`. TaskFlow использует имя функции в качестве идентификатора задачи, но вы можете также передать `task_id` в декораторе, чтобы задать другое имя.

Airflow поддерживает декораторы для задач, групп задач и DAG. В парадигме TaskFlow тот же пример будет выглядеть следующим образом:

```
@task(task_id="extract_people", retries=2) (1)
def extract_people(path: str) -> pd.DataFrame: (2)
    extracted_people = pd.read_csv(path, sep=',', header=1)
    return extracted_people (3)

@task()
def transform_people(extracted_people: pd.DataFrame) -> list:
    transformed_people = []
    for person in extracted_people.values:
        if int(person[1]) <= 18:
            transformed_people.append({
                'Name': person[0],
                'Age': person[1]
            })
    return transformed_people

extracted_data = extract_people(SOURCE_PATH) (4)
transformed_data = transform_people(extracted_data)
```

1. Определите задачу с помощью декоратора `@task`. При необходимости передайте нужные параметры задачи в декораторе.
2. Определите функцию задачи.
3. Здесь больше нет необходимости использовать `xcom_pull` и `xcom_push`, поскольку возвращаемое функцией значение автоматически передается другим задачам через XCom, когда определяется зависимости.
4. Определите зависимости задач, вызвав функции задач.

Использование TaskFlow с традиционными операторами

В то время, как традиционные операторы предлагают более детальный контроль над конвейерами данных, TaskFlow обеспечивает более четкие переходы в рабочих процессах и более читаемый код.

В Airflow вы можете использовать как традиционные операторы, так и API TaskFlow.

Передача данных между задачами

Чтобы перенести данные из декорированной задачи в традиционную, используйте атрибут `.output` объекта задачи при вызове декорированной функции.

Например:

```
def traditional_function():
    return 10

@task
def taskflow_function(x):
    return x * 5

sum_task = PythonOperator(
    task_id="sum_task",
    python_callable=traditional_function,
)

taskflow_function(sum_task.output)  # the task will return 15
```

Чтобы перенести данные из традиционной задачи в декорированную, укажите вызванную декорированную задачу непосредственно в параметре традиционного оператора.

Например:

```
@task
def taskflow_function():
    return 10

def traditional_function(x):
    return x * 5

sum_task = PythonOperator(
    task_id="sum_task",
    python_callable=traditional_function,
    op_args=[taskflow_function()]
)

traditional_function >> taskflow_function()

# sum_task will return 15
```

Кроме того, существует возможность передавать данные с помощью `xcom_pull` и `xcom_push` между традиционными и декорированными функциями.

Поскольку декораторы TaskFlow используют ту же традиционную функциональность XCom для сохранения возвращаемых значений функции, можно извлечь данные из декорированной задачи, используя тот же синтаксис `xcom_pull`.

Например:

```
@task
def taskflow_function():
    return 10

def traditional_function(**kwargs):
    ti = kwargs['ti']
    received_data = ti.xcom_pull(task_ids="taskflow_function")
    print(received_data)

traditional_task = PythonOperator(
    task_id="traditional_task",
    python_callable=traditional_function,
)

traditional_function >> taskflow_function()

# sum_task will return 15
```

ВАЖНО

TaskFlow сохраняет возвращаемое значение функции только в том случае, если оно используется где-либо еще в DAG.

Кросс-функциональные зависимости

При использовании двух типов функций в одном DAG необходимо определить зависимости в соответствии с каждым типом функции: передача параметров от функции к функции – для TaskFlow, и использование операторов битового сдвига – для традиционных функций.

Например:

```
@task
def taskflow_function_10():
    return 10

@task
def taskflow_function_20(x):
    return x + 20

def traditional_function(x):
    return x + 30

sum_task = PythonOperator(
    task_id="sum_task",
    python_callable=traditional_function,
)

taskflow_function_20(taskflow_function_10()) >> traditional_function

# sum_task will return 60
```

Структура DAG в TaskFlow

Задачи в Airflow записываются как функции Python, декорированные `@task`. При этом сами задачи должны находиться внутри функции DAG, декорированной `@dag`.

Например:

```
from datetime import timedelta, datetime
import pandas as pd
from airflow.decorators import dag, task (1)


SOURCE_PATH = '/opt/airflow/input/people_ages_titles.csv'
TARGET_PATH = '/opt/airflow/output/loaded_people.csv'

@dag(                                                   (2)
    schedule_interval=timedelta(days=1),
    start_date=datetime(2021, 12, 17),
    catchup=False,
    tags=['example'],
)
def etl_example():

    @task()                                             (3)
    def extract_people(path: str) -> pd.DataFrame:
        extracted_people = pd.read_csv(path, sep=',', header=1)
        return extracted_people

    @task()
    def transform_people(extracted_people: pd.DataFrame) -> list:
        transformed_people = []
        for person in extracted_people.values:
            if int(person[1]) <= 18:
                transformed_people.append({
                    'Name': person[0],
                    'Age': person[1]
                })
        return transformed_people

    @task()
    def load_people(transformed_people: list, path: str) -> None:
        loaded_people = pd.DataFrame(transformed_people)
        loaded_people.to_csv(path, index=None)

    extracted_data = extract_people(SOURCE_PATH)        (4)
    transformed_data = transform_people(extracted_data)
    load_people(transformed_data, TARGET_PATH)

etl_dag = etl_example()                                 (5)
```

1. Импортируйте декораторы задач (task) и DAG.
2. Создайте экземпляр DAG с помощью декоратора `@dag` и его функцию. По умолчанию имя функции используется в качестве DAG ID.
3. Внутри функции DAG определите функции задач.
4. Определите зависимости, чтобы обеспечить передачу данных между задачами.
5. Вызовите функцию DAG. Все декорированные функции в DAG должны быть вызваны в файле этого DAG, чтобы Airflow мог их зарегистрировать.

Использование сенсоров в Airflow

Елена Костюченко

Содержание

Назад к содержанию

Параметры сенсора
Работа с сенсорами
Создание кастомного сенсора

Сенсор в Airflow — это оператор, который используется, когда в DAG требуется дождаться выполнения условия для продолжения выполнения задач.

Условия срабатывания сенсора зависят от его типа. Airflow предоставляет набор стандартных операторов, которые позволяют, например, ждать истечения определенного времени (TimeDeltaSensor), ждать появления файла (FileSensor) или завершения задачи в другом DAG (ExternalTaskSensor). Когда условие сенсора выполняется, он может запустить последующие задачи.

Параметры сенсора

У сенсоров есть следующие базовые параметры:

- `poke_interval` — время, в течение которого задача должна ждать перед следующей проверкой условия. Значение по умолчанию — 60 секунд.
- `timeout` — максимальное время работы сенсора. Значение по умолчанию — 7 дней или 60 * 60 * 24 * 7 секунд. Когда время истечет, задача сенсора будет завершена. Значение должно быть меньше, чем дата следующего запуска DAG, чтобы избежать зависания задач.
- `mode` — индикатор того, как долго задача сенсора будет занимать рабочий слот. Может принимать одно из двух значений:
  - `poke` — сенсор всегда занимает слот, пока не завершится,
  - `reschedule` — сенсор занимает слот только во время проверок, а на время, указанное в `poke_interval`, планировщик освобождает слот.

Работа с сенсорами

Операторы сенсоров являются производными от класса `airflow.sensors.base` и наследуют его атрибуты. Airflow поддерживает несколько сенсоров, которые описаны в таблице ниже, но некоторые провайдеры предоставляют также дополнительные сенсоры.

Стандартные сенсоры Airflow

Название	Описание
airflow.sensors.bash	Выполняет команду/скрипт bash, который можно использовать в качестве настраиваемого сенсора. Команда должна возвращать `0` при успешном выполнении, а в остальных случаях — любое другое значение
airflow.sensors.date_time	Приостанавливает выполнение DAG до наступления определенной даты и времени. Главным преимуществом этого сенсора является идемпотентность для `target_time`. Он обрабатывает некоторые случаи, для которых TimeSensor и TimeDeltaSensor не подходят
airflow.sensors.external_task	Ожидает завершения задачи в другом DAG
airflow.sensors.filesystem	Ожидает появления файла или директории в системе. Если указанный путь — непустая директория, то сенсор вернет значение `True`
airflow.sensors.python	Выполняет произвольный Python-код и ждет его завершения. Рекомендуется использовать декоратор `@task.sensor` вместо классического PythonSensor для выполнения Python-функций
airflow.sensors.time_delta	Находится в ожидании определенное время
airflow.sensors.time_sensor	Ожидает наступления указанного времени дня
airflow.sensors.weekday	Ожидает определенного дня недели. Например, если день выполнения задачи — `2018-12-22` (суббота), а в сенсор передается значение `FRIDAY` (пятница), задача будет ждать до следующей пятницы

Более подробное описание каждого сенсора доступно в документации Airflow.

Пример возможного использования сенсора в DAG:

```
from datetime import datetime, timedelta

from airflow import DAG
from airflow.sensors.time_delta import TimeDeltaSensor, TimeDeltaSensorAsync 1

dag = DAG(
    dag_id="sensor_example",
    schedule=None,
    start_date=datetime(2023, 6, 4),
)

time_sensor_example = TimeDeltaSensor( 2
    task_id='time_sensor_example',
    dag=dag,
    delta=timedelta(minutes=1), 3
    poke_interval=10, 4
    timeout=38, 5
    soft_fail=True, 6
)
```

1. Импорт операторов сенсоров. `TimeDeltaSensorAsync` — это асинхронная версия оператора `TimeDeltaSensor`.
2. Создание экземпляра сенсора. Если режим сенсора не указан, по умолчанию используется `poke`.
3. Параметр `delta` указывает, как долго ждать, прежде чем задача может быть отмечена как выполненная.
4. Параметр `poke_interval` определяет, как часто сенсор проверяет условие (в секундах).
5. Параметр `timeout` определяет, как долго сенсор будет работать.
6. Параметр `soft_fail` отмечает задачу сенсора как `SKIPPED`, если она не выполнена.

Чтобы избежать блокировки (deadlock) задач и других проблем с производительностью, рассмотрите следующие рекомендации по работе с сенсорами:

- Устанавливайте `timeout` для сенсора, который подойдет для вашей задачи. Значение по умолчанию для этого параметра может быть слишком большим в некоторых случаях.
- Используйте режим `reschedule` для долго работающих сенсоров, чтобы они не занимали рабочий слот постоянно, а режим `poke` — для задач с частой проверкой условий (т.е. менее 5 минут), чтобы избежать перегрузки планировщика.
- Убедитесь, что параметр `poke_interval` подходит для конкретной задачи. Значение по умолчанию предполагает очень частые проверки условий.

Создание кастомного сенсора

Чтобы создать кастомный сенсор, необходимо унаследовать класс `BaseSensorOperator`. В примере ниже демонстрируется создание нового сенсора, который проверяет наличие файла в директории.

1. Добавьте файл Python, который будет содержать код сенсора, в каталог *dags* домашней директории Airflow.

```
$ sudo vi /opt/airflow/dags/custom_sensor.py
```

2. Напишите код для сенсора:

```
import os
from airflow.sensors.base import BaseSensorOperator 1

class MyCustomFileSensor(BaseSensorOperator): 2
    def __init__(self, filepath, *args, **kwargs): 3
        self.filepath = filepath
        super().__init__(*args, **kwargs)

    def poke(self, context): 4
        print(f'checking if a file {self.filepath} exists...')
        return os.path.exists(self.filepath) 5
```

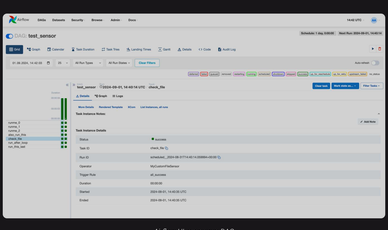
1. Импорт базового класса сенсора.
2. Наследование базового класса.
3. Добавление дополнительного аргумента `filepath` и наследование аргументов родительского класса.
4. Переопределение метода `poke`.
5. Python-логика для проверки существования файла.

3. Вызовите новый сенсор внутри любого DAG:

```
from custom_sensor import MyCustomFileSensor 1

check_file = MyCustomFileSensor(    2
    task_id='check_file',
    filepath='/opt/airflow/test_file.txt',
    poke_interval=10,
    timeout=60,
    dag=dag
)
```

1. Импорт класса кастомного сенсора.
2. Вызов функции сенсора с нужными параметрами.

4. Перейдите в веб-интерфейс Airflow, чтобы запустить DAG и убедиться, что функция кастомного сенсора работает корректно.



Airflow UI: страница DAG

На странице DAG видно, что задача `check_file` с классом `MyCustomFileSensor` успешно завершена.

Кастомизация расписания DAG

Елена Костенко

Содержание

Назад к содержанию

Обзор
Cron-расписания
Timedelta-расписания
Расписание событий
Кастомное расписание

Обзор

Airflow предоставляет возможность запускать конвейеры данных по расписанию, позволяя задавать временной интервал (data interval) и логическую дату (logical date) каждого запуска DAG.

Расписание DAG можно определить с помощью одного из следующих инструментов:

- встроенный объект `timedelta` модуля Python `datetime`;
- Linux Cron;
- кастомные временны классов `Timetable`.

Все параметры расписания, такие как интервал, дата, время и другие, определяются во внутреннем расписании DAG (timetable). DAG, запланированные с помощью выражения cron или объекта `timedelta`, также внутренне преобразуются в расписание.

В Airflow доступны расписания по умолчанию, которые обобщают выражения cron или `timedelta`, но для DAG с более специфичными требованиями к расписанию может потребоваться кастомный класс расписания DAG.

Примеры случаев, когда требуется кастомный класс `Timetable`:

- Конвейеры данных запускаются в разное время каждый день.
- Расписание DAG требует использования другого формата календаря.
- Расписание DAG с перекрывающимися интервалами данных. Например, запуск DAG должен охватывать период предыдущих запусков.
- DAG запускаются с неравными временными интервалами.

Cron-расписания

Выражения Cron в Airflow следуют стандартному синтаксису cron, который состоит из пяти полей, разделенных пробелами, где каждое поле может принимать значения, перечисленные в таблице ниже.

Cron-выражения задаются в формате `<1> <2> <3> <4> <5>`, где

- `<1>` – минуты в диапазоне: 0-59.
- `<2>` – час в диапазоне: 0-23.
- `<3>` – день месяца в диапазоне: 1-31.
- `<4>` – месяц в году. Диапазон от 1 до 12, где 1 – это январь.
- `<5>` – числовой индикатор дня недели. Каждый день представлен числом в диапазоне от 0 до 6, где 0 – это воскресенье.

Например, расписание DAG, представленное выражением cron `0 18 * * 5`, запускается каждую пятницу в 18:00.

Возможные значения	Пример
Единственное значение	5
Диапазон значений, определяемый с помощью тире	1-5
Список значений, разделенных запятыми	1,3,5
Каждое возможное значение (звездочка)	*

В Airflow предусмотрены два метода для обработки cron-расписаний: `CronTriggerTimetable` и `CronDataIntervalTimetable`. Оба расписания запускают DAG в одно и то же время, но временная метка для `run_id` у DAG будет разной.

Например, если DAG с расписанием, определенным как `0 8 * * *` (12:00 каждый день), включается 31 января в 15:00, `CronTriggerTimetable` запустит DAG 1 февраля в 12:00, но `CronDataIntervalTimetable` запустит DAG немедленно, поскольку запуск DAG для ежедневного временного интервала, начинающегося 31 января в 12:00, ещё не произошёл.

Пример DAG для `CronTriggerTimetable`, который запускает DAG в 1 час ночи в среду:

```
from datetime import timedelta
from airflow import DAG
from airflow.timetables.trigger import CronTriggerTimetable

def cron_timetable(*args, **kwargs):
    return CronTriggerTimetable(*args[*-1:])

dag = DAG(
    dag_id='my_dag',
    timetable=cron_timetable(cron_expression='0 1 * * 3', timezone='UTC'),
    start_date=days_ago(1),
    tags=['example'],
)

@dag(schedule=CronTriggerTimetable("0 1 * * 3", timezone="UTC"), ...)
def example_dag():
    pass
```

При использовании `CronDataIntervalTimetable` можно указать статический интервал данных с помощью аргумента `interval`. Он должен использовать объекты `datetime.timedelta` или `dateutil.relativedelta.relativedelta`.

Если статический интервал указан, временной интервал запуска DAG охватывает указанную продолжительность и заканчивается временем срабатывания (trigger time).

Например:

```
from datetime import timedelta
from airflow.timetables.trigger import CronTriggerTimetable

@dag(
    schedule=CronTriggerTimetable(
        "0 18 * * 5",
        timezone="UTC",
        interval=timedelta(days=4, hours=9),
    ),
    ...
)
def example_dag():
    pass
```

DAG из приведенного выше примера будет запускаться каждую пятницу в 18:00.

Timedelta-расписания

Расписание, использующее timedelta, позволяет запускать DAG с указанным временным интервалом. Это расписание опирается на значение временного интервала, предоставленное пользователем, и не обязательно согласуют даты выполнения с началом дня или часа.

Вы можете использовать timedelta-расписание, указав значение `datetime.timedelta` или `dateutil.relativedelta.relativedelta` в параметре `schedule`.

Например:

```
@dag(schedule=datetime.timedelta(minutes=30))
def example_dag():
    pass
```

Расписание событий

Расписание событий позволяет запускать DAG в указанные даты и время. Список дат должен быть небольшого размера, поскольку он должен загружаться каждый раз при обработке DAG.

Например:

```
from airflow import DAG
import pendulum
from airflow.timetables.events import EventsTimetable

@dag(
    schedule=EventsTimetable(
        event_dates=[
            pendulum.datetime(2024, 4, 5, 8, 27, tz="America/Vancouver"),
            pendulum.datetime(2024, 4, 17, 8, 27, tz="America/Vancouver"),
            pendulum.datetime(2024, 4, 22, 20, 50, tz="America/Vancouver"),
        ],
        description="Example dates",
        restrict_to_events=False,
    ),
    ...
)
def example_dag():
    pass
```

Кастомное расписание

Кастомные расписания позволяют задавать произвольный график запусков DAG в коде Python, когда расписаний на основе cron и `timedelta` недостаточно.

Чтобы создать кастомное расписание, необходимо:

- создать класс, который наследует класс расписания (`Timetable`);
- реализовать методы `next_dagrun_info()` и `infer_manual_data_interval()`, оба из которых возвращают объект `DataInterval`;
- зарегистрировать расписание как часть плагина Airflow.

Метод `next_dagrun_info()` возвращает интервал данных для стандартного расписания DAG и описывает логику для его расчета. Этот метод также содержит логику для расчета параметров `start_date`, `end_date` и `catchup`.

Метод `infer_manual_data_interval()` описывает, как определяется интервал работы DAG в случае ручного запуска.

В примере ниже показано, как написать DAG с кастомным расписанием.

Чтобы создать DAG с кастомным расписанием:

1. В директории рабочего домашнего каталога Airflow добавьте новый файл Python, который будет содержать кастомное расписание:

```
$ sudo vi /opt/airflow/plugins/custom_timetable.py
```

2. В файле custom_timetable.py определите структуру расписания:

```
from datetime import timedelta
from typing import Optional
from pendulum import Date, DateTime, Time, timezone
from airflow.plugins_manager import AirflowPlugin
from airflow.timetables.base import DagRunInfo, DataInterval, TimeRestriction, Timetable 1

UTC = timezone("UTC") 2

class CustomTimetable(Timetable): 3
    pass

class CustomTimetablePlugin(AirflowPlugin): 4
    name = "custom_timetable_plugin"
    timetables = [CustomTimetable]
```

1. Импорт зависимостей.
2. Объявление глобальных переменных.
3. Наследование базового класса.
4. Регистрация расписания как плагина Airflow.

3. В классе `CustomTimetable` напишите реализацию метода `next_dagrun_info()`:

```
class CustomTimetable(Timetable):

    def next_dagrun_info(
        self,
        *,
        last_automated_data_interval: Optional[DataInterval],
        restriction: TimeRestriction,
    ) -> Optional[DagRunInfo]:
        if last_automated_data_interval is not None: 1
            last_start = last_automated_data_interval.start
            delta = timedelta(days=1)
            if last_start.hour == 6:
                next_start = last_start.set(hour=16, minute=30).replace(tzinfo=UTC)
                next_end = (last_start+delta).replace(tzinfo=UTC)
            else:
                next_start = (last_start+delta).set(hour=6, minute=0).replace(tzinfo=UTC)
                next_end = (last_start+delta).replace(tzinfo=UTC)
        else: 2
            next_start = restriction.earliest
            if next_start is None:
                return None
            if not restriction.catchup: 3
                next_start = max(next_start, DateTime.combine(Date.today(), Time.min).replace(tzinfo=UTC))
            next_start = next_start.set(hour=6, minute=0).replace(tzinfo=UTC)
            next_end = next_start.set(hour=16, minute=30).replace(tzinfo=UTC)
        if restriction.latest is not None and next_start > restriction.latest: 4
            return None
        return DagRunInfo.interval(start=next_start, end=next_end)
```

1. Описание логики для случая, когда уже был предыдущий запуск DAG.
2. Описание логики первого запуска DAG.
3. Реализация поведения catchup.
4. Проверка того, что следующий запуск DAG не начнется вне расписания.

4. В классе `CustomTimetable` опишите логику планирования запусков в случае ручного запуска в методе `infer_manual_data_interval()`:

```
class CustomTimetable(Timetable):

    def infer_manual_data_interval(self, run_after: DateTime) -> DataInterval:
        delta = timedelta(days=1)
        if run_after >= run_after.set(hour=6, minute=0) and run_after <= run_after.set(hour=16, minute=30):
            start = (run_after-delta).set(hour=16, minute=30, second=0).replace(tzinfo=UTC)
            end = run_after.set(hour=6, minute=0, second=0).replace(tzinfo=UTC)
        elif run_after >= run_after.set(hour=16, minute=30) and run_after.hour <= 23:
            start = run_after.set(hour=6, minute=0, second=0).replace(tzinfo=UTC)
            end = run_after.set(hour=16, minute=30, second=0).replace(tzinfo=UTC)
        else:
            start = (run_after-delta).set(hour=6, minute=0).replace(tzinfo=UTC)
            end = (run_after-delta).set(hour=16, minute=30).replace(tzinfo=UTC)
        return DataInterval(start=start, end=end)
```

5. Используйте кастомное расписание в DAG:

```
from custom_timetable import CustomTimetable 1

@dag(
    dag_id="example_dag",
    start_date=datetime(2024, 3, 10),
    schedule=CustomTimetable(), 2
    default_args={
        "retries": 1,
        "retry_delay": duration(minutes=3),
    },
    catchup=True
)
```

1. Импорт кастомного расписания.
2. Вызов кастомного класса для использования его в качестве расписания запуска DAG.

Дополнительную информацию о запуске DAG можно получить в статье Создание простого DAG.

ВАЖНО!

Добавление или обновление плагина Airflow, включая пользовательские расписания, требует перезапуска Airflow Scheduler и веб-сервера.

Добавление кастомных операторов и хуков в Airflow

Елена Костюченко

Содержание

Назад к содержанию

Обзор

Создание кастомного оператора

Создание кастомного хука

Обзор

Airflow предоставляет стандартные операторы, которые позволяют, например, выполнять команды bash, вызывать произвольные функции Python, создавать соединение с внешним приложением или проверять выполнение определенного условия.

Airflow предоставляет "из коробки" большое количество операторов, но в некоторых случаях необходимо изменить функциональность существующего оператора или создать новый.

В этой статье описывается, как создавать пользовательские операторы и хуки. О создании кастомных сенсоров в Airflow можно прочитать в статье Использование сенсоров в Airflow.

При создании любого оператора Airflow необходимо:

1. Написать код для нового оператора в методе `__init__`.
2. Наследовать класс Airflow, который нужно переписать или изменить, переопределив методы в производном классе:
   - В конструкторе (constructor) определите параметры нового оператора.
   - В методе `execute` напишите код, который будет выполняться при вызове оператора.
3. Поместите файл с оператором в директорию, которая присутствует в `PYTHONPATH` Airflow. По умолчанию это директории *dags*, *_plugins* и *config*.

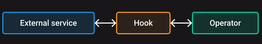
В примерах ниже показывается, как написать кастомные операторы и хуки, руководствуясь этими шагами.

Разделение логики между операторами и хуками

Оператор должен содержать бизнес-логику задачи и состояния в памяти. Если вам необходимо создать соединение с внешним сервисом (API, базой данных и т.д.), лучше написать его как хук.

Хук не должен содержать логику оператора, а только предоставлять методы, которые затем будут использоваться оператором для взаимодействия с внешним сервисом.

Разделив логику на два компонента, можно избежать дублирования кода и создать несколько операторов, которые используют один и тот же хук.



Взаимодействие операторов и хуков

Создание кастомного оператора

Ниже приведен пример создания кастомного оператора на основе класса `airflow.models.baseoperator.BaseOperator`:

1. В директории *dags* в домашнем каталоге Airflow добавьте файл Python, который будет содержать код оператора:

```
$ sudo vi /opt/airflow/dags/custom_operator.py
```

2. Добавьте код оператора:

```
from airflow.models.baseoperator import BaseOperator  1

class CustomOperator(BaseOperator):  2
    def __init__(self, name: str, **kwargs) -> None:  3
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):  4
        message = f"Hello, {self.name}!"
        print(message)
        return message
```

1. Импортируйте базовый класс оператора.
2. Наследуйте базовый класс в своем операторе.
3. Добавьте дополнительные аргументы и сохраните аргументы родительского класса.
4. Переопределите метод `execute`, задав желаемую функциональность.

3. Вызовите кастомный оператор в DAG:

```
from custom_operator import CustomOperator  1

print_message = CustomOperator(    2
    task_id='print_message',
    name='CustomOperator',
    poke_interval=10,
    timeout=60,
    dag=dag
)
```

1. Импортируйте класс кастомного оператора.
2. Вызовите функцию оператора.

Создание кастомного хука

Процесс создания кастомного хука отличается от процесса создания кастомного оператора тремя шагами:

- Хук должен быть унаследован от класса другого хука, например, класса `BaseHook`, или любого другого.
- В код хука можно включить метод `.get_conn()`, оборачивающий вызов метода `.get_connection()`, чтобы получить информацию о соединении Airflow.
- Должно существовать соединение (Airflow connection), которое будет использоваться хуком.

Ниже приведен пример того, как создать пользовательскую функцию хука, которая инициирует новое соединение с внешним сервисом:

1. В директории *dags* домашнего каталога Airflow добавьте файл Python, который будет содержать кастомный хук:

```
$ sudo vi /opt/airflow/dags/custom_hook.py
```

2. Добавьте код хука:

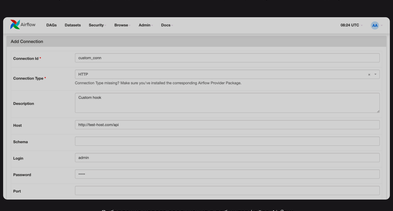
```
from airflow.hooks.base import BaseHook  1

class CustomHook(BaseHook):  2

    conn_name_attr = "custom_conn_id"  3
    default_conn_name = "custom_conn_default"  4
    conn_type = "http"  5
    hook_name = "CustomHook"  6

    def __init__(
        self, custom_conn_id: str = default_conn_name, *args, **kwargs
    ) -> None:  7
        super().__init__(*args, **kwargs)  8
        self.custom_conn_id = custom_conn_id  9
        self.get_conn()  10

    def get_conn(self):
        conn_id = getattr(self, self.conn_name_attr)
        conn = self.get_connection(conn_id)
        return conn
```

1. Импортируйте базовый класс хука.
2. Определите класс для наследования от базового класса.
3. Добавьте параметр, который сохраняет ID соединения.
4. Укажите ID соединения по умолчанию.
5. Укажите тип соединения.
6. Укажите имя хука.
7. Определите метод `.init()`.
8. Инициализируйте родительский хук.
9. Назначьте переменные класса.
10. При необходимости вызовите метод `.get_conn()` при инициализации.

3. Вызовите новый хук внутри DAG:

```
from custom_hook import CustomHook
```

4. Чтобы использовать пользовательский хук, необходимо создать соединение Airflow с параметрами, указанными в коде хука.



Добавление нового соединения в веб-интерфейсе Airflow

Более подробно о веб-интерфейсе Airflow читайте в статье Обзор UI в Airflow.

Динамическая генерация DAG в Airflow

Елена Костюченко

Содержание

Назад к содержанию

Обзор
Динамическая генерация DAG
Динамический мэппинг задач

Обзор

Традиционные DAG удобно использовать для конвейера данных с одним источником, например, когда нужно извлечь данные из файла, обработать их и загрузить результат.

В случае, когда необходимо запустить один и тот же пайплайн с разными параметрами или для нескольких источников данных, лучшим решением будет генерировать DAG динамически для каждого случая.

Существует два способа генерации DAG с вариативными параметрами:

- **Динамическая генерация DAG (dynamic DAGs generation)**
  В динамически генерируемых DAG структура DAG меняется в зависимости от определенных статических значений (файлов конфигурации, переменных среды, имен файлов и т.д.).
- **Динамический мэппинг задач (dynamic task mapping)**
  Динамический мэппинг задач позволяет генерировать параллельные задачи во время выполнения на основе вывода предыдущей задачи.

Для удобства примеры в этой статье написаны с использованием API TaskFlow, но вы также можете динамически генерировать DAG, используя традиционный синтаксис.

Динамическая генерация DAG

Процесс динамического создания DAG похож на процесс создания обычного DAG. Поскольку Airflow выполняет весь Python-код, размещенный в директории dags, вы можете запустить выполнение любого Python-файла, который генерирует объекты DAG.

Частота генерации новых DAG определяется параметром `min_file_process_interval`, значение которого по умолчанию составляет 30 секунд. Вы можете изменить значение этого параметра, отредактировав поле **cfg_properties_template** через ADCM.

Чтобы отредактировать свойство `min_file_process_interval`, через ADCM:

1. Перейдите в веб-интерфейс ADCM и выберите нужный кластер ADO.
2. Перейдите в **Services → Airflow2 → Primary configuration** и включите опцию **Show advanced**.
3. Откройте раздел **airflow.cfg** и выберите **cfg_properties_template**.
4. Установите желаемое значение для `min_file_process_interval`.
5. Сохраните конфигурацию, выбрав **Save → Create**, и перезапустите сервис, нажав **Actions → Restart**.

ВНИМАНИЕ

Если значение параметра `min_file_process_interval` слишком мало, DAG будут генерироваться слишком часто, что может создать большую нагрузку на планировщик.

Чтобы создать динамическая генерируемый DAG:

1. В директории dags домашнего каталога Airflow добавьте новый файл Python, который будет содержать код генератора DAG:

```
$ sudo vi /opt/airflow/dags/dag_generator.py
```

2. Добавьте код для генератора DAG:

```
from airflow import DAG
from airflow.decorators import task
from datetime import datetime    1

for file in ("dev_data.csv", "test_data.csv", "prod_data.csv"):    2
    dag_id = f"generated_dag_{file}"

    @dag(dag_id=dag_id, schedule="@daily",
    default_args=default_args, catchup=False,
    start_date=datetime(2025,1,1)) 3

    def create_dag(filename):

        @task 4
        def extract(filename):
            return filename

        @task
        def process(filename):
            return filename

        @task
        def load(filename):
            print(filename)
            return filename

        load(process(extract(filename))) 5

    create_dag(filename)    6
```

1. Импорт зависимостей.
2. Создание цикла для генерации DAG.
3. Создание функции DAG.
4. Добавление задач.
5. Определение зависимостей задач через вызов функций.
6. Вызов функции DAG.

При необходимости конфигурации DAG с помощью переменных рекомендуется использовать переменные окружения, а не переменные Airflow.

Использование переменных Airflow в коде верхнего уровня создает соединение с базой данных, где хранятся метаданные, что может замедлить синтаксический анализ и создать дополнительную нагрузку на базу данных.

Вы также можете импортировать переменные из одного из нескольких внешних источников, таких как файлы конфигурации.

Например, в директории *include/dag-configs/* домашнего каталога Airflow размещено два файла конфигурации JSON. Файлы имеют следующее содержимое:

- *dev_config.json*:

```
{
    "dag_id": "development_dag",
    "schedule": "@daily",
    "input": "dev_data.csv"
}
```

- *prod_config.json*:

```
{
    "dag_id": "production_dag",
    "schedule": "@daily",
    "input": "prod_data.csv"
}
```

Вы можете создать скрипт Python, который будет генерировать отдельные файлы DAG для каждого файла конфигурации, используя указанный файл DAG в качестве шаблона.

Чтобы генерировать DAG с использованием шаблона:

1. В домашней директории Airflow добавьте новый файл Python, который будет содержать шаблон DAG:

```
$ sudo vi /opt/airflow/include/templates/dag_template.py
```

2. Добавьте код, который будет использован генератором в качестве шаблона для DAG. Например, можно использовать код из предыдущего примера и определить параметры, которые будут меняться при генерации:

```
from airflow import DAG
from airflow.decorators import task
from datetime import datetime

@dag(dag_id = "dag_id_placeholder", schedule="schedule_placeholder",
catchup=False, start_date=datetime(2025,1,1)) 1

def create_dag(filename):

    @task
    def extract(filename):
        return filename

    @task
    def process(filename):
        return filename

    @task
    def load(filename):
        print(filename)
        return filename

    load(process(extract("input_placeholder"))) 2

create_dag(filename)
```

1. Добавление переменных, куда будут подставляться значения параметров `dag_id` и `schedule_interval`.
2. Добавление переменной для входного значения.

3. В домашней директории Airflow добавьте файл Python, который будет генерировать DAG с использованием шаблона:

```
$ sudo vi /opt/airflow/include/scripts/dag_generator.py
```

4. Напишите код генератора DAG:

```
import json
import os
import shutil
import fileinput

config_filepath = "include/dag-configs/" 1
template_filepath = "include/templates/dag_template.py" 2

for filename in os.listdir(config_filepath): 3
    if filename.endswith('.json'):
        config = json.load(open(f"include/data/{filename}"))
        generated_dag = f"dags/generated_dag_{config['dag_id']}.py"
        shutil.copyfile(template_filepath, generated_dag)
        for line in fileinput.input(generated_dag, inplace=True):
            line = line.replace("dag_id_placeholder", config['dag_id'])
            line = line.replace("schedule_placeholder", config['schedule'])
            line = line.replace("input_placeholder", config['input'])
            print(line, end="")
```

1. Путь к файлам конфигурации.
2. Путь к файлу шаблона.
3. Описание логики замены переменных на нужные значения из файлов конфигурации.

При выполнении скрипта директория dag-configs проверяется на наличие доступных файлов конфигурации JSON и генерируется файл DAG для каждого файла конфигурации на основе указанного шаблона. Переменные в шаблоне сопоставляются с полями из файлов конфигурации и заменяются на нужные значения.

После генерации файлов Python Airflow загрузит их как обычные DAG.

Динамический мэппинг задач

В примерах выше демонстрируется, как генерировать DAG с разными параметрами. При этом количество задач в сгенерированных DAG остается прежним и не может меняться.

Чтобы создавать задачи в DAG на основе меняющихся выходных данных другой задачи, можно использовать динамический мэппинг задач.

Этот подход использует фреймворк Mapfeature и работает с помощью метода `expand()`. Этот метод создает новые задачи путем копирования той, которая его вызывает, с использованием предоставленных аргументов и вывода предыдущей задачи.

Метод `expand()` используется с операторами и принимает список или словарь значений в качестве входных данных. Эти значения также могут быть выходными данными другой задачи в формате `XComArg`.

Пример динамического мэппинга задач:

```
from airflow.decorators import dag, task
from datetime import datetime

@dag(dag_id = "dynamic_task_dag", schedule="@daily", catchup=False, start_date=datetime(2025,1,1))
def create_tasks():

    @task
    def download_files(file: str):    1
        print(file)

    files = download_files.expand(file=["file_a", "file_b", "file_c"])    2

    download_files(file)

create_tasks()
```

1. Создание функции задачи. В данном случае — функция, которая выводит содержимое файла.
2. Вызов метода `expand()` и передача ему списка файлов в качестве аргумента. DAG сгенерирует три задачи для каждого файла.

Пример динамического мэппинга задач, когда вывод предыдущей задачи не определен:

```
from airflow.decorators import dag, task
from datetime import datetime
import random

@dag(dag_id = "dynamic_task_dag", schedule="@daily",
catchup=False, start_date=datetime(2025,1,1))

def create_tasks():

    @task
    def get_files():
        return ["file_{i}" for i in range(random.randint(1,3))]    1

    @task
    def download_files(file: str):
        print(file)

    files = download_files.expand(file=get_files())    2

create_tasks()
```

1. Создание функции со случайным выводом.
2. Вызов метода `expand()` и передача ему вывода функции в качестве аргумента. DAG сгенерирует задачу для каждого файла.

Использование шаблонов Jinja в Airflow

Елена Костюченко

Содержание

Назад к содержанию

- Обзор
- Jinja в Airflow
- Ограничения использования шаблонов
  - Путь поиска шаблонов
  - PythonOperator
- Макросы и фильтры
- Создание DAG с использованием шаблонов Jinja
  - Шаг 1. Подготовка файла шаблона
  - Шаг 2. Создание Python-кода для DAG
  - Шаг 3. Подготовка директории для отчетов
  - Шаг 4. Запуск DAG

Обзор

Airflow поддерживает использование шаблонов, что позволяет обновлять код DAG динамически во время выполнения с использованием переменных, макросов и управляющих конструкций. Эта функция позволяет в зависимости от контекста выполнения менять в DAG параметры, которые используют информацию о времени выполнения, экземпляре задачи и пользовательскую логику.

В Airflow используется фреймворк шаблонов Jinja. Это шаблонизатор на Python, который позволяет встраивать динамические выражения и управляющие конструкции в код.

В этой статье описано использование шаблонов Jinja внутри DAG и операторов в Airflow: рассмотрены основные шаблоны, шаблонизируемые поля, макросы и примеры использования.

Jinja в Airflow

Используя Jinja, Airflow рендерит определенные поля, называемые шаблонизируемыми (templated fields), во время выполнения. Эти поля определяются на уровне операторов. Типовые операторы, такие как `BashOperator`, `PythonOperator`, `EmailOperator` и `SqlOperator`, имеют шаблонизируемые поля вроде `bash_command`, `sql` и `subject`.

Все доступные опции описаны в справочнике шаблонов Airflow.

Пример использования шаблонов Jinja:

```
from airflow.operators.bash import BashOperator
from airflow import DAG
from datetime import datetime 1

with DAG("templating_example", start_date=datetime(2023, 1, 1), schedule_interval="@daily", catchup=False) as dag: 2
    t1 = BashOperator(
        task_id='print_date',
        bash_command='echo "Execution date is {{ ds }}"', 3
    )
```

1. Импорт зависимостей.
2. Инициализация DAG.
3. Добавление шаблона в строку вывода. В этом примере `{{ ds }}` будет заменен на дату выполнения во время запуска (например, `2025-04-22`).

Ограничения использования шаблонов

Для использования шаблонов в параметрах операторов используется синтаксис `{{ … }}` и `{% … %}`.

В Airflow поддерживаются следующие типы шаблонов:

- Переменные. Например: `{{ ds }}`, `{{ task_instance }}` и другие.
- Управляющие конструкции. Например: `{% if … %}`, `{% for … %}`.
- Пользовательские выражения. Например: `{{ macros.ds_add(ds, 7) }}`.

Чтобы узнать, какие поля являются шаблонизируемыми, обратитесь к документации нужного оператора. Также можно определить собственные шаблонизируемые поля, дополнив код оператора.

Путь поиска шаблонов

Airflow может загружать внешние шаблоны из разных директорий. Это позволяет разделять шаблоны по типам, например: SQL-запросы или шаблоны отчетов.

Пример:

```
dag = DAG(
    dag_id='template_path_example',
    start_date=datetime(2023, 1, 1),
    schedule_interval='@daily',
    template_searchpath=['/opt/airflow/templates/sql']
)
```

где `template_searchpath` – путь к шаблонам SQL.

PythonOperator

Jinja можно использовать в Python-функциях, подставляя значения шаблона до выполнения задачи. Шаблонные строки можно передать через `op_kwargs`.

Например:

```
from airflow.operators.python import PythonOperator

def greet_user(execution_date, **kwargs):
    print(f"Hello! Today is {execution_date}")

PythonOperator(
    task_id='greet',
    python_callable=greet_user,
    op_kwargs={"execution_date": "{{ ds }}"},
    dag=dag,
)
```

Airflow рендерит `"{{ ds }}"` в фактическую дату до вызова `greet_user`.

Макросы и фильтры

Airflow включает встроенные макросы, такие как `macros.ds_add`, `macros.datetime`, фильтры вроде `| ds_format`, `| replace` и другие макросы.

Например:

```
BashOperator(
    task_id='seven_days_later',
    bash_command='echo "In 7 days: {{ macros.ds_add(ds, 7) }}"',
)
```

Также можно определить пользовательские макросы внутри DAG или загрузить их через плагины.

Создание DAG с использованием шаблонов Jinja

Ниже приведен пример создания DAG, который генерирует ежедневный отчет в формате Markdown с использованием шаблонов Jinja.

Шаг 1. Подготовка файла шаблона

Создайте файл шаблона Jinja, который будет использоваться для генерации отчетов.

1. Создайте директорию для шаблонов в среде Airflow:

```
$ sudo mkdir -p /opt/airflow/templates
```

2. В директории *templates* создайте файл с именем *report_template.md* и следующим содержимым:

```
# Daily report

**Date:** {{ ds }}
**Task Status:** {{ task_instance.state }}

## Summary
- DAG ID: {{ dag.dag_id }}
- Task ID: {{ task.task_id }}
```

Значения в фигурных скобках – это плейсхолдеры, которые будут заполнены динамически во время выполнения.

Шаг 2. Создание Python-кода для DAG

Создайте файл *dynamic_report_dag.py* в директории DAG и добавьте следующий код:

```
from airflow import DAG 1
from airflow.operators.python import PythonOperator
from datetime import datetime
from jinja2 import Environment, FileSystemLoader

def render_report(ds, **context):
    env = Environment(loader=FileSystemLoader('/opt/airflow/templates'))
    template = env.get_template('report_template.md')
    report = template.render(
        ds=ds,
        dag=context['dag'],
        task=context['task'],
        task_instance=context['ti']
    )

    output_path = f"/opt/airflow/reports/report_{ds}.md" 3
    with open(output_path, "w") as f:
        f.write(report)

with DAG( 4
    dag_id='dynamic_report_dag',
    start_date=datetime(2023, 1, 1),
    schedule_interval='@daily',
    catchup=False,
    template_searchpath=['/opt/airflow/templates'],
) as dag:

    generate_report = PythonOperator(
        task_id='generate_report',
        python_callable=render_report,
        provide_context=True,
    )
```

1. Импорт зависимостей.
2. Определение функции для рендеринга шаблона Jinja.
3. Сохранение отчета.
4. Инициализация DAG. Дополнительную информацию о создании DAG можно получить в статье Создание простого DAG.

Шаг 3. Подготовка директории для отчетов

1. Создайте директорию, в которую будут сохраняться сгенерированные отчеты:

```
$ sudo mkdir -p /opt/airflow/reports
```

2. Настройте необходимые права доступа:

```
$ sudo chmod -R 755 /opt/airflow/reports/
$ sudo chown -R airflow:airflow /opt/airflow/reports
```

Шаг 4. Запуск DAG

1. Перейдите в веб-интерфейс Airflow.
2. Найдите и снимите с паузы DAG `dynamic_report_dag`.
3. Запустите DAG.

После его выполнения в директории *reports* появится файл с текущей датой в названии и со следующим содержимым:

```
# Daily report

**Date:** 2025-04-28
**Task Status:** running

## Summary
- DAG ID: dynamic_report_dag
- Task ID: generate_report
```

Файл содержит дату выполнения, статус задачи, DAG ID и ID задачи, соответствующие контексту выполнения.

Airflow CLI

Стандартный интерфейс

Содержание

Назад к содержанию

Обзор
Общие команды
Компоненты Celery
Просмотр конфигурации
Управление соединениями
Управление DAG
Управление базой данных
Управление заданиями
Управление KubernetesExecutor
Управление пулами
Управление провайдерами
Управление ролями
Управление задачами
Управление пользователями
Управление переменными

Обзор

В этот раздел включена справочная документация по интерфейсу командной строки Airflow.

Используемая команда:

```
$ airflow COMMAND [GENERIC_OPTIONS] SUBCOMMAND [COMMAND_OPTIONS]
```

Общие команды

Команды общего назначения для управления кластером Airflow.

Команда	Описание
cheat-sheet	Выводит подсказку по командам
dag-processor	Запускает отдельный экземпляр процессора DAG
info	Отображает информацию об Airflow и окружении
kerberos	Запускает обновление тикетов Kerberos
plugins	Выводит информацию о загруженных плагинах
rotate-fernet-key	Выполняет ротацию зашифрованных учетных данных соединений и переменных
scheduler	Запускает экземпляр планировщика
standalone	Запускает полнофункциональную копию Airflow
sync-perm	Обновляет разрешения для существующих ролей и групп DAG (если необходимо)
triggerer	Запускает экземпляр триггера
version	Показывает версию Airflow
webserver	Показывает экземпляр веб-сервера Airflow

Компоненты Celery

Команды для управления Airflow Celery.

Команда	Описание
flower	Запускает Celery Flower
stop	Асинхронно останавливает Celery worker
worker	Запускает worker-кластер Celery

Просмотр конфигурации

Команды для просмотра конфигурации Airflow.

Команда	Описание
get-value	Выводит значение конфигурации
list	Выводит список параметров конфигурации

Управление соединениями

Команды для управления соединениями Airflow.

Команда	Описание
add	Добавляет соединение
webserver	Запускает экземпляр веб-сервера Airflow
delete	Удаляет соединение
export	Экспортирует соединения
get	Отображает соединение
import	Импортирует соединения из файла
list	Выводит список соединений
test	Тестирует соединение

Управление DAG

Команды для управления DAG.

Команда	Описание
backfill	Запускает подзадачи DAG для указанного диапазона дат
delete	Удаляет все записи БД, связанные с указанным DAG
details	Отображает информацию о DAG
list	Выводит список всех DAG
list-import-errors	Выводит список всех DAG, имеющих ошибки импорта
list-jobs	Выводит список задач
list-runs	Выводит список запусков DAG
next-execution	Отображает даты выполнения следующих DAG
pause	Приостанавливает DAG
report	Показывает отчет о загрузке DAG
reserialize	Повторно сериализует все DAG, анализируя файлы DAG
show	Отображает задачи DAG с их зависимостями
show-dependencies	Отображает DAG с их зависимостями
state	Отображает состояние запуска DAG
test	Выполняет тестовый запуск DAG
trigger	Запускает DAG
unpause	Возобновляет приостановленный DAG

Управление базой данных

Команды для управления базой данных Airflow.

Команда	Описание
check	Проверяет, доступна ли база данных
check-migrations	Проверяет, завершены ли миграции
clean	Удаляет старые записи в таблицах metastore
downgrade	Понижает версию базы метаданных
drop-archived	Удаляет архивные таблицы, созданные с помощью команды `db clean`
export-archived	Экспортирует данные из архивных таблиц
init	Инициализирует базу метаданных
reset	Удаляет и пересоздаёт базу метаданных
shell	Запускает shell для доступа к базе данных
upgrade	Обновляет базу метаданных до последней версии

Управление заданиями

Команды для управления заданиями Airflow.

Команда	Описание
check	Проверяет статус задания

Управление KubernetesExecutor

Команды для управления KubernetesExecutor.

Команда	Описание
cleanup-pods	Удаляет модуль Kubernetes (созданные KubernetesExecutor/KubernetesPodOperator) в состояниях `Succeeded`, `Failed`, `Scheduled`, `Unknown`
generate-dag-yaml	Генерирует файлы YAML, для всех задач в DAG. Полезно для отладки задач без запуска в кластере

Управление пулами

Команды для управления пулами Airflow.

Команда	Описание
delete	Удаляет пул
export	Экспортирует все пулы
get	Отображает размер пула
import	Импортирует пулы
list	Отображает список пулов
set	Настраивает пул

Управление провайдерами

Команды для получения информации о провайдерах.

Команда	Описание
auth	Отображает информацию о бэкендах аутентификации API
behaviours	Отображает информацию о зарегистрированных типах подключений с настоящим поведением
get	Отображает подробную информацию о провайдере
hooks	Показывает список зарегистрированных хуков провайдера
links	Показывает список дополнительных ссылок, зарегистрированных провайдерами
list	Показывает список установленных провайдеров
logging	Отображает информацию об обработчиках логирования
secrets	Отображает информацию о бэкендах конфиденциальной информации
widgets	Отображает информацию о зарегистрированных подключениях виджетов

Управление ролями

Команды для управления ролями Airflow.

Команда	Описание
add-perms	Добавляет разрешения для роли
create	Создает роль
del-perms	Удаляет разрешения для роли
delete	Удаляет роль
export	Экспортирует роли (без разрешений) из базы данных в файл JSON
import	Импортирует роли (без разрешений) из файла JSON в базу данных
list	Выводит список ролей

Управление задачами

Команды для управления задачами Airflow.

Команда	Описание
clear	Очищает набор экземпляров задач
failed-deps	Возвращает известные планировщику зависимости задачи, которые вызвали ошибку
list	Отображает задачи в DAG
render	Отображает шаблоны задачи
run	Запускает один экземпляр задачи
state	Отображает статус задачи
states-for-dag-run	Отображает состояние всех задач в DAG
test	Выполняет тестовый запуск задачи

Управление пользователями

Команды для управления пользователями Airflow.

Команда	Описание
add-role	Добавляет роль пользователю
create	Создает пользователя
delete	Удаляет пользователя
export	Экспортирует всех пользователей
import	Импортирует пользователей
list	Отображает список пользователей
remove-role	Удаляет роль у пользователя

Управление переменными

Команды для управления переменными Airflow.

Команда	Описание
delete	Удаляет переменную
export	Экспортирует все переменные
get	Выводит информацию о переменной
import	Импортирует переменные
list	Выводит список переменных
set	Редактирует переменную

cheat-sheet

Елена Костюченко

Содержание

Назад к содержанию

Описание
Использование
└ Аргументы
Примеры

Описание

Показывает подсказку с командами Airflow CLI и их описаниями, сгруппированными по типам команд.

Использование

```
$ airflow cheat-sheet
                    [-h]
                    [-v]
```

Аргументы

Параметр	Описание
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить подсказку, выполните команду:

```
$ airflow cheat-sheet
```

Пример вывода

```
Miscellaneous commands
airflow cheat-sheet                     | Display cheat sheet
airflow dag-processor                   | Start a standalone Dag Processor instance
airflow info                            | Show information about current Airflow and environment
airflow kerberos                        | Start a kerberos ticket renewer
airflow plugins                         | Dump information about loaded plugins
airflow rotate-fernet-key               | Rotate encrypted connection credentials and variables
airflow scheduler                       | Start a scheduler instance
airflow standalone                      | Run an all-in-one copy of Airflow
airflow sync-perm                       | Update permissions for existing roles and optionally DAGs
airflow triggerer                       | Start a triggerer instance
airflow version                         | Show the version
airflow webserver                       | Start a Airflow webserver instance

Celery components
airflow celery flower                   | Start a Celery Flower
airflow celery stop                     | Stop the Celery worker gracefully
airflow celery worker                   | Start a Celery worker node

View configuration
airflow config get-value                | Print the value of the configuration
airflow config list                     | List options for the configuration

Manage connections
airflow connections add                 | Add a connection
airflow connections delete              | Delete a connection
airflow connections export              | Export all connections
airflow connections get                 | Get a connection
airflow connections import              | Import connections from a file
airflow connections list                | List connections
airflow connections test                | Test a connection

Manage DAGs
airflow dags backfill                   | Run subsections of a DAG for a specified date range
airflow dags delete                     | Delete all DB records related to the specified DAG
airflow dags details                    | Get DAG details given a DAG id
airflow dags list                       | List all the DAGs
airflow dags list-import-errors         | List all the DAGs that have import errors
airflow dags list-jobs                  | List the jobs
airflow dags list-runs                  | List DAG runs given a DAG id
airflow dags next-execution             | Get the next execution datetimes of a DAG
airflow dags pause                      | Pause a DAG
airflow dags report                     | Show DagBag loading report
airflow dags reserialize                | Reserialize all DAGs by parsing the DagBag files
airflow dags show                       | Displays DAG's tasks with their dependencies
airflow dags show-dependencies          | Displays DAGs with their dependencies
airflow dags state                      | Get the status of a dag run
airflow dags test                       | Execute one single DagRun
airflow dags trigger                    | Trigger a DAG run
airflow dags unpause                    | Resume a paused DAG

Database operations
airflow db check                        | Check if the database can be reached
airflow db check-migrations             | Check if migration have finished
airflow db clean                        | Purge old records in metastore tables
airflow db downgrade                    | Downgrade the schema of the metadata database.
airflow db drop-archived                | Drop archived tables created through the db clean command
airflow db export-archived              | Export archived data from the archive tables
airflow db init                         | Initialize the metadata database
airflow db reset                        | Burn down and rebuild the metadata database
airflow db shell                        | Runs a shell to access the database
airflow db upgrade                      | Upgrade the metadata database to latest version

Manage jobs
airflow jobs check                      | Checks if job(s) are still alive

Tools to help run the KubernetesExecutor
airflow kubernetes cleanup-pods         | Clean up Kubernetes pods (created by KubernetesExecutor/KubernetesPodOperator) in
                                        | evicted/failed/succeeded/pending states
airflow kubernetes generate-dag-yaml    | Generate YAML files for all tasks in DAG. Useful for debugging tasks without
launching into a cluster

Manage pools
airflow pools delete                    | Delete pool
airflow pools export                    | Export all pools
airflow pools get                       | Get pool size
airflow pools import                    | Import pools
airflow pools list                      | List pools
airflow pools set                       | Configure pool

Display providers
airflow providers auth                  | Get information about API auth backends provided
airflow providers behaviours            | Get information about registered connection types with custom behaviours
airflow providers get                   | Get detailed information about a provider
airflow providers hooks                 | List registered provider hooks
airflow providers links                 | List extra links registered by the providers
airflow providers list                  | List installed providers
airflow providers logging               | Get information about task logging handlers provided
airflow providers secrets               | Get information about secrets backends provided
airflow providers widgets               | Get information about registered connection form widgets

Manage roles
airflow roles add-perms                 | Add roles permissions
airflow roles create                    | Create role
airflow roles del-perms                 | Delete roles permissions
airflow roles delete                    | Delete role
airflow roles export                    | Export roles (without permissions) from db to JSON file
airflow roles import                    | Import roles (without permissions) from JSON file to db
airflow roles list                      | List roles

Manage tasks
airflow tasks clear                     | Clear a set of task instance, as if they never ran
airflow tasks failed-deps               | Returns the unmet dependencies for a task instance
airflow tasks list                      | List the tasks within a DAG
airflow tasks render                    | Render a task instance's template(s)
airflow tasks run                       | Run a single task instance
airflow tasks state                     | Get the status of a task instance
airflow tasks states-for-dag-run        | Get the status of all task instances in a dag run
airflow tasks test                      | Test a task instance

Manage users
airflow users add-role                  | Add role to a user
airflow users create                    | Create a user
airflow users delete                    | Delete a user
airflow users export                    | Export all users
airflow users import                    | Import users
airflow users list                      | List users
airflow users remove-role               | Remove role from a user

Manage variables
airflow variables delete                | Delete variable
airflow variables export                | Export all variables
airflow variables get                   | Get variable
airflow variables import                | Import variables
airflow variables list                  | List variables
airflow variables set                   | Set variable
```

dag-processor

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Запускает автономный экземпляр процессора DAG. Вы можете выполнить эту команду с помощью действия ***Add/Remove components*** в ADCM.

Использование

```
$ airflow dag-processor [-h] [-D] [-p] [-l LOG_FILE] [-n NUM_RUNS] [--pid [PID]]
                        [--stderr STDERR] [--stdout STDOUT] [-S SUBDIR] [-v]
```

Аргументы

Параметр	Описание
-D, --daemon	Запустить выполнение команды в качестве процесса-демона
-p, --do-pickle	Попытаться законсервировать объект DAG для отправки рабочим процессам вместо того, чтобы позволить рабочим процессам выполнить свою версию кода
-l, --log-file	Расположение логов
-n, --num-runs	Количество запусков, которые необходимо выполнить перед выходом. Значение по умолчанию — `-1` (не делать дополнительных попыток запуска)
--pid	Расположение PID-файла
--stderr	Перенаправить stderr в указанный файл
--stdout	Перенаправить stdout в указанный файл
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

info

Елена Костюченко

Содержание

Назад к содержанию

Описание
Использование
└ Аргументы
Примеры

Описание

Предоставляет информацию о системе.

Использование

```
$ airflow info [-h] [-v]
```

Аргументы

Параметр	Описание
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы получить информацию о системе, выполните:

```
$ airflow info
```

Пример вывода

```
Apache Airflow
version                 | 2.6.3
executor                | CeleryExecutor
task_logging_handler    | airflow.utils.log.file_task_handler.FileTaskHandler
sql_alchemy_conn        | postgresql+psycopg2://airflow:admin@host:5432/airflow
dags_folder             | /opt/airflow/dags
plugins_folder          | /opt/airflow/plugins
base_log_folder         | /var/log/airflow
remote_base_log_folder |


System info
OS              | Linux
architecture    | x86_64
uname           | uname_result(system='Linux', node='host', release='3.10.0-1160.119.1.el7.x86_64', version='#1 SMP Tue
                | Jun 4 14:43:51 UTC 2024', machine='x86_64')
locale          | ('en_US', 'UTF-8')
python_version  | 3.10.4 (main, Feb 24 2025, 09:32:37) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]
python_location | /opt/airflow/bin/python3.10


Tools info
git             | NOT AVAILABLE
ssh             | OpenSSH_7.4p1, OpenSSL 1.0.2k-fips  26 Jan 2017
kubectl         | NOT AVAILABLE
gcloud          | NOT AVAILABLE
cloud_sql_proxy | NOT AVAILABLE
mysql           | NOT AVAILABLE
sqlite3         | NOT AVAILABLE
psql            | NOT AVAILABLE


Paths info
airflow_home    | /opt/airflow
system_path     | /sbin:/bin:/usr/sbin:/usr/bin
python_path     | /opt/airflow/bin:/opt/python3.10/lib/python310.zip:/opt/python3.10/lib/python3.10:/opt/python3.
10/lib/python3.10/lib-dynload:/opt/
                | airflow/lib/python3.10/site-packages:/opt/airflow/dags:/opt/airflow/config:/opt/airflow/plugins
airflow_on_path | True


Providers info
apache-airflow-providers-alibaba          | 2.6.0
apache-airflow-providers-amazon           | 8.12.0
apache-airflow-providers-apache-beam      | 5.3.0
apache-airflow-providers-apache-cassandra | 3.3.0
apache-airflow-providers-apache-drill     | 2.5.0
apache-airflow-providers-apache-druid     | 3.6.0
apache-airflow-providers-apache-flink     | 1.2.0
apache-airflow-providers-apache-hdfs      | 4.2.0
apache-airflow-providers-apache-hive      | 6.2.0
apache-airflow-providers-apache-impala    | 1.2.1
apache-airflow-providers-apache-kafka     | 1.2.0
apache-airflow-providers-apache-kylin     | 3.3.0
apache-airflow-providers-apache-livy      | 3.6.0
apache-airflow-providers-apache-spark     | 4.4.0
apache-airflow-providers-apache-sqoop     | 4.1.0
apache-airflow-providers-apprise          | 1.1.0
apache-airflow-providers-arangodb         | 2.3.0
apache-airflow-providers-asana            | 2.3.0
apache-airflow-providers-atlassian-jira   | 2.3.0
apache-airflow-providers-celery           | 3.4.1
apache-airflow-providers-cloudant         | 3.3.0
apache-airflow-providers-cncf-kubernetes  | 7.10.0
apache-airflow-providers-common-sql       | 1.8.1
apache-airflow-providers-dbt-cloud        | 3.4.1
apache-airflow-providers-dingding         | 3.3.0
apache-airflow-providers-discord          | 3.4.1
apache-airflow-providers-docker           | 3.8.2
apache-airflow-providers-elasticsearch    | 5.2.0
apache-airflow-providers-exasol           | 4.3.0
apache-airflow-providers-facebook         | 3.3.0
apache-airflow-providers-ftp              | 3.4.2
apache-airflow-providers-github           | 2.4.0
apache-airflow-providers-google           | 10.12.0
apache-airflow-providers-grpc             | 3.3.0
apache-airflow-providers-hashicorp        | 3.4.1
apache-airflow-providers-http             | 4.4.2
apache-airflow-providers-imap             | 3.2.2
apache-airflow-providers-influxdb         | 2.3.0
apache-airflow-providers-jdbc             | 4.1.0
apache-airflow-providers-jenkins          | 3.4.0
apache-airflow-providers-microsoft-azure  | 8.3.0
apache-airflow-providers-microsoft-mssql  | 3.5.0
apache-airflow-providers-microsoft-psrp   | 2.4.0
apache-airflow-providers-microsoft-winrm  | 3.3.0
apache-airflow-providers-mongo            | 3.4.0
apache-airflow-providers-mysql            | 5.4.0
apache-airflow-providers-odbc             | 4.2.0
apache-airflow-providers-oracle           | 3.8.0
apache-airflow-providers-postgres         | 5.8.0
apache-airflow-providers-redis            | 3.4.1
apache-airflow-providers-salesforce       | 5.5.1
apache-airflow-providers-samba            | 4.3.0
apache-airflow-providers-segment          | 3.3.0
apache-airflow-providers-sendgrid         | 3.3.0
apache-airflow-providers-sftp             | 4.7.0
apache-airflow-providers-singularity      | 3.3.0
apache-airflow-providers-slack            | 8.4.0
apache-airflow-providers-smtp             | 1.4.1
apache-airflow-providers-sqlite           | 3.4.2
apache-airflow-providers-ssh              | 3.8.1
apache-airflow-providers-tableau          | 4.3.0
apache-airflow-providers-tabular          | 1.3.0
apache-airflow-providers-telegram         | 4.2.0
apache-airflow-providers-vertica          | 3.6.0
apache-airflow-providers-yandex           | 3.6.0
```

kerberos

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Запускает обновление тикетов Kerberos.

Использование

```
$ airflow kerberos [-h] [-D] [-k [KEYTAB]] [-l LOG_FILE] [-o] [--pid [PID]]
                 [--stderr STDERR] [--stdout STDOUT] [-v]
                 [principal]
```

Аргументы

Параметр	Описание
principal	Принципал Kerberos
-D, --daemon	Запустить выполнение команды в качестве процесса-демона
-k, --keytab	Используемый keytab. Значение по умолчанию: `airflow.keytab`
-l, --log-file	Расположение логов
-o, --one-time	Запустить Airflow Kerberos только один раз
--pid	Расположение PID-файла
--stderr	Перенаправить stderr в указанный файл
--stdout	Перенаправить stdout в указанный файл
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для запуска обновления тикетов Kerberos:

```
$ airflow kerberos --keytab custom.keytab airflow
```

plugins

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает информацию о загруженных плагинах или сохраняет ее в файл.

Использование

```
$ airflow plugins [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить информацию о плагинах, выполните:

```
$ airflow plugins
```

The output:

```
name | source                                                                  | macros
=====+=========================================================================+=================================================
===================
hive | apache-airflow-providers-apache-hive==6.2.0: EntryPoint(name='hive',  | airflow.providers.apache.hive.macros.hive.
max_partition,airflow.provi
     | value='airflow.providers.apache.hive.plugins.hive:HivePlugin',        | ders.apache.hive.macros.hive.closest_ds_partition
     | group='airflow.plugins')                                              |
```

rotate-fernet-key

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

Описание

Выполняет ротацию всех зашифрованных учетных данных и переменных соединения. Вы можете выполнить эту команду с помощью действия ***Rotate fernet key*** в ADCM.

Использование

```
$ airflow rotate-fernet-key [-h]
```

scheduler

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Запускает экземпляр планировщика. Вы можете выполнить эту команду с помощью действия ***Add/Remove components*** в ADCM.

Использование

```
$ airflow scheduler [-h] [-D] [-p] [-l LOG_FILE] [-n NUM_RUNS] [--pid [PID]]
                    [-s] [--stderr STDERR] [--stdout STDOUT] [-S SUBDIR] [-v]
```

Аргументы

Параметр	Описание
-D, --daemon	Запустить выполнение команды в качестве процесса-демона
-p, --do-pickle	Попытаться консервировать объект DAG для отправки worker вместо того, чтобы выполнять версию кода worker
-l, --log-file	Расположение логов
-n, --num-runs	Количество запусков, которые необходимо выполнить перед выходом
--pid	Расположение PID-файла
-s, --skip-serve-logs	Не запускать процесс обслуживания логов вместе с рабочими процессами
--stderr	Перенаправить stderr в указанный файл
--stdout	Перенаправить stdout в указанный файл
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

standalone

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

Описание

Запускает полнофункциональную копию Airflow.

Использование

```
$ airflow standalone [-h]
```

sync-perm

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Обновляет разрешения для существующих ролей и, при необходимости, групп DAG. Вы можете выполнить эту команду с помощью действия ***Sync permission*** в ADCM.

Использование

```
$ airflow sync-perm [-h] [--include-dags] [-v]
```

Аргументы

Параметр	Описание
--include-dags	Синхронизировать разрешения DAG
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

triggerer

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Запускает экземпляр триггера. Вы можете выполнить эту команду с помощью действия ***Add/Remove components*** в ADCM.

Использование

```
$ airflow triggerer [-h] [--capacity CAPACITY] [-D] [-l LOG_FILE] [--pid [PID]]
                    [-s] [--stderr STDERR] [--stdout STDOUT] [-v]
```

Аргументы

Параметр	Описание
--capacity	Максимальное количество триггеров, которые могут быть запущены одновременно
-D, --daemon	Запустить выполнение команды в качестве процесса-демона
-l, --log-file	Расположение логов
--pid	Расположение PID-файла
-s, --skip-serve-logs	Не запускать процесс обслуживания логов вместе с рабочими процессами
--stderr	Перенаправить stderr в указанный файл
--stdout	Перенаправить stdout в указанный файл
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

version

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

Описание

Показывает версию Airflow.

Использование

```
$ airflow version [-h]
```

webserver

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Запускает экземпляр веб-сервера Airflow. Вы можете выполнить эту команду с помощью действия ***Add/Remove components*** в ADCM.

Использование

```
$ airflow webserver [-h] [-A ACCESS_LOGFILE] [-L ACCESS_LOGFORMAT] [-D] [-d]
                  [-E ERROR_LOGFILE] [-H HOSTNAME] [-l LOG_FILE] [--pid [PID]]
                  [-p PORT] [--ssl-cert SSL_CERT] [--ssl-key SSL_KEY]
                  [--stderr STDERR] [--stdout STDOUT] [-t WORKER_TIMEOUT]
                  [-k {sync,eventlet,gevent,tornado}] [-w WORKERS]
```

Аргументы

Параметр	Описание
-A, --access-logfile	Файл для хранения логов доступа к веб-серверу. Используйте `-` для печати в stdout
-L, --access-logformat	Формат логов доступа для логов gunicorn
-D, --daemon	Запустить выполнение команды в качестве процесса-демона
-d, --debug	Использовать сервер, который поставляется с Flask, в режиме отладки
-E, --error-logfile	Файл журнала для хранения логов ошибок веб-сервера. Укажите `-` для печати в stderr
-H, --hostname	Имя хоста, на котором будет работать веб-сервер
-l, --log-file	Расположение логов
--pid	Расположение PID-файла
-p, --port	Порт, на котором будет запущен сервер
--ssl-cert	Путь к SSL-сертификату для веб-сервера
--ssl-key	Путь к ключу для использования с SSL-сертификатом
--stderr	Перенаправить stderr в указанный файл
--stdout	Перенаправить stdout в указанный файл
-t, --worker-timeout	Тайм-аут ожидания на рабочих процессах веб-сервера
-k, --workerclass	Рабочий класс, используемый для Gunicorn. Возможные варианты: `sync`, `eventlet`, `gevent`, `tornado`. Значение по умолчанию: `sync`
-w, --workers	Количество worker-процессов для запуска веб-сервера

Примеры

Пример команды для запуска веб-сервера:

```
$ airflow webserver -p 8080 -H 0.0.0.0 -w 4 -D -A /logs/access.log -E /logs/error.log --ssl-cert /etc/ssl/certs/airflow.crt --ssl-key /etc/ssl/private/airflow.key
```

flower

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Запускает Celery Flower. Вы можете выполнить эту команду с помощью действия ***Add/Remove components*** в ADCM.

Использование

```
$ airflow celery flower [-h] [-A BASIC_AUTH] [-a BROKER_API] [-D]
                        [-c FLOWER_CONF] [-H HOSTNAME] [-l LOG_FILE]
                        [--pid [PID]] [-p PORT] [--stderr STDERR]
                        [--stdout STDOUT] [-u URL_PREFIX] [-v]
```

Аргументы

Параметр	Описание
-A, --basic-auth	Включить базовую аутентификацию для Flower. Принимает пары `user:password`, разделенные запятой. Например: `flower_basic_auth = user1:password1,user2:password2`
-a, --broker-api	Broker API
-D, --daemon	Запустить выполнение команды в качестве процесса-демона
-c, --flower-conf	Файл конфигурации для Flower
-H, --hostname	Имя хоста, на котором будет запущен сервер
-l, --log-file	Расположение логов
--pid	Расположение PID-файла
-p, --port	Порт, на котором будет запущен сервер. Значение по умолчанию: `5555`
--stderr	Перенаправить stderr в указанный файл
--stdout	Перенаправить stdout в указанный файл
-u, --url-prefix	URL-префикс для Flower
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

stop

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Безопасная остановка Celery worker.

Использование

```
$ airflow celery stop
                    [-h]
                    [--pid [PID]]
                    [-v]
```

Аргументы

Параметр	Описание
--pid	Расположение PID-файла
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

worker

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Запуск worker-ноды Celery. Вы можете выполнить эту команду с помощью действия ***Add/Remove components*** в ADCM.

Использование

```
$ airflow celery worker [-h] [-a AUTOSCALE] [-H CELERY_HOSTNAME]
                        [-c CONCURRENCY] [-D] [-l LOG_FILE] [--pid [PID]]
                        [-q QUEUES] [-s] [--stderr STDERR] [--stdout STDOUT]
                        [-u UMASK] [-v] [--without-gossip] [--without-mingle]
```

Аргументы

Параметр	Описание
-a, --autoscale	Минимальное и максимальное количество рабочих процессов для автоматического масштабирования
-H, --celery-hostname	Имя хоста Celery worker, если у вас несколько worker на одном хосте
-c, --concurrency	Количество рабочих процессов. Значение по умолчанию: `16`
-D, --daemon	Запустить выполнение команды в качестве процесса-демона. Значение по умолчанию: `False`
-l, --log-file	Расположение логов
--pid	Расположение PID-файла
-q, --queues	Список очередей для обслуживания, разделенных запятыми. Значение по умолчанию: `default`
-s, --skip-serve-logs	Не запускать процесс обработки логов одновременно с процессами обработки. Значение по умолчанию: `False`
--stderr	Перенаправить stderr в указанный файл
--stdout	Перенаправить stdout в указанный файл
-u, --umask	Установить umask для Celery worker в режиме демона
--without-gossip	Не подписываться на события других worker. Значение по умолчанию: `False`
--without-mingle	Не синхронизироваться с другими worker на начальном этапе. Значение по умолчанию: `False`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

get-value

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит на экран значение параметра конфигурации.

Использование

```
$ airflow config get-value [-h] [-v] section option
```

Аргументы

Параметр	Описание
section	Название раздела конфигурации
option	Название параметра
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы узнать значение параметра `dags_folder` в настройках core, выполните:

```
$ airflow config get-value core dags_folder
```

Пример вывода:

```
/opt/airflow/dags
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание
Использование
└ Аргументы
Примеры

Описание

Выводит список параметров в конфигурации.

Использование

```
$ airflow config list [-h] [--color {off,auto,on}] [-c] [-a] [-p] [-d] [-V] [-e]
                      [-s] [--section SECTION] [-v]
```

Аргументы

Параметр	Описание
--color	Сделать вывод цветным. Возможные значения: `off`, `auto`, `on`. Значение по умолчанию: `auto`
-c, --comment-out-everything	Отключить все параметры конфигурации. Полезно в качестве отправной точки для новой установки
-a, --defaults	Показывать только значения по умолчанию, не показывать локальную конфигурацию, источники, описания, примеры, переменные
-p, --exclude-providers	Исключить конфигурацию провайдера (по умолчанию включена)
-d, --include-descriptions	Показать описания для переменных конфигурации
-V, --include-env-vars	Показать переменную среды для каждого параметра
-e, --include-examples	Показать примеры для переменных конфигурации
-s, --include-sources	Показать источник переменной конфигурации
--section	Название раздела
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы показать основную конфигурацию Airflow, выполните команду:

```
$ airflow config list --section core
```

Пример вывода:

```
[core]
dags_folder = /opt/airflow/dags
hostname_callable = airflow.utils.net.getfqdn
might_contain_dag_callable = airflow.utils.file.might_contain_dag_via_default_heuristic
default_timezone = utc
executor = CeleryExecutor
parallelism = 32
max_active_tasks_per_dag = 16
dags_are_paused_at_creation = True
max_active_runs_per_dag = 16
load_examples = True
plugins_folder = /opt/airflow/plugins
execute_tasks_new_python_interpreter = False
fernet_key = 9smzpoHUQlm4eu_E2vufmVUWgm3A6tClkEuhYu48aC8=
donot_pickle = True
dagbag_import_timeout = 30
dagbag_import_error_tracebacks = True
dagbag_import_error_traceback_depth = 2
dag_file_processor_timeout = 50
task_runner = StandardTaskRunner
default_impersonation =
security =
unit_test_mode = False
enable_xcom_pickling = False
allowed_deserialization_classes = airflow\..*
killed_task_cleanup_time = 60
dag_run_conf_overrides_params = True
dag_discovery_safe_mode = True
dag_ignore_file_syntax = regexp
default_task_retries = 0
default_task_retry_delay = 300
max_task_retry_delay = 86400
default_task_weight_rule = downstream
default_task_execution_timeout =
min_serialized_dag_update_interval = 30
compress_serialized_dags = False
min_serialized_dag_fetch_interval = 10
max_num_rendered_ti_fields_per_task = 30
check_slas = True
xcom_backend = airflow.models.xcom.BaseXCom
lazy_load_plugins = True
lazy_discover_providers = True
hide_sensitive_var_conn_fields = True
sensitive_var_conn_names =
default_pool_task_slot_count = 128
max_map_length = 1024
daemon_umask = 0o077
database_access_isolation = False
fernet_key_secret = fernet_key_value
```

add

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Добавляет соединение.

Использование

```
$ airflow connections add [-h] [--conn-description CONN_DESCRIPTION]
                          [--conn-extra CONN_EXTRA] [--conn-host CONN_HOST]
                          [--conn-json CONN_JSON] [--conn-login CONN_LOGIN]
                          [--conn-password CONN_PASSWORD]
                          [--conn-port CONN_PORT] [--conn-schema CONN_SCHEMA]
                          [--conn-type CONN_TYPE] [--conn-uri CONN_URI]
                          conn_id
```

Аргументы

Параметр	Описание
conn_id	Идентификатор соединения, необходимый для добавления соединения
--conn-description	Описание соединения (опционально)
--conn-extra	Дополнительное поле подключения (опционально)
--conn-host	Хост соединения (опционально)
--conn-json	Соединение JSON, необходимо для добавления соединения с использованием представления JSON
--conn-login	Логин подключения (опционально)
--conn-password	Пароль подключения (опционально)
--conn-port	Порт соединения (опционально)
--conn-schema	Схема соединения (опционально)
--conn-type	Тип соединения. Необходимо указать либо `conn_uri`, либо `conn_type`
--conn-uri	URI подключения. Необходимо указать либо `conn_uri`, либо `conn_type`

Примеры

Чтобы добавить соединение, выполните:

```
$ airflow connections add --conn-type http test_connection
```

Пример вывода:

```
Successfully added `conn_id`=test_connection
```

delete

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет соединение.

Использование

```
$ airflow connections delete [-h] [--color {off,auto,on}] [-v] conn_id
```

Аргументы

Параметр	Описание
conn_id	Идентификатор соединения, которое необходимо удалить
--color	Включает цветной вывод. Возможные значения: `off`, `auto`, `on`. Значение по умолчанию: `auto`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы удалить указанное соединение, выполните:

```
$ airflow connections delete test_connection
```

Пример вывода:

```
Successfully deleted connection with `conn_id`=test_connection
```

export

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Экспортирует все соединения в stdout или файл.

Использование

```
$ airflow connections export [-h] [--file-format {json,yaml,env}]
                             [--format {json,yaml,env}]
                             [--serialization-format {json,uri}] [-v]
                             file
```

Аргументы

Параметр	Описание
file	Путь к файлу, в который необходимо сохранить соединения
--file-format	Формат файла для экспорта. Возможные значения: `json`, `yaml`, `env`
--format	Формат файла для экспорта. Возможные значения: `json`, `yaml`, `env`. Устаревшая опция. Вместо нее рекомендуется использовать `--file-format`
--serialization-format	При экспорте в формат ENV определяет, как должны быть сериализованы соединения. Возможные значения: `json`, `uri` (по умолчанию)
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Следующая команда экспортирует соединения в формате JSON:

```
$ airflow connections export /tmp/connections.json
```

Пример вывода:

```
Connections successfully exported to /tmp/connections.json.
```

get

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит информацию о соединении.

Использование

```
$ airflow connections get [-h] [--color {off,auto,on}]
                          [-o table, json, yaml, plain] [-v]
                          conn_id
```

Аргументы

Параметр	Описание
conn_id	Идентификатор соединения, необходимый для вывода информации о соединении
--color	Цветной вывод команды. Возможные значения: `off`, `auto`, `on`. Значение по умолчанию: `auto`
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить информацию о соединении, выполните:

```
$ airflow connections get ftp_default
```

Пример вывода:

```
   |            |           |             |           |        |         |          |      | is_encrypte | is_extra_en |
extra_dejso |
id | conn_id     | conn_type | description | host      | schema | login   | password | port | d           | crypted     | n
| get_uri
===+=============+===========+=============+===========+========+=========+==========+======+=============+=============+========
=====+=============
18 | ftp_default | ftp       | None        | localhost | None   | airflow | airflow  | 21   | True        | True        |
{'key_file' | ftp://airflo
   |            |           |             |           |        |         |          |      |             |             | :
| w:airflow@lo
   |            |           |             |           |        |         |          |      |             |             | '~/.
ssh/id_ | calhost:21/?
   |            |           |             |           |        |         |          |      |             |             | rsa',
| __extra__=%7
   |            |           |             |           |        |         |          |      |             |             |
'no_host_ke | B%22key_file
   |            |           |             |           |        |         |          |      |             |             |
y_check':   | %22%3A+%22~%
   |            |           |             |           |        |         |          |      |             |             | True}
| 2F.ssh%2Fid_
   |            |           |             |           |        |         |          |      |             |             |
| rsa%22%2C+%2
   |            |           |             |           |        |         |          |      |             |             |
| 2no_host_key
   |            |           |             |           |        |         |          |      |             |             |
| _check%22%3A
   |            |           |             |           |        |         |          |      |             |             |
| +true%7D
```

import

Елена Костюченко

Содержание

Назад к содержанию

Описание
Использование
Аргументы
Примеры

Описание

Импортируйте соединения в Airflow из файла. Этот файл может быть выводом команды Airflow export.

Тип файла, указанный в команде, должен быть JSON, YAML или ENV и будет определен автоматически.

Использование

```
$ airflow connections import [-h] [--overwrite] [-v] file
```

Аргументы

Параметр	Описание
file	Путь к файлу, из которого нужно импортировать соединения
--overwrite	Перезаписать существующие записи в случае возникновения конфликта
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы импортировать соединения из файла, которые перезапишут существующие, выполните:

```
$ airflow connections import --overwrite /tmp/connections.json
```

Пример вывода:

```
Imported connection airflow_db
Imported connection aws_default
Imported connection azure_batch_default
Imported connection azure_cosmos_default
Imported connection azure_data_explorer_default
Imported connection azure_data_lake_default
Imported connection azure_default
Imported connection cassandra_default
Imported connection databricks_default
Imported connection dingding_default
Imported connection drill_default
Imported connection druid_broker_default
Imported connection druid_ingest_default
Imported connection elasticsearch_default
Imported connection emr_default
Imported connection facebook_default
Imported connection fs_default
Imported connection ftp_default
Imported connection google_cloud_default
Imported connection hive_cli_default
Imported connection hiveserver2_default
Imported connection http_default
Imported connection impala_default
Imported connection kubernetes_default
Imported connection kylin_default
Imported connection leveldb_default
Imported connection livy_default
Imported connection local_mysql
Imported connection metastore_default
Imported connection mongo_default
Imported connection mssql_default
Imported connection mysql_default
Imported connection opsgenie_default
Imported connection oracle_default
Imported connection oss_default
Imported connection pig_cli_default
Imported connection pinot_admin_default
Imported connection pinot_broker_default
Imported connection postgres_default
Imported connection presto_default
Imported connection qubole_default
Imported connection redis_default
Imported connection redshift_default
Imported connection segment_default
Imported connection sftp_default
Imported connection spark_default
Imported connection sqlite_default
Imported connection sqoop_default
Imported connection ssh_default
Imported connection tableau_default
Imported connection tabular_default
Imported connection test_connection
Imported connection trino_default
Imported connection vertica_default
Imported connection wasb_default
Imported connection webhdfs_default
Imported connection yandexcloud_default
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит список соединений.

Использование

```
$ airflow connections list [-h] [--conn-id CONN_ID]
                           [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
--conn-id	Если передано, будут отображены только элементы с указанным идентификатором соединения
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить информацию об указанном соединении, выполните:

```
$ airflow connections list --conn-id ssh_default
```

Пример вывода:

```
   |            |          |            |          |       |      |         |     |              | is_extra_en |
|
id | conn_id    | conn_type | description | host     | schema | login | password | port | is_encrypted | crypted     |
extra_dejson | get_uri
===+============+===========+============+==========+========+=======+==========+======+==============+=============+=========
=====+=============
50 | ssh_default | ssh       | None        | localhost | None   | None  | None     | None | False        | False       | {}
| ssh://localh
   |            |          |            |          |       |      |         |     |              |             |
| ost
```

test

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Проверяет соединение.

Использование

```
$ airflow connections test [-h] [-v] conn_id
```

Аргументы

Параметр	Описание
conn_id	Идентификатор подключения, которое требуется проверерить
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы проверить указанное соединение, выполните:

```
$ airflow connections test --conn-id ssh_default
```

backfill

Елена Костюченко

Содержание

Назад к содержанию

Описание
Использование
└ Аргументы
Примеры

Описание

`backfill` — это подкоманда в группе команд `dag`. Она позволяет запускать подразделы DAG для указанного диапазона дат.

Если используется опция `reset_dag_run`, команда сначала запросит, следует ли Airflow очистить все предыдущие запуски DAG и экземпляры задач в указанном диапазоне дат.

Если используется опция `rerun_failed_tasks`, Airflow повторно запустит предыдущие экземпляры задач, завершившихся неудачно, в указанном диапазоне дат.

Использование

```
$ airflow dags backfill [-h] [-c CONF] [--continue-on-failures]
                        [--delay-on-limit DELAY_ON_LIMIT] [--disable-retry] [-x]
                        [-n] [-e END_DATE] [-i] [-I] [-l] [-m] [--pool POOL]
                        [--rerun-failed-tasks] [--reset-dagruns] [-B]
                        [-s START_DATE] [-S SUBDIR] [-t TASK_REGEX]
                        [--treat-dag-as-regex] [--treat-dag-id-as-regex] [-v]
                        [-y]
                        dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG для заполнения
-s, --start-date	Дата начала периода заполнения в формате ГГГГ-ММ-ДД
-e, --end-date	Дата окончания периода заполнения в формате ГГГГ-ММ-ДД
-m, --mark-success	Отметить задания как выполненные, не запуская их
-i, --ignore-dependencies	Пропустить задачи верхнего уровня; запустить только те задачи, которые соответствуют регулярному выражению
-t, --task-regex	Шаблон регулярного выражения для фильтрации определенных идентификаторов задач для заполнения
-I, --ignore-first-depends-on-past	Игнорировать зависимости `depends_on_past` только для первого набора задач
-l, --local	Запустить задачу с помощью LocalExecutor
-x, --donot-pickle	Не консервировать объект DAG для отправки к worker-процессам. В этом случае они будут использовать доступную версию кода
-n, --dry-run	Выполнить тестовый запуск команды
--pool	Пул ресурсов для использования
--rerun-failed-tasks	Повторно запустить ранее неудавшиеся экземпляры задач в указанном диапазоне дат
--reset-dagruns	Удалить существующие запуски DAG, связанные с обратным заполнением, и начать заново
-B, --run-backwards	Сначала запускать задачи с самой последней даты. Несовместимо с `depend_on_past`
--delay-on-limit	Время ожидания (в секундах) перед повторной попыткой при достижении максимального количества активных запусков DAG
--disable-retry	Отметить неуспешные задачи как завершенные с ошибкой (failed) без повторных попыток
-c, --conf	Строка JSON, которая будет сохранена в атрибуте `conf` DAG run
-y	Пропустить запрос подтверждения при сбросе DAG-запусков
-S, --subdir	Путь к каталогу файлов DAG; по умолчанию `[AIRFLOW_HOME]/dags`
--treat-dag-as-regex	Интерпретировать аргумент `dag_id` как шаблон регулярного выражения для сопоставления с несколькими DAG
--treat-dag-id-as-regex	Алиас для `--treat-dag-as-regex`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы запустить одну выбранную задачу из указанной группы DAG, выполните:

```
$ airflow dags backfill example_complex -s 2025-04-03 -e 2025-04-04 -t get_entry_group
```

delete

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удалить все записи БД, относящиеся к указанной группе DAG.

Использование

```
$ airflow dags delete [-h] [-v] [-y] dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG для удаления
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

Примеры

Чтобы удалить информацию об указанном DAG, выполните:

```
$ airflow dags delete test_dag
```

Пример вывода:

```
[2025-04-03T14:38:17.121+0000] {__init__.py:42} INFO - Loaded API auth backend: airflow.api.auth.backend.session
[2025-04-03T14:38:17.123+0000] {__init__.py:42} INFO - Loaded API auth backend: airflow.api.auth.backend.basic_auth
This will drop all existing records related to the specified DAG. Proceed? (y/n)
[2025-04-03T14:38:44.671+0000] {delete_dag.py:49} INFO - Deleting DAG: test_dag
Removed 1 record(s)
```

details

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает сведения о DAG для конкретного идентификатора DAG.

Использование

```
$ airflow dags details [-h] [-o table, json, yaml, plain] [-v] dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить информацию об указанном DAG, выполните:

```
$ airflow dags details tutorial
```

Пример вывода:

```
property_name                   | property_value
================================+===================================================================================
===================
schedule_interval               | {'__type': 'TimeDelta', 'days': 1, 'seconds': 0, 'microseconds': 0}
last_expired                    | None
scheduler_lock                  | None
has_task_concurrency_limits     | False
next_dagrun                     | 2025-04-13T14:44:10.740990+00:00
next_dagrun_create_after        | 2025-04-14T14:44:10.740990+00:00
next_dagrun_data_interval_start | 2025-04-13T14:44:10.740990+00:00
pickle_id                       | None
file_token                      | .
eJw9ykESQDAMAMC_9K5hPMgEURkhmSaG_t7NnjeBWgBy3UQfEJ7BWux6jXnowTmoM1wOLOR_ohdPE5pWLA5xh1ZGydbSByZCHCk.Cm0xzMLrV1KcD
                                | KI1UB5RJx-4EOs
has_import_errors               | False
is_paused                       | True
next_dagrun_data_interval_end   | 2025-04-14T14:44:10.740990+00:00
max_active_tasks                | 16
dag_id                          | tutorial
is_subdag                       | False
is_active                       | True
timetable_description           |
description                     | A simple tutorial DAG
fileloc                         | /opt/airflow/lib/python3.10/site-packages/airflow/example_dags/tutorial.py
last_pickled                    | None
owners                          | airflow
root_dag_id                     | None
default_view                    | grid
max_active_runs                 | 16
tags                            | {'name': 'example'}
last_parsed_time                | 2025-04-14T14:44:10.740850+00:00
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит список всех DAG.

Использование

```
$ airflow dags list [-h] [-o table, json, yaml, plain]
                    [-S SUBDIR] [-v]
```

Аргументы

Параметр	Описание
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

Примеры

Чтобы получить список DAG, выполните:

```
$ airflow dags list
```

Пример вывода:

```
dag_id                                                    | filepath
| owner   | paused
==========================================================+=================================================================================
=+=========+=======
adcm_check                                                | adcm_check.py
| airflow | True
dataset_consumes_1                                        | /opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_
| airflow | True
                                                          | datasets.py
|         |
dataset_consumes_1_and_2                                  | /opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_
| airflow | True
                                                          | datasets.py
|         |
dataset_consumes_1_never_scheduled                        | /opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_
| airflow | True
                                                          | datasets.py
|         |
dataset_consumes_unknown_never_scheduled                  | /opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_
| airflow | True
```

list-import-errors

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Перечисляет все группы DAG, имеющие ошибки импорта.

Использование

```
$ airflow dags list-import-errors [-h] [-o table, json, yaml, plain] [-S SUBDIR]
                                  [-v]
```

Аргументы

Параметр	Описание
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

list-jobs

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит информацию о задачах.

Использование

```
$ airflow dags list-jobs [-h] [-d DAG_ID] [--limit LIMIT]
                         [-o table, json, yaml, plain]
                         [--state running, success, restarting, failed] [-v]
```

Аргументы

Параметр	Описание
-d, --dag-id	Идентификатор DAG
--limit	Возврат ограниченного количества записей
--state	Показать только задания в определенном состоянии. Возможные варианты: `running`, `success`, `restarting`, `failed`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

Примеры

Чтобы отобразить невыполненные задания, используйте команду:

```
$ airflow dags list-jobs --state failed
```

Пример вывода:

```
dag_id | state  | job_type     | start_date                      | end_date
=======+========+==============+=================================+=========
None   | failed | SchedulerJob | 2025-04-13 06:05:32.765139+00:00 | None
None   | failed | SchedulerJob | 2025-04-04 07:04:45.232394+00:00 | None
None   | failed | SchedulerJob | 2025-03-20 08:14:14.726727+00:00 | None
None   | failed | SchedulerJob | 2025-03-18 08:42:55.290413+00:00 | None
None   | failed | SchedulerJob | 2025-03-12 12:26:23.617197+00:00 | None
None   | failed | SchedulerJob | 2025-03-10 08:30:22.448855+00:00 | None
None   | failed | SchedulerJob | 2025-03-06 10:55:55.349423+00:00 | None
```

list-runs

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Показывает запуски для выбранного DAG.

Использование

```
$ airflow dags list-runs [-h] -d DAG_ID [-e END_DATE] [--no-backfill]
                         [-o table, json, yaml, plain] [-s START_DATE]
                         [--state queued, running, success, failed] [-v]
```

Аргументы

Параметр	Описание
-d, --dag-id	Идентификатор DAG
-s, --start-date	Если указана `start_date`, будут отфильтрованы все запуски DAG, которые были выполнены до этой даты
-e, --end-date	Если указана `end_date`, будут отфильтрованы все запуски DAG, которые были выполнены после этой даты
--no-backfill	Если используется опция `no_backfill`, будут отфильтрованы все backfill-запуски для указанного DAG
--state	Вывести запуски DAG с указанным состоянием. Возможные варианты: `queued`, `running`, `success`, `failed`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

Примеры

Следующая команда отображает информацию о запущенных группах DAG с идентификатором `example_complex`:

```
$ airflow dags list-runs -d example_complex --state running
```

Пример вывода:

```
dag_id          | run_id                      | state   | execution_date               | start_date                   | end_date
================+=============================+=========+==============================+==============================+===========
===================
example_complex | manual__2025-04-14T15:29:17 | success | 2025-04-14T15:29:17.298007+0 | 2025-04-14T15:29:17.556324+ | 2025-04-
14T15:29:43.118988+00
                | .298007+00:00               |         | 0:00                         | 00:00                        | :00
example_complex | manual__2025-04-14T14:33:55 | success | 2025-04-14T14:33:55.429076+0 | 2025-04-14T14:33:55.901567+ | 2025-04-
14T14:34:23.912882+00
                | .429076+00:00               |         | 0:00                         | 00:00                        | :00
```

next-execution

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Позволяет получить дату и время следующего выполнения DAG. Возвращает одну дату если не указана опция `--num-executions`.

Использование

```
$ airflow dags next-execution [-h] [-n NUM_EXECUTIONS] [-S SUBDIR] [-v] dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
-n, --num-executions	Количество дат, которые нужно отобразить. Значение по умолчанию: `1`
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы получить дату следующего выполнения DAG, используйте команду:

```
$ airflow dags next-execution adcm_check
```

Пример вывода:

```
[INFO] Please be reminded this DAG is PAUSED now.
2025-04-04T04:37:12.246784+00:00
```

pause

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Приостанавливает один или несколько DAG. Эта команда позволяет остановить выполнение указанных DAG, отключая дальнейшее планирование задач. Используйте опцию `-treat-dag-id-as-regex` для выбора нескольких DAG, рассматривая `-dag-id` как шаблон регулярного выражения.

Использование

```
$ airflow dags pause [-h] [-o table, json, yaml, plain] [-S SUBDIR]
                     [--treat-dag-id-as-regex] [-v] [-y]
                     dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
--treat-dag-id-as-regex	Рассматривать `dag_id` как регулярное выражение
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

Примеры

Чтобы приостановить работу DAG, выполните:

```
$ airflow dags pause tutorial
```

Пример вывода:

```
Dag: tutorial, paused: True
```

report

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает отчет по парсингу DAG.

Использование

```
$ airflow dags report [-h] [-o table, json, yaml, plain] [-S SUBDIR] [-v]
```

Аргументы

Параметр	Описание
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить отчет о загрузке DAG, выполните:

```
$ airflow dags report
```

Пример вывода:

```
=====================================================+================+==========+===========+===================================
===================
/opt/airflow/lib/python3.10/site-packages/airflow/exa | 0:00:00.424125 | 0        | 0         |
mple_dags/plugins/workday.py                          |                |          |           |
/opt/airflow/lib/python3.10/site-packages/airflow/exa | 0:00:00.201374 | 1        | 7         | example_kubernetes_executor
mple_dags/example_kubernetes_executor.py              |                |          |           |
/opt/airflow/lib/python3.10/site-packages/airflow/exa | 0:00:00.031733 | 1        | 6         | example_xcom
mple_dags/example_xcom.py                             |                |          |           |
/opt/airflow/lib/python3.10/site-packages/airflow/exa | 0:00:00.013801 | 1        | 1         | example_params_ui_tutorial
mple_dags/example_params_ui_tutorial.py               |                |          |           |
```

reserialize

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Удаляет все сериализованные DAG из базы метаданных. Это приведет к повторной сериализации всех DAG из директории DAG.

Это может быть полезно, если сериализованные DAG не синхронизируются с используемой версией Airflow.

Использование

```
$ airflow dags reserialize [-h] [--clear-only] [-S SUBDIR] [-v]
```

Аргументы

Параметр	Описание
--clear-only	Если передано, сериализованные DAG будут удалены, но не будут повторно сериализованы
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

show

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Эта команда создает визуальное представление задач и зависимостей в группе DAG.

Использование

```
$ airflow dags show [-h] [--imgcat] [-s SAVE] [-S SUBDIR] [-v]
```

Аргументы

Параметр	Описание
--imgcat	Отображает график с помощью инструмента imgcat. Эта опция работает только в iTerm
-s, --save	Сохраняет результат в указанный файл. Формат файла определяется расширением файла. Для получения дополнительной информации о поддерживаемых форматах см. Output formats
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы создать PNG-файл с графиком информации DAG, выполните следующую команду:

```
$ airflow dags show -save output.png
```

show-dependencies

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Создает визуальное представление зависимостей в DAG.

Использование

```
$ airflow dags show-dependencies [-h] [--imgcat] [-s SAVE] [-S SUBDIR] [-v]
```

Аргументы

Параметр	Описание
--imgcat	Отображает график с помощью инструмента imgcat. Эта опция работает только в iTerm
-s, --save	Сохраняет результат в указанный файл. Формат файла определяется расширением файла. Для получения дополнительной информации о поддерживаемых форматах см. Output formats
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить зависимости, выполните:

```
$ airflow dags show-dependencies
```

Пример вывода:

```
digraph {
        graph [rankdir=LR]
        subgraph dataset_produces_2 {
                graph [label=dataset_produces_2 labelloc=t rankdir=LR]
                dataset_produces_2 -> s3:"//dag2/output_1.txt"
                s3:"//dag2/output_1.txt" -> dataset
        }
}
```

state

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает статус выполнения DAG.

Использование

```
$ airflow dags state [-h] [-S SUBDIR] [-v] dag_id execution_date
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
execution_date	Дата исполнения DAG
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы получить статус выполнения DAG, выполните следующее:

```
$ airflow dags state tutorial 04.04.2025
```

test

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Выполняет один запуск DAG для указанного DAG и даты выполнения.

Использование

```
$ airflow dags test [-h] [-c CONF] [--imgcat-dagrun]
                    [--mark-success-pattern MARK_SUCCESS_PATTERN]
                    [--save-dagrun SAVE_DAGRUN] [--show-dagrun] [-S SUBDIR]
                    [--use-executor] [-v]
                    dag_id [execution_date]
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
execution_date	Дата исполнения DAG (необязательно)
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-c, --conf	Строка JSON, которая добавляется в атрибут `conf` запуска DAG
--imgcat-dagrun	После завершения запуска DAG выводит на экран диаграмму для текущего запуска DAG с помощью инструмента imgcat. Эта опция работает только в iTerm
--mark-success-pattern	Не запускать `task_ids`, соответствующие регулярному выражению `<MARK_SUCCESS_PATTERN>`, а вместо этого отметить их как успешные. Может использоваться для пропуска сенсоров проверки зависимостей или шагов очистки при локальном тестировании
--save-dagrun	После завершения сохраняет диаграмму для текущего запуска DAG в указанный файл
--show-dagrun	После завершения показывает диаграмму для текущего запуска DAG. Диаграмма в формате DOT
--use-executor	Использовать для проверки DAG исполнителя (executor), настроенного в среде. По умолчанию команда запускается без него
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

trigger

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Запускает DAG. Если DAG приостановлен, то он останется в очереди и задача не будет запущена.

Использование

```
$ airflow dags trigger [-h] [-c CONF] [-e EXEC_DATE] [--no-replace-microseconds]
                       [-o table, json, yaml, plain] [-r RUN_ID] [-S SUBDIR]
                       [-v]
                       dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
-c, --conf	Строка JSON, которая добавляется в атрибут `conf` запуска DAG
-e, --exec-date	Дата исполнения DAG
--no-replace-microseconds	Следует ли обнулять микросекунды
-r, --run-id	Идентификатор запуска
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы запустить tutorial DAG, выполните:

```
$ airflow dags trigger tutorial
```

Пример вывода:

```
     |          |              | data_interva | data_interva |          | external_tri | last_schedul | logical_dat |          |
|
conf | dag_id   | dag_run_id   | l_end        | l_start      | end_date | gger         | ing_decision | e           | run_type |
start_date | state
=====+==========+==============+==============+==============+==========+==============+==============+=============+==========+=
===========+=======
{}   | tutorial | manual__2025 | 2025-04-04T0 | 2025-04-04T0 | None     | True         | None         | 2025-04-04T | manual   |
None       | queued
     |          | -04-04T05:16 | 5:16:28+00:0 | 5:16:28+00:0 |          |              |              | 05:16:28+00 |          |
|
     |          | :28+00:00    | 0            | 0            |          |              |              | :00         |          |
|
```

unpause

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Восстанавливает выполнение указанных DAG, позволяя осуществлять дальнейшее планирование задач.

Использование

```
$ airflow dags unpause [-h] [-o table, json, yaml, plain] [-S SUBDIR]
                       [--treat-dag-id-as-regex] [-v] [-y]
                       dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
--treat-dag-id-as-regex	Рассматривать `dag_id` как регулярное выражение
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

Примеры

Чтобы возобновить работу tutorial DAG, выполните:

```
$ airflow dags unpause tutorial
```

Пример вывода:

```
Dag: tutorial, paused: False
```

check

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Проверяет, доступна ли база данных.

Использование

```
$ airflow db check
                    [-h]
                    [--retry RETRY]
                    [--retry-delay RETRY_DELAY]
                    [-v]
```

Аргументы

Параметр	Описание
--retry	Повторить проверку базы данных в случае сбоя. Значение по умолчанию — `0` (не повторять)
--retry-delay	Время ожидания между повторными попытками в секундах. Значение по умолчанию: `1`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Для проверки соединения с базой данных выполните:

```
$ airflow db check
```

Пример вывода:

```
[2025-04-08T05:20:05.969+0000] {db.py:1741} INFO - Connection successful.
```

check-migrations

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Проверяет закончилась ли миграция базы данных. Проверка может быть однократной или постоянной до истечения тайм-аута.

Использование

```
$ airflow db check-migrations
                            [-h]
                            [-t MIGRATION_WAIT_TIMEOUT]
                            [-v]
```

Аргументы

Параметр	Описание
-t, --migration-wait-timeout	Тайм-аут ожидания переноса базы данных в секундах. Значение по умолчанию: `60`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы проверить, завершилась ли миграция базы данных через 5 минут, выполните:

```
$ airflow db check-migrations -t 300
```

clean

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет старые записи в таблицах метахранилища.

Использование

```
$ airflow db clean [-h] --clean-before-timestamp CLEAN_BEFORE_TIMESTAMP
                [--dry-run]
                [--skip-archive]
                [-t TABLES]
                [-v]
                [-y]
```

Аргументы

Параметр	Описание
--clean-before-timestamp	Удалить данные, созданные до указанной временной метки. Если информация о часовом поясе не указана, то предполагается, что даты соответствуют часовому поясу Airflow по умолчанию. Пример: `2022-01-01 00:00:00+01:00`
--dry-run	Выполнить пробный запуск команды. Значение по умолчанию: `False`
--skip-archive	Не сохранять удаленные записи в архивной таблице. Значение по умолчанию: `False`
-t, --tables	Имена таблиц, которые необходимо удалить. Принимает список, разделенный запятыми. Возможные значения: `callback_request`, `celery_taskmeta`, `celery_tasksetmeta`, `dag`, `dag_run`, `dataset_event`, `import_error`, `job`, `log`, `session`, `sla_miss`, `task_fail`, `task_instance`, `task_instance_history`, `task_reschedule`, `trigger`, `xcom`
-y, --yes	Не запрашивать подтверждение. Значение по умолчанию: `False`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы удалить записи из таблицы `task_fail`, выполните:

```
$ airflow db clean --clean-before-timestamp 2025-04-03T12:42:04.284+0000 -t task_fail
```

Пример вывода:

```
You have requested that we purge all data prior to 2025-04-03T12:42:04.284000+00:00 for tables ['task_fail'].
This is irreversible.  Consider backing up the tables first and / or doing a dry run with option --dry-run.
Enter 'delete rows' (without quotes) to proceed.
delete rows
Checking table task_fail
Found 1 rows meeting deletion criteria.
Performing Delete...
Moving data to table _airflow_deleted__task_fail__20250403043257
Finished Performing Delete
```

downgrade

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Понижает версию схемы базы метаданных.

Вы можете понизить версию Airflow до той, которая вам необходима, используя параметр `--to-version`. В качестве альтернативы вы можете указать идентификатор ревизии Alembic для понижения версии с помощью параметра `--to-revision`.

Параметры `--from-revision` и `--from-version` могут использоваться только вместе с параметром `--show-sql-only`, поскольку при фактическом запуске миграций версия должна понижаться с текущей.

Если вы хотите просмотреть команды, но не выполнять их, используйте параметр `--show-sql-only`.

Для сопоставления версии Airflow и ревизии Alembic см. справочник по миграции баз данных.

РЕКОМЕНДАЦИЯ

Рекомендуется сделать резервную копию базы данных перед изменением версии или любой другой операцией с базой данных.

Использование

```
$ airflow db downgrade [-h] [--from-revision FROM_REVISION]
                       [--from-version FROM_VERSION] [-s] [-r TO_REVISION]
                       [-n TO_VERSION] [-v] [-y]
```

Аргументы

Параметр	Описание
--from-revision	Понижение версии Alembic с указанной версии (необязательно)
--from-version	Понижение указанной версии (необязательно)
-s, --show-sql-only	Вывести только команды SQL для офлайн-миграции. Необходимо указать эту опцию если используются параметры `--from-revision` или `--from-version`
-r, --to-revision	Alembic-ревизия, к которой нужно перейти
-n, --to-version	Если указано, запускать миграции только до этой версии
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

drop-archived

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет архивные таблицы, созданные с помощью команды очистки БД.

Использование

```
$ airflow db drop-archived [-h] [-t TABLES] [-y]
```

Аргументы

Параметр	Описание
-t, --tables	Имена таблиц, которые нужно удалить (принимает список, разделенный запятыми). Например: `callback_request`, `celery_taskmeta`, `celery_tasksetmeta`, `dag`, `dag_run`, `dataset_event`, `import_error`, `job`, `log`, `session`, `sla_miss`, `task_fail`, `task_instance`, `task_instance_history`, `task_reschedule`, `trigger`, `xcom`
-y, --yes	Не запрашивать подтверждение

Примеры

Чтобы удалить архивную таблицу `session`, выполните:

```
$ airflow db drop-archived -t session
```

export-archived

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Экспортирует данные из архивных таблиц.

Использование

```
$ airflow db export-archived [-h] [--drop-archives] [--export-format {csv}]
                             --output-path DIRPATH [-t TABLES] [-y]
```

Аргументы

Параметр	Описание
--drop-archives	Удалить архивные таблицы после экспорта
--export-format	Формат файла для экспорта данных. Значение по умолчанию: `csv`
--output-path	Путь к каталогу для экспорта данных. Каталог должен существовать
-t, --tables	Имена таблиц для выполнения обслуживания (используйте список, разделенный запятыми). Параметры: `callback_request`, `celery_taskmeta`, `celery_tasksetmeta`, `dag`, `dag_run`, `dataset_event`, `import_error`, `job`, `log`, `session`, `sla_miss`, `task_fail`, `task_instance`, `task_instance_history`, `task_reschedule`, `trigger`, `xcom`
-y, --yes	Не запрашивать подтверждение

Примеры

Чтобы экспортировать архивную таблицу `session`, выполните:

```
$ airflow db export-archived --output-path /tmp/exported-sessions.json session
```

init

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Инициализирует базу метаданных.

Использование

```
$ airflow db init [-h] [-v]
```

Аргументы

Параметр	Описание
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы инициализировать базу метаданных, выполните:

```
$ airflow db init
```

reset

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Удаляет и заново создает базу метаданных.

Использование

```
$ airflow db reset [-h] [-s] [-m] [-v] [-y]
```

Аргументы

Параметр	Описание
-s, --skip-init	Только удалить таблицы и не выполнять `db init`
-m, --use-migration-files	Использовать файлы миграции
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды
-y, --yes	Не запрашивать подтверждение

shell

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Запускает shell-скрипт для доступа к базе данных.

Использование

```
$ airflow db shell [-h] [-v]
```

Аргументы

Параметр	Описание
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

upgrade

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Обновляет базу метаданных до выбранной версии.

Использование

```
$ airflow db upgrade [-h] [--from-revision FROM_REVISION]
                     [--from-version FROM_VERSION] [-s] [-r TO_REVISION]
                     [-n TO_VERSION] [-m] [-v]
```

Аргументы

Параметр	Описание
--from-revision	(Необязательно) При генерации SQL передайте в этом параметре Alembic-версию, с которой необходимо обновиться
--from-version	(Опционально) При генерации SQL передайте в этом параметре версию, с которой необходимо обновиться
-s, --show-sql-only	Вместо запуска миграции сгенерировать скрипты SQL для офлайн-миграции. Обязательная опция при использовании параметров `--from-revision` или `--from-version`
-r, --to-revision	Выполнить миграции до указанной Alembic-версии
-n, --to-version	Версия Airflow, на которую необходимо перейти
-m, --use-migration-files	Использовать файлы миграции
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды, отображающей инструкции SQL для обновления базы данных:

```
$ airflow db upgrade --from-version 2.3.2 --to-version 2.3.3  --show-sql-only
```

Пример вывода:

```
DB: postgresql+psycopg2://airflow:***@elenas-ado.ru-central1.internal:5432/airflow
Generating sql for upgrade -- upgrade commands will *not* be submitted.
BEGIN;

-- Running upgrade 3c94c427fdf6 -> f5fcbda3e651

CREATE INDEX idx_task_fail_task_instance ON task_fail (dag_id, task_id, run_id, map_index);

CREATE INDEX idx_task_reschedule_dag_run ON task_reschedule (dag_id, run_id);

CREATE INDEX idx_xcom_task_instance ON xcom (dag_id, task_id, run_id, map_index);

UPDATE alembic_version SET version_num='f5fcbda3e651' WHERE alembic_version.version_num = '3c94c427fdf6';

COMMIT;
```

jobs check

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Проверяет статус всех заданий в Airflow.

Использование

```
$ airflow jobs check [-h] [--allow-multiple] [--hostname HOSTNAME]
                     [--job-type {BackfillJob,LocalTaskJob,SchedulerJob,TriggererJob,DagProcessorJob}]
                     [--limit LIMIT] [--local] [-v]
```

Аргументы

Параметр	Описание
--allow-multiple	Выполнить команду даже если будет найдено несколько соответствующих активных заданий
--hostname	Проверить задания на указанном хосте
--job-type	Тип проверяемых заданий. Возможные варианты: `BackfillJob`, `LocalTaskJob`, `SchedulerJob`, `TriggererJob`, `DagProcessorJob`
--limit	Количество последних заданий, которые будут проверены. Чтобы отключить лимит, установите `0`. Значение по умолчанию: `1`
--local	Показать только задания с локального хоста. То есть те, у которых имя хоста совпадает с тем, что возвращает `hostname_callable`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы проверить статус всех задач на хосте, выполните:

```
$ airflow jobs check --hostname 10.92.41.34
```

cleanup-pods

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет модули Kubernetes (созданные KubernetesExecutor/KubernetesPodOperator) в состояниях `evicted` / `failed` / `succeeded` / `pending`.

Использование

```
$ airflow kubernetes cleanup-pods [-h]
                                  [--min-pending-minutes MIN_PENDING_MINUTES]
                                  [--namespace NAMESPACE] [-v]
```

Аргументы

Параметр	Описание
--min-pending-minutes	Пороговое значение времени создания модуля (в минутах). Команда удалит модули, время создания которых больше указанного. Значение по умолчанию: `30`. Минимальное значение: `5`
--namespace	Пространство имен Kubernetes. Значение по умолчанию — пространство имен Kubernetes в конфигурации Airflow
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы удалить модули, созданные более часа назад, выполните:

```
$ airflow kubernetes cleanup-pods --min-pending-minutes 60
```

generate-dag-yaml

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Создает файлы YAML для всех задач в DAG. Эта команда полезна для отладки задач без запуска в кластере.

Использование

```
$ airflow kubernetes generate-dag-yaml [-h] [-o OUTPUT_PATH] [-S SUBDIR] [-v]
                                       dag_id logical_date
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
logical_date	Логическая дата (logical date) DAG
-o, --output-path	Путь для сгенерированных файлов YAML. Значение по умолчанию: `[CWD]`
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы сгенерировать файлы YAML для DAG `tutorial`, выполните:

```
$ airflow kubernetes generate-dag-yaml tutorial
```

delete

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Удаляет пул.

Использование

```
$ airflow pools delete [-h] [-o table, json, yaml, plain] [-v] NAME
```

Аргументы

Параметр	Описание
NAME	Имя пула
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

export

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Экспортирует все пулы в файл.

Использование

```
$ airflow pools export [-h] [-v] FILEPATH
```

Аргументы

Параметр	Описание
FILEPATH	Экспортировать все пулы в файл JSON по указанному адресу
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы экспортировать пулы в файл, выполните:

```
$ airflow pools export /tmp/exported-pools.json
```

get

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает размер пула.

Использование

```
$ airflow pools get [-h] [-o table, json, yaml, plain] [-v] NAME
```

Аргументы

Параметр	Описание
NAME	Имя пула
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы получить информацию о пуле `default_pool`, выполните:

```
$ airflow pools get default_pool
```

Пример вывода:

```
pool         | slots | description
=============+=======+=============
default_pool | 128   | Default pool
```

import

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Импортирует пулы из JSON-файла в базу данных.

Использование

```
$ airflow pools import [-h] [-v] file
```

Аргументы

Параметр	Описание
file	Файл JSON, из которого импортируются пулы
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы импортировать пулы из файла, выполните:

```
$ airflow pools import /tmp/pools.json
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает список существующих пулов в выбранном формате.

Использование

```
$ airflow pools list [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. По умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы получить список существующих пулов в выводе консоли, выполните:

```
$ airflow pools list
```

Пример вывода:

```
pool         | slots | description
=============+=======+=============
default_pool | 128   | Default pool
```

set

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Настраивает пул.

Использование

```
$ airflow pools set [-h] [--include-deferred] [-o table, json, yaml, plain] [-v]
                    NAME slots description
```

Аргументы

Параметр	Описание
NAME	Имя пула
slots	Слоты пула
description	Описание пула
--include-deferred	Включить отложенные задачи в расчеты для пула
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. По умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для настройки пула:

```
$ airflow pools set test_pool 50
```

auth

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит информацию о предоставленных бэкендах API Auth.

Использование

```
$ airflow providers auth [-h]
                         [-o table, json, yaml, plain]
                         [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы вывести информацию о предоставленных бэкендах API Auth, выполните команду:

```
$ airflow providers auth
```

Пример вывода:

```
api_auth_backand_module
=======================================================
airflow.providers.google.common.auth_backend.google_openid
```

behaviours

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит информацию о зарегистрированных типах соединений с кастомным поведением.

Использование

```
$ airflow providers behaviours
                                [-h]
                                [-o table, json, yaml, plain]
                                [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы вывести информацию о зарегистрированных соединениях, выполните:

```
$ airflow providers behaviours
```

Пример вывода:

```
field_behaviours
========================
adls
apprise
arangodb
asana
aws
azure
azure_batch
azure_container_registry
azure_container_volume
azure_cosmos
azure_data_explorer
azure_data_factory
azure_data_lake
azure_fileshare
azure_service_bus
azure_synapse
azure_synapse_pipeline
chime
cloudant
dbt_cloud
docker
emr
gcpssh
github
google_cloud_platform
jdbc
jenkins
kafka
kubernetes
postgres
redshift
salesforce
sftp
slack
slackwebhook
smtp
spark
ssh
tabular
vault
wasb
yandexcloud
```

get

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит информацию о провайдере.

Использование

```
$ airflow providers get [-h] [--color {auto,on,off}] [-f]
                        [-o table, json, yaml, plain] [-v]
                        provider_name
```

Аргументы

Параметр	Описание
provider_name	Идентификатор провайдера в Airflow
--color	Цветной вывод команды. Возможные значения: `off`, `auto`, `on`. Значение по умолчанию: `auto`
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить информацию о провайдере, выполните:

```
$ airflow providers get test_provider
```

hooks

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Отображает зарегистрированные хуки провайдера.

Использование

```
$ airflow providers hooks [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

links

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает дополнительные ссылки, зарегистрированные провайдерами.

Использование

```
$ airflow providers links [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить ссылки провайдера, выполните:

```
$ airflow providers links
```

Пример вывода:

```
extra_link_class_name
=====================================================================================
airflow.providers.amazon.aws.links.batch.BatchJobDefinitionLink
airflow.providers.amazon.aws.links.batch.BatchJobDetailsLink
airflow.providers.amazon.aws.links.batch.BatchJobQueueLink
airflow.providers.amazon.aws.links.emr.EmrClusterLink
airflow.providers.amazon.aws.links.emr.EmrLogsLink
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Выводит список установленных провайдеров.

Использование

```
$ airflow providers list [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. По умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для отображения списка установленных провайдеров:

```
$ airflow providers list -o table
```

logging

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает информацию о предоставленных обработчиках логирования.

Использование

```
$ airflow providers logging [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить информацию о провайдерах логирования, выполните:

```
$ airflow providers logging
```

Пример вывода:

```
logging_class_name
==================================================================================
airflow.providers.alibaba.cloud.log.oss_task_handler.OSSTaskHandler
airflow.providers.amazon.aws.log.cloudwatch_task_handler.CloudwatchTaskHandler
airflow.providers.amazon.aws.log.s3_task_handler.S3TaskHandler
airflow.providers.elasticsearch.log.es_task_handler.ElasticsearchTaskHandler
airflow.providers.google.cloud.log.gcs_task_handler.GCSTaskHandler
airflow.providers.google.cloud.log.stackdriver_task_handler.StackdriverTaskHandler
airflow.providers.microsoft.azure.log.wasb_task_handler.WasbTaskHandler
```

secrets

Елена Костюченко

Содержание

Назад к содержанию

Описание
Использование
└ Аргументы
Примеры

Описание

Отображает информацию о предоставленных бэкендах конфиденциальных данных.

Использование

```
$ airflow providers secrets [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить информацию о доступных бэкендах конфиденциальных данных, выполните:

```
$ airflow providers secrets
```

Пример вывода:

```
secrets_backend_class_name
==========================================================================================
airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend
airflow.providers.amazon.aws.secrets.systems_manager.SystemsManagerParameterStoreBackend
airflow.providers.google.cloud.secrets.secret_manager.CloudSecretManagerBackend
airflow.providers.hashicorp.secrets.vault.VaultBackend
airflow.providers.microsoft.azure.secrets.key_vault.AzureKeyVaultBackend
```

widgets

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает информацию с виджетов о зарегистрированных соединениях.

Использование

```
$ airflow providers widgets [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы получить список зарегистрированных подключений виджетов, выполните:

```
$ airflow providers widgets
```

Пример вывода:

```
connection_parameter_name                                  | class                      | package_name
| field_type
============================================================+============================+=====================================
====+==============
extra__adls__connection_string                             | AzureDataLakeStorageV2Hook | apache-airflow-providers-microsoft-
azure | PasswordField
extra__adls__managed_identity_client_id                    | AzureDataLakeStorageV2Hook | apache-airflow-providers-microsoft-
azure | StringField
extra__adls__tenant_id                                     | AzureDataLakeStorageV2Hook | apache-airflow-providers-microsoft-
azure | StringField
extra__adls__workload_identity_tenant_id                   | AzureDataLakeStorageV2Hook | apache-airflow-providers-microsoft-
azure | StringField
extra__apprise__config                                     | AppriseHook                | apache-airflow-providers-apprise
| PasswordField
```

add-perms

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Добавляет разрешения для роли.

Использование

```
$ airflow roles add-perms [role ...]
                          [-h] -a
                          [ACTION ...]
                          -r [RESOURCE ...]
                          [-v]
```

Аргументы

Параметр	Описание
role	Название роли, для которой необходимо добавить разрешения
-a, --action	Действие, которое будет разрешено с выбранным ресурсом
-r, --resource	Ресурс, к которому дается доступ
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы разрешить пользователям с ролью `viewer` редактировать соединения, выполните команду:

```
$ airflow roles add-perms Viewer -a can_edit -r Connections
```

Пример вывода:

```
Added can edit on Connections to role Viewer
Added 1 permission(s)
```

create

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Создает роль.

Использование

```
$ airflow roles create [-h] [-v] [role ...]
```

Аргументы

Параметр	Описание
role	Название роли
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы создать тестовую роль, выполните:

```
$ airflow roles create test_role
```

Пример вывода:

```
Added 1 role(s)
```

del-perms

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удалить разрешения ролей.

Использование

```
$ airflow roles del-perms [-h] [-a [ACTION ...]] -r [RESOURCE ...] [-v]
                          [role ...]
```

Аргументы

Параметр	Описание
role	Название роли
-a, --action	Действие
-r, --resource	Ресурс, доступ к которому нужно запретить
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отменить возможность редактирования подключений из тестовой роли, выполните:

```
$ airflow roles del-perms can_edit connections test_role
```

delete

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет роль.

Использование

```
$ airflow roles delete [-h] [-v] [role ...]
```

Аргументы

Параметр	Описание
role	Название роли
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы удалить тестовую роль, выполните:

```
$ airflow roles delete test_role
```

export

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Экспортирует роли (без разрешений) из базы данных в файл JSON.

Использование

```
$ airflow roles export [-h] [-p] [-v] file
```

Аргументы

Параметр	Описание
file	Имя JSON-файла с экспортированными ролями
-p, --pretty	Отформатировать выходной файл JSON, отсортировав имена ролей и добавив отступ в 4 пробела
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для экспорта ролей:

```
$ airflow roles export /tmp/exported-roles.json
```

import

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Импортирует роли (без разрешений) из JSON-файла в базу данных.

Использование

```
$ airflow roles import [-h] [-v] file
```

Аргументы

Параметр	Описание
file	Файл JSON, из которого импортируются роли
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы импортировать роли из файла, выполните:

```
$ airflow roles import /tmp/roles.json
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает список ролей.

Использование

```
$ airflow roles list [-h] [-o table, json, yaml, plain] [-p] [-v]
```

Аргументы

Параметр	Описание
-p, --permission	Показать разрешения ролей
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить список ролей, выполните:

```
$ airflow roles list
```

Пример вывода:

```
name
=========
Admin
Op
Public
User
Viewer
test_role
```

clear

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет информацию о запусках задач, возвращая их к состоянию как до первого запуска.

Использование

```
$ airflow tasks clear [-h] [-R] [-d] [-e END_DATE] [-X] [-x] [-f] [-r]
                    [-s START_DATE] [-S SUBDIR] [-t TASK_REGEX] [-u] [-v] [-y]
                    dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
-R, --dag-regex	Рассматривать значение `dag_id` как регулярное выражение
-d, --downstream	Включить последующие задачи
-e, --end-date	Переопределить `end_date`. Формат даты `YYYY-MM-DD`
-X, --exclude-parentdag	Исключить родительские DAG, если задача является частью sub-DAG
-x, --exclude-subdags	Исключить sub-DAG
-f, --only-failed	Включить только задания со статусом `failed`
-r, --only-running	Включить только запущенные задачи
-s, --start-date	Переопределить `start_date`. Формат даты — `YYYY-MM-DD`
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию — `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-t, --task-regex	Регулярное выражение для фильтрации определенных `task_ids`
-u, --upstream	Включить upstream-задачи
-y, --yes	Не запрашивать подтверждение
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы удалить все невыполненные задачи для указанного DAG, воспользуйтесь командой:

```
$ airflow tasks clear example_dag_decorator --only-failed
```

Пример вывода:

```
[2025-04-03T04:50:11.935+0000] {dagbag.py:541} INFO - Filling up the DagBag from /opt/airflow/dags
You are about to delete these 1 tasks:
<TaskInstance: example_dag_decorator.send_email manual__2025-04-03T11:27:54.722760+00:00 [failed]>

Are you sure? [y/n]
```

failed-deps

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Возвращает известные планировщику зависимости задачи, которые вызвали ошибку. Эта команда помогает определить, почему экземпляр задачи не запланирован и не поставлен в очередь планировщиком.

Использование

```
$ airflow tasks failed-deps [-h] [--map-index MAP_INDEX] [-S SUBDIR] [-v]
                            dag_id task_id execution_date_or_run_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
task_id	Идентификатор задачи
execution_date_or_run_id	`execution_date` группы DAG или `run_id` запуска группы DAG
--map-index	Индекс сопоставленной задачи. Значение по умолчанию: `-1`
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для отображения неудовлетворенных зависимостей для DAG:

```
$ airflow tasks failed-deps example_dag extract_people manual__2025-04-14T15:29:17.298007+00:00
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает список задач в группе DAG.

Использование

```
$ airflow tasks list [-h] [-S SUBDIR] [-t] [-v] dag_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-t, --tree	Устаревшая опция. Вместо нее используйте команду `dags show`. Отображает задачи в виде дерева. Обратите внимание, что генерация дерева может быть медленной, а вывод может быть очень большим для некоторых DAG
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить список задач в `tutorial` DAG, выполните:

```
$ airflow tasks list tutorial
```

Пример вывода:

```
print_date
sleep
templated
```

render

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает шаблоны задач.

Использование

```
$ airflow tasks render [-h] [--map-index MAP_INDEX] [-S SUBDIR] [-v]
                       dag_id task_id execution_date_or_run_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
task_id	Идентификатор задачи
execution_date_or_run_id	`execution_date` группы DAG или `run_id` запуска группы DAG
--map-index	Индекс сопоставленной задачи. Значение по умолчанию: `-1`
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для отображения шаблонов экземпляра задачи:

```
$ airflow tasks render example_dag extract_people manual__2025-04-14T15:29:17.298007+00:00
```

run

Елена Костюченко

Содержание

Назад к содержанию

Описание
Использование
└ Аргументы
Примеры

Описание

Запускает один экземпляр задачи в DAG.

Использование

```
$ airflow tasks run [-h] [--cfg-path CFG_PATH] [-d {ignore,wait,check}] [-f]
                    [-A] [-i] [-I] [-N] [-l] [--map-index MAP_INDEX] [-m]
                    [-p PICKLE] [--pool POOL] [--read-from-db] [--ship-dag]
                    [-S SUBDIR] [-v]
                    dag_id task_id execution_date_or_run_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
task_id	Идентификатор задачи
execution_date_or_run_id	`execution_date` группы DAG или `run_id` запуска группы DAG
--cfg-path	Путь к файлу конфигурации, который следует использовать вместо *airflow.cfg*
-d, --depends-on-past	Метод работы Airflow с зависимостями. Возможные варианты: ▪ `check` (по умолчанию) — Airflow проверит, выполнены ли зависимости для задач, имеющих `depends_on_past=True`, перед их запуском. ▪ `ignore` — игнорировать зависимости. ▪ `wait` — если `depends_on_past=True`, Airflow будет ждать, пока не будут выполнены прошлые зависимости, прежде чем запустить или пропустить задачу.
-f, --force	Игнорировать предыдущее состояние экземпляра задачи. Перезапустить независимо от того, была ли задача уже выполнена успешно или нет
-A, --ignore-all-dependencies	Игнорирует все некритические зависимости, включая `ignore_ti_state` и `ignore_task_deps`
-i, --ignore-dependencies	Игнорировать зависимости, специфичные для задач. Например, `upstream`, `depend_on_past` и зависимости задержки повторного выполнения (retry delay)
-I, --ignore-depends-on-past	Устаревшая опция. Вместо этого используйте `--depends-on-past ignore`
-N, --interactive	Не захватывать стандартный вывод и потоки ошибок (полезно для интерактивной отладки)
-l, --local	Запустить задачу с помощью LocalExecutor
--map-index	Индекс сопоставленной задачи
-m, --mark-success	Отметить задания как выполненные, не запуская их
-p, --pickle	Сериализованный объект pickle всего DAG (для внутреннего использования)
--pool	Пул ресурсов для использования
--read-from-db	Читать DAG из БД вместо DAG-файла
--ship-dag	Выполнить сериализацию DAG и отправить его рабочему процессу
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для отдельного запуска задачи:

```
$ airflow tasks run tutorial print_date 2025-04-04
```

state

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает статус экземпляра задачи.

Использование

```
$ airflow tasks state [-h] [--map-index MAP_INDEX] [-S SUBDIR] [-v]
                      dag_id task_id execution_date_or_run_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
task_id	Идентификатор задачи
execution_date_or_run_id	`execution_date` группы DAG или `run_id` запуска группы DAG
--map-index	Индекс сопоставленной задачи
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить статус экземпляра задачи `print_date`, выполните:

```
$ airflow tasks state tutorial print_date scheduled__2025-04-13T15:26:57.044494+00:00
```

Пример вывода:

```
success
```

states-for-dag-run

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает состояние всех экземпляров задач в ходе выполнения DAG.

Использование

```
$ airflow tasks states-for-dag-run [-h] [-o table, json, yaml, plain] [-v]
                                   dag_id execution_date_or_run_id
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
execution_date_or_run_id	`execution_date` группы DAG или `run_id` запуска группы DAG
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для отображения информации о статусе задачи:

```
$ airflow tasks states-for-dag-run tutorial scheduled__2025-04-13T15:26:57.044494+00:00
```

Пример вывода:

```
dag_id   | execution_date                  | task_id    | state   | start_date                      | end_date
=========+=================================+============+=========+=================================+=========================
=======
tutorial | 2025-04-13T15:26:57.044494+00:00 | print_date | success | 2025-04-14T15:27:12.075391+00:00 | 2025-04-14T15:27:12.
311052+00:00
tutorial | 2025-04-13T15:26:57.044494+00:00 | templated  | success | 2025-04-14T15:27:13.274360+00:00 | 2025-04-14T15:27:13.
533015+00:00
tutorial | 2025-04-13T15:26:57.044494+00:00 | sleep      | success | 2025-04-14T15:27:13.245757+00:00 | 2025-04-14T15:27:18.
510164+00:00
```

test

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Тестирует экземпляр задачи. Команда запускает задачу без проверки зависимостей или записи ее состояния в базу данных.

Использование

```
$ airflow tasks test [-h] [-n] [--env-vars ENV_VARS] [--map-index MAP_INDEX]
                     [-m] [-S SUBDIR] [-t TASK_PARAMS] [-v]
                     dag_id task_id [execution_date_or_run_id]
```

Аргументы

Параметр	Описание
dag_id	Идентификатор DAG
task_id	Идентификатор задачи
execution_date_or_run_id	`execution_date` группы DAG или `run_id` запуска группы DAG (необязательно)
-n, --dry-run	Выполнить пробный запуск для каждой задачи. Отображает только поля шаблона (Template Fields) для каждой задачи
--env-vars	Установите переменную среды как во время анализа, так и во время выполнения для каждой записи, предоставленной в словаре JSON
--map-index	Индекс сопоставленной задачи
-m, --post-mortem	Открыть отладчик при неперехваченном исключении
-t, --task-params	Отправляет словарь параметров JSON в задачу
-S, --subdir	Расположение файла или каталога, в котором следует искать DAG. По умолчанию `[AIRFLOW_HOME]/dags`, где `[AIRFLOW_HOME]` — это значение, которое задано для конфигурации `AIRFLOW_HOME` в *airflow.cfg*
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для тестирования экземпляра задачи:

```
$ airflow tasks test tutorial print_date scheduled__2025-04-13T15:26:57.044494+00:00
```

Пример вывода:

```
[2025-04-15T07:41:32.631+0000] {dagbag.py:541} INFO - Filling up the DagBag from /opt/airflow/dags
[2025-04-15T07:41:33.134+0000] {example_python_operator.py:90} WARNING - The virtalenv_python example task requires virtualenv,
please install it.
[2025-04-15T07:41:33.729+0000] {tutorial_taskflow_api_virtualenv.py:29} WARNING - The tutorial_taskflow_api_virtualenv example
DAG requires virtualenv, please install it.
[2025-04-15T07:41:34.220+0000] {taskinstance.py:1103} INFO - Dependencies all met for dep_context=non-requeueable deps ti=
<TaskInstance: tutorial.print_date scheduled__2025-04-13T15:26:57.044494+00:00 [success]>
[2025-04-15T07:41:34.227+0000] {taskinstance.py:1103} INFO - Dependencies all met for dep_context=requeueable deps ti=
<TaskInstance: tutorial.print_date scheduled__2025-04-13T15:26:57.044494+00:00 [success]>
[2025-04-15T07:41:34.227+0000] {taskinstance.py:1308} INFO - Starting attempt 2 of 2
[2025-04-15T07:41:34.232+0000] {taskinstance.py:1327} INFO - Executing <Task(BashOperator): print_date> on 2025-04-13 15:26:57.
044494+00:00
[2025-04-15T07:41:34.281+0000] {taskinstance.py:1545} INFO - Exporting env vars: AIRFLOW_CTX_DAG_EMAIL='airflow@example.com'
AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='tutorial' AIRFLOW_CTX_TASK_ID='print_date' AIRFLOW_CTX_EXECUTION_DATE='2025-
04-13T15:26:57.044494+00:00' AIRFLOW_CTX_TRY_NUMBER='2' AIRFLOW_CTX_DAG_RUN_ID='scheduled__2025-04-13T15:26:57.044494+00:00'
[2025-04-15T07:41:34.282+0000] {subprocess.py:63} INFO - Tmp dir root location: /tmp
[2025-04-15T07:41:34.283+0000] {subprocess.py:75} INFO - Running command: ['/bin/bash', '-c', 'date']
[2025-04-15T07:41:34.291+0000] {subprocess.py:86} INFO - Output:
[2025-04-15T07:41:34.293+0000] {subprocess.py:93} INFO - Tue Apr 15 07:41:34 UTC 2025
[2025-04-15T07:41:34.293+0000] {subprocess.py:97} INFO - Command exited with return code 0
[2025-04-15T07:41:34.307+0000] {taskinstance.py:1345} INFO - Marking task as SUCCESS. dag_id=tutorial, task_id=print_date,
execution_date=20250413T152657, start_date=20250414T152712, end_date=20250415T074134
```

add-role

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Добавляет роль для пользователя.

Использование

```
$ airflow users add-role
                        [-h]
                        [-e EMAIL]
                        -r ROLE
                        [-u USERNAME]
                        [-v]
```

Аргументы

Параметр	Описание
-e, --email	Электронная почта пользователя. Укажите либо `--username`, либо `--email`, но не оба сразу
-r, --role	Роль пользователя. Например: `Admin`, `User`, `Op`, `Viewer`, `Public`
-u, --username	Имя пользователя. Укажите либо `--username`, либо `--email`, но не оба сразу
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы добавить пользователю роль администратора, выполните:

```
$ airflow users add-role -e user2@user.com -r Admin
```

Пример вывода:

```
[2025-04-03T12:42:04.284+0000] {manager.py:226} INFO - Updated user john doe
User "user2" added to role "Admin"
```

create

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Создает пользователя.

Использование

```
$ airflow users create [-h] -e EMAIL -f FIRSTNAME -l LASTNAME [-p PASSWORD] -r
                       ROLE [--use-random-password] -u USERNAME [-v]
```

Аргументы

Параметр	Описание
-e, --email	Электронная почта пользователя
-f, --firstname	Имя пользователя
-l, --lastname	Фамилия пользователя
-p, --password	Пароль пользователя. Необходимо указать при создании пользователя без применения опции `--use-random-password`
-r, --role	Роль пользователя. Существующие роли включают: `Admin`, `User`, `Op`, `Viewer`, `Public`
--use-random-password	Сгенерировать случайный пароль. Необходимо указать, если не задан пароль с помощью опции `--password`
-u, --username	Имя пользователя
-v, --verbose	Сделать вывод логов более подробным. Значение по умолчанию: `False`

Примеры

Чтобы создать пользователя с ролью администратора, выполните:

```
$ airflow users create -e user@user.com -f john -l doe -p user1 -r Admin -u user1
```

Пример вывода:

```
[2025-04-03T11:13:35.841+0000] {manager.py:212} INFO - Added user user1
User "user1" created with role "Admin"
```

delete

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет пользователя.

Использование

```
$ airflow users delete [-h] [-e EMAIL] [-u USERNAME] [-v]
```

Аргументы

Параметр	Описание
-e, --email	Электронная почта пользователя
-u, --username	Имя пользователя
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы удалить тестового пользователя, выполните:

```
$ airflow users delete test_user
```

export

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Экспортирует список пользователей в JSON-файл.

Использование

```
$ airflow users export [-h] [-v] FILEPATH
```

Аргументы

Параметр	Описание
FILEPATH	Имя JSON-файла, в котором необходимо сохранить пользователей
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

import

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Импортирует пользователей из JSON-файла в базу данных.

Использование

```
$ airflow users import [-h] [-v] FILEPATH
```

Аргументы

Параметр	Описание
file	Файл JSON, из которого импортируются пользователи
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Пример формата JSON-файла с пользователями:

```
[
    {
        "email": "foo@bar.org",
        "firstname": "Jon",
        "lastname": "Doe",
        "roles": ["Public"],
        "username": "jondoe"
    }
]
```

Примеры

Чтобы импортировать пользователей из файла, выполните:

```
$ airflow users import /tmp/users.json
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает список пользователей.

Использование

```
$ airflow users list [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные значения: `table`, `json`, `yaml`, `plain`. Значение по умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы отобразить список пользователей, выполните:

```
$ airflow users list
```

remove-role

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Удаляет роль пользователя.

Использование

```
$ airflow users remove-role [-h] [-e EMAIL] -r ROLE [-u USERNAME] [-v]
```

Аргументы

Параметр	Описание
-e, --email	Электронная почта пользователя
-r, --role	Роль пользователя. Например: `Admin`, `User`, `Op`, `Viewer`, `Public`
-u, --username	Имя пользователя
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы удалить роль администратора у тестового пользователя, выполните:

```
$ airflow users remove-role -e test@user.com -r admin
```

delete

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Удаляет переменную.

Использование

```
$ airflow variables delete [-h] [-v] key
```

Аргументы

Параметр	Описание
key	Ключ переменной
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

export

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Описание

Экспортирует все переменные в stdout или файл.

Использование

```
$ airflow variables export [-h] [-v] file
```

Аргументы

Параметр	Описание
file	Имя JSON-файла, куда необходимо сохранить переменные
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

get

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Показывает информацию о переменной.

Использование

```
$ airflow variables get [-h] [-d VAL] [-j] [-v] key
```

Аргументы

Параметр	Описание
key	Ключ переменной
-d, --default	Вернуть значение по умолчанию, если переменная не существует
-j, --json	Десериализовать переменную JSON
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы получить информацию о `test_variable`, выполните:

```
$ airflow variables get test_variable
```

import

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Импортирует переменные из файла JSON.

Использование

```
$ airflow variables import [-h] [-a {overwrite,fail,skip}] [-v] file
```

Аргументы

Параметр	Описание
file	Имя JSON-файла, из которого импортируются переменные
-a, --action-on-existing-key	Действие, которое следует предпринять, если переменная уже существует. Возможные варианты: `overwrite`, `fail`, `skip`. По умолчанию: `overwrite`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Чтобы импортировать переменные из файла, выполните:

```
$ airflow variables import /tmp/variables.json
```

list

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Отображает список переменных.

Использование

```
$ airflow variables list [-h] [-o table, json, yaml, plain] [-v]
```

Аргументы

Параметр	Описание
-o, --output	Формат вывода. Возможные варианты: `table`, `json`, `yaml`, `plain`. По умолчанию: `table`
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для отображения списка переменных:

```
$ airflow variables list -o table
```

set

Елена Костюченко

Содержание

Назад к содержанию

Описание

Использование

└ Аргументы

Примеры

Описание

Создает переменную.

Использование

```
$ airflow variables set [-h] [--description DESCRIPTION] [-j] [-v] key VALUE
```

Аргументы

Параметр	Описание
key	Ключ переменной
VALUE	Значение переменной
--description	Описание переменной (необязательно)
-j, --json	Сериализовать переменную JSON
-h, --help	Вывести справку для команды
-v, --verbose	Более подробный вывод команды

Примеры

Пример команды для создания тестовой переменной:

```
$ airflow variables set x 1
```

Пример вывода:

```
Variable x created
```

Релизы ADO

Елена Костюченко

Содержание

Назад к содержанию

2.10.5.1
2.6.3.2
2.6.3.1

2.10.5.1

Дата: 25.03.2026

Обновления

Версия Airflow обновлена до 2.10.5_arenadata1

Версия Monitoring обновлена до 1.3

Версия duckdb обновлена до 1.4.4

Новые функции

Добавлена поддержка интеграции с сервисами, требующими доступа клиентов к воркерам (например, ADH Spark), через концепцию общих хостов

Добавлена поддержка строк подключения к HA-кластерам Metastore

Добавлена поддержка прокси-сервера для репозиториев пакетов

Реализована поддержка ограничений при управлении Python-зависимостями

Вкладка ***Info*** теперь учитывает пользовательские настройки конфигурации

Добавлена поддержка AltLinux 10 SP

Добавлена поддержка Ansible 2.16

Реализовано автоматическое удаление артефактов проверки состояния при изменениях конфигурации кластера (распределение компонентов между хостами) для предотвращения ложных срабатываний

Исправленные баги

Airflow и Redis больше не завершают работу с ошибкой при запуске на новых хостах во время переноса компонентов в режиме обслуживания (maintenance mode)

Исправлена ошибка выполнения действия ***Sync requirements*** на хостах с AltLinux

Прочее

Минимальная версия ADCM — 2.9.0

2.6.3.2

Дата: 22.07.2025

Обновления

Prometheus 1.2_arenadata1

Новые функции

Реализована поддержка режима обслуживания (maintenance mode) для сервисов

Добавлена возможность управлять Python-зависимостями через ADCM

Добавлена поддержка Astra Linux 1.7.6 "Воронеж" SE

Улучшения

Действие ***Manage SSL*** теперь позволяет управлять параметрами мониторинга

Конфигурацию мониторинга теперь можно задавать гибко с помощью файлов `scrape_config_files`

Выполнен внутренний рефакторинг для оптимизации и безопасности

Исправленные баги

Исправлена ошибка при переключении Flower на HTTPS после включения SSL

Добавлена отсутствующая проверка группы `airflow.flower` в шаблоне *flower_scraper.yml*

Прочее

Минимальная версия ADCM — 2.6.0

Переработаны пакет и бандл Airflow для добавления возможности отката версии ADO. Эта возможность станет доступной в следующем релизе

2.6.3.1

Дата: 18.03.2025

Новые сервисы

Redis теперь является кластерным сервисом для простоты настройки и развертывания

Реализован новый сервис мониторинга на основе Prometheus и Grafana

Добавлена поддержка node-exporter "из коробки"

Добавлена поддержка statsd-exporter "из коробки"

Добавлены панели мониторинга в Grafana, которые отображают метрики хостов, метрики кластера Airflow (серверы и рабочие процессы), запущенные DAG и очереди Celery

Новые функции

DAG Processor и Triggerer разделены на отдельные компоненты Airflow для гранулярной настройки кластера

Добавлены новые действия для сервиса Airflow в ADCM:

- ***Sync permission***
- ***Rotate fernet key***

Конфигурация компонента Flower стала более подробной и предлагает больше опций

Добавлена возможность нативной настройки бэкенда для хранения конфиденциальных данных с помощью интеграции с HashiCorp Vault

Ссылка на веб-интерфейс Flower теперь отображается в информационном блоке сервиса

Улучшения

Конфигурация сервиса теперь явно отображает все настройки с их описаниями. Дополнительные параметры можно добавить через кастомные опции

Улучшена обработка конфиденциальной информации:

- Все конфиденциальные данные (ключи, пароли, токены и т. д.) имеют тип "секрет" (secret) и не отображаются явно
- Все секреты генерируются — `fernet_key`, `secret_key` и т. д.

Введено единое действие для изменения топологии сервиса (добавление, удаление, перемещение компонентов и т. д.). Вы можете запустить его как для сервиса, так и для всего кластера

Исправленные баги

Исправлены ошибки, возникавшие при анализе конфигурации Airflow со специальными символами в `secret_key`

Исправлена ошибка в конфигурации ulimits

Прочее

Минимальная версия ADCM — 2.5.0

Поддерживаемые сервисы

Елена Костюченко

Назад к содержанию

В этой статье описывается список сервисов, включенных в текущую версию ADO. Столбец ***Версия*** содержит ссылки на release notes каждого компонента.

Сервис	Компонент	Версия
Apache Airflow2	Airflow DAG Processor	2.10.5_arenadata1
	Airflow Flower	2.10.5_arenadata1
	Airflow Scheduler	2.10.5_arenadata1
	Airflow Server	2.10.5_arenadata1
	Airflow Triggerer	2.10.5_arenadata1
	Airflow Worker	2.10.5_arenadata1
Redis	Redis Server	6.2.7
Arenadata Postgres	Arenadata PostgreSQL	16
Arenadata Monitoring	Grafana	1.3
	Prometheus	1.3
	Node Exporter	1.3