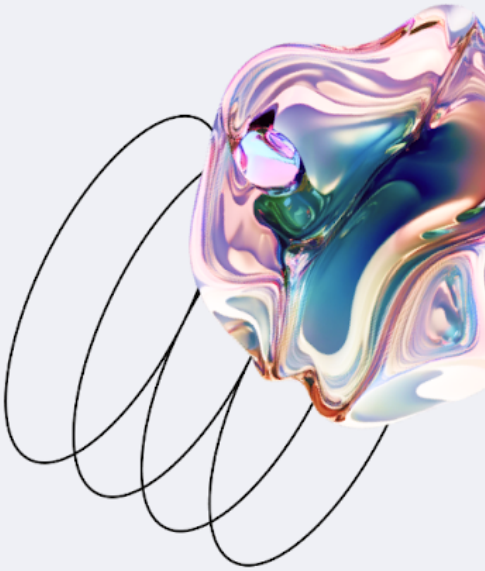
GeekBrains



Обзор возможностей Airflow, установка и настройка

ETL: автоматизация подготовки данных

Оглавление

Введение 4

Системы управления данными. Apache Airflow и его особенности 4

- Три проблемы дата-инженеров 5
  - Неясная стратегия 5
  - Интеграция разных систем 5
  - Обслуживание конвейера данных 6
- Зачем внедрять систему управления рабочим процессом 6
  - Визуализация бизнес-процесса 7
  - Автоматизация 7
  - Масштабируемость 7
  - Мониторинг 8
- Что такое Apache Airflow 8
- Уникальность Airflow 9
  - DAG 9
  - Задачи 10
  - Операторы 11
  - Датчики 12
  - Администрирование 12
  - Соединения 12
  - Переменные 13
- Apache Airflow: три истории из жизни 13
- Кому не подойдет Airflow 13

Установка и настройка Apache Airflow 14

Установка из исходников 14

Установка с помощью PyPI 15

Установка с помощью docker images 16

Пример: установка с помощью PyPI 17

Заключение 20

Введение

Всем привет! Сегодня наша пятая лекция курса «ETL: автоматизация подготовки данных».

На прошлом уроке мы рассмотрели способы партицирования данных и разобрались, как они реализуются на практике. Узнали, какие типы партицирования существуют и как их имплементировать. Погрузились в способы горизонтального партицирования на примере PostgreSQL.

Сегодня мы узнаем, что такое системы управления данными и зачем они нужны. Рассмотрим особенности Apache Airflow и основные элементы проектирования системы управления данными на его основе. Разберем способы установки Apache Airflow, их плюсы и минусы. Результатом урока будет локальная установка Airflow на ваш компьютер.

Системы управления данными. Apache Airflow и его особенности

За последние 15 лет в мире ETL многие компании перешли от картотек к кластерам хранилищ данных, чтобы справиться с их взрывным ростом. Объем данных огромен, и это мешает многим инженерам правильно разрабатывать и проектировать пайплайны для их обработки.

Наряду с данными, растет и количество инструментов. Руководителям и инженерам приходится решать, какие из них выбрать для своих корпоративных архитектур. Часто инструментов слишком много — значительная часть из них нужна редко, функционал между платформами данных и приложениями может пересекаться.

Наряду с этим разнообразием, есть еще одна проблема — разрыв между инструментами: например, Spark, Kubernetes, Bash- и Python-скриптами, а также собственными приложениями. Решив эти проблемы, инженеры данных смогут лучше ориентироваться в распределенных экосистемах и архитектурах организаций, а также улучшат свои продукты.

На уроке подробнее рассмотрим проблемы, с которыми сталкиваются дата-инженеры, и узнаем, как интеграция с Apache Airflow поможет их решить.

Три проблемы дата-инженеров

Алгоритм машинного обучения эффективен настолько, насколько эффективны данные, которые попадают на вход. Задача дата-инженеров — создать среду, в которой данные будут полезными для бизнеса.

В тестировании продукта есть стратегия shift left, когда продукт начинают тестировать уже на ранних этапах жизненного цикла разработки ПО. Она полезна и в обработке данных: если уделять внимание процессу и решать проблемы на ранних этапах, разработка пройдет быстрее, а результат будет более качественным.

Ниже обсудим три проблемы, с которыми сталкивается бизнес и дата-инженеры.

Неясная стратегия

Большие данные льются в ИТ-системы, как вода в руки. Мало кто знает, что с ними делать, так что большая их часть остается неиспользованной.

В разработке данных без стратегии не обойтись, иначе дата-инженеры будут страдать от перегрузки инструментами: для каждого шага будет отдельный, их функции будут пересекаться.

Это противоречит принципу Agile: «Люди и взаимодействие важнее процессов и инструментов». Команда дата-инженеров должна учитывать этот принцип и внимательно выбирать инструменты, которые будут полезны для задач компании.

Вспомним закон Конвея: «Организации, разрабатывающие системы, вынуждены создавать проекты, которые являются копиями коммуникационных структур этих организаций». Без ясной стратегии дата-инженеры могут выбрать инструменты со множеством функций, но результат их работы будет неэффективным для бизнеса.

Интеграция разных систем

Инженеры используют Spark, Kafka, BigQuery, Redshift, Tableau, Snowflake и многие другие инструменты. Следовательно, должна быть общая среда, чтобы консолидировать процесс. А для этой среды нужен язык, который поможет дата-инженерам установить нужные связи и создать подходящий конвейер данных.

Обслуживание конвейера данных

Хорошая практика при разработке конвейера данных — на ранних этапах задуматься о правильном обслуживании и долговечности.

Если конвейер состоит из череды разбросанных скриптов, файлов оболочки и множества встроенных команд, обслуживать его непросто. А вот удобочитаемые и повторяемые элементы, наоборот, облегчат процесс.

Повторяемость полезна в долгосрочной перспективе — она поможет избежать серьезной перестройки архитектуры или изменений в конвейере. Обслуживание обеспечит предсказуемость и гибкость одновременно — предсказуемость, потому что данные будут работать как и должны, гибкость, потому что конвейер сможет справляться с растущими потребностями в данных.

Решить эти проблемы поможет система управления рабочими процессами.

Зачем внедрять систему управления рабочим процессом

Рабочий процесс состоит из множества этапов: от сбора, обработки и перемещения данных до получения из них ценных сведений. Каждый из этих этапов вносит свой вклад в достижение бизнес-результатов. Поэтому важно, чтобы у ИТ-специалистов была возможность изучить отношения между этапами, усовершенствовать их и разработать подходящие конвейеры данных.

Если есть структурированная система работы с данными, проще найти и устранить ее слабые места. Системы управления рабочими процессами — это то, что помогает командам инженеров данных организовать надежную структуру анализа и оценки полезности данных и бизнес-показателей.

Рассмотрим четыре причины внедрить системы управления рабочим процессом.

Визуализация бизнес-процесса

Визуализация — мощный элемент системы управления рабочим процессом.

Технические термины дата-инженеров могут быть непонятны другим. А визуализация поможет лучше разобраться в компонентах конвейера данных и подсветит ценную информацию.

Компании используют Miro, LucidChart, Diagrams.net и другие подобные инструменты, чтобы визуализировать бизнес-процессы и извлечь результаты, которые иначе были бы невидимы через распределенные системы и команды. Такие возможности есть и у систем управления рабочим процессом

Визуализация дает подробную картину и помогает задавать правильные вопросы, чтобы получать правильные ответы.

Автоматизация

Автоматизация расширяет возможности дата-инженеров и делает их работу более гладкой. Исследования Harvard Business Review и Lean Enterprises показали, что улучшение опыта разработчиков и инженеров за счет автоматизации позволяет командам работать более продуктивно. У автоматизации есть не только технические плюсы: благодаря ей у инженеров появляется больше мотивации для решения проблем.

Когда конвейер данных может запускаться автоматически на основе набора настраиваемых условий, команда инженеров может сосредоточиться на более важных проблемах. Ручное вмешательство вместо автоматизации станет исключением. Такой подход поможет избежать многих ошибок.

Масштабируемость

Одно из следствий автоматизации — рабочие процессы становятся более предсказуемыми, поэтому масштабировать их проще.

Это значит, что системы управления рабочими процессами будут иметь стандартизированное поведение и могут быть включены в экосистемы данных разного размера. Эффективный и надежный конвейер данных для 10 клиентов можно будет масштабировать для 100.

Кроме того, благодаря масштабируемости можно встраивать компоненты в пайплайн, тем самым расширяя их возможности, а не ограничиваться только возможностью самого компонента.

Когда системы управления рабочими процессами контролируют ваш конвейер данных, вы можете чаще получать один и тот же результат, независимо от того, кто или что выполняет процесс.

Мониторинг

Системы управления рабочими процессами позволяют инженерам улучшать процессы за счет мониторинга. Система, которая не контролируется, — это система, которую нельзя улучшить.

Если вы хотите улучшить свой бизнес-процесс, в вашем конвейере данных должна быть возможность мониторинга. Одна возможность улучшения, найденная в ходе мониторинга, может стать решающим фактором в оценке производительности вашего пайплайна.

Мониторинг поможет оценить производительность конвейера. Это важно, потому что для бизнес-процессов часто существуют соглашения об уровне обслуживания (SLA). Мониторинг также помогает позаботиться об аудите, что может убедительно позиционировать группу разработки данных как участника усилий по обеспечению соответствия.

Теперь, когда мы изучили возможности визуализации, автоматизации, масштабируемости и мониторинга систем управления рабочими процессами, давайте познакомимся с одним из самых мощных инструментов в этой области.

Что такое Apache Airflow

Airflow — это платформа с открытым исходным кодом для разработки, планирования и мониторинга рабочих процессов. Эти рабочие процессы могут помочь вам перемещать данные, фильтровать их наборы, применять политики данных, манипулировать, отслеживать и даже вызывать микросервисы для запуска задач управления базой данных.

Преимущество Airflow по сравнению с другими похожими инструментами в том, что Airflow позволяет планировать и отслеживать рабочие процессы, а не просто создавать их.

Особенности Airflow:

- **Динамический.** DAG написаны на Python, что позволяет создавать динамические конвейеры.
- **Расширяемый.** Легко создавать собственные операторы, исполнители и библиотеки.
- **Элегантный.** DAG Airflow компактны и понятны. Параметризовать конвейеры можно с помощью шаблонов Jinja.
- **Масштабируемый.** Airflow модульный, использует очередь сообщений для управления исполнителями, которые выполняют задачи.

Преимущества Airflow

- **Открытый исходный код.** Следствие — низкая стоимость, инновации и поддержка сообщества.
- **Широкая интеграция.** Может использоваться в облачных провайдерах большой тройки — AWS, Azure и GCP.
- **Пользовательский интерфейс.** Позволяет пользователям легко отслеживать конвейеры и устранять неполадки.
- **Программный подход.** Используя Python, пользователи могут создавать инновационные расширения.
- **Повторы.** Многие конвейеры данных должны быть настроены для повторных попыток, в Airflow это встроено.
- **Предупреждение.** Пайплайны в Airflow могут быть на виду, даже когда системы не мониторятся.

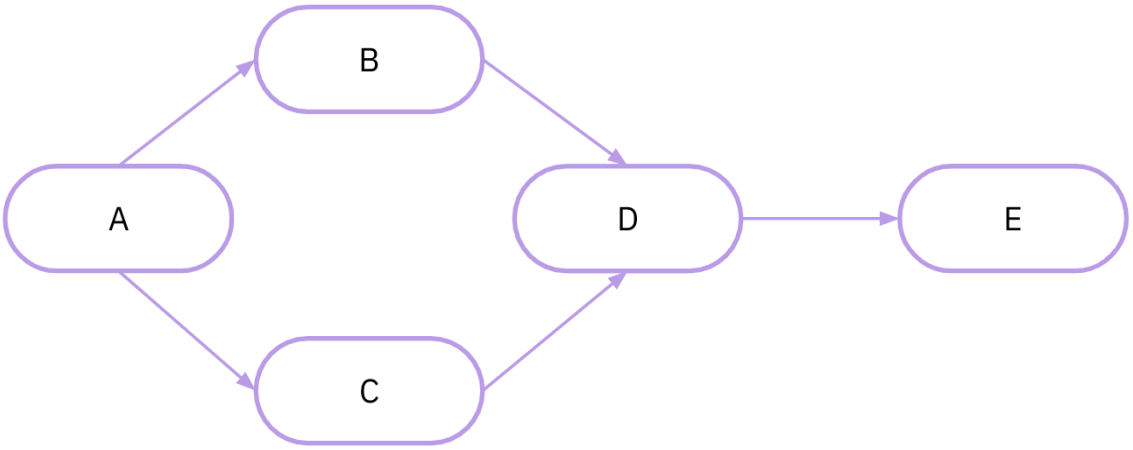
Уникальность Airflow

Рассмотрим уникальные компоненты Airflow, которые помогут вывести вашу стратегию конвейера данных на новый уровень.

DAG

DAG — это направленный ациклический граф, основа концепции Airflow. DAG состоит из узлов, соединенных направленными ребрами

Пример:



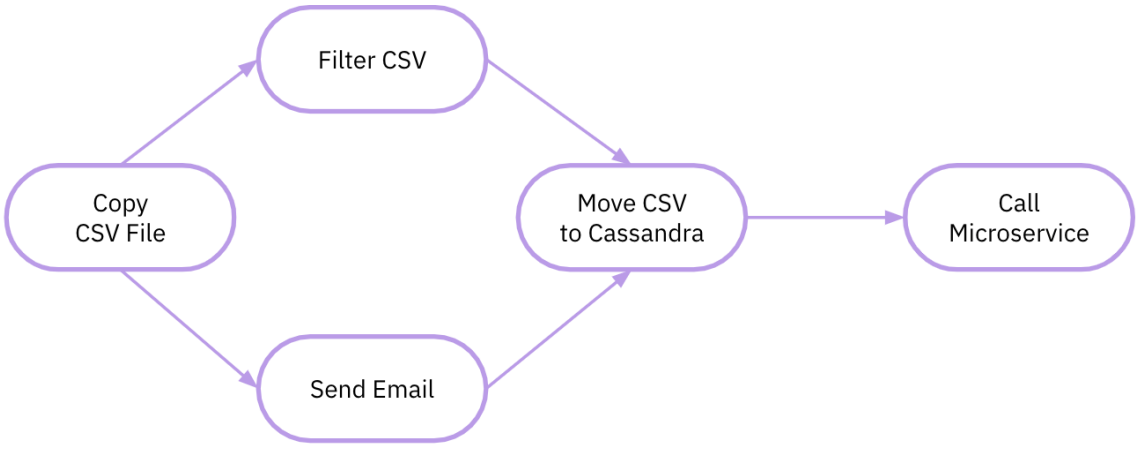
Направленный ациклический граф (DAG)

Одна из самых мощных особенностей Airflow в том, что DAG написаны на Python. Этот факт помогает нам решить одну из самых больших проблем, с которыми сталкиваются инженеры по данным, — интеграцию различных систем.

Задачи

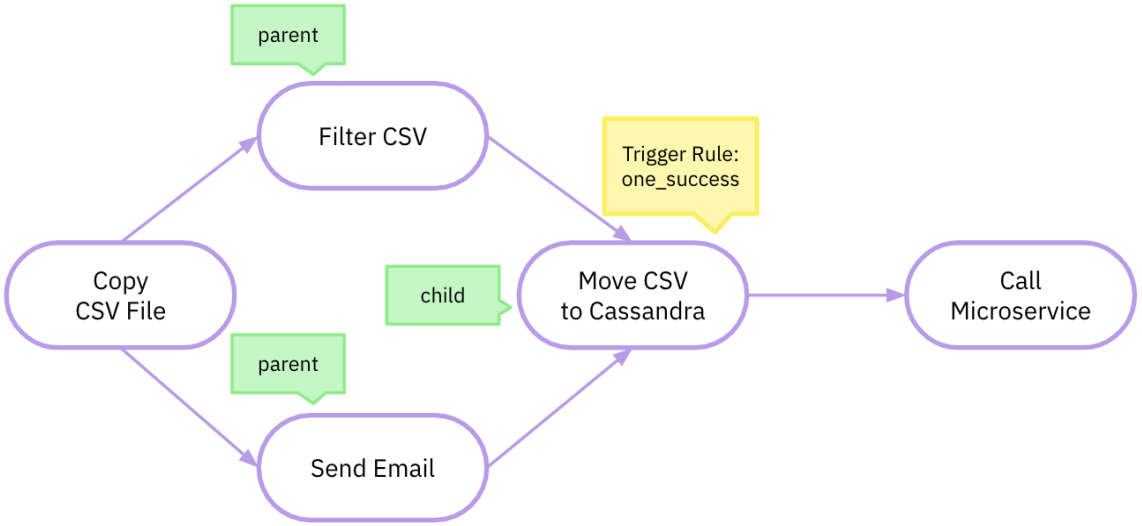
Задачи похожи на события в рабочем процессе. Они могут быть зависимыми от других или параллельными, могут передавать информацию между собой.

На диаграмме ниже мы заменили литеры конкретными событиями, которые могут происходить в конвейере данных:



Пример задач

В любой момент у всех задачи в Airflow будет состояние. **Задачи меняют свое состояние в зависимости от правила триггера задачи**. Если вы не хотите, чтобы задача запускалась только при условии успешного выполнения вышестоящей задачи, вы можете установить правило запуска.



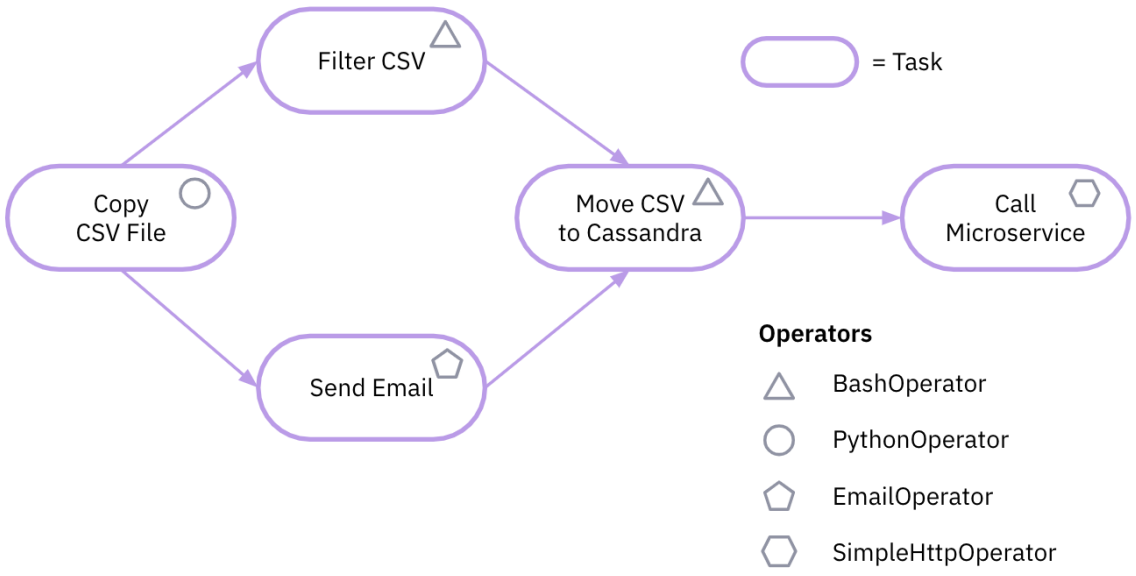
Пример триггерных правил при запуске

Операторы

Операторы позволяют выполнять разные типы задач с разной функциональностью.

Популярные операторы:

- **BashOperator** — выполняет команду bash;
- **PythonOperator** — вызывает функцию Python, определенную пользователем;
- **EmailOperator** — отправляет электронное письмо;
- **SimpleHttpOperator** — вызывает конечную точку в системе HTTP (POST, GET и так далее).



Пример операторов

Датчики

Датчики — это подмножество операторов. Они ждут, пока произойдет определенное действие, например, копирование файла или завершение рабочей нагрузки в Kubernetes.

Совместная работа операторов и датчиков позволяет внедрить передовой опыт в конвейер данных за счет проверки и предотвращения потери данных.

Администрирование

У пользовательского интерфейса Airflow есть набор опций, одна из центральных — «Администратор».

В разделе «Администратор» можно настроить конфигурации для предоставления определенной информации в группах DAG.

Соединения

Соединения — это объекты для хранения учетных данных и другой информации, нужной для подключения к внешним службам. Администраторы могут настраивать подключение к БД, конечным точкам HTTP, FTP-серверам и так далее.

Переменные

Переменные — это общий способ хранения и извлечения контента или настроек в виде простого хранилища пары ключ-значение в Airflow.

В Airflow есть встроенная защита при использовании переменных. Они автоматически шифруются с помощью Fernet, платформа предоставляет возможность использования бэкенда Secrets для дополнительной безопасности.

Apache Airflow: три истории из жизни

Крупные компании используют REST API для разных целей, включая ETL, MLOps, планирование рабочих процессов и обработку данных.

Adobe использовала Airflow, чтобы улучшить свою платформу взаимодействия. Отличительной чертой стратегии Adobe в отношении данных было то, что Airflow мог выполнять и отслеживать задания Spark.

Dish — американский телевизионный провайдер и владелец спутника прямого вещания. Компании нужна была помощь, чтобы преодолеть ограничения ресурсов и повторных попыток с их cronjobs. Dish внедрил Airflow, чтобы решить проблемы планирования заданий и сократить задержки с часов до минут.

BigFish — игровая компания, которой нужно было разработать платформу ETL для управления аналитическими рабочими процессами. Они пытались использовать Apache Oozie, но без успеха из-за повторных попыток неудачных рабочих нагрузок. Airflow помог им перенести свои рабочие нагрузки ETL на Google Cloud Platform (GCP).

Кому не подойдет Airflow

Airflow создан для ограниченных пакетных рабочих процессов. Хотя CLI и REST API позволяют запускать рабочие процессы, Airflow не был создан для бесконечно работающих рабочих процессов на основе событий.

Airflow — не потоковое решение. Однако потоковая система, такая как Apache Kafka, часто работает вместе с Apache Airflow. Kafka можно использовать для приема и обработки в режиме реального времени, данные о событиях записываются в место хранения, а Airflow периодически запускает рабочий процесс, обрабатывающий пакет данных.

Если вы предпочитаете использовать графические интерфейсы вместо кода, Airflow, вероятно, не станет правильным решением. Веб-интерфейс Airflow призван максимально упростить управление рабочими процессами, а инфраструктура постоянно совершенствуется, чтобы сделать работу разработчиков максимально удобной. Однако философия Airflow в том, чтобы определять рабочие процессы как код, поэтому кодинг всегда будет нужен.

Установка и настройка Apache Airflow

Чтобы начать работу с Apache Airflow, нужно развернуть его на устройстве. Есть несколько способов сделать это. Рассмотрим их подробнее.

Установка из исходников

Когда выбрать этот вариант:

- Если вы планируете собирать все свое ПО из исходников.
- Apache Airflow — один из проектов Apache Software Foundation. Для всех проектов ASF требуется, чтобы они могли быть установлены с использованием официальных исходных кодов, опубликованных через официальный центр Apache Downloads.
- Если вам нужно проверить целостность и происхождение ПО.

Кому подойдет:

- Пользователям, знакомым с установкой и сборкой ПО из исходных кодов, осознающим целостность и происхождение используемого ПО вплоть до самого низкого уровня.

Что вы должны уметь:

- Ожидается, что вы сами создадите и установите Airflow и его компоненты.
- Вы должны разработать и выполнить развертывание для всех компонентов Airflow.
- Вы отвечаете за настройку базы данных, создание и управление схемой базы данных с помощью команд, автоматический запуск и восстановление, обслуживание, очистку и обновление Airflow.

Как поможет Apache Airflow Community:

- Есть инструкции по сборке программного обеспечения. Но из-за разных сред и инструментов могут возникнуть проблемы, характерные для вашего развертывания и среды. Их придется решать самостоятельно.

Установка с помощью PyPI

Когда выбрать этот вариант:

- Если вы не знакомы с контейнерами и Docker и хотите установить Apache Airflow на физические или виртуальные машины, при этом вы привыкли устанавливать и запускать ПО с использованием пользовательского механизма развертывания.
- Единственный официально поддерживаемый механизм установки — pip. Файлы ограничений управляются менеджерами пакетов Apache Airflow, чтобы убедиться, что вы можете повторно установить Airflow из PyPI со всеми необходимыми зависимостями. Целостность установки гарантируется создателем пакета.
- В случае установки PyPI вы также можете проверить целостность и происхождение пакетов, загруженных из PyPI, как описано на странице установки. Но ПО, которое вы загружаете из PyPI, предварительно создано для вас, вы можете установить его без сборки, собирать ПО из исходников не придется.

Кому подойдет:

- Пользователям, знакомым с установкой и настройкой приложений Python, с управлением средами Python, зависимостями и запуском ПО с помощью собственных механизмов развертывания.

Что вы должны уметь:

- Ожидается, что вы установите Airflow (все его компоненты) самостоятельно.
- Вы должны разработать и выполнить развертывание для всех компонентов Airflow.
- Вы отвечаете за настройку базы данных, создание и управление схемой базы данных с помощью команд, за автоматический запуск и восстановление, обслуживание, очистку и обновления Airflow и Airflow Providers.

Как поможет Apache Airflow Community:

- Есть инструкция по установке из PyPI. Но из-за разных сред и инструментов могут возникнуть проблемы, характерные для вашего развертывания и среды. Их придется решать самостоятельно.
- Есть инструкция быстрого старта с примерами локального запуска Airflow. Их можно использовать для быстрого запуска Airflow, локального тестирования и разработки. Однако это всего лишь примеры. Не ожидайте, что этот docker-compose готов к установке в продакшн, вам нужно создать собственное решение.

Установка с помощью Docker Images

Когда выбрать этот вариант:

- Если вы знакомы со стеком Container/Docker. Этот метод обеспечивает возможность запуска компонентов Airflow изолированно от других программ, работающих на тех же физических или виртуальных машинах с простым указанием зависимостей.
- Образы создаются менеджерами пакетов Apache Airflow, используются официально выпущенные пакеты из PyPI и официальные файлы ограничений — те же, что и для установки Airflow из PyPI.

Кому подойдет:

- Пользователям, знакомым с контейнерами и стеком Docker. Нужно понимать, как создавать собственные образы контейнеров.
- Пользователям, которые понимают, как устанавливать провайдеры и зависимости из PyPI с ограничениями, если они хотят расширить или настроить образ.
- Пользователям, которые знают, как создавать развертывания с помощью Docker, связывая вместе несколько контейнеров Docker и поддерживая такие развертывания.

Что вы должны уметь:

- Ожидается, что вы сможете настраивать или расширять образы Container/Docker. Если хотите добавить дополнительные зависимости, вы

должны собрать развертывание из нескольких контейнеров (например, с помощью docker-compose) и убедиться, что они связаны друг с другом.

- Вы отвечаете за настройку базы данных, создание и управление схемой базы данных с помощью команд, автоматический запуск и восстановление, обслуживание, очистку и обновление Airflow и Airflow Providers.
- Вы несете ответственность за управление собственными настройками и расширениями для ваших пользовательских зависимостей. Airflow и Airflow Providers — часть базового образа. Учитывайте, что появляются новые версии от сообщества, и не забывайте обновлять базовый образ при сборке своего контейнера.
- Вы должны выбрать правильный механизм развертывания. Есть ряд доступных вариантов развертывания контейнеров. Вы можете использовать собственный настраиваемый механизм, настраиваемые развертывания Kubernetes, Docker Compose, диаграммы Helm и так далее. Выбирайте на основе своего опыта и ожиданий.

Так как Apache Airflow написан на Python и основной метод установки, рекомендуемый сообществом, — установка с помощью pip, рассмотрим его подробнее.

Пример: установка с помощью PyPI

Установка Airflow может быть сложной, потому что Airflow — это и библиотека, и приложение. Обычно библиотеки держат свои зависимости открытыми, а приложения их закрывают. Сообщество решило оставить зависимости как можно более открытыми (в setup.cfg и setup.py), чтобы пользователи могли при необходимости устанавливать разные версии библиотек. Это значит, что время от времени команда pip install apache-airflow не будет работать или приведет к ошибке установки Airflow.

Чтобы обеспечить возможность повторной установки и обновления, разработчики также хранят набор «заведомо работающих» файлов ограничений constraints-main, constraints-2-0, constraints-2-1и так далее, а затем создают тег для каждой выпущенной версии, например, constraints-2.5.0. Таким образом, сохраняется возможность получить проверенный и работающий набор зависимостей.

Эти «заведомо работающие» ограничения относятся к основной или дополнительной версии Python. Вы можете использовать их в качестве файлов ограничений при установке Airflow из PyPI. Обратите внимание, что вы должны указать правильные версии Airflow и Python в URL-адресе.

Вы можете создать URL-адрес файла, заменив переменные в шаблоне ниже.

```
https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt
```

В данной ссылке изменяются параметры:

- AIRFLOW_VERSION — версия Airflow (например, 2.5.0) или main, 2-0 для последней версии разработки;
- PYTHON_VERSION — версия Python, например, 3.8, 3.7.

Существует также файл ограничений constraints-no-providers, который содержит только ограничения, необходимые для установки ядра Airflow. Позволяет устанавливать и обновлять Airflow отдельно и независимо от провайдеров. Также вы можете использовать тег latest — он позволит установить последнюю стабильную версию Airflow.

Итак, давайте установим Airflow. Сперва создадим новый virtualenv, чтобы не устанавливать в корень системы все библиотеки. После этого выполним следующие команды:

```
AIRFLOW_VERSION=2.5.0
PYTHON_VERSION="$(python --version | cut -d " " -f 2 | cut -d "." -f 1-2)"
# For example: 3.7
CONSTRAINT_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-no-providers-${PYTHON_VERSION}.txt"
# For example: https://raw.githubusercontent.com/apache/airflow/constraints-2.5.0/constraints-no-providers-3.7.txt
pip install "apache-airflow==${AIRFLOW_VERSION}" --constraint "${CONSTRAINT_URL}"
```

Мы указываем версию Airflow, которую хотим установить, версию Python, которую будем использовать, ссылку на файл ограничений. Выполняем установку с помощью PyPI, используя параметры, которые определили заранее.

Если после установки команда airflow не распознается системой (чаще всего это происходит на Windows при использовании WSL), нужно убедиться, что в

переменные окружения добавлен путь к установочной директории Airflow. Если нет, нужно добавить.

```
PATH=$PATH:~/.local/bin
```

Или можно вызывать Airflow с помощью Python.

```
python -m airflow
```

Часто при вызове Airflow встречается ошибка Symbol not found: _Py_GetArgcArgv. Она может означать, что вы используете несовместимую версию Python. Суть проблемы в том, что библиотека Airflow зависит от setproctitle, использует непубличный API Python, который недоступен в стандартной установке /usr/local/opt/ (которая указывает на путь под /usr/local/Cellar).

Простое решение — убедиться, что вы используете версию Python, в которой есть dylib библиотеки Python. Например:

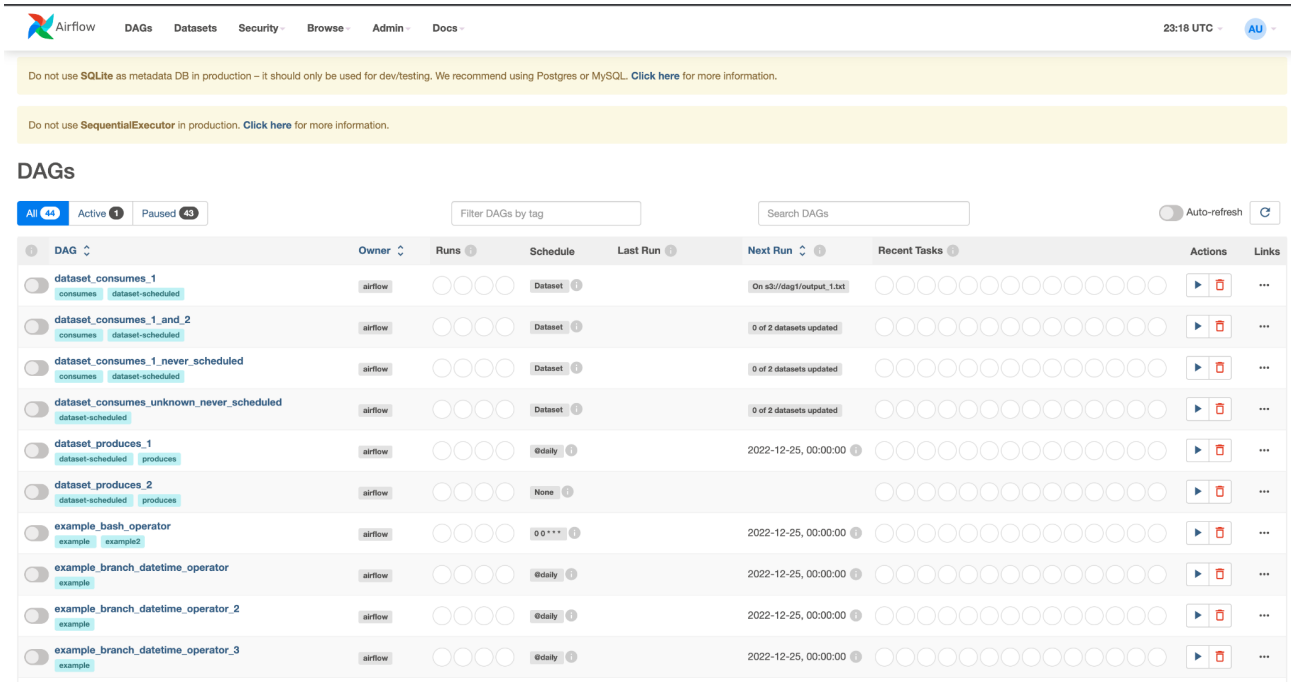
```
# Note: these instructions are for python3.7 but can be loosely modified for other versions
brew install python@3.7
virtualenv -p /usr/local/opt/python@3.7/Frameworks/Python.framework/Versions/3.7/bin/python3 .toy-venv
source .toy-venv/bin/activate
pip install apache-airflow
python
>>> import setproctitle
# Success!
```

Кроме того, вы можете загрузить и установить Python с официального сайта.

После завершения установки запустите команду

```
airflow standalone
```

Перейдите по ссылке http://localhost:8080/. В результате вы должны увидеть окно:



Заключение

Сегодня мы рассмотрели особенности Apache Airflow. Разобрали примеры его использования и составные части. В заключительной части урока узнали, какие есть способы установки Airflow, в чем плюсы и минусы каждой из опций.