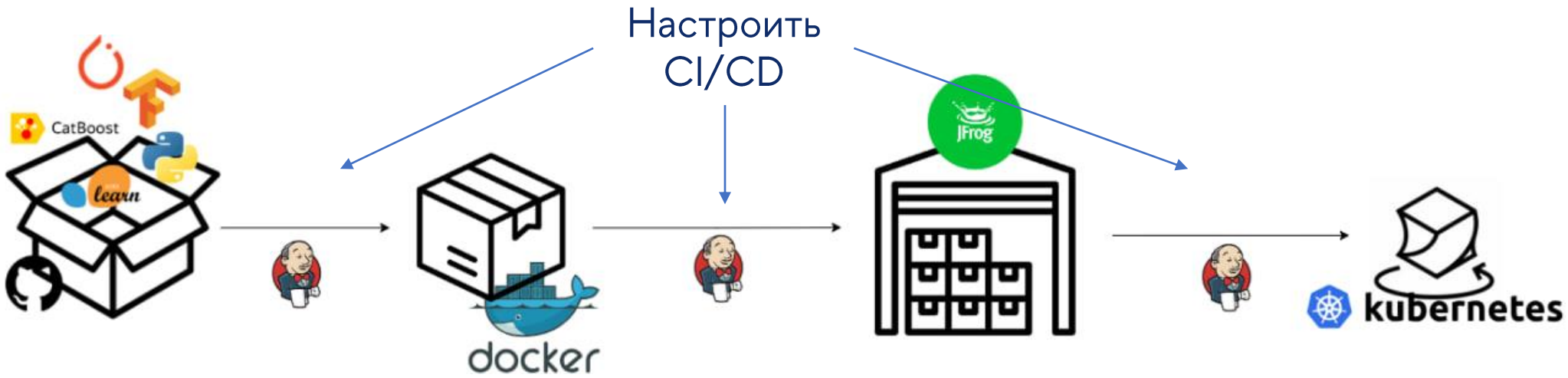
Машинное обучение и высоконагруженные системы

Москва, осень 2023

MLOps. Начало

Гаврилова Елизавета,
Senior ML-engineer

Как развернуть приложение?



Шаг1
Собрать приложение

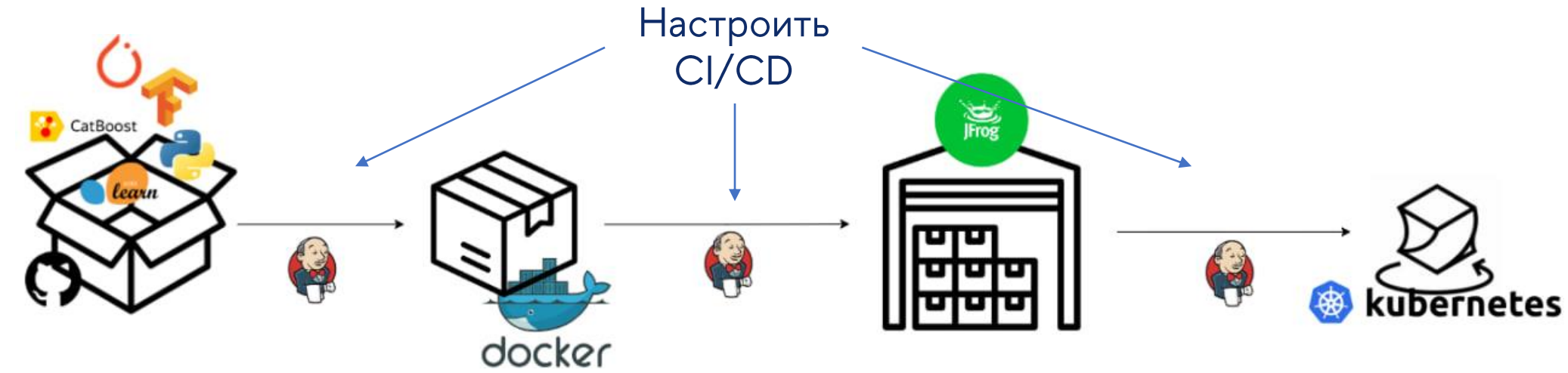
Шаг2
Упаковать в контейнер

Шаг3
Сложить образ в хранилище

Шаг4
Задеплоить

[Ссылка на обзорный вебинар]

Как развернуть приложение?



Шаг1
Собрать приложение

Шаг2
Упаковать в контейнер

Шаг3
Сложить образ в хранилище

Шаг4
Задеплоить

Об этом - на курсах «Инструменты разработки» и «Основы промышленной разработки».

Что было «до» Шага1?

Шаг1
Собрать приложение

Шаг0
Из ноутбука сделать дообучаемую модель

Прежде чем собирать и деплоить приложение, необходимо позаботиться о модели:

- о том, чтобы она обучалась на актуальных данных
- ни за что не делать этого руками☺

Программа курса

15 сентября 18:00-19:30 MLOps интро. MLOps для регулярного обучения моделей: обзор стека AirFlow+MLFlow+S3.

22 сентября 18:00-19:30 Виртуальные машины и виртуальные окружения. Сетап необходимой инфраструктуры: установка и настройка AirFlow, MLFlow, S3.

29 сентября 18:00-19:30 Что такое DAG? Мой первый DAG для обучения модели: из чего состоит пайплайн обучения?

6 октября 18:00-19:30 Обучение нескольких моделей одновременно. Чтение и хранение данных на S3.

13 октября 18:00-19:30 Установка и настройка MLFlow. Повторяем эксперимент, но теперь версионируем с помощью MLFlow.

20 октября 18:00-19:30 Обзор финального задания. Подводя итоги: сравнение кастомного версионирования с готовым решением MLFlow.

Чему я научусь

Устанавливать Linux на Windows ☺ (для win-юзеров) ✓

Поднимать виртуальное окружение ✓

Поднимать AirFlow ✓

Создавать DAG для обучения модели ✓

Поднимать MLFlow ✓

Версионировать эксперименты ✓

Подключаться к S3 ✓

Общаться с S3 ✓

15.09 Что будем обсуждать сегодня?

- Вспомним, что включает в себя **MLOps**
- Обсудим, как устроен **AirFlow**
- Поймём, как выбрать **Executor** и тип **базы метаданных** под него
- Обсудим, как устроен **MLFlow**
- Разберёмся, что такое **S3**

Всеми любимый AirFlow

Оркестраторы -
это инструменты, которые позволяют разрабатывать, планировать и осуществлять мониторинг сложных рабочих процессов.

Задачи

- Создание заданий
- Запуск заданий
- Трэкинг задач
- Настройка зависимостей
- Мониторинг

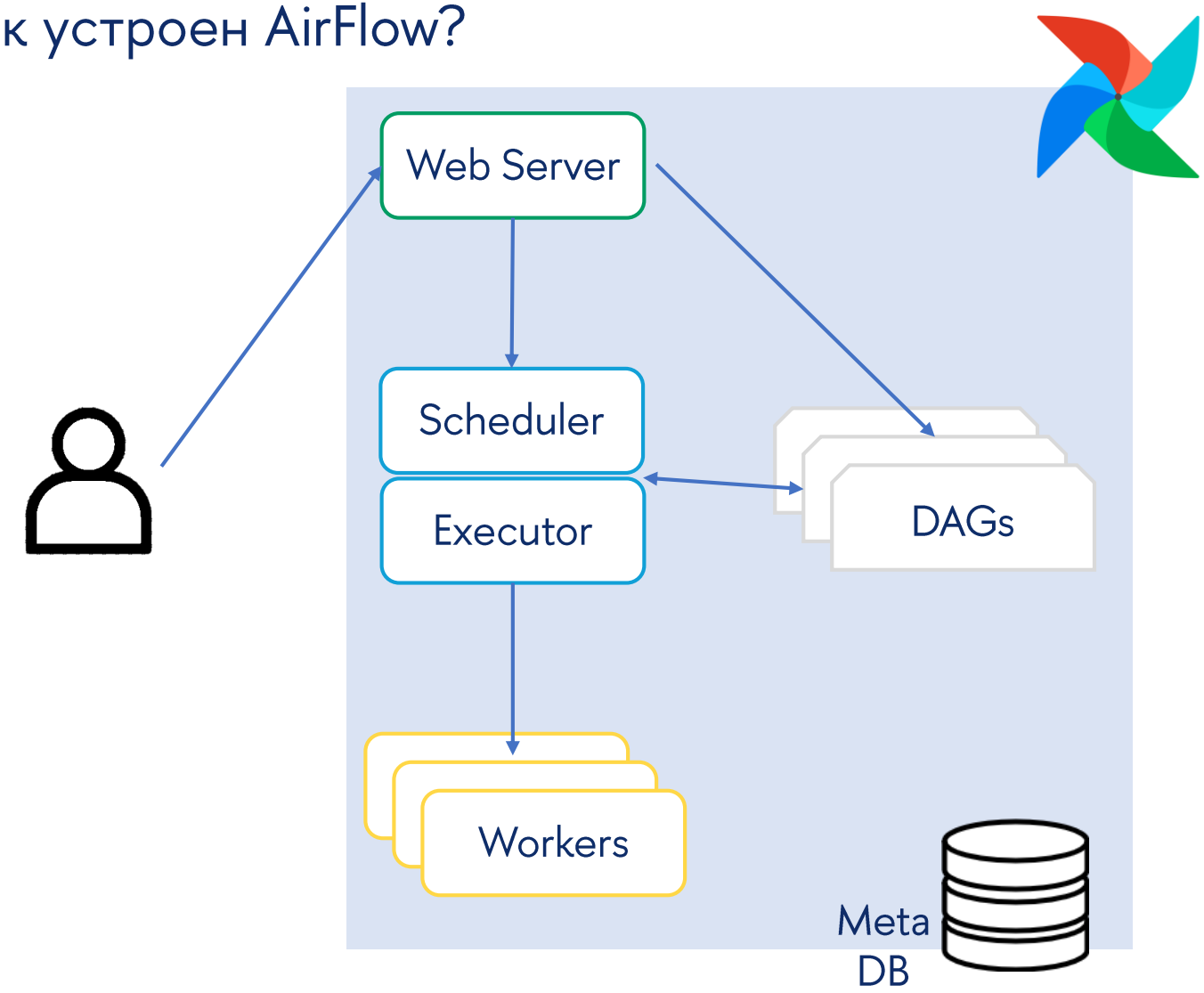
Сферы

- ML модели
- ETL процессы

AirFlow – автоматизировать всё...

Ссылка на документацию

Как устроен AirFlow?



Web Server — пользовательский интерфейс.

Scheduler (планировщик) — служба, отвечающая за планирование в Airflow. Отслеживая все задания, инициализирует их запуск. Для выполнения задач планировщик использует указанный в настройках **Executor**.

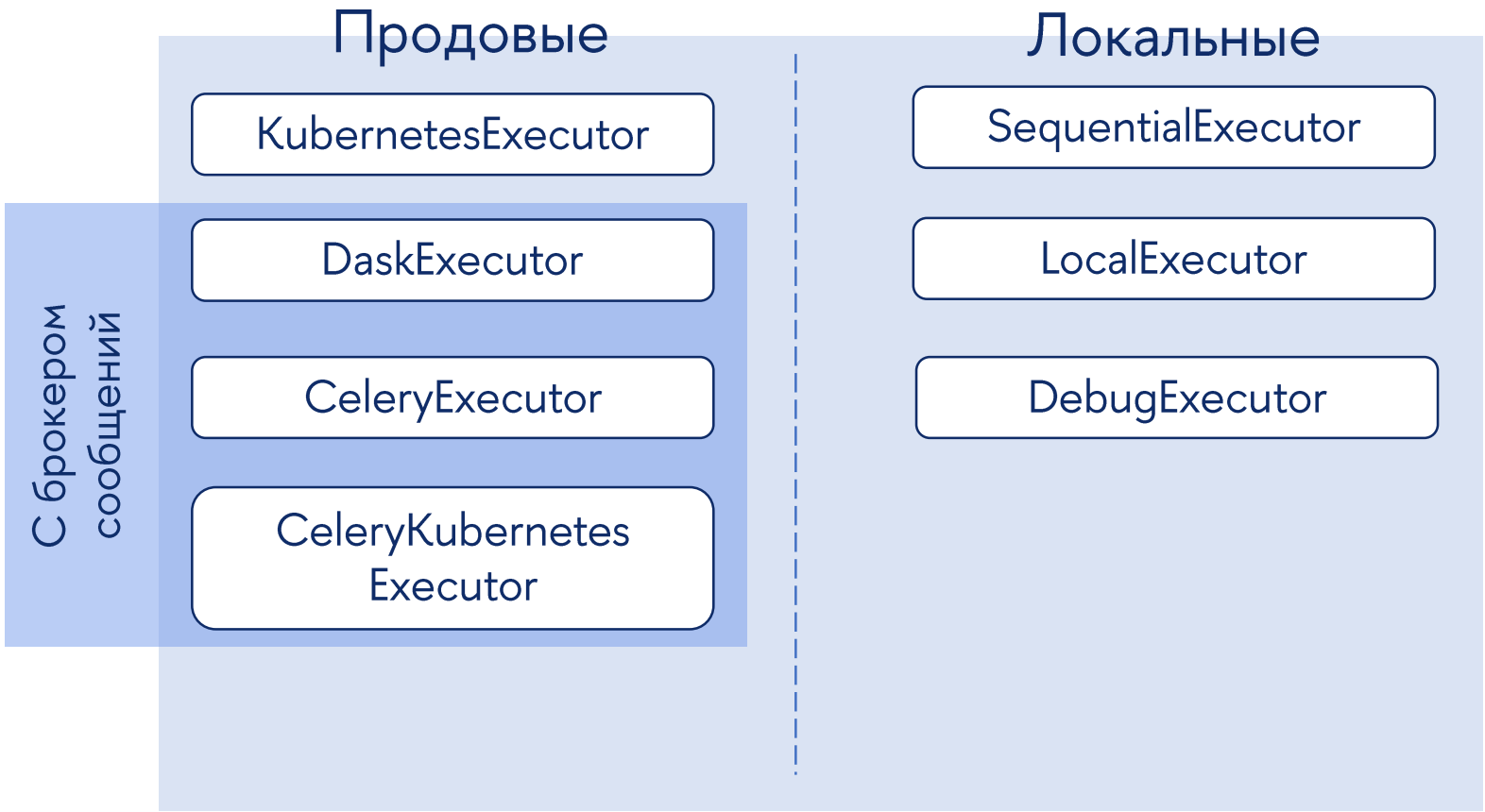
Executor (исполнитель) — механизм, с помощью которого запускаются экземпляры задач. Работает в связке с планировщиком в рамках одного процесса.

Worker (рабочий) — отдельный процесс, в котором выполняются задачи.

Metadata DB (база метаданных) — репозиторий метаданных на базе библиотеки SqlAlchemy для хранения глобальных переменных, настроек, статусов выполнения заданий и т.д.

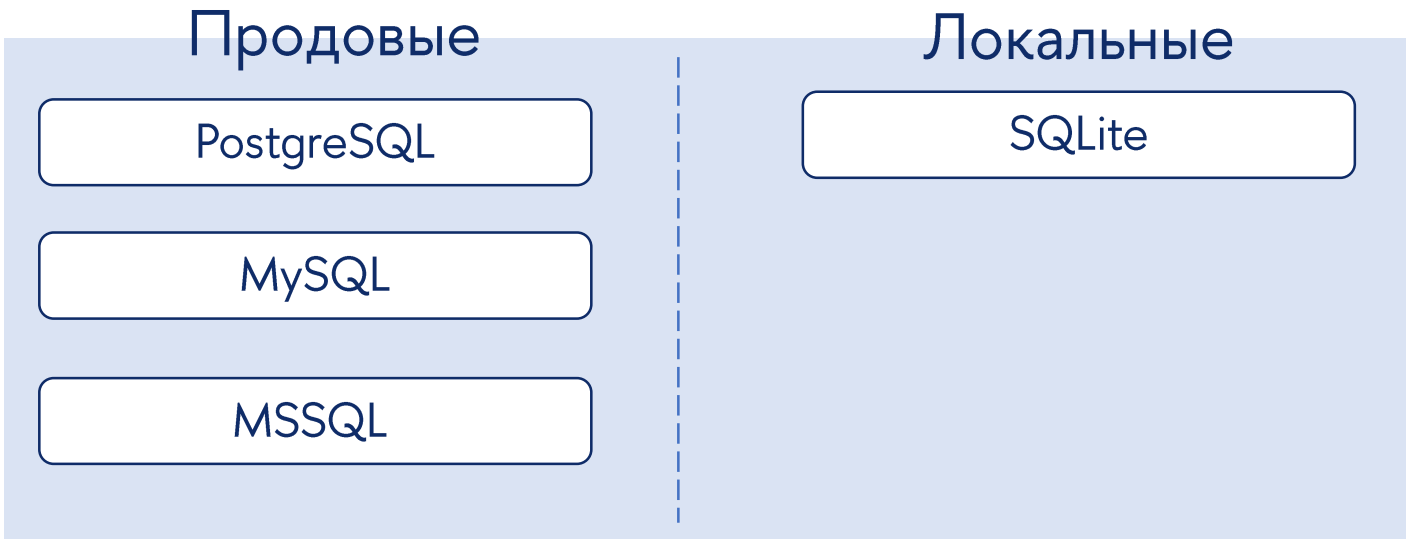
Типы исполнителя (**Executor**)

Ссылка на документацию



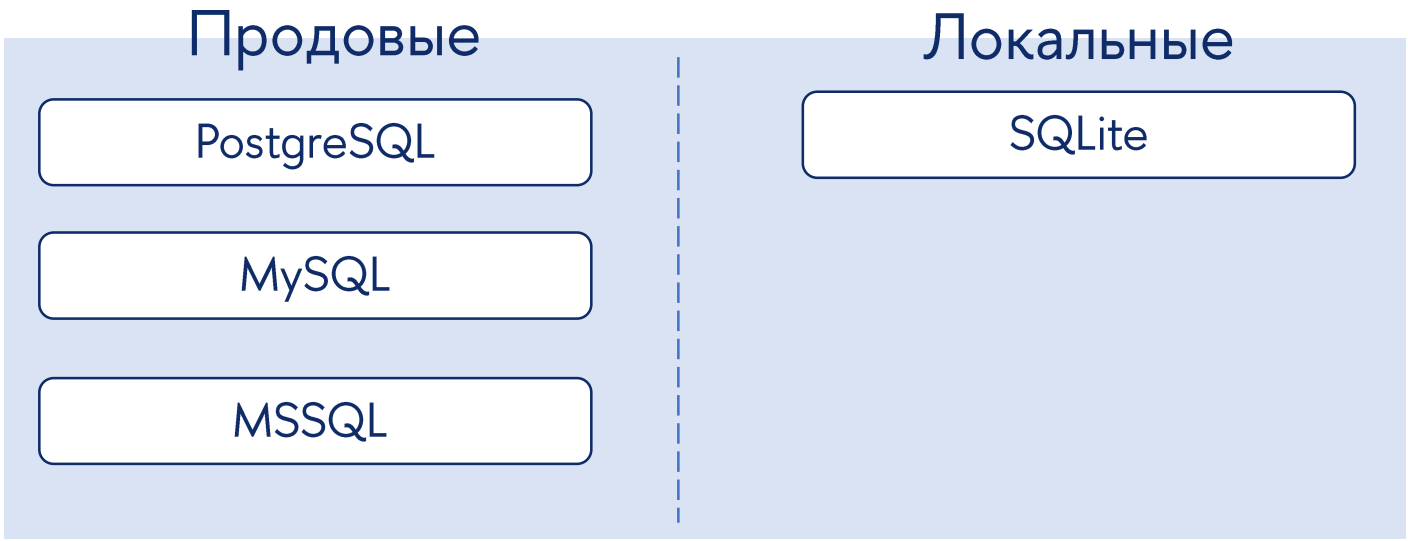
*локальные типы исполнителя не подходят для промышленной разработки из-за невозможности масштабирования и отсутствия отказоустойчивости

Типы баз метаданных (**Metadata DB**)



Ссылка на документацию

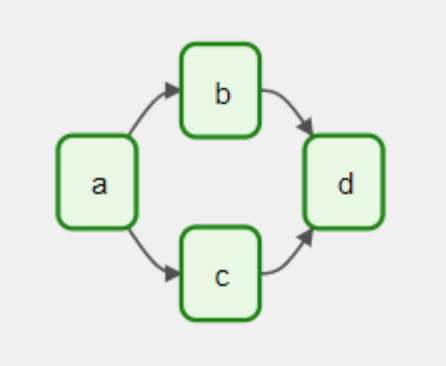
Типы баз метаданных (**Metadata DB**)

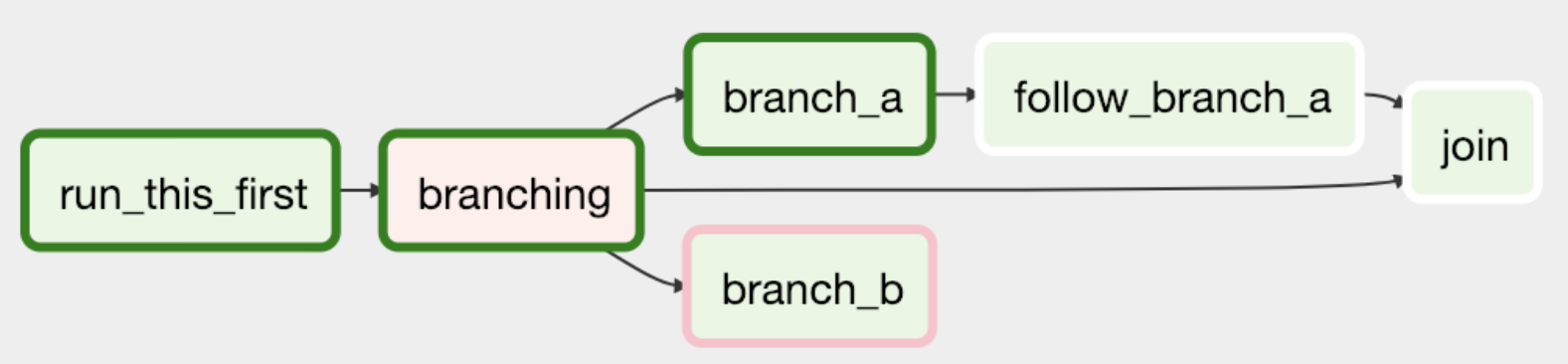


SQLite не поддерживает одновременные подключения с целью записи в БД, что не подходит для промышленного использования.

Ссылка на документацию

Концепция DAG в AirFlow

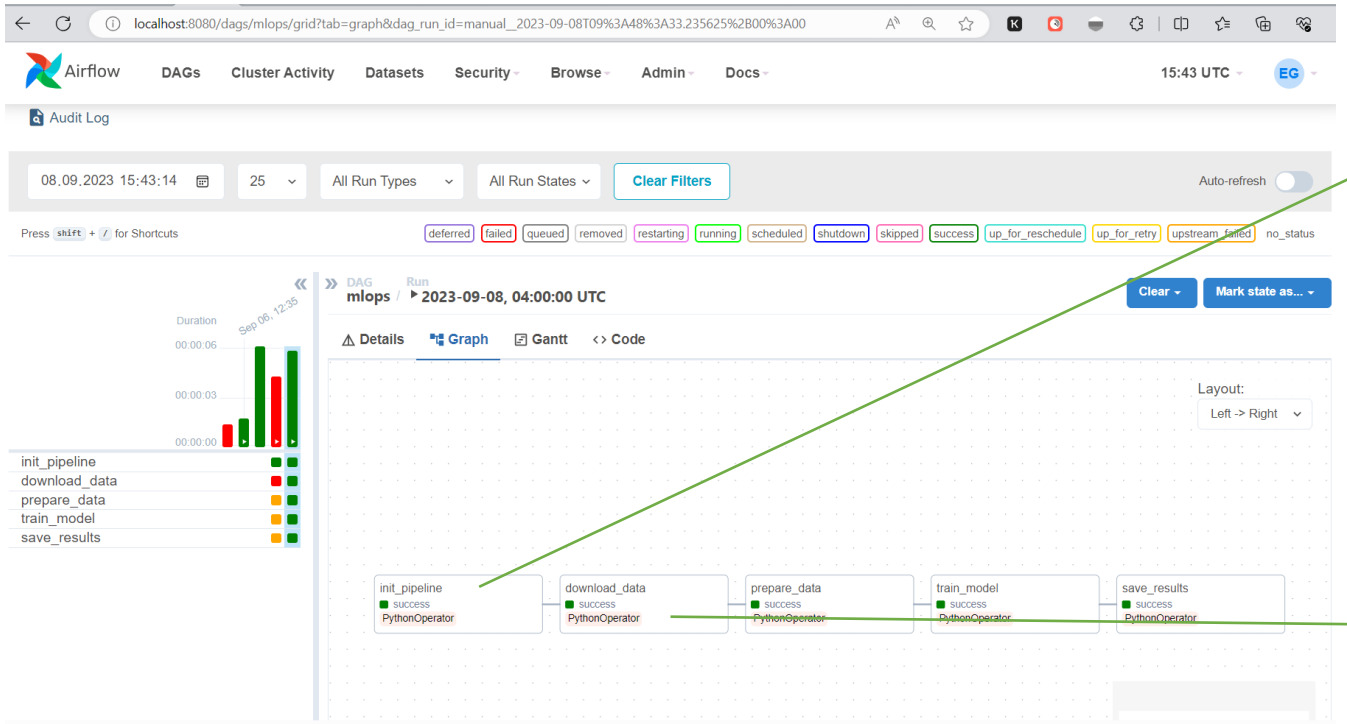




DAG — это ориентированный ациклический граф, т.е. граф с началом и концом, без циклов, но допускает параллельные пути.

Это сущность, которая объединяет задач в единый пайплайн (цепочку задач).

Как выглядит UI




```python
def init_pipeline() -> Dict[str, Any]:
    # Инициализируем переменную для сбора метрик.
    metrics = {}

    # Записываем время запуска пайплайна.
    pipeline_start = datetime.now()
    metrics["pipeline_start"] = pipeline_start

    return metrics
```

```python
def download_data(**kwargs) -> Dict[str, Any]:
    import connectors
    import utils
    # Подтягиваем метрики из предыдущего шага.
    ti = kwargs["ti"]
    metrics = ti.xcom_pull(task_ids="init_pipeline")

    # Скачиваем данные.
    data = connecotrs.download_data()
    prepared_data = utils.prepare_data(data)

    # Записываем количество данных в метрики.
    metrics["data_n"] = data.chape[0]

    # Сохраняем данные в облако.
    connectors.save_data_to_s3(data)

    return metrics
```

MLFlow

Инструмент с открытым исходным кодом для разработки и обеспечения жизненного цикла модели машинного обучения

MLFlow Tracking

Отслеживание экспериментов

MLFlow Projects

Упаковка экспериментов

MLFlow Models

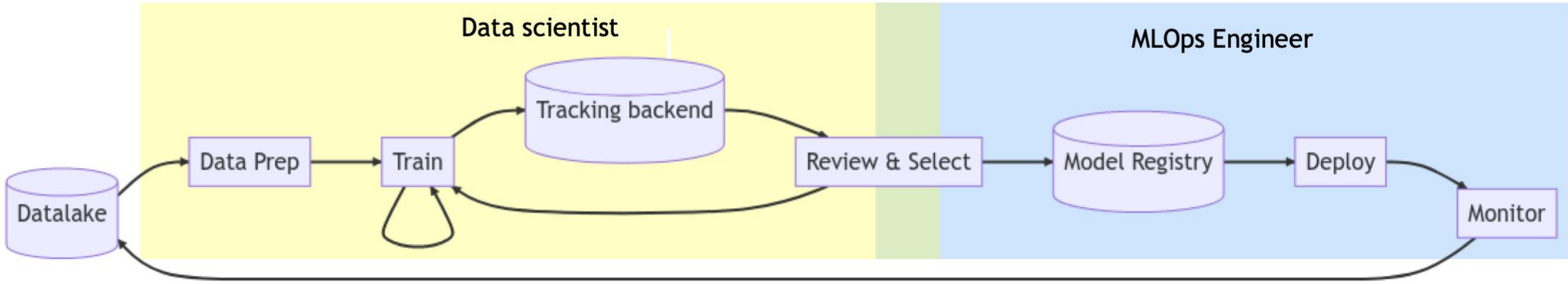
Упаковка моделей

MLFlow Registry

Репозиторий моделей

Ссылка на документацию

MLflow Tracking



Ссылка на документацию

S3 – Simple Storage Service

Объектное хранилище, работающее по одноименному протоколу.

S3 API был разработан в Amazon Web Services (AWS).

Любой объект в хранилище можно получить с помощью URL-ссылки с уникальным идентификатором объекта.

Общение с объектным хранилищем происходит либо через API, либо через Web-интерфейс.

Ссылка на документацию

Давайте поможем Jupyter Notebook стать автоматически обучаемой моделью

Разобрали сегодня

- Чем и зачем будем заниматься весь курс
- Как устроены AirFlow и MLFlow
- Что такое S3

В следующей серии

- Создание виртуального окружения
- Установка MLFlow
- Установка и настройка AirFlow