# Модуль 3. Управление потоком операций (xFlow).

## В этом модуле

Конвейеры состоят из операций, ходом выполнения которых необходимо управлять: контролировать последовательность выполнения, осуществлять мониторинг, своевременно устранять ошибки, версионировать сами конвейеры и результаты их работы. Операции в конвейере объединяются с использованием направленного ациклического графа (DAG, Directed Acyclic Graph). Последовательное и параллельное выполнение операций в порядке, заданном в DAG, создает поток (flow) действий и появляющихся артефактов. Существуют различные инструменты для работы с таким потоком операций, два таких инструмента, AirFlow и MLFlow рассматриваются в этом модуле. В юнитах 5-8 поставлена и подобно разобрана практическая задача из области работы с данными и машинного обучения, в которой применяются Airflow для автоматизации операций и MLFlow для мониторинга процессов. Решение этой практической задачи позволит вам получить навыки использования Airflow и MLFlow, достаточные для выполнения задания самостоятельной работы, которое дано в юните 9.

## Содержание юнитов

|  |
| --- |
| 1. Постановка задачи |
| 1. Настройка окружения проекта |
| 1. Apache Airflow для автоматизации сбора данных и обучения |
| 1. Мониторинг выполнения операций с использованием MLFlow |
| 1. Задание для самостоятельной работы |

# Модуль 2. Юнит 5. Постановка задачи

## Введение

В этом юните описана практическая задача, которую вы будете решать с применением Apache Airflow и MLFlow в следующих юнитах 6-8. Задача заключается в сборе данных и их обработке, а также последующем обучении модели ML. Airflow используется для автоматизации операций, а MLFlow для мониторинга процессов.

## Содержание

Предположим, что в нашем проекте понадобилось собирать информацию о рейтинге фильма или телепередачи из доступных открытых источников. Такими источниками могут быть:

* классический датасет отзывов на фильмы IMDB,
* youtube.ru,
* wikipedia,
* тематические сайты,
* тематические группы в мессенджерах,
* и многие другие источники…

Необходимость регулярного сбора данных связана с тем, что рейтинги постоянно обновляются, а также могут выходить новые фильмы и передачи. Поэтому в практических проектах необходимо запускать такой сбор по расписанию. Интерфейсы, по которым может быть получена эта информация, тоже различны, это могут быть:

* реляционные и нереляционные базы данных,
* стандартные датасеты, например, в xml или csv формате,
* API к агрегаторам новостей, социальным сетям, мессенджерам,
* SparQL интерфейс DBpedia (API для доступа к информационным базам wikipedia),
* RSS каналы,
* html страницы.

В зависимости от типа интерфейса для интеграции с соответствующим источником информации в программном обеспечении создается соответствующий коннектор, поддерживающий интерфейс для интеграции с источником данных.

Для реализации задачи сбора информации о рейтингах фильмов необходимо создать конвейер, состоящий из операций запуска коннекторов для разных источников по расписанию. Коннекторы будут запускаться в определенное время, брать из конфигурационного файла перечень фильмов, для которых необходимо получить информацию о рейтинге, делать опрос доступных источников и сохранять информацию в базу данных. Далее будет запускаться обработка полученных данных и преобразование в числовые признаки (векторы эмбеддингов), на основании которых алгоритм регрессии будет предсказывать рейтинг фильма в будущем.

Для автоматизации запуска коннекторов для сбора информации по расписанию мы будем использовать Apache Airflow. Также Apache Airflow мы используем для сохранения собранной информации в базе данных в «сыром» виде. Для упрощения задачи ограничимся чтением информации из youtube.com. Следующие операции, которые войдут в автоматизированный конвейер Apache Airflow, это обработка данных и запуск обучения и тестирования модели. Для мониторинга статуса выполнения операций мы используем MLFlow.

## Тест

1. Какие инструменты MLOps вы будете применять в этом модуле? (0.25)
   1. **MLFlow**
   2. ClearML
   3. dvc
   4. **Airflow**
2. С каким источником информации будет выполняться интеграция в задании? (0.25)
   1. **youtube**
   2. twitter
   3. yandex
   4. rbc.ru
3. Какие интерфейсы могут быть использованы для получения данных из внешних источников? (0.25)
   1. **SQL запросы**
   2. **API и JSON**
   3. ICMP
   4. **SparQL**
4. Какие задачи автоматизирует Airflow в конвейере проекта? (0.25)
   1. **сбор данных**
   2. **обработка данных**
   3. **обучение модели**
   4. выведение модели в production

## Итоги/выводы

В этом юните сформулирована задача, которую вы будете решать в юнитах 6-8, и даны рекомендации по ее решению.

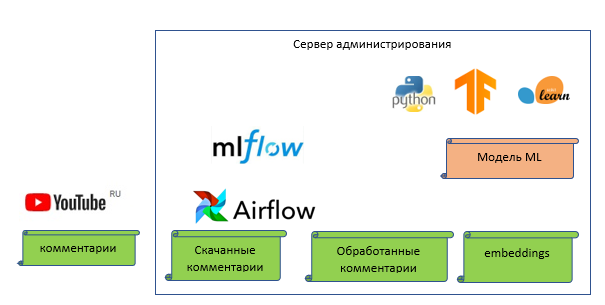
# Модуль 3. Юнит 6. Настройка окружения проекта.

## Введение

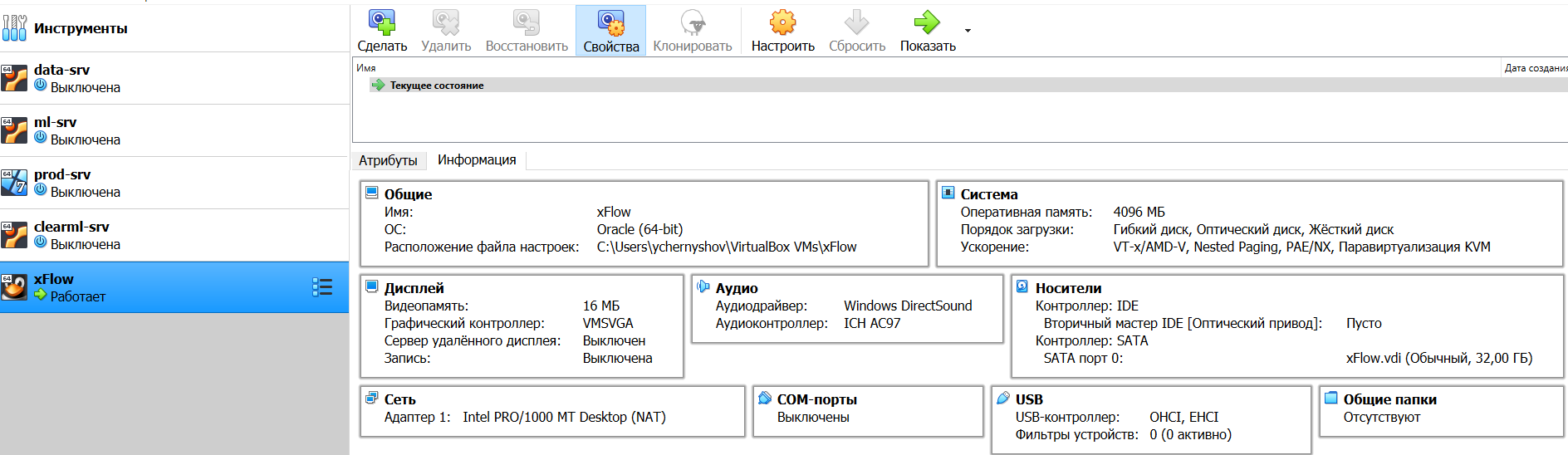
В этом юните вы настроите рабочее окружение для решения задачи: создадите необходимую инфраструктуру, установите системное и прикладное программное обеспечение и выполните интеграцию с внешними источниками данных.

## Содержание

Для работы вам понадобится виртуальная машина, на которую будет установлено все необходимое программное обеспечение. Главная задача в реализуемом проекте - организация автоматизированного сбора данных, обработка этих данных, обучение модели ML и первичное тестирование. То есть вам понадобится инструмент для автоматизации операций сбора данных, в этой роли будем использовать Airflow, а также инструмент для мониторинга сбора данных, сохранения и трекинга артефактов, и обучения моделей, для этого будем использовать MLFlow. В этом задании нет необходимости разделять функции разных участников команды и специализированное программное обеспечение на нескольких рабочих серверах, как вы это делали в предыдущих модулях. Поэтому далее создадим одну виртуальную машину и на нее установим все необходимое программное обеспечение. Целевая архитектура, которая будет создана в этом юните, имеет следующий вид:



Сначала создайте виртуальную машину «xflow» в virtualbox, как это делалось ранее



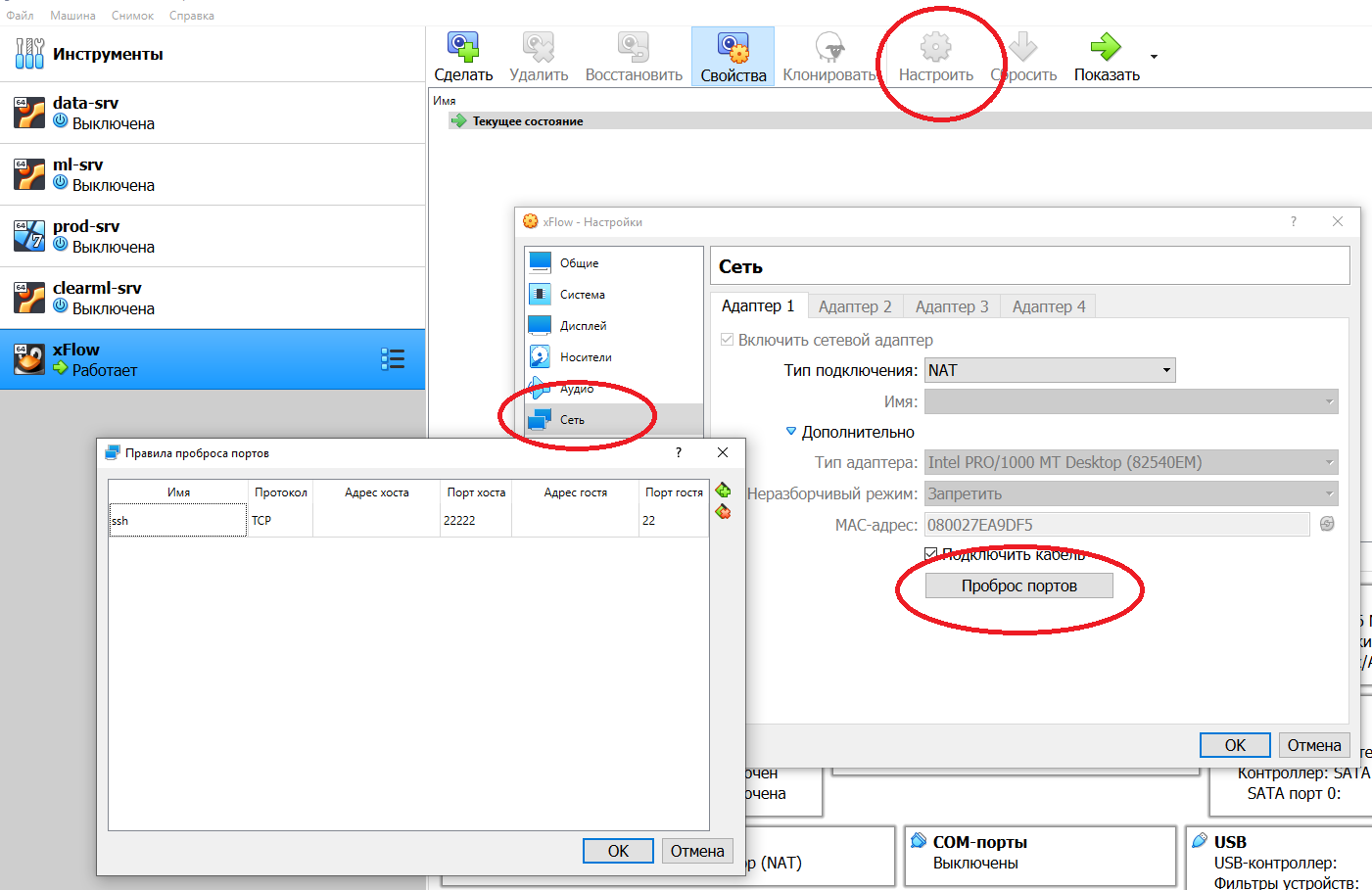
В качестве операционной системы, также, как и в предыдущих модулях, рекомендуем использовать Ubuntu 20.4.

## Установка и настройка прикладного программного обеспечения разработчика.

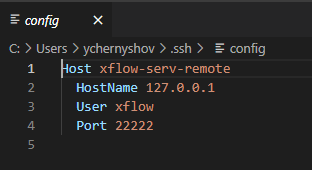
Для удобства работы нам понадобится инструмент разработчика IDE, например уже знакомый вам VS Code, который подключается удаленно к серверу через ssh и позволяет редактировать и выполнять скрипты на удаленном сервере. Для организации такой инфраструктуры требуется установить openssh-server на виртуальной машине командой в консоли

**sudo apt install openssh-server**

Далее необходимо настроить порт в virtualbox (настроить “проброс” порта)



После чего в конфигурационном файле ssh в VS Code прописать параметры подключения



И открыть удаленное подключение, нажав на кнопку слева внизу в интерфейсе VS Code.



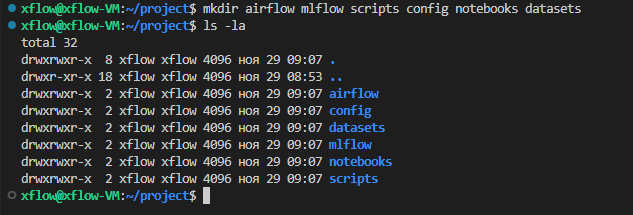
## Создание файловой структуры и виртуального окружения.

Для создания нужной конфигурации для работы вам понадобится организовать файловую структуру. Для этого в своей пользовательской директории создайте директорий проекта и в нем директории mlflow, airflow, notebooks (если планируете хранить и использовать jupyter ноутбуки), config (для конфигурационных файлов) и другие папки, которые нужны вам для работы.

**mkdir project**

**cd project**

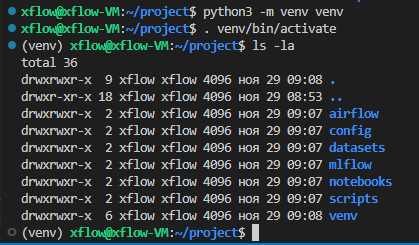
**mkdir airflow mlflow scripts config notebooks datasets**



Находясь в главной папке проекта создайте и активируйте виртуальное окружение venv

**python3 -m venv venv**

**source venv/bin/activate**



Теперь у вас есть виртуальное окружение, в котором вы можете установить все необходимые пакеты, в том числе airflow и mlflow.

## Установка и настройка airflow

Программное обеспечение Apache Airflow предназначено для соединения отдельных операций в единый конвейер, который описывается ациклическим направленным графом операций (DAG, Directed Acyclic Graph). В этот граф вы объедините все операции в проекте. Процесс установки Apache Airflow описан в официальной документации на сайте <https://airflow.apache.org/docs/apache-airflow/stable/installation/index.html>. Есть различные способы развернуть Airflow, например с использованием docker и helm. Мы используем самый простой, уже известный вам вариант, установку с использованием python установщика pip. Официально поддерживаемым установщиком является только pip, разработчики Apache Airflow не гарантируют работоспособность при использовании в качестве установщика poetry или другие подобные программы. Важно, что airflow устанавливается вместе с зависимостями, если этого не сделать, то возможна некорректная работа airflow. Вот образец команды из официальной документации на сайте, Раздел «Installation from PyPI» <https://airflow.apache.org/docs/apache-airflow/stable/installation/installing-from-pypi.html>:

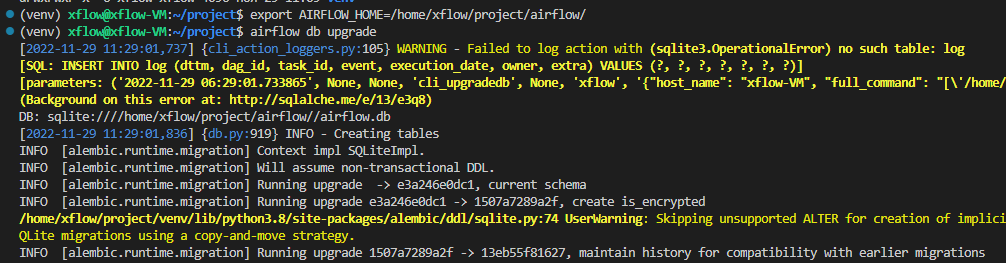
**pip install "apache-airflow[celery]==2.2.5" \**

**--constraint https://raw.githubusercontent.com/apache/airflow/constraints-2.2.5/constraints-3.8.txt**

В этом команде «2.2.5» это устанавливаемая версия airflow, а «3.8» - версия используемого интерпретатора python. После установки необходимо установить значение переменной окружения AIRFLOW\_HOME и выполнить инициализацию базы данных.

**export AIRFLOW\_HOME=/home/xflow/project/airflow**

**airflow db init** или **airflow db upgrade**



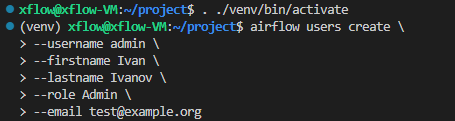
В файле airflow.cfg в разделе webserver необходимо установить значение параметров rbac, а также load\_examples (False, если не хотите подгружать примеры, актуально для production запусков airflow)

**[webserver]**

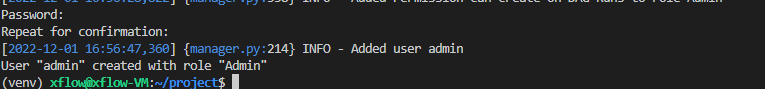
**rbac = True**

**load\_examples = False**

Для работы в airflow с соответствующими привилегиями надо создать пользователя



Далее система попросит вас задать пароль, после чего пользователь будет успешно создан.

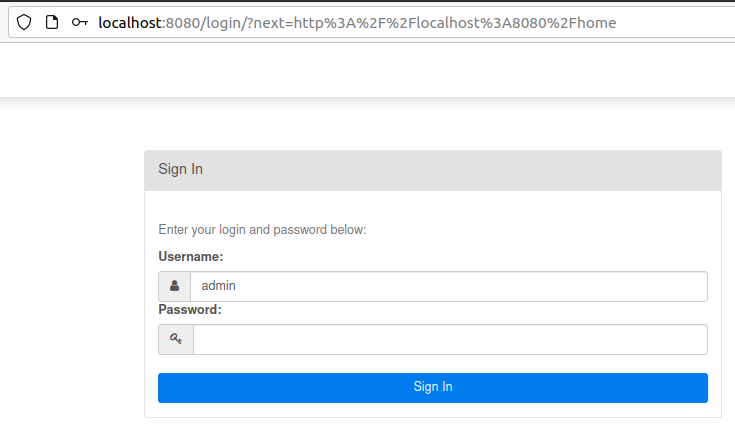


Теперь можно запустить airflow web-сервер для того, чтобы работать в графическом интерфейсе, а также scheduler, обеспечивающий выполнение задач по расписанию

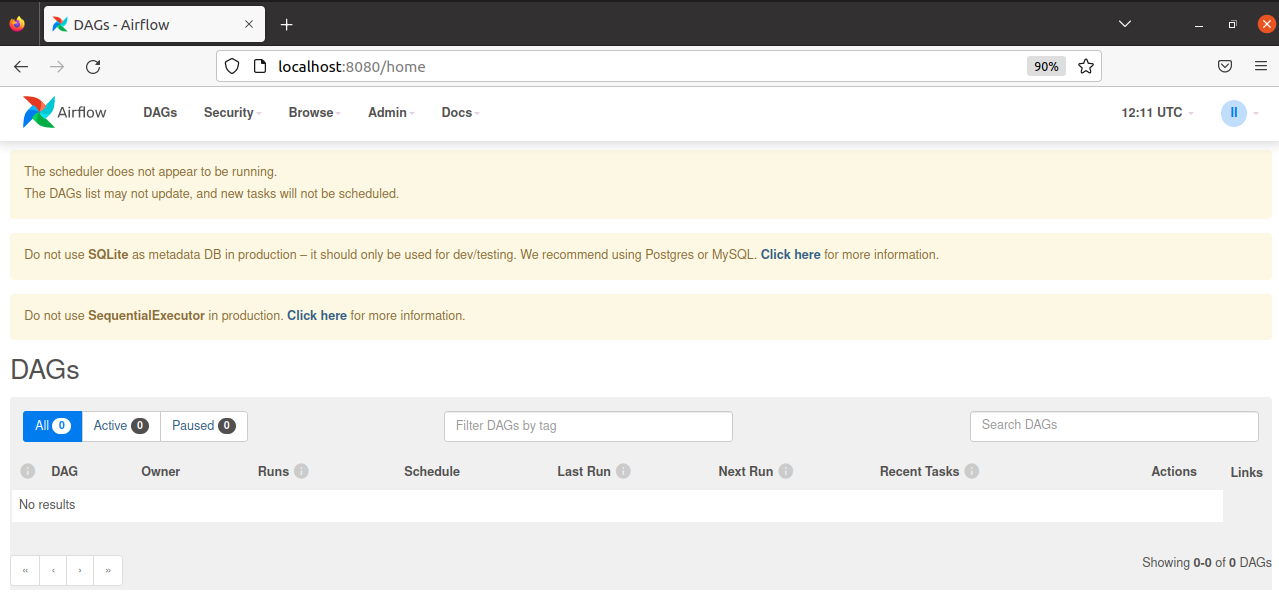
**airflow webserver -p 8080**

**airflow scheduler**

После этого можно через браузер зайти на виртуальный сервер на порт 8080 и с использованием реквизитов созданного пользователя осуществить вход в графический интерфейс airflow



Если все шаги выполнены правильно, то вы увидите следующее окно, являющееся рабочим пространством для пользователя airflow



На тот случай, когда что-то «пошло не так», знайте полезную команду, которая находит и останавливает процессы, содержащие в названии airflow

**kill -9 `ps aux | grep airflow | awk ‘{print $2}’`**

Теперь airflow готов к использованию.

## Установка и настройка MLFlow

Программное обеспечение MLFlow понадобится в проекте для того, чтобы осуществлять мониторинг выполнения всех операций в проекте: сбор и обработка данных, формирование векторов эмбеддингов, обучение модели. Для установки MLFlow необходимо выполнить следующие операции:

**pip install mlflow**

**export MLFLOW\_REGISTRY\_URI=mlflow**

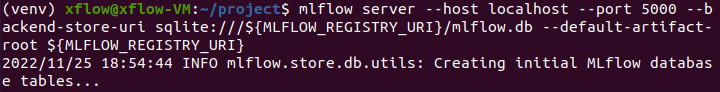
В качестве служебной базы данных также используется sqlite, как и для airflow, поэтому специально программное обеспечение базы данных sqlite устанавливать не надо. Однако, если airflow и mlflow в вашем проекте функционируют на разных серверах, потребуется sqlite установить каждый раз отдельно для каждого сервера.

Запуск mlflow осуществляется командой

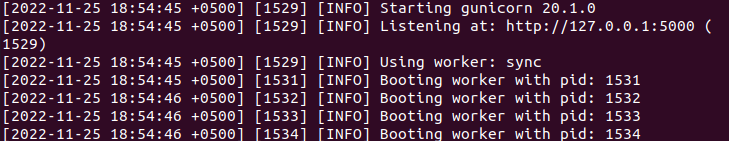
**mlflow server --host localhost --port 5000 \**

**--backend-store-uri sqlite:///${MLFLOW\_REGISTRY\_URI}/mlflow.db \**

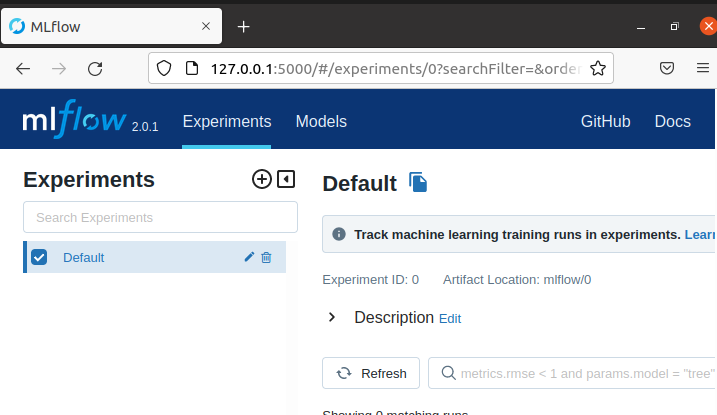
**--default-artifact-root ${MLFLOW\_REGISTRY\_URI}**



**…**



Теперь можно открыть в браузере приложение на localhost:5000 (обратите внимание, что по умолчанию используется небезопасный http, а не https)



Теперь MLFLow готов к использованию.

Установка и настройка Jupyter notebook

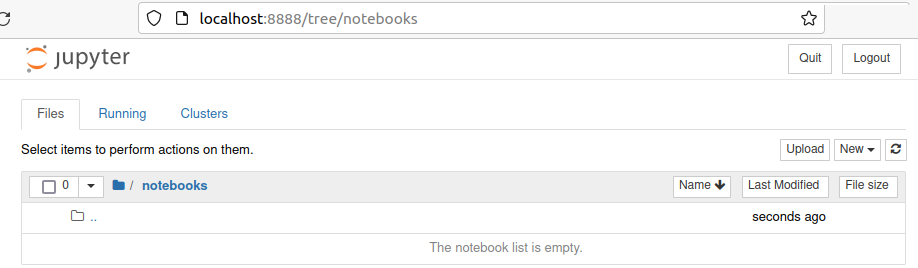
Для организации работы исследователя данных или для проведения первичных экспериментов и проверки гипотез инженером ML понадобится jupyter notebook, давайте его также установим в системе. Находясь в созданном вами ранее виртуальном окружении установите jupyter с помощью команды

**pip install jupyter**

в том числе будет установлен jupyter notebook, который можно запустить в командной строке

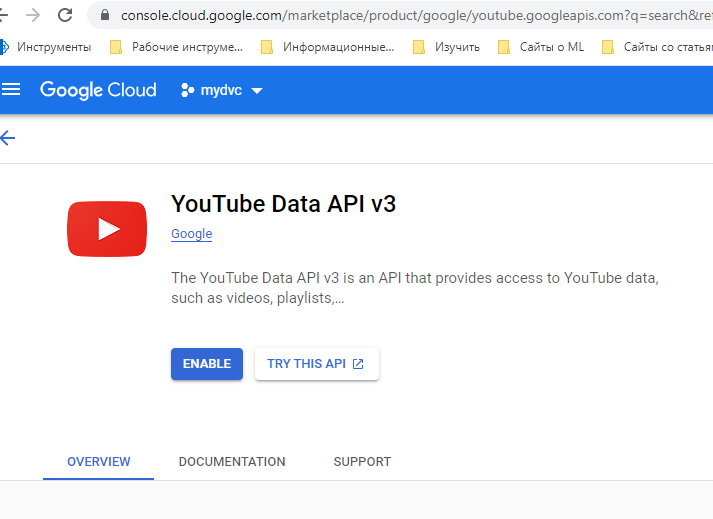
**jupyter notebook**

после чего появится возможность открывать jupyter ноутбуки в браузере

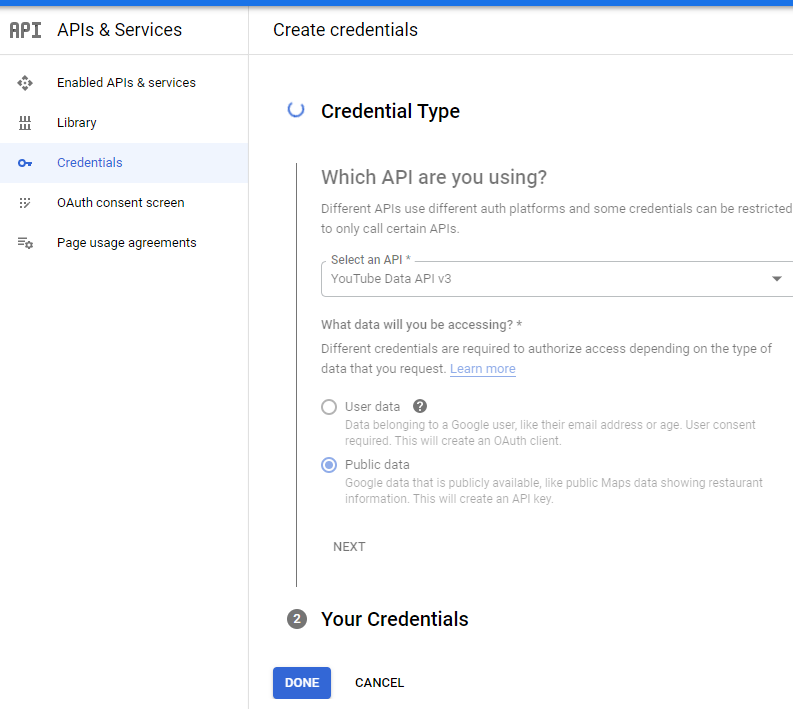


## Интеграция с внешним источником данных youtube

Сервис youtube входит в экосистему продуктов компании Google, которая предоставляет широкие возможности для разработчиков. Доступные сервисы Google для проектов разработки, обучающие курсы, ссылки на блоги сообществ и другую информацию можно найти на сайте <https://developers.google.com.> Мы будем интегрироваться с API «YouTube Data API V3», информацию можно найти здесь: [https://developers.google.com/youtube/v3/docs](https://developers.google.com/youtube/v3/docs/).



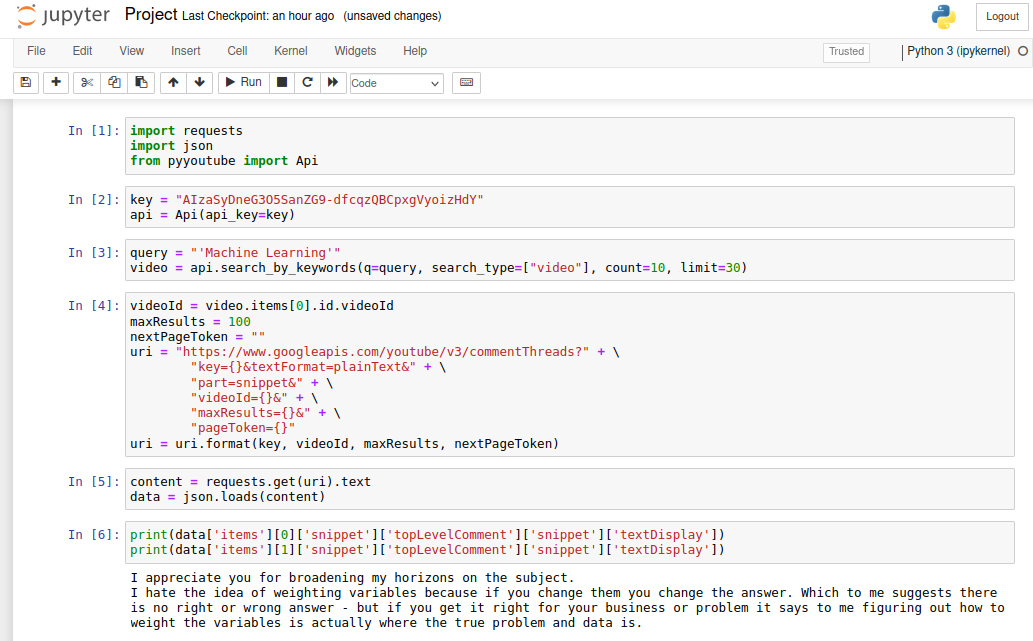
Для дальнейшей работы вам понадобится ключ разработчика Google для интеграции с youtube через API. Ключ разработчика можно получить в консоли разработчиков Google, для доступа к которой необходимо создать аккаунт в экосистеме google.com. Например, этот аккаунт может использоваться дл доступа к почте gmal.com, пользовательского профиля youtube.com или любых других персонифицированных сервисов Google. Также для подключения к ресурсам разработчика и обеспечения безопасности этого подключения понадобится зарегистрировать проект на странице <https://console.cloud.google.com.>



Для автоматизированного получения данных от платформы youtube через унифицированный API понадобится библиотека python-youtube, которую можно установить в виртуальное окружение командой

**pip install python-youtube**

Из установленной библиотеки python-youtube необходимо импортировать класс Api для подключения по API к youtube. С помощью метода pyyoutube.Api.search\_by\_keywords() осуществляется поиск видео на платформе youtube по соответствующему поисковому запросу. Полученные результаты используются для формирования запроса комментариев к видео с помощью стандартной python библиотеки requests. Пример jupyter ноутбука, демонстрирующего вышеизложенное, приведен ниже.



Итак, вы создали необходимую инфраструктуру и установили программное обеспечение для дальнейшей работы в юнитах 6-8. В том числе, вы

* установили виртуальную машину virtualbox и операционную систему Ubuntu 20.4,
* настроили IDE VS Code для удаленного доступа к проекту на виртуальной машине с использованием ssh,
* установили и настроили библиотеку python-venv для работы с виртуальными окружениями,
* установили прикладное программное обеспечение Apache Airflow, MLFlow и Jupyter notebook,
* выполнили интеграцию с внешним источником получения данных API youtube.

## Тест

1. Какая база данных используется в качестве служебной для airflow и mlflow в примере? (0.25)
   1. mysql
   2. **sqlite**
   3. postgresql
   4. oracle
2. Какая команда удалит все процессы, содержащие в названии «mlflow»? (0.25)
   1. **kill -9 `ps aux | grep mlflow | awk ‘{print $2}’`**
   2. delete all mlflow
   3. ps | grep mlflow | kill
   4. remove mlflow
3. Для чего предназначен airflow sheduler? (0.25)
   1. **Автоматизированное выполнение задач по расписанию**
   2. Тестирование модели
   3. Настройка базы данных
   4. Аутентификация пользователей
4. Какой интерфейс используется для получения данных из платформы youtube в проекте? (0.25)
   1. youtube-xml
   2. **API**
   3. csv
   4. SQL

## Итоги/выводы

В этом юните вы создали необходимую инфраструктуру и настроили окружение для дальнейшей работы, в том числе установили Apache Airflow и MLFlow, а также обеспечили подключение к внешнему источнику данных youtube.

# Модуль 3. Юнит 7. Apache Airflow для автоматизации сбора данных и обучения.

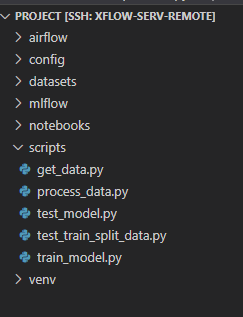
## Введение

В этом юните вы настроите автоматизацию выполнения конвейера операций в проекте. Для этого будет использован Apache Airflow.

## Содержание

Apache Airflow позволяет объединять операции в граф операций, DAG. Это эффективно для рутинных, часто повторяющихся в определенной последовательности операций, для которых необходимо соблюдать расписание их выполнения. Хороший пример подобной задачи это организация конвейера ETL (Extract, Transfer, Load), выполняющего извлечение, передачу и загрузку данных в проектах, использующих большие данные.

В нашем проекте необходимо извлекать информацию из API, преобразовывать эту информацию, разделять данные на обучающую и тестовую выборки, обучать и тестировать модель. Давайте создадим для этих операций соответствующие python скрипты в папке scripts нашего проекта



Сначала реализуем логику отдельных скриптов.

**get\_data.py**

Предназначен для получения данных из внешнего источника, платформы youtube. Для этих целей используется API. По поисковому запросу “Mission Impossible” вы можете получить списки комментариев роликов youtube, которые дает платформа по этому поисковому запросу. В полученном json файле можно достать значение параметра likeCount, которое и сохраняется в используемом далее наборе данных, в файле data.csv.

import requests

import json

from pyyoutube import Api

key = "AIzaSyDneG3O5SanZG9-dfcqzQBCpxgVyoizHdY"

api = Api(api\_key=key)

query = "'Mission Impossible'"

video = api.search\_by\_keywords(q=query, search\_type=["video"], count=10, limit=30)

maxResults = 100

nextPageToken = ""

s = 0

for id\_ in [x.id.videoId for x in video.items]:

    uri = "https://www.googleapis.com/youtube/v3/commentThreads?" + \

            "key={}&textFormat=plainText&" + \

            "part=snippet&" + \

            "videoId={}&" + \

            "maxResults={}&" + \

            "pageToken={}"

    uri = uri.format(key, id\_, maxResults, nextPageToken)

    content = requests.get(uri).text

    data = json.loads(content)

    for item in data['items']:

        s += int(item['snippet']['topLevelComment']['snippet']['likeCount'])

with open('/home/xflow/project/datasets/data.csv', 'a') as f:

    f.write("{}\n".format(s))

**process\_data.py**

Этот скрипт предназначен для обработки данных, при которой полученные значения нормализуются, переводятся в диапазон от 0 до 1. Также в выходной файл записывается индекс значения.

import pandas as pd

df = pd.read\_csv('/home/xflow/project/datasets/data.csv', header=None)

df[0] = (df[0]-df[0].min())/(df[0].max()-df[0].min())

with open('/home/xflow/project/datasets/data\_processed.csv', 'w') as f:

    for i, item in enumerate(df[0].values):

        f.write("{},{}\n".format(i, item))

**train\_test\_split.py**

Предназначен для разделения полученных и обработанных данных на тренировочную и тестовую выборки.

import pandas as pd

import numpy as np

df = pd.read\_csv('/home/xflow/project/datasets/data\_processed.csv', header=None)

idxs = np.array(df.index.values)

np.random.shuffle(idxs)

l = int(len(df)\*0.7)

train\_idxs = idxs[:l]

test\_idxs = idxs[l+1:]

df.loc[train\_idxs, :].to\_csv('/home/xflow/project/datasets/data\_train.csv',

                        header=None,

                        index=None)

df.loc[test\_idxs, :].to\_csv('/home/xflow/project/datasets/data\_test.csv',

                        header=None,

                        index=None)

**train\_model.py**

Предназначен для обучения модели ML

from sklearn.linear\_model import LinearRegression

import pickle

import pandas as pd

df = pd.read\_csv('/home/xflow/project/datasets/data\_train.csv', header=None)

df.columns = ['id', 'counts']

model = LinearRegression()

model.fit(df['id'].values.reshape(-1,1), df['counts'])

with open('/home/xflow/project/models/data.pickle', 'wb') as f:

    pickle.dump(model, f)

**test\_model.py**

Предназначен для тестирования модели ML

from sklearn.linear\_model import LinearRegression

import pickle

import pandas as pd

df = pd.read\_csv('/home/xflow/project/datasets/data\_test.csv', header=None)

df.columns = ['id', 'counts']

model = LinearRegression()

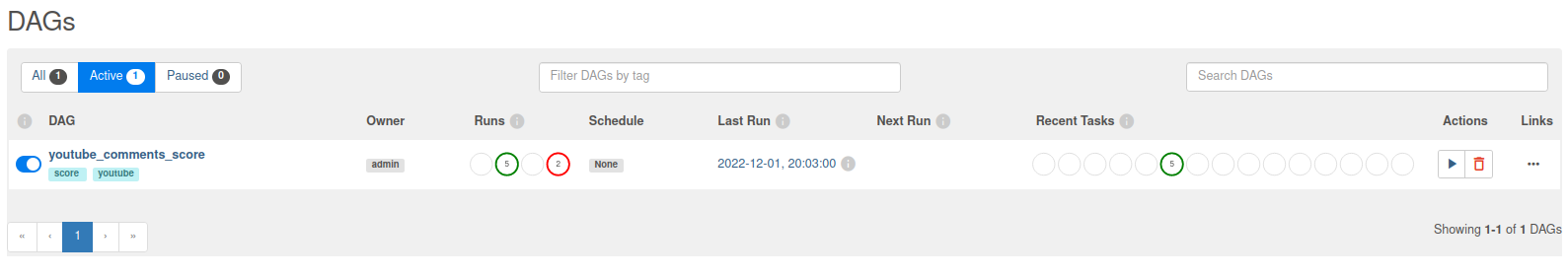
with open('/home/xflow/project/models/data.pickle', 'rb') as f:

    model = pickle.load(f)

score = model.score(df['id'].values.reshape(-1,1), df['counts'])

print("score=", score)

Теперь давайте реализуем управление последовательностью этих операций по заданному расписанию. Для этого создадим файл youtube\_comments\_score.py и разместим его в папке dags рабочего директория airflow. Запущенный командой airflow scheduler планировщик задач («шедулер») отвечает за распознавание новых DAGов, описанных в python скриптах, размещенных в папке dags. После распознавания нового DAG он будет отображен в графическом интерфейсе airflow



**Листинг файла youtube\_comments\_score.py**

from airflow import DAG

from airflow.operators.bash import BashOperator

import pendulum

import datetime as dt

args = {

    "owner": "admin",

    "start\_date": dt.datetime(2022, 12, 1),

    "retries": 1,

    "retry\_delays": dt.timedelta(minutes=1),

    "depends\_on\_past": False

}

with DAG(

    dag\_id='youtube\_comments\_score',

    default\_args=args,

    schedule\_interval=None,

    tags=['youtube', 'score'],

) as dag:

    get\_data = BashOperator(task\_id='get\_data',

                            bash\_command="python3 /home/xflow/project/scripts/get\_data.py",

                            dag=dag)

    process\_data = BashOperator(task\_id='process\_data',

                            bash\_command="python3 /home/xflow/project/scripts/process\_data.py",

                            dag=dag)

    train\_test\_split\_data = BashOperator(task\_id='train\_test\_split\_data',

                            bash\_command="python3 /home/xflow/project/scripts/train\_test\_split\_data.py",

                            dag=dag)

    train\_model = BashOperator(task\_id='train\_model',

                            bash\_command="python3 /home/xflow/project/scripts/train\_model.py",

                            dag=dag)

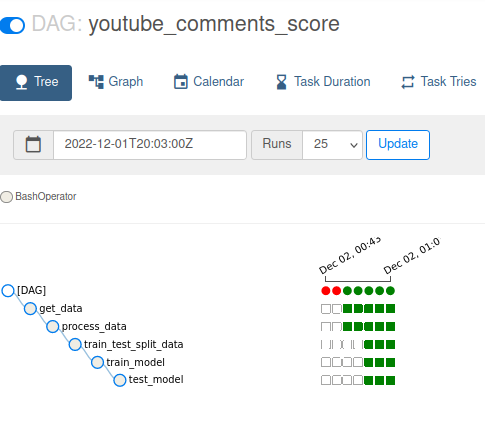
    test\_model = BashOperator(task\_id='test\_model',

                            bash\_command="python3 /home/xflow/project/scripts/test\_model.py",

                            dag=dag)

    get\_data >> process\_data >> train\_test\_split\_data >> train\_model >> test\_model

Таким образом, с использованием BashOperator вы вызываете выполнение python скриптов и объединяете эти операторы в конвейер оператором >>. В итоге, запустив конвейер операций на выполнение, вы увидите успешно выполненную задачу.



Итак, вы реализовали конвейер операций, который выполняется автоматически под контролем airflow.

## Тест

1. В какой переменной окружения хранится путь до рабочего директория Airflow? (0.25)
   1. **AIRFLOW\_HOME**
   2. AIRFLOW\_MAIN
   3. AIRFLOW\_PATH
   4. AIRFLOW\_DIR
2. Как в crontab задать выполнение “каждые 5 минут”? (0.25)
   1. **5 \* \* \* \***
   2. \* 5 \* \* \*
   3. Every 5 minutes
   4. 5 5 5 5 5
3. В какой папке должны храниться python файлы с описанием DAG? (0.25)
   1. python
   2. **dags**
   3. files
   4. flows
4. Какой оператор позволяет задать очередность выполнения операций? (0.25)
   1. ++
   2. **>>**
   3. ::
   4. ->

## Итоги/выводы

В этом юните вы создали конвейер выполнения операций проекта, который выполняется автоматически, с использованием Apache Airflow.

# Модуль 3. Юнит 8. Мониторинг выполнения операций с использованием MLFlow.

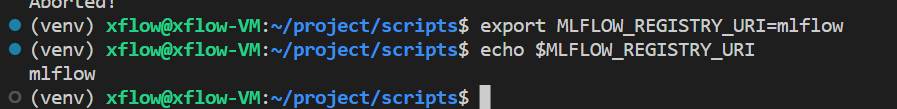
## Введение

В предыдущем юните вы собрали все операции проекта в единый конвейер выполнения. В этом юните вы организуете мониторинг выполнения всех операций проекта с использованием MLFlow.

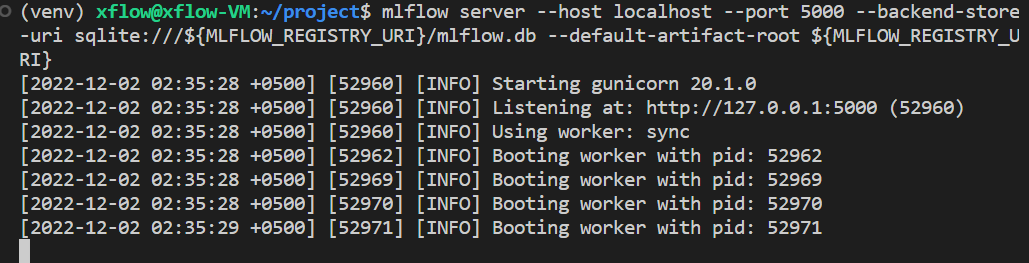
## Содержание

MLFlow позволяет устаналивать наблюдение за параметрами операций в конвейере проекта машинного обучения, обеспечивая мониторинг их выполнения и сохранение результатов отдельных операций (артефактов). В MLFlow есть специальные модули, позволяющие “из коробки” работать с популярными в машинном обучении библиотеками, например, sklearn. Удобство использования MLFlow для нашего проекта, также как и Airflow из предыдущего юнита, заключается в том, что вся интеграция в проект описывается на языке python.

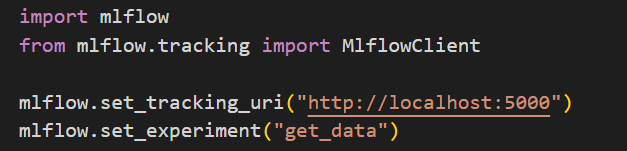
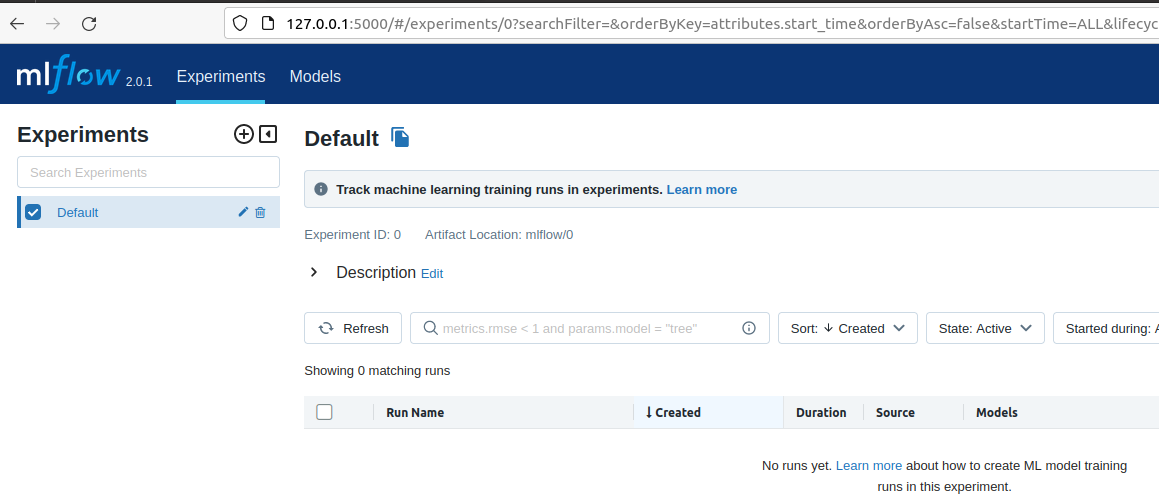
Для постановки этапов проекта под мониторинг MLFlow необходимо добавить код в уже созданные нами файлы python скриптов. Перед началом работы, находясь в созданном нами в юните 6 виртуальном окружении, необходимо убедиться, что переменная MLFLOW\_REGISTRY\_URI установлена в правильное значение



Если процесс mlflow не запущен, то его необходимо запустить



После чего появится возможность заходить в графический интерфейс MLFlow через браузер



**Обновленный листинг get\_data.py**

import requests

import json

from pyyoutube import Api

import os

import mlflow

from mlflow.tracking import MlflowClient

os.environ["MLFLOW\_REGISTRY\_URI"] = "/home/xflow/project/mlflow/"

mlflow.set\_tracking\_uri("http://localhost:5000")

mlflow.set\_experiment("get\_data")

key = "AIzaSyDneG3O5SanZG9-dfcqzQBCpxgVyoizHdY"

api = Api(api\_key=key)

query = "'Mission Impossible'"

video = api.search\_by\_keywords(q=query, search\_type=["video"], count=10, limit=30)

maxResults = 100

nextPageToken = ""

s = 0

with mlflow.start\_run():

    for i, id\_ in enumerate([x.id.videoId for x in video.items]):

        uri = "https://www.googleapis.com/youtube/v3/commentThreads?" + \

              "key={}&textFormat=plainText&" + \

              "part=snippet&" + \

              "videoId={}&" + \

              "maxResults={}&" + \

              "pageToken={}"

        uri = uri.format(key, id\_, maxResults, nextPageToken)

        content = requests.get(uri).text

        data = json.loads(content)

        for item in data['items']:

            s += int(item['snippet']['topLevelComment']['snippet']['likeCount'])

    mlflow.log\_artifact(local\_path="/home/xflow/project/scripts/get\_data.py",

                        artifact\_path="get\_data code")

    mlflow.end\_run()

with open('/home/xflow/project/datasets/data.csv', 'a') as f:

    f.write("{}\n".format(s))

**Обновленный листинг train\_model.py**

from sklearn.linear\_model import LinearRegression

import pickle

import pandas as pd

import os

import mlflow

from mlflow.tracking import MlflowClient

mlflow.set\_tracking\_uri("http://localhost:5000")

mlflow.set\_experiment("train\_model")

df = pd.read\_csv('/home/xflow/project/datasets/data\_train.csv', header=None)

df.columns = ['id', 'counts']

model = LinearRegression()

with mlflow.start\_run():

    mlflow.sklearn.log\_model(model,

                             artifact\_path="lr",

                             registered\_model\_name="lr")

    mlflow.log\_artifact(local\_path="/home/xflow/project/scripts/train\_model.py",

                        artifact\_path="train\_model code")

    mlflow.end\_run()

model.fit(df['id'].values.reshape(-1,1), df['counts'])

with open('/home/xflow/project/models/data.pickle', 'wb') as f:

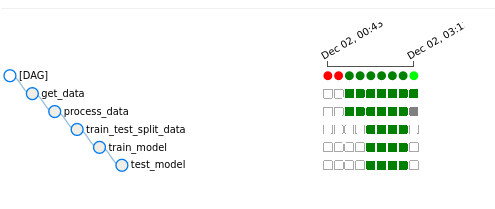
    pickle.dump(model, f)

Теперь вы установили нужные объекты python скриптов под наблюдение mlflow.

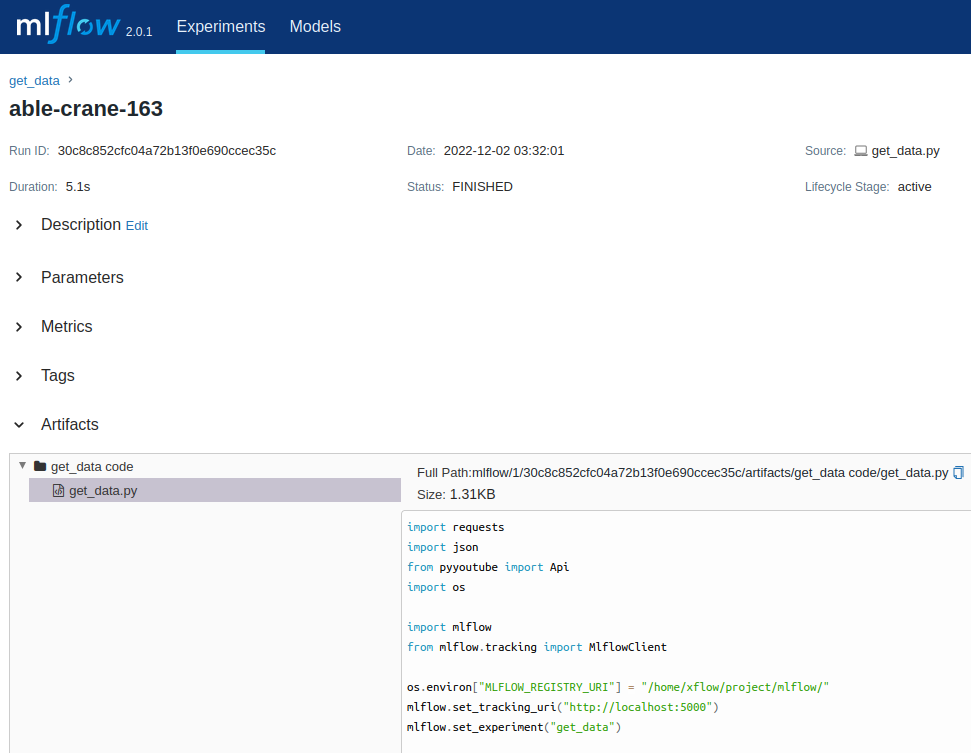
Возможно при выполнении python скрипта с добавленными частями mlflow интерпретатор python будет выдавать ошибку, связанную с отсутствием git на сервере, на котором выполняется python скрипт. В этом случаче необходимо установить git командой

**sudo apt install git**

Для запуска выполнения конвейера еще раз стартуем выполнение в Airflow



Также, поскольку мы сохранили код python скрипта как артифакт функцией mlflow.log\_artifact, мы можем посмотреть этот код в графическом интерфейсе mlflow.



Пользуясь другими функциями библиотеки mlflow вы можете устанавливать и другие объекты под трекинг, получая возможность мониторинга процессов и сверки полученных результатов: параметров, метрик. В итоге, наш конвейер операций проекта машинного обучения, который запускается автоматизированно с помощью Airflow, находится под мониторингом Mlflow.

## Тест

1. Какой номер порта обычно используется для web сервиса mlflow? (0.25)
   1. 1000
   2. **5000**
   3. 5050
   4. 5555
2. Какие функции используются для инициализации перед работой с mlflow в python скрипте? (0.25)
   1. mlflow.init()
   2. **mlflow.set\_tracking\_uri()**
   3. **mlflow.set\_experiment()**
   4. mlflow.start()
3. Какая функция сохраняет артефакт? (0.25)
   1. mlflow.write\_artifact()
   2. mlflow.set\_artifact()
   3. **mlflow.log\_artifact()**
   4. mlflow.save\_artifact
4. Какие функции используются в python скрипте для работы с библиотекой mlflow? (0.25)
   1. mlflow.begin\_run()
   2. **mlflow.start\_run()**
   3. m**lflow.end\_run()**
   4. mlflow.stop\_run()

## Итоги/выводы

В этом юните вы организовали мониторинг выполнения всех операций автоматизированного конвейера проекта, а также логирование результатов, с использованием MLFlow.

# Модуль 3. Юнит 9. Задача для самостоятельной работы.

## Содержание

В юнитах 6-8 вы научились применять основные возможности фреймворков Apache Airflow и MLFlow для автоматизации выполнения конвейера операций, мониторинга выполнения этих операций, сохранения и анализа результатов. В конвейер операций объединяются задачи, повторяющиеся в процессе жизненного цикла модели машинного обучения, идущие друг за другом в определенной последовательности, запускаемые на регулярной основе (по определенному расписанию). В юните 5 была поставлена задача извлечения данных из внешнего источника, голосов (“лайков”) для видеоклипов на платформе youtube, обработки и сохранения этих данных, обучения на их основе модели машинного обучения, линейной регрессии для предсказания будущих значений.

Знания, полученные при решении задачи в юнитах 5-8, можно применить для решения подобных задач. При выполнении итоговой самостоятельной работы по этому модулю вам необходимо:

* Определить какой-нибудь внешний источник получения данных и способ получения этих данных (http, curl, wget, API, SQL, SparQL, ...)
* Поставить задачу для алгоритма машинного обучения, выбрать модель и метрику
* Создать инфраструктуру, например, виртуальные машины virtualbox, установить и настроить для работы необходимое программное обеспечение, в том исле airflow и mlflow, а также venv для организации работы виртуального окружения
* Создать python скрипты для
  + Получения данных из внешнего источника
  + Преобразования данных
  + Формирования рабочего набора данных для обучения (train) и тестирования (test) модели
  + Обучения модели на тренировочных (train) данных и ее сохранения
  + Загрузки модели и проверки качества ее работы на тестовых (test) данных
* Добавить код airflow, позволяющий создавать и запускать на регулярной основе описанные операции проекта.
* Добавить код mlflow, позволяющий мониторить ход выполнения конвейера, сохранять и анализировать полученных артифакты.

В результате выполнения задачи вам необходимо создать и предъявить для проверки git репозиторий, который содержит весь исходный код и необходимые конфигурационные файлы для воссоздания условия выполнения эксперимента проверяющим (например, requirements.txt).

## Итоги/выводы

В этом юните сформулирована задача для самостоятельной работы и даны рекомендации по ее решению.

# Список источников

1. Видеолекция «Airflow и MLFlow автоматизаций пайплайнов Machine Learning / MLOps»

<https://www.youtube.com/watch?v=NfPf0Y770DA&t=2s> (канал miracl6)

1. Библиотека python для работы по API с youtube

<https://pypi.org/project/python-youtube/>

<https://developers.google.com/youtube/v3/docs/>