

## 2

# ЛР 2. Airflow + Spark

### ▼ Задача

Подключить Airflow к Apache Spark и выполнить новый DAG через SparkSession

### ▼ Описание

В рамках задания необходимо переделать свой **ИЛИ** сделать новый DAG, который будет выполнять spark-job из папки spark, через оператор *SparkSubmitOperator* или другой

Новые скрипты (или переделанные старые) необходимо добавить в папку spark, а саму папку не забыть “прокинуть” в образ через Dockerfile. Также требуется добавить в Dockerfile шаг, который устанавливает библиотеку для питона `apache-airflow-providers-apache-spark`, а также пакеты `procps` и `default-jre`. Установка пакетов через `apt` может быть только от рута, но работа сервиса должна идти под пользователем airflow, поэтому в докерфайле требуется прописать директиву USER.

▼ Должно получиться примерно следующее:

```
FROM apache/airflow:2.7.1

WORKDIR /opt/airflow

USER root # Переключаемся на админского пользователя, тк этого требует apt
RUN apt update && apt -y install procps default-jre # это требуется, чтобы в контейнере airflow мог запускаться spark-job (spark-submit)

USER airflow # Переключаемся обратно на юзера airflow, чтоб сам сервис работал корректно и не сломались пермишены
COPY ./dags/* ./dags/
COPY ./spark/* ./spark/

RUN pip3 install apache-airflow-providers-apache-spark
...
```

▼ Далее необходимо добавить в существующий `docker-compose.yml` кусок, отвечающий за Spark, чтобы он поднимался вместе с остальными сервисами (не забыть добавить папку spark в volumes). Требуется мастер и воркер, пример:

```
...
...
volumes:
  - ./dags:/opt/airflow/dags
  - ./logs:/opt/airflow/logs
  - ./spark:/opt/airflow/spark # добавляем новую точку монтирования
...
...
spark-master: # мастер нода в кластере Spark
  container_name: spark-master # ОБЯЗАТЕЛЬНО, т.к. на этот параметр многое завязано. В докере container_name = hostname внутри контейнера, т.е. это доменное имя виртуального хоста
  image: apache/spark:latest # на момент написания текста лабораторной latest = 3.5.0
  ports:
    - '4040:8080' # Spark WebUI port
    - '7077:7077' # Spark connection port
  command: '/opt/spark/bin/spark-class org.apache.spark.deploy.master.Master' # команда для запуска сервиса в режиме мастер

spark-worker: # воркер в кластере Spark
  container_name: spark-worker
  image: apache/spark:latest
  depends_on:
    - spark-master
  ports:
    - "7000:7000"
  command: '/opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077' # команда для запуска сервиса в режиме воркер и подключения к мастеру
...
...
```

▼ В админке Airflow требуется руками создать подключение к кластеру Spark (*Admin* → *Connections* → + *New*)

- *Имя подключения* - любое, должно совпадать с именем, которое будет использоваться в параметрах SparkSubmitOperator (если будет)
- *Тип подключения* - Spark
- *Хост* - spark://spark-master (должно совпадать с container\_name в файле композы)
- *Порт* - 7077

Airflow

DAGs

Cluster Activity

Datasets

Security

Browse

Admin

Docs

Edit Connection

Connection Id \*

spark\_local

Connection Type \*

Spark

Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.

Description

Host

spark://spark-master

Port

7077

Extra

{}

Save

Test

- Основные требования:
- Очередность деплоя сервисов: *postgres* → *spark-master* → *spark-worker* → *airflow-init* → *airflow*
  - В рамках *spark-job* должна использоваться питоновская библиотека *pyspark* (*SparkSession*)

▼ Пример DAGa и spark-job:

```
...
default_args = {
    ...
}

dag = DAG(
    ...
)

spark_job = SparkSubmitOperator(task_id='my_spark_job', # имя любое
                                application=f'/opt/airflow/spark/my_srcipt.py' # путь до скрипта, который надо прогнать
                                name='my_spark_job', # имя любое
                                conn_id='spark_local', # должно совпадать с именем, заданным через админку при создании подключения
                                dag=dag)

spark_job

# Import the necessary modules
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

conf = SparkConf().setAppName("My PySpark App").setMaster("spark://spark-master:7077") # опять-таки должно совпадать с container_name мастер-ноды
sc = SparkContext(conf=conf)
# Start the SparkSession
spark = SparkSession.builder \
    .config(conf=conf) \
    .getOrCreate()

# Тут тело скрипта, что вообще требуется выполнить/посчитать и тд

# Stop the SparkSession
spark.stop()
```

Задание можно считать успешно выполненным, если а) DAG завершился успешно и б) в админке Spark ( <http://localhost:4040> ) видно воркер и выполненную задачу (задачи).

▼ Пример:

```
URL: spark://172.18.0.6:7077
Alive Workers: 1
Cores in use: 6 Total, 0 Used
Memory in use: 1024.0 MiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE
```

▼ Completed Applications (1)

Ссылка на Gitlab-репозиторий от команды, где содержится

- директория или файл, содержащая DAG

<https://spark.apache.org/docs/latest/submitting-applications.html>

<https://www.manning.com/books/data-analysis-with-python-and-pyspark>

<https://medium.com/codex/executing-spark-jobs-with-apache-airflow-3596717bbbe3>

<https://medium.com/@mehmood9501/using-apache-spark-docker-containers-to-run-pyspark-programs-using-spark-submit-afd6da480e0f>

<https://becominghuman.ai/real-world-python-workloads-on-spark-standalone-clusters-2246346c7040>

<https://medium.com/swlh/using-airflow-to-schedule-spark-jobs-811becf3a960#> (пример не подойдет для текущей работы, так что просто для погружения)

[https://gitlab.com/itmo\\_samon/airflow-spark-example](https://gitlab.com/itmo_samon/airflow-spark-example) (рабочий пример для наглядности)