

## Практическая работа №4

### Знакомство с функционалом сервиса Apache Airflow

#### Цель работы

Получение практических навыков по разворачиванию Apache Airflow с помощью Docker.

#### Индивидуальное задание

Выполнить разворачивание Apache Airflow с помощью Docker.

#### Ход выполнения работы

Для разворачивания Apache Airflow с помощью Docker объёма памяти по умолчанию, доступного для Docker в macOS, часто недостаточно. Чтобы решить эту проблему, в настройках Docker выделяется не менее 4 ГБ памяти, в данном случае 8 ГБ (рис. 1).



Рисунок 1 – Выделение памяти для Docker

После успешного скачивания файла `docker-compose.yaml` необходимо изменить его под текущий проект.

Чтобы отменить загрузку примеров DAG файлов необходимо изменить: `AIRFLOW__CORE__LOAD_EXAMPLES () : 'false'` (рис. 2).

```
docker-compose.yaml ×
Users > isherz > docker-compose.yaml
47   x-airflow-common:
54     environment:
57       AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql://psys
58       AIRFLOW__CELERY__RESULT_BACKEND: db+postgresql://ai
59       AIRFLOW__CELERY__BROKER_URL: redis://:@redis:6379/0
60       AIRFLOW__CORE__FERNET_KEY: ''
61       AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: 'true'
62       AIRFLOW__CORE__LOAD_EXAMPLES: 'false'
63       AIRFLOW__API__AUTH_BACKENDS: 'airflow.api.auth.backe
64       # yamllint disable rule:line-length
```

Рисунок 2 – Отключение загрузок DAG файлов

Чтобы успешно сохранять файлы в папку data необходимо добавить в подразделе volumes раздела x-airflow-common строчку (рис. 3):

- \${AIRFLOW\_PROJ\_DIR:-.}/data:/opt/airflow/data

```
docker-compose.yaml ×
Users > isherz > docker-compose.yaml
47   x-airflow-common:
54     environment:
75     volumes:
76       - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/dags
77       - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/logs
78       - ${AIRFLOW_PROJ_DIR:-.}/config:/opt/airflow/config
79       - ${AIRFLOW_PROJ_DIR:-.}/plugins:/opt/airflow/plugins
80       - ${AIRFLOW_PROJ_DIR:-.}/data:/opt/airflow/data
81     user: "${AIRFLOW_UID:-50000}:0"
```

Рисунок 3 – Параметры volumes

Для взаимодействия с БД Postgres на локальной машине, необходимо указать локальный порт (рис. 4).

```

  >> Run All Services
services:
  > Run Service
  postgres:
    image: postgres:13
    ports:
      - "5433:5432"
    environment:
      POSTGRES_USER: airflow
      POSTGRES_PASSWORD: airflow
      POSTGRES_DB: airflow

```

Рисунок 4 – Указание локального порта

Где первое значение (5433) – номер порта на локальной машине, а второе значение (5432) – стандартный номер порта Postgres внутри контейнера.

В подразделе `command` раздела `airflow-init` необходимо добавить директорию `data` (рис. 5).

```

243     fi
244     mkdir -p /sources/logs /sources/dags /sources/plugins /sources/data
245     chown -R "${AIRFLOW_UID}:0" /sources/{logs,dags,plugins,data}
246     exec /entrypoint airflow version
247 # yamllint enable rule:line-length

```

Рисунок 5 – Добавление директории data

Также во всех разделах заменяются `restart:always` на `restart:unless-stopped`, чтобы контейнеры самостоятельно не запускались при каждом открытии Docker Desktop (рис. 6).

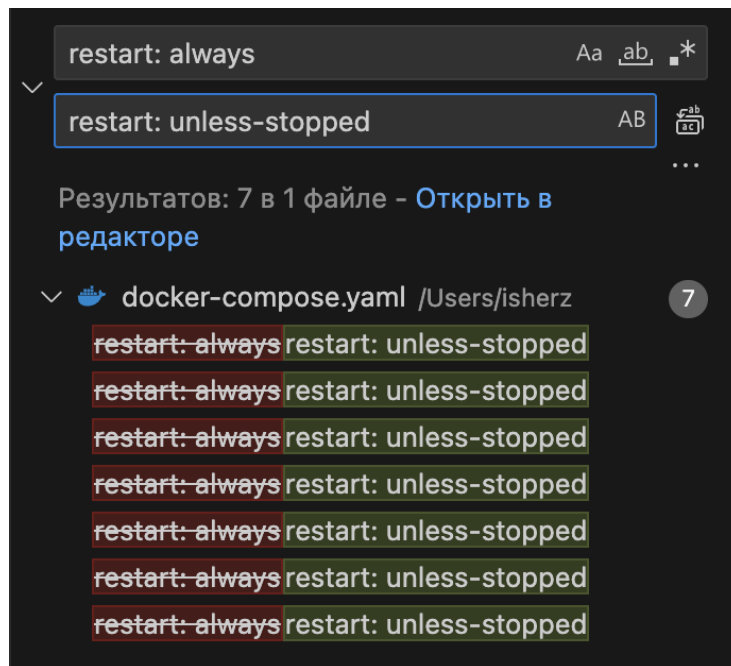


Рисунок 6 – Изменение параметров запуска

Далее необходимо файл «docker-compose.yml» поместить в папку проекта, где необходимо создать дополнительные папки (рис. 7).

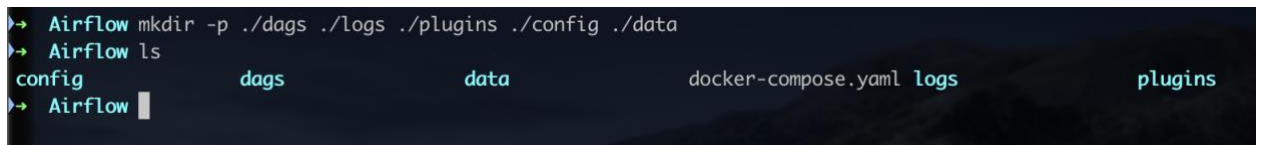


Рисунок 7 - Создание директорий

Для корректной работы Airflow необходимо создать «.env» (рис. 8).

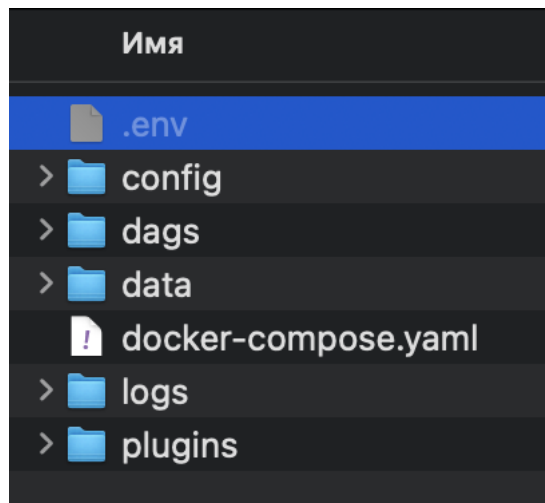


Рисунок 9 – Создание «.env» файла

Во всех операционных системах необходимо запустить миграции базы данных и создать первую учетную запись пользователя (рис. 10). Для этого используется команда: `docker compose up airflow-init`

```

➤ Airflow docker compose up airflow-init
[+] Running 42/42
✓ postgres Pulled
✓ redis Pulled
✓ airflow-init Pulled

[+] Running 5/5
✓ Network airflow_default Created
✓ Volume "airflow_postgres-db-volume" Created
✓ Container airflow-redis-1 Created
✓ Container airflow-postgres-1 Created
✓ Container airflow-airflow-init-1 Created
Attaching to airflow-init-1
airflow-init-1 | The container is run as root user
airflow-init-1 |
airflow-init-1 | DB: postgresql+psycopg2://airflow

```

Рисунок 10 – Запуск миграций

Для запуска всех сервисов приложения воспользуется команда:  
`docker compose up`.

Результат выполнения команды представлен на рисунке 11.

```

➤ Airflow docker compose up
[+] Running 7/7
✓ Container airflow-redis-1 Running
✓ Container airflow-postgres-1 Running
✓ Container airflow-airflow-init-1 Created
✓ Container airflow-airflow-worker-1 Created
✓ Container airflow-airflow-scheduler-1 Created
✓ Container airflow-airflow-triggerer-1 Created
✓ Container airflow-airflow-webserver-1 Created

```

Рисунок 11 – Запуск сервисов приложения

Список запущенных контейнеров можно посмотреть в Docker Desktop (рис. 12 и 13).

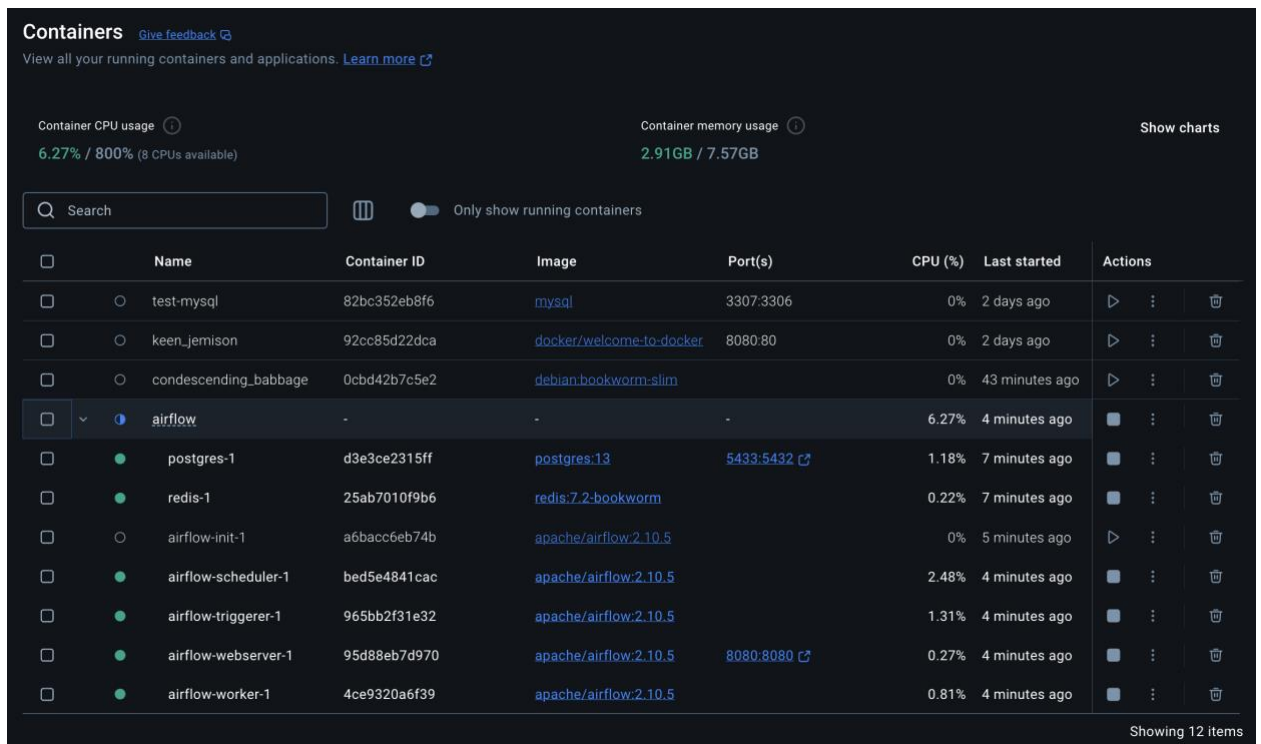


Рисунок 12 – Список запущенных контейнеров в Docker Desktop

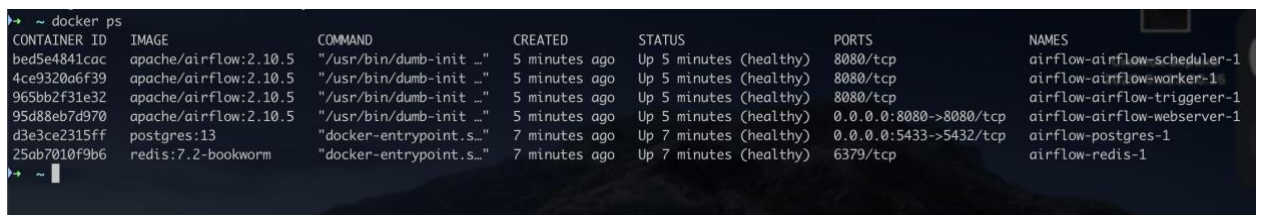


Рисунок 13 – Список запущенных контейнеров через команду

После успешного запуска всех сервисов Airflow, можно заметить, что контейнер airflow-init-1 не запущен. Это связано с тем, что он автоматически останавливается после успешного выполнения.

Далее выполняется переход к веб-интерфейсу и выполняется авторизация. После авторизации открывается веб-интерфейс для взаимодействия с Airflow (рис. 14).

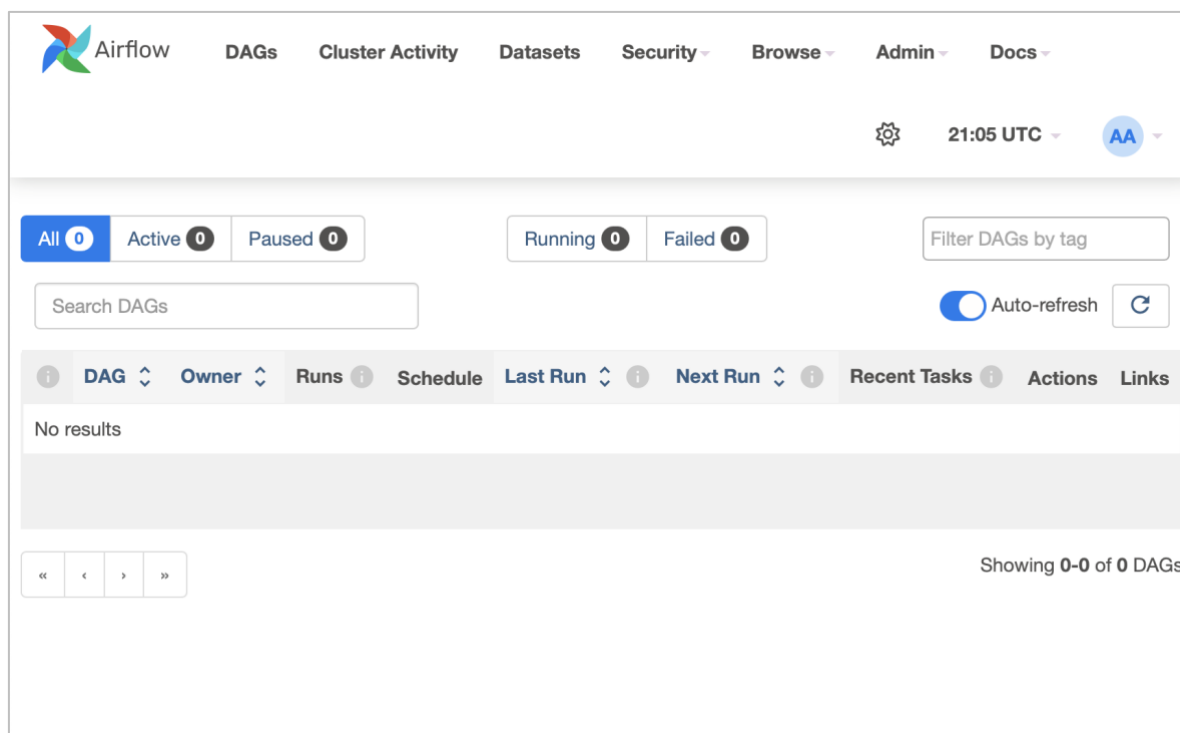


Рисунок 14 – Веб-интерфейс Airflow

## Вывод

В ходе выполнения работы я получил практические навыки по разворачиванию Apache Airflow с помощью Docker, настроил параметры конфигурации и обеспечил корректную работу с PostgreSQL. Внес изменения в docker-compose.yaml, отключил загрузку примеров DAG, настроил volumes для сохранения данных и параметры перезапуска контейнеров. После запуска сервисов и выполнения миграций успешно авторизовался в веб-интерфейсе Airflow.