Практическая работа №4

Знакомство с функционалом сервиса Apache Airflow

Цель работы

Получение практических навыков по развертыванию Apache Airflow с помощью Docker.

Индивидуальное задание

Выполнить развертывание Apache Airflow с помощью Docker.

Ход выполнения работы

Для развертывания 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).

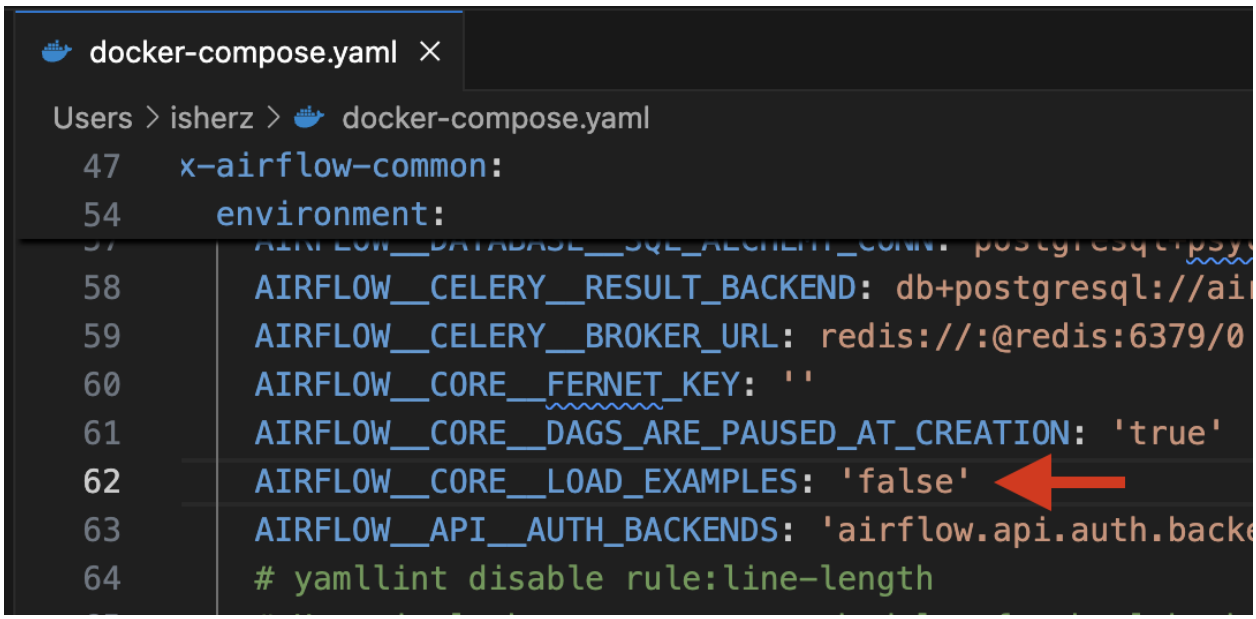

Рисунок 2 – Отключение загрузок DAG файлов

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
```

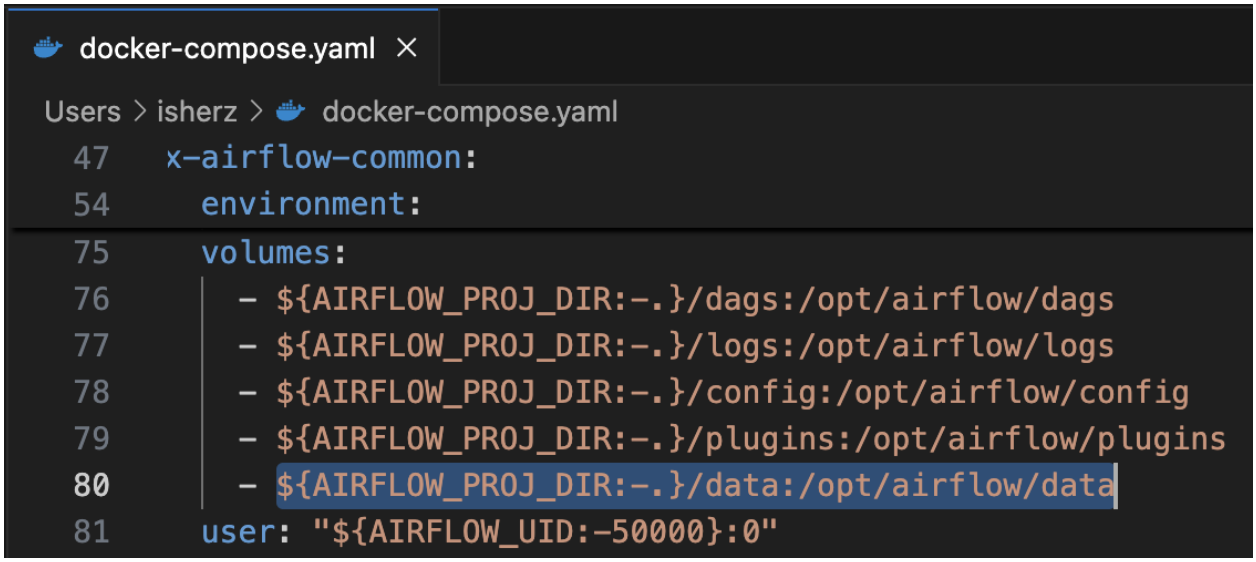

Рисунок 3 – Параметры volumes

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

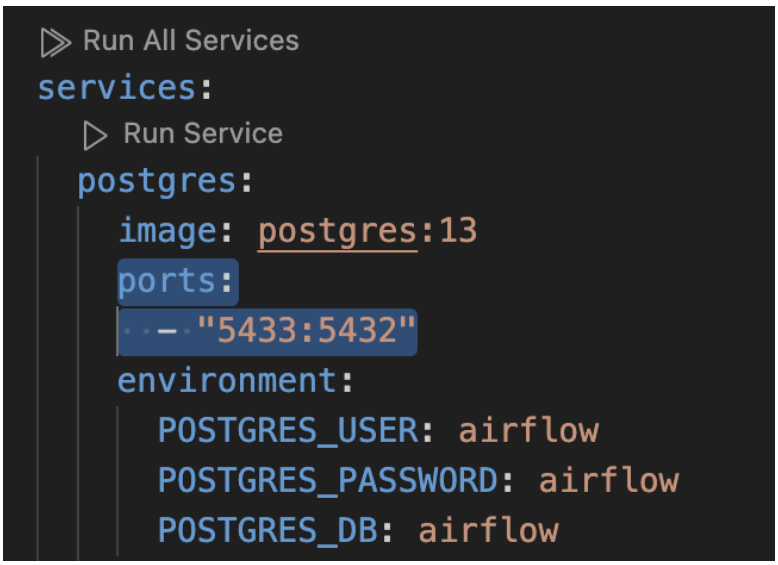

Рисунок 4 – Указание локального порта

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).

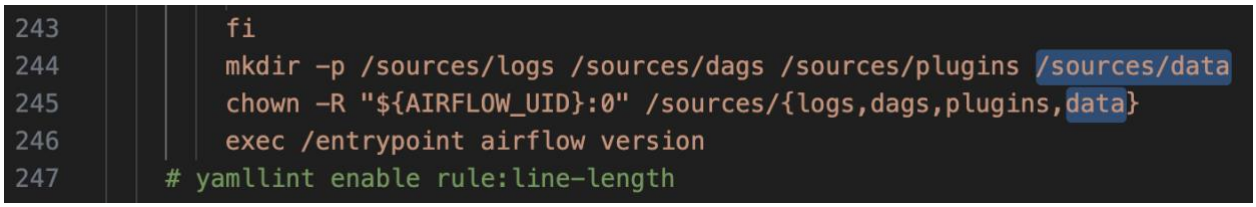

Рисунок 5 – Добавление директории data

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

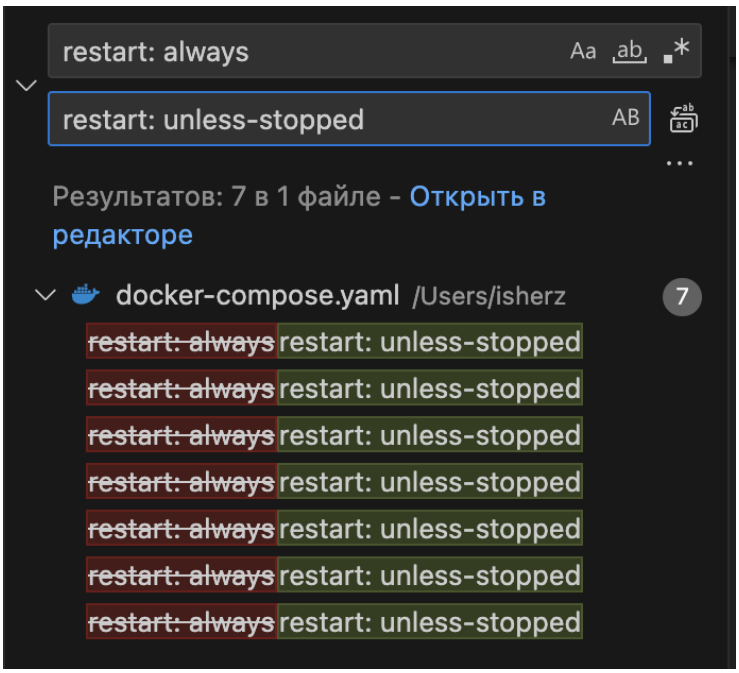

Рисунок 6 – Изменение параметров запуска

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

Рисунок 7 - Создание директорий

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

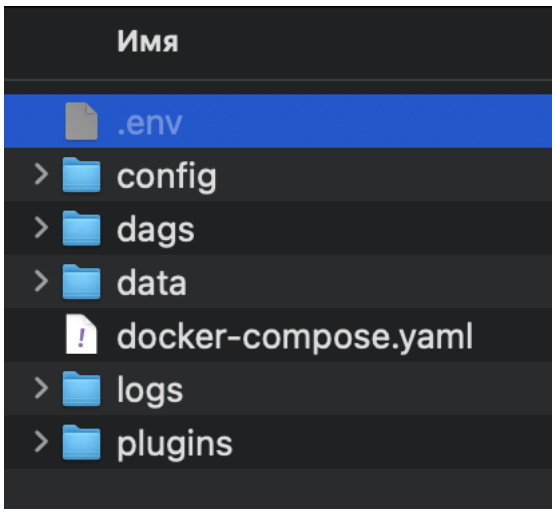

Рисунок 9 – Создание «.env» файла

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`

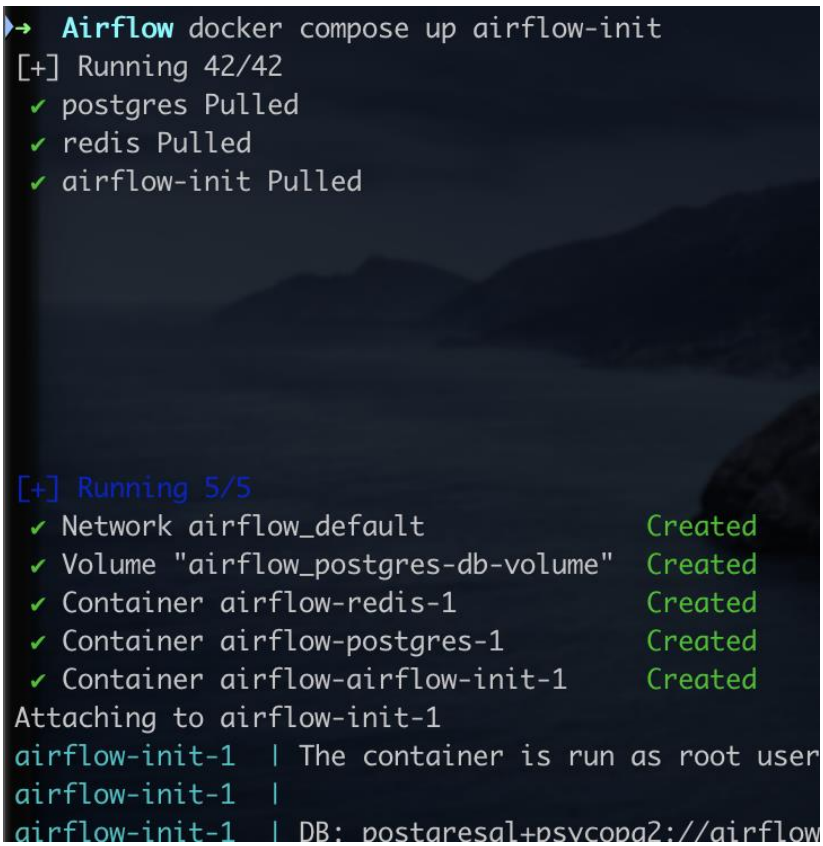

Рисунок 10 – Запуск миграций

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

Результат выполнения команды представлен на рисунке 11.

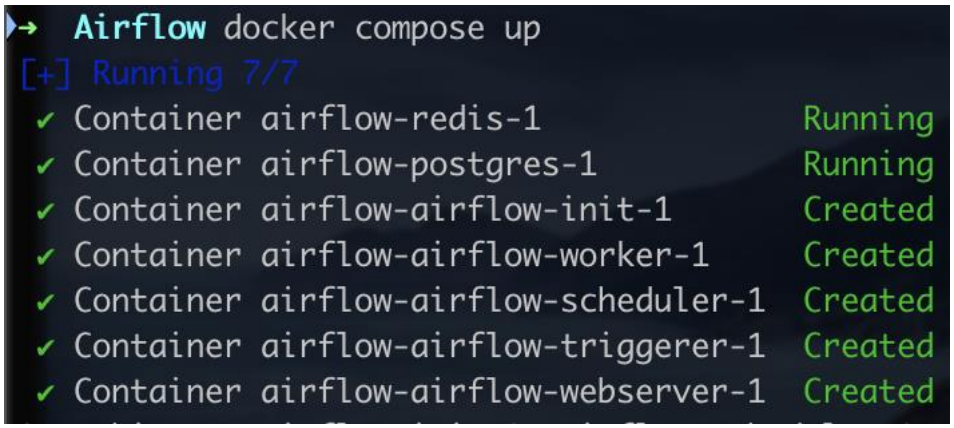

Рисунок 11 – Запуск сервисов приложения

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

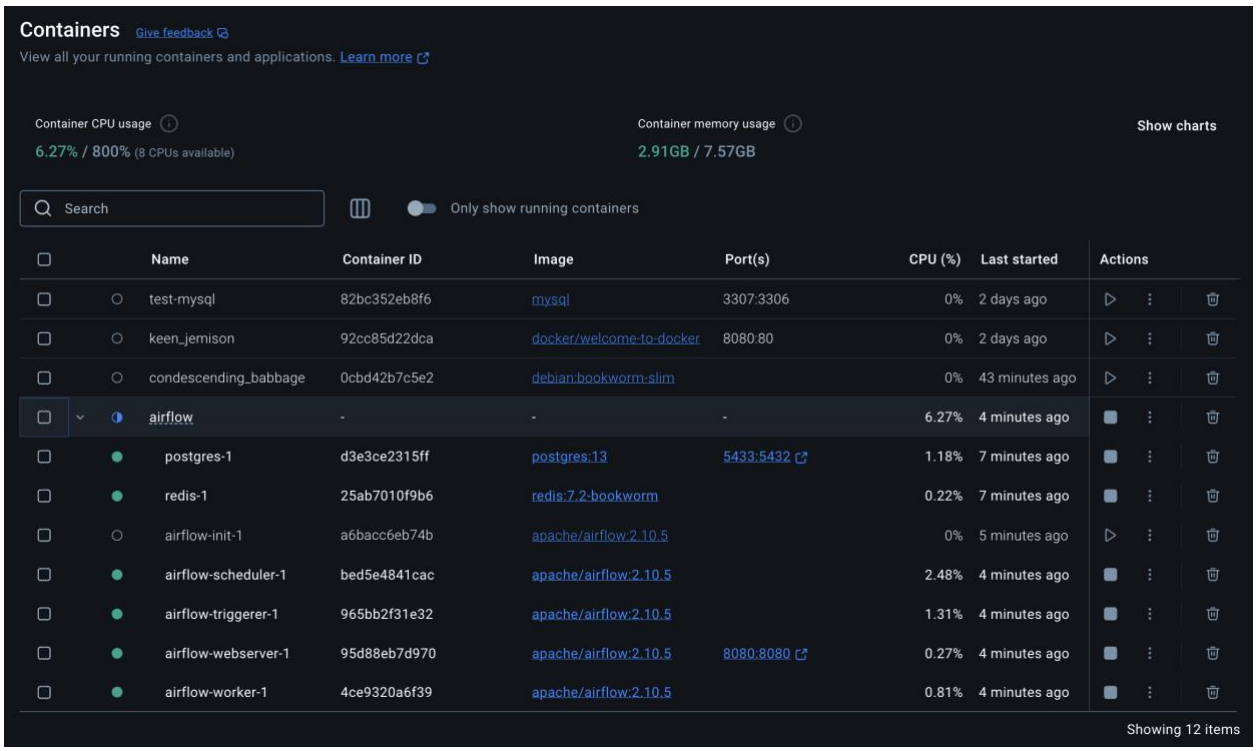

Рисунок 12 – Список запущенных контейнеров в Docker Desktop

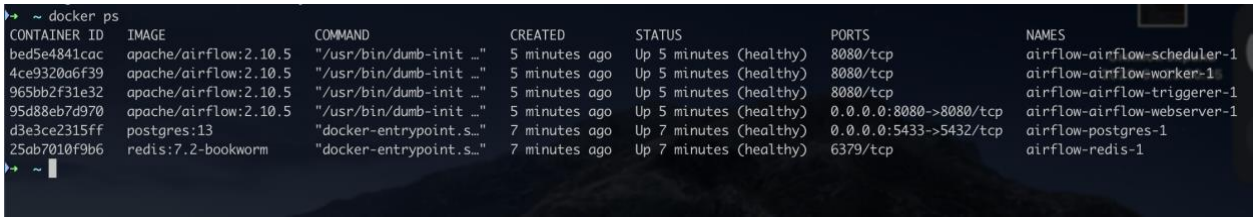

Рисунок 13 – Список запущенных контейнеров через команду

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).

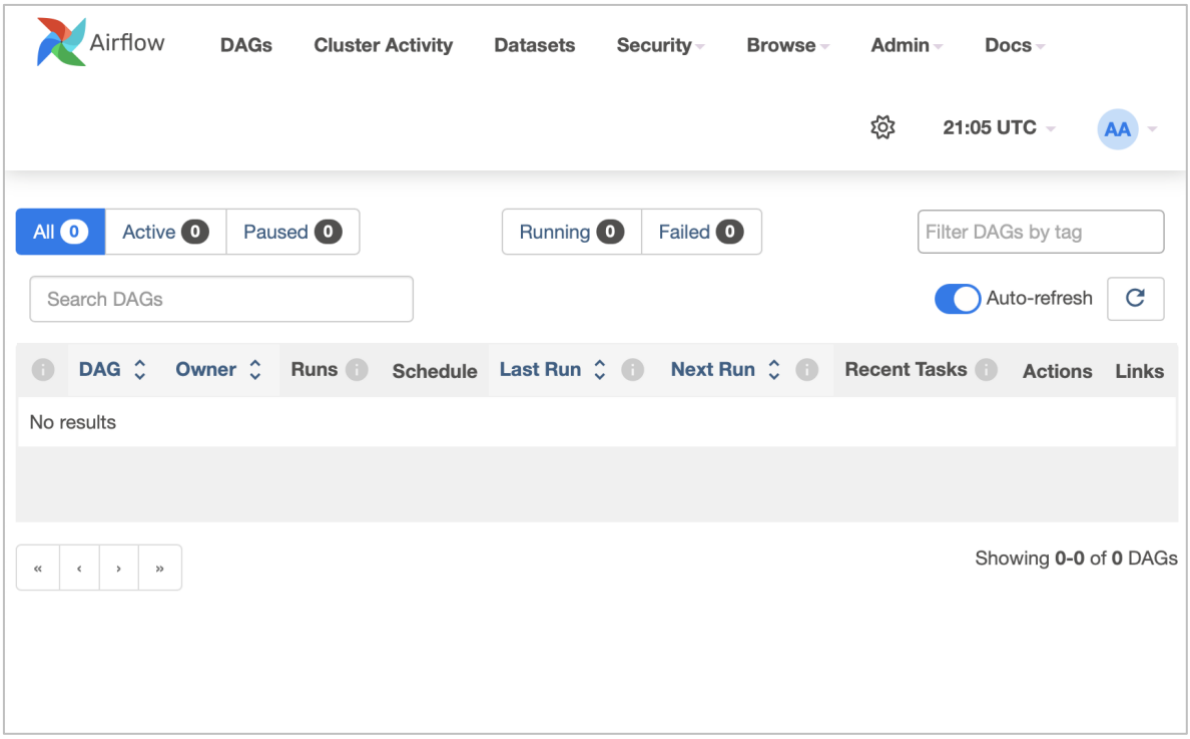

Рисунок 14 – Веб-интерфейс Airflow

Вывод

В ходе выполнения работы я получил практические навыки по развертыванию 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.