

# Docker

- Что такое докер и зачем он нужен
- Подготовка
  - Получаем локальный репозиторий и собираем папку docker:
- Работа с Docker
  - Установка Docker
  - Если у Вас процессор "М" серии
  - Запуск Docker
    - Сборка
    - Запуск
    - Остановка
    - Переборка образов
    - Misc
  - Connection
  - Pools
    - dbt dags - фильтрации
- Популярные ошибки
  - 1) failed to solve: no match for platform in manifest
  - 2) Загрузки ods слоя из постгреса зависают в состоянии scheduled
  - 3) Повышение потребления оперативной памяти

## Что такое докер и зачем он нужен

blocked URL

Определение [Докера](#) в Википедии звучит так:



программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы; позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, а также предоставляет среду по управлению контейнерами.

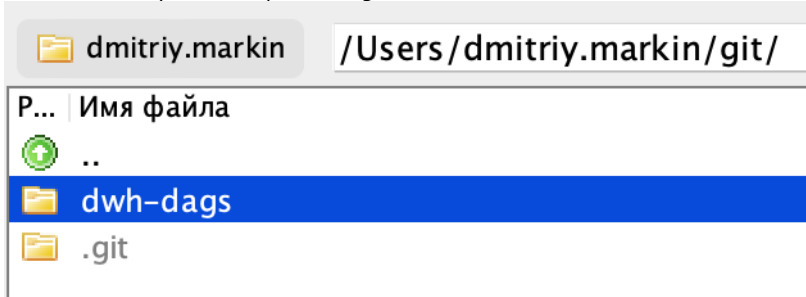
Подробнее можно почитать [тут](#) и [тут](#).

## Подготовка

### Получаем локальный репозиторий и собираем папку docker:

Как настроить гит, забрать проект, а так же катнуть MR подробно расписано [здесь](#)

1. Скачиваем актуальный проект из gitlab



2. Копируем из папки `dwh-dags` необходимые файлы и папки в `docker`;  
Создаем свои файлы `docker-compose.yaml` и `.env`. Они требуются для создания `connection` в `airflow`. В репозитории хранятся примеры файлов, которые называется `docker-compose.example.yaml` и `.env.template`. Перед следующими шагами необходимо создать свои файлы `docker-compose.yaml` и `.env` и для примера скопировать в них содержимое из файлов `docker-compose.example.yaml` и `.env.template` соответственно. Файлы `docker-compose.yaml` и `.env` добавлены в `.gitignore`, поэтому их изменения не будут отслеживаться в репозитории. Примеры файлов не удаляйте и не изменяйте!

Папка `docker` будет выглядеть так:

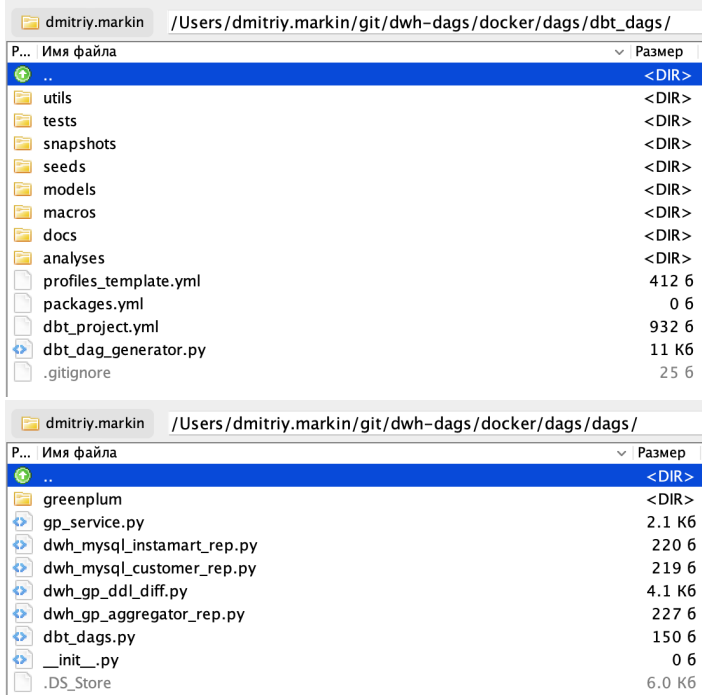


3. Наполняем папку `dwh-dags/docker/dags/`

- а. Из `dwh-dags/` копируем папки **`config`**, **`dags`**, **`dbt_dags`**, **`libs`** в директорию `dwh-dags/docker/dags/`



- б. В скопированных папках `dwh-dags/docker/dags/dags/` и `dwh-dags/docker/dags/dbt_dags/` оставляем только нужные нам для тестирования файлы. У меня в итоге выглядит так:



## Работа с Docker

### Установка Docker

Скачать docker на [официальном сайте](#)

## Если у Вас процессор "М" серии

Желательно обновиться до последней версии Mac OS

Требуется установить rosetta

```
softwareupdate --install-rosetta
```

И поставить галочку "Use rosetta..." в интерфейсе докера

### ☒ Use Rosetta for x86/amd64 emulation on Apple Silicon

Turns on Rosetta to accelerate x86/amd64 binary emulation on Apple Silicon. Note - you must have the Virtualization Framework enabled (via the toggle on the General panel).

## Запуск Docker

### Сборка

Печальная новость: 30.05.2024 Docker Hub прекратил работу в России. Для того, чтобы все образы собрались, необходимо сделать несколько правок в файлах: Dockerfile.yml, Docker-compose.yml

В Dockerfile.yml первую строчку меняем на:

```
FROM dreg.sbmt.io/dhub/apache/airflow:2.2.0-python3.8
```

То есть берем образы не с Docker Hub, а с внутреннего Docker registry.

Далее в Docker-compose.yml необходимо для postgres и redis также указать dreg

```
# postgres:
  image: dreg.sbmt.io/dhub/library/postgres:13
# redis:
  image: dreg.sbmt.io/dhub/library/redis:latest
```

Для сборки необходимых docker образов необходимо

```
# docker. , cd /Users/dmitriy.markin/git/dwh-dags/docker/
cd docker
#
cp ../requirements.txt ./
#
docker-compose build
```

Или если установлена утилита make - просто выполнить

```
make docker-build
```

## Запуск

Для запуска кластера airflow надо выполнить

```
docker-compose up -d
```

или

```
make docker-start
```

Будет запущено несколько docker контейнеров. В течении 1-2 минут по адресу <http://localhost:8080/> будет доступен airflow web server. Логин/пароль: airflow/airflow

В течении следующих 1-2 минут автоматически создаются connection. Для того, чтобы ваш dag появлялся в локальном airflow надо просто скопировать файл dag-а и все требуемые ресурсы (например кастомные операторы или конфига и папку `docker /libs/` и `docker/config` соответственно) Если вы меняете конфиг или оператор - оно не пересчитывается (airflow автоматически пересчитывает только папку с dag-ами) Для того чтобы обновилось все - просто надо перезапустить `airflow scheduler`:

```
docker-compose restart airflow-scheduler
```

## Остановка

Можно все остановить

```
docker-compose down
```

или

```
make docker-stop
```

При этом все останутся контейнеры и volume

Чтобы остановить все с очисткой

```
docker-compose down --remove-orphans --volumes
```

## Переборка образов

Чтобы все пересобрать - надо сначала все удалить а потом просто заново все собрать. Лучше это делать как только начинаете новую задачу (подтягиваем последние изменения из master ветки и пересобираем airflow с актуальными используемыми версиями пакетов)

```
make docker-clean  
make docker-build
```

## Misc

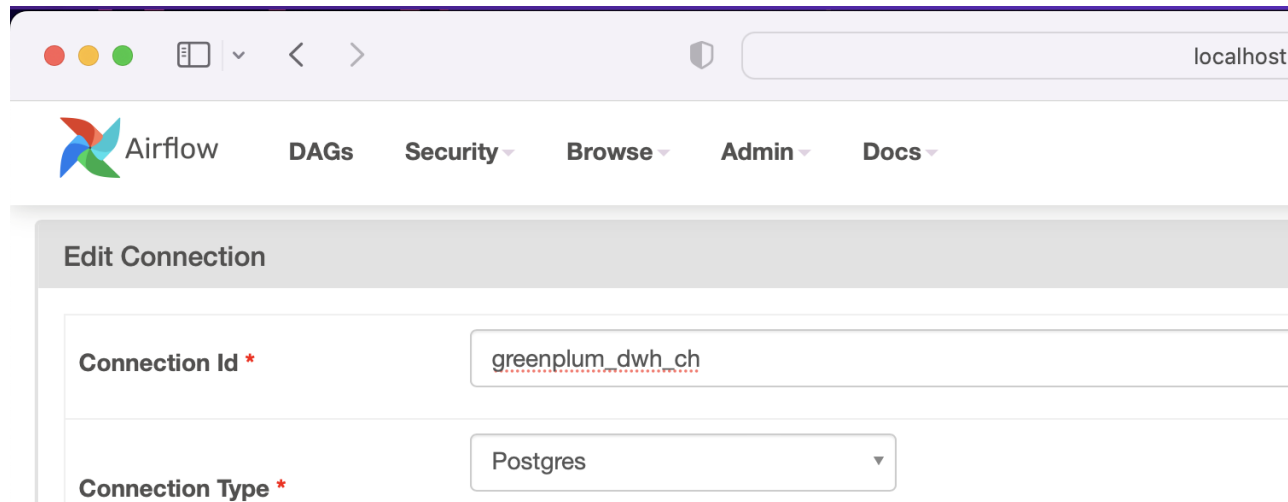
Часто бывает что остается куча docker образов или контейнеров которые уже не нужны. Самый простой способ все почистить

```
docker container prune  
docker image prune
```

## Connection

Далее, в развернутом airflow на <http://localhost:8080/> dbt dags не будут собираться из-за отсутствия коннектора greenplum\_dwh\_ch.

Если есть УЗ dwh\_ch, используем её. Если нет - создаем пустой коннект с Connection Id = greenplum\_dwh\_ch и Connection Type = "Postgres". Остальные поля оставляем пустыми.



**Edit Connection**

**Connection Id \*** greenplum\_dwh\_ch

**Connection Type \*** Postgres

Что бы не прописывать коннекторы каждый раз при поднятии локального эирфлоу, их нужно прописать в *.env* и *docker-compose.yml*

### **.env**

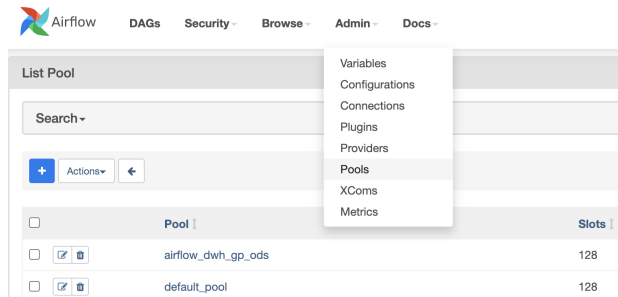
```
GREENPLUM_CONNECTION_HOST=c-c9qfrnqkuod6v7ive03r.rw.mdb.yandexcloud.net
GREENPLUM_CONNECTION_PORT=5432
GREENPLUM_CONNECTION_LOGIN=< >
GREENPLUM_CONNECTION_PASSWORD=< >
GREENPLUM_CONNECTION_DATABASE=dwh
```

## docker-compose.yaml

```
airflow connections delete 'greenplum_dwh'
airflow connections add 'greenplum_dwh' \
  --conn-type 'postgres' \
  --conn-login '${GREENPLUM_CONNECTION_LOGIN}' \
  --conn-password '${GREENPLUM_CONNECTION_PASSWORD}' \
  --conn-host '${GREENPLUM_CONNECTION_HOST}' \
  --conn-port '${GREENPLUM_CONNECTION_PORT}' \
  --conn-schema '${GREENPLUM_CONNECTION_DATABASE}'
airflow connections delete 'greenplum_dwh_ch'
airflow connections add 'greenplum_dwh_ch' \
  --conn-type 'postgres'
  --conn-login '' \
  --conn-password '' \
  --conn-host '' \
  --conn-port '' \
  --conn-schema ''
airflow connections delete 'greenplum_dwh_ch_hr'
airflow connections add 'greenplum_dwh_ch_hr' \
  --conn-type 'postgres'
  --conn-login '' \
  --conn-password '' \
  --conn-host '' \
  --conn-port '' \
  --conn-schema ''
airflow pools delete 'airflow_dwh_gp_ods'
airflow pools set 'airflow_dwh_gp_ods' 128 ''
```

## Pools

Для того, чтобы работали даги ods слоя, загружающие данные из постгресовых источников, необходимо добавить пул для ods слоя



- 1) Перейти по ссылке <http://localhost:8080/pool/list/> (Admin > Pools)
- 2) Добавить pool `airflow_dwh_gp_ods` с количеством слотов 128 (описание любое)

## dbt dags - фильтрации

Необходимо использовать если компьютер испытывает сильную нагрузку или возникает ошибка по таймауту:



DagBag import timeout for /opt/airflow/dags/dags/dbt\_dags.py after 30.0s. Please take a look at these docs to improve your DAG import time

Для фильтрации нужных дагов необходимо в папке **dbt\_dags** создать папку+файл: **target/filtered\_models.yml** и в нем прописать:

dbt\_dags/target/filtered\_models.yml

```
included_models:          #
- s_product__inst
  - h_product
  - s_offer
  etc
```

## Популярные ошибки

### 1) failed to solve: no match for platform in manifest

возникает на mac m2

```
=> [flower internal] load .dockerignore
=> => transferring context: 64B
=> [airflow-init internal] load .dockerignore
=> => transferring context: 64B
=> [airflow-init internal] load build definition from Dockerfile
=> => transferring dockerfile: 575B
=> [airflow-webserver internal] load build definition from Dockerfile
=> => transferring dockerfile: 575B
=> [airflow-webserver internal] load .dockerignore
=> => transferring context: 64B
=> [airflow-worker internal] load .dockerignore
=> => transferring context: 64B
=> [airflow-worker internal] load build definition from Dockerfile
=> => transferring dockerfile: 575B
=> ERROR [flower internal] load metadata for docker.io/apache/airflow:2.2.0-python3.8
=> [airflow-scheduler auth] apache/airflow:pull token for registry-1.docker.io
-----
> [flower internal] load metadata for docker.io/apache/airflow:2.2.0-python3.8:
-----
```

Решается добавлением в *docker-compose.yml* свойства для airflow компонентов "platform: linux/amd64" (как на скрине ниже). Так мы явно указываем архитектуру, в которой будем производить сборку.

```
airflow-webserver:
  <<: *airflow-common
  platform: linux/amd64
  command: webserver
  ports:
    - 8080:8080
  healthcheck:
    test: [ "CMD", "curl", "--fail", "http://localhost:8080/health" ]
    interval: 10s
    timeout: 10s
    retries: 5
  restart: always
```

### 2) Загрузки ods слоя из постгреса зависают в состоянии scheduled

Добавьте pool согласно [инструкции](#).

### 3) Повышение потребления оперативной памяти

С обновленным Docker Desktop запускает процессы docker-scout, которые потребляют по 5-10 гб оперативной памяти. Это часть экспериментальной фишки, которая включена по дефолту. Чтобы выключить снимаем галочки, выключаем SBOM indexing и перезапускаемся. (инфа от [Неизвестный пользователь \(nelyubov.ds\)](#) из [треда](#) )

