



## Инструкция по установке

---



### Windows

#### Установка Airflow с помощью Docker

([официальная инструкция](#))

1. Устанавливаем shell (интерпретатор командной строки), если не установлен: например, [GitBash](#) или [PowerShell](#).
2. Устанавливаем [Docker Desktop](#) для Windows (вместе с ним установится [docker-compose](#)) для работы контейнерами.
3. Открываем shell, переходим в папку (`cd <путь>`), где хотим развернуть Airflow, например, в домашнюю директорию (`cd ~`), и выполняем команды:

- `mkdir airflow-docker`
- `cd airflow-docker`
- `curl -O https://airflow.apache.org/docs/apache-airflow/stable/docker-compose.yaml`
- `mkdir dags logs plugins`
- `echo -e "AIRFLOW_UID=500000" > .env`
- `docker-compose up airflow-init`
- `docker-compose up`

4. Данными командами мы запустили docker-контейнеры с компонентами Airflow (webserver, scheduler и другие). Проверим, что всё работает, с помощью команды

- `docker ps`

Все контейнеры должны быть в статусе *healthy*.

- 4.1. Обратите внимание на контейнер **worker** и его id — в нём будет выполняться код DAG и сохранятся результаты расчётов в \*.csv файлах. Для работы с файлами нужно войти в `bash` контейнера и далее выполнять команды в нём:
    - `docker exec -it <container_id> bash`
  5. В браузере набираем <http://localhost:8080/> и видим интерфейс Airflow.
- 



### Linux



### macOS

#### Установка Airflow с помощью pip

([официальная инструкция](#))

Официальная инструкция (по ссылке выше) для запуска Airflow предлагает базовую конфигурацию с экзекьютором [SequentialExecutor](#) и мета-базой SQLite. Можно следовать ей для настройки Airflow, этого будет достаточно для практики.

Ниже предлагается расширенный способ установки с дополнительными настройками. Это позволит лучше разобраться с компонентами Airflow.

## Шаг 1. Окружение и airflow package

Откроем **Terminal**, создадим виртуальное окружение Python (требуется версия Python **3.6** или выше) и активируем его (отключить виртуальное окружение можно с помощью команды **deactivate**):

- `python3 -m venv .venv_airflow`
- `source .venv_airflow/bin/activate`

Теперь в командной строке видим префикс с именем окружения:

- `(.venv_airflow)...`

Создадим переменную окружения с директорией для установки Airflow, например:

- `export AIRFLOW_HOME=~/.airflow`

С этой папкой будем постоянно работать в дальнейшем.

Укажем версии Airflow, Python и установим Airflow с учётом зависимостей (для заданных версий Airflow и Python есть перечень необходимых пакетов конкретных версий, доступный по ссылке `CONSTRAINT_URL`):

- `AIRFLOW_VERSION=2.2.5`
- `PYTHON_VERSION="$(python --version | cut -d " " -f 2 | cut -d "." -f 1-2)"`
- `CONSTRAINT_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt"`
- `pip install "apache-airflow==${AIRFLOW_VERSION}" --constraint "${CONSTRAINT_URL}"`

Такая установка с фиксированными версиями позволяет добиться воспроизводимости установки. У Airflow много зависимостей, поэтому в терминале при установке будет много информации. После установки Airflow можете посмотреть зависимости с помощью команды **pip freeze**.

После установки пакета `apache-airflow` в виртуальном окружении будет доступна команда **airflow**. Запустите её без параметров, чтобы увидеть список доступных команд. Например, основную информацию об Airflow можно узнать с помощью команды **airflow info**, а список наиболее полезных команд — **airflow cheat-sheet**. [Подробное знакомство с Airflow CLI](#).

## Шаг 2. База данных и airflow.cfg

Выполним инициализацию базы данных ([подробнее](#)):

- `airflow db init`

По умолчанию в качестве базы данных Airflow использует SQLite. Для демонстрационных возможностей этого достаточно, но в продакшн-среде лучше переключиться на PostgreSQL.

Если PostgreSQL ещё не установлен, сделаем это (для MacOS инструкция ниже, для Linux можно ориентироваться [на шпаргалку](#)).

---

Установим PostgreSQL 14:

- `brew install postgresql@14`

Проверим установленную версию:

- `postgres --version`

Создадим базу данных и запустим PostgreSQL сервер (остановить можно с помощью такой же команды с **stop**):

- `initdb /usr/local/var/postgres`
- `pg_ctl -D /usr/local/var/postgres start`  
(флаг **-D** означает, что сервер запущен как [демон](#) и будет работать фоном)

При выполнении этих команд может появиться ошибка вида:

```
initdb: error: The program "postgres" is needed by initdb but was not found in the same directory as "/usr/local/Cellar/libpq/14.3/bin/initdb"
```

Это означает, что **postgres** и **initdb** запускаются из разных мест. Для решения проблемы определим, где находится postgres и запустим **initdb** и **pg\_ctl** оттуда:

- `which postgres`  
`/usr/local/bin/postgres`
  - `/usr/local/bin/initdb /usr/local/var/postgres`
  - `/usr/local/bin/pg_ctl -D /usr/local/var/postgres start`
- 

Войдём в интерактивный терминал PostgreSQL, создадим базу и пользователя к ней:

- `psql postgres`
  - `postgres=# CREATE database airflow_metadata;`
  - `postgres=# CREATE user airflow WITH password 'airflow';`
  - `postgres=# GRANT all privileges on database airflow_metadata to airflow;`
  - `postgres=# exit`

Теперь настроим Airflow на работу с этой базой данных. Airflow хранит свои настройки в файле **airflow.cfg**, который располагается по пути `$AIRFLOW_HOME/airflow.cfg`. В частности, в начале этого файла указан путь к папке, где будут располагаться DAG'и (**dags\_folder**).

Откроем `$AIRFLOW_HOME/airflow.cfg` любым текстовым редактором и изменим значения параметров:

1. **executor** = LocalExecutor
2. **sql\_alchemy\_conn** = postgresql+psycopg2://airflow:airflow@localhost/airflow\_metadata

### 3. `load_examples` = False

Поясним каждый из параметров:

1. В стандартной конфигурации Airflow предлагает нам использовать самый простой `SequentialExecutor`, поэтому изменим на `LocalExecutor`, чтобы было ближе к продакшн-среде (кстати, если в качестве базы метаданных использовать однопоточный SQLite, то `LocalExecutor` будет работать как `SequentialExecutor`).
2. Airflow взаимодействует с базой данных при помощи фреймворка SQL Alchemy, а в качестве python-драйвера для PostgreSQL используется **psycopg2**, поэтому его необходимо не только указать в конфиге, но и установить:
  - `pip install psycopg2==2.9.3`
3. `load_examples` отвечает за загрузку примеров с DAGs, они в общем случае не нужны, но можно оставить и посмотреть примеры. Если оставить False, то при инициализации папка `dags` не создастся, и при копировании первого DAG в папку с Airflow нужно будет создать папку `dags` (`mkdir $AIRFLOW_HOME/dags`).

В случае чего, default версию конфига можно восстановить из [репозитория Airflow](#).

Наконец, инициализируем базу данных с учётом новых настроек:

- `airflow db init`

### Шаг 3. Запуск `webserver` и `scheduler`

Создадим пользователя:

- `airflow users create \`  
    `--username admin \`  
    `--password admin \`  
    `--firstname John \`  
    `--lastname Doe \`  
    `--role Admin \`  
    `--email johndoe@example.com`

Запустим веб-приложение на 8080 порту и залогинимся под созданным пользователем (если этот порт у вас занят, укажите другой свободный):

- `airflow webserver -p 8080`

Если всё настроено верно, то увидим перед собой интерфейс:

The scheduler does not appear to be running.  
The DAGs list may not update, and new tasks will not be scheduled.

## DAGs

All 0
Active 0
Paused 0

DAG	Owner	Runs	Schedule	Last Run
No results				

<<
<
>
>>

На странице видим сообщение:

"The scheduler does not appear to be running. The DAGs list may not update, and new tasks will not be scheduled."

Сообщение указывает на то, что не запущен планировщик Airflow - Scheduler. Он отвечает за обнаружение новых DAGs и планирование их запуска. Запустим планировщик командой:

- `airflow scheduler`

Можно запустить планировщик в отдельном терминале, либо использовать менеджер терминалов **tmux**. Итого, база настроена, веб-приложение и планировщик запущены.

## Импорт DAGs

После копирования файлов с готовыми к запуску DAG в директорию Airflow в папку dags, в интерфейсе Airflow можно увидеть ошибку вида:

! DAG Import Errors (1)

Broken DAG: [/Users/erinfedor/airflow/dags/first\_dag.py] Traceback (most recent call last):  
 File "<frozen importlib.\_bootstrap>", line 219, in \_call\_with\_frames\_removed  
 File "/Users/erinfedor/airflow/dags/first\_dag.py", line 2, in <module>  
 import pandas as pd  
 ModuleNotFoundError: No module named 'pandas'

Это означает, что в файле DAG была допущена ошибка. В данном случае не найден модуль `pandas`, и его необходимо установить (`pip install pandas`) в виртуальное окружение с Airflow.