Member-only story

How to Run Apache Airflow Locally

Ansam Yousry · Follow
Published in Technology Hits · 7 min read · May 26, 2023

step-by-step guide for running Apache Airflow locally, along with explanations for each step

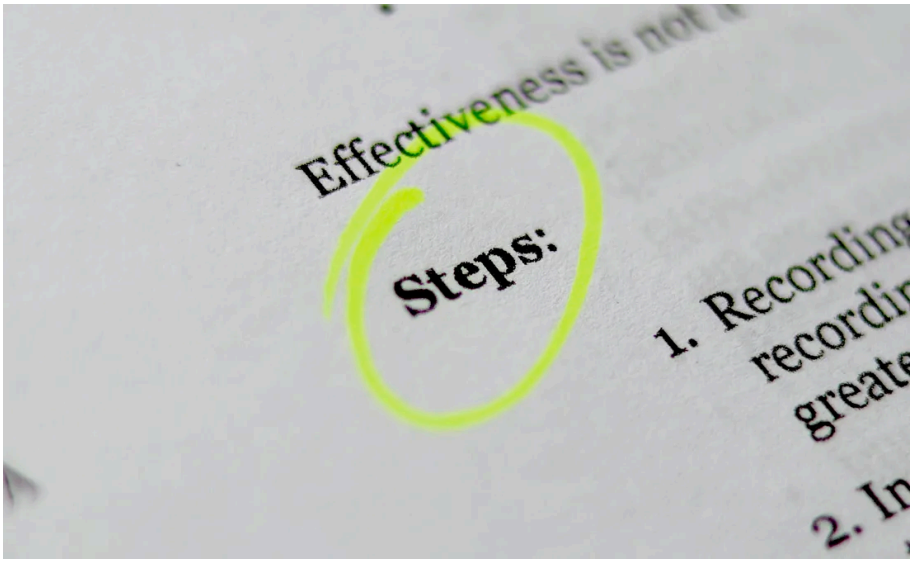


Photo by Clayton Robbins on Unsplash

Are you looking to run Apache Airflow locally on your machine? If so, In this article, we'll walk you through the steps to run Apache Airflow locally. We'll cover everything you need to know, from installing Docker and pulling the Airflow image to running Airflow containers and accessing the Airflow web UI. Every word in this article is designed to be useful to you and will save you time if you focus on the steps and explanations provided. So, let's get started and have you up and running with Apache Airflow in no time!

1- Create and start a new Docker container

(make sure that you installed Docker Desktop and run it before executing the following command).

```
docker run -it --rm -p 8080:8080 python:3.8-slim /bin/bash
```

Let me explain the meaning of each option and parameter in the `docker run` command:

- `docker run` is the command used to create and start a new Docker container.
- `-it` is a combination of two options: `-i` and `-t`. `-i` stands for "interactive" and `-t` stands for "tty" or "terminal". Together, they allow you to interact with the container through a terminal shell.
- `--rm` specifies that the container should be automatically removed when it exits. This is a good practice to avoid leaving unused containers and cluttering up your system.
- `-p 8080:8080` maps the host port 8080 to the container port 8080. This allows you to access services running inside the container via the host's IP address and port 8080.
- `python:3.8-slim` specifies the Docker image to use for the container. In this case, it is the official Python 3.8 slim image.
- `/bin/bash` is the command to run inside the container. It starts a new Bash shell session, which allows you to interact with the container's file system and execute commands.

So, when you run this command, Docker creates a new container based on the Python 3.8 slim image, starts a Bash shell session inside the container, and maps the host's port 8080 to the container's port 8080. Once you're inside the container, you can execute Python scripts, install packages, and perform other operations as needed. When you exit the Bash shell session, Docker automatically removes the container because of the `--rm` option.

```
ansamali@Ansams-Air airflow-materials % docker run -it --rm -p 8080:8080 python:3.8-slim /bin/bash
Unable to find image 'python:3.8-slim' locally
3.8-slim: Pulling from library/python
d981f2c20c93: Pull complete
1cf577bcf494: Pull complete
a3309830373f: Pull complete
c55bb8a26771: Pull complete
f3cf5aff6bf8: Pull complete
Digest: sha256:378836ee20d54fcb46d36063890d97010d118d5f8699d689e8952ff153816014
Status: Downloaded newer image for python:3.8-slim
```

The output of the Running docker run command

2- Setting the AIRFLOW_HOME Environment Variable

```
export AIRFLOW_HOME=/usr/local/airflow
```

let me explain the meaning of the `export AIRFLOW_HOME=/usr/local/airflow` command.

In Apache Airflow, the `AIRFLOW_HOME` environment variable is used to specify the directory where Airflow should store its configuration files, logs, and other data. By default, `AIRFLOW_HOME` is set to `$HOME/airflow`, which is the user's home directory.

In this case, you're setting `AIRFLOW_HOME` to `/usr/local/airflow`, which is a directory on your file system. This directory will be used by Airflow to store its configuration files, logs, and other data. By setting `AIRFLOW_HOME` it to a custom directory, you can specify a location that works best for your needs.

Setting the `AIRFLOW_HOME` environment variable is an important step when working with Apache Airflow, as it allows you to configure Airflow to use a specific directory for storing its data. This can be especially useful when running Airflow in a containerized environment, where you may want to store data outside of the container to avoid losing it when the container is deleted.

3- Install all tools and dependencies that can be required by Airflow

```
apt-get update -y && apt-get install -y wget libczmq-dev curl libssl-dev git ine
```

4- Create the user airflow

```
useradd -ms /bin/bash -d ${AIRFLOW_HOME} airflow
```

the `useradd -ms /bin/bash -d ${AIRFLOW_HOME} airflow` command is used to create a new user account named "airflow" with a home directory set to the value of the `AIRFLOW_HOME` environment variable. This ensures that the "airflow" user has the necessary permissions to access and modify the files and directories in the `AIRFLOW_HOME` directory.

```
cat /etc/passwd | grep airflow
```

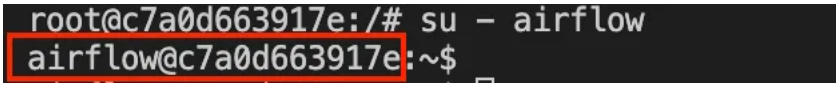
- Show the file /etc/passwd to check that the airflow user has been created*

5- Log into the airflow user

```
su - airflow
```

The `-` option used with the `su` command tells the system to simulate a full login for the "airflow" user, which means that the system will read the "airflow" user's profile and environment settings. This is important because it ensures that the "airflow" user has access to all the environment variables and settings needed to run Apache Airflow.

Overall, the `su - airflow` the command is used to switch to the "airflow" user account and simulate a full login for that user. This ensures that the "airflow" user has access to all the necessary environment variables and settings needed to run Apache Airflow. As you can see below now you're in the airflow profile.



`su - airflow` 's output

6- Create the virtual env named newenv

(This is a very important step to avoid the conflicts between Python versions)

```
python -m venv .newenv
```

Activate the virtual environment newenv

```
source .newenv/bin/activate
```

7- Download the requirement file to install the right version of Airflow's dependencies

(The most important step you must do and fewer developers that did this step)

```
wget https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/constrai
```

let me explain why the `wget https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/constraints-3.8.txt` command is important.

In Apache Airflow, dependencies are managed using Python's `pip` package manager. The `constraints.txt` file is used to specify the versions of the dependencies that should be installed for a specific version of Airflow. This ensures that the dependencies are compatible with the version of Airflow being used and prevents any incompatibility issues.

The `wget` command is used to download files from the internet. In this case, we're using `wget` to download the `constraints-3.8.txt` file from the official Apache Airflow repository on GitHub.

The `constraints-3.8.txt` file contains a list of dependencies and their versions that are compatible with Python 3.8, which is the version of Python that we're using in this case.

By downloading the `constraints-3.8.txt` file, we ensure that we have the correct versions of the dependencies needed to run Apache Airflow with Python 3.8. This is important because if we install the wrong version of a dependency, it can lead to compatibility issues and cause Apache Airflow to fail.

8- Install version 2.0.2 of apache-airflow with all sub packages

```
pip install "apache-airflow[crypto,celery,postgres,cncf.kubernetes,docker]"==2.0
```

The `pip install` command is used to install Python packages and their dependencies. In this case, we're using `pip` to install the Apache Airflow package and its dependencies.

The `"apache-airflow[crypto,celery,postgres,cncf.kubernetes,docker]"` argument is used to specify which optional dependencies of Apache Airflow we want to install. These dependencies are required for certain Airflow features to work properly. In this case, we're specifying that we want to install the following optional dependencies: `crypto` (for encrypting

connections), `celery` (for using Celery as a task queue), `postgres` (for using Postgres as a metadata database), `cncf.kubernetes` (for running Airflow on Kubernetes), and `docker` (for using Docker-related operators).

The `==2.0.2` argument is used to specify the version of Apache Airflow that we want to install. In this case, we're installing version 2.0.2 of Apache Airflow.

The `--constraint ./constraints-3.8.txt` the argument is used to specify the path to the constraints file that we downloaded earlier. This ensures that we install the correct versions of the dependencies that are compatible with the version of Apache Airflow that we're installing.

9- Initialize the metadatabase

```
airflow db init
```

You may find a warning that asks you to create a user so let’s create it

```
airflow users create --username  admin --firstname admin --lastname admin --role
```

10- Start Airflow’s scheduler in the background

```
airflow scheduler &
```

11- Start Airflow’s webserver in the background

```
airflow webserver &
```

Now congratulations you can check localhost:8080 to see this beautiful UI

Apache Airflow UI

And login with username admin and password admin as we configured earlier.

Login into Apache Airflow UI

In conclusion, running Apache Airflow locally via Docker can be a convenient and efficient way to set up your Airflow environment. By following the steps outlined in this article, you will be able to quickly and easily set up your Airflow environment and start building and managing your data pipelines. By using Docker, you can avoid the hassle of manual installation and configuration, and ensure that your Airflow environment is consistent across different machines and environments. We hope that this article has been helpful to you and that you feel confident in running Apache Airflow locally via Docker.

I hope you enjoyed reading this and finding it informative, feel free to follow, add your comments, thoughts, or feedback, and don't forget to get in touch on LinkedIn or follow my medium account to keep updated.

to every story on Medium.

Data Science | Machine Learning | Artificial Intelligence | Technology | Software Engineering

Written by Ansam Yousry

Follow

860 Followers · Writer for Technology Hits

Help data engineers grow their skills by sharing real-world demos and in-depth technical articles. https://www.linkedin.com/in/ansam-yousry/

More from Ansam Yousry and Technology Hits

Ansam Yousry in Technology Hits

7 Graph Database Use Cases That Will Change Your Mind

Source: Canva

6 min read · Sep 26, 2023

15

Aditya Darekar in Technology Hits

I Updated My 2015 15" MacBook Pro to macOS Sonoma—One...

I never thought my beautiful 8-year-old electronic buddy would ever see the light of ...

12 min read · Nov 2, 2023

271 1

Nuno Campos in Technology Hits

Ansam Yousry

Obsidian Plugins Review—57

Elevate Your Obsidian Experience: Discover the Best New Plugins for Enhanced Note-...

· 7 min read · May 9, 2024

402 7

Apache Spark vs. Apache Flink: A Comprehensive Comparison

Apache Spark and Apache Flink are two popular open-source data processing...

· 3 min read · Sep 14, 2023

78 1

See all from Ansam Yousry

See all from Technology Hits

Recommended from Medium

Dirk Steynberg

Developing Locally with Dockerized Apache Airflow and...

In this guide, we'll explore a comprehensive setup for a Dockerized Apache Airflow dev...

6 min read · Jan 19, 2024

4

Ayyoub Maulana in Data Engineer Things

Developing Multi Nodes Hadoop, Spark Cluster, and Airflow in...

Apache Airflow is the de facto standard of workflow orchestration

8 min read · Feb 14, 2024

36

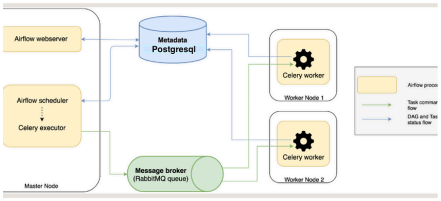
Lists

Predictive Modeling w/ Python
20 stories · 1253 saves

AI Regulation
6 stories · 473 saves

ChatGPT prompts
47 stories · 1640 saves

Natural Language Processing
1492 stories · 1008 saves



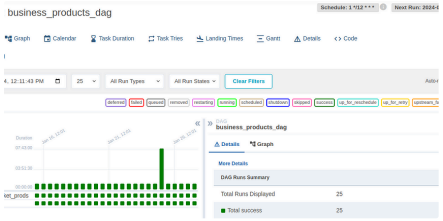
Saket Jain

(PART — B)Setting Up Apache Airflow in a Multi-Node Cluster

Introduction

3 min read · Jan 11, 2024

1



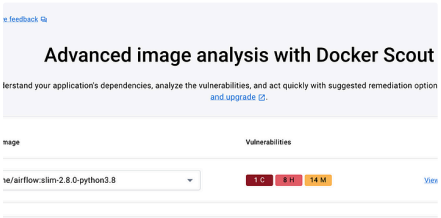
Breno Teixeira

How to Automate tasks with Airflow, Docker, and Python on yo...

Context

7 min read · Jan 28, 2024

199

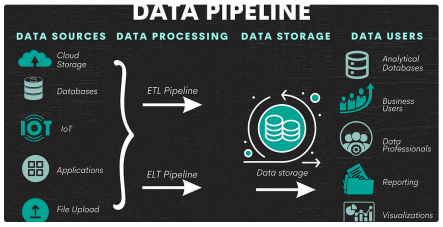


YenChinLee

Securing Your Airflow on Production: Building Docker...

6 min read · Dec 20, 2023

54



Rajat Belgundi

Efficient ETL: Cleaning, Transforming, and Loading CSV...

Data Engineering is the current talk of the data industry. Companies have started...

9 min read · Jan 8, 2024

113

See more recommendations