**Airflow with docker**

[**https://airflow.apache.org/docs/apache-airflow/stable/tutorial/pipeline.html**](https://airflow.apache.org/docs/apache-airflow/stable/tutorial/pipeline.html)

**Detailed tricks & usage**

[**https://www.youtube.com/watch?v=K9AnJ9\_ZAXE&list=PLwFJcsJ61oujFW8pTo9S8\_b6wujg5NgGW**](https://www.youtube.com/watch?v=K9AnJ9_ZAXE&list=PLwFJcsJ61oujFW8pTo9S8_b6wujg5NgGW)

**to start airflow with docker**

*# Download the docker-compose.yaml file*

curl -LfO 'https://airflow.apache.org/docs/apache-airflow/stable/docker-compose.yaml'

*# Make expected directories and set an expected environment variable*

mkdir -p ./dags ./logs ./plugins

echo -e "AIRFLOW\_UID=**$(**id -u**)**" > .env

*# Initialize the database*

docker-compose up airflow-init

*# Start up all services*

docker-compose up

**turn on just airflow without init db: init db will reset all settings in db**

docker-compose up

**turn down docker and release volume assigned to docker**

docker-compose down -v

**to change port  
Text

Description automatically generated**

**8080 is the docker container**

**5050 will be the airflow link** [**http://localhost:5050**](http://localhost:5050)

**Db**

**Text

Description automatically generated**

**To access postgres outside of airflow**

**And**

**To connect airflow to the db**

**Note that this postgres is created inside a docker image so it needs docker to be up for access .**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Change to port** other than 5432 **so dbeaver can connect eg 6432:5432**

**DAGS**

1. [task1, task2]>>task3

**# means task1 and task2 will run at the same time, once both complete, task 3 will run**

1. **Use bash command to execute python files/packages**
2. **Preferred way to use for python functions: taskflow api**
3. **Dag using python operator with parameter:**
4. **To backfill after setting catchup = False**

In terminal run:

docker ps

**this will give you instance of docker running**

Text

Description automatically generated

docker exec -it c4bc….. bash

**to backfill**

airflow dags backfill -s 2022-01-01 -e 2022-02-22

exit

**refresh UI to see refreshed in the past dates**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

1. **a good DAG example:**

**https://docs.astronomer.io/learn/dags**

**Text

Description automatically generated**

**To manage python dependencies (library) on airflow,**

**Create**

requirements.txt

Dockerfile

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Then restart the airflow using docker-compose up

Try docker image extending

**To run existing env**

python3 -m venv /path/to/new/virtual/environment

**to create env**

python3 -m venv [name of your env eg: env\_airflow]

source env\_ariflow/bin/activate

pip install apache-airflow

**To use Minio on S3: https://min.io/docs/minio/container/index.html**

mkdir -p ~/minio/data

docker run \

-p 9000:9000 \

-p 9090:9090 \

--name minio \

-v ~/minio/data:/data \

-e "MINIO\_ROOT\_USER=ROOTNAME" \

-e "MINIO\_ROOT\_PASSWORD=CHANGEME123" \

quay.io/minio/minio server /data --console-address ":9090"

* **mkdir** creates a new local directory at **~/minio/data** in your home directory.
* **docker run** starts the MinIO container.
* **-p** binds a local port to a container port.
* **-name** creates a name for the container.
* **-v** sets a file path as a persistent volume location for the container to use. When MinIO writes data to **/data**, that data mirrors to the local path **~/minio/data**, allowing it to persist between container restarts. You can replace **~/minio/data** with another local file location to which the user has read, write, and delete access.
* **-e** sets the environment variables [**MINIO\_ROOT\_USER**](https://min.io/docs/minio/linux/reference/minio-server/minio-server.html#envvar.MINIO_ROOT_USER) and [**MINIO\_ROOT\_PASSWORD**](https://min.io/docs/minio/linux/reference/minio-server/minio-server.html#envvar.MINIO_ROOT_PASSWORD), respectively. These set the [root user credentials](https://min.io/docs/minio/container/administration/identity-access-management/minio-user-management.html#minio-users-root). Change the example values to use for your container.

Use <http://127.0.0.1:9090/> to access minio

In terminal

docker ps

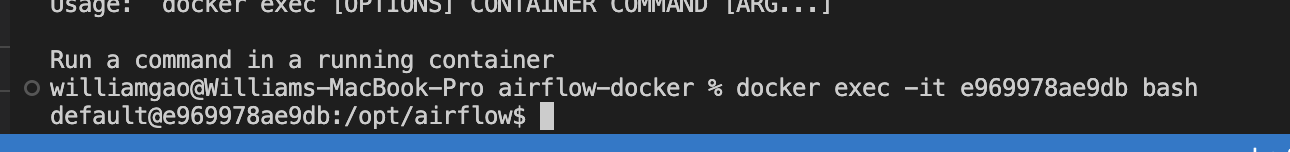
find the running container and copy the id Text

Description automatically generated

then run

docker exec -it e969978ae9db bash (replace with running id)

now you are in container



pip list | grep amazon

this will give you the amazon version Graphical user interface, text

Description automatically generated

now go to <https://airflow.apache.org/docs/>

click on amazon

select version, top left to version

click on python API

# look for [airflow.providers.amazon.aws.sensors.s3](https://airflow.apache.org/docs/apache-airflow-providers-amazon/7.1.0/_api/airflow/providers/amazon/aws/sensors/s3/index.html#module-airflow.providers.amazon.aws.sensors.s3)

# import this in the dag

# go to airflow UI and add connection

# if s3 is not installed

# [pip install apache-airflow-providers-amazon==7.1.0](https://pypi.org/project/apache-airflow-providers-amazon/7.1.0/)

Create a new connection with the following attributes:

**Conn Id:** my\_conn\_S3

**Conn Type:** S3

**Extra:**

{"aws\_access\_key\_id":"\_your\_aws\_access\_key\_id\_", "aws\_secret\_access\_key": "\_your\_aws\_secret\_access\_key\_"}

### **Long version, setting up UI connection:**

* On Airflow UI, go to Admin > Connections
* Create a new connection with the following attributes:
* Conn Id: my\_conn\_S3
* Conn Type: S3
* Extra: {"aws\_access\_key\_id":"\_your\_aws\_access\_key\_id from minio server\_", "aws\_secret\_access\_key": "\_your\_aws\_secret\_access\_key\_", ‘host’: ‘http://host.docker.internal:9000’}