Airflow with docker
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

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

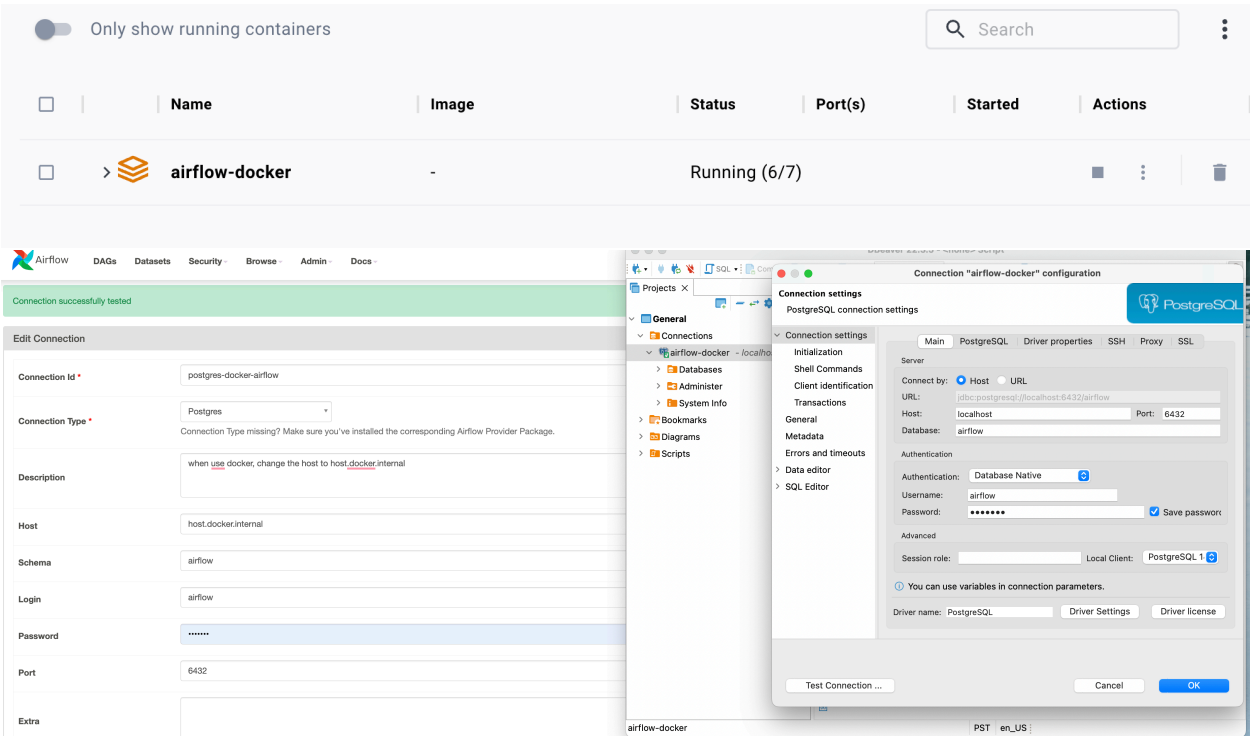
```yaml
  airflow-webserver:
    <<: *airflow-common
    command: webserver
    ports:
      - 5050:8080
    # {host : container}
    healthcheck:
      test: ["CMD", "curl", "--fail", "http://localhost:5050/health"]
      interval: 10s
      timeout: 10s
      retries: 5
    restart: always
    depends_on:
      <<: *airflow-common-depends-on
      airflow-init:
        condition: service_completed_successfully
```

8080 is the docker container
5050 will be the airflow link http://localhost:5050

Db

```yaml
services:
  postgres:
    image: postgres:13
    environment:
      POSTGRES_USER: airflow
      POSTGRES_PASSWORD: airflow
      POSTGRES_DB: airflow
    volumes:
      - postgres-db-volume:/var/lib/postgresql/data
    ports:
    - 6432:5432
    # {host : container}
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "airflow"]
      interval: 5s
      retries: 5
    restart: always
```

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 .



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

2. **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
    eks (healthy)    0.0.0.0:8080->8080/t
    ebserver_1
    c4bc255242a1    apache/airflow:2.0.1
    eks              8080/tcp
    cheduler_1
    cf9e7fc0f1a4    wordpress:latest
    eks              0.0.0.0:8000->80/tcp
    0c6b8c3b217e    mysql:5.7
    eks              3306/tcp, 33060/tcp
    4cf44f55b07b    postgres:13
    eks (healthy)    5432/tcp
    1
docker exec -it c4bc….. bash
      to backfill
airflow dags backfill -s 2022-01-01 -e 2022-02-22
exit
```

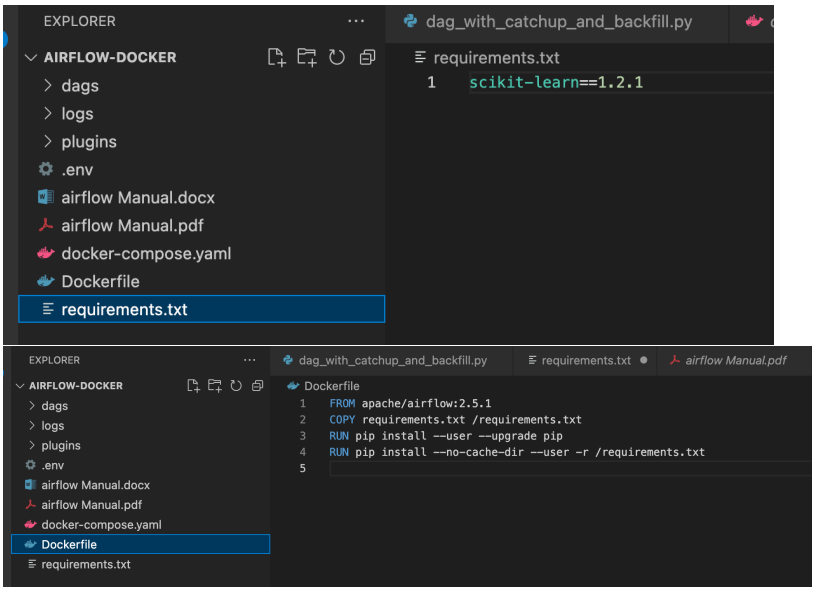
refresh UI to see refreshed in the past dates

5. **a good DAG example: https://docs.astronomer.io/learn/dags**

```python
from airflow import DAG
from airflow.operators.email import EmailOperator
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
from airflow.providers.snowflake.transfers.s3_to_snowflake import S3ToSnowflakeOperator

from pendulum import datetime, duration

# Instantiate DAG
with DAG(
    dag_id="s3_to_snowflake",
    start_date=datetime(2023, 1, 1),
    schedule="@daily",
    default_args={"retries": 1, "retry_delay": duration(minutes=5)},
    catchup=False,
):
    # Instantiate tasks within the DAG context
    load_file = S3ToSnowflakeOperator(
        task_id="load_file",
        s3_keys=["key_name.csv"],
        stage="snowflake_stage",
        table="my_table",
        schema="my_schema",
        file_format="csv",
        snowflake_conn_id="snowflake_default",
    )

    snowflake_query = SnowflakeOperator(
        task_id="run_query", sql="SELECT COUNT(*) FROM my_table"
    )

    send_email = EmailOperator(
        task_id="send_email",
        to="noreply@astronomer.io",
        subject="Snowflake DAG",
        html_content="<p>The Snowflake DAG completed successfully.<p>",
    )

    # Define dependencies
    load_file >> snowflake_query >> send_email
```

To manage python dependencies (library) on airflow,
Create

requirements.txt
Dockerfile



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