Build documentation

Prerequisite

- Docker installed and configured
    - `docker –version` ~ Docker version 24.0.6
    - `docker compose version` ~ Docker Compose version v2.23.0-desktop.1
- Amazon S3 Bucket (https://aws.amazon.com/en/s3/)
    - create and setup an S3 Bucket for file storage
- Keycloak
    - https/SSL is required for keycloak to work. README.md

Main setup

Clone the repo:

```
git clone https://github.com/amosproj/amos2023ws04-pipeline-manager.git
```

Navigate to the main root folder using:

```
cd amos2023ws04-pipeline-manager
```

As we have secrets in the backend app, we need to copy the template env to an .env

```
cp src/backend/.env.template src/backend/.env
cp src/backend/client_secrets.template.json src/backend/client_secrets.json
```

And then configure the environment variables to connect to your ASW and Apache Airflow connections.

To build the images:

```
docker compose build
```

And then in order to get the system up and running, execute the following:

```
docker compose up -d # in detached mode
```

ℹ️ For first time it could take a while to download images and configure it , but the consecutive builds will be faster.

Frontend application only - Angular

Prerequisite:

- node version ~ 21.1.0
- Npm version ~ 10.2.3

Install angular locally

- check that npm is installed by running

```
npm --version
```

- install Angular cli using version 16.2.10.

```
sudo npm i @angular/cli@16.2.10
```

- install the required packages

```
npm install
```

- for running the application

```
ng serve
```

The frontend should be visible on : http://localhost:4200/

Backend application only - Flask / MongoDB

Database setup

For manually setting up the mongoDB. Please follow the official documentation. (https://www.mongodb.com)

Or use an adjusted docker-compose.yml without the app.

```
version: '3.8'
services:
  db:
    image: mongo:latest
    hostname: amos_mongodb
    environment:
      - MONGO_INITDB_DATABASE=dpms_db
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=pass
    volumes:
       - ./tmpDatabase:/docker-databases
    ports:
      - 27017:27017
```

Installing dependencies with pipenv

Navigate to the backend directory. Run pipenv install to install dependencies. To add dependencies, simply type

```
pipenv install -r requirements.txt
```

This will update the Pipfile and Pipfile.lock automatically.

Environment variables

Copy the template into an .env file `cp .env.template .env`

You need the following environment variables in a .env file and adjust them to the right parameters.

1. AWS_ACCESS_KEY - you can generate a keypair in the aws console
2. AWS_SECRET_KEY - you can generate a keypair in the aws console
3. REGION - aws region that the bucket is in
4. BUCKET_NAME - name of the s3 bucket
5. AIRFLOW_SERVER_URL - url of the airflow server

6. AIRFLOW_USERNAME
7. AIRFLOW_PASSWORD
8. OIDC_SECRET_KEY - oidc secret if you enable kecloak
9. ENABLE_KEYCLOAK - True/False

Running the Backend App

Navigate to the src directory. In your terminal, type:

```
python -u app.py
```

Or else just use the docker-compose.yml in the backend directory

```
docker compose up -d # detached mode
```

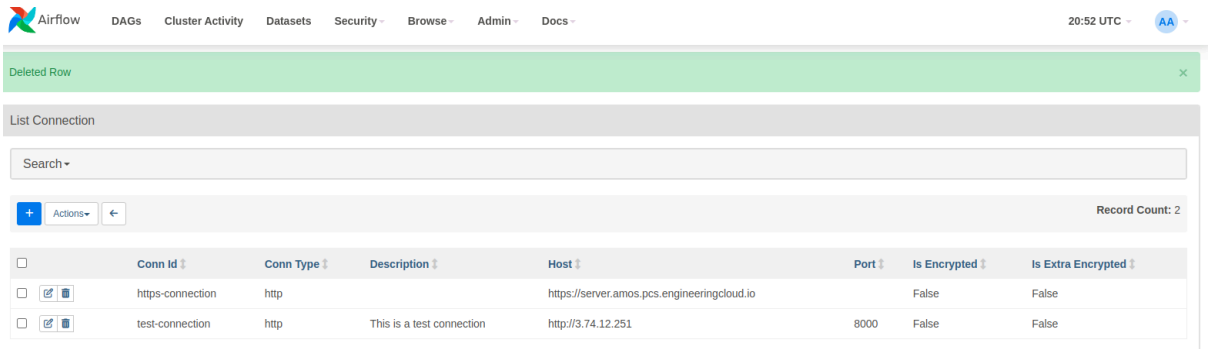
Running the Apache Airflow

Use the docker-compose.yml in the datapipeline directory

```
docker compose up -d # detached mode
```

Apache Airflow Configuration

Go to admin -> connections and add the 'https-connection' or the one that is used in the dags. The IP should be your server ip / server url.




```
)

send_response = SimpleHttpOperator(
    task_id="sendresponse",
    http_conn_id="https-connection",
    method="POST",
    endpoint="inputData",
    data=json.dumps("{{ task_instance.xc
    headers={"Content-Type": "applicatio
    response_check=lambda response: True
    dag=dag
)
```

Here you can see which http_conn_id is used for the datapipeline DAG. You need to configure it.