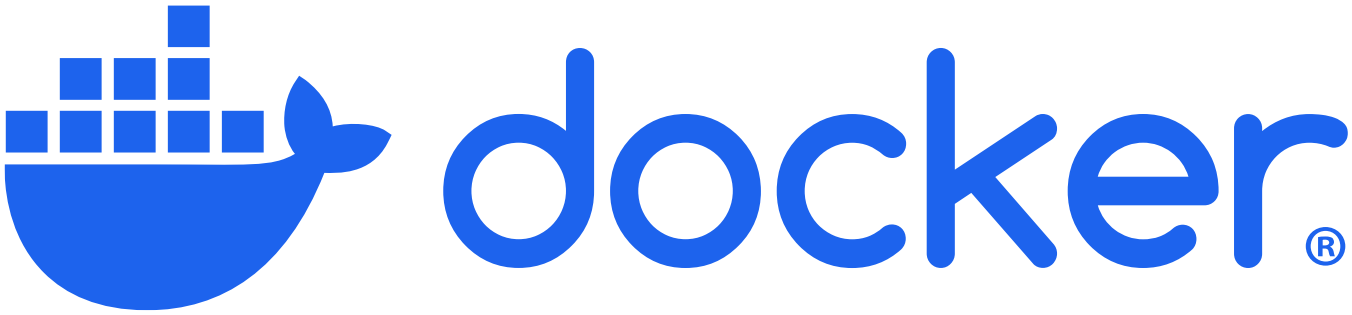


Docker CI-CD (Containerization)

Edrees Nabeel Ashab

Table of Contents

This is a report regarding the Docker CI-CD (Containerization) assignment. This report has two sections: **Services**, this section explains the services I created and their purpose. And I will explain how these services work together and interact with each other. **CI/CD Pipeline**, talks about how does my pipeline work.

1.0 Services:

mysqldb:

This service builds up a mysql server and starts it. It uses a volume to keep the data saved even if the containers were deleted.

```yaml
mysqldb:
  build: ./mysqldb
  image: mysqldb
  volumes:
    - ./mysqldb/mysql-data:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: Atypon#123
    MYSQL_DATABASE: temps
  ports:
    - "3307:3306"
  restart: always
```

mongodb:

The same as **mysqldb** services but this one use MongoDB as it name suggests.

```yaml
mongodb:
  build: ./mongodb
  image: mongodb
  ports:
    - "27017:27017"
  environment:
    MONGO_INITDB_ROOT_USERNAME: root
    MONGO_INITDB_ROOT_PASSWORD: Atypon#123
  restart: always
  volumes:
    - ./mongodb/mongo-data:/data/db
    - ./mongodb/init-scripts:/docker-entrypoint-initdb.d
```

authentication:

The **authentication** service makes sure whether the user entered valid credentials or not. It depends on the **mysqldb** services because the user's data is stored on the **mysqldb**.

```yaml
authentication:
  build: ./authentication
  image: authentication
  environment:
    MYSQL_DATABASE: temps
    MYSQL_USER: root
    MYSQL_PASSWORD: Atypon#123
    MYSQL_HOST: mysqldb
  ports:
    - "8002:8002"
  depends_on:
    - mysqldb
  restart: on-failure
```

enter-data:

The purpose of this services is to let the user enter some data (temperatures) and saves it to the **mysqldb** service, but before letting the user enter any data or even let him into the **enter-data** page, he has to be authenticated first using the **authentication** service.

```yaml
enter-data:
  build: ./enter-data
  image: enter-data
  environment:
    MYSQL_DATABASE: temps
    MYSQL_USER: root
    MYSQL_PASSWORD: Atypon#123
    MYSQL_HOST: mysqldb
  ports:
    - "8001:8001"
  depends_on:
    - mysqldb
    - authentication
  restart: on-failure
```

analytics:

The **analytics** services takes the temperature data from the **mysqldb** service, and then do some analysis that returns max, average, and min temperatures, after that it saves these analytics into the **mongodb** service.

```yaml
analytics:
  build: ./analytics
  image: analytics
  environment:
    MYSQL_DATABASE: temps
    MYSQL_USER: root
    MYSQL_PASSWORD: Atypon#123
    MYSQL_HOST: mysqldb
  ports:
    - "8004:8004"
  links:
    - mongodb
  depends_on:
    - mysqldb
    - mongodb
  restart: on-failure
```

show-results:

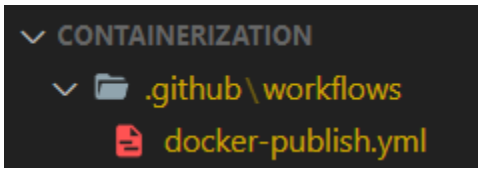
Lastly is the **show-results** service which show the results of the **analytics** service, this service also as the **enter-data** service, depends on the **authentication** service first before doing anything else.

```yaml
analytics:
  build: ./analytics
  image: analytics
  environment:
    MYSQL_DATABASE: temps
    MYSQL_USER: root
    MYSQL_PASSWORD: Atypon#123
    MYSQL_HOST: mysqldb
  ports:
    - "8004:8004"
  links:
    - mongodb
  depends_on:
    - mysqldb
    - mongodb
  restart: on-failure
```

2.0 CI/CD Pipeline:

This pipeline builds the docker images and pushes them into my docker repo.

At first, I added the **.github/workflows** directory, this directory automatically adds the workflow specified in it to the **GitHub Actions** once its pushed into **GitHub**.



Then I created a **.yml** called **docker-publish** that contains the workflow that will be executed.

The pipeline will run a job whenever a **push** happens on the **main** branch.

```yaml
name: Docker Image Publish

on:
  push:
    paths-ignore:
      - '**/README.md'
      - '**/.gitignore'
    branches: [ 'main' ]
```

The job runs on the **ubuntu-latest** image, and it has two steps: the **Checkout code** step that uses **actions/checkout@v2** which is an official **GitHub Action** used to check-out a repository so a workflow can access it.

```yaml
jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2
```

The second step is called **Build and push Docker images** that runs some commands, first is logging in to my **Docker Hub** account using **secrets** to provide a secure way to use sensitive information, then **docker-compose build** to build the services I have in the **docker-compose.yaml** file, and lastly it loops over the services and give each one a tag and pushes it to my **Docker Hub** repo.

```yaml
- name: Build and push Docker images
  run: |
    docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.DOCKERHUB_TOKEN }}
    docker-compose build
    services=(mysqldb mongodb authentication enter-data analytics show-results)
    for service in "${services[@]}"; do
      docker tag $service ${{ secrets.DOCKERHUB_USERNAME }}/$service:latest
      docker push ${{ secrets.DOCKERHUB_USERNAME }}/$service
    done
```