Web Service and Cloud Based - 2024

This README describes how to run demo for assignment 3.1 and 3.2

1. Project Structure

- In assignment 3.1, we've enhanced our Flask-based URL shortening and user authentication services by incorporating a MySQL database for persistent data storage and employing an Nginx reverse proxy for unified port access. Also involved containerizing the services, optimizing image sizes through multi-stage builds, and orchestrating the deployment using Docker-Compose.
- In assignment 3.2, we deployed project on three virtual machines provided by the University of Amsterdam, utilizing the latest Kubernetes version 1.29.2. The environment setup included installing Docker and Kubernetes on each machine, followed by node configuration with one master node and two worker nodes.

The project is structured as follows.

> - wscb
>   - **docker-compose.yml** : for 3.1
>   - **k8s**: for 3.2
>   - auth-deployment.yaml
>   - url-shorten-deployment.yaml
>   - mysql-deployment.yaml
>   - nginx-deployment.yaml
>   - secret.yaml
>   - **Authentication_Service** : for 3.1 and 3.2
>   - app.py
>   - models.py
>   - mysql_config.py
>   - utils.py
>   - Dockerfile
>   - wait-for-it.sh
>   - requirements.txt
>   - **Url_Shorten_Service**: for 3.1 and 3.2
>   - app.py
>   - models.py
>   - mysql_config.py
>   - utils.py
>   - Dockerfile

        - wait-for-it.sh
        - requirements.txt
    - **mysql**: for 3.1 and 3.2
        - Dockerfile
        - init_db.sql
    - **nginx**: for 3.1 and 3.2
        - Dockerfile
        - nginx.conf
    - test: for assignment 1 and 2
        - A bunch of test scripts of Canvas
    - docs : ignore this directory
        - A bunch of assignment descriptions of Canvas
        - Reports
    - deprecate : ignore this directory
        - Codes no longer used

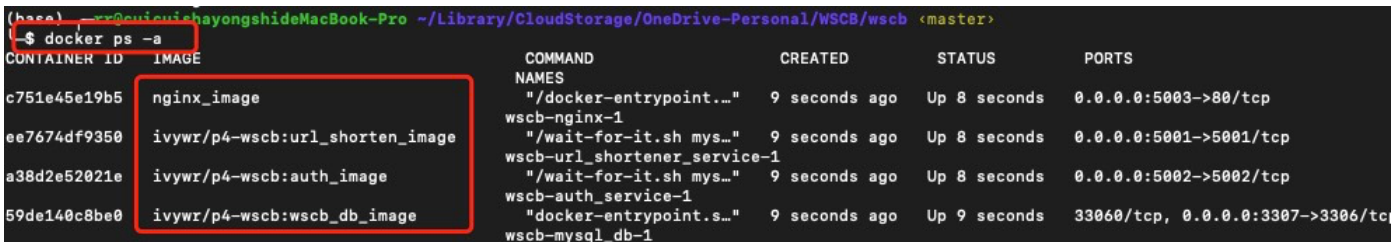
2. How to Run Demo with Docker Compose

Navigate to the project directory (that has docker-compose.yml there)and execute the following command:

> No need to build images in advance. Docker-compose will pull images from docker hub.

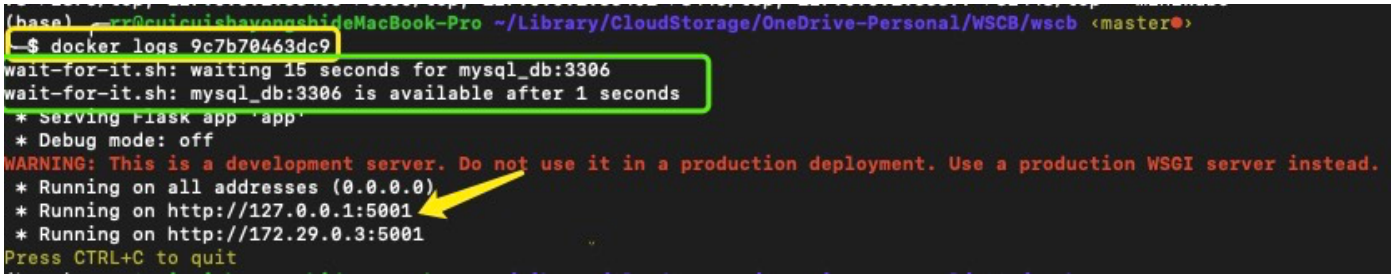
```
docker-compose up -d
```


```
docker ps
```



By checking the container logs, you can see that the url shorten service has been started on port 5001 (which we specified).

```
docker logs <url_shorten_container_id>
```

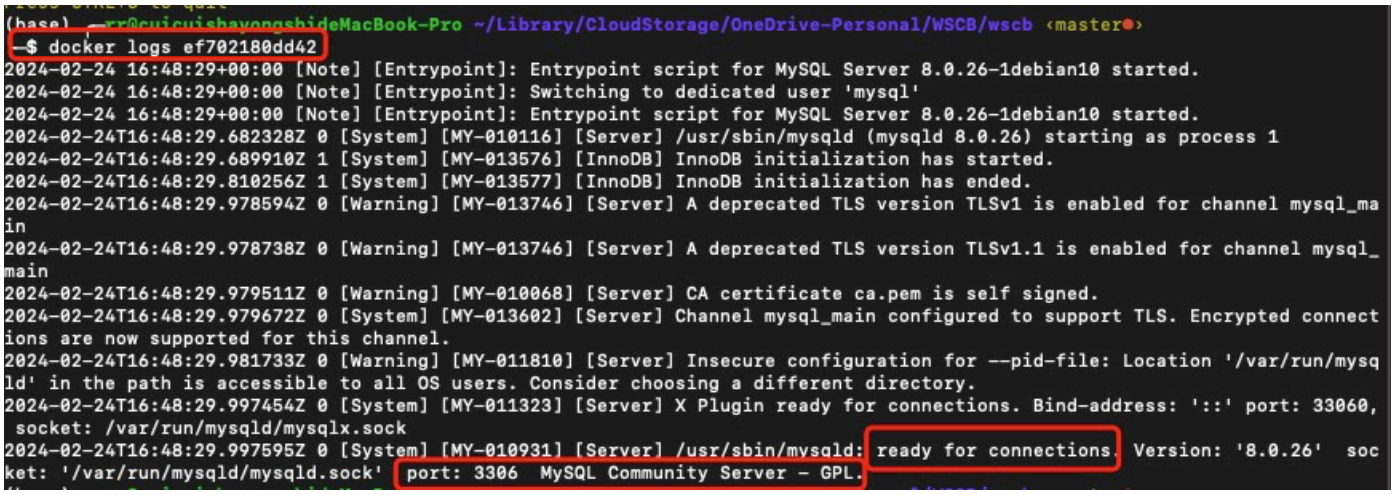


Inside the green box, it shows that we used the *wait-for-it.sh* script before running the python command to start the flask application. The reason for using this third-party wait script in the startup command of the Flask app is to **wait for the database port to become available and then start flask application**.

> You can get wait-for-it.sh by executing this on command line:
>
> wget https://raw.githubusercontent.com/vishnubob/wait-for-it/master/wait-for-it.sh

Similarly you can see authentication service has been started on port 5002 (which we speicified).

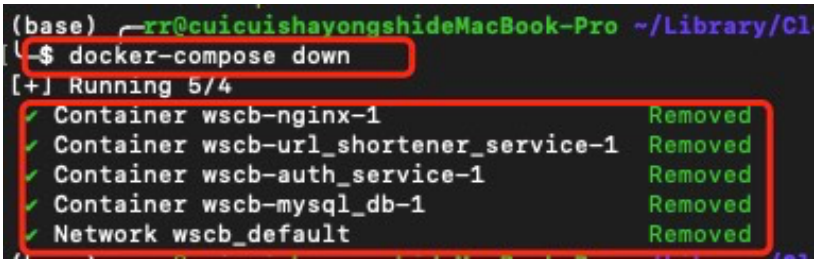
You can also check logs of MySQL database and see the mysql starts on port 3306. Note that this port can only be accessed within the containers (can only access by URL_shorten_service_container and Authentication_service_container).



We use **nginx proxy** to make two services available on one single entry. You can use Postman to test by sending requests to http://0.0.0.0:5003/ for url shorten service and http://0.0.0.0:5003/users/ for identity authentication service.

To remove the containers started by docker-compose up, you can use

```
docker-compose down
```



3. How to deploy project on k8s cluster

We deployed project on three virtual machines provided by the University of Amsterdam, accessed via remote SSH to deploy Kubernetes (k8s), with the latest version being 1.29.2.

(0) Set up k8s environment

The first step involved installing Docker and Kubernetes on all three machines. Then, we proceeded to configure the nodes. In setting up a Kubernetes cluster, nodes are generally classified into two types: master and worker nodes. On the master node, the command `kubeadm init` was used to initialize the cluster's control plane. Upon completion, there was an output with instructions on how to join worker nodes to the cluster. Then, each worker node was joined to the cluster using the `kubeadm join` command.

(1) Copy configuration files to master node

In k8s directory, you can see five files below: Copy all of them to master node.

- k8s:
  - auth-deployment.yaml
  - url-shorten-deployment.yaml
  - mysql-deployment.yaml
  - nginx-deployment.yaml
  - secret.yaml

(2) Apply configuration files

Apply all of them via kubectl apply -f command.

```
kubectl apply -f secret.yaml
kubectl apply -f mysql-deployment.yaml
kubectl apply -f auth-deployment.yaml
kubectl apply -f url-shorten-deployment.yaml
kubectl apply -f nginx-deployment.yaml
```

(3) Test

You can use Postman to test by sending requests to http://145.100.135.206:30000/ test url shorten service and http://145.100.135.206:30000/users/ for authentication service.