Web Service and Cloud Based - 2024 February

The project is structured as follows.

> - wscb
>   - docker-compose.yml
>   - README.md
>   - Authentication_Service
>   - app.py
>   - models.py
>   - mysql_config.py
>   - utils.py
>   - Dockerfile
>   - wait-for-it.sh
>   - requirements.txt
>   - Url_Shorten_Service
>   - app.py
>   - models.py
>   - mysql_config.py
>   - utils.py
>   - Dockerfile
>   - wait-for-it.sh
>   - requirements.txt
>   - mysql
>   - Dockerfile
>   - init_db.sql
>   - test
>   - A bunch of test scripts of Canvas
>   - docs
>   - A bunch of assignment descriptions of Canvas
>   - **Report for each assignment**
>   - deprecate
>   - Codes no longer used

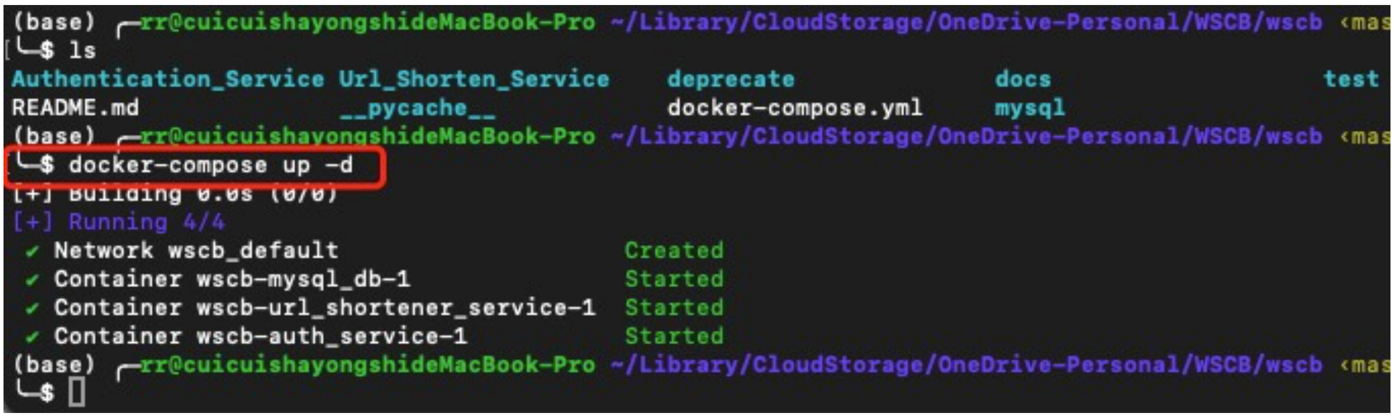
Container Virtualization

This part is about to start two Flask applications (Authentication_Service and Url_Shorten_Service) and MySQL database containers using Docker Compose and let the containers to communicate with database and the database data to persist.

How to Run Demo with Docker Compose

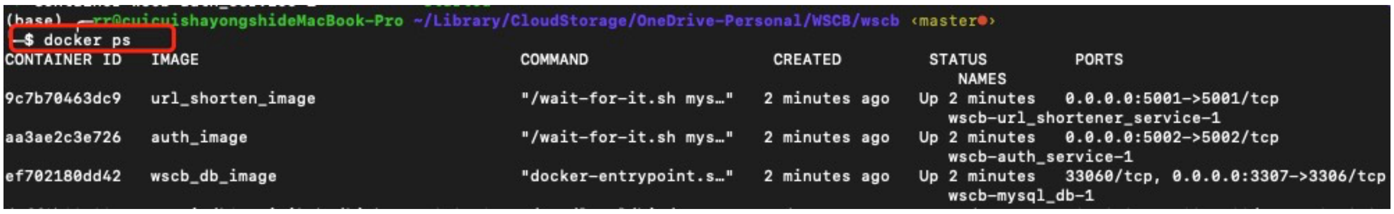
Navigate to the project directory (that has docker-compose.yml there)and execute the following command:

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



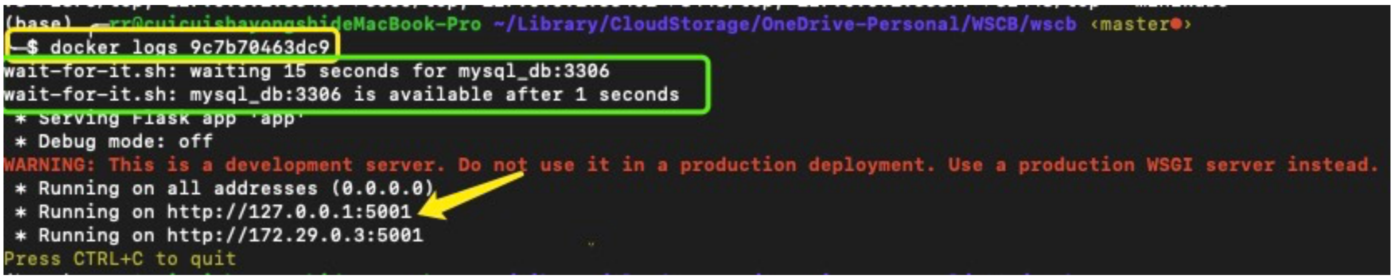
```
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/CloudStorage/OneDrive-Personal/WSCB/wscb ‹mas
└─$ ls
Authentication_Service Url_Shorten_Service    deprecate           docs               test
README.md              __pycache__            docker-compose.yml  mysql
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/CloudStorage/OneDrive-Personal/WSCB/wscb ‹mas
└─$ docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 4/4
 ✔ Network wscb_default                       Created
 ✔ Container wscb-mysql_db-1                  Started
 ✔ Container wscb-url_shortener_service-1     Started
 ✔ Container wscb-auth_service-1              Started
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/CloudStorage/OneDrive-Personal/WSCB/wscb ‹mas
└─$ ▯
```

```
docker ps
```



```
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/CloudStorage/OneDrive-Personal/WSCB/wscb ‹master●›
└─$ docker ps
CONTAINER ID   IMAGE                    COMMAND                   CREATED          STATUS         PORTS
                                                                                                     NAMES
9c7b70463dc9   url_shorten_image        "/wait-for-it.sh mys…"   2 minutes ago   Up 2 minutes   0.0.0.0:5001->5001/tcp
                                                                                                    wscb-url_shortener_service-1
aa3ae2c3e726   auth_image               "/wait-for-it.sh mys…"   2 minutes ago   Up 2 minutes   0.0.0.0:5002->5002/tcp
                                                                                                    wscb-auth_service-1
ef702180dd42   wscb_db_image            "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   33060/tcp, 0.0.0.0:3307->3306/tcp
                                                                                                    wscb-mysql_db-1
```

```
docker logs <url_shorten_container_id>
```



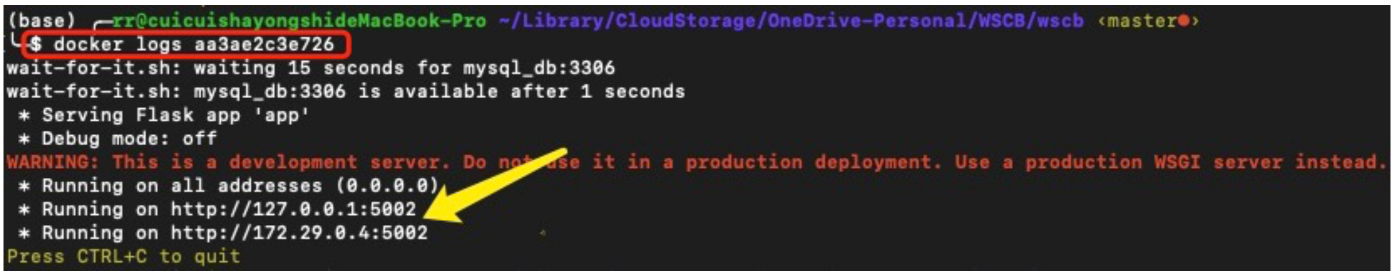
```
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/CloudStorage/OneDrive-Personal/WSCB/wscb ‹master●›
└─$ docker logs 9c7b70463dc9
wait-for-it.sh: waiting 15 seconds for mysql_db:3306
wait-for-it.sh: mysql_db:3306 is available after 1 seconds
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://172.29.0.3:5001
Press CTRL+C to quit
```

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

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**.

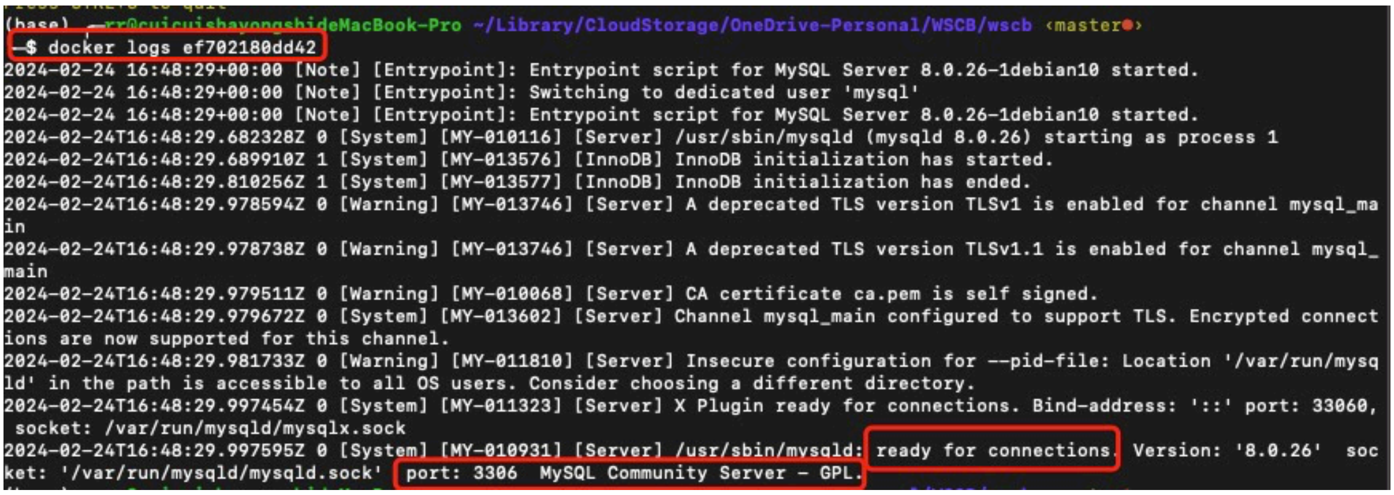
> You can get wait-for-it.sh by executing this on command line:
>
> wget https://raw.githubusercontent.com/vishnubob/wait-for-it/master/wait-for-it.sh

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



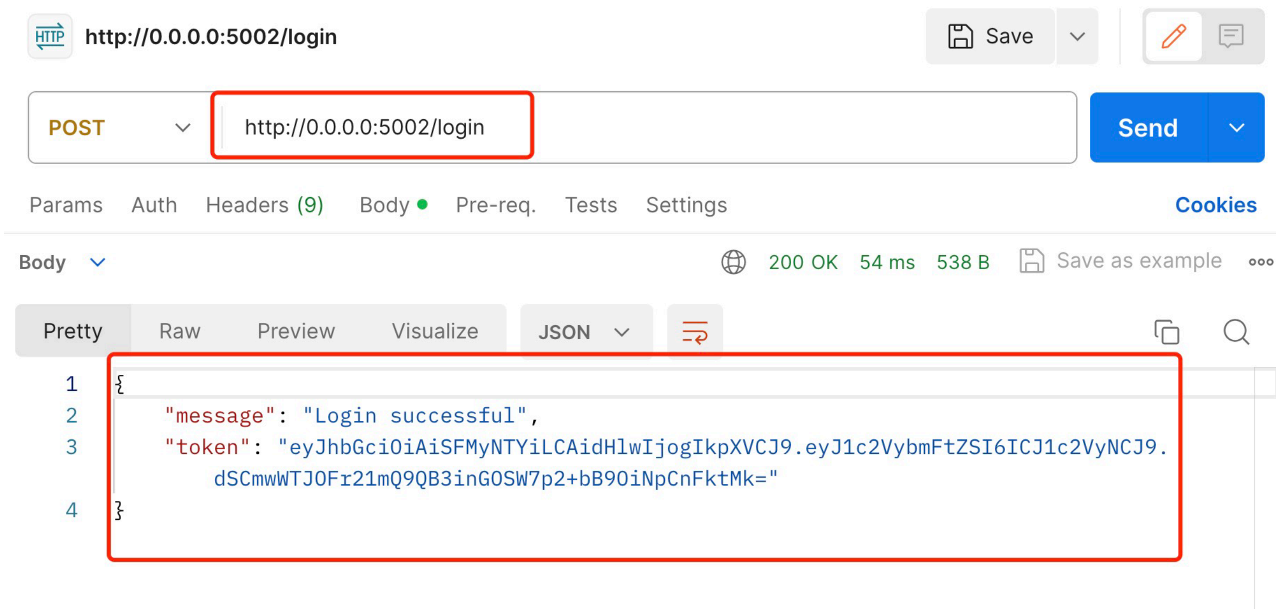
```
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/CloudStorage/OneDrive-Personal/WSCB/wscb ‹master●›
└─$ docker logs aa3ae2c3e726
wait-for-it.sh: waiting 15 seconds for mysql_db:3306
wait-for-it.sh: mysql_db:3306 is available after 1 seconds
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://172.29.0.4:5002
Press CTRL+C to quit
```

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).



```
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/CloudStorage/OneDrive-Personal/WSCB/wscb ‹master●›
└─$ docker logs ef702180dd42
2024-02-24 16:48:29+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.26-1debian10 started.
2024-02-24 16:48:29+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2024-02-24 16:48:29+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.26-1debian10 started.
2024-02-24T16:48:29.682328Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.26) starting as process 1
2024-02-24T16:48:29.689910Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2024-02-24T16:48:29.810256Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2024-02-24T16:48:29.978594Z 0 [Warning] [MY-013746] [Server] A deprecated TLS version TLSv1 is enabled for channel mysql_ma
in
2024-02-24T16:48:29.978738Z 0 [Warning] [MY-013746] [Server] A deprecated TLS version TLSv1.1 is enabled for channel mysql_
main
2024-02-24T16:48:29.979511Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2024-02-24T16:48:29.979672Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connect
ions are now supported for this channel.
2024-02-24T16:48:29.981733Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysq
ld' in the path is accessible to all OS users. Consider choosing a different directory.
2024-02-24T16:48:29.997454Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060,
 socket: /var/run/mysqld/mysqlx.sock
2024-02-24T16:48:29.997595Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.26'  soc
ket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server - GPL.
```

You can use Postman to test by send request to http://0.0.0.0:5001 (url shorten service) and http://0.0.0.0:5002 (identity authentication service)

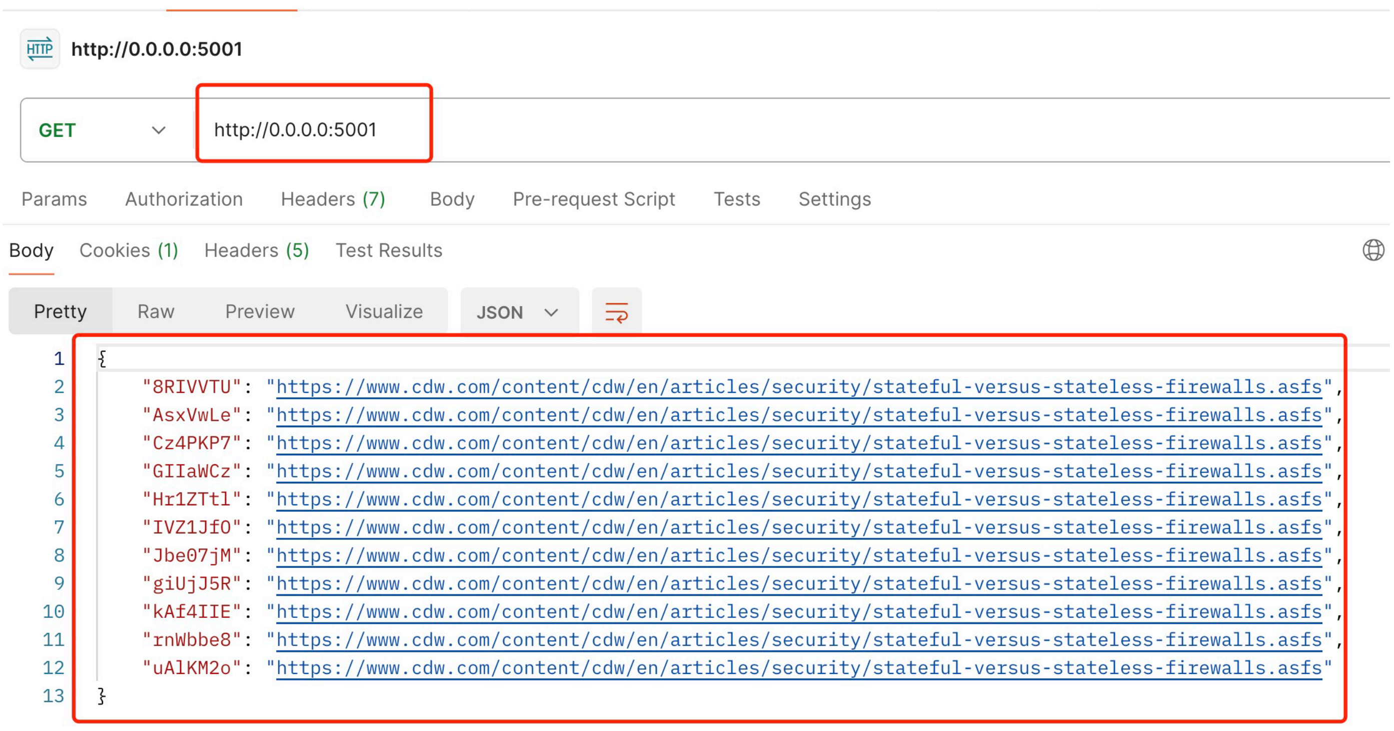


http://0.0.0.0:5002/login

POST http://0.0.0.0:5002/login

200 OK 54 ms 538 B

```
{
    "message": "Login successful",
    "token": "eyJhbGciOiAiSFMyNTYiLCAidHlwIjogIkpXVCJ9.eyJ1c2VybmFtZSI6ICJ1c2VyNCJ9.
        dSCmwWTJOFr21mQ9QB3inGOSW7p2+bB9OiNpCnFktMk="
}
```


http://0.0.0.0:5001

GET http://0.0.0.0:5001

Params Authorization Headers (7) Body Pre-request Script Tests Settings

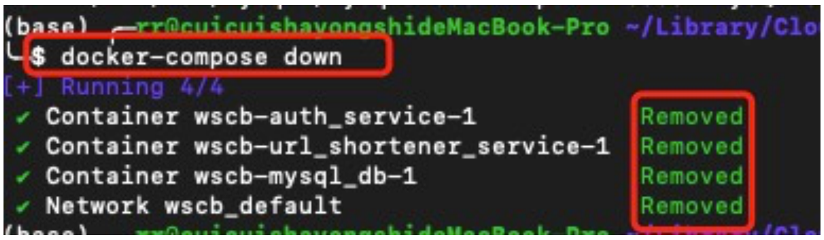
Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "8RIVVTU": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "AsxVwLe": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "Cz4PKP7": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "GIIaWCz": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "Hr1ZTtl": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "IVZ1Jf0": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "Jbe07jM": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "giUjJ5R": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "kAf4IIE": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "rnWbbe8": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs",
    "uAlKM2o": "https://www.cdw.com/content/cdw/en/articles/security/stateful-versus-stateless-firewalls.asfs"
}
```

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

```
docker-compose down
```



```
(base) ┌─rr@cuicuishayongshideMacBook-Pro ~/Library/Clo
└$ docker-compose down
[+] Running 4/4
 ✔ Container wscb-auth_service-1            Removed
 ✔ Container wscb-url_shortener_service-1   Removed
 ✔ Container wscb-mysql_db-1                Removed
 ✔ Network wscb_default                     Removed
```