CSE 486/586 Project_phase1

Jiahui Zhang
Department of Computer Science & Engineering
zhang377@buffalo.edu

1 Introduction

It can be very complex and time consuming when we develop and maintain a web application using many different technologies. Consider reducing complexity and development time through lightweight design to be more flexible and controllable. As a micro-framework for the python platform, Flask offers developers an extensible way through plugins that can be integrated into projects. To expand the developer's technology stack, MongoDB is a NoSQL database designed to handle large-scale and variable data feeds. Developers can use Docker technology to simplify the packaging and deployment of applications.

Docker Compose further simplifies the development environment by allowing you to customize basic components in separate files, including your application services, network volumes, and mount directories. Using Docker Compose provides ease of use when running multiple Docker containers to run commands. It allows you to define all your services in a single Compose file, and you can create and run all services from a configuration file with just one command. This ensures that version control runs through the container structure. Docker Compose uses project names to isolate environments from each other, which allows you to run multiple environments on a single host.

2 Design overview

The goal of the phase1 is to build a web application that can be used to implement a specific function using docker. The function is that we send a message through the interface we designed and then receive the message form the server. The massage we send will be stored in mongodb and returned via server.

In phase1, we use flask and mongodb and docker to build a web application to implement a function which is sending and receiving the message through the interface of a web. In the whole process, we use flask and mongo and docker to implement building and packing as well as running for the web application we want. We also need to define the docker-compose.yml file to ensure containers interacting with each other. Overall, the gerneral idea is about three components: flask-based app, mongodb and docker-compose.yml. Below is the gerneral work flow diagram.

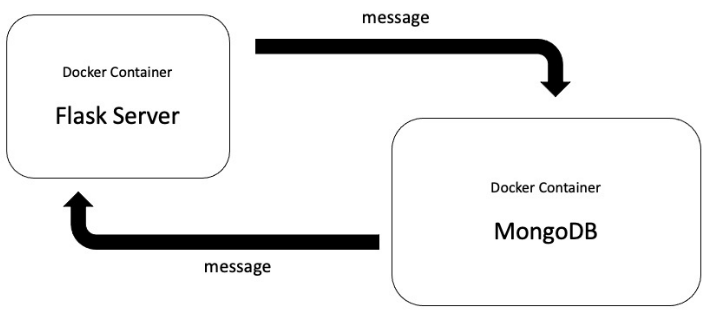


Figure 1: flow diagram

3 Implementation

Firstly, we need to download and install the docker(docker desktop) and be familiar with some terminology such that image and container. We did some preliminary work, set the mongo admin username and password and pull the mongo image.

Secondly, we need to create the codebase for our flask-based application. A web application under the portable flask framework, the front end uses html language. We wrote some files like main.py, dockerfile for flask image, and requirements.txt.

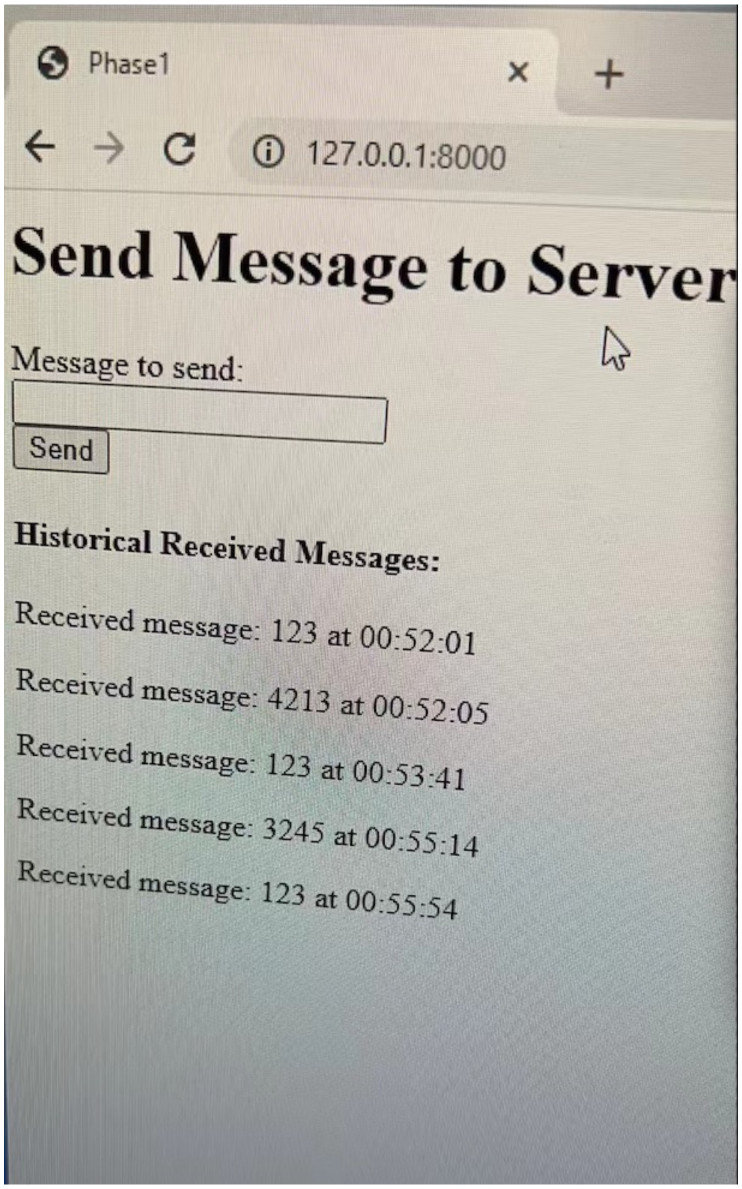
Thirdly, we need to create codebase for docker images and containers. Also, note here that we have exposed port 8000 for app and port 27017 for db as we are using the same in Flask app code.

Lastly, we need to create docker-compose.yml file to link containers(flask-server and mongodb) and make whole app run. After finishing all these, we are just nned one command to deploy this whole application. Run below command wait till all the things are deployed(docker-compose up).

4 Validation

The way we use to validate the phase1 is that we open the web page, and enter some massages in the edit box, then press the submit button. If we receive the message which shown below with the timestamp, then we can ensure that our web application works.

At first, we tested our flask-based application in pycharm without docker, and the result shown below seems worked. And then we tried to dockerize our application.



Phase1

127.0.0.1:8000

Send Message to Server

Message to send:

Send

Historical Received Messages:

Received message: 123 at 00:52:01

Received message: 4213 at 00:52:05

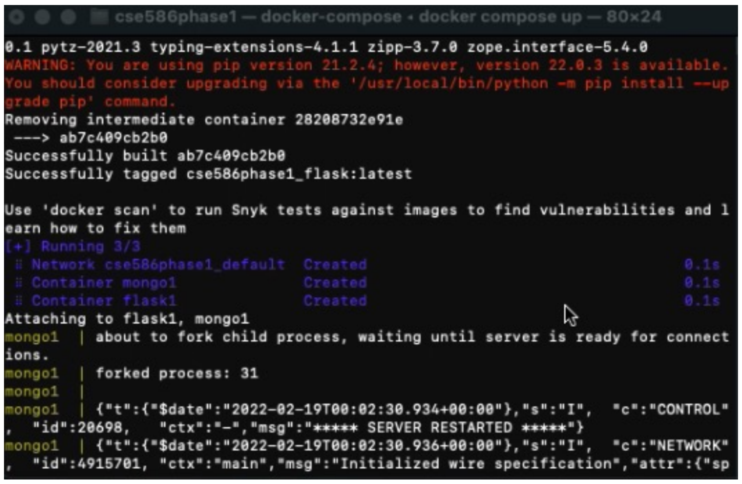
Received message: 123 at 00:53:41

Received message: 3245 at 00:55:14

Received message: 123 at 00:55:54

Figure 2: Running result without docker

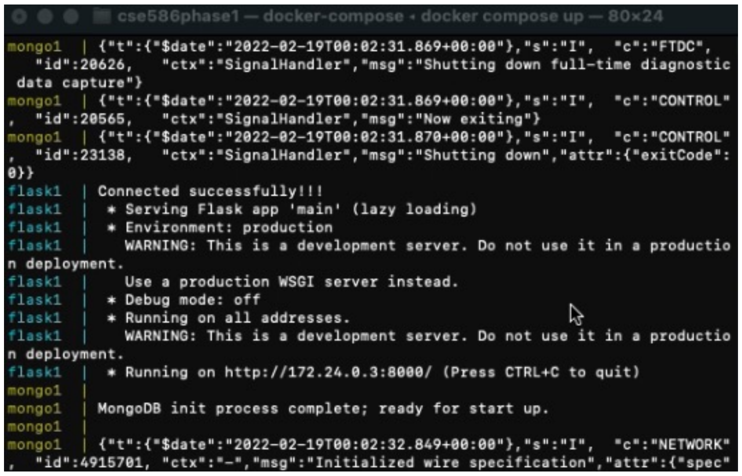
The result of running on docker failed. All the process seems well, but there always exists internal server error which leads to the failure of the dockerized application. Below is screenshots of the process.



```
cse586phase1 — docker-compose ◂ docker compose up — 80×24
0.1 pytz-2021.3 typing-extensions-4.1.1 zipp-3.7.0 zope.interface-5.4.0
WARNING: You are using pip version 21.2.4; however, version 22.0.3 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --up
grade pip' command.
Removing intermediate container 28208732e91e
 ---> ab7c409cb2b0
Successfully built ab7c409cb2b0
Successfully tagged cse586phase1_flask:latest

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and l
earn how to fix them
[+] Running 3/3
 ⠿ Network cse586phase1_default  Created                                    0.1s
 ⠿ Container mongo1              Created                                    0.1s
 ⠿ Container flask1              Created                                    0.1s
Attaching to flask1, mongo1
mongo1  | about to fork child process, waiting until server is ready for connect
ions.
mongo1  | forked process: 31
mongo1  |
mongo1  | {"t":{"$date":"2022-02-19T00:02:30.934+00:00"},"s":"I",  "c":"CONTROL"
,  "id":20698,   "ctx":"-","msg":"***** SERVER RESTARTED *****"}
mongo1  | {"t":{"$date":"2022-02-19T00:02:30.936+00:00"},"s":"I",  "c":"NETWORK"
,  "id":4915701, "ctx":"main","msg":"Initialized wire specification","attr":{"sp
```

Figure 3: terminal run log



```
cse586phase1 — docker-compose ◂ docker compose up — 80×24
mongo1  | {"t":{"$date":"2022-02-19T00:02:31.869+00:00"},"s":"I",  "c":"FTDC",
   "id":20626,   "ctx":"SignalHandler","msg":"Shutting down full-time diagnostic
 data capture"}
mongo1  | {"t":{"$date":"2022-02-19T00:02:31.869+00:00"},"s":"I",  "c":"CONTROL"
,  "id":20565,   "ctx":"SignalHandler","msg":"Now exiting"}
mongo1  | {"t":{"$date":"2022-02-19T00:02:31.870+00:00"},"s":"I",  "c":"CONTROL"
,  "id":23138,   "ctx":"SignalHandler","msg":"Shutting down","attr":{"exitCode":
0}}
flask1  | Connected successfully!!!
flask1  |  * Serving Flask app 'main' (lazy loading)
flask1  |  * Environment: production
flask1  |    WARNING: This is a development server. Do not use it in a productio
n deployment.
flask1  |    Use a production WSGI server instead.
flask1  |  * Debug mode: off
flask1  |  * Running on all addresses.
flask1  |    WARNING: This is a development server. Do not use it in a productio
n deployment.
flask1  |  * Running on http://172.24.0.3:8000/ (Press CTRL+C to quit)
mongo1  |
mongo1  | MongoDB init process complete; ready for start up.
mongo1  |
mongo1  | {"t":{"$date":"2022-02-19T00:02:32.849+00:00"},"s":"I",  "c":"NETWORK"
,  "id":4915701, "ctx":"-","msg":"Initialized wire specification","attr":{"spec"
```

Figure 4: terminal run log

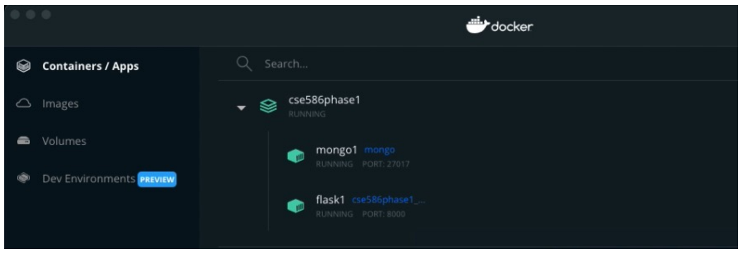
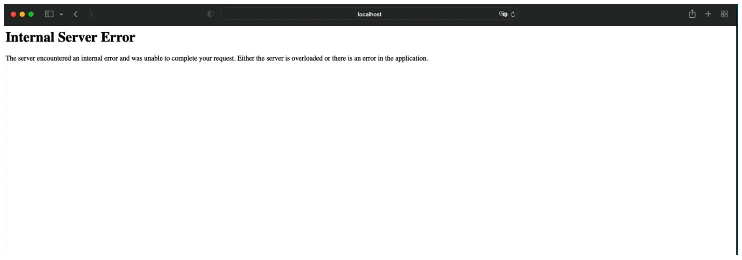


Figure 5: docker desktop

When we login http://localhost:8000, the page always shows that there exists internal server error. We still cannot figure out the solution, so just make it be recorded and turn in it.



Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

Figure 6: page error

After finishing the phase1, we are supposed to be more familiar with the docker and Command line experience.

5 Reference

https://www.runoob.com/docker/docker-install-mongodb.html
https://docs.docker.com/get-started/