CSE 586 Distributed Systems

Project – Understanding Consensus in Distributed Systems

Submitted by Madhavi Sajja (50417103), Sai Srinivas Chetti (50418655)

PHASE-1 – Creating Docker Application – [DonationApp]

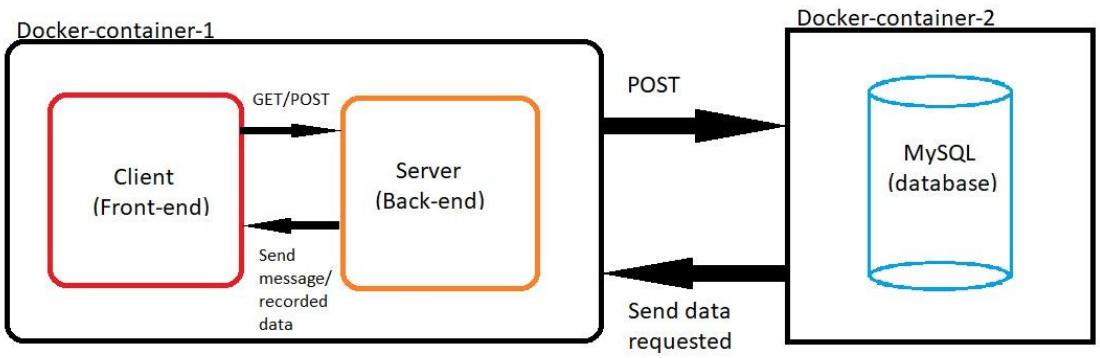
1. Introduction

The idea is to familiarize with docker technology and work with containers, by creating a simple docker application with a database, webpage and communicating with each other through well-defined channels. Created a web application called DonorsApp to record the financial donations from customers and store them in a MySQL database. The code is implemented using Python-Flask, HTML, MySQL and all the files – app.py, templates, env, Dockerfile, docker-compose.dev.yml, demo video, screenshot of own system are included in the submission.

2. Design Overview

The design includes a docker web application, web page, MySQL database, network. Web application is built into a docker image and run using docker container. Similarly, the MySQL database is built into a docker image and run as a docker container. A network is created between the two containers for communication.

Fig. Phase-1 DonorApp Docker Application - System Design



3. Implementation

Installed docker and relevant software – python flask, pyaml in the local system and developed a simple docker application.

3.1 Creating web application

Using Python-Flask web framework, developed a web page that accepts customer name, email and donation amount as inputs and stores it into to MySQL Database through POST request and also displaces the data in the database to the webpage as a table in route.

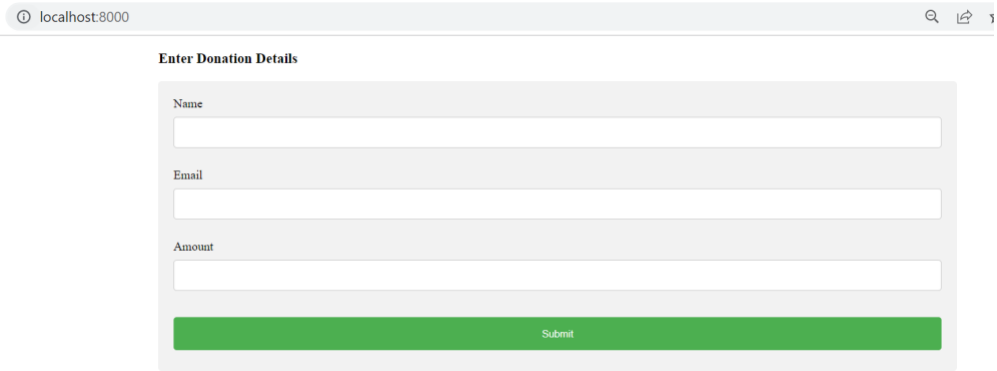
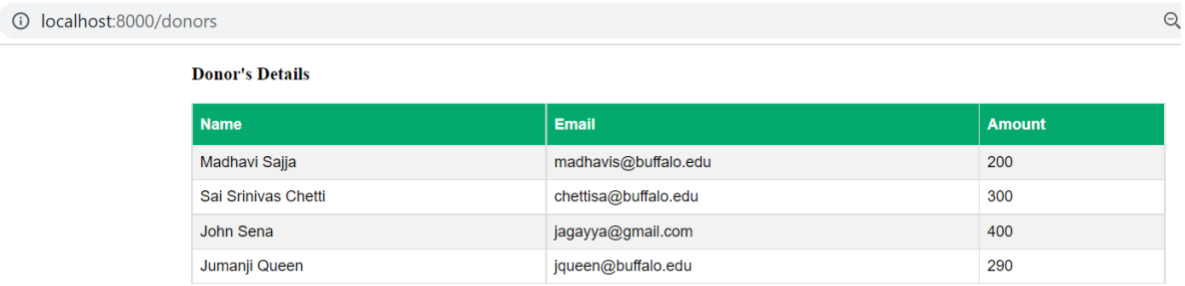


Fig. Web page for DonorApp



Name	Email	Amount
Madhavi Sajja	madhavis@buffalo.edu	200
Sai Srinivas Chetti	chettisa@buffalo.edu	300
John Sena	jagayya@gmail.com	400
Jumanji Queen	jqueen@buffalo.edu	290

Fig. Response page with donation records collected from MySQL database.

3.2 Dockerizing web application

Create a Dockerfile in the same directory as app.py. Create a 'requirements.txt' file which contains the necessary software installations required to run. Use "python:3.8-slim-buster" as the base image, and "`python -m flask run`" as the command to run the application.

```
Dockerfile > ...
# syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

3.2 Creating docker-compose file

We are going to use Docker-Compose file to start our Python-Flask application and MySQL database and to establish a connection between them to communicate.

```yaml
docker-compose.dev.yml
version: '3.8'

services:
  web:
    build:
      context: .
    ports:
    - 8000:5000
    volumes:
    - ./:/app

  mysqldb:
    image: mysql
    platform: linux/x86_64
    container_name: mysqldb
    ports:
    - 3306:3306
    environment:
    - MYSQL_ROOT_PASSWORD=root
    volumes:
    - mysql:/var/lib/mysql
    - mysql_config:/etc/mysql

volumes:
  mysql:
  mysql_config:
```

With this docker-compose.dev.yml file, the full application can be built and run using a single command as below.

**** Note ** after running the below command, pls wait for a min for the docker to start and run, before opening the localhost**

```
>> docker-compose -f docker-compose.dev.yml up –build -d
```

```
PS C:\Users\Madhavi Sajja\Desktop\Docker_Maddie\Srinu\DonorsApp> docker-compose -f docker-compose.dev.yml up --build -d
[+] Running 13/13
 - mysqldb Pulled
   - 6552179c3509 Already exists
   - d69aa66e4482 Pull complete
   - 3b19465b002b Pull complete
   - 7b0d0cfe99a1 Pull complete
   - 9ccd5a5c8987 Pull complete
   - 2dab00d7d232 Pull complete
   - 5d726bac08ea Pull complete
   - 11bb049c7b94 Pull complete
   - 7fcdd679c458 Pull complete
   - 11585aaf4aad Pull complete
 => [1/5] FROM docker.io/library/python:3.8-slim-buster@sha256:0ac2a12df86b01d6a482fd07b62b6ca2afe3355cd520260c7d561c4e5c966cdb
 => [internal] load build context
 => => transferring context: 6.75kB
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY requirements.txt requirements.txt
 => CACHED [4/5] RUN pip3 install -r requirements.txt
 => [5/5] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:49e75cc5610248431c8e4c1ccf09752123a83e2601763083b24c3ed76f7013ef
 => => naming to docker.io/library/donorsapp_web
[+] Running 5/5
 - Network donorsapp_default        Created
 - Volume "donorsapp_mysql"         Created
 - Volume "donorsapp_mysql_config"  Created
 - Container mysqldb                Started
 - Container donorsapp-web-1        Started
```

4. Validation

The below screenshots show,

1. How the docker images are created for mysql and donorsapp.

2. How the docker containers are created for mysql, donorsapp.
3. How the webpage works when docker is running.
4. How the user inputs are stored in the mysql database named “DB_Phase1”
5. How the database is retrieved to the webpage (client) from the database node.

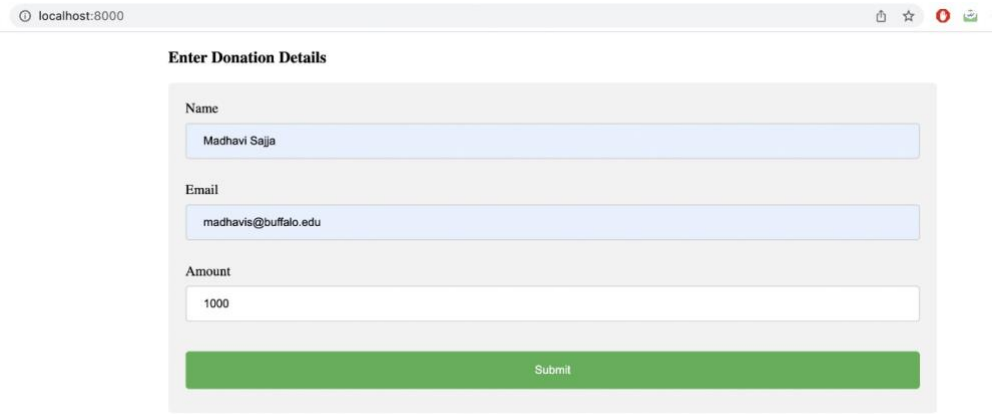
4.1 Images - The docker images are created for donorsapp_web, mysql.

```
PS C:\Users\Madhavi Sajja\Desktop\Docker_Maddie\Srinu\DonorsApp> docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
donorsapp_web   latest    49e75cc56102   3 minutes ago    282MB
mysql           latest    17b062d639f4   20 hours ago     519MB
```

4.2 Containers - The below docker containers are created, validating that the application is running in docker containers.

```
PS C:\Users\Madhavi Sajja\Desktop\Docker_Maddie\Srinu\DonorsApp> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS                                  NAMES
3a5e8821a92f   donorsapp_web   "python3 -m flask ru…"   3 minutes ago   Up 3 minutes   0.0.0.0:8000->5000/tcp                 donorsapp-web-1
6153f29da136   mysql           "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp      mysqldb
```

4.3 Webpage - The application runs in the web page http://localhost:8000 which takes input from the user.



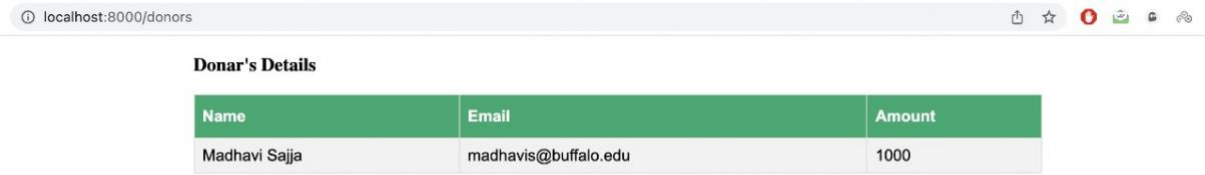
4.4 Webpage to Database connection - The inputs from the user are recorded into the database named **DB_Phase1**, and inputs are inserted into the table **donors**. An alert will be displaced in the webpage for the same.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| DB_Phase1          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.02 sec)

mysql> use DB_Phase1
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from donors;
+---------------+----------------------+--------+
| name          | email                | amount |
+---------------+----------------------+--------+
| Madhavi Sajja | madhavis@buffalo.edu |   1000 |
+---------------+----------------------+--------+
1 row in set (0.01 sec)
```

4.5 Database to Web Page connection- http://localhost:8000/donors page shows the data is retrieved from the database and sent to the client (web page).



5. References - weblinks

https://docs.docker.com/language/python/

https://www.youtube.com/watch?v=QjtW-wnXlUY&t=314s

https://www.youtube.com/watch?v=6L3HNyXEais&t=1054s