

CSE 586 Distributed Systems

Project – Understanding Consensus in Distributed Systems

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

PHASE-2 – Adding Persistence and Data Stack – [DonorApp]

1. Introduction

The aim is to create multiple(three) instances of docker web application containers and establish communications with each other. Each docker app container will have its own database built into their own docker containers. One of the three docker apps will act as a leader. The leader app 'Node A' will communicate with other apps - 'Node B' and then 'Node C' through HTTP requests, and finally back to Node A.

2. Design Overview

The design includes a connected network of three docker web applications and three MySQL databases, all in their respective docker containers. Each docker web application container is part of 2 networks – one with other application containers, one with its local MySQL database. Node-1 can access and modify only DB1 connected to it through dbnet1. Similarly, Node-2 can access and modify DB2, Node-3 can access and modify only DB3. Networking helps to create isolation.

The client accesses the web application of leader host "http:localhost:8000", fills the form with the details and clicks on submit (POST request). The leader then executes HTTP POST request to Node-2, which runs the functionality - collecting user-entered data, storing in local database, and retrieving data logs redirected to "/donors" and service goes back to Leader server. Now, leader sends HTTP POST request to Node-2, which performs the same functionality and saves the data into its local database. Now, Leader performs the same functionality. This ensures that the same data is stored in all three databases at any point of time.

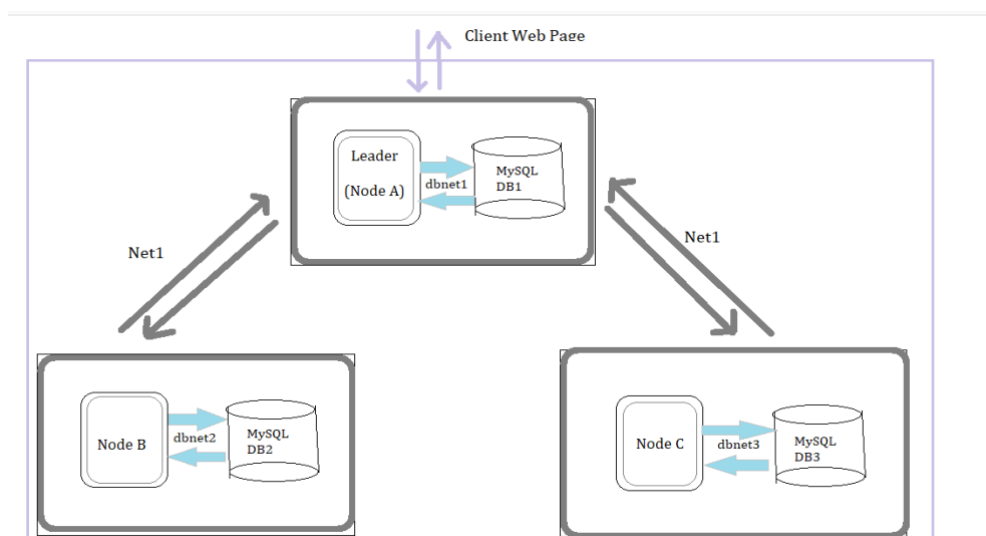


Fig. System Design Overview of Docker Networking

3. Implementation

Updated docker-compose.dev.yml file to create the required instances of web apps and 3 SQL databases. Volumes are created for each database to ensure data persistence. In the python flask application, the code is written in such a way that server-1 will invoke server-2, server-3 applications, and their functionalities.

3.1 Docker-compose YML file

```
services:
  server_1:
    depends_on:
      - mysqladb_1
    build:
      context: ./
    container_name: server_1
    hostname: server_1
    networks:
      - net1
      - dbnet1
    ports:
      - 8000:5000
    environment:
      - IS_LEADER=true

  mysqladb_1:
    image: mysql
    platform: linux/x86_64
    container_name: mysqladb_1
    networks:
      - dbnet1
    ports:
      - 3306:3306
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_TCP_PORT=3306
    volumes:
      - mysqladb_1:/var/lib/mysql

  server_2:
```

All the required network elements, volumes, port configurations, services, databases are configured in the docker-compose.dev.yml file.

Server_1: port 8000:5000

Server_2: port 8001:5000

Server_3: port 8002:5000

3.1.1 Volumes

Below volumes are created to save all the records, which will not be deleted when app restarts.

3.1.2 Networks

Network “net1” is among application containers. Local networks dbnet1, dbnet2, dbnet3 are individual networks used to connect app containers with their db containers.

```
volumes:
  mysqladb_1:
  mysqladb_2:
  mysqladb_3:

networks:
  net1:
  dbnet1:
  dbnet2:
  dbnet3:
```

3.2 Application development using Python Flask

When the docker application is started, all the 3 app containers (server_1, server_2, server_3) and all 3 database containers (mysqladb_1, mysqladb_2, mysqladb_3) are booted and starts running. The client accesses “https:localhost:8000” web form of server-1 (leader node) and fills in data.

In the post request method of the application, if the node under processing is a leader node, http post request is sent to server-2 node (node-2) using the server’s name and port information, as the

servers are on the same network (net1). Here server address is http://server_2:5000. This runs the hello_world() function again in POST method, and the database DB2 connected to server_2 is populated, and the service is returned to server_1. Now server_1 sends http request to server_3 and DB3 is populated, and service comes back to server_1. Server_1 populates its database DB1. Hence, all the databases will have the same user data replicated.

```
@app.route('/', methods=('GET', 'POST'))
def hello_world():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        amount = request.form['amount']

        if os.environ["IS_LEADER"]=="true":
            final_outputs = []
            x = requests.post("http://server_2:5000/", {'name':name, 'email': email, 'amount': amount})
            x = requests.post("http://server_3:5000/", {'name':name, 'email': email, 'amount': amount})
            Populate_database(db['host1'], '3306', name, email, amount)

        elif request.host_url == "http://server_2:5000/":
            Populate_database(db['host2'], '3307', name, email, amount)

        elif request.host_url == "http://server_3:5000/":
            Populate_database(db['host3'], '3308', name, email, amount)

        return redirect('/leaders')

    return render_template('index.html')
```

4. Validation

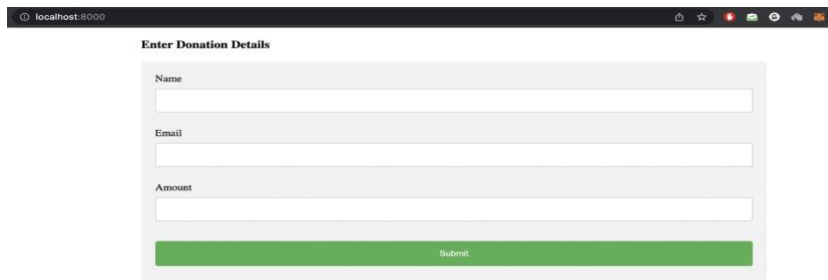
The user data submitted in <http://localhost:8000> is populated to all the databases DB1, DB2, DB3. This is verified both in mysql database, and also by the “/donors” route of all 3 web pages, <http://localhost:8000/donors>, <http://localhost:8001/donors>, <http://localhost:8002/donors>

Step-1: >> docker-compose -f docker-compose.dev.yml up --build -d

Docker application starts running. Docker containers are created and starts running.

```
PS C:\Users\Madhavi_Sajja\Desktop\SAJJA_Docs\SEM2-Spring2022\CSE586-DistributedSystems\Project-consensusAlgo\phase-2trial\madhavis_phase2\DonorApp_Phase2> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
ddc2725a44b1   donorapp_phase2_server_2   "python3 -m flask ru..." About a minute ago Up About a minute   0.0.0.0:8001->5000/tcp             server_2
b17bf3fb4177   donorapp_phase2_server_3   "python3 -m flask ru..." About a minute ago Up About a minute   0.0.0.0:8002->5000/tcp             server_3
77af09a187a7   donorapp_phase2_server_1   "python3 -m flask ru..." About a minute ago Up About a minute   0.0.0.0:8000->5000/tcp             server_1
b5638fe99c66   mysql         "docker-entrypoint.s..." About a minute ago Up About a minute   0.0.0.0:3306->3306/tcp, 33060/tcp   mysql_db_1
e7885c3bdb6a   mysql         "docker-entrypoint.s..." About a minute ago Up About a minute   3306/tcp, 33060/tcp, 0.0.0.0:3308->3308/tcp   mysql_db_3
118da466ea52   mysql         "docker-entrypoint.s..." About a minute ago Up About a minute   3306/tcp, 33060/tcp, 0.0.0.0:3307->3307/tcp   mysql_db_2
PS C:\Users\Madhavi_Sajja\Desktop\SAJJA_Docs\SEM2-Spring2022\CSE586-DistributedSystems\Project-consensusAlgo\phase-2trial\madhavis_phase2\DonorApp_Phase2>
```

Step-2: Entering data in <http://localhost:8000>



Enter Donation Details

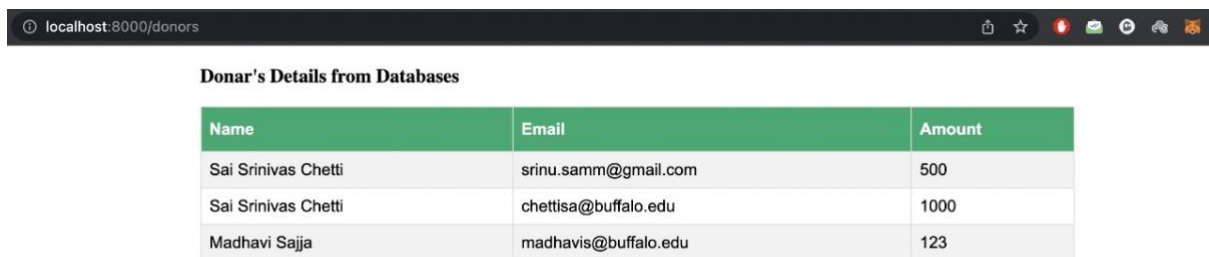
Name

Email

Amount

Submit

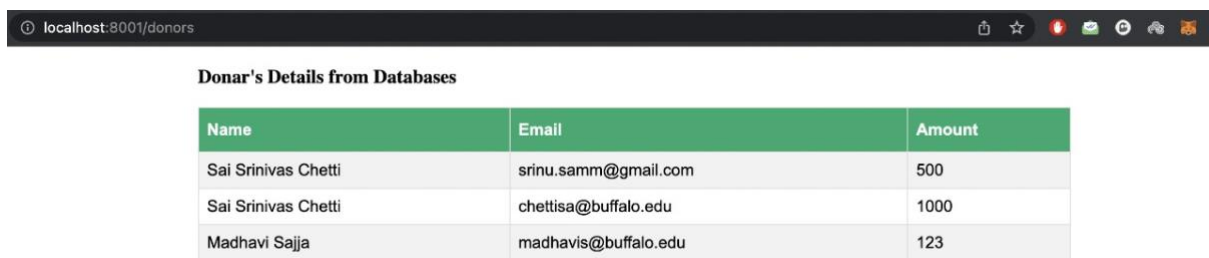
Step-3: On clicking submit button, the webpage is redirected to /donors route, which shows information stored in the connected database. While this step is performed the donors details are also stored in DB2, DB3.



Donar's Details from Databases

Name	Email	Amount
Sai Srinivas Chetti	srinu.samm@gmail.com	500
Sai Srinivas Chetti	chettisa@buffalo.edu	1000
Madhavi Sajja	madhavis@buffalo.edu	123

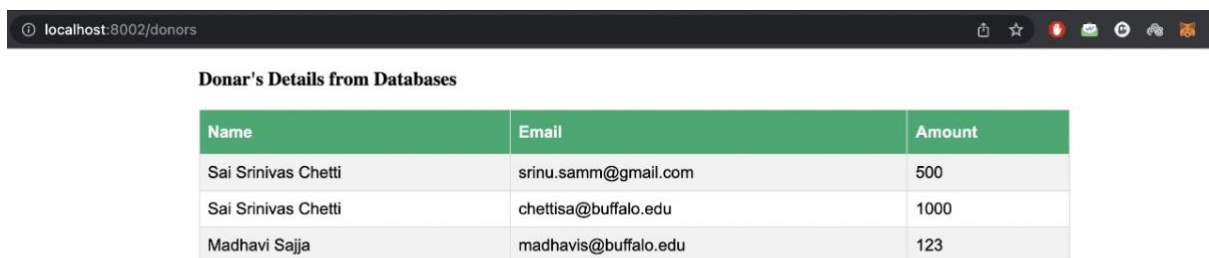
Step-4: Hence <http://localhost:8001/donors> shows:



Donar's Details from Databases

Name	Email	Amount
Sai Srinivas Chetti	srinu.samm@gmail.com	500
Sai Srinivas Chetti	chettisa@buffalo.edu	1000
Madhavi Sajja	madhavis@buffalo.edu	123

<http://localhost:8002/donors> shows:



Donar's Details from Databases

Name	Email	Amount
Sai Srinivas Chetti	srinu.samm@gmail.com	500
Sai Srinivas Chetti	chettisa@buffalo.edu	1000
Madhavi Sajja	madhavis@buffalo.edu	123

Step-5: Checking the same at MySQL databases, to prove the concept.

```

(base) srinivas@Srinivass-MacBook-Air ~ % mysql -h 127.0.0.1 -P 3307 -u root -p DB_Phase2
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from donors;
+-----+-----+-----+
| name          | email                | amount |
+-----+-----+-----+
| Sai Srinivas Chetti | srinu.samm@gmail.com | 500    |
| Sai Srinivas Chetti | chettisa@buffalo.edu | 1000   |
| Madhavi Sajja      | madhavis@buffalo.edu | 123    |
+-----+-----+-----+
3 rows in set (0.03 sec)

```

5. References - weblinks

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