Load Test Setup

Task requirement

- To show the result of response time in grafana with the help of load test.

Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously.

Environment details

- OS: Ubuntu 20.04
- Podman 3.4.1

List of tools and technologies

- Grafana: Version 2
- Postgres: Version 4
- Locust: Customised

Definition of tools

- `Grafana` :- is a multi-platform open source analytics and interactive visualisation web application. It provides charts, graphs, and alerts for the web when connected to supported data sources. It can be easily installed using Docker or Docker Compose.

- `PostgreSQL` :- also known as Postgres, is a free and open-source relational database management system emphasising extensibility and SQL compliance.

- `Locust` :- is an open-source load-testing tool. Load testing is a type of software testing that is conducted to check the tolerance/behaviour of the system under a specific expected load.

Command for the setup or configuration

1. Load test without grafana and timescale-db

- pip3 install locust
- pip3 install locust_plugins
- Create directory mkdir locust
- Go to the directory

- Create file touch locustfile.py

Note

1. Use the system IP as host to run the project.

2. System ip can be get using

IP a command

Now install the maven

Run the service :- **finobank-ptaplus-settlement.zip** , and the run the commands

3. mvn clean compile quarkus:dev -Dquarkus.http.host=192.168.29.116

****

```
sudo apt install maven
```

```
 mvn clean compile quarkus:dev -DskipTests
```

```
 mvn clean compile quarkus:dev -DskipTests
-Dquarkus.http.host=192.168.29.155
```

****

```
touch locustfile.py
```

```
from locust import HttpUser, constant, task, between,
TaskSet,SequentialTaskSet,events
import locust_plugins
```

```
@events.test_start.add_listener
def on_test_start(environment, **kwargs):
      print("A new test is starting")

@events.test_stop.add_listener
def on_test_stop(environment, **kwargs):
      print("A new test is ending")

class inactiveUsers(SequentialTaskSet):
      @task
      def helloworld(self):
      with self.client.get("/hello",catch_response=True,
name="helloworld",json={

},headers={"accept":"application/json","Content-Type":"application/json"
})as response:
            if response.status_code!=200:
                  response.failure("Failed to get item: StatusCode
"+str(response.status_code))
            else:
                  json_response_data=response.json()

class MyinactiveUsers(HttpUser):
      wait_time=between(1,2)
      tasks= [inactiveUsers]
      host = "http://192.168.1.101:8080"
```

Locust -f locustfile.py

****

Service access link :
https://drive.google.com/file/d/1lDkmU7Gx9KoEICj1ZWgPMAKnqoJ3PO_G/view?usp=drive_link

Using container:-
podman run -itd --name loadtesting -p 8089:8089 -v <Directory Path>:/mnt/locust localhost/loadtest:v1 -f /mnt/locust/locustfile.py

Note: localhost/loadtest:v1 is an image with the in-built locust_plugins.
Please find the images below.

Image name :-

Loadtestimage.tar.gz

https://drive.google.com/file/d/1rTd02C1ypJtOlnHLIQPaiWp5N9TrUnX_/view?ts=64b0edc6

2. Load test with Grafana and timescale db.

2.1. Setting up grafana and postgres db.

##Create a pod with name timescale and expose ports 5432 and 3000

podman pod create --name timescale --publish 5432:5432 --publish 3000:3000

```
##Create a postgres container, give it the desired environment
variables, attach it to the created pod
podman run -dt \
    --pod timescale \
    --name timescale-postgres \
    -e POSTGRES_PASSWORD=password \
    -e TIMESCALEDB_TELEMETRY=off \
    -v <directory Path>:/var/lib/postgresql/data \
    cyberw/locust-timescale:4

##Create the grafana container, give it the desired environment
variables, importantly PGHOST, attach it to the created pod
podman run -dt \
      --pod timescale \
      --name timescale-grafana \
      -e GF_AUTH_DISABLE_LOGIN_FORM=true \
      -e GF_AUTH_ANONYMOUS_ENABLED=true \
      -e GF_AUTH_ANONYMOUS_ORG_ROLE=Admin \
      -e GF_SECURITY_ALLOW_EMBEDDING=true \
      -e GF_LOG_LEVEL=warn \
      -e PGHOST=localhost \
      -v <Directory Path>:/var/lib/grafana \
        cyberw/locust-grafana:2
```

****

This is my bash script

```
#!/bin/bash
```

```
# Create the first directory
mkdir -p /home/user/Downloads/postgres/data


# Create the second directory
mkdir -p /home/user/Downloads/grafana/data


# Run a command using the first directory
echo "Running command1 using /path/to/first/directory"
command1 --option /home/user/Downloads/postgres/data


# Run a command using the second directory
echo "Running command2 using /path/to/second/directory"
command2 --option /home/user/Downloads/postgres/data



##Create a postgres container, give it the desired environment
variables, attach it to the created pod
podman run -dt \
      --pod timescale \
      --name timescale-postgres \
      -e POSTGRES_PASSWORD=password \
      -e TIMESCALEDB_TELEMETRY=off \
      -v /home/user/Downloads/postgres/data:/var/lib/postgresql/data \
      cyberw/locust-timescale:4

##Create the grafana container, give it the desired environment
variables, importantly PGHOST, attach it to the created pod
podman run -dt \
      --pod timescale \
      --name timescale-grafana \
      -e GF_AUTH_DISABLE_LOGIN_FORM=true \
      -e GF_AUTH_ANONYMOUS_ENABLED=true \
      -e GF_AUTH_ANONYMOUS_ORG_ROLE=Admin \
      -e GF_SECURITY_ALLOW_EMBEDDING=true \
      -e GF_LOG_LEVEL=warn \
      -e PGHOST=localhost \
      -v /home/user/Downloads/grafana/data:/var/lib/grafana \
      cyberw/locust-grafana:2
```

****

2.2 Create another temporary script to manage users data and run the same script.

```
GRAFANA_CRED="admin:admin"
GRAFANA_HOST="http://192.168.1.101:3000"
GRAFANA_OVERWRITE=false
DS_NAME="locust_timescale"
PGHOST="postgres"
PGPORT="5432"

curl -u "$GRAFANA_CRED" $GRAFANA_HOST/api/datasources -XPOST -H "Accept:
application/json" -H "Content-Type: application/json" -d '{"access":
"proxy","basicAuth": false,"basicAuthPassword": "","basicAuthUser":
"","database": "postgres","isDefault": false,"jsonData":
{"postgresVersion": 1200,"sslmode": "disable", "timescaledb":
true},"name": "'$DS_NAME'","orgId": 1,"password": "","readOnly":
false,"secureJsonData": {"password": "password"},"type":
"postgres","url": "'$PGHOST':'$PGPORT'","user": "postgres","version":
3,"withCredentials": false}'

ds=(10878 14423 14422 15419);
for d in "${ds[@]}"; do
  echo -n "Processing $d: "
  j=$(curl -s -k -u "$GRAFANA_CRED" $GRAFANA_HOST/api/gnet/dashboards/$d
| jq .json)
  echo
"{\"dashboard\":"${j}",\"overwrite\":$GRAFANA_OVERWRITE,\"inputs\":[{\"n
ame\":\"DS_LOCUST\",\"type\":\"datasource\",
\"pluginId\":\"postgres\",\"value\":\"$DS_NAME\"}]}" > payload.json
  curl -v -k -u "$GRAFANA_CRED" -H "Accept: application/json"\
      -H "Content-Type: application/json"\
      -d @payload.json \
      $GRAFANA_HOST/api/dashboards/import; echo ""
done
```

****
****

```
Install jq file
```

```
sudo apt-get -y install jq
```

2.3.Validate setup

validate postgres

```
podman exec -it timescale-postgres bash
```

```
bash-5.1# psql -U postgres
```

psql (13.7)
Type "help" for help.

```
postgres=# \dt
```

```
           List of relations
 Schema |    Name    | Type  |  Owner
--------+------------+-------+----------
 public | events     | table | postgres
 public | request    | table | postgres
 public | testrun    | table | postgres
 public | user_count | table | postgres
(4 rows)
```

```
postgres=# select * from user_count;
```

```
postgres=# select *from user_count;
```

```
testplan          | user_count |          time               |           run_id

--------------------------+------------+-------------------------------+------------
------------------
 /mnt/locust/locustfile.py |        1 | 2023-09-21 10:58:08.705627+00 | 2023-09-21
10:58:08.646645+00

(0 rows)
```

##validate grafana

Open browser http://localhost:3000 will show the home page of grafana, check the datasource and test the connection.

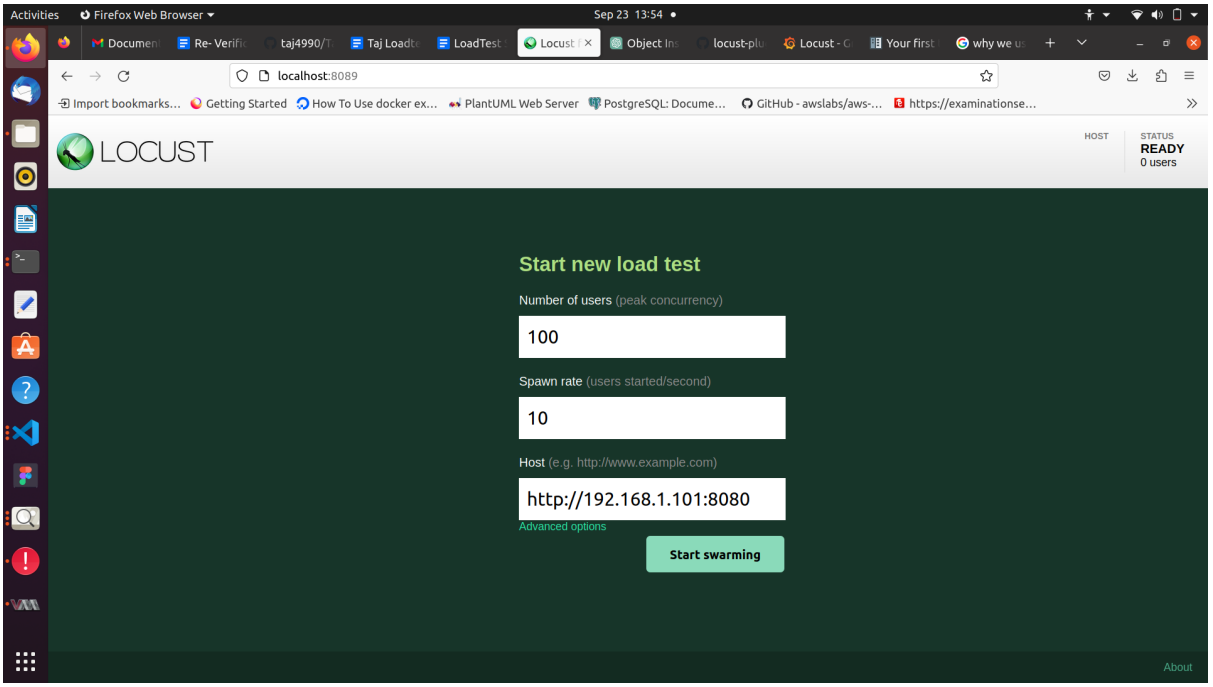
Step-6: Setup locust

Run the locust container with grafana and postgres configuration

```
podman run -itd --name loadtesting -p 8089:8089 -v <directory path to
mount>:/mnt/locust localhost/loadtest:v1 -f /mnt/locust/locustfile.py
--timescale --grafana-url=http://<machine ip>:3000 --pghost=<machine ip>
--pgport=5432 --pgpassword=password --pguser=postgres
```

6.1- Run the load test with grafana and timescale-DB

Locust ui exposed to the http://localhost:8089
Start a new test with the **number of users** and **spawn rate.**



You can see the user_count table as follows-

```
postgres=# select * from user_count;
```

You can add panels in grafana to get the graphical view of the responses.