Implementation Documentation

Term Project

Submitted in fulfillment of the requirement for

CLOUD COMPUTING

CS/SS G527

By

Kamaldeep Verma (2018H1120281P)

Mayur Namjoshi (2018H1120290P)

Vikas Yadav (2018H1120291P)

Siddhant Jain (2016A7PS0113P)

Under the supervision of

Dr. Hari Babu
Assistant Professor
Department of Computer Science and Information Systems

Birla Institute of Technology and Science
Pilani-333031, Rajasthan (India)
November, 2019

Interface for User

In our system, we have 3 end points for user

1. /configuration
2. /data
3. /query

1. **http://PaaSIP:5000/configuration**

METHOD: POST

BODY:

```
{
    'User-id': 'user id',
    'service -pipeline-conf': '1-2-3'
}
```

RESPONSE:

```
{
    'Token': 'generated token for further processing'
}
```

2. **http://PaaSIP:5000/data**

METHOD: POST

BODY:

```
        {
                ‘Token’: “above generated token”
                ‘record’: ‘user record in csv’
        }
    RESPONSE:
        {
                ‘status’: ‘success’
        }
```

3. **http://PaaSIP:5000/query**

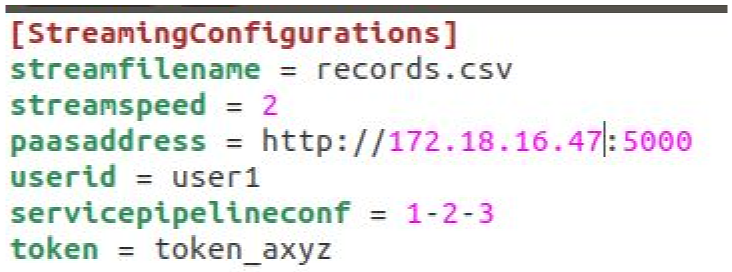
```
    METHOD:     GET
    RESPONSE:   JSON object of processed records
        {
                ‘records’: ‘[record 1,
                             record 2,
                             ……
                                 ]
        }
```

The User has three files

1. “config.ini”
2. “config.py”
3. “streaming.py”

1. **Config.ini**

   This file contains the configuration parameters for the user.



```
[StreamingConfigurations]
streamfilename = records.csv
streamspeed = 2
paasaddress = http://172.18.16.47:5000
userid = user1
servicepipelineconf = 1-2-3
token = token_axyz
```

   StreamfileName - User can provide the name of Real Time Streaming Record File

   StreamSpeed - User can tune the speed of stream. The data rate can be increased dynamically to check the system response.

   ServicePipelineConf - User can provide pipeline configuration through this parameter

   PaaSaddress - IP Address of the Gateway Server

   UserId - User Id of the User

   Token - Used for User Request Identification

2. **Config.py**

   After setting up configuration, user runs "config.py" file through which he/she gets new new token which is automatically modified in "config.ini" file.

3. **Streaming.py**

   When user runs streaming.py, data starts streaming to the web server. Streaming speed can be tuned at runtime by modifying streaming speed value in "config.ini" file. Lower the value, the higher the Stream Speed.

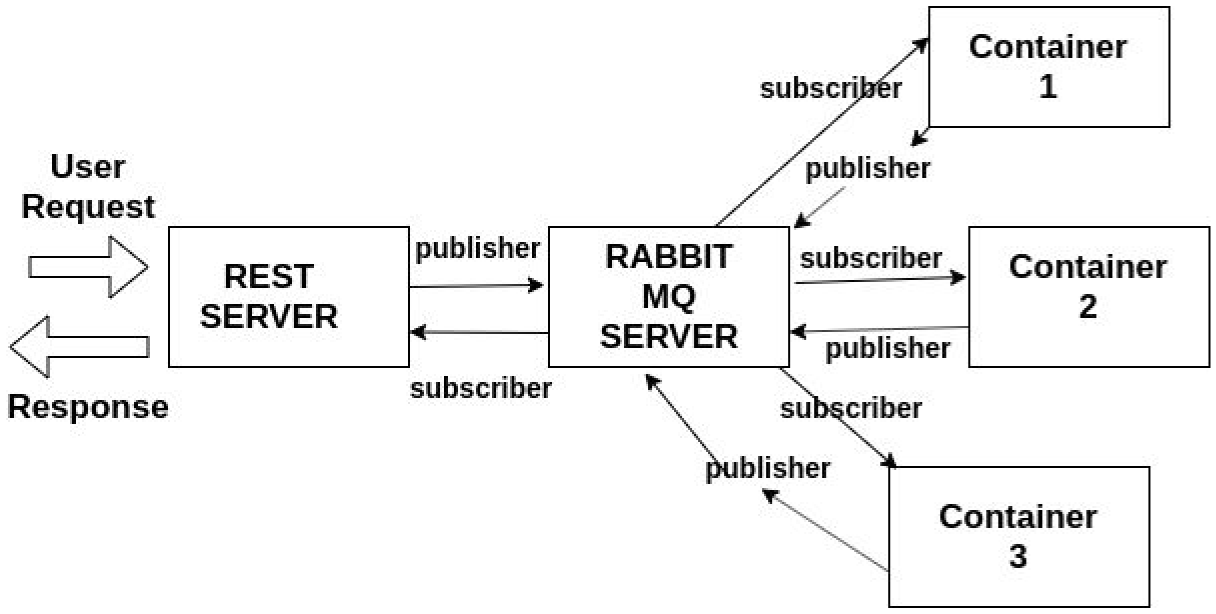


Fig. 1.

The complete architecture a single physical system is depicted in figure 1 above. The user requests to the Rest Server/API Gateway. Based on his Pipeline configuration, microservice containers are automatically fetched into the memory.These containers communicate via Rabbit MQ Server which works on the basis of publisher subscriber pattern. The REST server in our system is multi threaded which is creating separate thread for every user.

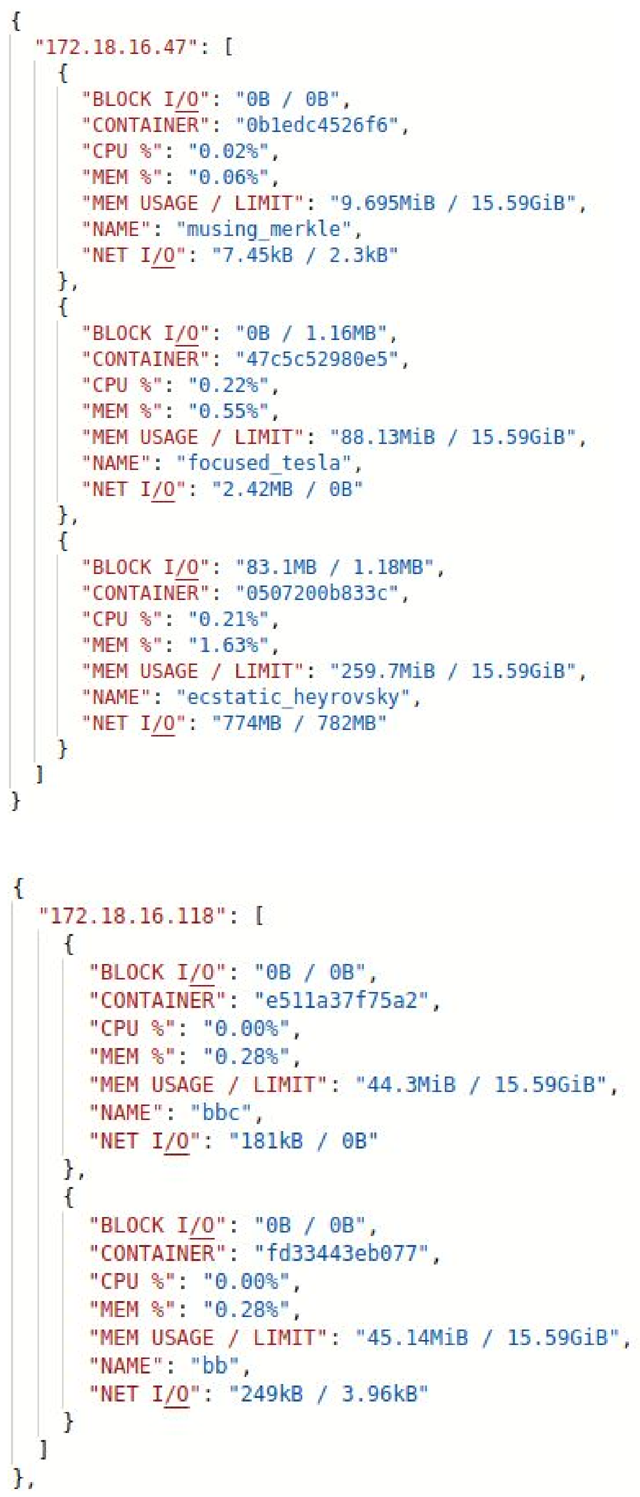
Interface for Admin

For admin, we have various endpoints for monitoring the overall system.Some of the endpoints are listed below.

1. Container Health Stats

   Endpoint : http://172.18.16.118:6000/containers/stats **Method** : GET

   When admin wishes to monitor statistics consumed resources, he requests the central monitoring server which collects various performance statistics from containers on every physical machine and returns them to the Admin in the following format. Various performance stats returned to the Admin for each container on every physical machine are listed below.



```
{
  "172.18.16.47": [
    {
      "BLOCK I/O": "0B / 0B",
      "CONTAINER": "0b1edc4526f6",
      "CPU %": "0.02%",
      "MEM %": "0.06%",
      "MEM USAGE / LIMIT": "9.695MiB / 15.59GiB",
      "NAME": "musing_merkle",
      "NET I/O": "7.45kB / 2.3kB"
    },
    {
      "BLOCK I/O": "0B / 1.16MB",
      "CONTAINER": "47c5c52980e5",
      "CPU %": "0.22%",
      "MEM %": "0.55%",
      "MEM USAGE / LIMIT": "88.13MiB / 15.59GiB",
      "NAME": "focused_tesla",
      "NET I/O": "2.42MB / 0B"
    },
    {
      "BLOCK I/O": "83.1MB / 1.18MB",
      "CONTAINER": "0507200b833c",
      "CPU %": "0.21%",
      "MEM %": "1.63%",
      "MEM USAGE / LIMIT": "259.7MiB / 15.59GiB",
      "NAME": "ecstatic_heyrovsky",
      "NET I/O": "774MB / 782MB"
    }
  ]
}
```

```
{
  "172.18.16.118": [
    {
      "BLOCK I/O": "0B / 0B",
      "CONTAINER": "e511a37f75a2",
      "CPU %": "0.00%",
      "MEM %": "0.28%",
      "MEM USAGE / LIMIT": "44.3MiB / 15.59GiB",
      "NAME": "bbc",
      "NET I/O": "181kB / 0B"
    },
    {
      "BLOCK I/O": "0B / 0B",
      "CONTAINER": "fd33443eb077",
      "CPU %": "0.00%",
      "MEM %": "0.28%",
      "MEM USAGE / LIMIT": "45.14MiB / 15.59GiB",
      "NAME": "bb",
      "NET I/O": "249kB / 3.96kB"
    }
  ]
},
```

Fig. 2

Above figure depicts how the admin can monitor the performance of all the containers at every physical node.

2. Resource Usage Pricing

   Endpoint : http://172.18.16.118:6000/users/pricing **Method** : GET

   The Admin can request the current pricing per user based on his/her usage and agreements.