De La Salle University
Computer Technology Department

NSDSYST

Members: _____ Group No.: _____

Date: _____

Hands-On 2 – Message Queue on Python

1 Introduction

Message queueing allows applications to communicate by sending messages to each other. The message queue provides temporary storage when the destination program is busy or not connected. This storing of messages allows asynchronous communication between different components or systems which increase reliability scalability and flexibility. Message queues are achieved by using a message broker like RabbitMQ to pass messages. In this laboratory, the RabbitMQ message broker is used to hold the message queue while Python programs with the Pika library is used to create producer and consumer programs.

2 Objectives

Objectives for the laboratory session are:

- Understand how message queues work in distributed system
- Create a server that executes code for client

3 Requirements

- Ubuntu Desktop with Virt-Manager or Virtual-Box
- Ubuntu Server virtual machine

4 Lab Procedures

1. Download and install docker on the ubuntu virtual machine by typing in the following commands:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

2. Install RabbitMQ on Docker by typing in the command:

```
sudo docker run -d --hostname rabbit-svr --name nsdsyst-rmq -p 15672:15672 -p
 5672:5672  -e RABBITMQ_DEFAULT_USER=user -e RABBITMQ_DEFAULT_PASS=password
 rabbitmq:3-management
```

Note: This command is a one-line command

3. Using a browser, connect to the RabbitMQ server by typing the in the URL `http://<ubuntu server ip address>:15672`. Login page for RabbitMQ server is shown in Fig. 1.

Fig. 1 RabbitMQ server login page

4. After logging in to the RabbitMQ server, command console using the username user and password “password”. The RabbitMQ dashboard is shown in Fig. 2.

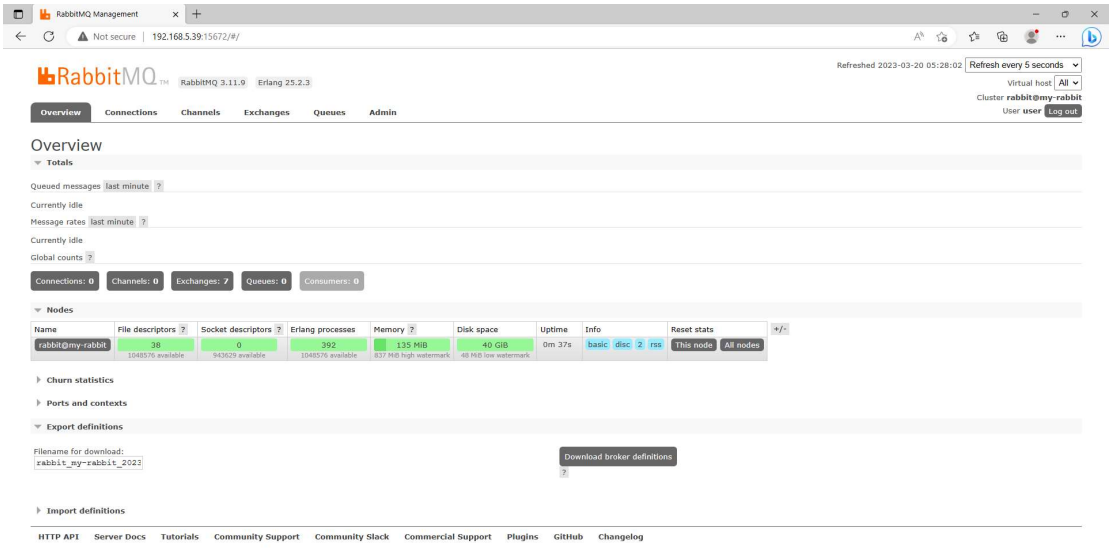

Fig. 2 RabbitMQ server dashboard

5. In the management dashboard, go to the Admin tab and create a user account with username “`rabbituser`” with password “`rabbit1234`”. This user account is to be used by programs to connect to the message queue broker.

6. Type in the following the program for the server:

```
1  import pika
2
3  credentials = pika.PlainCredentials('rabbituser','rabbit1234')
4
5  connection = pika.BlockingConnection(pika.ConnectionParameters('<ip address
6  rabbit broker>',5672,'/',credentials))
7  channel = connection.channel()
8
9  channel.queue_declare(queue='rqueue')
10
11 def callback(ch, method, properties, body):
12     print(" [x] Received %r" % body)
13
14 channel.basic_consume('rqueue', callback)
15
16 print(' [*] Waiting for messages. To exit press CTRL+C')
17 channel.start_consuming()
```

Do not forget to replace the correct IP address of the RabbitMQ broker in line 5.

7. Run the server and wait for data from remote client.
8. Type in the following the program for the remote client:

```
1  import pika, json
2
3  credentials = pika.PlainCredentials('rabbituser','rabbit1234')
4
5  connection = pika.BlockingConnection(pika.ConnectionParameters(' ip address
6  rabbit broker ',5672,'/',credentials))
7  channel = connection.channel()
8
9  channel.queue_declare(queue='rqueue')
10
11 name=input('Enter your name: ')
12 msg=input('Enter your message: ')
13
14 msg_dict={'name':name, 'msg':msg}
15
16 print (msg_dict)
17 msg_json=json.dumps(msg_dict)
18
19 channel.basic_publish(exchange='', routing_key='rqueue', body=msg_json)
20 print(" [x] Sent '%s'",(msg))
21 connection.close()
```

Do not forget to replace the correct IP address of the RabbitMQ broker in line 5.

9. Run the remote client and observe the output at the remote server. Describe the output at the remote server.

10. Run another server in other machine, this means that there are two servers running. Run the remote client and observe the output on the two servers.

11. Type in the following the program for the server:

```
1  import pika
2
3  credentials = pika.PlainCredentials('rabbituser','rabbit1234')
4
5  connection =
6  pika.BlockingConnection(pika.ConnectionParameters('192.168.5.39',5672,'/',cr
7  edentials))
8  channel = connection.channel()
9
10 channel.exchange_declare(exchange='logs', exchange_type='fanout')
11 result = channel.queue_declare(queue='', exclusive=True)
12 queue_name = result.method.queue
13
14 channel.queue_bind(exchange='logs', queue=queue_name)
15
16 def callback(ch, method, properties, body):
17     print(" [x] Received %r" % body)
18
19 channel.basic_consume( queue=queue_name, on_message_callback=callback,
20 auto_ack=True)
21
22 print(' [*] Waiting for messages. To exit press CTRL+C')
23 channel.start_consuming()
```

Do not forget to replace the correct IP address of the RabbitMQ broker in line 5.

12. Run the server and wait for data from remote client.

13. Type in the following the program for the remote client:

```
1  import pika, json
2
3  credentials = pika.PlainCredentials('rabbituser','rabbit1234')
4
5  connection =
6  pika.BlockingConnection(pika.ConnectionParameters('192.168.5.39',5672,'/',cr
7  edentials))
8  channel = connection.channel()
9
10 channel.exchange_declare(exchange='logs', exchange_type='fanout')
11
12 name=input('Enter your name: ')
13 msg=input('Enter your message: ')
14
15 msg_dict={'name':name, 'msg':msg}
16
17 print (msg_dict)
18 msg_json=json.dumps(msg_dict)
19
20 channel.basic_publish(exchange='logs', routing_key='', body=msg_json)
```

```
21
22  print(" [x] Sent '%s'",(msg_dict))
23  connection.close()
```

Do not forget to replace the correct IP address of the RabbitMQ broker in line 5.

14. Run the remote client and observe the output at the remote server. Describe the output at the remote server.

15. Run another server in other machine, this means that there are two servers running. Run the remote client and observe the output on the two servers. What is the difference from the previous example?

5 Reflection / Questions

1. What is a message queue?

2. What are the different techniques of distributing messages in message queues?

3. How can message queues be used in distributed systems?