

# Project 2

## Exercise 1:

Firstly, based on information in exercise 1 and the introduction, we created 5 classes (student.py, teaching\_assistant.py, demonstrator\_dm.py, demonstrator\_cc.py and module\_coordinator.py).

**Module\_coordinator.py:** For this class, we should declare 4 queues and set the connection to localhost. First, we set assignment\_exchange and set the type to topic'. Then for Result\_queue, cloud\_computing\_queue, data\_mining\_queue, and validation queue. And use their name as routing\_key. For function callback, we add a str Module name to know which modules are submitted, and when the status is validated we can set it to confirmed and send it to result\_queue.

```

1 import pika, sys, os, json
2 def main():
3     #docker
4     #credentials = pika.PlainCredentials("guest", "guest")
5     #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))
6     #local
7     connection = pika.BlockingConnection(
8         pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
9                                   credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
10                                                                           os.environ.get("RABBITMQ_PASSWORD", "guest"))))
11
12     channel = connection.channel()
13
14     channel.exchange_declare(exchange='assignment_exchange', exchange_type='topic')
15     channel.queue_declare(queue='Result_queue')
16     channel.queue_bind(exchange='assignment_exchange', queue='Result_queue', routing_key='Result_queue')
17     channel.queue_declare(queue='Cloud_computing_queue')
18     channel.queue_bind(exchange='assignment_exchange', queue='Cloud_computing_queue', routing_key='Cloud_computing_queue')
19     channel.queue_declare(queue='Data_mining_queue')
20     channel.queue_bind(exchange='assignment_exchange', queue='Data_mining_queue', routing_key='Data_mining_queue')
21     channel.queue_declare(queue='Validation_queue')
22     channel.queue_bind(exchange='assignment_exchange', queue='Validation_queue', routing_key='Validation_queue')
23
24     def callback(ch, method, properties, body):
25
26         #channel.basic_publish(exchange='assignment_exchange', body='')
27         assignment = json.loads(body)
28         print("Received "+str(assignment['StudentID'])+" "+str(assignment['Module'])+" "+str(assignment['status'])+" and submit it to Brightspace")
29         if(assignment['status']=='validated'):
30             assignment['status']='confirmed'
31         #else:
32         #     channel.basic_publish(exchange='assignment_exchange', routing_key='', body=json.dumps(assignment))
33
34     channel.basic_consume(queue='Result_queue', on_message_callback=callback, auto_ack=False)
35
36     print(' [*] Module Coordinator is Waiting...')
37     channel.start_consuming()
38
39 if __name__ == '__main__':
40     try:
41         main()
42     except KeyboardInterrupt:
43         print('Interrupted')
44         try:
45             sys.exit(0)
46         except SystemExit:
47             os._exit(0)

```

**Student.py:** Set connection to localhost. we changed the two assignments to data mining (dm) and cloud computing (cc), which show the Module name, student ID, answer, and status. And also set the routing\_key which links to module\_coordinator settings.

```

1 import pika
2 import json,os
3
4 #docker
5 #credentials = pika.PlainCredentials("guest", "guest")
6 #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))
7
8 #local
9 connection = pika.BlockingConnection(
10     pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
11                             credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
12                             os.environ.get("RABBITMQ_PASSWORD", "guest"))))
13
14 channel = connection.channel()
15
16 dm_assignment = json.dumps({'Module':'Data_mining','StudentID': 1234, 'answer': 'some answers, probably correct','status':'submitted'})
17 cc_assignment = json.dumps({'Module':'Cloud_computing','StudentID': 5678, 'answer': 'some answers, probably correct','status':'submitted'})
18
19 #send assignment for correction
20 channel.basic_publish(exchange='assignment_exchange',routing_key='Data_mining_queue', body=dm_assignment)
21 print("Data mining Assignment sent for correction")
22 channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment)
23 print("Cloud computing Assignment sent for correction")
24 connection.close()
25
26
27
28
29

```

**Teaching\_assistant.py:** Set connection to localhost. When TA receive a corrected assignment, set the status to validated and it will send to Result\_queue, so we change the routing\_key to Result\_queue.

```

1 import pika, sys, os,json
2
3 def main():
4     #docker
5     #credentials = pika.PlainCredentials("guest", "guest")
6     #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))
7
8     #local
9     connection = pika.BlockingConnection(
10         pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
11                                 credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
12                                 os.environ.get("RABBITMQ_PASSWORD", "guest"))))
13
14     #credentials = pika.PlainCredentials(username='guest', password='guest')
15     #parameters = pika.ConnectionParameters(host='rabbitmq', credentials=credentials)
16
17     channel = connection.channel()
18
19
20     def callback(ch, method, properties, body):
21         assignment = json.loads(body)
22         print("Received "+str(assignment['StudentID'])+" "+str(assignment['status'])+" "+str(assignment['Module']))
23         if(assignment['status']=='corrected'):
24             assignment['status']='validated'
25             channel.basic_publish(exchange='assignment_exchange',routing_key='Result_queue',body=json.dumps(assignment))
26         #else:
27         #     channel.basic_publish(exchange='assignment_exchange',routing_key='',body=json.dumps(assignment))
28
29     channel.basic_consume(queue='Validation_queue',on_message_callback=callback, auto_ack=False)
30
31     print(' [*] Teaching assistant is Waiting...')
32     channel.start_consuming()
33
34 if __name__ == '__main__':
35     try:
36         main()
37     except KeyboardInterrupt:
38         print('Interrupted')
39         try:
40             sys.exit(0)
41         except SystemExit:
42             os._exit(0)

```

**Demonstrator\_dm.py:** Firstly, we set the connection to localhost. Then we add the “Module” name into print str to see which module received, and then set the status to corrected after it submitted, then consume to its module “data\_mining\_queue”.

```
1 import pika, sys, os, json
2 def main():
3     #docker
4     #credentials = pika.PlainCredentials("guest", "guest")
5     #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))
6
7     #local
8     connection = pika.BlockingConnection(
9         pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
10                                     credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
11                                     os.environ.get("RABBITMQ_PASSWORD", "guest"))))
12     channel = connection.channel()
13
14     def callback(ch, method, properties, body):
15
16         assignment = json.loads(body)
17         print("Received "+str(assignment['StudentID'])+" "+str(assignment['Module'])+" "+str(assignment['status']))
18         if(assignment['status']=='submitted'):
19             assignment['status']='corrected'
20             channel.basic_publish(exchange='assignment_exchange', routing_key='Validation_queue', body=json.dumps(assignment))
21
22     channel.basic_consume(queue='Data_mining_queue', on_message_callback=callback, auto_ack=False)
23     print(' [*] Demonstrator is Waiting...')
24     channel.start_consuming()
25
26 if __name__ == '__main__':
27     try:
28         main()
29     except KeyboardInterrupt:
30         print('Interrupted')
31     try:
32         sys.exit(0)
33     except SystemExit:
34         os._exit(0)
```

**Demonstrator\_cc.py:** Firstly, we set the connection to localhost. Then we add the “Module” name into print str to see which module received, and then set the status to corrected after it submitted, then consume to its module “cloud\_computing\_queue”.

```
1 import pika, sys, os, json
2 def main():
3     #docker
4     #credentials = pika.PlainCredentials("guest", "guest")
5     #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))
6
7     #local
8     connection = pika.BlockingConnection(
9         pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
10                                     credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
11                                     os.environ.get("RABBITMQ_PASSWORD", "guest"))))
12     channel = connection.channel()
13
14     def callback(ch, method, properties, body):
15
16         assignment = json.loads(body)
17         print("Received "+str(assignment['StudentID'])+" "+str(assignment['Module'])+" "+str(assignment['status']))
18         if(assignment['status']=='submitted'):
19             assignment['status']='corrected'
20             channel.basic_publish(exchange='assignment_exchange', routing_key='Validation_queue', body=json.dumps(assignment))
21     channel.basic_consume(queue='Cloud_computing_queue', on_message_callback=callback, auto_ack=False)
22     print(' [*] Demonstrator is Waiting...')
23     channel.start_consuming()
24
25 if __name__ == '__main__':
26     try:
27         main()
28     except KeyboardInterrupt:
29         print('Interrupted')
30     try:
31         sys.exit(0)
32     except SystemExit:
33         os._exit(0)
```

After fill the files, we Pull a RabbitMQ Docker image from DockerHub with

### **docker pull rabbitmq:3.13-rc-management**

```
((base) jerryzzr@JerryZzrdeMacBook-Pro ex2 % docker pull rabbitmq:3.13-rc-management
3.13-rc-management: Pulling from library/rabbitmq
aece8493d397: Pull complete
19047c6a5ad1: Pull complete
9a2e556e3287: Pull complete
08587746fc47: Pull complete
c0361febb350: Pull complete
f9ef71a98d2a: Pull complete
727d5b9dab1b: Pull complete
d5ec5fe27c9e: Pull complete
65b055f1a80c: Pull complete
d4d8d6eee484: Pull complete
fe1b2bde4342: Pull complete
208d01d87e58: Pull complete
Digest: sha256:3687bfd4532e8a7801c12b3acdcbb1521647e1931a09bd48a15ccaeba4d43d759
Status: Downloaded newer image for rabbitmq:3.13-rc-management
docker.io/library/rabbitmq:3.13-rc-management

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview rabbitmq:3.13-rc-management
```

Then we start a RabbitMQ container with

### **docker run -d -p 5672:5672 -p 15672:15672 rabbitmq:3.13-rc-management**

(copy from practical 4)

```
((base) jerryzzr@JerryZzrdeMacBook-Pro ex2 % docker run -d -p 5672:5672 -p 15672:15672 rabbitmq:3.13-rc-management
46325ab562642b15e88db6b520b240a3452f6169e0fe7e2d60f75d0e3040db9a
```

After starting the container we run the py file that we created before,

With this process:

1. **python module\_coordinator.py**
2. **python demonstrator\_dm.py**
3. **python demonstrator\_cc.py**
4. **python teaching\_assistant.py**
5. **python student.py**

**For student.py:**

Students submit their assignments to the assignment exchange and set the status of their submissions to “**submitted**” For correction.

```
(base) jerryzzr@dhcp-892bf7f7 ex2 % python student.py
```

Data minging Assignment sent for correction

Cloud computiong Assignment sent for\_correction

Then the data mining demonstrator listens to the data mining queue. Received assignment from student 1234, changed the status to “corrected” and returned it to the assignment exchange.

```
[(base) jerryzzr@dhcp-892bf7f7 ex2 % python demonstrator_dm.py
```

```
  [*] Demonstrator is Waiting...
```

```
  Received 1234,Data_minging,submitted
```

The cloud computing demonstrator listens to the cloud computing queue. Received assignment from student 5678, changed the status to “corrected”and returns it to the assignment exchange.

```
(base) jerryzzr@dhcp-892bf7f7 ex2 % python demonstrator_cc.py
```

```
  [*] Demonstrator is Waiting...
```

```
  Received 5678,Cloud_computing,submitted
```

The teaching assistant listens to the validation queue. Once an assignment is received from demonstrators, it changes the status to “**validated**” and returns it to the assignment exchange.

```
[(base) jerryzzr@dhcp-892bf7f7 ex2 % python teaching_assistant.py
[*] Teaching assistant is Waiting...
Received 1234,corrected,Data_minging
Received 5678.corrected.Cloud computing
```

We can see we received student number 1234 with his Data\_mining assignment and student number 5678 with his Cloud\_computing assignment and came with validated status. The module coordinator listens to the results queue. Once an assignment is received, the status changes to “**confirmed.**”

```
.(base) jerryzzr@dhcp-892bf7f7 ex2 % python module_coordinator.py
[*] Module Coordinator is Waiting...
Received 1234,Data_minging,validated, and submit it to Brightspace
Received 5678,Cloud_computing,validated, and submit it to Brightspace
```

## Exercise 2:

Dockerize your solution of Exercise 1. Use docker-compose to run six containers.

First I copied the python file in ex1 to ex2 and corrected student.py to create 5 assignments. (two data mining assignments and three cloud computing assignments)

```
ex2 > student.py > ...
1  import pika
2  import json,os
3
4  #docker
5  credentials = pika.PlainCredentials("guest", "guest")
6  connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))
7
8  #local
9  # connection = pika.BlockingConnection(
10 # pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
11 #                             credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
12 #                             os.environ.get("RABBITMQ_PASSWORD", "guest"))))
13
14
15  channel = connection.channel()
16
17  dm_assignment1 = json.dumps({'Module':'Data_mining','StudentID': 1234, 'answer': 'some answers, probably correct','status':'submitted'})
18  dm_assignment2 = json.dumps({'Module':'Data_mining','StudentID': 1432, 'answer': 'some answers, probably correct','status':'submitted'})
19  cc_assignment1 = json.dumps({'Module':'Cloud_computing','StudentID': 5678, 'answer': 'some answers, probably correct','status':'submitted'})
20  cc_assignment2= json.dumps({'Module':'Cloud_computing','StudentID': 6789, 'answer': 'some answers, probably correct','status':'submitted'})
21  cc_assignment3 = json.dumps({'Module':'Cloud_computing','StudentID': 6987, 'answer': 'some answers, probably correct','status':'submitted'})
22
23
24  #send assignment for correction
25  channel.basic_publish(exchange='assignment_exchange',routing_key='Data_mining_queue', body=dm_assignment1)
26  print("Data mining Assignment sent for correction")
27  channel.basic_publish(exchange='assignment_exchange',routing_key='Data_mining_queue', body=dm_assignment2)
28  print("Data mining Assignment sent for correction")
29  channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment1)
30  print("Cloud computing Assignment sent for correction")
31  channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment2)
32  print("Cloud computing Assignment sent for correction")
33  channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment3)
34  print("Cloud computing Assignment sent for correction")
35  connection.close()
36
37
```

Then Create **docker-compose.yaml** and **DockerFile**.

For **docker-compose.yaml** Which has 5 classes under services, they are

student link to Dockerfile\_student,

module\_coordinator link to Dockerfile\_modCo,

teaching\_assistant link to Dockerfile\_TA,

dm\_demo link to Dockerfile\_DM,

cc\_demo link to Dockerfile\_CC.

All these depends on rabbitmq

For class rabbitmq:

it pulls a RabbitMQ Docker image first then start the RabbitMQ container.

Set localhost:15672 username and password to 'guest'.

```
ex2 > docker-compose.yml
1  version: '3'
2  networks:
3    rabbitmq_go_net:
4      driver: bridge
5  services:
6    rabbitmq:
7      image: "rabbitmq:3.13-rc-management"
8      container_name: rabbitmq_container
9      ports:
10     - "5672:5672" # AMQP
11     - "15672:15672" # Management UI
12     environment:
13       RABBITMQ_DEFAULT_USER: guest
14       RABBITMQ_DEFAULT_PASS: guest
15     networks:
16       - rabbitmq_go_net
17
18    student:
19      build:
20        context: .
21        dockerfile: Dockerfile_student
22      restart: unless-stopped
23      depends_on:
24        - rabbitmq
25      networks:
26        - rabbitmq_go_net
27
28    module_coordinator:
29      build:
30        context: .
31        dockerfile: Dockerfile_modCo
32      restart: unless-stopped
33      depends_on:
34        - rabbitmq
35      networks:
36        - rabbitmq_go_net
37
38    teaching_assistant:
39      build:
40        context: .
41        dockerfile: Dockerfile_TA
42      restart: unless-stopped
43      depends_on:
44        - rabbitmq
45      networks:
46        - rabbitmq_go_net
47
48    dm_demo:
49      build:
50        context: .
51        dockerfile: Dockerfile_DM
52      restart: unless-stopped
53      depends_on:
54        - rabbitmq
55      networks:
56        - rabbitmq_go_net
57
58    cc_demo:
59      build:
60        context: .
61        dockerfile: Dockerfile_CC
62      restart: unless-stopped
63      depends_on:
64        - rabbitmq
65      networks:
66        - rabbitmq_go_net
```



## 5 DockerFile:

```
ex2 > 🚢 Dockerfile_modCo > 📦 FROM
1 FROM python:slim-buster
2 WORKDIR app
3 RUN pip install pika
4 COPY module_coordinator.py ./
5 CMD python module_coordinator.py

ex2 > 🚢 Dockerfile_DM > 📦 FROM
1 FROM python:slim-buster
2 WORKDIR app
3 RUN pip install pika
4 COPY demonstrator_dm.py ./
5 CMD python demonstrator_dm.py

ex2 > 🚢 Dockerfile_modCo > 📦 FROM
1 FROM python:slim-buster
2 WORKDIR app
3 RUN pip install pika
4 COPY module_coordinator.py ./
5 CMD python module_coordinator.py

ex2 > 🚢 Dockerfile_student > 📦 FROM
1 FROM python:slim-buster
2 WORKDIR app
3 RUN pip install pika
4 COPY student.py ./
5 CMD python student.py

ex2 > 🚢 Dockerfile_TA > 📦 FROM
1 FROM python:slim-buster
2 WORKDIR app
3 RUN pip install pika
4 COPY teaching_assistant.py ./
5 CMD python teaching_assistant.py
```

After these files created, we run **docker compose up**:



Students sent 5 assignments:

```
ex2-student-1 | Data minging Assignment sent for correction
ex2-student-1 | Data minging Assignment sent for correction
ex2-student-1 | Cloud computiong Assignment sent for correction
ex2-student-1 | Cloud computiong Assignment sent for correction
ex2-student-1 | Cloud computiong Assignment sent for correction
```

Demonstrator CC received 3 cloud computing assignments.

```
ex2-cc_demo-1 | [*] Demonstrator is Waiting...
ex2-module_coordinator-1 exited with code 0
ex2-cc_demo-1 | Received 5678,Cloud_computing,submitted
ex2-cc_demo-1 | Received 6789,Cloud_computing,submitted
ex2-cc_demo-1 | Received 6987,Cloud_computing,submitted
```

Demonstrator DM received 2 data mining assignments.

```
ex2-dm_demo-1 | [*] Demonstrator is Waiting...
ex2-dm_demo-1 | Received 1234,Data_minging,submitted
ex2-dm_demo-1 | Received 1432,Data_minging,submitted
```

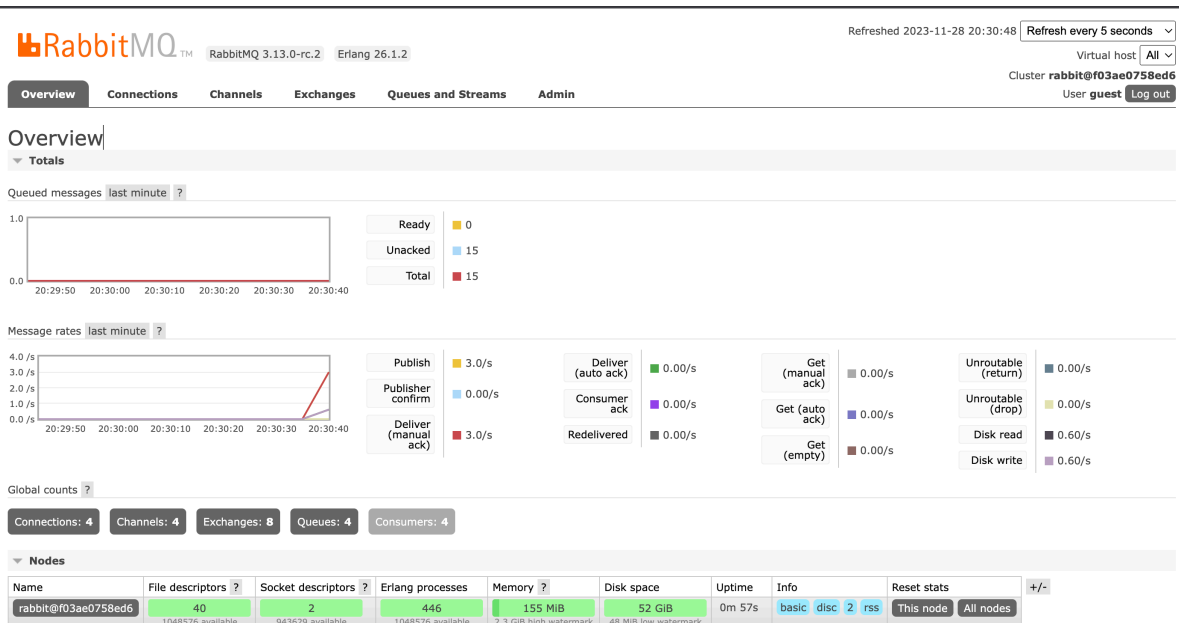
TA received 5 students' assignments from the demonstrator.

```
ex2-teaching_assistant-1 | [*] Teaching assistant is Waiting...
ex2-teaching_assistant-1 | Received 1234,corrected,Data_minging
ex2-teaching_assistant-1 | Received 1432,corrected,Data_minging
ex2-teaching_assistant-1 | Received 5678,corrected,Cloud_computing
ex2-teaching_assistant-1 | Received 6789,corrected,Cloud_computing
ex2-teaching_assistant-1 | Received 6987,corrected,Cloud_computing
```

Module coordinator received 5 assignments from TA.

```
ex2-module_coordinator-1 | [*] Module Coordinator is Waiting...
ex2-module_coordinator-1 | Received 1234,Data_minging,validated, and submit it to Brightspace
ex2-module_coordinator-1 | Received 1432,Data_minging,validated, and submit it to Brightspace
ex2-module_coordinator-1 | Received 5678,Cloud_computing,validated, and submit it to Brightspace
ex2-module_coordinator-1 | Received 6789,Cloud_computing,validated, and submit it to Brightspace
ex2-module_coordinator-1 | Received 6987,Cloud_computing,validated, and submit it to Brightspace
```

Go to localhost check the status:



## Queues

▼ All queues (4)

Pagination

Page 1 of 1 - Filter:  ☐ Regex ?

Displaying 4 items , pag

Overview					Messages			Message rates			+/-
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/	Cloud_computing_queue	classic		<span>running</span>	0	42	42	0.00/s	0.00/s	0.00/s	
/	Data_mining_queue	classic		<span>running</span>	0	28	28	0.00/s	0.00/s	0.00/s	
/	Result_queue	classic		<span>running</span>	0	140	140	0.00/s	0.00/s	0.00/s	
/	Validation_queue	classic		<span>running</span>	0	105	105	0.00/s	0.00/s	0.00/s	

► Add a new queue

4 queue is running now.