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.

```python
import pika, sys, os,json
def main():
    #docker
    #credentials = pika.PlainCredentials("guest", "guest")
    #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))
    #local
    connection = pika.BlockingConnection(
    pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
                    credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
                    os.environ.get("RABBITMQ_PASSWORD", "guest"))))

    channel = connection.channel()

    channel.exchange_declare(exchange='assignment_exchange',exchange_type='topic')
    channel.queue_declare(queue='Result_queue')
    channel.queue_bind(exchange='assignment_exchange',queue='Result_queue',routing_key='Result_queue')
    channel.queue_declare(queue='Cloud_computing_queue')
    channel.queue_bind(exchange='assignment_exchange',queue='Cloud_computing_queue',routing_key='Cloud_computing_queue')
    channel.queue_declare(queue='Data_mining_queue')
    channel.queue_bind(exchange='assignment_exchange',queue='Data_mining_queue',routing_key='Data_mining_queue')
    channel.queue_declare(queue='Validation_queue')
    channel.queue_bind(exchange='assignment_exchange',queue='Validation_queue',routing_key='Validation_queue')

    def callback(ch, method, properties, body):

        #channel.basic_publish(exchange='assignment_exchange',body='')
        assignment = json.loads(body)
        print("Received "+str(assignment['StudentID'])+","+str(assignment['Module'])+","+str(assignment['status'])+", and submit it to Brightspace")
        if(assignment['status']=='validated'):
            assignment['status']='confirmed'
        #else:
        #    channel.basic_publish(exchange='assignment_exchange',routing_key='',body=json.dumps(assignment))

    channel.basic_consume(queue='Result_queue', on_message_callback=callback, auto_ack=False)

    print(' [*] Module Coodinator is Waiting...')
    channel.start_consuming()

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            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_coodinator settings.

```python
import pika
import json,os

#docker
#credentials = pika.PlainCredentials("guest", "guest")
#connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))

#local
connection = pika.BlockingConnection(
pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
                credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
                os.environ.get("RABBITMQ_PASSWORD", "guest"))))


channel = connection.channel()

dm_assignment = json.dumps({'Module':"Data_minging",'StudentID': 1234, 'answer': 'some answers, probably correct','status':'submitted'})
cc_assignment = json.dumps({'Module':"Cloud_computing",'StudentID': 5678, 'answer': 'some answers, probably correct','status':'submitted'}



#send assignment for correction
channel.basic_publish(exchange='assignment_exchange',routing_key='Data_mining_queue', body=dm_assignment)
print("Data minging Assignment sent for correction")
channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment)
print("Cloud computiong Assignment sent for correction")
connection.close()
```

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.

```python
import pika, sys, os,json

def main():
    #docker
    #credentials = pika.PlainCredentials("guest", "guest")
    #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))

    #local
    connection = pika.BlockingConnection(
    pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
                    credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
                    os.environ.get("RABBITMQ_PASSWORD", "guest"))))

    #credentials = pika.PlainCredentials(username='guest', password='guest')
    #parameters = pika.ConnectionParameters(host='rabbitmq', credentials=credentials)

    channel = connection.channel()


    def callback(ch, method, properties, body):
        assignment = json.loads(body)
        print("Received "+str(assignment['StudentID'])+","+str(assignment['status'])+","+str(assignment['Module']))
        if(assignment['status']=='corrected'):
            assignment['status']='validated'
            channel.basic_publish(exchange='assignment_exchange',routing_key='Result_queue',body=json.dumps(assignment))
        #else:
        #    channel.basic_publish(exchange='assignment_exchange',routing_key='',body=json.dumps(assignment))

    channel.basic_consume(queue='Validation_queue',on_message_callback=callback, auto_ack=False)

    print(' [*] Teaching assistant is Waiting...')
    channel.start_consuming()

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            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".

```python
import pika, sys, os,json
def main():
    #docker
    #credentials = pika.PlainCredentials("guest", "guest")
    #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))

    #local
    connection = pika.BlockingConnection(
    pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
                    credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
                     os.environ.get("RABBITMQ_PASSWORD", "guest"))))
    channel = connection.channel()

    def callback(ch, method, properties, body):

        assignment = json.loads(body)
        print("Received "+str(assignment['StudentID'])+","+str(assignment['Module'])+","+str(assignment['status']))
        if(assignment['status']=='submitted'):
            assignment['status']='corrected'
            channel.basic_publish(exchange='assignment_exchange',routing_key='Validation_queue',body=json.dumps(assignment))

    channel.basic_consume(queue='Data_mining_queue', on_message_callback=callback, auto_ack=False)
    print(' [*] Demonstrator is Waiting...')
    channel.start_consuming()

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            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".

```python
import pika, sys, os,json
def main():
    #docker
    #credentials = pika.PlainCredentials("guest", "guest")
    #connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))

    #local
    connection = pika.BlockingConnection(
    pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
                    credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
                    os.environ.get("RABBITMQ_PASSWORD", "guest"))))
    channel = connection.channel()

    def callback(ch, method, properties, body):

        assignment = json.loads(body)
        print("Received "+str(assignment['StudentID'])+","+str(assignment['Module'])+","+str(assignment['status']))
        if(assignment['status']=='submitted'):
            assignment['status']='corrected'
            channel.basic_publish(exchange='assignment_exchange',routing_key='Validation_queue',body=json.dumps(assignment))
    channel.basic_consume(queue='Cloud_computing_queue', on_message_callback=callback, auto_ack=False)
    print(' [*] Demonstrator is Waiting...')
    channel.start_consuming()

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            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: sha256:3687bfd4532e8a7801c12b3acdcb1521647e1931a09bd48a15ccaeba4d43d759
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 Coodinator 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 > ...

```python
import pika
import json,os

#docker
credentials = pika.PlainCredentials("guest", "guest")
connection = pika.BlockingConnection(pika.ConnectionParameters("rabbitmq", 5672, "/", credentials))

#local
# connection = pika.BlockingConnection(
# pika.ConnectionParameters(host=os.environ.get("RABBITMQ_HOSTNAME", "localhost"),
#                 credentials=pika.PlainCredentials(os.environ.get("RABBITMQ_USERNAME", "guest"),
#                 os.environ.get("RABBITMQ_PASSWORD", "guest"))))


channel = connection.channel()

dm_assignment1 = json.dumps({'Module':"Data_minging",'StudentID': 1234, 'answer': 'some answers, probably correct','status':'submitted'})
dm_assignment2 = json.dumps({'Module':"Data_minging",'StudentID': 1432, 'answer': 'some answers, probably correct','status':'submitted'})
cc_assignment1 = json.dumps({'Module':"Cloud_computing",'StudentID': 5678, 'answer': 'some answers, probably correct','status':'submitted'})
cc_assignment2= json.dumps({'Module':"Cloud_computing",'StudentID': 6789, 'answer': 'some answers, probably correct','status':'submitted'})
cc_assignment3 = json.dumps({'Module':"Cloud_computing",'StudentID': 6987, 'answer': 'some answers, probably correct','status':'submitted'})


#send assignment for correction
channel.basic_publish(exchange='assignment_exchange',routing_key='Data_mining_queue', body=dm_assignment1)
print("Data minging Assignment sent for correction")
channel.basic_publish(exchange='assignment_exchange',routing_key='Data_mining_queue', body=dm_assignment2)
print("Data minging Assignment sent for correction")
channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment1)
print("Cloud computiong Assignment sent for correction")
channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment2)
print("Cloud computiong Assignment sent for correction")
channel.basic_publish(exchange='assignment_exchange',routing_key='Cloud_computing_queue', body=cc_assignment3)
print("Cloud computiong Assignment sent for correction")
connection.close()
```

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.yaml

```yaml
version: '3'
networks:
  rabbitmq_go_net:
    driver: bridge
services:
  rabbitmq:
    image: "rabbitmq:3.13-rc-management"
    container_name: rabbitmq_container
    ports:
      - "5672:5672"  # AMQP
      - "15672:15672"  # Management UI
    environment:
      RABBITMQ_DEFAULT_USER: guest
      RABBITMQ_DEFAULT_PASS: guest
    networks:
      - rabbitmq_go_net

  student:
    build:
      context: .
      dockerfile: Dockerfile_student
    restart: unless-stopped
    depends_on:
      - rabbitmq
    networks:
      - rabbitmq_go_net

  module_coordinator:
    build:
      context: .
      dockerfile: Dockerfile_modCo
    restart: unless-stopped
    depends_on:
      - rabbitmq
    networks:
      - rabbitmq_go_net

  teaching_assistant:
    build:
      context: .
      dockerfile: Dockerfile_TA
    restart: unless-stopped
    depends_on:
      - rabbitmq
    networks:
      - rabbitmq_go_net

  dm_demo:
    build:
      context: .
      dockerfile: Dockerfile_DM
    restart: unless-stopped
    depends_on:
      - rabbitmq
    networks:
      - rabbitmq_go_net

  cc_demo:
    build:
      context: .
      dockerfile: Dockerfile_CC
    restart: unless-stopped
    depends_on:
      - rabbitmq
    networks:
      - 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
```

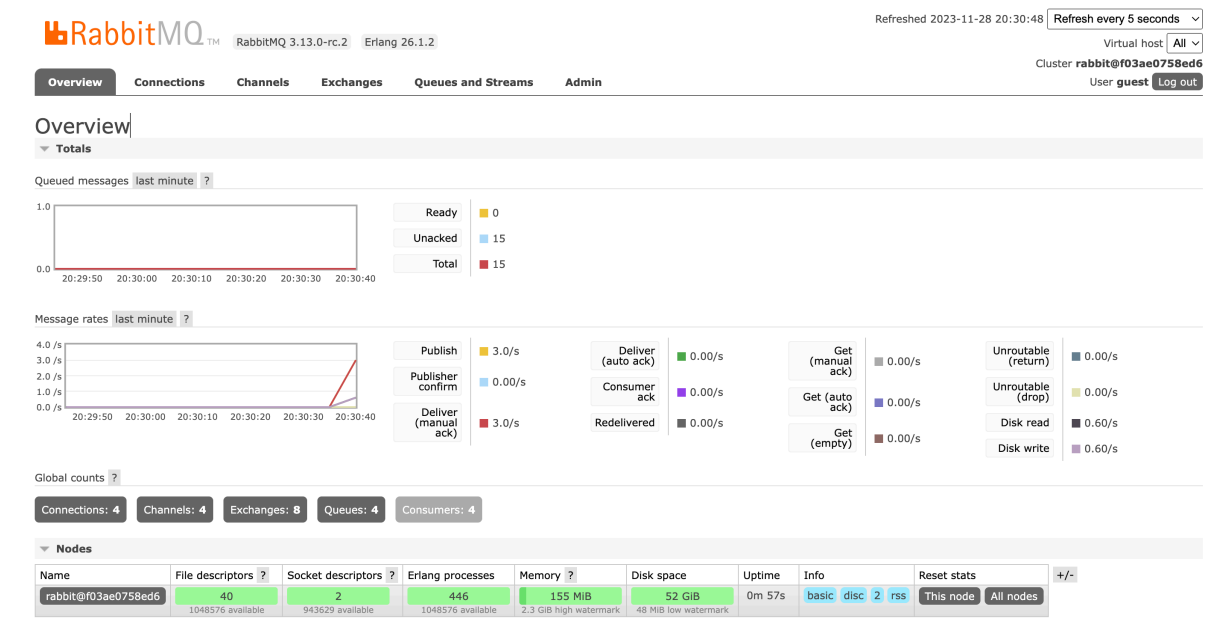
TA received 5 students' assignments from the demonstrator.

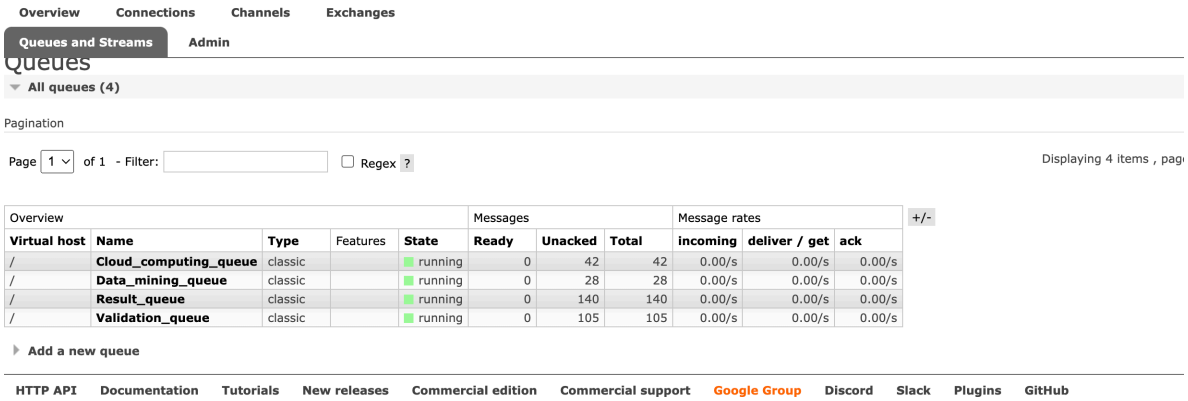
```
ex2-teaching_assistant-1  | [*] Teaching assistant is Waiting...
ex2-teaching_assistant-1  | Received 1234,corrected,Data_minging
ex2-teaching_assistant-1  | Received 1432,corrected,Data_minging
ex2-teaching_assistant-1  | Received 5678,corrected,Cloud_computing
ex2-teaching_assistant-1  | Received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





4 queue is running now.