Firstly, I created a file and stored the script from the website.

```
debjyotisarkar@sumonta-22341019:~/Desktop$ touch rabbitmq.sh
debjyotisarkar@sumonta-22341019:~/Desktop$ gedit rabbitmq.sh
debjyotisarkar@sumonta-22341019:~/Desktop$
```

rabbitmq.sh
~/Desktop

```
#!/bin/sh

sudo apt-get install curl gnupg apt-transport-https -y

## Team RabbitMQ's main signing key
curl -1sLf "https://keys.openpgp.org/vks/v1/by-fingerprint/0A9AF2115F4687BD29803A206B73A36E6026DFCA" | sudo gpg --dearmor | sudo tee /usr/share/keyrings/com.rabbitmq.team.gpg > /dev/null
## Community mirror of Cloudsmith: modern Erlang repository
curl -1sLf https://github.com/rabbitmq/signing-keys/releases/download/3.0/cloudsmith.rabbitmq-erlang.E495BB49CC4BBE5B.key | sudo gpg --dearmor | sudo tee /usr/share/keyrings/rabbitmq.E495BB49CC4BBE5B.gpg > /dev/null
## Community mirror of Cloudsmith: RabbitMQ repository
curl -1sLf https://github.com/rabbitmq/signing-keys/releases/download/3.0/cloudsmith.rabbitmq-server.9F4587F226208342.key | sudo gpg --dearmor | sudo tee /usr/share/keyrings/rabbitmq.9F4587F226208342.gpg > /dev/null
```

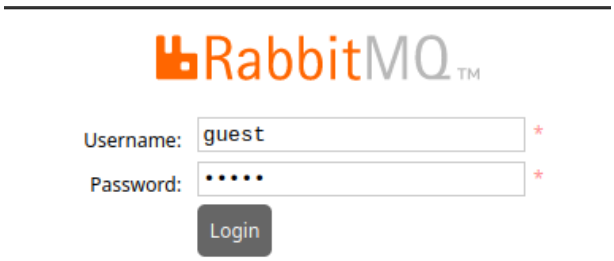
I used chmod +x to give permission to the file and ran the file to check if the server is working properly.

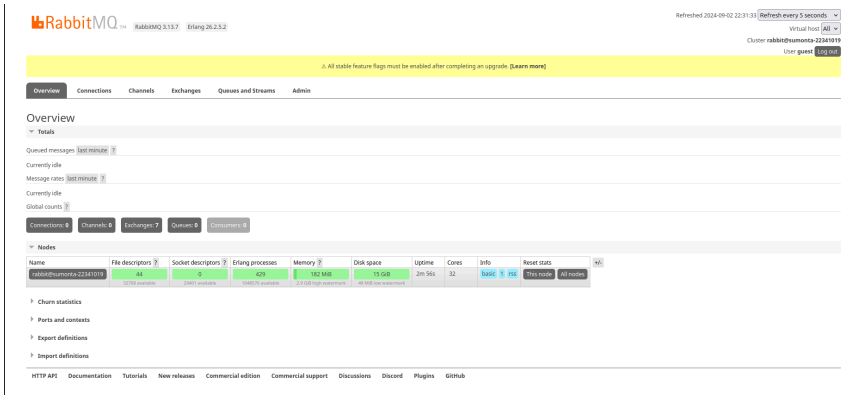
```
debjyotisarkar@sumonta-22341019:~/Desktop$ chmod +x rabbitmq.sh
debjyotisarkar@sumonta-22341019:~/Desktop$ ./rabbitmq.sh
[sudo] password for debjyotisarkar:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
Processing triggers for man-db (2.12.0-4build2) ...
debjyotisarkar@sumonta-22341019:~/Desktop$ sudo systemctl start rabbitmq-server
debjyotisarkar@sumonta-22341019:~/Desktop$ sudo systemctl status rabbitmq-server
● rabbitmq-server.service - RabbitMQ broker
     Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; >
     Active: active (running) since Mon 2024-09-02 22:28:32 +06; 1min 23s ago
   Main PID: 6652 (beam.smp)
      Tasks: 87 (limit: 8696)
     Memory: 115.0M (peak: 169.7M)
```

I enabled the plugins inside the server and signed in as a guest to ensure that rabbitmq was working properly.

```
debjyotisarkar@sumonta-22341019:~/Desktop$ sudo rabbitmq-plugins enable rabbitmq
_management
Enabling plugins on node rabbit@sumonta-22341019:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
```




I installed pip and pika and created two files named send.py and receive.py

```
debjyotisarkar@sumonta-22341019:~/Desktop$ sudo apt-get install pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
debjyotisarkar@sumonta-22341019:~/Desktop$ sudo apt-get install python3-pika
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

**Task 1**

Firstly, I established the connection with the RabbitMQ server and created a queue and delivered the message “Hello world” through an exchange. Lastly, we closed the connection.

```python
#!/usr/bin/env python
import pika

connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='localhost'))
channel = connection.channel()

channel.queue_declare(queue='hello')

channel.basic_publish(exchange='', routing_key='hello', body='Hello World!')
print(" [x] Sent 'Hello World!'")
connection.close()
```

The receive.py file is used to get the messages sent by send.py, it also requires a queue. With the help of a callback function the program is able to show the message.

```python
#!/usr/bin/env python
import pika, sys, os

def main():
    connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
    channel = connection.channel()

    channel.queue_declare(queue='hello')

    def callback(ch, method, properties, body):
        print(f" [x] Received {body}")

    channel.basic_consume(queue='hello', on_message_callback=callback, auto_ack=True)

    print(' [*] Waiting for messages. To exit press CTRL+C')
    channel.start_consuming()

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

If we run the send.py program and later run receive.py we can see the output in the terminal.

```
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 send.py
 [x] Sent 'Hello World!'
debjyotisarkar@sumonta-22341019:~/Desktop$
```

```
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 receive.py
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received b'Hello World!'
```

**Task 2**

I opened two new files named new_task.py and worker.py

```
debjyotisarkar@sumonta-22341019:~/Desktop$ touch newtask.py
debjyotisarkar@sumonta-22341019:~/Desktop$ touch worker.py
debjyotisarkar@sumonta-22341019:~/Desktop$ gedit new_task.py
```

The new_task.py is built on top of the send.py file where the message allows for it to be typed or if it is not it will show “hello world” by default.

Open

new_task.py
~/Desktop

```python
#!/usr/bin/env python

import pika
import sys

connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='localhost'))

channel = connection.channel()


channel.queue_declare(queue='hello')


message = ' '.join(sys.argv[1:]) or "Hello World!"

channel.basic_publish(
    exchange='',
    routing_key='hello',
    body=message,
    )

print(f" [x] Sent {message}")
connection.close()
```

The worker.py is built on the receive.py file and it just has an edited callback function, the time.sleep method is used to simulate work and to check how the workers will respond in certain situations.

```python
#!/usr/bin/env python
import pika, sys, os
import time
def main():
    connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
    channel = connection.channel()

    channel.queue_declare(queue='hello')


    def callback(ch, method, properties, body):
        print(f" [x] Received {body.decode()}")
        time.sleep(body.count(b'.'))
        print(" [x] Done")

    channel.basic_consume(queue='hello', on_message_callback=callback, auto_ack=True)

    print(' [*] Waiting for messages. To exit press CTRL+C')
    channel.start_consuming()

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

If the new_task.py file is called with the messages we can see the two shells show the messages one by one; one shell shows first and third and the other one shows second and fourth.

```
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 new_task.py First Message
 [x] Sent First Message
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 new_task.py Second Message
 [x] Sent Second Message
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 new_task.py Third Message
 [x] Sent Third Message
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 new_task.py Fourth Message
 [x] Sent Fourth Message
```

```
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 worker.py
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received Second Message
 [x] Done
 [x] Received Fourth Message
 [x] Done
```

```
debjyotisarkar@sumonta-22341019:~/Desktop$ gedit new_task.py
debjyotisarkar@sumonta-22341019:~/Desktop$ gedit worker.py
debjyotisarkar@sumonta-22341019:~/Desktop$ python3 worker.py
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received First Message
 [x] Done
 [x] Received Third Message
 [x] Done
```