BRAC
UNIVERSITY

Inspiring Excellence

Name: NAIMUR ISLAM NAVID
Id:24141160
CSE484 Cloud Computing
Assignment 4

Task-1: Send and Recieve Hello World by using Message Broker

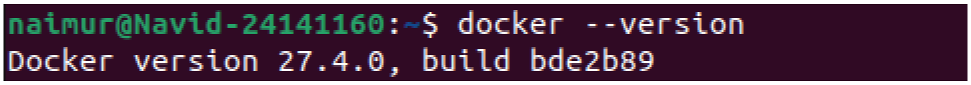
For this task Im going to use RabitMq and use it as my message broker.

Now lets first install RabitMq. For this im going to use Docker and pull RabbitMq image from dockerhub. You can install also Rabitmq in your os by following this guide: https://www.rabbitmq.com/docs/download

But theirs a reminder rabitmq stopped their older version releases. So you might find some issues. Instead use docker container.

First lets check our docker version

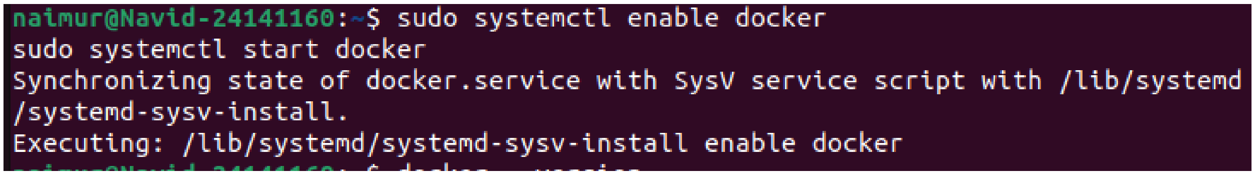
$docker --version



```
naimur@Navid-24141160:~$ docker --version
Docker version 27.4.0, build bde2b89
```

Now lets enable and start docker

$sudo systemctl enable docker
$sudo systemctl start docker

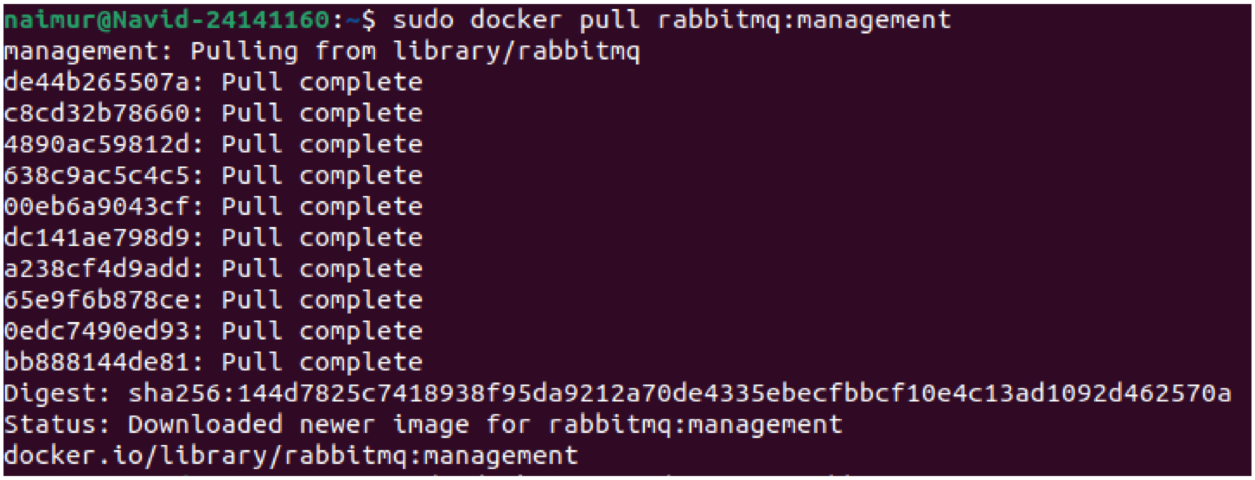


```
naimur@Navid-24141160:~$ sudo systemctl enable docker
sudo systemctl start docker
Synchronizing state of docker.service with SysV service script with /lib/systemd
/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
```

Now lets pull rabitmq image from dockerhub

$sudo docker pull rabbitmq:management

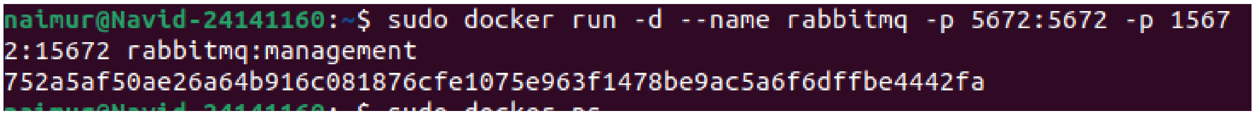
The management tag includes the RabbitMQ Management UI for easy interaction.



```
naimur@Navid-24141160:~$ sudo docker pull rabbitmq:management
management: Pulling from library/rabbitmq
de44b265507a: Pull complete
c8cd32b78660: Pull complete
4890ac59812d: Pull complete
638c9ac5c4c5: Pull complete
00eb6a9043cf: Pull complete
dc141ae798d9: Pull complete
a238cf4d9add: Pull complete
65e9f6b878ce: Pull complete
0edc7490ed93: Pull complete
bb888144de81: Pull complete
Digest: sha256:144d7825c7418938f95da9212a70de4335ebecfbbcf10e4c13ad1092d462570a
Status: Downloaded newer image for rabbitmq:management
docker.io/library/rabbitmq:management
```

Run the RabbitMQ Container

$sudo docker run -d --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:management



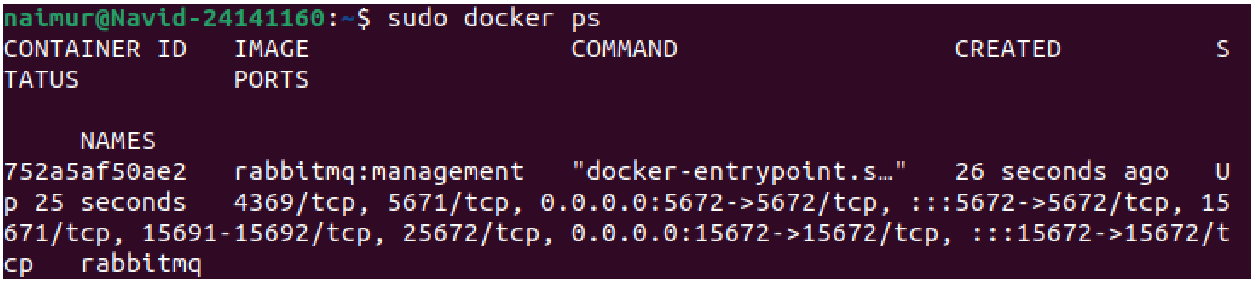
```
naimur@Navid-24141160:~$ sudo docker run -d --name rabbitmq -p 5672:5672 -p 1567
2:15672 rabbitmq:management
752a5af50ae26a64b916c081876cfe1075e963f1478be9ac5a6f6dffbe4442fa
```

here,
5672: Port for RabbitMQ messaging.
15672: Port for the RabbitMQ Management UI.

Check if the Container is Running:

$sudo docker ps

You should see the rabbitmq container running.

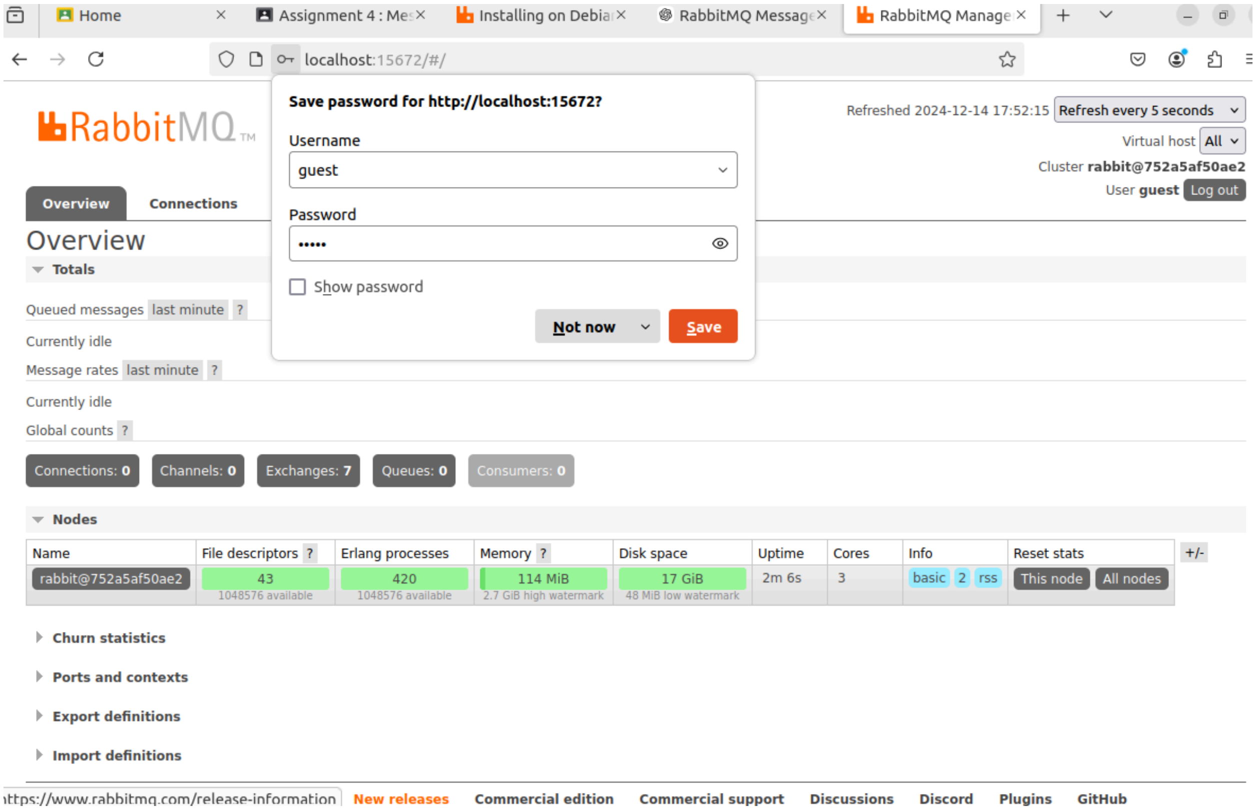


```
naimur@Navid-24141160:~$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          S
TATUS          PORTS

     NAMES
752a5af50ae2   rabbitmq:management   "docker-entrypoint.s…"   26 seconds ago   U
p 25 seconds   4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp, 15
671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp, :::15672->15672/t
cp   rabbitmq
```

Now we need to access RabitMq from our web

If you're running RabbitMQ on the same machine, use http://localhost:15672

Use user and pass for guest



Save password for http://localhost:15672?

Username
guest

Password
•••••

□ Show password

Refreshed 2024-12-14 17:52:15 Refresh every 5 seconds

Virtual host All

Cluster **rabbit@752a5af50ae2**

User **guest** Log out

Overview

Totals

Queued messages last minute ?

Currently idle

Message rates last minute ?

Currently idle

Global counts ?

Connections: **0** Channels: **0** Exchanges: **7** Queues: **0** Consumers: **0**

Nodes

Name	File descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Cores	Info	Reset stats
rabbit@752a5af50ae2	43 1048576 available	420 1048576 available	114 MiB 2.7 GiB high watermark	17 GiB 48 MiB low watermark	2m 6s	3	basic 2 rss	This node All nodes

And your RabitMQ is done installing and running

Now let's send Hello world

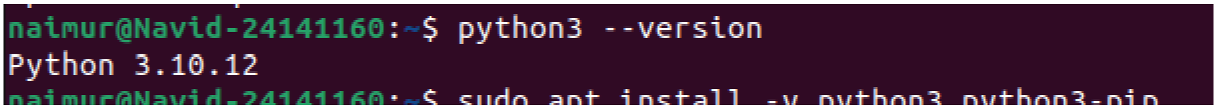
First we need to install python and pika library

For python check

$python3 --version

If not installed then use

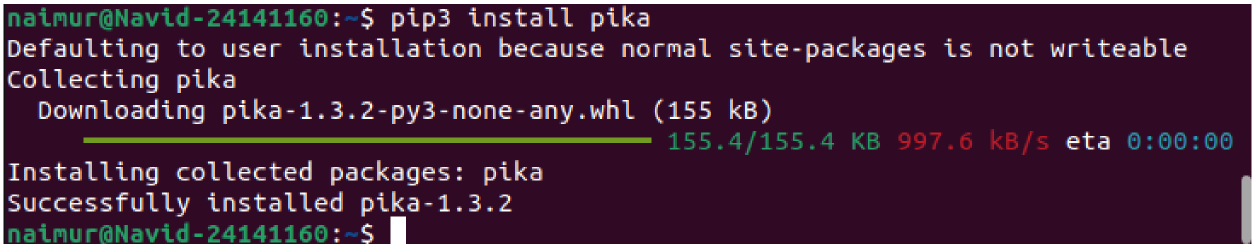
$sudo apt install -y python3 python3-pip



```
naimur@Navid-24141160:~$ python3 --version
Python 3.10.12
naimur@Navid-24141160:~$ sudo apt install -y python3 python3-pip
```

Install pika Library (RabbitMQ Client for Python)

$pip3 install pika



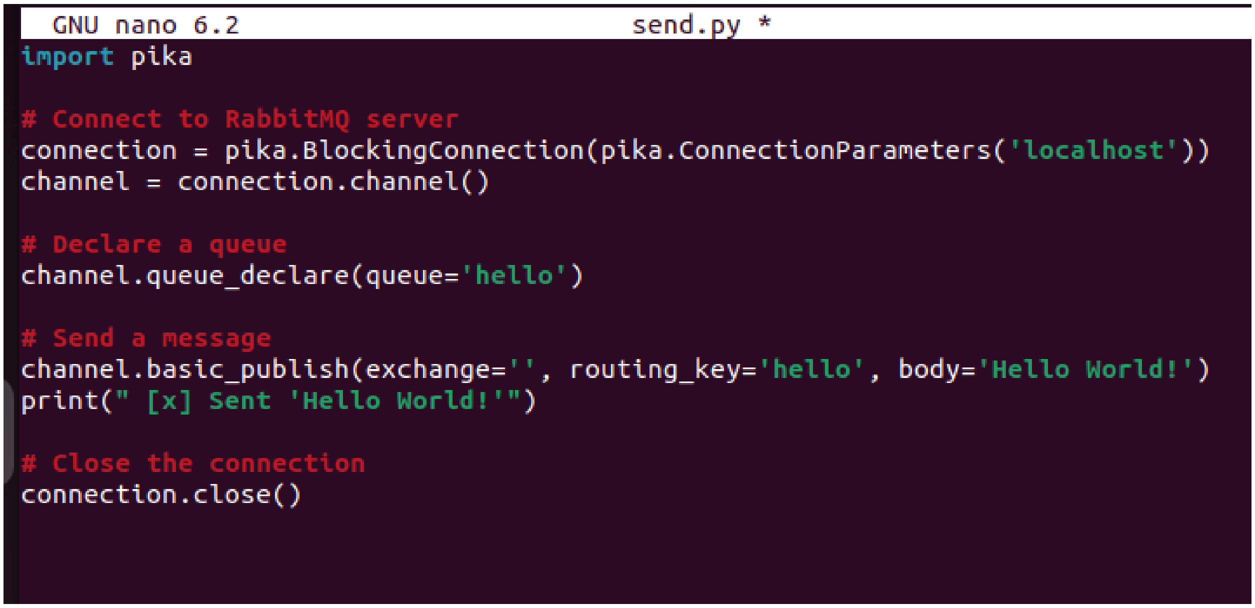
```
naimur@Navid-24141160:~$ pip3 install pika
Defaulting to user installation because normal site-packages is not writeable
Collecting pika
  Downloading pika-1.3.2-py3-none-any.whl (155 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 155.4/155.4 KB 997.6 kB/s eta 0:00:00
Installing collected packages: pika
Successfully installed pika-1.3.2
naimur@Navid-24141160:~$ 
```

Now we need to write the Sender Script

Create a file named send.py to send a "Hello World" message.

Open $nano send.py and write this code into it

```
import pika

# Connect to RabbitMQ server

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

channel = connection.channel()

# Declare a queue

channel.queue_declare(queue='hello')

# Send a message

channel.basic_publish(exchange='', routing_key='hello', body='Hello World!')

print(" [x] Sent 'Hello World!'")

# Close the connection

connection.close()
```



GNU nano 6.2 send.py *

```
import pika

# Connect to RabbitMQ server
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Declare a queue
channel.queue_declare(queue='hello')

# Send a message
channel.basic_publish(exchange='', routing_key='hello', body='Hello World!')
print(" [x] Sent 'Hello World!'")

# Close the connection
connection.close()
```

Save the file and run it

```
$python3 send.py
```

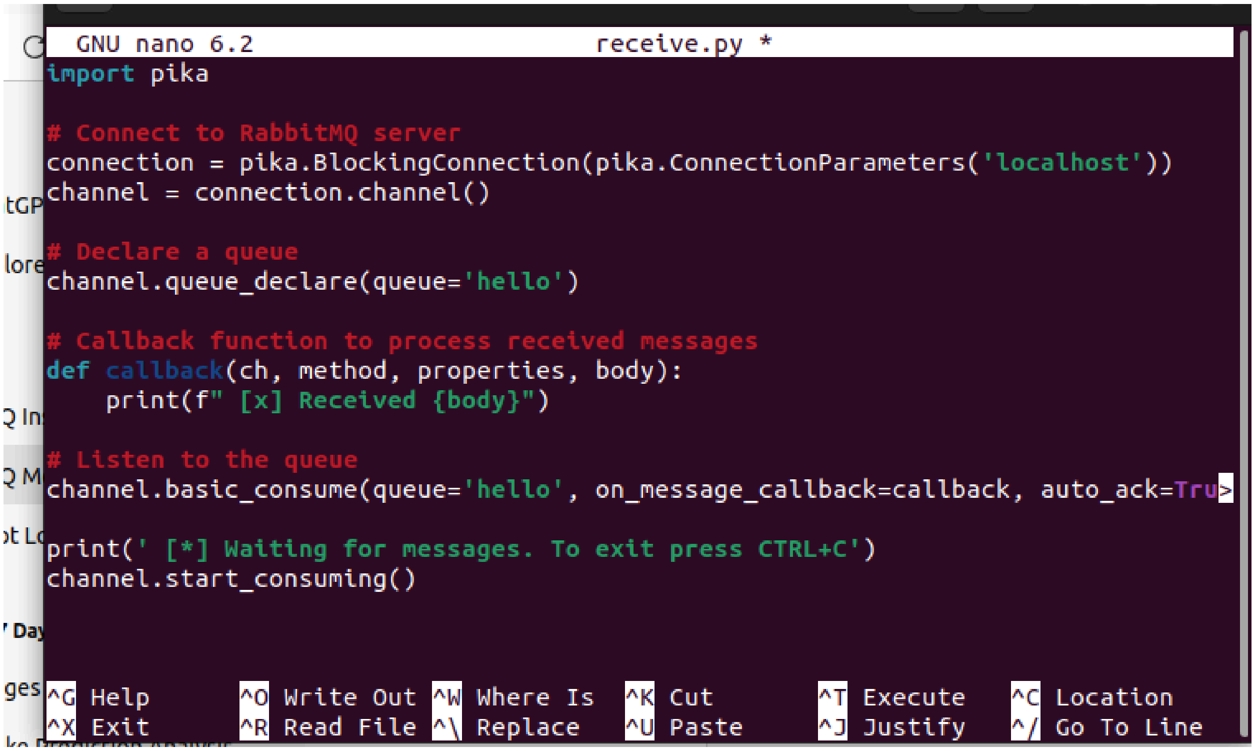
Now in another terminal write the Receiver Script

Create a file named receive.py to receive messages.

```
import pika

# Connect to RabbitMQ server

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

channel = connection.channel()



# Declare a queue

channel.queue_declare(queue='hello')



# Callback function to process received messages

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


# Listen to the queue

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


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

channel.start_consuming()
```

Save the file and run it

$python3 receive.py



```
  GNU nano 6.2                    receive.py *
import pika

# Connect to RabbitMQ server
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Declare a queue
channel.queue_declare(queue='hello')

# Callback function to process received messages
def callback(ch, method, properties, body):
    print(f" [x] Received {body}")

# Listen to the queue
channel.basic_consume(queue='hello', on_message_callback=callback, auto_ack=Tru>

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

Run the Receiver Script:
$python3 receive.py

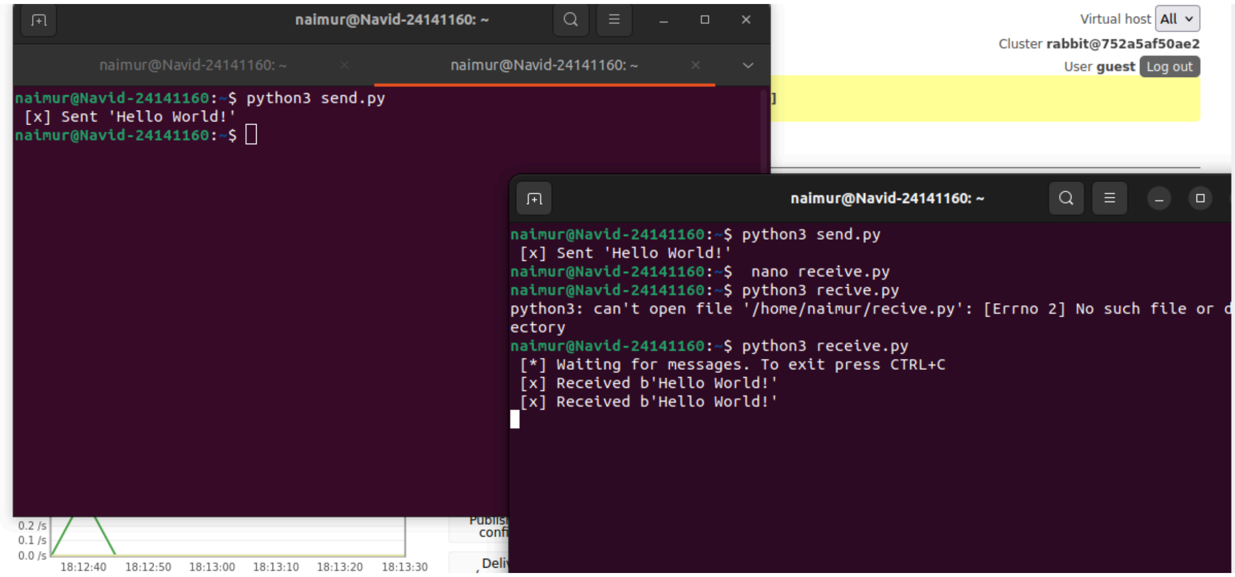
The receiver will start and wait for messages.

Run the Sender Script in another terminal:

$python3 send.py

You should see the receiver print:

[x] Received b'Hello World!'



```
naimur@Navid-24141160:~$ python3 send.py
 [x] Sent 'Hello World!'
naimur@Navid-24141160:~$ 
```

```
naimur@Navid-24141160:~$ python3 send.py
 [x] Sent 'Hello World!'
naimur@Navid-24141160:~$  nano receive.py
naimur@Navid-24141160:~$ python3 recive.py
python3: can't open file '/home/naimur/recive.py': [Errno 2] No such file or d
ectory
naimur@Navid-24141160:~$ python3 receive.py
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received b'Hello World!'
 [x] Received b'Hello World!'
```

And we are done with sending Hello world.

Task-2: "Work queues"- Distributing tasks among workers

In this task we will distribute tasks among multiple workers to process them concurrently.

Heres an overview of this task

Producer: Sends tasks (messages) to the queue.
Queue: Stores tasks until a worker processes them.
Workers: Multiple consumers that pull tasks from the queue for processing.

Lets start then

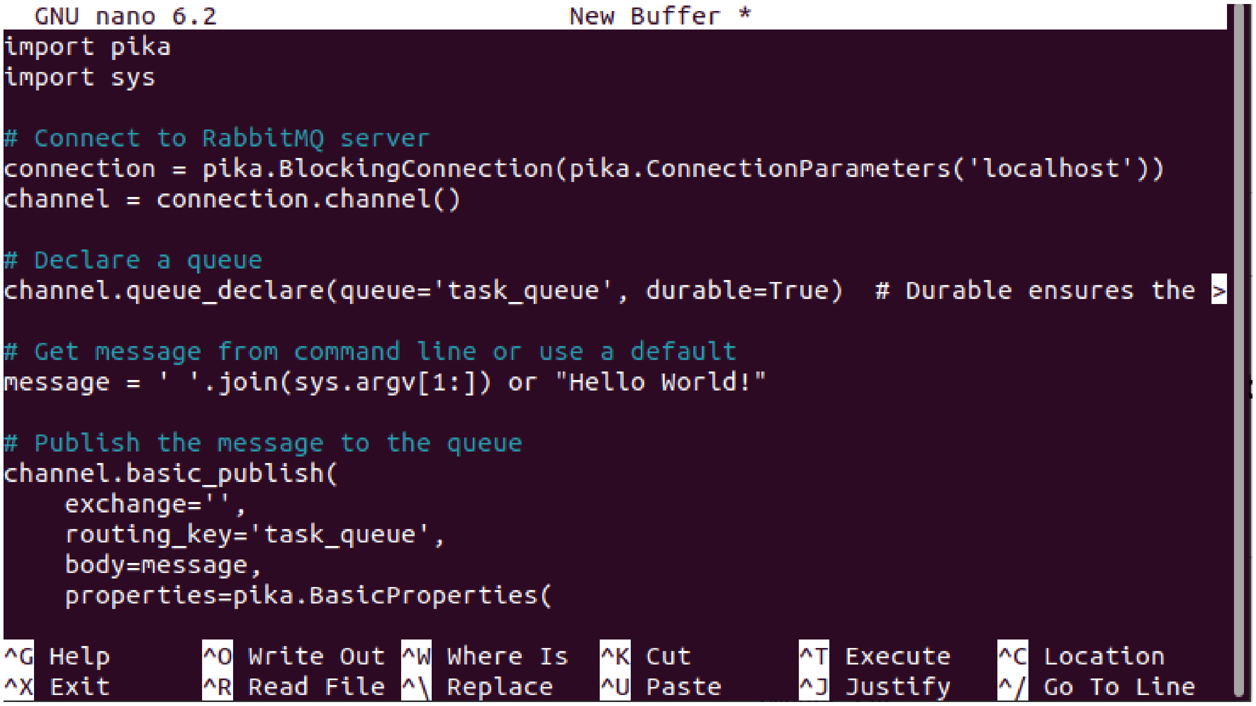
First we will open our editor (nano/vim)

```
$nano
```

```
naimur@Navid-24141160:~$ nano
```

Then write this code into it

```
import pika
import sys

# Connect to RabbitMQ server
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Declare a queue
channel.queue_declare(queue='task_queue', durable=True)  # Durable ensures the queue survives RabbitMQ restarts

# Get message from command line or use a default
message = ' '.join(sys.argv[1:]) or "Hello World!"

# Publish the message to the queue
channel.basic_publish(
    exchange='',
    routing_key='task_queue',
    body=message,
    properties=pika.BasicProperties(
        delivery_mode=pika.spec.PERSISTENT_DELIVERY_MODE  # Make message persistent
    )
)
print(f" [x] Sent {message}")

# Close the connection
connection.close()
```



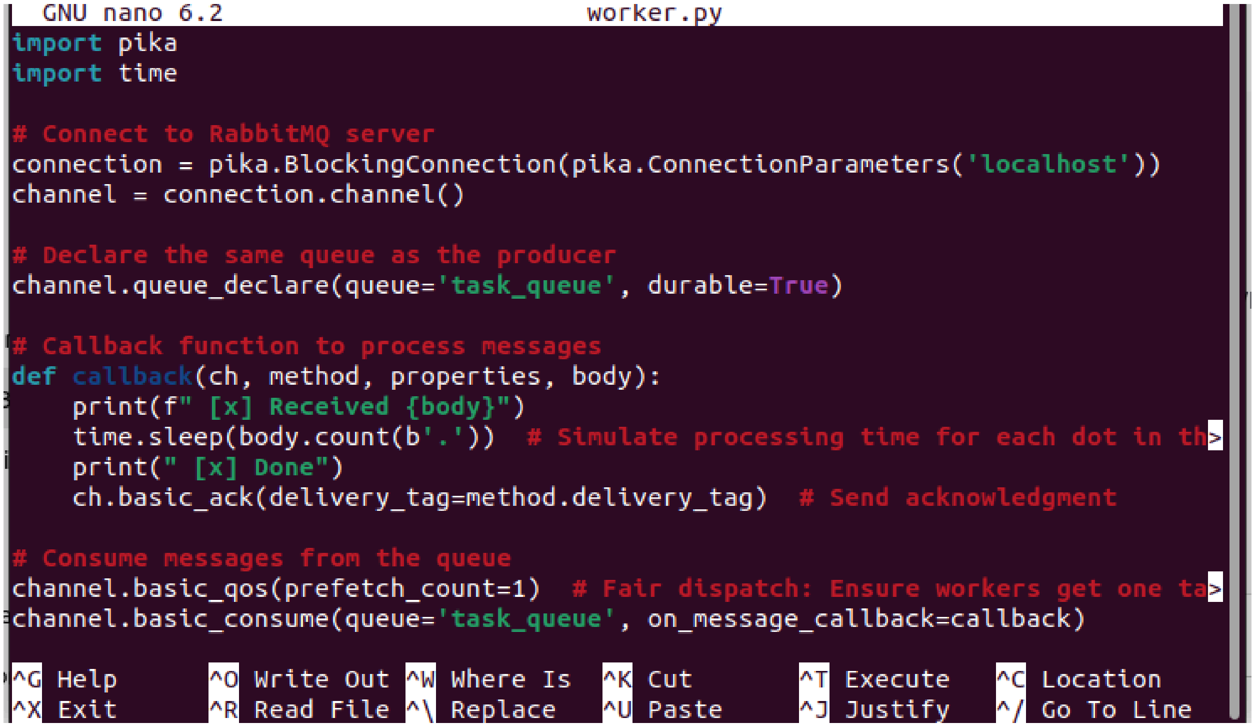
```
import pika
import sys

# Connect to RabbitMQ server
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Declare a queue
channel.queue_declare(queue='task_queue', durable=True)  # Durable ensures the >

# Get message from command line or use a default
message = ' '.join(sys.argv[1:]) or "Hello World!"

# Publish the message to the queue
channel.basic_publish(
    exchange='',
    routing_key='task_queue',
    body=message,
    properties=pika.BasicProperties(
```

Save this file as new_task.py. This was the producer part.

Now again write this code in nano editor and save it as worker.py

```
import pika
import time

# Connect to RabbitMQ server
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Declare the same queue as the producer
channel.queue_declare(queue='task_queue', durable=True)

# Callback function to process messages
def callback(ch, method, properties, body):
    print(f" [x] Received {body}")
    time.sleep(body.count(b'.'))  # Simulate processing time for each dot in the message
    print(" [x] Done")
    ch.basic_ack(delivery_tag=method.delivery_tag)  # Send acknowledgment

# Consume messages from the queue
```

```
channel.basic_qos(prefetch_count=1)  # Fair dispatch: Ensure workers get one task at a time
channel.basic_consume(queue='task_queue', on_message_callback=callback)

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



```
  GNU nano 6.2                          worker.py
import pika
import time

# Connect to RabbitMQ server
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Declare the same queue as the producer
channel.queue_declare(queue='task_queue', durable=True)

# Callback function to process messages
def callback(ch, method, properties, body):
    print(f" [x] Received {body}")
    time.sleep(body.count(b'.'))  # Simulate processing time for each dot in th>
    print(" [x] Done")
    ch.basic_ack(delivery_tag=method.delivery_tag)  # Send acknowledgment

# Consume messages from the queue
channel.basic_qos(prefetch_count=1)  # Fair dispatch: Ensure workers get one ta>
channel.basic_consume(queue='task_queue', on_message_callback=callback)
```

Save it

Run the Producer and Workers

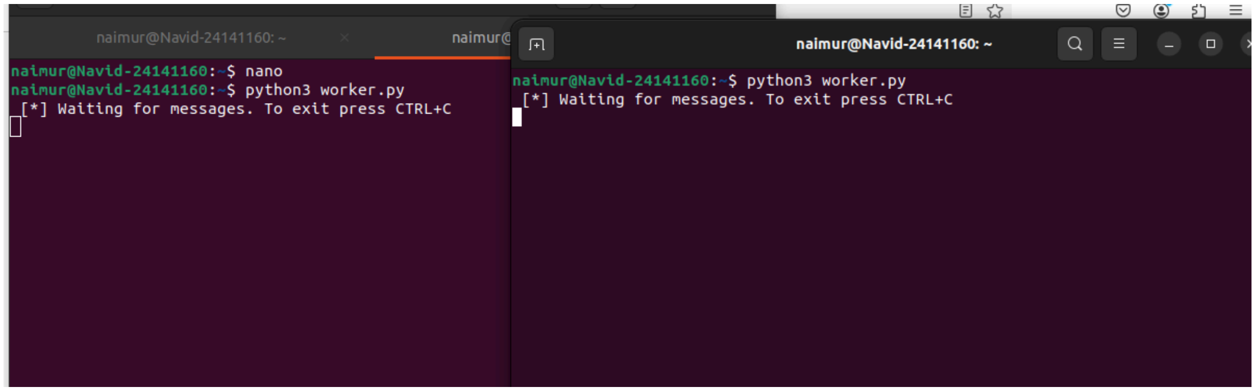
Start Workers: Open two terminals and run the worker script in each:

```
$python3 worker.py
```

Send Tasks: Use the producer script to send tasks with varying workloads (e.g., using dots to indicate workload):

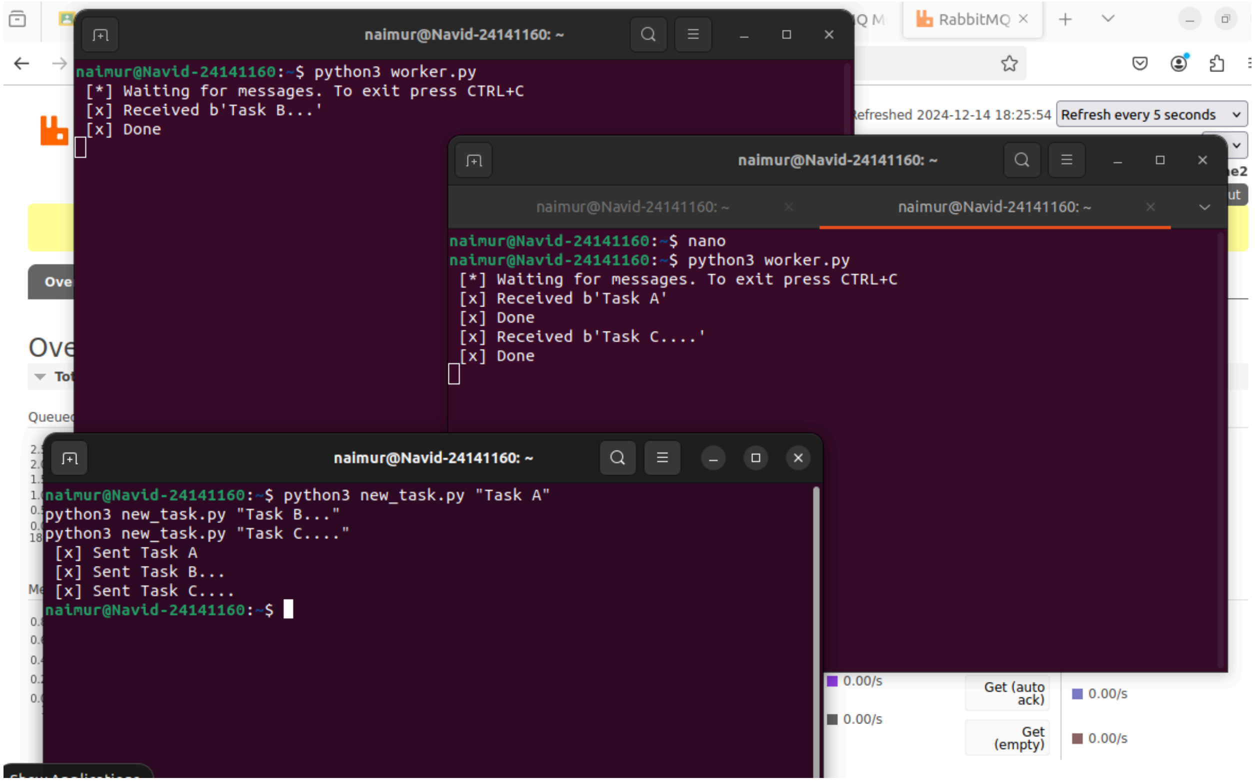
```
$python3 new_task.py "Task A"
python3 new_task.py "Task B..."
python3 new_task.py "Task C...."
```

Each dot (.) in the message simulates one second of processing time.



```
naimur@Navid-24141160:~$ nano
naimur@Navid-24141160:~$ python3 worker.py
 [*] Waiting for messages. To exit press CTRL+C
```

```
naimur@Navid-24141160:~$ python3 worker.py
 [*] Waiting for messages. To exit press CTRL+C
```



```
naimur@Navid-24141160:~$ python3 worker.py
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received b'Task B...'
 [x] Done
```

```
naimur@Navid-24141160:~$ nano
naimur@Navid-24141160:~$ python3 worker.py
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received b'Task A'
 [x] Done
 [x] Received b'Task C....'
 [x] Done
```

```
naimur@Navid-24141160:~$ python3 new_task.py "Task A"
python3 new_task.py "Task B..."
python3 new_task.py "Task C...."
 [x] Sent Task A
 [x] Sent Task B...
 [x] Sent Task C....
naimur@Navid-24141160:~$
```

Here we can see that the work was sent to one terminal and received by the other two terminals. The work gets queued by workers. We can see that task A was taken at first by one terminal and then task B by another terminal and since I didn't open another for task C it came back to the first terminal and did the work. thats how we did the work distribution by using the message broker RabitMQ.