



Assignment 4: Message Broker

Installing Rabbitmq:

At first we need to update the system by running:

```
sudo apt update && upgrade
```

Installing Erlang:

RabbitMQ requires Erlang. So, we need to install the erlang first.

To do that, we can run the following commands:

```
wget -O- https://packages.erlang-solutions.com/ubuntu/erlang_solutions.asc | sudo gpg  
--dearmor -o /usr/share/keyrings/erlang-solutions-archive-keyring.gpg  
echo "deb [signed-by=/usr/share/keyrings/erlang-solutions-archive-keyring.gpg] http  
s://packages.erlang-solutions.com/ubuntu $(lsb_release -cs) contrib" | sudo tee /etc/a  
pt/sources.list.d/erlang-solutions.list  
sudo apt install erlang
```

```
sabid@mahmud-22301172: ~  
sabid@mahmud-22301172:~$ wget -O- https://packages.erlang-solutions.com/ubuntu/erlang_solutions.asc | sudo gpg --dearmor -o /usr/share/keyrings/erlang-solutions-archive-keyring.gpg  
echo "deb [signed-by=/usr/share/keyrings/erlang-solutions-archive-keyring.gpg] https://packages.erlang-solutions.com/ubuntu $(lsb_release -cs) contrib" | sudo tee /etc/apt/sources.list.d/erlang-solutions.list  
--2023-12-01 20:14:26-- https://packages.erlang-solutions.com/ubuntu/erlang_solutions.asc  
Resolving packages.erlang-solutions.com (packages.erlang-solutions.com)... [sudo] password for sabid: 3.160.196.114, 3.160.196.25, 3.160.196.5, ...  
Connecting to packages.erlang-solutions.com (packages.erlang-solutions.com)|3.160.196.114|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 3133 (3.1K) [text/plain]  
Saving to: 'STDOUT'  
  
- 100%[=====] 3.06K --.-KB/s in 0s  
2023-12-01 20:14:29 (873 MB/s) - written to stdout [3133/3133]  
  
Sorry, try again.  
[sudo] password for sabid:  
deb [signed-by=/usr/share/keyrings/erlang-solutions-archive-keyring.gpg] https://packages.erlang-solutions.com/ubuntu jammy contrib  
sabid@mahmud-22301172:~$ sudo apt install erlang  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  ca-certificates-java default-jre-headless erlang-asn1 erlang-base erlang-common-test erlang-crypto erlang-debugger erlang-dev erlang-dialyzer erlang-diameter erlang-edoc erlang-eldap erlang-erl-docgen erlang-et erlang-eunit erlang-examples erlang-ftp erlang-inets erlang-jinterface erlang-megaco erlang-mnesia erlang-mode erlang-observer erlang-odbc erlang-os-mon erlang-parsetools erlang-public-key erlang-reltool erlang-runtime-tools erlang-snmp erlang-src erlang-ssh erlang-ssl erlang-syntax-tools erlang-tftp erlang-tools erlang-wx erlang-xmerl java-common libjs-jquery-metadata libjs-jquery-tablesorter libodbc2 libsctp1 libwxbase3.0-0v5 libwxgtk-webview3.0-gtk3-0v5 libwxgtk3.0-gtk3-0v5 openjdk-11-jre-headless  
Suggested packages:  
  default-jre erlang-manpages erlang-doc xsltproc fop odbc-postgresql tdsodbc lksctp-tools fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei
```

Installing RabbitMQ:

In this step I will install rabbitmq-server in my machine. To do that, I have to run this command on terminal.

```
sudo apt update  
wget -O- https://github.com/rabbitmq/signing-keys/releases/download/2.0/rabbitmq-release-signing-key.asc | sudo gpg --dearmor -o /usr/share/keyrings/rabbitmq-archive-keyring.gpg  
echo "deb [signed-by=/usr/share/keyrings/rabbitmq-archive-keyring.gpg] https://packagecloud.io/rabbitmq/rabbitmq-server/ubuntu/ $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/rabbitmq.list  
sudo apt install rabbitmq-server
```

```
sabid@mahmud-22301172:~$ # Now insatlling RabbitMQ
sabid@mahmud-22301172:~$ wget -O- https://github.com/rabbitmq/signing-keys/releases/download/2.0/rabbitmq-release-signing-key.asc
| sudo gpg --dearmor -o /usr/share/keyrings/rabbitmq-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/rabbitmq-archive-keyring.gpg] https://packagecloud.io/rabbitmq/rabbitmq-server/ubuntu/ $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/rabbitmq.list
sudo apt install rabbitmq-server
--2023-12-01 20:21:53-- https://github.com/rabbitmq/signing-keys/releases/download/2.0/rabbitmq-release-signing-key.asc
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/151086393/5f9f3800-c591-11e8-9fb0-451ca602c7dd?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20231201%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20231201T142039Z&X-Amz-Expires=300&X-Amz-Signature=4ab6d8ef43922db1923612eb3fe2084404050ab976bd3f8ff7cbf9fdc57906b3&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=151086393&response-content-disposition=attachment%3B%20filename%3Drabbitmq-release-signing-key.asc&response-content-type=application%2Foctet-stream [following]
--2023-12-01 20:21:53-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/151086393/5f9f3800-c591-11e8-9fb0-451ca602c7dd?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20231201%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20231201T142039Z&X-Amz-Expires=300&X-Amz-Signature=4ab6d8ef43922db1923612eb3fe2084404050ab976bd3f8ff7cbf9fdc57906b3&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=151086393&response-content-disposition=attachment%3B%20filename%3Drabbitmq-release-signing-key.asc&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.111.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3187 (3.1K) [application/octet-stream]
Saving to: 'STDOUT'

-
100%[=====] 3.11K --.-KB/s in 0s

2023-12-01 20:21:54 (42.9 MB/s) - written to stdout [3187/3187]

deb [signed-by=/usr/share/keyrings/rabbitmq-archive-keyring.gpg] https://packagecloud.io/rabbitmq/rabbitmq-server/ubuntu/ jammy main
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  socat
The following NEW packages will be installed:
```

Enabling rabbitmq:

once installed, by running this command, I can start and enable rabbitmq-server in my machine.

```
sudo systemctl start rabbitmq-server
sudo systemctl enable rabbitmq-server
```

Checking the status:

We can check the status of the rabbitmq-server by running this command.

```
sudo systemctl status rabbitmq-server
```

```
sabid@mahmud-22301172: ~
sabid@mahmud-22301172:~$ # Starting and enabling the Rabbitmq-server
sabid@mahmud-22301172:~$ sudo systemctl start rabbitmq-server
sudo systemctl enable rabbitmq-server
Synchronizing state of rabbitmq-server.service with SysV service script with /lib/sy
stemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable rabbitmq-server
sabid@mahmud-22301172:~$ # now, checking the status of rabbitmq
sabid@mahmud-22301172:~$ sudo systemctl status rabbitmq-server
● rabbitmq-server.service - RabbitMQ Messaging Server
   Loaded: loaded (/lib/systemd/system/rabbitmq-server.service; enabled; vendor p
   Active: active (running) since Fri 2023-12-01 20:22:10 +06; 2min 29s ago
   Main PID: 13444 (beam.smp)
     Tasks: 43 (limit: 8650)
    Memory: 126.9M
       CPU: 5.375s
    CGroup: /system.slice/rabbitmq-server.service
            └─13444 /usr/lib/erlang/erts-12.2.1/bin/beam.smp -W w -MBas ageffcbf ->
              └─13455 erl_child_setup 65536
                └─13547 inet_gethost 4
                  └─13548 inet_gethost 4

Dec 01 20:22:05 mahmud-22301172 systemd[1]: Starting RabbitMQ Messaging Server...
Dec 01 20:22:10 mahmud-22301172 systemd[1]: Started RabbitMQ Messaging Server.
lines 1-15/15 (END)
```

Install Pika

Pika requires python and pip. So, If we do not have python and pip installed, we need to install it first. In this case, I am using python3.

So, let's install python3 and pip3.

```
sudo apt install python3 python3-pip
```

Then, we can use the `pip` to install pika.

```
pip3 install pika
```

```
sabid@mahmud-22301172: ~  
sabid@mahmud-22301172:~$ sudo apt-get install python3-dev python3-pip  
[sudo] password for sabid:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
python3-dev is already the newest version (3.10.6-1~22.04).  
python3-dev set to manually installed.  
python3-pip is already the newest version (22.0.2+dfsg-1ubuntu0.4).  
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.  
sabid@mahmud-22301172:~$ pip 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 1.2 MB/s eta 0:00:00  
Installing collected packages: pika  
Successfully installed pika-1.3.2  
sabid@mahmud-22301172:~$
```

Let's verify that pika is installed

```
pip3 show pika
```

```
sabid@mahmud-22301172: ~  
sabid@mahmud-22301172:~$ pip3 show pika  
Name: pika  
Version: 1.3.2  
Summary: Pika Python AMQP Client Library  
Home-page:  
Author:  
Author-email:  
License: BSD-3-Clause  
Location: /home/sabid/.local/lib/python3.10/site-packages  
Requires:  
Required-by:  
sabid@mahmud-22301172:~$
```

TASK1: Hello World

Creating a producer and a consumer script using pika:

Create a file `producer.py` using a text editor. In this case I am using my default text editor neovim.

```
nvim producer.py
```

Then write this code in the file:

```
import pika

# Connect to RabbitMQ server (assuming it's running on localhost)
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

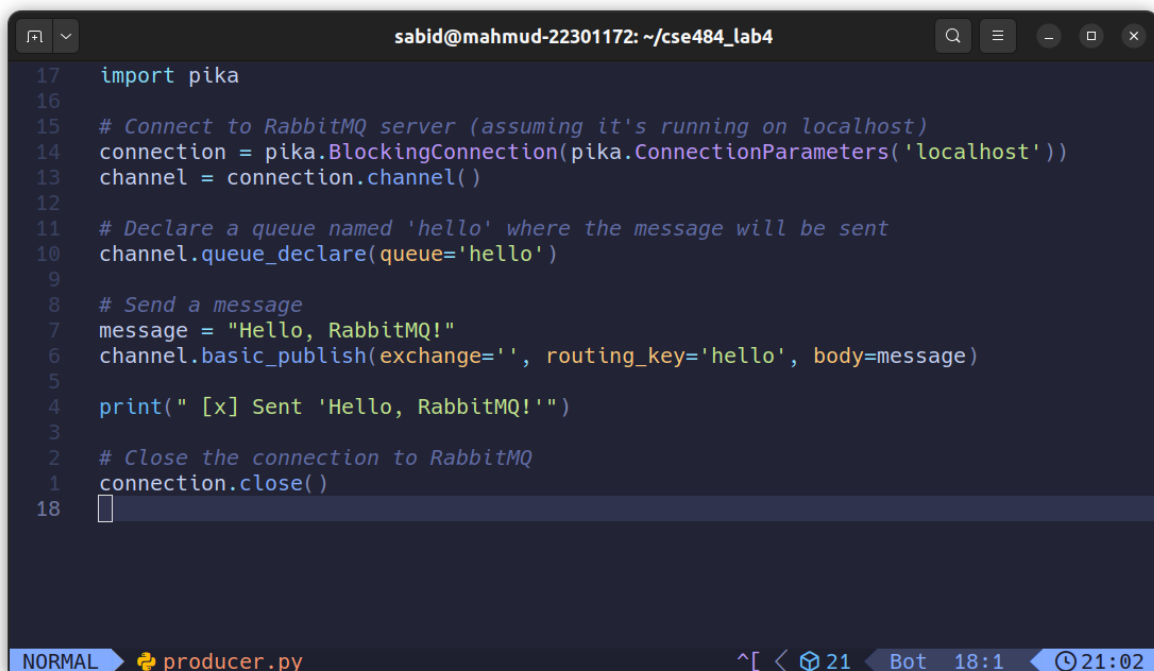
# Declare a queue named 'hello' where the message will be sent
channel.queue_declare(queue='hello')

# Send a message
message = "Hello, RabbitMQ!"
channel.basic_publish(exchange='', routing_key='hello', body=message)

print(" [x] Sent 'Hello, RabbitMQ!'")

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

save the file and exit.

A screenshot of a terminal window with a dark background. The window title is 'sabid@mahmud-22301172: ~/cse484_lab4'. The code is written in a light blue font. Line numbers 1 through 18 are visible on the left. The code is identical to the one in the previous block. At the bottom, there is a status bar with 'NORMAL' in a blue box, a Python icon, 'producer.py', and some navigation icons on the right.

```
17 import pika
16
15 # Connect to RabbitMQ server (assuming it's running on localhost)
14 connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
13 channel = connection.channel()
12
11 # Declare a queue named 'hello' where the message will be sent
10 channel.queue_declare(queue='hello')
9
8 # Send a message
7 message = "Hello, RabbitMQ!"
6 channel.basic_publish(exchange='', routing_key='hello', body=message)
5
4 print(" [x] Sent 'Hello, RabbitMQ!'")
3
2 # Close the connection to RabbitMQ
1 connection.close()
18
```

NORMAL producer.py 21 Bot 18:1 21:02

Now, let us create another file `consumer.py` using the same method:

```
nvim consumer.py
```

```

import pika

# Connect to RabbitMQ server (assuming it's running on localhost)
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Declare the same queue as in the producer
channel.queue_declare(queue='hello')

# Define a callback function to handle received messages
def callback(ch, method, properties, body):
    print(" [x] Received %r" % body)

# Consume messages from the 'hello' queue with the callback function
channel.basic_consume(queue='hello', on_message_callback=callback, auto_ack=True)

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

```

```

17 import pika
16
15 # Connect to RabbitMQ server (assuming it's running on localhost)
14 connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
13 channel = connection.channel()
12
11 # Declare the same queue as in the producer
10 channel.queue_declare(queue='hello')
9
8 # Define a callback function to handle received messages
7 def callback(ch, method, properties, body):
6 |     print(" [x] Received %r" % body)
5
4 # Consume messages from the 'hello' queue with the callback function
3 channel.basic_consume(queue='hello', on_message_callback=callback, auto_ack=True)
2
1 print(' [*] Waiting for messages. To exit, press CTRL+C')
18 channel.start_consuming()

```

NORMAL consumer.py 21 Bot 18:25 21:06

Now let us run the `consumer.py` file

```
python3 consumer.py
```

Now, Run the `producer.py` file

```
python3 producer.py
```

If everything runs perfectly, the output should seem like this:



```
sabid@mahmud-22301172: ~/cse484_lab4
[I] ♦ cse484_lab4 >>> python3 consumer.py
[*] Waiting for messages. To exit, press CTRL+C
[x] Received b'Hello, RabbitMQ!'
[x] Received b'Hello, RabbitMQ!'

[I] ♦ cse484_lab4 >>> python3 producer.py
[x] Sent 'Hello, RabbitMQ!'
[I] ♦ cse484_lab4 >>> python3 producer.py
[x] Sent 'Hello, RabbitMQ!'
[I] ♦ cse484_lab4 >>> []
```

TAKS2: Work queues

To complete Task 2, "Work Queues," using RabbitMQ, we will create a task sender script and a worker script that processes tasks with varying complexities.

Create the Task Sender Script:

Lets write a python script `new_task.py` that will send messages simulating tasks with various processing time.

```
nvim new_task.py
```

Now write this code in the file.

```
import pika
import sys

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

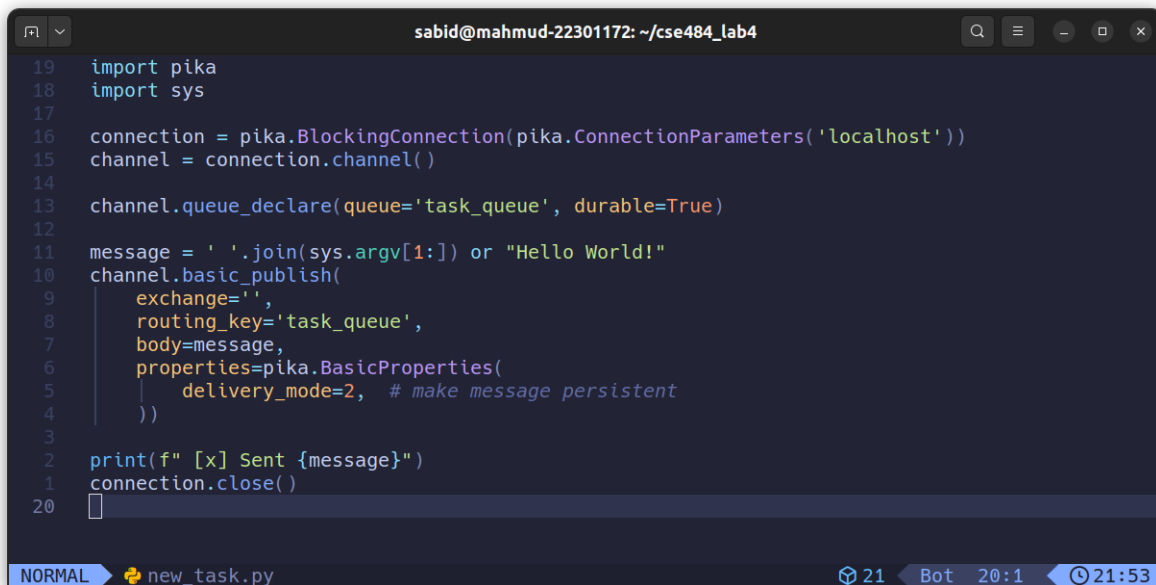
channel.queue_declare(queue='task_queue', durable=True)

message = ' '.join(sys.argv[1:]) or "Hello World!"
channel.basic_publish(
    exchange='',
    routing_key='task_queue',
    body=message,
    properties=pika.BasicProperties(
        delivery_mode=2, # make message persistent
    ))
```



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

Save and exit.



```
sabid@mahmud-22301172: ~/cse484_lab4
19 import pika
18 import sys
17
16 connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
15 channel = connection.channel()
14
13 channel.queue_declare(queue='task_queue', durable=True)
12
11 message = ' '.join(sys.argv[1:]) or "Hello World!"
10 channel.basic_publish(
9     exchange='',
8     routing_key='task_queue',
7     body=message,
6     properties=pika.BasicProperties(
5         delivery_mode=2, # make message persistent
4     ))
3
2 print(f" [x] Sent {message}")
1 connection.close()
20
```

Create the Worker Script

Let us write another python script `worker.py` which will pick up tasks from the queue and simulate the processing of the tasks.

```
nvim worker.py
```

Now let us write this scrip in the file.

```
import pika
import time

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

channel.queue_declare(queue='task_queue', durable=True)

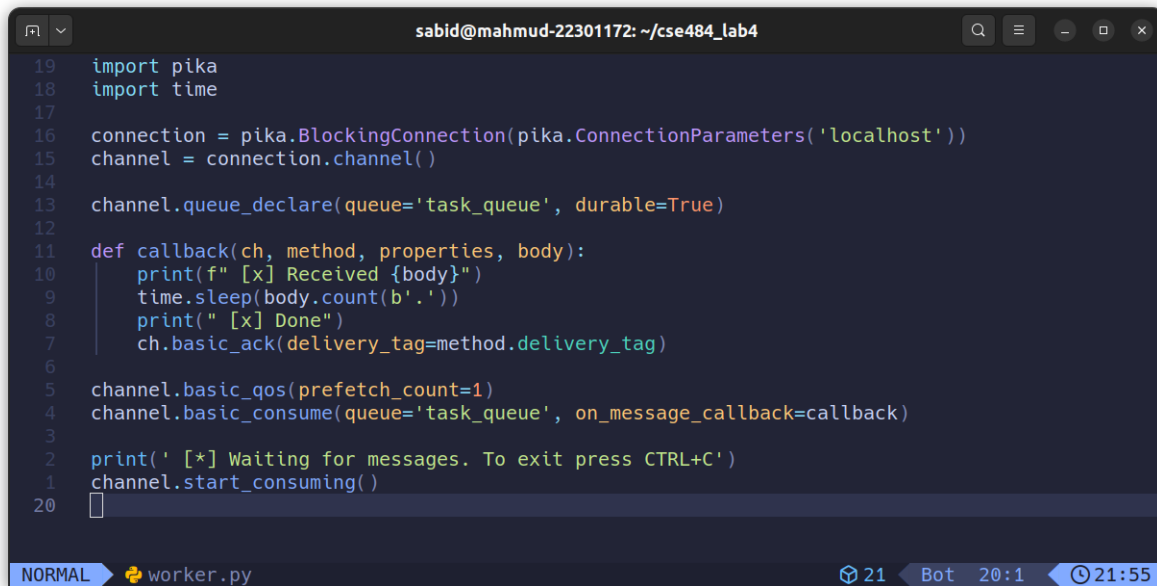
def callback(ch, method, properties, body):
    print(f" [x] Received {body}")
    time.sleep(body.count(b'.'))
    print(" [x] Done")
    ch.basic_ack(delivery_tag=method.delivery_tag)

channel.basic_qos(prefetch_count=1)
```

```
channel.basic_consume(queue='task_queue', on_message_callback=callback)

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

Let us save and exit.



```
sabid@mahmud-22301172: ~/cse484_lab4
19 import pika
18 import time
17
16 connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
15 channel = connection.channel()
14
13 channel.queue_declare(queue='task_queue', durable=True)
12
11 def callback(ch, method, properties, body):
10     print(f" [x] Received {body}")
9     time.sleep(body.count(b'.'))
8     print(" [x] Done")
7     ch.basic_ack(delivery_tag=method.delivery_tag)
6
5 channel.basic_qos(prefetch_count=1)
4 channel.basic_consume(queue='task_queue', on_message_callback=callback)
3
2 print(' [*] Waiting for messages. To exit press CTRL+C')
1 channel.start_consuming()
20
```

NORMAL worker.py 21 Bot 20:1 21:55

Run the `worker.py` and Observe:

Now, let us run the `worker.py` and the `new_task.py` in a separate terminal.

```
cse484_lab4:python3
[I] ♦ cse484_lab4 >>> python3 worker.py
[*] Waiting for messages. To exit press CTRL+C
[x] Received b'TaskA.'
[x] Done
[x] Received b'TaskD..'
[x] Done
[x] Received b'TaskE..'
[x] Done
[x] Received b'TaskXYZ..'
[x] Done
[]

cse484_lab4:python3
[I] ♦ cse484_lab4 >>> python3 worker.py
[*] Waiting for messages. To exit press CTRL+C
[x] Received b'TaskB..'
[x] Done
[x] Received b'TaskC..'
[x] Done
[x] Received b'TaskF..'
[x] Done
[]

cse484_lab4:fish
[I] ♦ ~ >>> cd cse484_lab4/
[I] ♦ cse484_lab4 >>> python3 new_task.py TaskA.
[x] Sent TaskA.
[I] ♦ cse484_lab4 >>> python3 new_task.py TaskB..
[x] Sent TaskB..
[I] ♦ cse484_lab4 >>> python3 new_task.py TaskD..
[x] Sent TaskD..
[I] ♦ cse484_lab4 >>> python3 new_task.py TaskC..
[x] Sent TaskC..
[I] ♦ cse484_lab4 >>> python3 new_task.py TaskE..
[x] Sent TaskE..
[I] ♦ cse484_lab4 >>> python3 new_task.py TaskF..
[x] Sent TaskF..
[I] ♦ cse484_lab4 >>> python3 new_task.py TaskXYZ..
[x] Sent TaskXYZ..
[I] ♦ cse484_lab4 >>>
```

My observation:

To illustrate RabbitMQ's work queue distribution, I set up three instances of `worker.py` in separate terminal windows, ordered from top to bottom based on their launch sequence. This arrangement aimed to showcase how RabbitMQ allocates tasks across different workers.

In a separate terminal acting as the task sender, I ran the `new_task.py` script using various commands to simulate tasks of differing lengths. For instance, commands like `python3 new_task.py TaskA.` and `python3 new_task.py TaskB..` created tasks labeled from TaskA to TaskF, with increasing numbers of dots indicating longer processing times.

By monitoring the worker terminals, I observed the task distribution. The first worker (top terminal) handled 'TaskA.' and 'TaskD..' and "TaskE..", the second worker (bottom terminal) processed 'TaskB..' and 'TaskC..' and TaskF.. This demonstrated RabbitMQ's ability to evenly distribute tasks among available workers, showcasing its load-balancing capabilities.