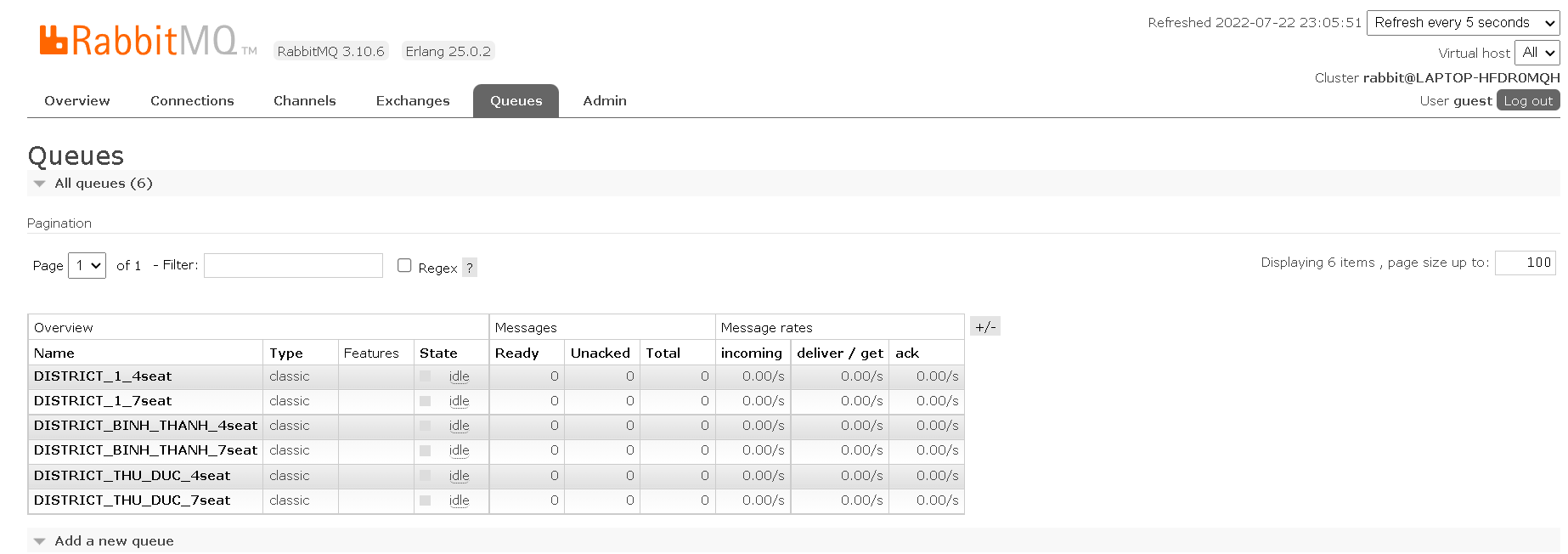
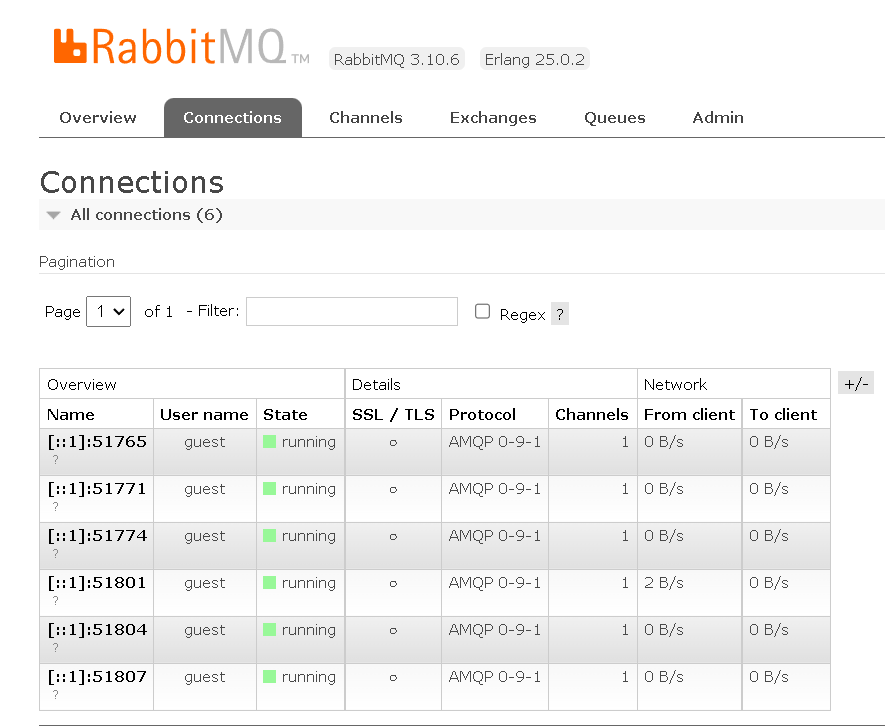
Taxi

At first step, we run files *taxi4\_binhthanh.py, taxi4\_d1.py, taxi4\_thuduc.py, taxi7\_binhthanh.py, taxi7\_d1.py, taxi7\_thuduc.py* to create queques on Rabbitmq website such as:

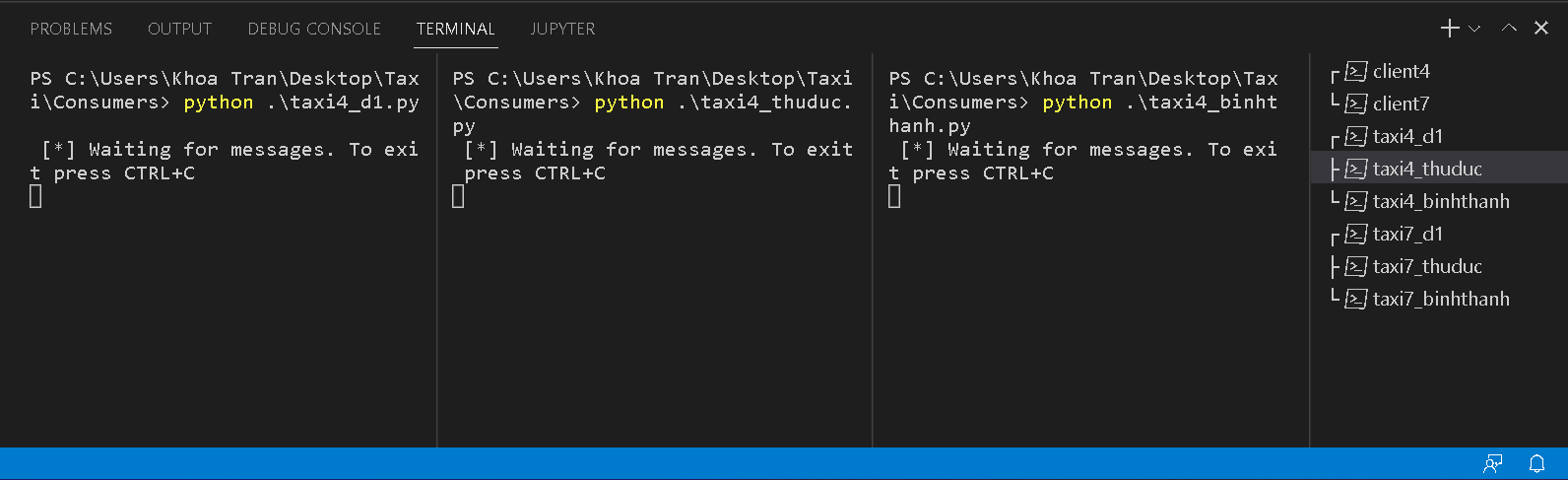
DISTRICT\_BINH\_THANH\_4seat, DISTRICT\_1\_4seat, DISTRICT\_THU\_DUC\_4seat, DISTRICT\_BINH\_THANH\_7seat, DISTRICT\_1\_7seat, DISTRICT\_THU\_DUC\_7seat.

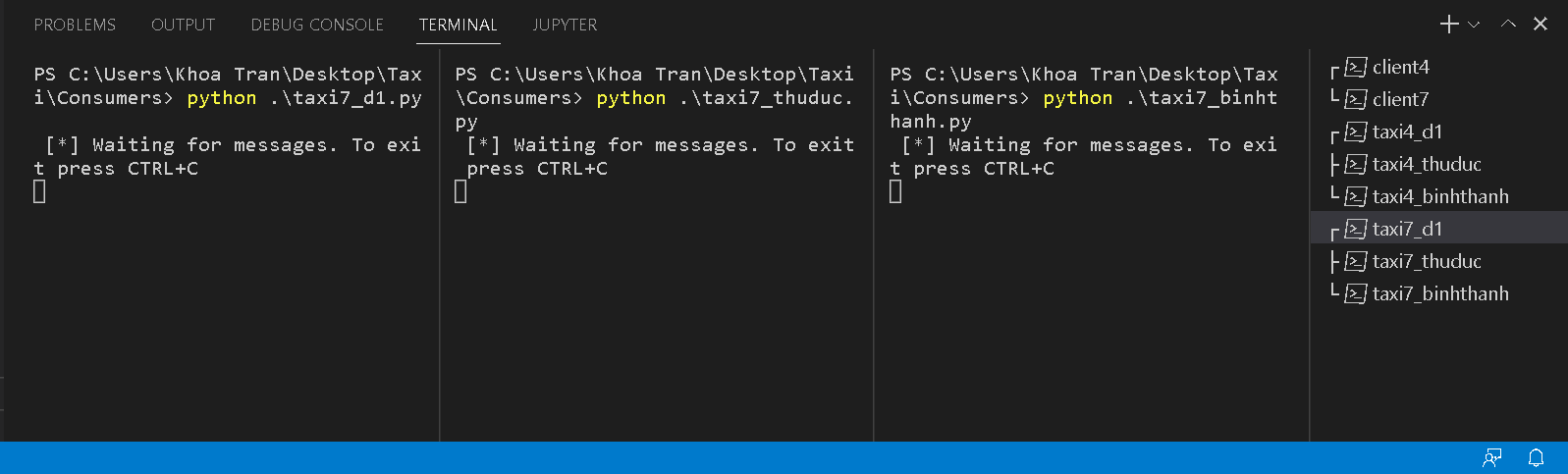


Next, we move to Connections. There are 6 connections corresponding to 6 python files are running.

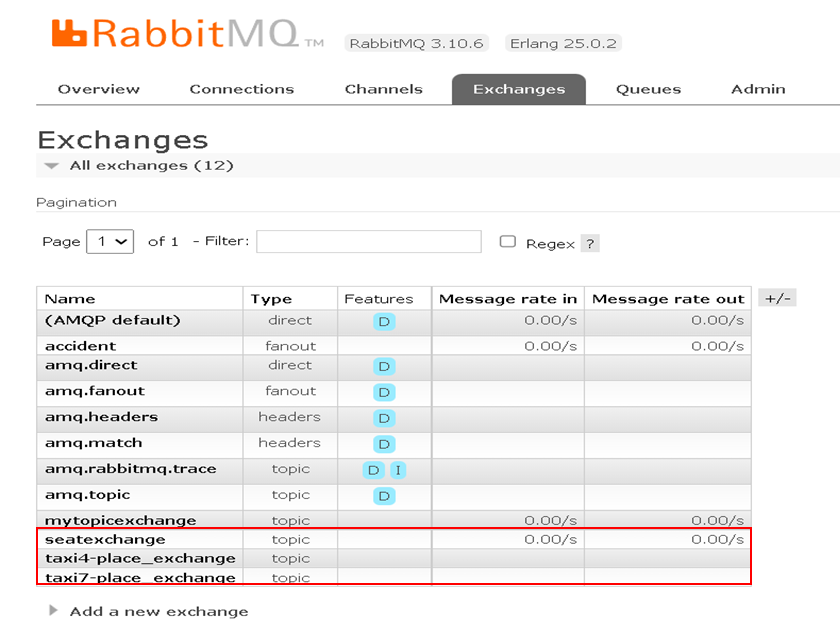


In Terminal, after we run python files:

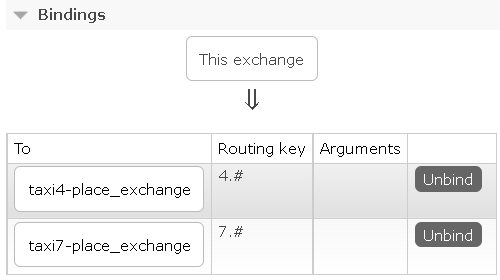




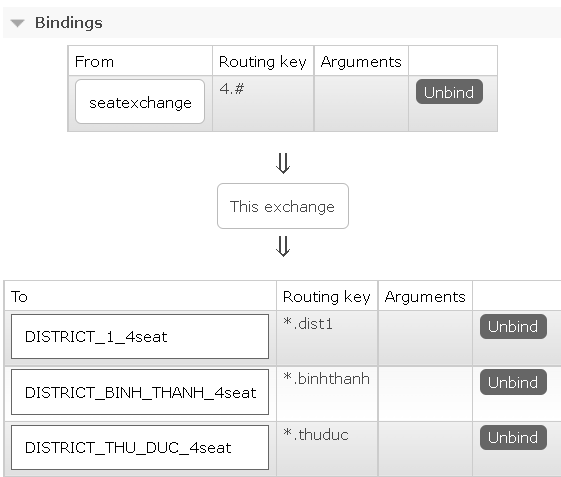
Second, we run files *client4.py, client7.py* to declare topic exchanges ‘seatexchange’, ‘taxi4-place\_exchange’ and ‘taxi7-place\_exchange’ on Rabbitmq website.



Inside ‘seatexchange’, we can see in Bindings. There are 2 exchange will be received message from ‘seatexchange’ with the routing keys which we declared before.

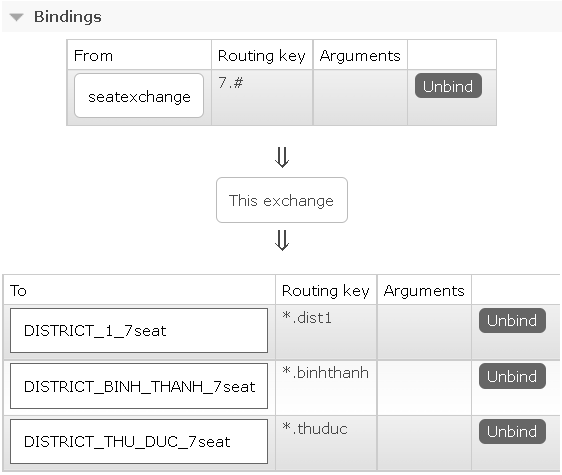


Inside ‘taxi4-place\_exchange’ and ‘taxi7-place\_exchange’



*taxi4-place\_exchange*

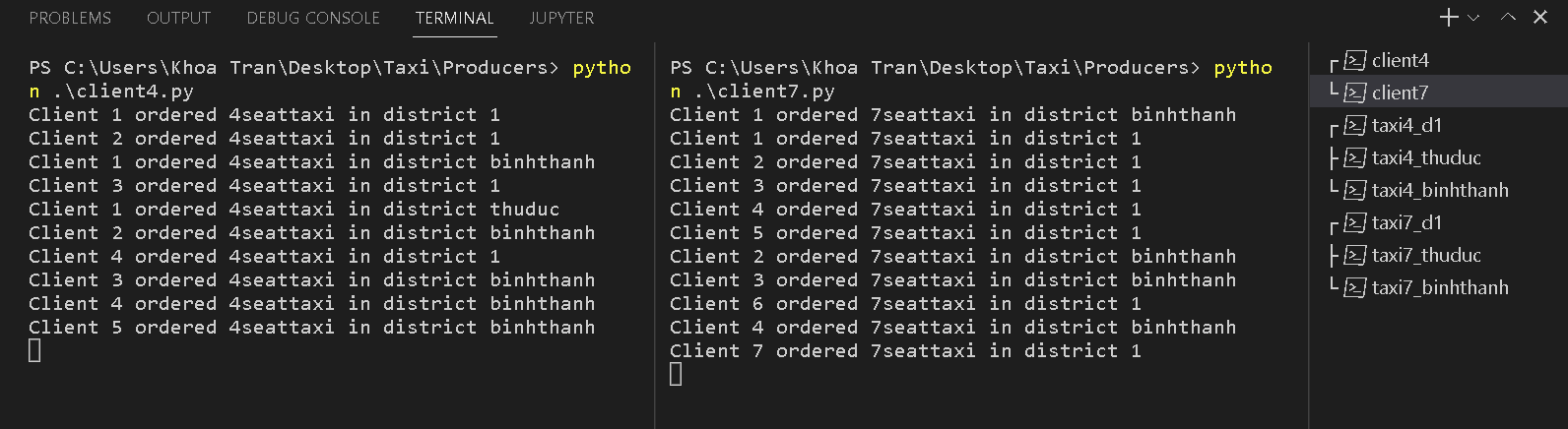
This exchange will receive message from ‘seatexchange’ and send it to 3 different queues with 3 relevant routing keys.

**

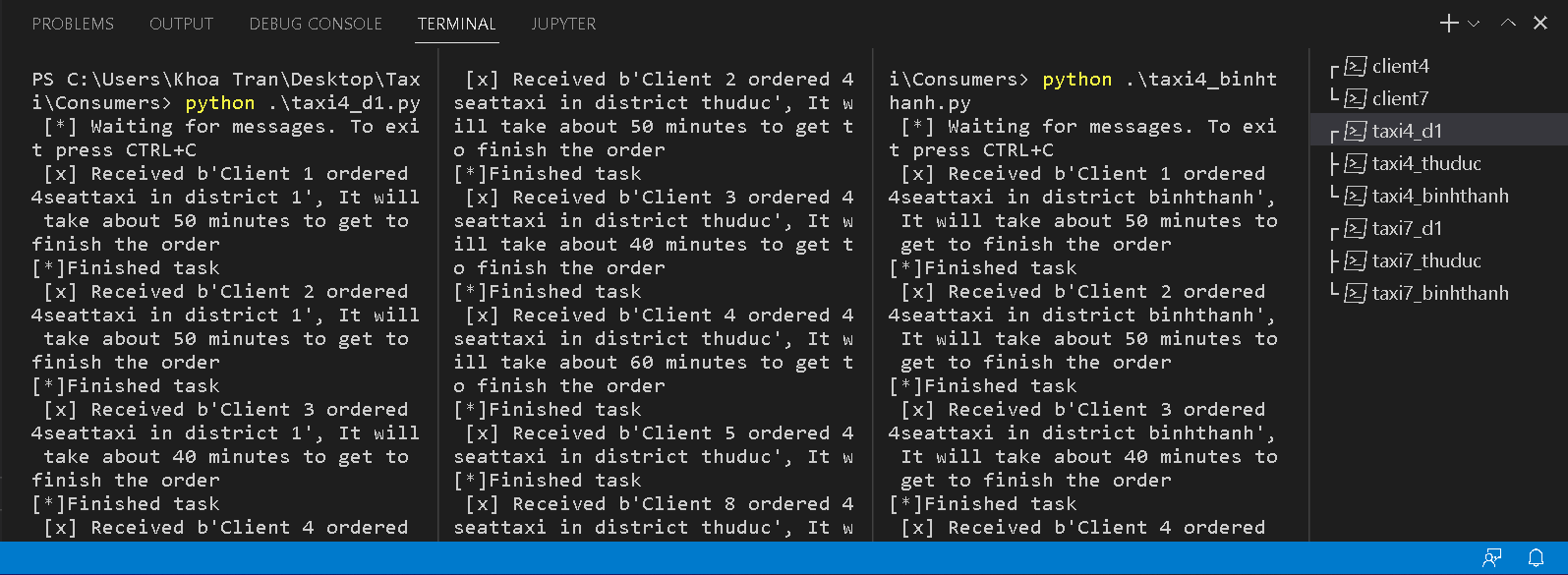
*taxi7-place\_exchange*

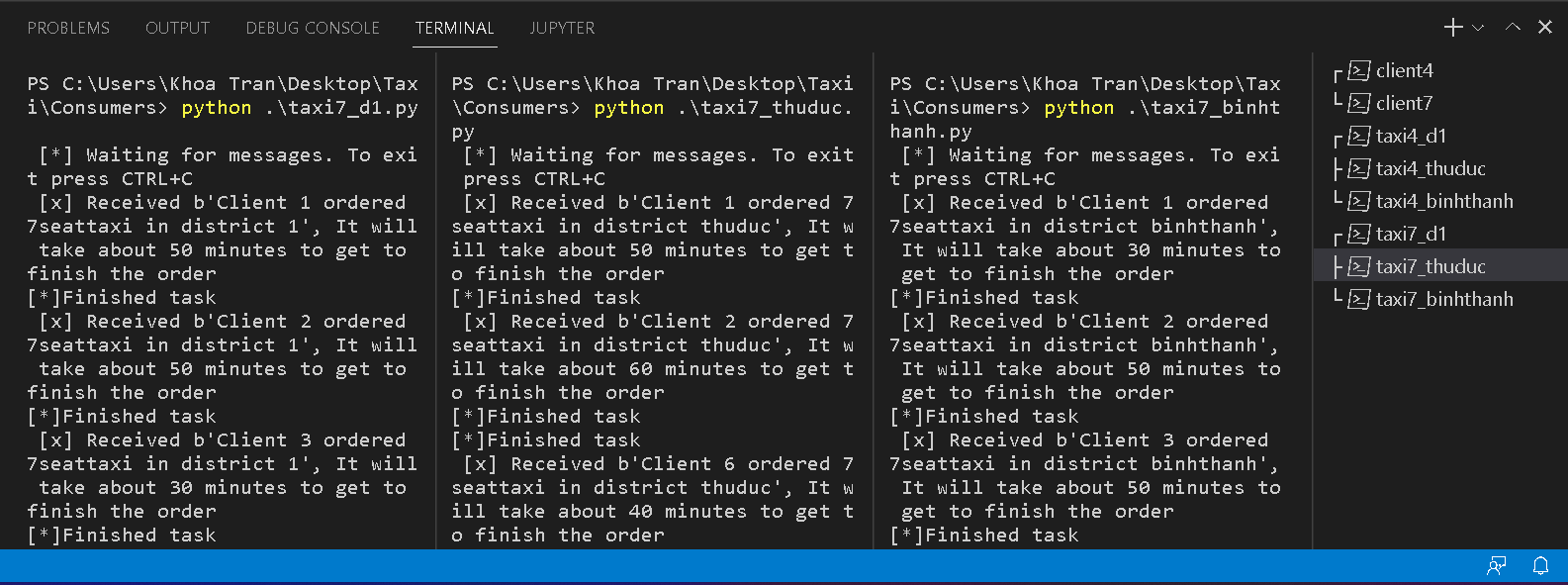
This exchange will also receive message from ‘seatexchange’ and send it to 3 different queues with 3 relevant routing keys.

In Terminal, after we run python files:

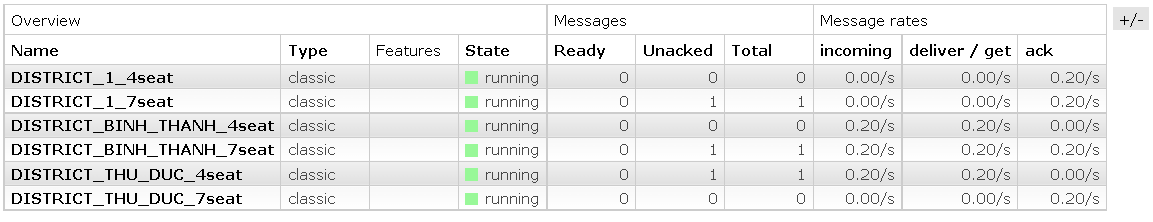


At the same time, all consumers receive the messages but for each message, it will take an unpridictable period of time to process.

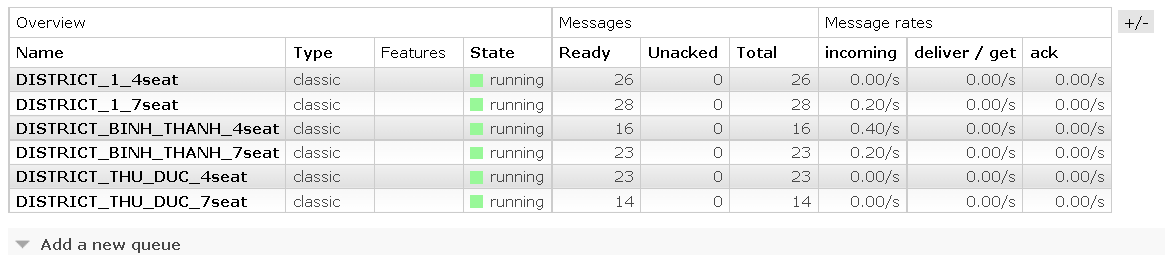




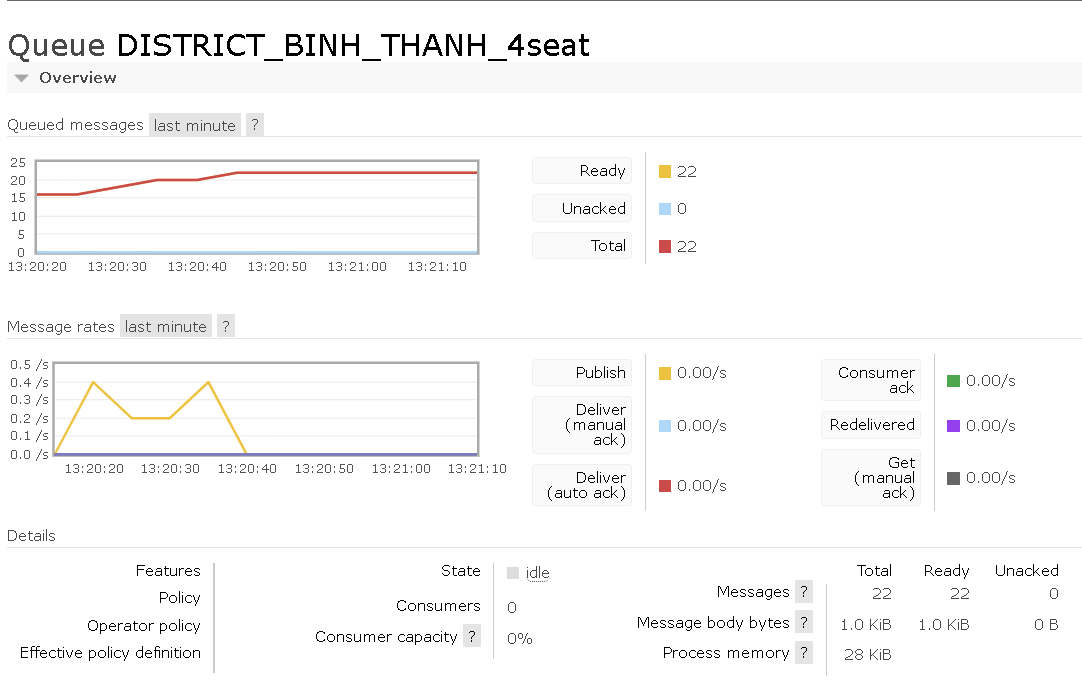
Now, at Queues on Rabbitmq website, there are some queues have Total = 0 and some have Total = 1 with Unacked = 1 (there’s still a message not be acked) and Ready = 0 (After the message be acked, there will be no message in the queue).



When all the consumers stop running. Total and Ready keep increasing the number of messages because those messages are not acked.

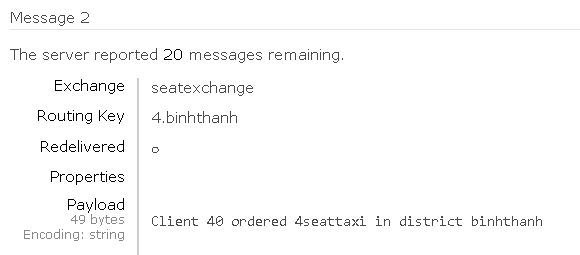


Then we move into queue DISTRICT\_BINH\_THANH\_4seat



Consumers = 0 so there is no place to receive the messages. 22 messages are now waiting in this queue.

This is a message is waiting inside the queue. It is delivered from ‘seatexchange’ bound with routing key name ‘4.binhthanh’. Since this message was delivered successfully so there is no flag at Redelivered. At the Payload, we can see a line of message from the producer.



**Conclusion**

RabbitMQ helps the web server to send response to requests very quickly instead of being forced to run a resource-hungry procedure on a system. Storing messages in the queues is a great solution or we want to distribute the message to many recipients to help reduce the load on the workers to handle.

**Withdraw**

Networks are unreliable and applications may fail to process messages, therefore, the AMQP model has a notion of message acknowledgments: when a message is delivered to a consumer the consumer notifies the broker, either automatically or as soon as the application developer chooses to do so. When message acknowledgments are in use, a broker will only completely remove a message from a queue when it receives a notification for that message (or group of messages).

For example, when a message cannot be routed, messages may be returned to publishers, dropped, or, if the broker implements an extension, placed into a so-called "dead letter queue". Publishers choose how to handle situations like this by publishing messages using specific parameters.

**Development orientation**

Because ours shortage of knownledge about programing in Python, we cannot add more suitable features for this project. In the future, we are going make upgrade the code which can help the taxi able to send every kind of messages back to the clients such as Confirming message, Canceling message, Late-notifying message,.. On the other hand, taxi can send their realtime destination to the clients.