

# Assignment 3 – CS6650

Yuchen Tian

1. URL for Github Repo: <https://github.com/tyuchn/cs6650-hw3-ytian>
2. Database Design and deployment topology

- a. Database Design

```
CREATE SCHEMA SkierApplication;  
use SkierApplication;
```

```
create table LiftRides(  
  id INT NOT NULL AUTO_INCREMENT,  
  skierId INT NOT NULL,  
  dayId INT NOT NULL,  
  time INT NOT NULL,  
  liftId INT NOT NULL,  
  PRIMARY KEY ( id )  
);
```

```
create table Resorts(  
  id INT NOT NULL AUTO_INCREMENT,  
  dayId INT NOT NULL,  
  year INT NOT NULL,  
  time INT NOT NULL,  
  resortId INT NOT NULL,  
  skierId INT NOT NULL,  
  liftId INT NOT NULL,  
  PRIMARY KEY ( id ),  
);
```

LiftRides table is used to store the skier and lift ride information for specific season. To consume message from message queue ASAP, each request will be stored as a single entry in the DB. As a result, the LiftRides table has the fields of skier id, id, time, lift id, the primary key is automatically assigned by MySQL. To enable the queries like “For skier N, XXX”, we need to group the rows by skier Id while executing SQL queries.

Vice versa, Resort table is used to store data pertinent to the ski resort. It has the fields of dayId, year, time, resortId, skierId and liftId. The primary key is also automatically incremented for each record. To enable queries like “For day N, XXX”, we can select on the specific day and do some “group by” operations.

- b. Deployment topology

I created an MySQL instance on RDS, the attribute is db.t2.micro with 1 vCPU and 1 GB RAM at the beginning. The JDBC connection was established on consumer microservices.

3. Test Results – original setup

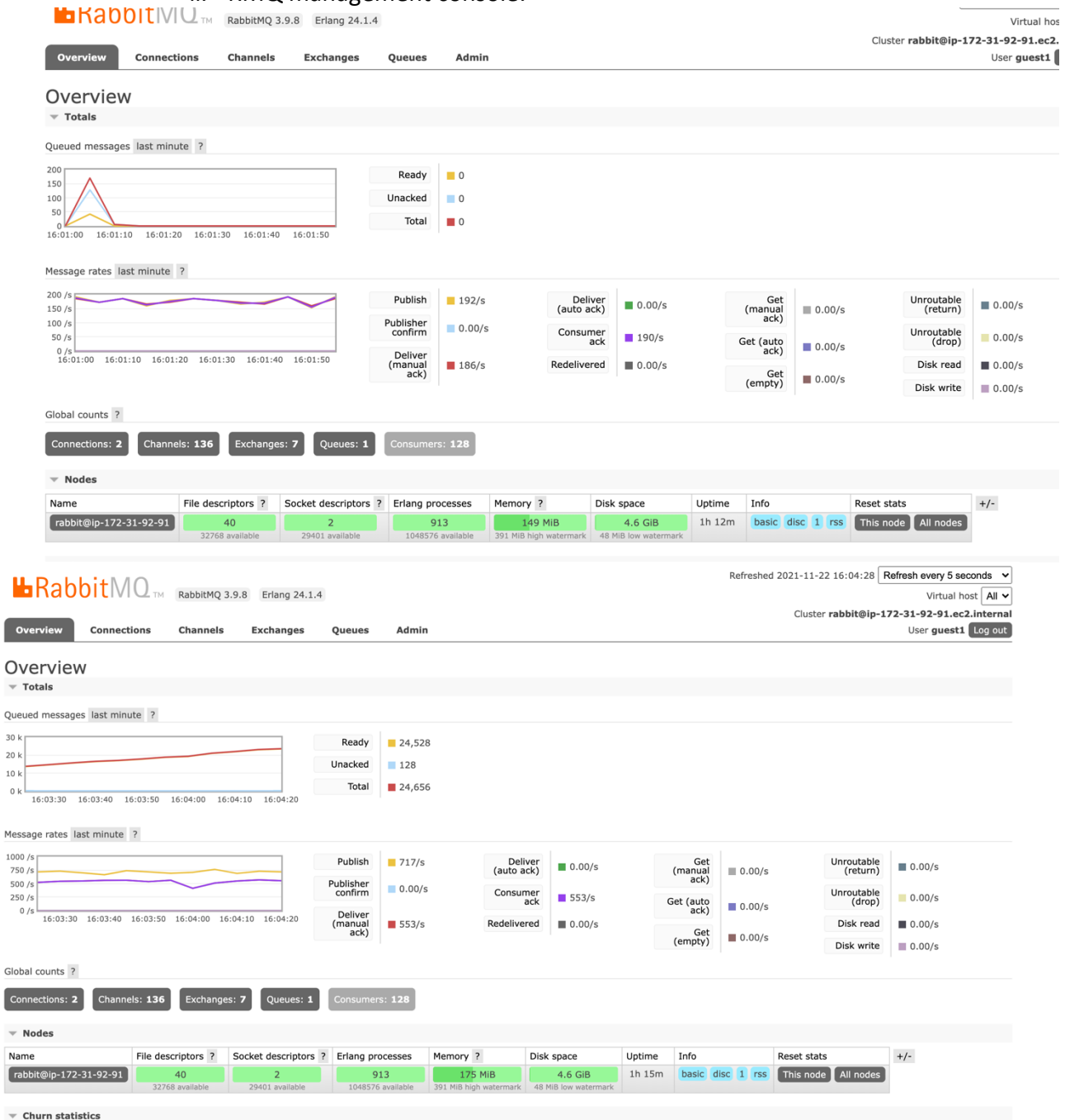
At the first step, both the consumer and db instance are t2.micro, the running result showed as below

a. Step1 – 128 threads

i. Commands lines:

*-numThreads 128 -numSkiers 100000 -address 54.82.5.13:8080*

ii. RMQ management console:



iii. Result:

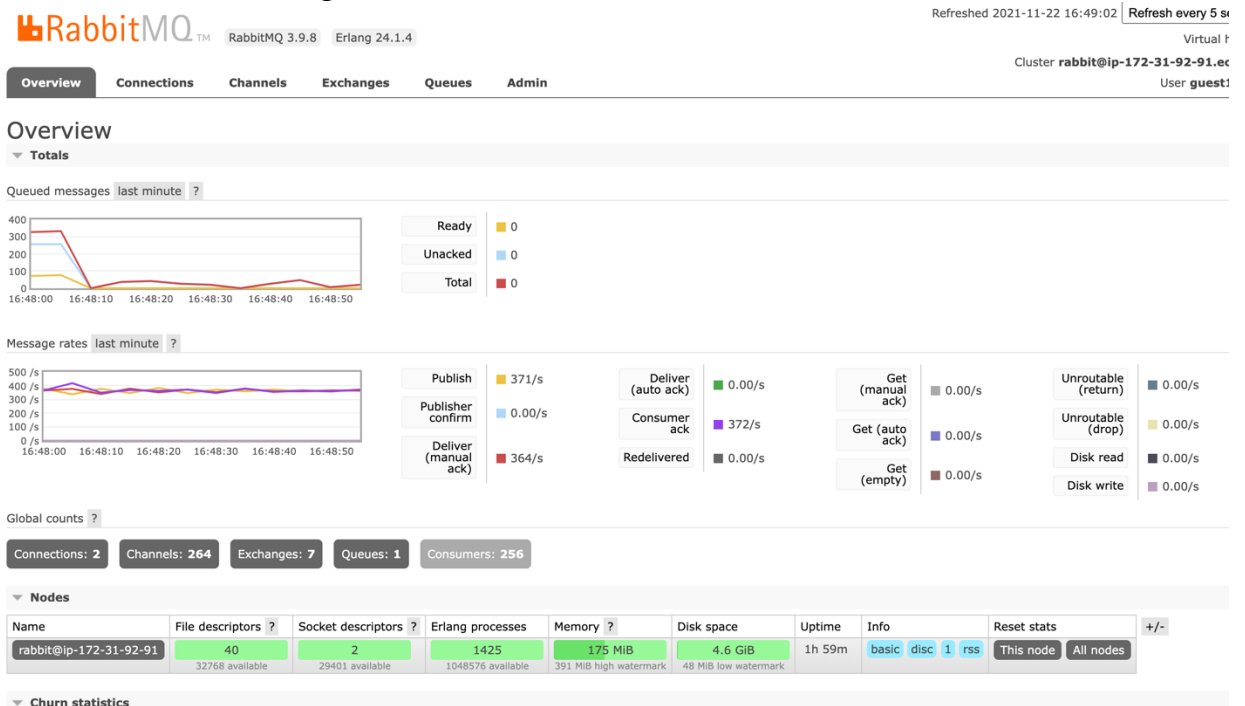
```
SkierClient x
/Library/Java/JavaVirtualMachines/temu
Successful requests sent 159840
Unsuccessful requests 0
WallTime 392736ms
Total throughput per ms 406.99094
Process finished with exit code 0
```

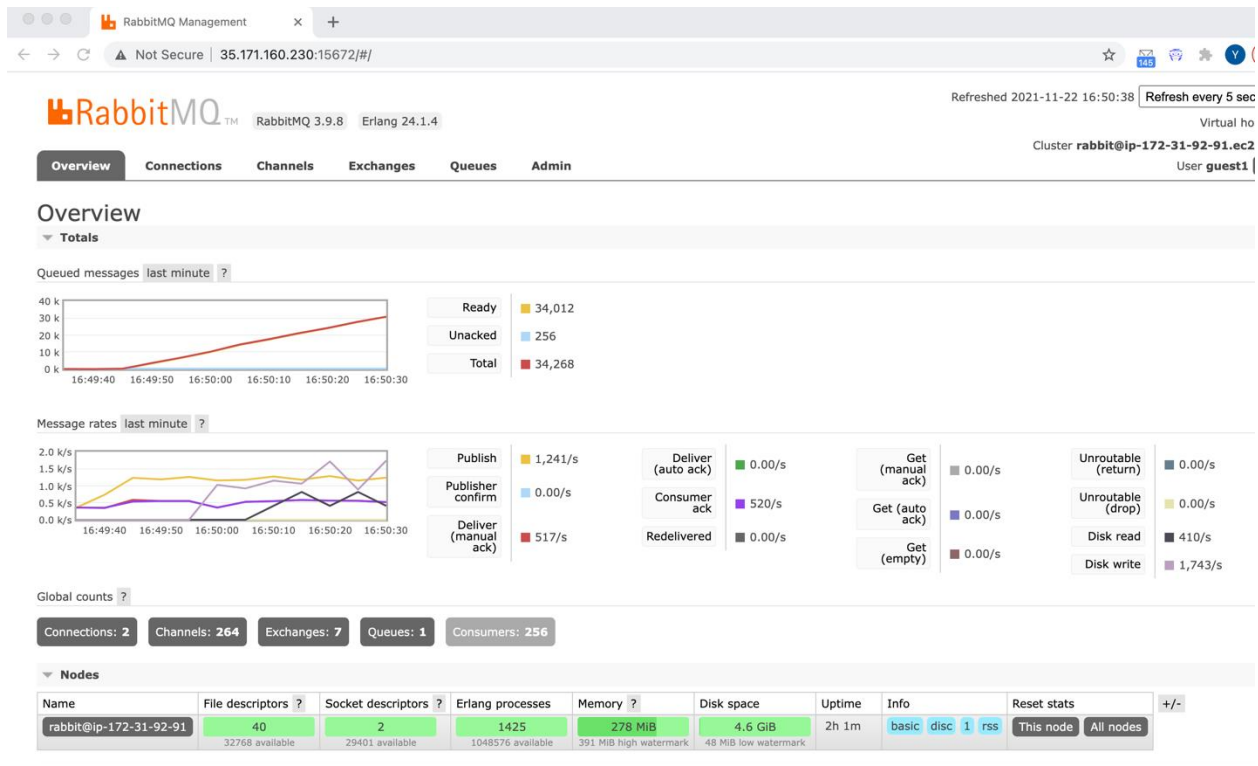
b. Step1 – 256 threads

i. Commands lines:

*-numThreads 256 -numSkiers 100000 -address 54.82.5.13:8080*

ii. RMQ management console:





iii. Result:

```
SkierClient x
/Library/Java/JavaVirtualMachines/temu
Successful requests sent 159808
Unsuccessful requests 0
WallTime 215512ms
Total throughput per ms 741.52716
Process finished with exit code 0
```

c. Step2 – 128 threads

i. Commands lines:

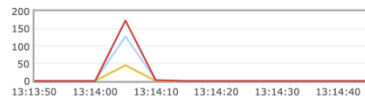
*-numThreads 128 -numSkiers 100000 -address 54.82.5.13:8080*

ii. RMQ management console:

## Overview

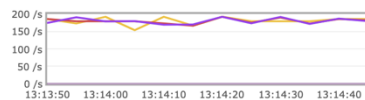
### Totals

Queued messages last minute ?



Ready 0  
Unacked 0  
Total 0

Message rates last minute ?



Publish 186/s  
Publisher confirm 0.00/s  
Deliver (manual ack) 182/s  
Deliver (auto ack) 0.00/s  
Consumer ack 180/s  
Redelivered 0.00/s  
Get (manual ack) 0.00/s  
Get (auto ack) 0.00/s  
Get (empty) 0.00/s  
Unroutable (return) 0.00/s  
Unroutable (drop) 0.00/s  
Disk read 0.00/s  
Disk write 0.00/s

Global counts ?

Connections: 2 Channels: 136 Exchanges: 7 Queues: 2 Consumers: 128

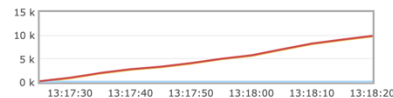
### Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@ip-172-31-92-91	40 32768 available	2 29401 available	915 1048576 available	164 MiB 391 MiB high watermark	4.6 GiB 48 MiB low watermark	22h 25m	basic disc 1 rss	This node All nodes	

## Overview

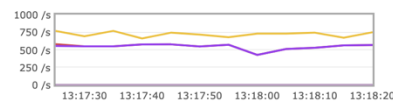
### Totals

Queued messages last minute ?



Ready 10,419  
Unacked 128  
Total 10,547

Message rates last minute ?



Publish 749/s  
Publisher confirm 0.00/s  
Deliver (manual ack) 567/s  
Deliver (auto ack) 0.00/s  
Consumer ack 567/s  
Redelivered 0.00/s  
Get (manual ack) 0.00/s  
Get (auto ack) 0.00/s  
Get (empty) 0.00/s  
Unroutable (return) 0.00/s  
Unroutable (drop) 0.00/s  
Disk read 0.00/s  
Disk write 0.00/s

Global counts ?

Connections: 2 Channels: 136 Exchanges: 7 Queues: 2 Consumers: 128

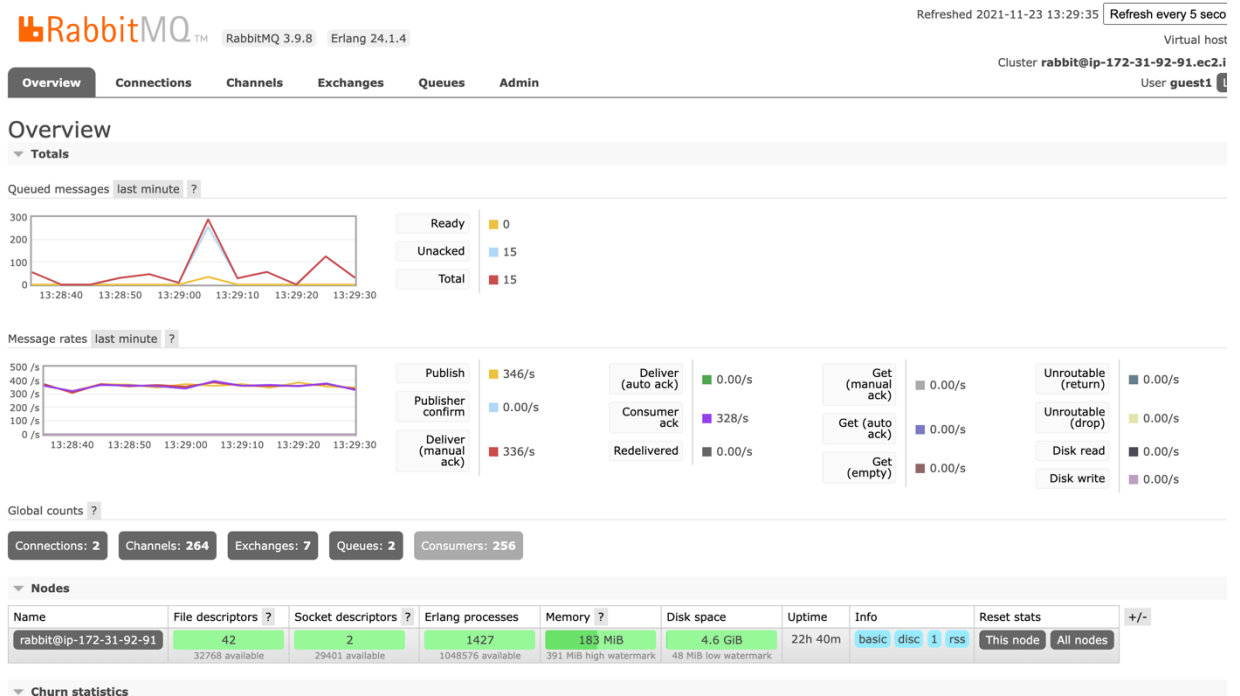
### Nodes

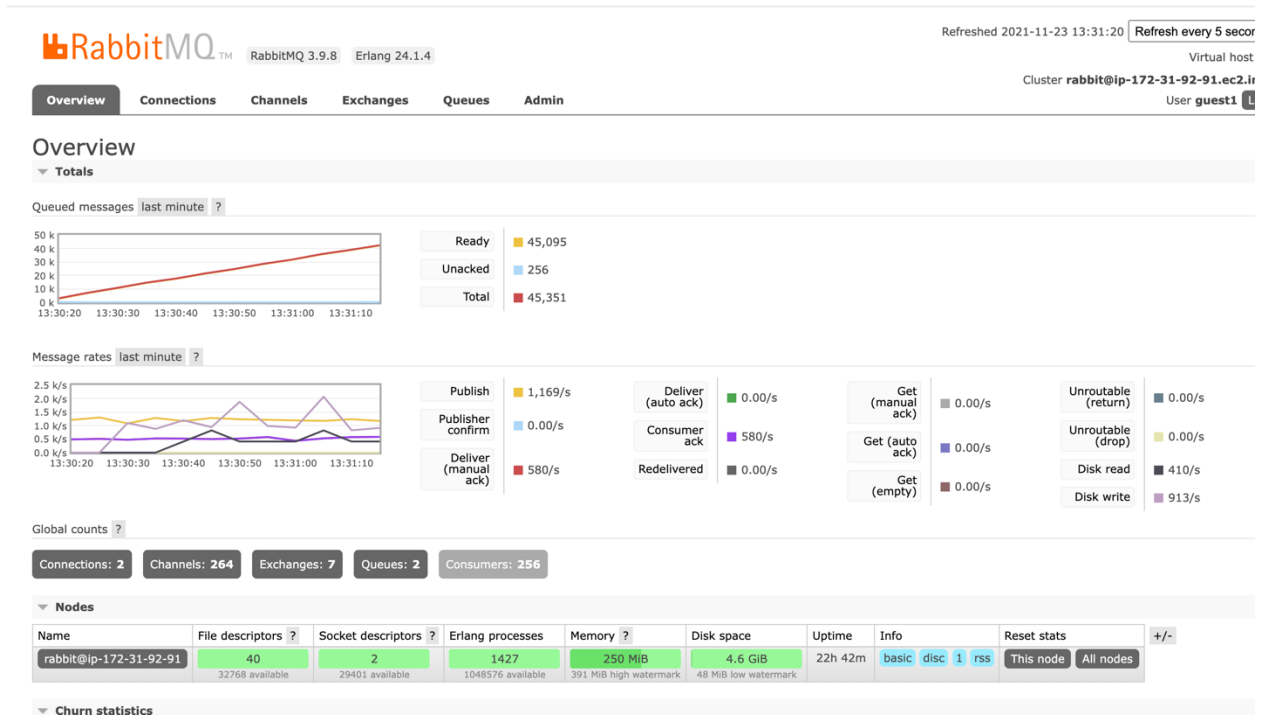
Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@ip-172-31-92-91	42 32768 available	2 29401 available	915 1048576 available	203 MiB 391 MiB high watermark	4.6 GiB 48 MiB low watermark	22h 29m	basic disc 1 rss	This node All nodes	

iii. Result:

```
SkierClient x
/Library/Java/JavaVirtualMachines/te
Successful requests sent 159840
Unsuccessful requests 0
WallTime 392670ms
Total throughput per ms 407.05936
Process finished with exit code 0
```

- d. Step2 – 256 threads
  - i. Commands lines:  
*-numThreads 256 -numSkiers 100000 -address 54.82.5.13:8080*
  - ii. RMQ management console:





### iii. Result:

```
SkierClient x
/Library/Java/JavaVirtualMachines/temurin-1
Successful requests sent 159808
Unsuccessful requests 0
WallTime 215391ms
Total throughput per ms 741.9437
Process finished with exit code 0
```

### 4. Test Results – after increasing capacity

At this step, both the consumer and db instance are increased from t2.micro to t2.large

#### a. Step1 – 128 threads

##### i. Commands lines:

*-numThreads 128 -numSkiers 100000 -address 54.82.5.13:8080*

##### ii. RMQ management console:



RabbitMQ 3.9.8 Erlang 24.1.4

Refreshed 2021-11-23 15:20:01 Refresh every 5 sec

Virtual host

Cluster rabbit@ip-172-31-92-91.ec2

User guest1

Overview Connections Channels Exchanges Queues Admin

## Overview

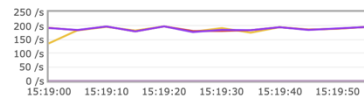
### Totals

Queued messages last minute ?



Ready 0  
Unacked 0  
Total 0

Message rates last minute ?



Publish 196/s  
Publisher confirm 0.00/s  
Deliver (manual ack) 195/s  
Deliver (auto ack) 0.00/s  
Consumer ack 195/s  
Redelivered 0.00/s  
Get (manual ack) 0.00/s  
Get (auto ack) 0.00/s  
Get (empty) 0.00/s  
Unroutable (return) 0.00/s  
Unroutable (drop) 0.00/s  
Disk read 0.00/s  
Disk write 0.00/s

Global counts ?

Connections: 2 Channels: 136 Exchanges: 7 Queues: 2 Consumers: 128

### Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@ip-172-31-92-91	40 32768 available	2 29401 available	915 1048576 available	160 MIB 391 MIB high watermark	4.6 GiB 48 MIB low watermark	1d 0h	basic disc 1 rss	This node All nodes	

### Churn statistics



RabbitMQ 3.9.8 Erlang 24.1.4

Refreshed 2021-11-23 21:48:30 Refresh every 5 sec

Virtual host

Cluster rabbit@ip-172-31-92-91.ec2

User guest1

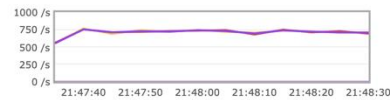
Overview Connections Channels Exchanges Queues Admin

Queued messages last minute ?



Ready 0  
Unacked 72  
Total 72

Message rates last minute ?



Publish 709/s  
Publisher confirm 0.00/s  
Deliver (manual ack) 690/s  
Deliver (auto ack) 0.00/s  
Consumer ack 707/s  
Redelivered 0.00/s  
Get (manual ack) 0.00/s  
Get (auto ack) 0.00/s  
Get (empty) 0.00/s  
Unroutable (return) 0.00/s  
Unroutable (drop) 0.00/s  
Disk read 0.00/s  
Disk write 0.00/s

Global counts ?

Connections: 2 Channels: 136 Exchanges: 7 Queues: 2 Consumers: 128

### Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@ip-172-31-92-91	40 32768 available	2 29401 available	914 1048576 available	150 MIB 388 MIB high watermark	4.5 GiB 48 MIB low watermark	6m 54s	basic disc 1 rss	This node All nodes	

### Churn statistics

Connection operations last minute ?

iii. Result:



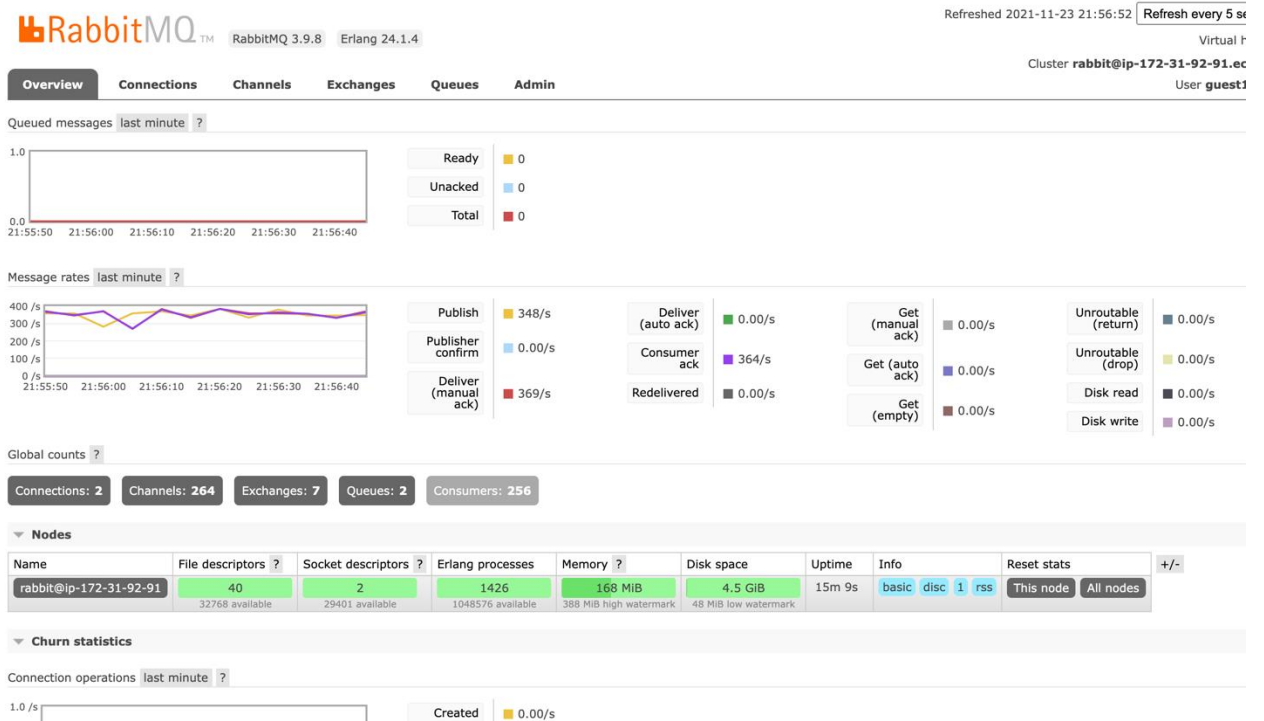
```
/Library/Java/JavaVirtualMachines/temu
Successful requests sent 159840
Unsuccessful requests 0
WallTime 390395ms
Total throughput per ms 409.4315
Process finished with exit code 0
```

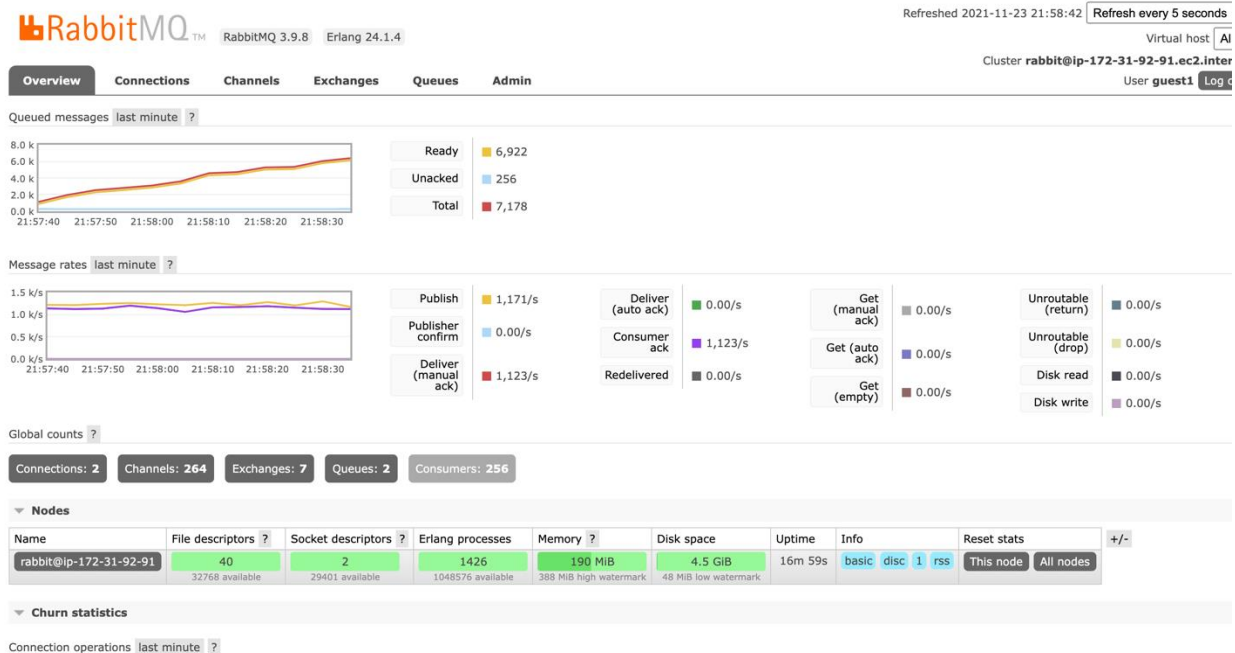
b. Step1 – 256 threads

i. Commands lines:

*-numThreads 256 -numSkiers 100000 -address 54.82.5.13:8080*

ii. RMQ management console:





### iii. Result:

```

/Library/Java/JavaVirtualMachines/temuri
Successful requests sent 159808
Unsuccessful requests 0
WallTime 215358ms
Total throughput per ms 742.0574

Process finished with exit code 0

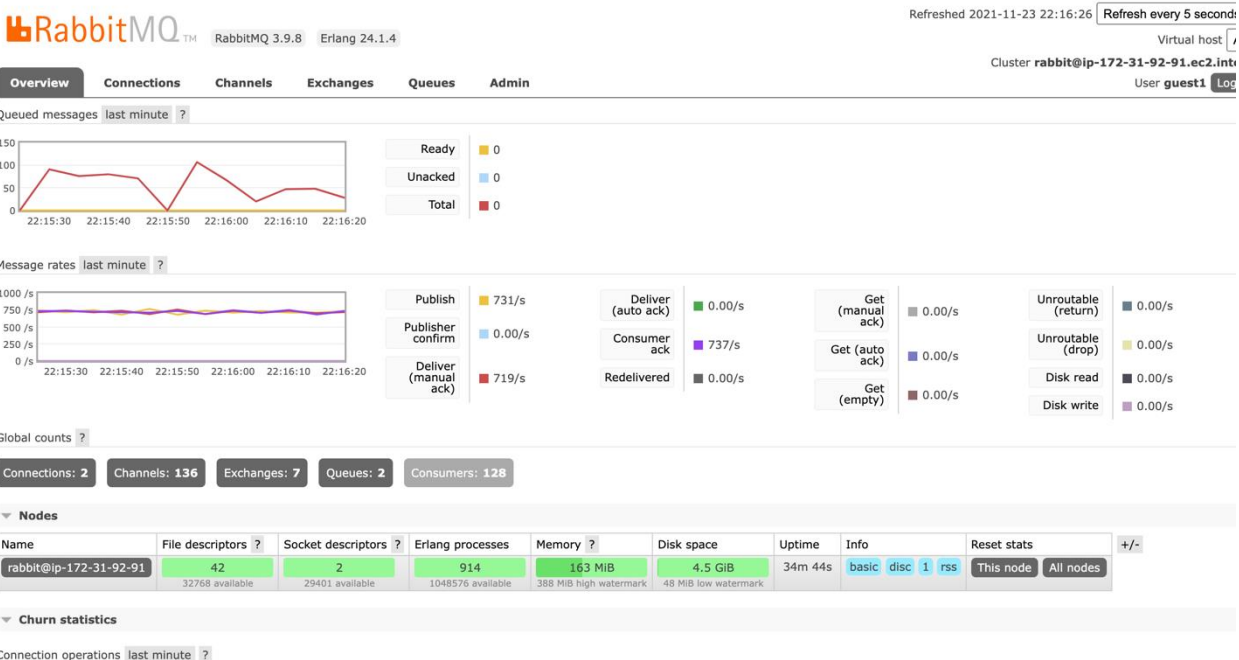
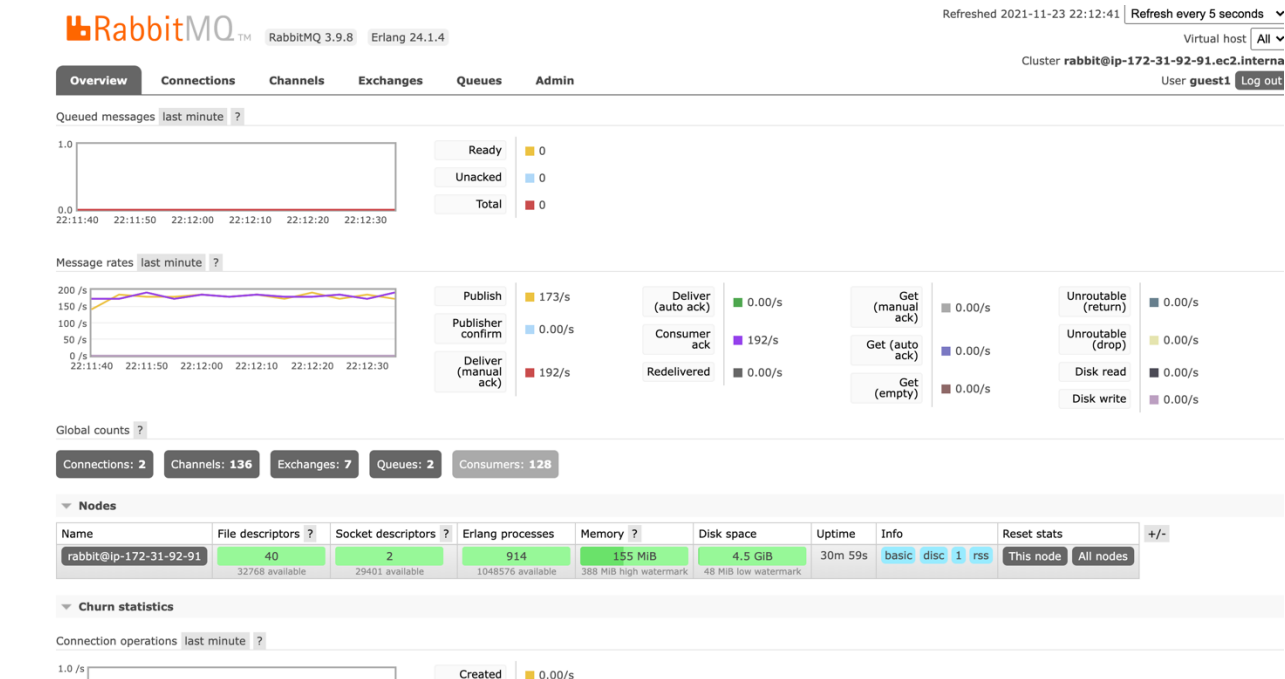
```

### c. Step2 – 128 threads

#### i. Commands lines:

*-numThreads 128 -numSkiers 100000 -address 54.82.5.13:8080*

#### ii. RMQ management console:



iii. Result:

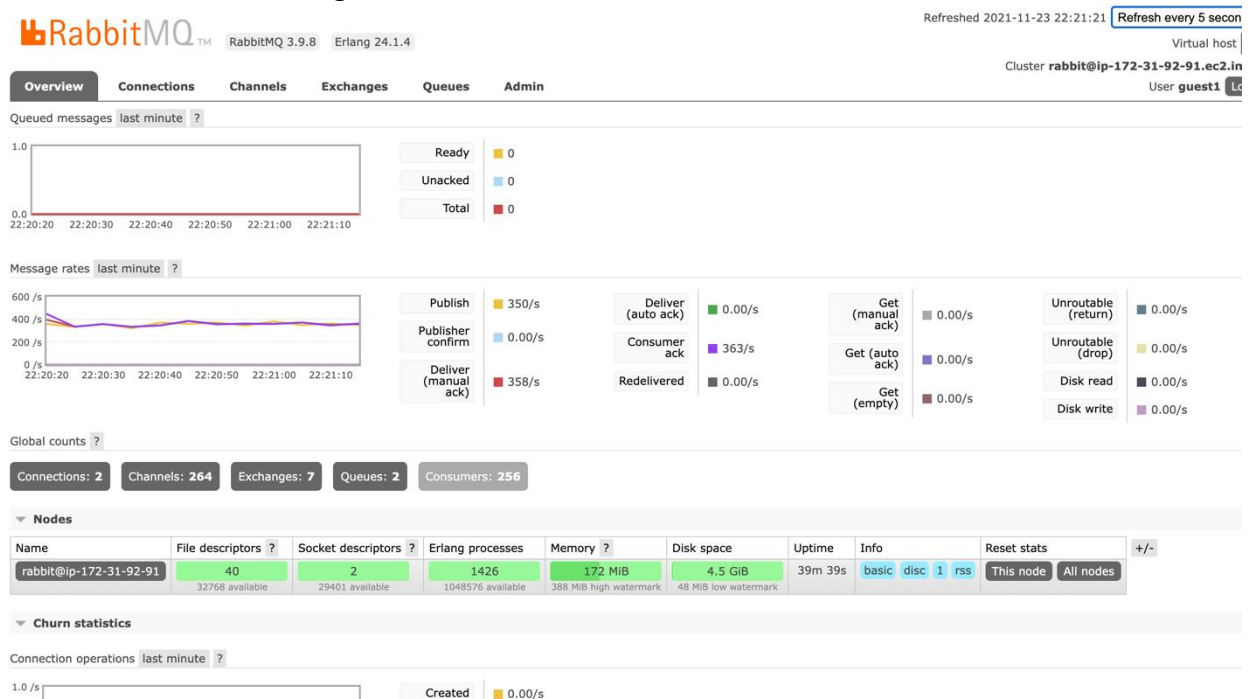
```
SkierClient x
/Library/Java/JavaVirtualMachines/temur
Successful requests sent 159840
Unsuccessful requests 0
WallTime 390730ms
Total throughput per ms 409.08044
Process finished with exit code 0|
```

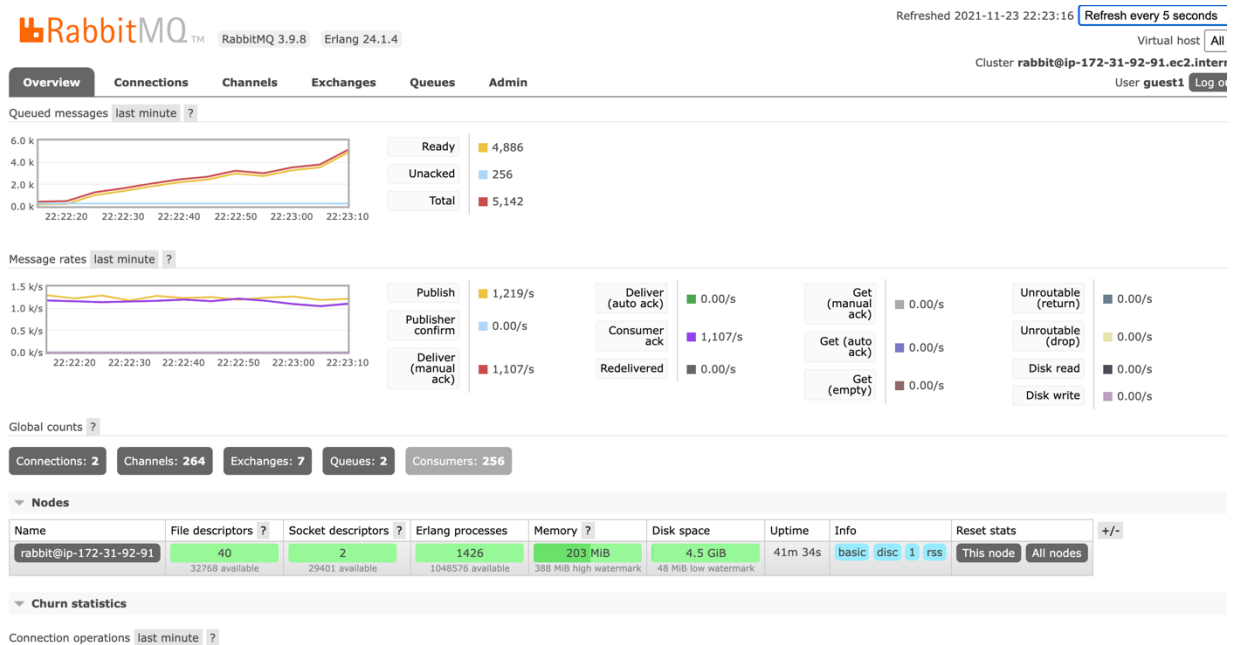
d. Step2 – 256 threads

i. Commands lines:

*-numThreads 256 -numSkiers 100000 -address 54.82.5.13:8080*

ii. RMQ management console:





### iii. Result:

```

/Library/Java/JavaVirtualMachines/tem
Successful requests sent 159808
Unsuccessful requests 0
WallTime 214721ms
Total throughput per ms 744.25885

Process finished with exit code 0

```

## 5. Result Analysis

### a. Comparisons:

	Max Queue Length (backlogs)
Step1 – 128 threads (before)	24528
Step1 – 128 threads (after)	72
Step 1 – 256 threads (before)	45548
Step 1 – 256 threads (after)	6922
Step 2 – 128 threads (before)	28584
Step 2 – 128 threads (after)	100
Step 2 – 256 threads (before)	45095
Step 2 – 256 threads (after)	4886

- b. Analysis: At the beginning, for both 128 threads and 256 threads in Step 1&2, we can see significant (tens of thousand) backlog in the message queue, which was caused by the DB bottleneck with no double. In order to eliminate it, I chose to increase the capacity of both the DB and consumer. After increase the size from t2.micro to t2.large, we can see the result for 128 thread, peek of the backlog is less than 100, while for 256 threads, peek of backlog is less than 7k. Because we enhanced the capacity of our RAM and add more memory to the instance, we have higher I/O speed. As a result, the decrement of backlogs is significant, the bottleneck was mitigated significantly. So, we can say our solution is successful.