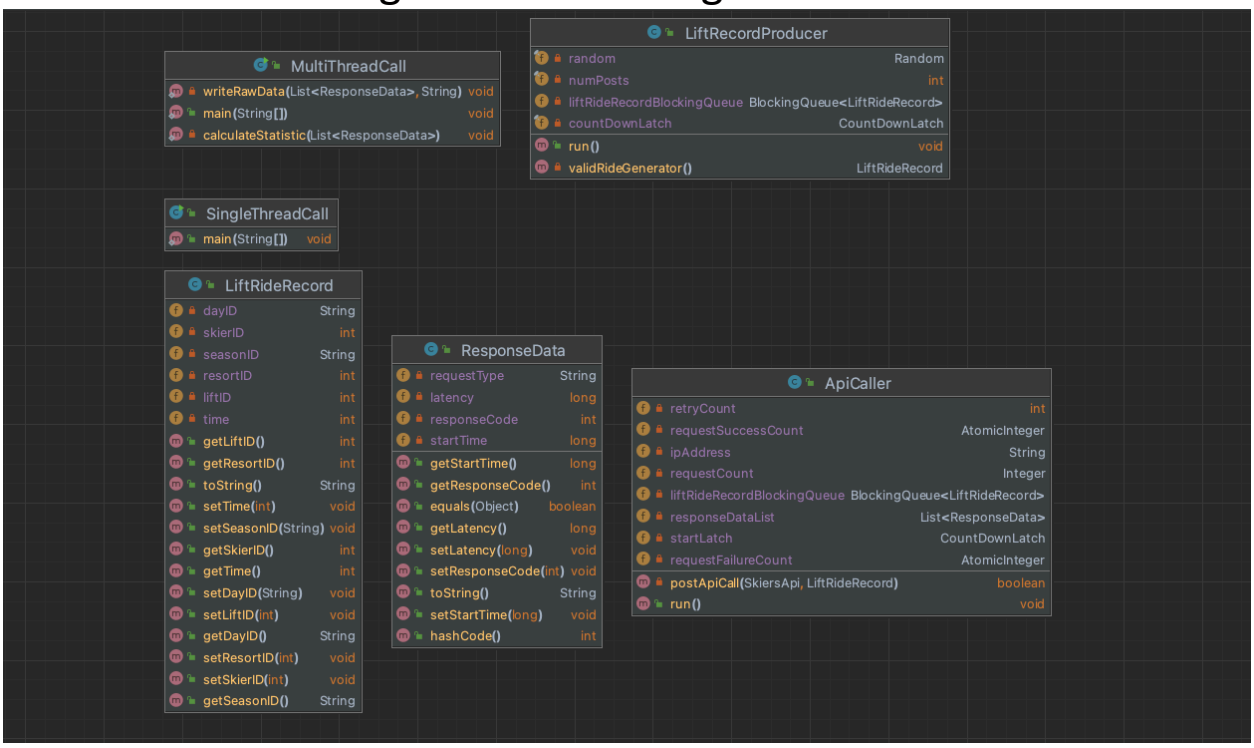
Assignment 2 Zegui Jiang

Client 2 – Remaining same with Assignment 1



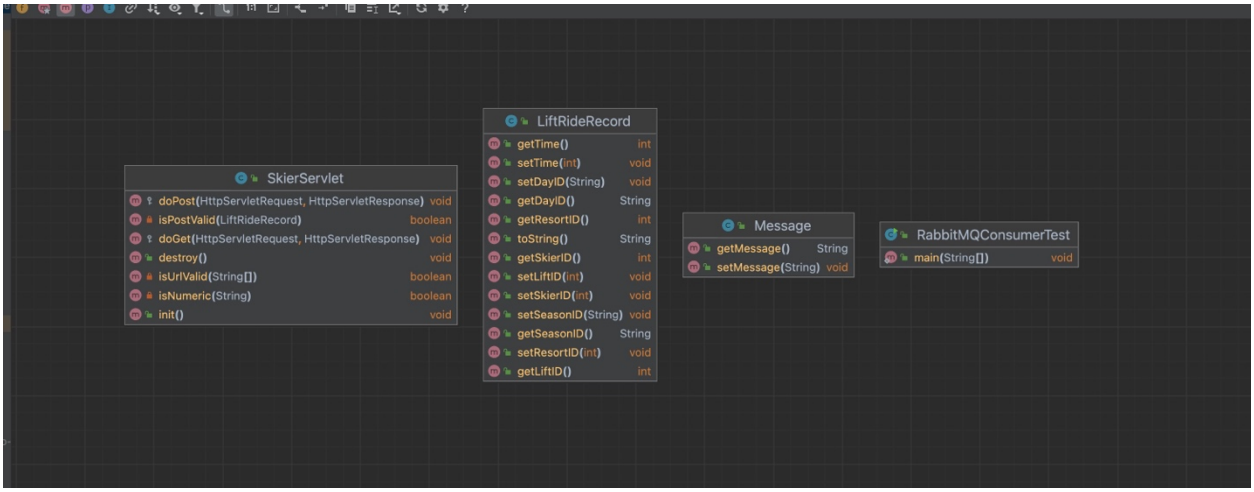
Client 2 extends the functionality of client 1 by enhancing the MultiThreadCall class to include capabilities for writing raw data to a CSV file and calculating statistical metrics. To fulfill these additional requirements, a new class named ResponseData is introduced to encapsulate the response records.

The MultiThreadCall class in client 2 would have these new responsibilities:

Writing Response Records: After each API call, the response details are captured in an instance of the ResponseData class. This object includes all relevant information that needs to be logged, such as timestamps, response status, and any payload data returned by the API.

Calculating Statistics: The class is responsible for computing various performance metrics based on the response data collected. These statistics might include the total number of requests, success rate, failure rate, average response time, and other relevant performance indicators.

LIftServer



The server upgraded the functionality for processing messages. Firstly, it connects to RabbitMQ, then it sends the received message queue to RabbitMQ.

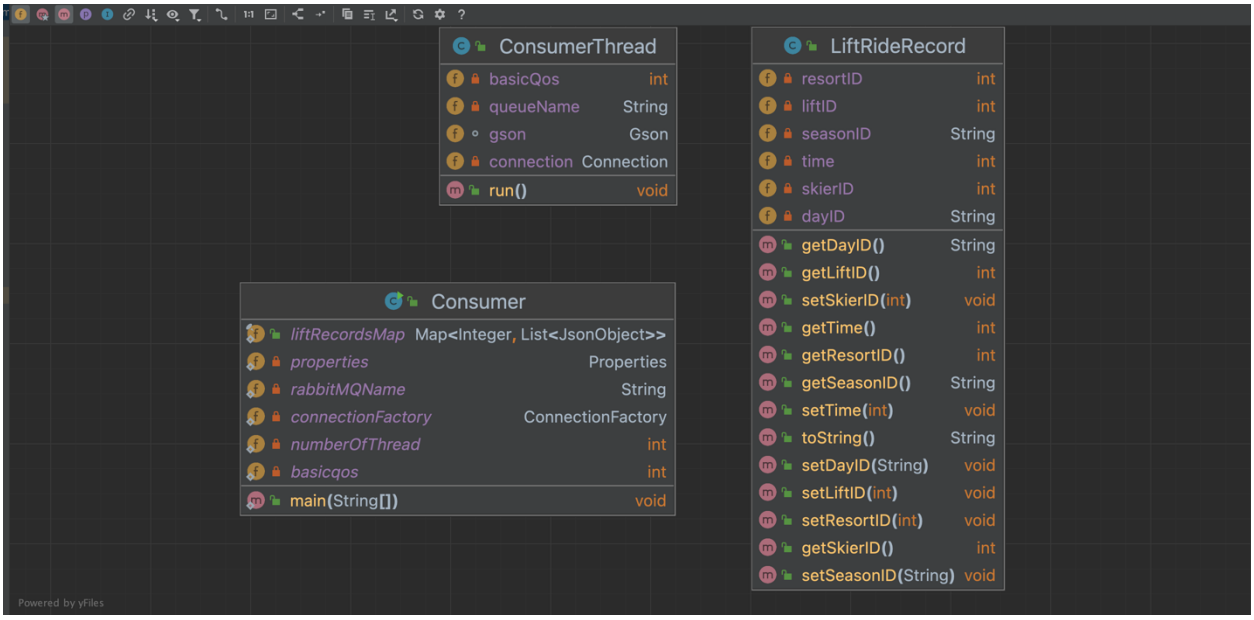
Init(): Reads and configures settings from the `rabbitmq.conf` configuration file found in resources. It extracts the RabbitMQ queue name and the number of channels from the properties file. All channels are stored in a channel pool, which is a Linked Blocking Queue.

Destroy(): Closes all open resources when the process is terminated.

IsPostValid and IsUrlValid: These functions are likely responsible for validating the incoming request's POST data and the URL, respectively.

doPost: Handles incoming POST requests by receiving a message and forwarding it to RabbitMQ.

RabbitMQ Consumer



ConsumerThread:

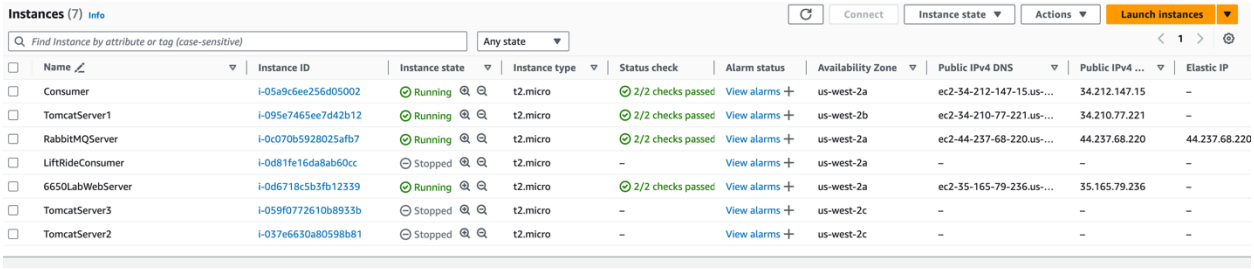
- It creates a new channel Thread on the provided connection.
- It declares a queue with the given queueName without any extra arguments
- It sets the basicQos for the channel, which controls how many messages the server will deliver to the consumer before acknowledgments are received.
- Attempts to parse and consume the message into a JsonObject and retrieves a SkierID, and store them to `liftRecordsMap`

Consumer:

- Loading Configuration: load a RabbitMQ configuration file (`rabbitmq.conf`). including settings such as host, port, username, password, queue name, the default number of threads for consumers `numberOfThread`, and the default `basicQos`
- Setting Up Connection Factory and Starting Consumer Threads: An `ExecutorService` with a fixed thread pool size, based on `numberOfThread`, is created to manage consumer threads. For each thread, a new `ConsumerThread` is executed. This thread is responsible for handling messages from the specified RabbitMQ queue (`rabbitMQName`) with the given basicQoS.

With Single Server

Basic Structure, and all run in ec2:



Instances (7) Info

Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | Any state

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Consumer	i-05a9c6ee256d05002	Running	t2.micro	2/2 checks passed	View alarms	us-west-2a	ec2-34-212-147-15.us-...	34.212.147.15	–
TomcatServer1	i-095e7465ee7d42b12	Running	t2.micro	2/2 checks passed	View alarms	us-west-2b	ec2-34-210-77-221.us-...	34.210.77.221	–
RabbitMQServer	i-0c070b5928025afb7	Running	t2.micro	2/2 checks passed	View alarms	us-west-2a	ec2-44-237-68-220.us-...	44.237.68.220	44.237.68.220
LiftRideConsumer	i-0d81fe16da8ab60cc	Stopped	t2.micro	–	View alarms	us-west-2a	–	–	–
6650LabWebServer	i-0d6718c5b3fb12339	Running	t2.micro	2/2 checks passed	View alarms	us-west-2a	ec2-35-165-79-236.us-...	35.165.79.236	–
TomcatServer3	i-059f0772610b8933b	Stopped	t2.micro	–	View alarms	us-west-2c	–	–	–
TomcatServer2	i-037e6630a80598b81	Stopped	t2.micro	–	View alarms	us-west-2c	–	–	–

Client -> Tomcat -> RabbitMq -> Consumer

Following data points includes Overall throughput (should be greater than no load balancer), short queue sizes and profile

Number of Thread in Client: 100
Number of Thread in Consumer: 24
Number of basicqos: 1

```
/Users/monarch/Library/Java/JavaVirtualMachines/corretto-11.0.22/Contents/Home/bin/java ...
Statistic Metrics
Mean Response Time: 51.323105 ms
Median Response Time: 49.0 ms
P99 Response Time: 70 ms
Min Response Time: 36 ms
Max Response Time: 4681 ms
Summary:
Number of thread in process 2: 100
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 137496
Response Time: 0.68748 ms/request
RPS: 1454 requests/second

Process finished with exit code 0
```

Number of Thread in Client: 200
Number of Thread in Consumer: 24
Number of basicqos: 1

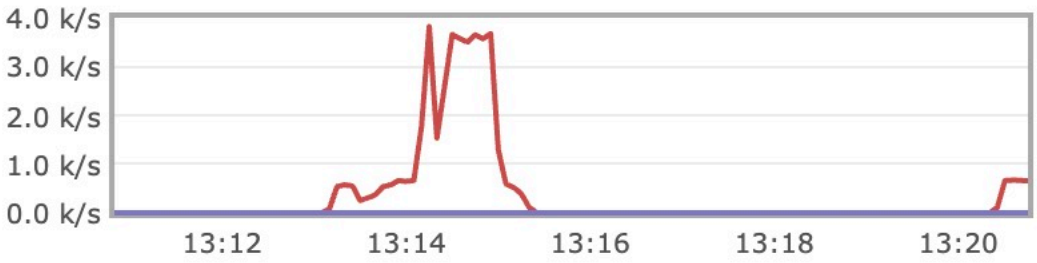
```
Statistic Metrics
Mean Response Time: 72.96736114236002 ms
Median Response Time: 51.0 ms
P99 Response Time: 390 ms
Min Response Time: 35 ms
Max Response Time: 10850 ms
Summary:
Number of thread in process 2: 200
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 110576
Response Time: 0.55288 ms/request
RPS: 1808 requests/second
```

Queue liftQueue

Overview

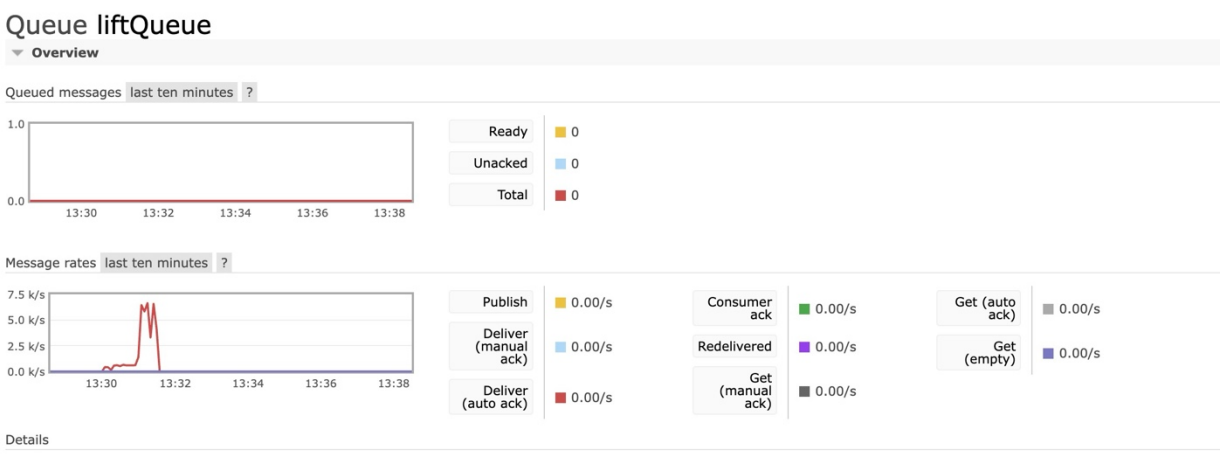
Queued messages last ten minutes ?

Message rates last ten minutes ?



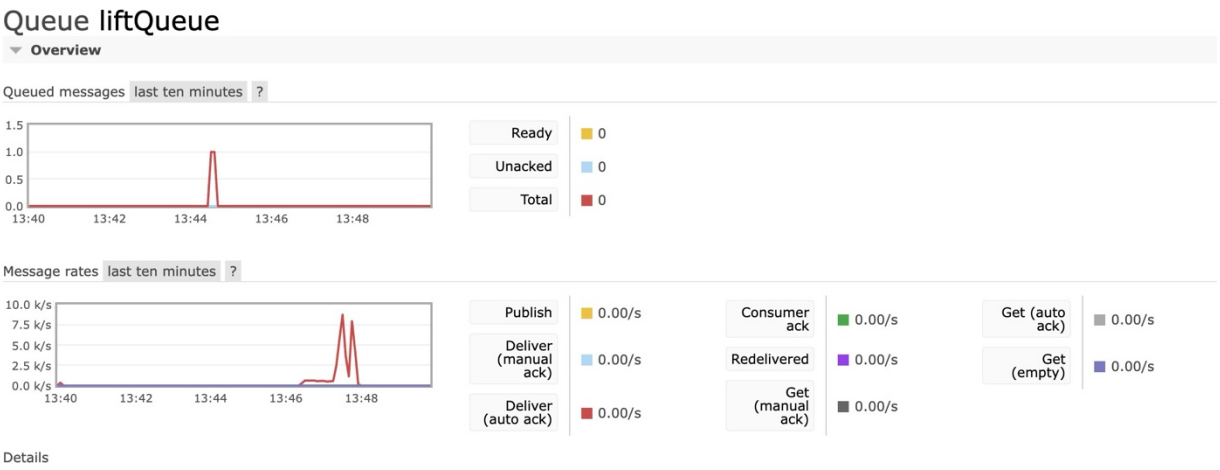
Number of Thread in Client: 400
Number of Thread in Consumer: 24
Number of basicqos: 1

```
Successful consume object: {"skierID":86546,"resortID":7,"liftID":3,"seasonID":"2024","dayID":"1","time":198}, Thread Id is: 81
Successful consume object: {"skierID":85187,"resortID":10,"liftID":11,"seasonID":"2024","dayID":"1","time":39}, Thread Id is: 81
Successful consume object: {"skierID":25735,"resortID":7,"liftID":10,"seasonID":"2024","dayID":"1","time":73}, Thread Id is: 81
Successful consume object: {"skierID":56914,"resortID":3,"liftID":5,"seasonID":"2024","dayID":"1","time":259}, Thread Id is: 81
Successful consume object: {"skierID":72317,"resortID":2,"liftID":4,"seasonID":"2024","dayID":"1","time":62}, Thread Id is: 81
Successful consume object: {"skierID":34552,"resortID":9,"liftID":4,"seasonID":"2024","dayID":"1","time":135}, Thread Id is: 81
Successful consume object: {"skierID":26379,"resortID":9,"liftID":35,"seasonID":"2024","dayID":"1","time":290}, Thread Id is: 81
Successful consume object: {"skierID":87954,"resortID":10,"liftID":24,"seasonID":"2024","dayID":"1","time":2}, Thread Id is: 81
Successful consume object: {"skierID":91827,"resortID":8,"liftID":37,"seasonID":"2024","dayID":"1","time":125}, Thread Id is: 81
Successful consume object: {"skierID":62549,"resortID":9,"liftID":29,"seasonID":"2024","dayID":"1","time":193}, Thread Id is: 81
Successful consume object: {"skierID":59324,"resortID":2,"liftID":33,"seasonID":"2024","dayID":"1","time":103}, Thread Id is: 81
Successful consume object: {"skierID":29832,"resortID":7,"liftID":21,"seasonID":"2024","dayID":"1","time":271}, Thread Id is: 81
Successful consume object: {"skierID":31727,"resortID":10,"liftID":17,"seasonID":"2024","dayID":"1","time":59}, Thread Id is: 81
Successful consume object: {"skierID":52333,"resortID":6,"liftID":33,"seasonID":"2024","dayID":"1","time":267}, Thread Id is: 81
Successful consume object: {"skierID":42444,"resortID":2,"liftID":10,"seasonID":"2024","dayID":"1","time":277}, Thread Id is: 81
Successful consume object: {"skierID":46357,"resortID":8,"liftID":29,"seasonID":"2024","dayID":"1","time":89}, Thread Id is: 81
/Users/monarch/Library/Java/JavaVirtualMachines/corretto-11.0.22/Contents/Home/bin/java ...
Statistic Metrics
Mean Response Time: 67.984135 ms
Median Response Time: 56.0 ms
P99 Response Time: 331 ms
Min Response Time: 37 ms
Max Response Time: 8457 ms
Summary:
Number of thread in process 2: 400
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 85952
Response Time: 0.42976 ms/request
RPS: 2326 requests/second

Process finished with exit code 0
```

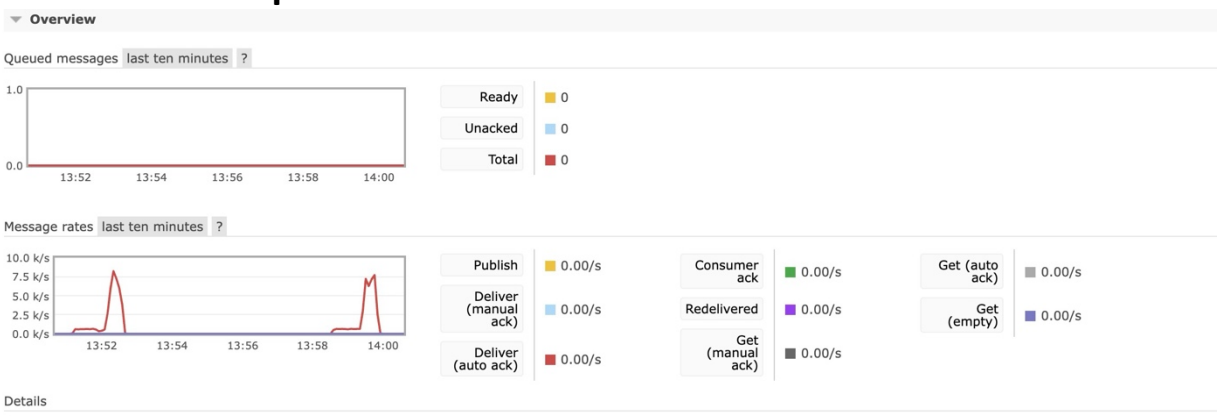


Number of Thread in Client: 800
Number of Thread in Consumer: 24
Number of basicqos: 1
Note: I have a test on connection so it has a peak in queue

```
Statistic Metrics
Mean Response Time: 130.57612732625853 ms
Median Response Time: 66.0 ms
P99 Response Time: 1125 ms
Min Response Time: 38 ms
Max Response Time: 10792 ms
Summary:
Number of thread in process 2: 800
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 78955
Response Time: 0.394775 ms/request
RPS: 2533 requests/second
```



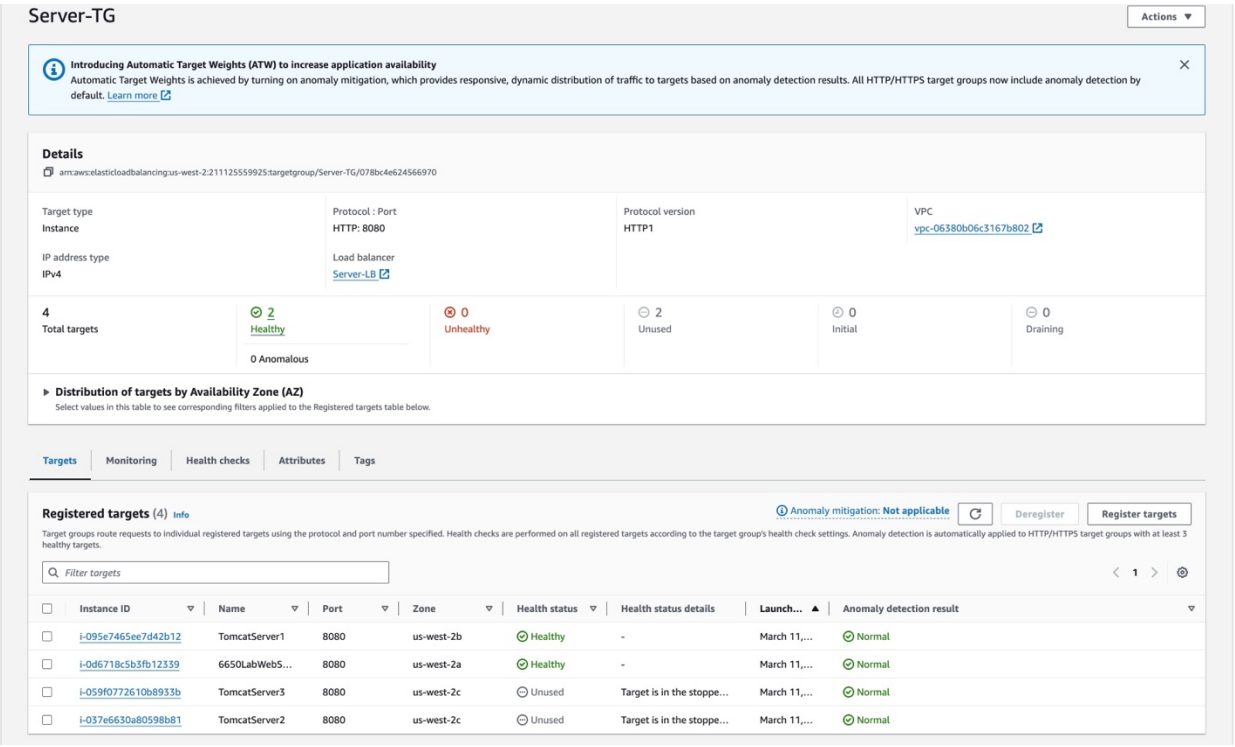
Number of Thread in Client: 1200
Number of Thread in Consumer: 24
Number of basicqos: 1



```
Mean Response Time: 148.34545091081736 ms
Median Response Time: 66.0 ms
P99 Response Time: 1469 ms
Min Response Time: 37 ms
Max Response Time: 14810 ms
Summary:
Number of thread in process 2: 1200
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 65446
Response Time: 0.32723 ms/request
RPS: 3055 requests/second
```

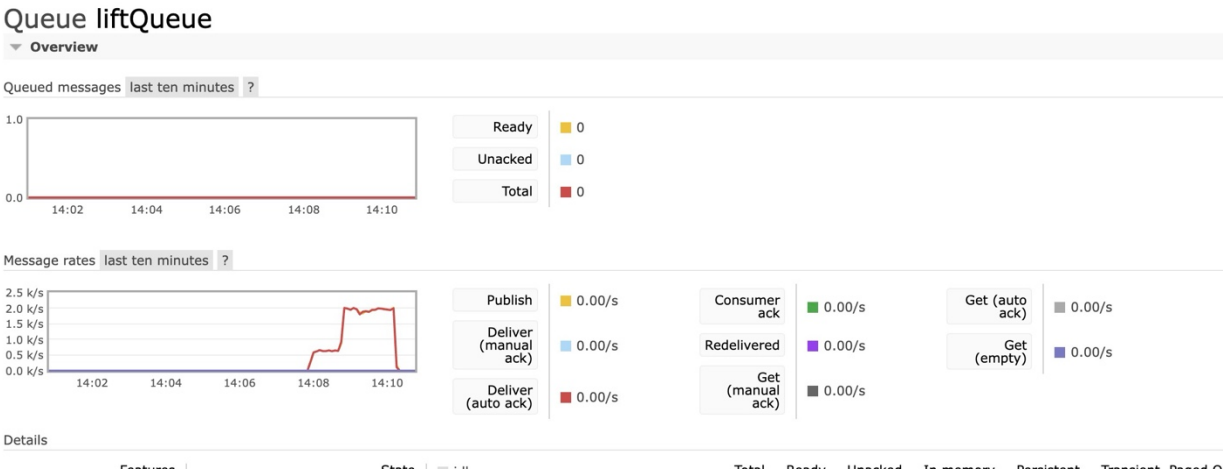
With Two Server

Create image from first server and stat a new instance TomcatServer1 base on the image. Put the TomcatServer1 in to Target group with same vpc and create load balancer for the group. Checked healthy with port 8080. Let’s see performance

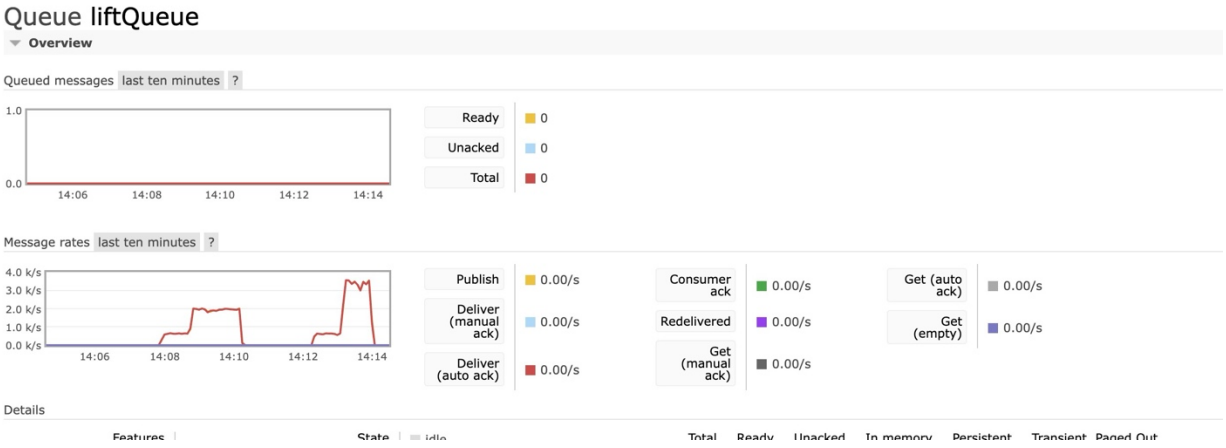


Number of Thread in Client: 100
Number of Thread in Consumer: 24
Number of basicqos: 1
Note : RPS does not change much with single server, but the connection is stable

```
/Users/monarch/Library/Java/JavaVirtualMachines/corretto-11.0.22/Contents/Home/bin/java ...
Statistic Metrics
Mean Response Time: 51.65838 ms
Median Response Time: 50.0 ms
P99 Response Time: 67 ms
Min Response Time: 35 ms
Max Response Time: 928 ms
Summary:
Number of thread in process 2: 100
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 137855
Response Time: 0.689275 ms/request
RPS: 1450 requests/second

Process finished with exit code 0
```



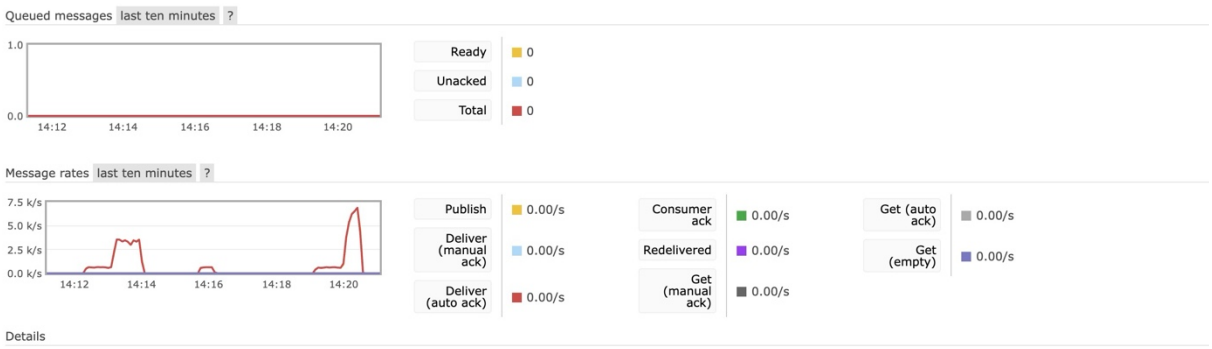
Number of Thread in Client: 200
Number of Thread in Consumer: 24
Number of basicqos: 1
Note: RPS start increased, which is higher than single server



```
/Users/monarch/Library/Java/JavaVirtualMachines/corretto-11.0.22/Contents/Home/bin/java ...
Statistic Metrics
Mean Response Time: 58.364825 ms
Median Response Time: 53.0 ms
P99 Response Time: 301 ms
Min Response Time: 35 ms
Max Response Time: 1119 ms
Summary:
Number of thread in process 2: 200
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 99311
Response Time: 0.496555 ms/request
RPS: 2013 requests/second

Process finished with exit code 0
```

Number of Thread in Client: 400
Number of Thread in Consumer: 24
Number of basicqos: 1

```
/Users/monarch/Library/Java/JavaVirtualMachines/corretto-11.0.22/Contents/Home/bin/java ...
Statistic Metrics
Mean Response Time: 67.294405 ms
Median Response Time: 57.0 ms
P99 Response Time: 324 ms
Min Response Time: 37 ms
Max Response Time: 4040 ms
Summary:
Number of thread in process 2: 400
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 79604
Response Time: 0.39802 ms/request
RPS: 2512 requests/second

Process finished with exit code 0
```

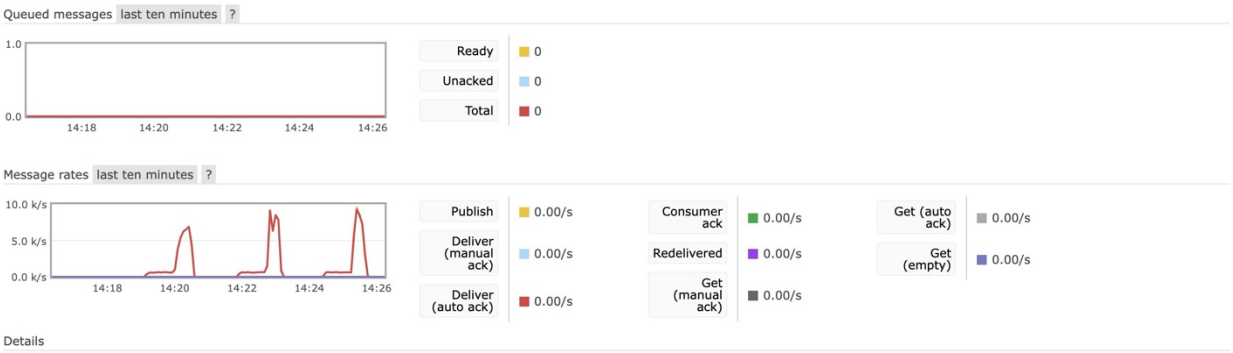


Number of Thread in Client: 800
Number of Thread in Consumer: 24
Number of basicqos: 1
Note: Performance greater than single server

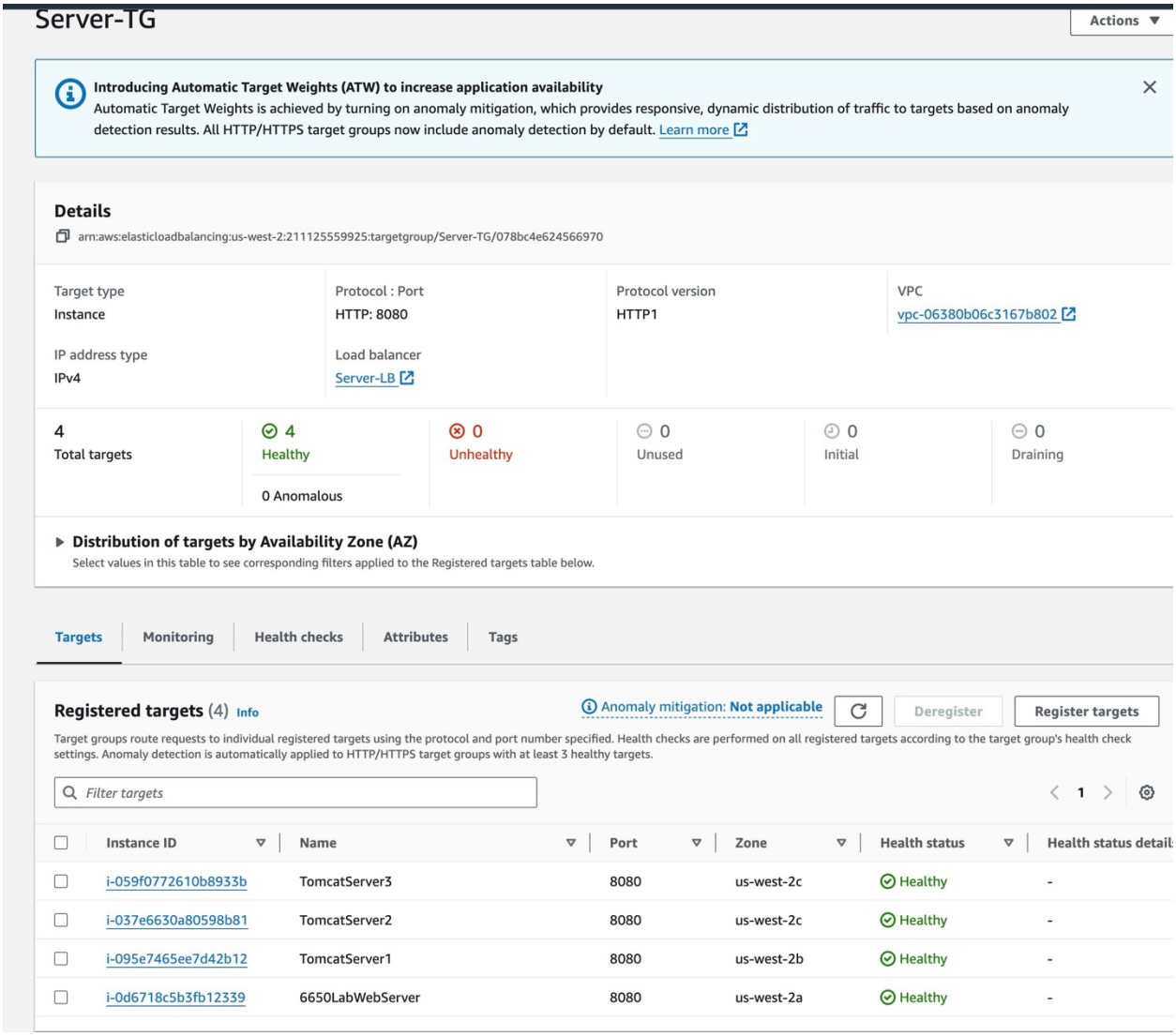

```
/Users/monarch/Library/Java/JavaVirtualMachines/corretto-11.0.22/Contents/Home/bin/java ...
Statistic Metrics
Mean Response Time: 93.51975 ms
Median Response Time: 63.0 ms
P99 Response Time: 629 ms
Min Response Time: 35 ms
Max Response Time: 4573 ms
Summary:
Number of thread in process 2: 800
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 69586
Response Time: 0.34793 ms/request
RPS: 2874 requests/second
```

Number of Thread in Client: 1200
Number of Thread in Consumer: 24
Number of basicqos: 1
Note: Starting unstable

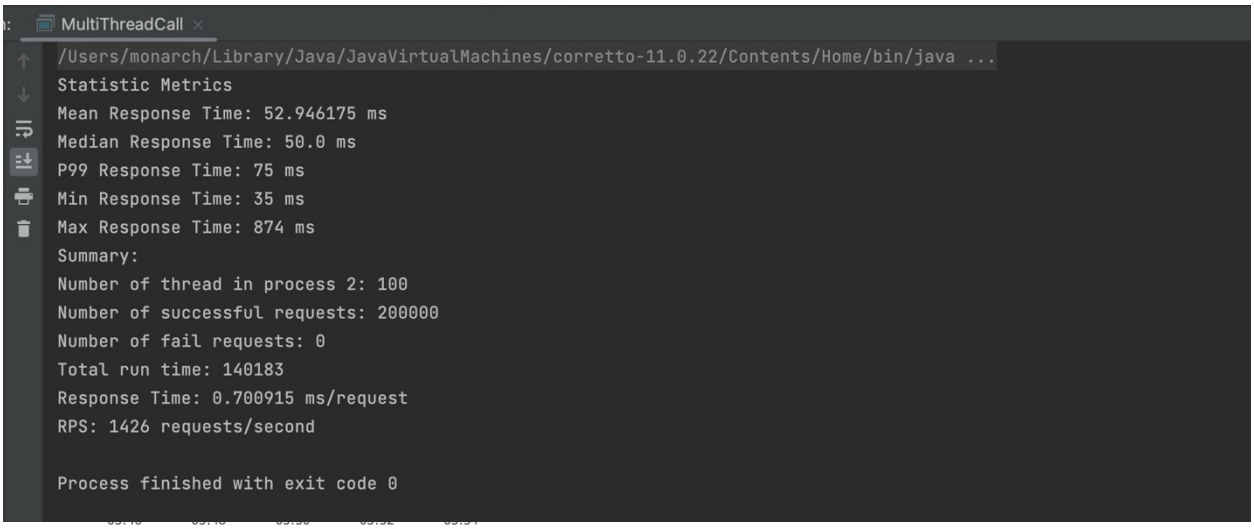
```
Statistic Metrics
Mean Response Time: 122.44801259937003 ms
Median Response Time: 65.0 ms
P99 Response Time: 1073 ms
Min Response Time: 37 ms
Max Response Time: 10434 ms
Summary:
Number of thread in process 2: 1200
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 64122
Response Time: 0.32061 ms/request
RPS: 3119 requests/second
```



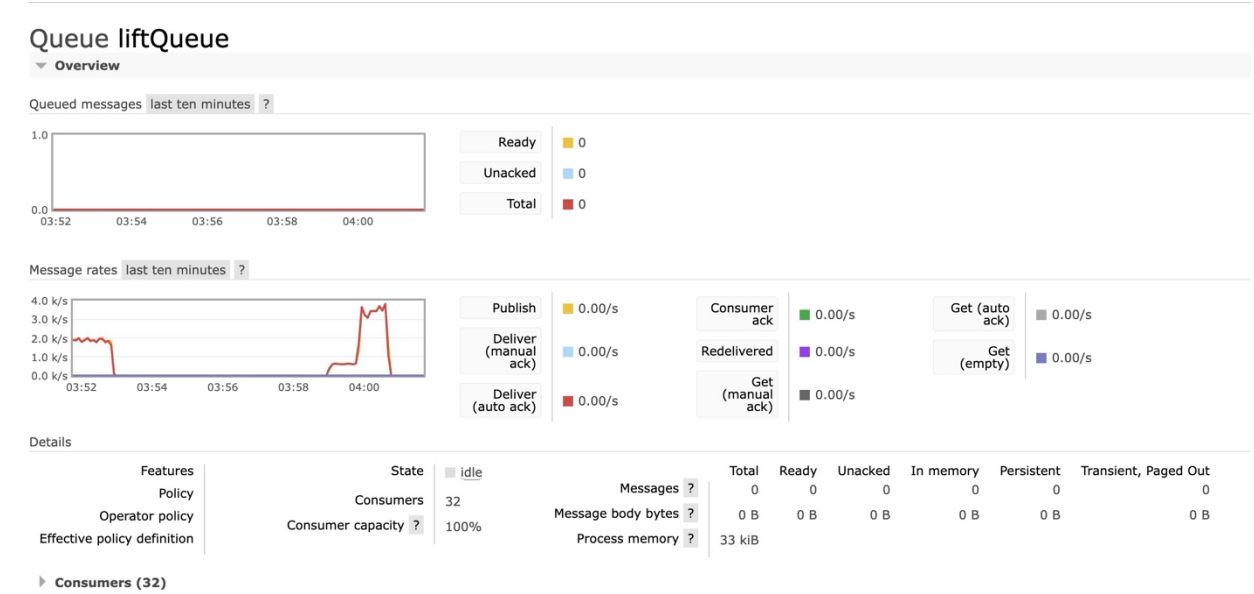
With Four Server

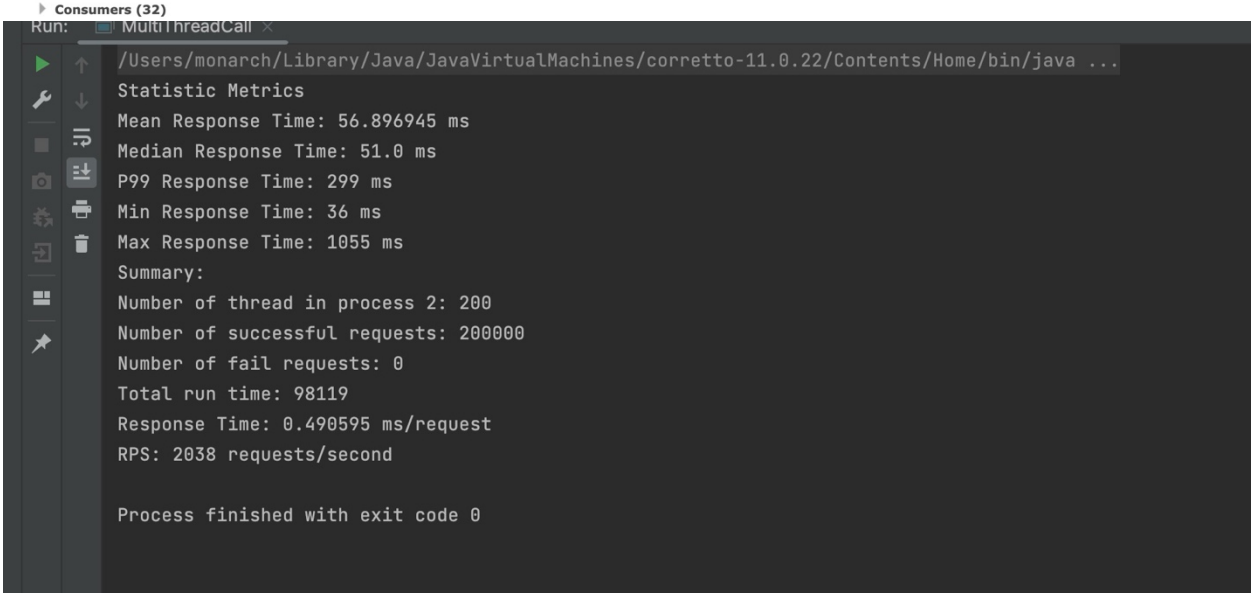


Number of Thread in Client: 100
Number of Thread in Consumer: 24
Number of basicqos: 1

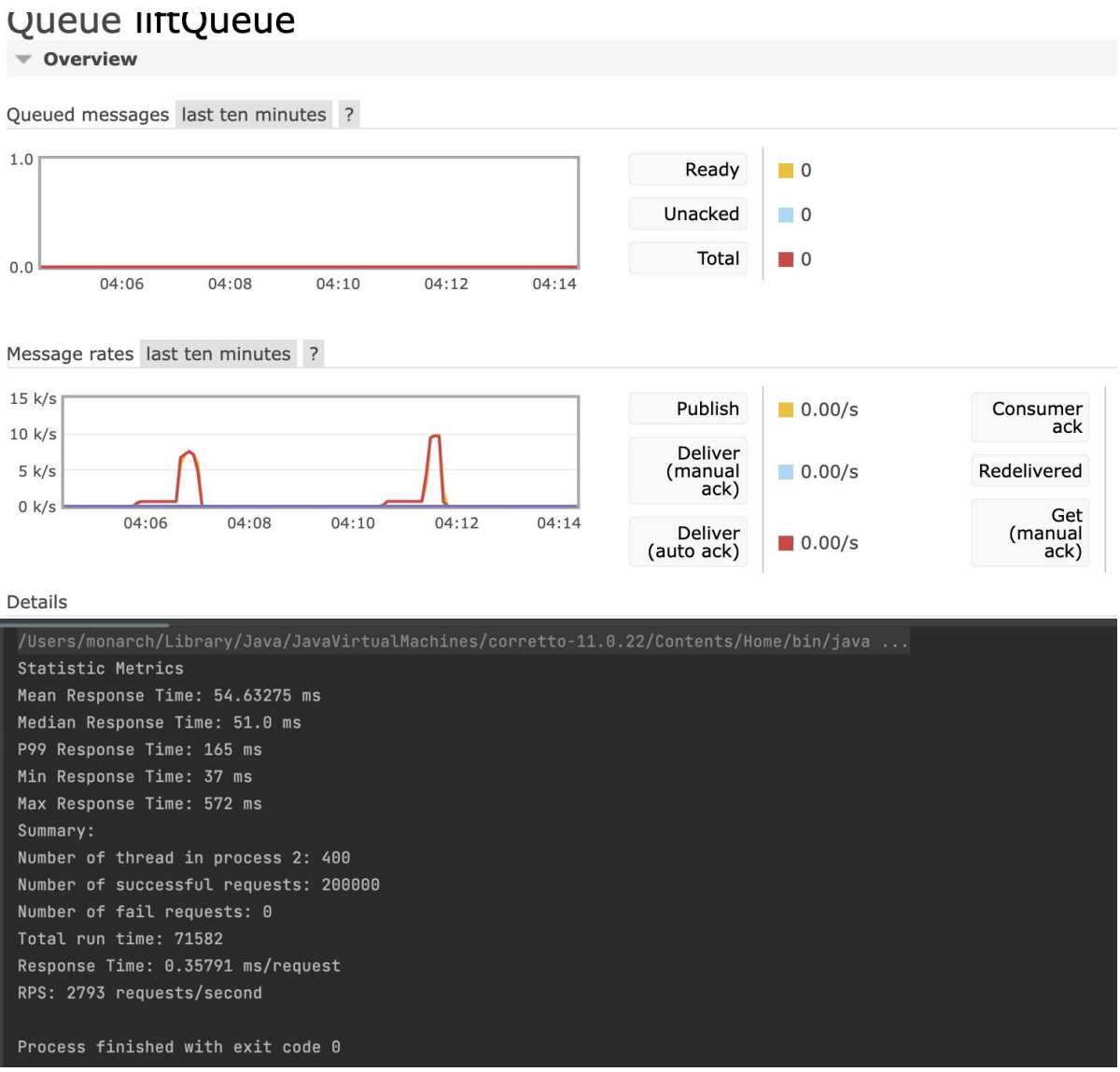


Number of Thread in Client: 200
Number of Thread in Consumer: 24
Number of basicqos: 1



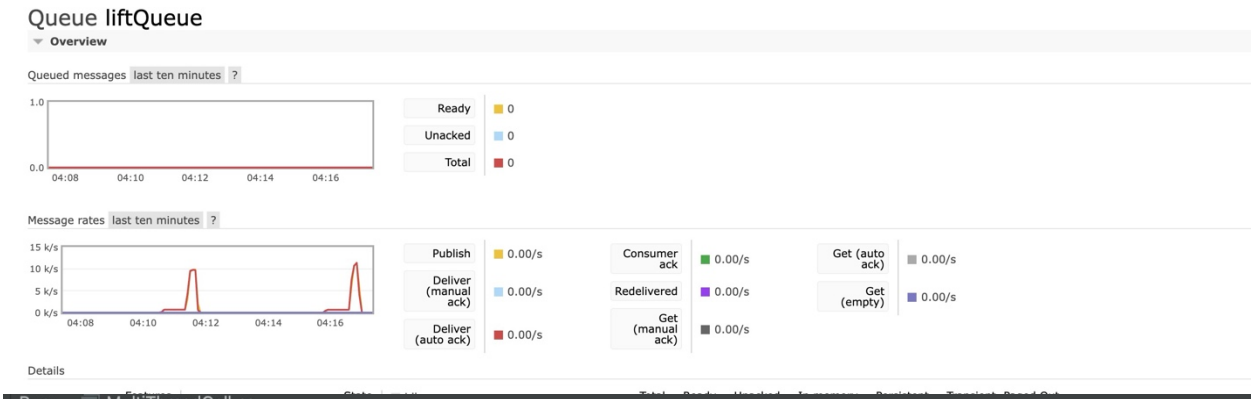


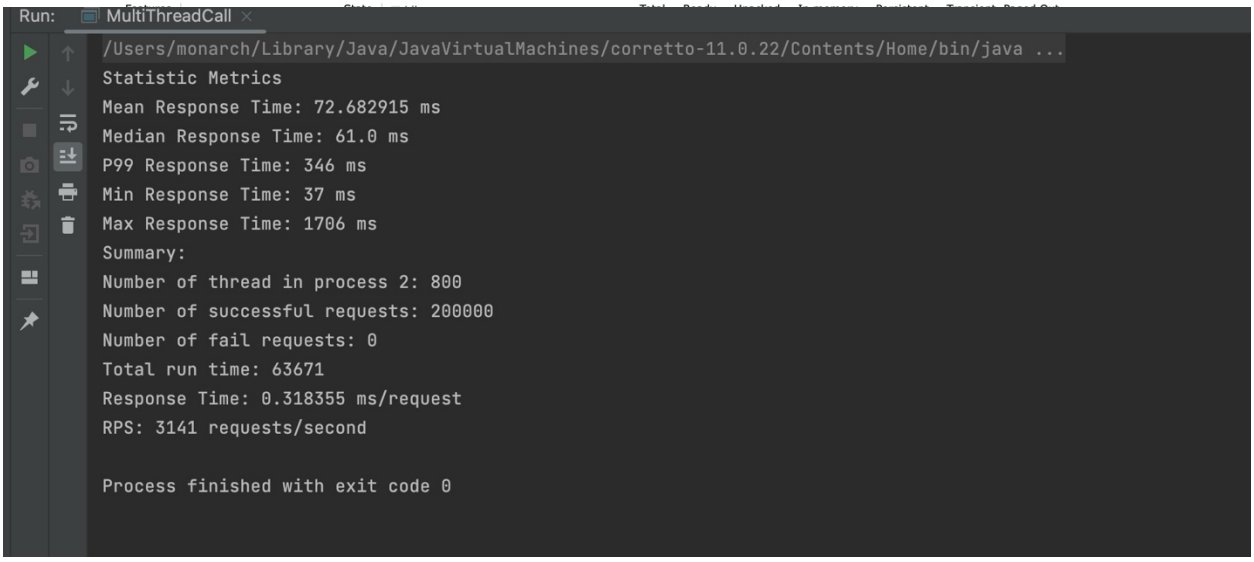
Number of Thread in Client: 400
Number of Thread in Consumer: 24
Number of basicqos: 1



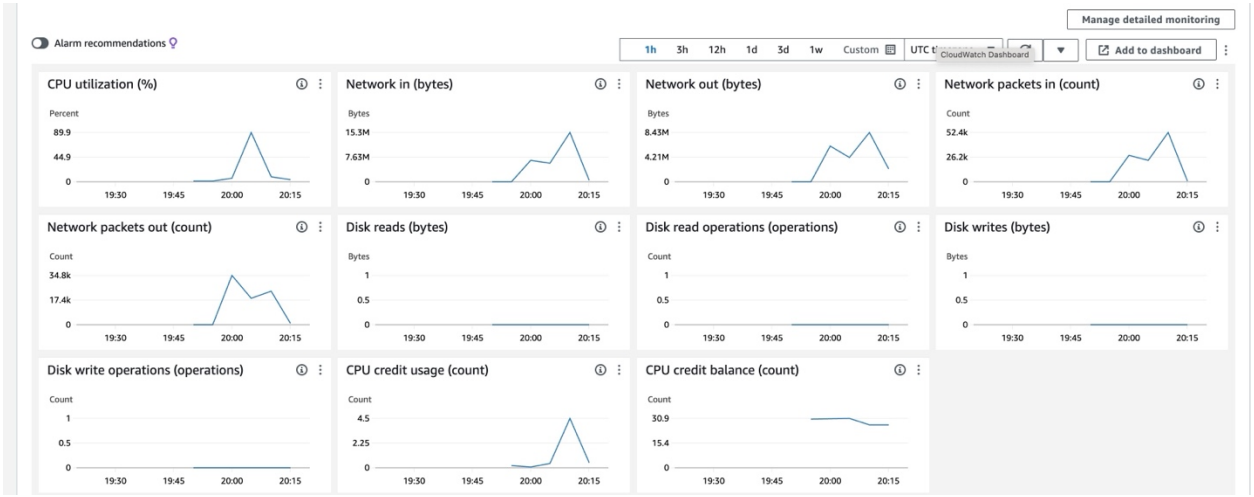

```
/Users/monarch/Library/Java/JavaVirtualMachines/corretto-11.0.22/Contents/Home/bin/java ...
Statistic Metrics
Mean Response Time: 54.63275 ms
Median Response Time: 51.0 ms
P99 Response Time: 165 ms
Min Response Time: 37 ms
Max Response Time: 572 ms
Summary:
Number of thread in process 2: 400
Number of successful requests: 200000
Number of fail requests: 0
Total run time: 71582
Response Time: 0.35791 ms/request
RPS: 2793 requests/second

Process finished with exit code 0
```

Number of Thread in Client: 800
Number of Thread in Consumer: 24
Number of basicqos: 1





CPU analyse:



There is a significant peak where utilization going up to 89.9. And following this peak, CPU utilization drops sharply back to a lower level. I think this is healthy.

RPS Plot with Different server:

Here is the data represented in a table format with all the given values:

Number of Servers	100 RPS	200 RPS	400 RPS	800 RPS	1200 RPS
Single Server	1454	1808	2326	2533	3055
Double Server	1450	2013	2512	2874	3119
Four Server	1426	2038	2793	3141	-

Conclusion:

Here's the plot showing the performance comparison between single, double, and four servers across different Thread number. Each additional server in the configuration leads to an increase in the Speed of PRS, showing that adding more servers can improve the system's ability to handle higher loads. The four-server setup, in particular, appears to offer the best performance, the data for 1200 thread is missing because the connection become unstable. However, we can see, under 800 threads, four servers have greater performance than single server, which increased 24%. [(3141 - 2533) / 2533 = 0.24].

Thanks
Zegui Jiang