Assignment 2 - Yang Chen

Git repo:

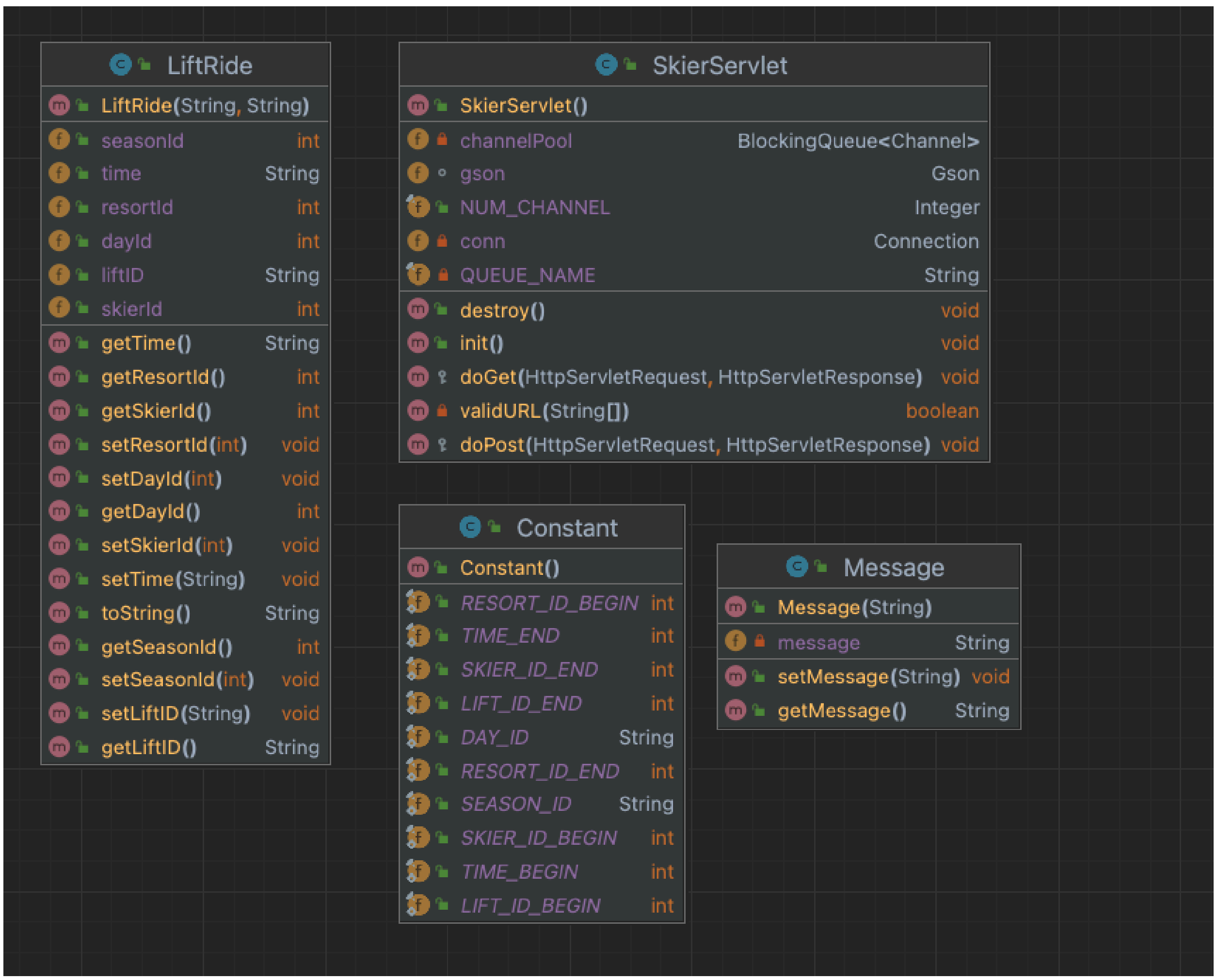
https://github.com/Yangyanggogo/SkiResortDistributedSystem/tree/hw2

A note for this report

The load tests were launched in New Mexico state instead of in Seattle. As a result, the overall test results would be less significant than other students who tested in Seattle.

Server design description

Here is my Server UML diagram, which presents the classes included in my server.



The server design is quite similar to the server design in my assignment 1, in addition to a complete validation for the URL and JSON payload.

If the server receives valid post requests, it formats the incoming data and sends it as a payload to the RabbitMQ, which is set remotely in a separate EC2 instance. To increase the efficiency of sending payloads to RabbitMQ, a connection pool was designed in my server using "BlockingQueue", with 20 channels initialized in this connection pool. When the server receives a valid post request, it will poll an available channel out of this connection pool; when finishing sending messages to remote RabbitMQ, this channel will be put back into the connection pool.

The correct connection to remote RabbitMQ was ensured in the init() method. All the information needed for this connection was stored in rabbitmq.conf file, which ensured that private information was not exposed in my code. Since my server and the RabbitMQ were located in two different EC2 instances, I used a private IPV4 address for their connections.

The detailed implementation can be found in my code, under the Server file path.

Consumer design description

The consumer was designed to pull messages off the RabbitMQ with a multi-thread program to consume as soon as the queue receives a message. It receives messages from the RabbitMQ and keeps a record of the individual lift rides for each skier in a hash map "ConcurrentHashMap", which is a good choice for multithread design.

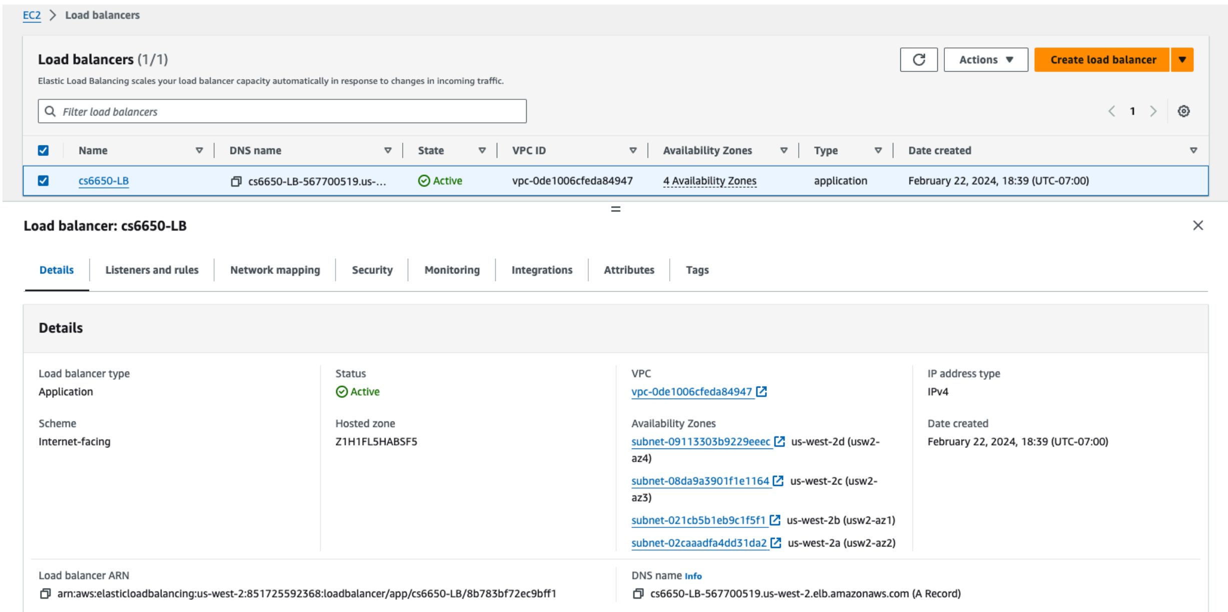
The consumer is also located in a separate EC2 and connects to RabbitMQ using the same methodology as the server.

The number of threads in the Consumer and the basicQos setting in each channel can be passed as "args" when running Consumer.jar, which is convenient for load tests.

The detailed implementation can be found in my code, under the Consumer file path.

Load balancers description

I have four instances registered in the registered targets. The setting details can be found in the figures below.
The load balancer listens to port 80, distributes requests to its target groups, and then the target groups route requests to the Tomcat server.



Client

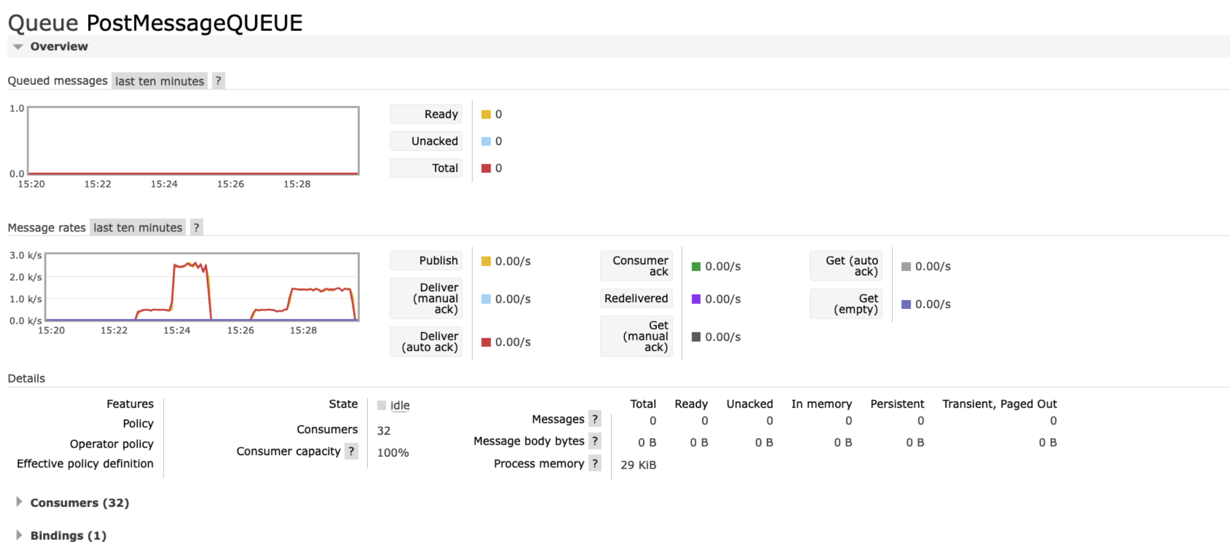
There are no changes for the client part, I used Client2 for all the load tests in this assignment.

Load test with a single servlet.

```
47 - thread received {"time":"348","liftID":"12","skierID":98486}
47 - thread received {"time":"316","liftID":"5","skierID":15547}
47 - thread received {"time":"199","liftID":"27","skierID":61928}
47 - thread received {"time":"11","liftID":"26","skierID":16781}
47 - thread received {"time":"208","liftID":"27","skierID":84733}
47 - thread received {"time":"241","liftID":"20","skierID":42516}
47 - thread received {"time":"163","liftID":"26","skierID":89254}
47 - thread received {"time":"150","liftID":"11","skierID":82488}
47 - thread received {"time":"333","liftID":"30","skierID":72341}
47 - thread received {"time":"90","liftID":"28","skierID":75089}
47 - thread received {"time":"262","liftID":"7","skierID":48230}
47 - thread received {"time":"303","liftID":"2","skierID":50248}
47 - thread received {"time":"123","liftID":"34","skierID":84948}
47 - thread received {"time":"233","liftID":"39","skierID":2248}
47 - thread received {"time":"180","liftID":"25","skierID":78160}
47 - thread received {"time":"2","liftID":"23","skierID":70969}
47 - thread received {"time":"234","liftID":"12","skierID":76252}
47 - thread received {"time":"216","liftID":"26","skierID":74475}
47 - thread received {"time":"199","liftID":"13","skierID":41385}
47 - thread received {"time":"309","liftID":"1","skierID":40722}
47 - thread received {"time":"343","liftID":"4","skierID":22252}
```

Client number of threads in phase 2: 100

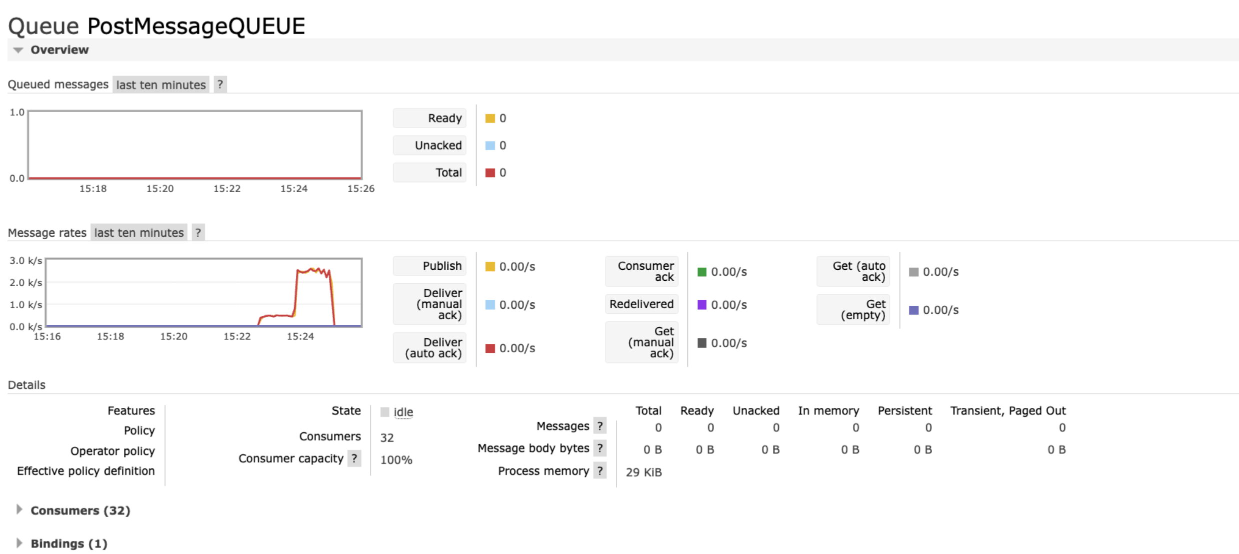
Consumer parameters:
Number of threads: 32
basicQos = 1



```
-------------------------------------
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 100
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 191298
Mean latency= 71.72291 ms
Median latency= 66.0 ms
P99 latency= 151.0 ms
Throughout: 1047 requests/second
Min latency= 52
Max latency= 638
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Client number of threads in phase 2: 200

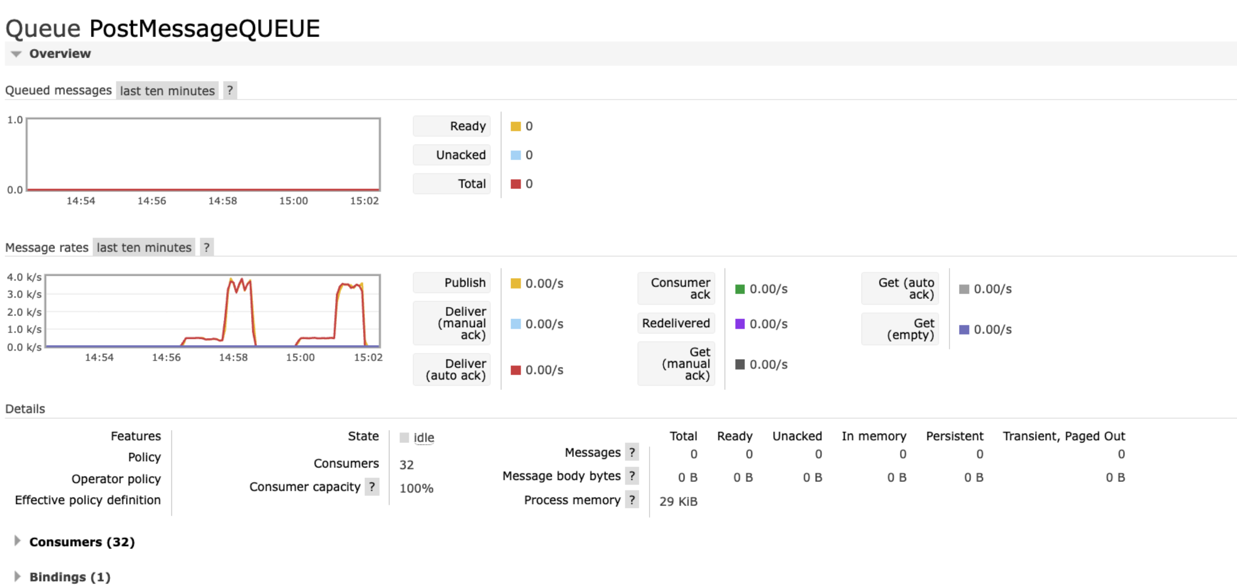
Consumer parameters:
Number of threads: 32
basicQos = 1



```
Number of threads in phase 2: 200
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 137421
Mean latency= 79.94868102637948 ms
Median latency= 72.0 ms
P99 latency= 179.0 ms
Throughout: 1459 requests/second
Min latency= 52
Max latency= 10306
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Client number of threads in phase 2: 300

Consumer parameters:
Number of threads: 32
basicQos = 1



```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 300
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 117835
Mean latency= 84.39824 ms
Median latency= 75.0 ms
P99 latency= 196.0 ms
Throughout: 1709 requests/second
Min latency= 51
Max latency= 815
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Client number of threads in phase 2: 400

Consumer parameters:
Number of threads: 32
basicQos = 1

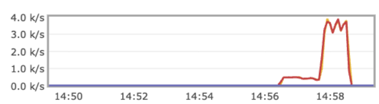
Queue PostMessageQUEUE

Overview

Queued messages last ten minutes ?

Ready	0
Unacked	0
Total	0

Message rates last ten minutes ?



Publish	0.00/s	Consumer ack	0.00/s	Get (auto ack)	0.00/s
Deliver (manual ack)	0.00/s	Redelivered	0.00/s	Get (empty)	0.00/s
Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s		

Details

Features		State	idle
Policy		Consumers	32
Operator policy		Consumer capacity ?	100%
Effective policy definition			

	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Messages ?	0	0	0	0	0	0
Message body bytes ?	0 B	0 B	0 B	0 B	0 B	0 B
Process memory ?	29 KiB					

Consumers (32)

Bindings (1)

```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 400
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 124165
Mean latency= 109.67791 ms
Median latency= 93.0 ms
P99 latency= 407.0 ms
Throughout: 1612 requests/second
Min latency= 52
Max latency= 1492
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

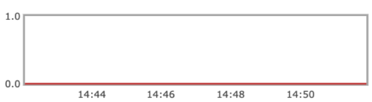
Client number of threads in phase 2: 500

Consumer parameters:
Number of threads: 32
basicQos = 1

Queue PostMessageQUEUE

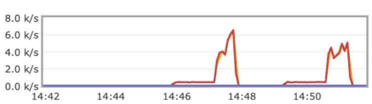
Overview

Queued messages last ten minutes ?



Ready	0
Unacked	0
Total	0

Message rates last ten minutes ?



Publish	0.00/s	Consumer ack	0.00/s	Get (auto ack)	0.00/s
Deliver (manual ack)	0.00/s	Redelivered	0.00/s	Get (empty)	0.00/s
Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s		

Details

Features	
Policy	
Operator policy	
Effective policy definition	

State	idle
Consumers	32
Consumer capacity ?	100%

	Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Messages ?	0	0	0	0	0	0
Message body bytes ?	0 B	0 B	0 B	0 B	0 B	0 B
Process memory ?	32 KiB					

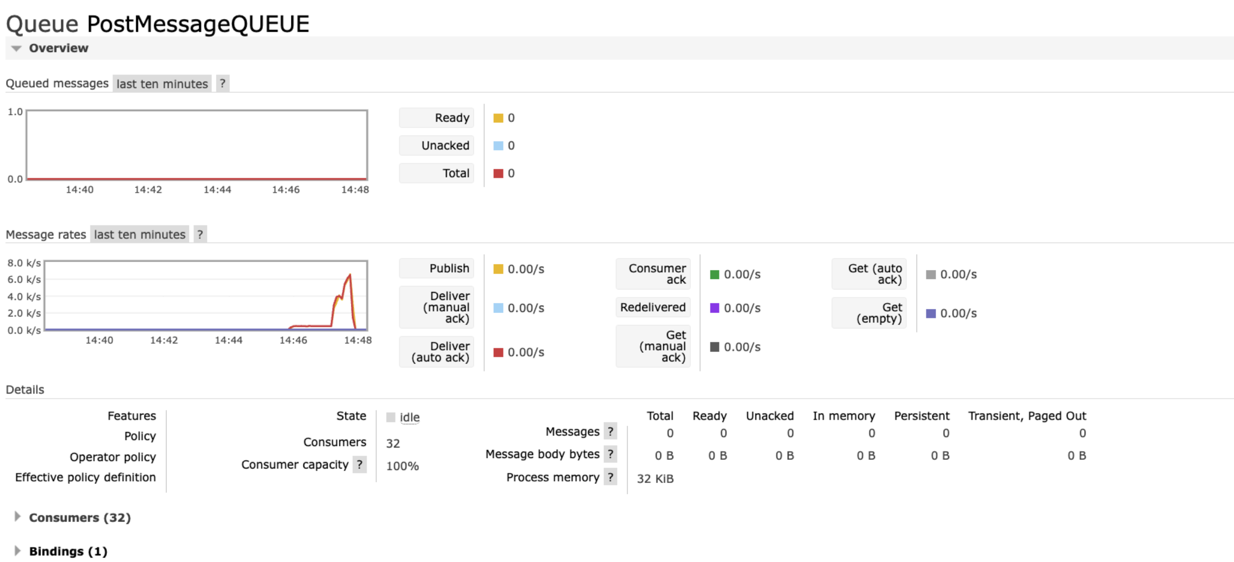
Consumers (32)

Bindings (1)

```
Number of threads in phase 2: 500
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 119452
Mean latency= 114.52774305798594 ms
Median latency= 92.0 ms
P99 latency= 443.0 ms
Throughout: 1680 requests/second
Min latency= 52
Max latency= 10474
-------------------------------------
Multi threads consumer test end
```

Client number of threads in phase 2: 800

Consumer parameters:
Number of threads: 32
basicQos = 1

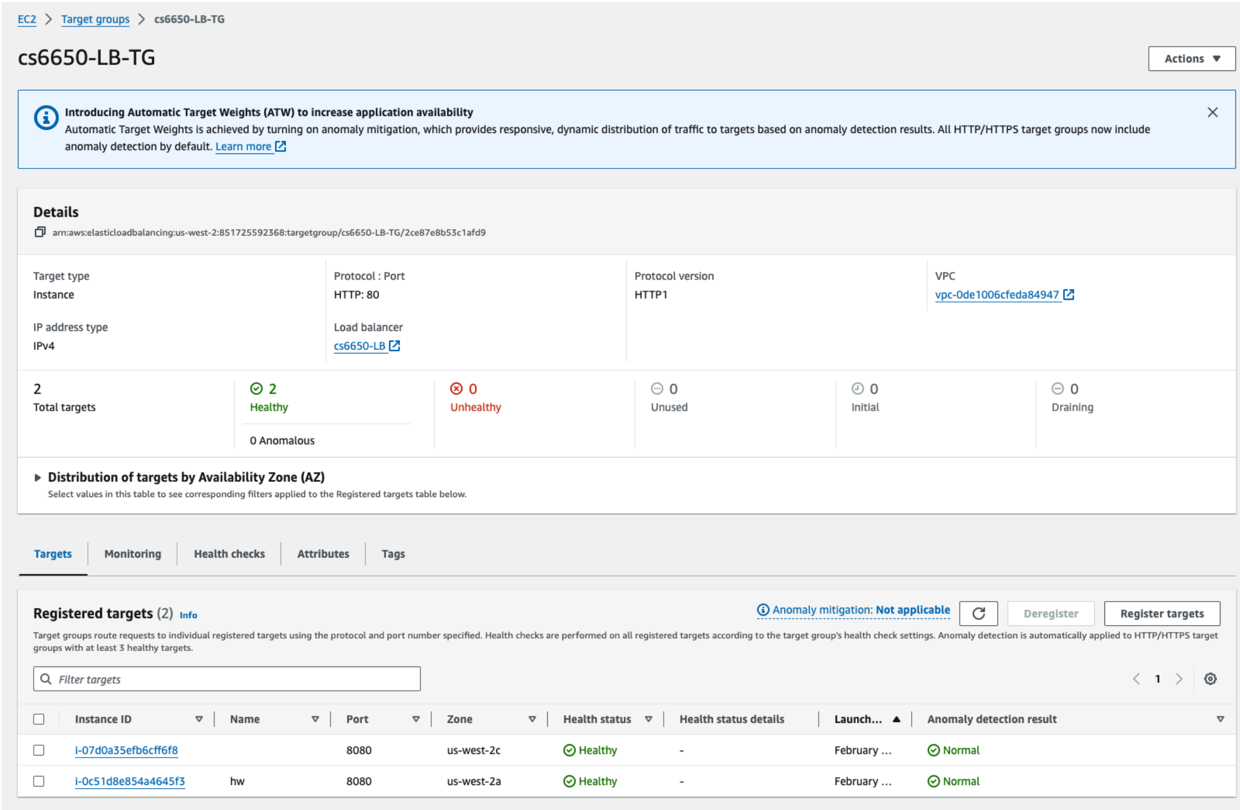


```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 800
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 115043
Mean latency= 152.26968 ms
Median latency= 122.0 ms
P99 latency= 639.0 ms
Throughout: 1739 requests/second
Min latency= 52
Max latency= 1652
-------------------------------------
Multi threads consumer test end
```

Load test with a load-balanced servlet

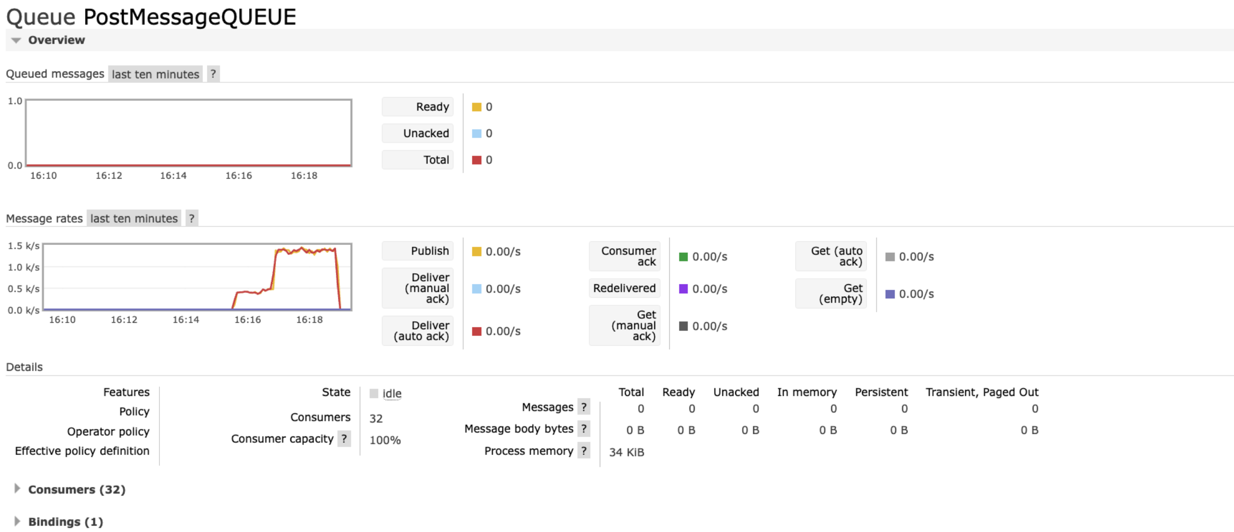
All tests are launched under this condition for Consumer:
Number of threads: 32
basicQos = 1

Load test with a load-balanced servlet - 2 instances

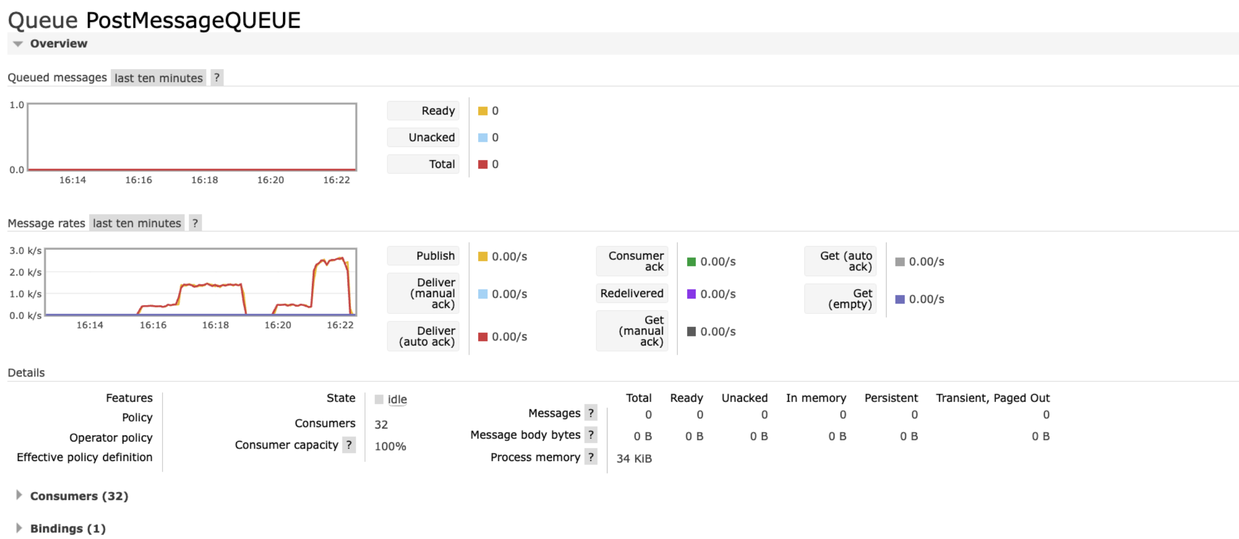


Client number of threads in phase 2: 100

```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 100
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 200496
Mean latency= 74.163965 ms
Median latency= 67.0 ms
P99 latency= 167.0 ms
Throughout: 1000 requests/second
Min latency= 52
Max latency= 650
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

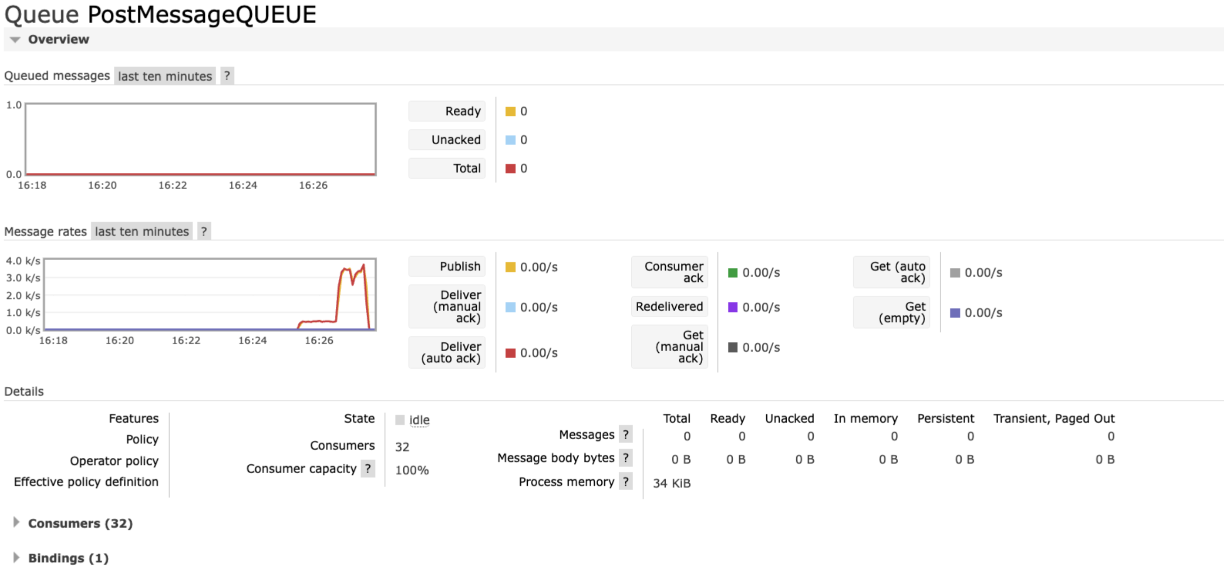


Client number of threads in phase 2: 200



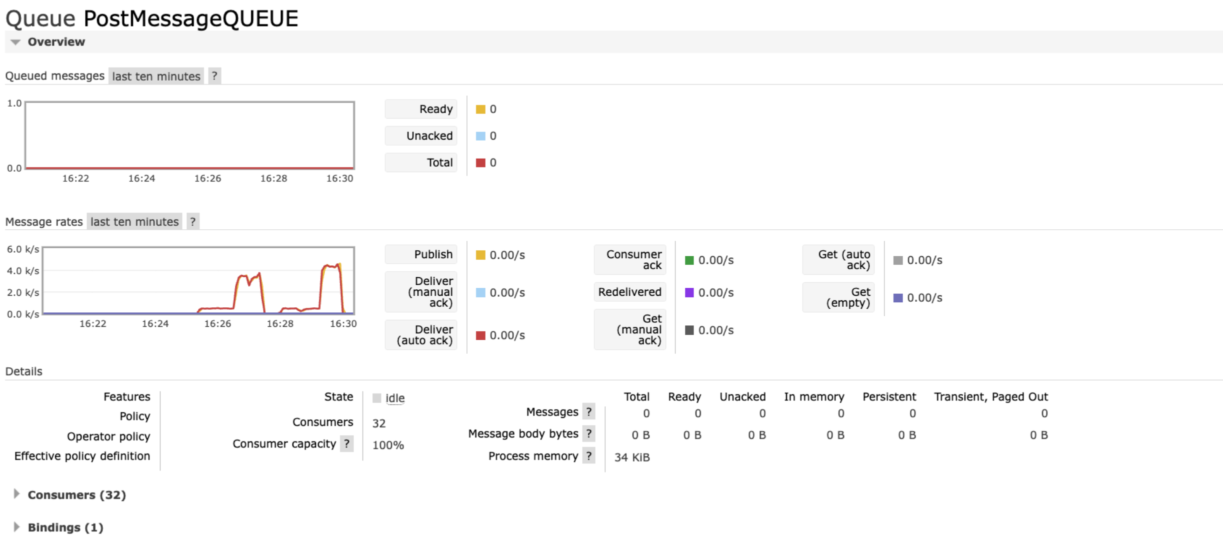

```
-------------------------------------
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 200
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 142182
Mean latency= 80.760925 ms
Median latency= 72.0 ms
P99 latency= 195.0 ms
Throughout: 1408 requests/second
Min latency= 52
Max latency= 1162
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Client number of threads in phase 2: 300



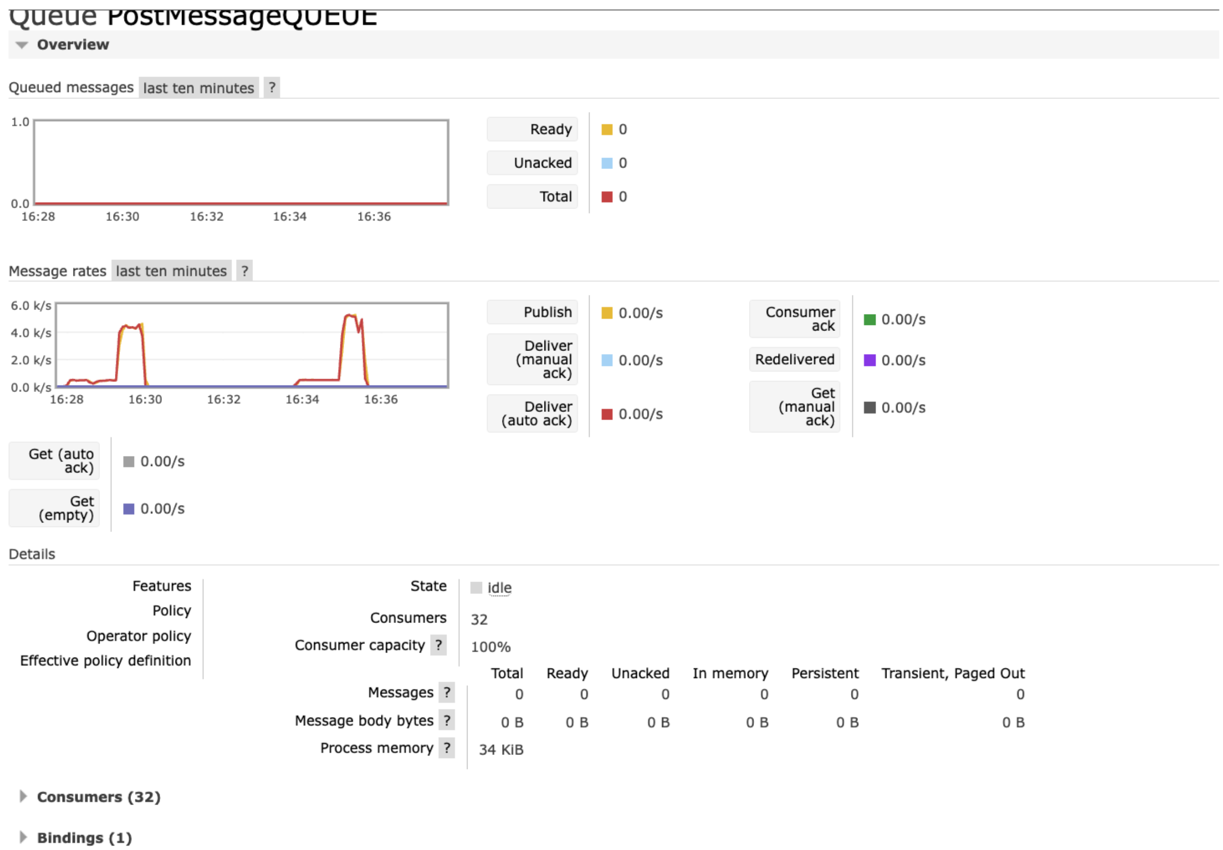

```
-------------------------------------
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 300
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 121485
Mean latency= 88.037265 ms
Median latency= 75.0 ms
P99 latency= 360.0 ms
Throughout: 1652 requests/second
Min latency= 52
Max latency= 1639
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Client number of threads in phase 2: 400



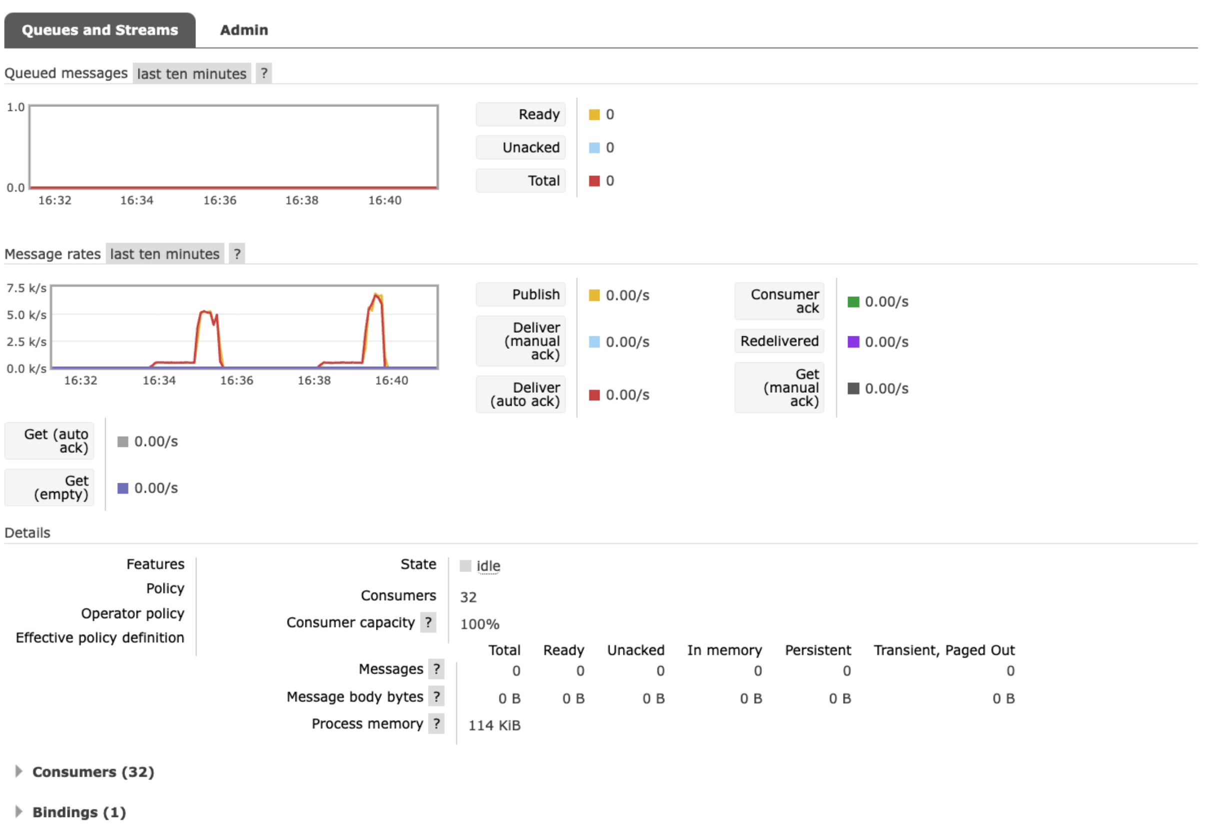

```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 400
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 115863
Mean latency= 90.453235 ms
Median latency= 79.0 ms
P99 latency= 349.0 ms
Throughout: 1739 requests/second
Min latency= 52
Max latency= 2348
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Client number of threads in phase 2: 500



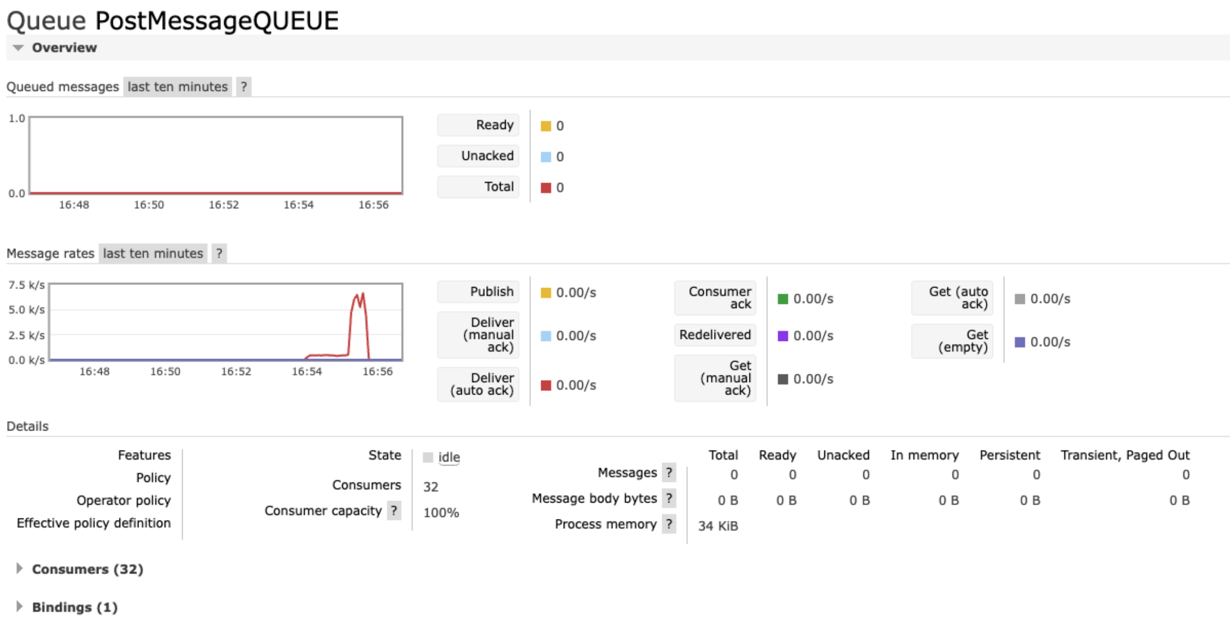

```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 500
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 103245
Mean latency= 98.39125 ms
Median latency= 84.0 ms
P99 latency= 370.0 ms
Throughout: 1941 requests/second
Min latency= 53
Max latency= 1270
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Client number of threads in phase 2: 800



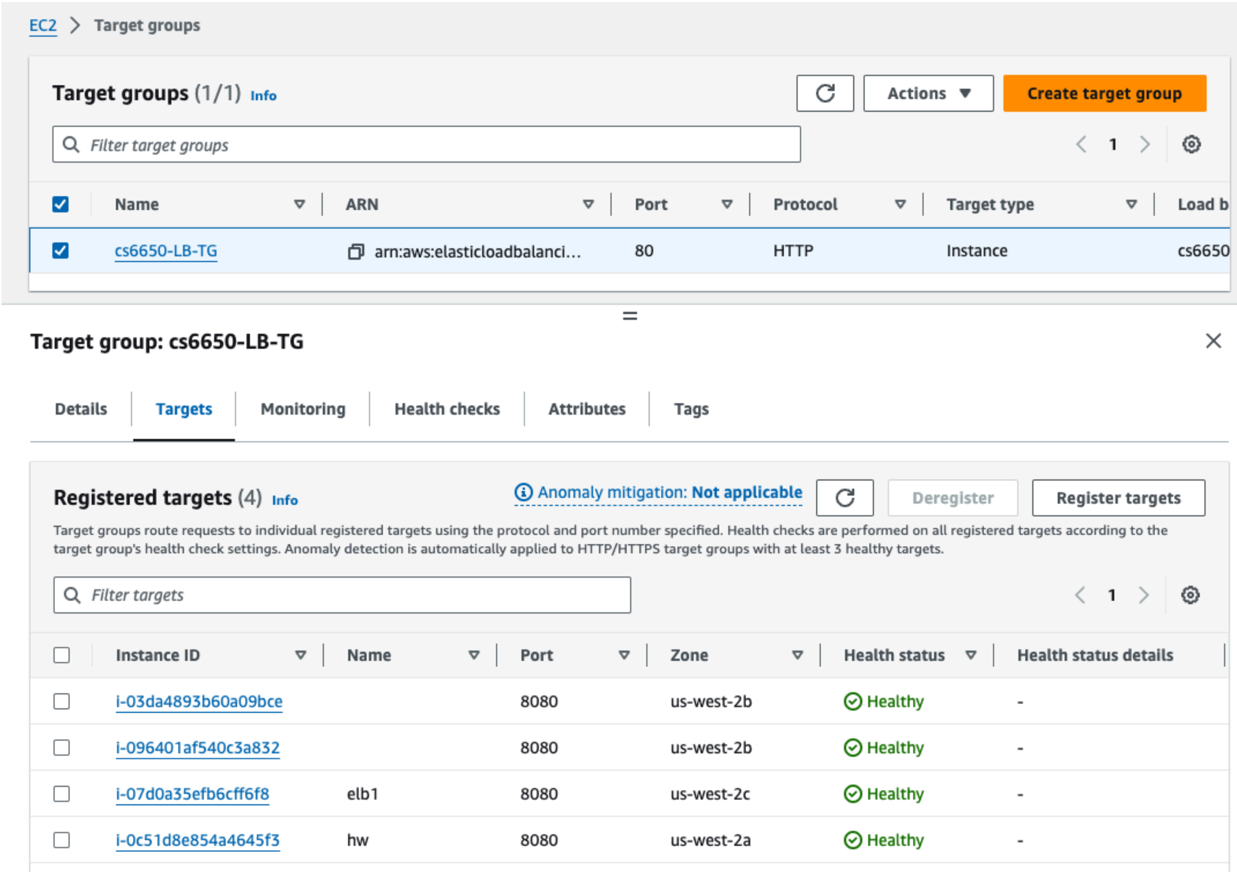

```
------------------------------------
Multi threads consumer test start
------------------------------------
Number of threads in phase 2: 800
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 97766
Mean latency= 123.22004 ms
Median latency= 100.0 ms
P99 latency= 569.0 ms
Throughout: 2061 requests/second
Min latency= 52
Max latency= 2927
------------------------------------
Multi threads consumer test end
------------------------------------
```

Client number of threads in phase 2: 1000

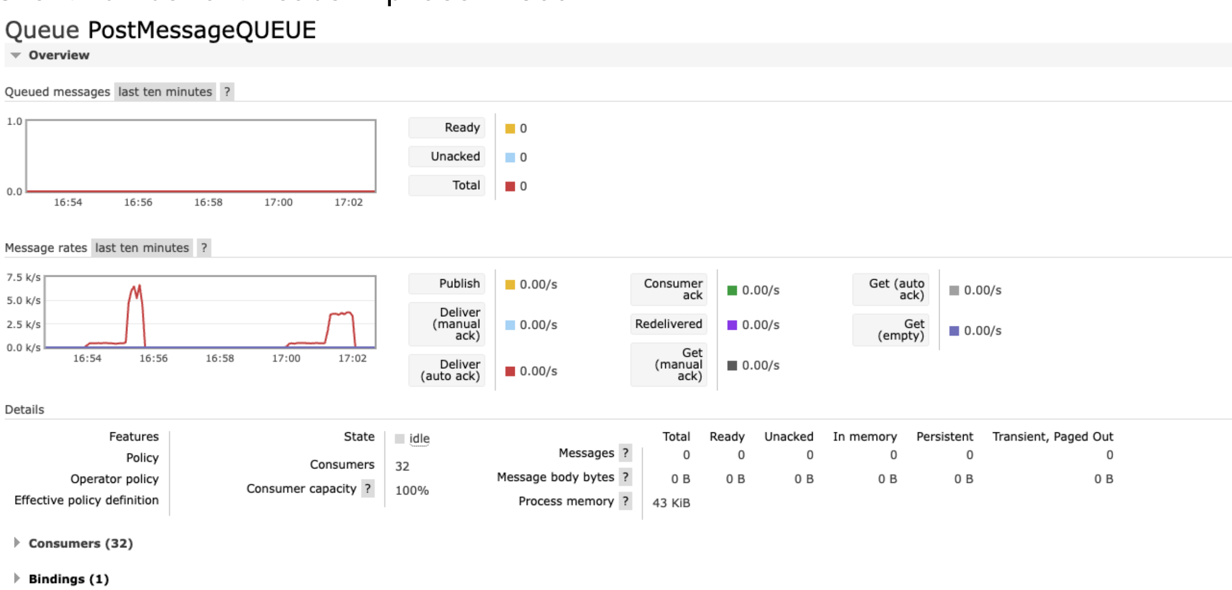



```
-------------------------------------
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 1000
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 103384
Mean latency= 157.938165 ms
Median latency= 122.0 ms
P99 latency= 794.0 ms
Throughout: 1941 requests/second
Min latency= 53
Max latency= 2973
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

Load test with a load-balanced servlet - 4 instances

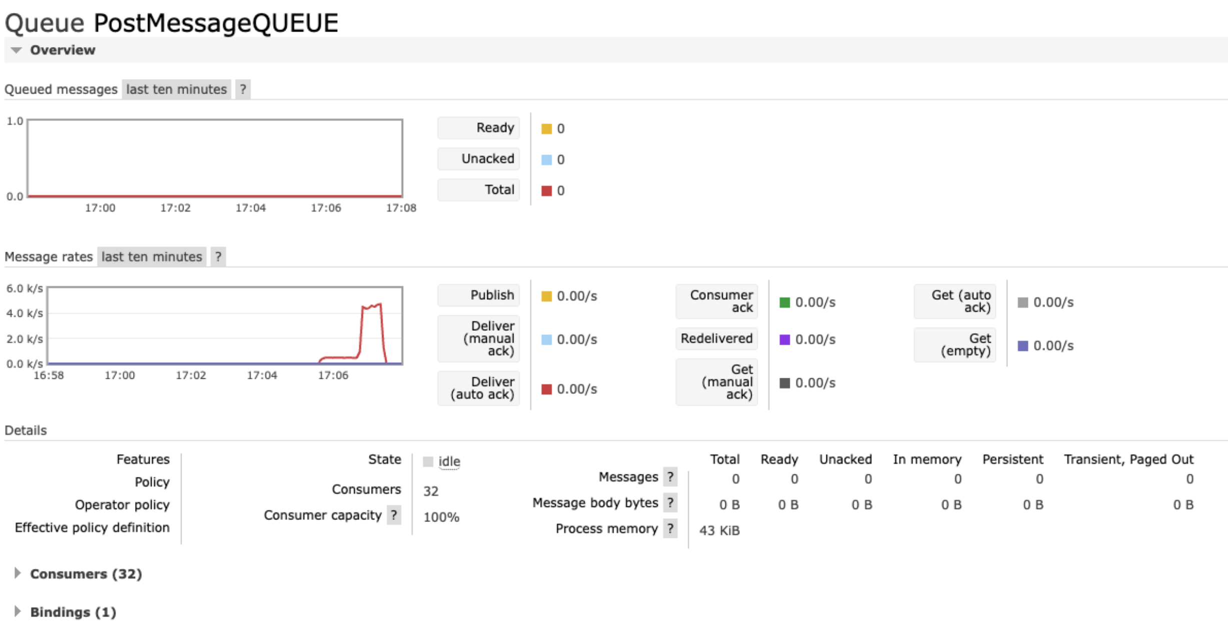


Client number of threads in phase 2: 300



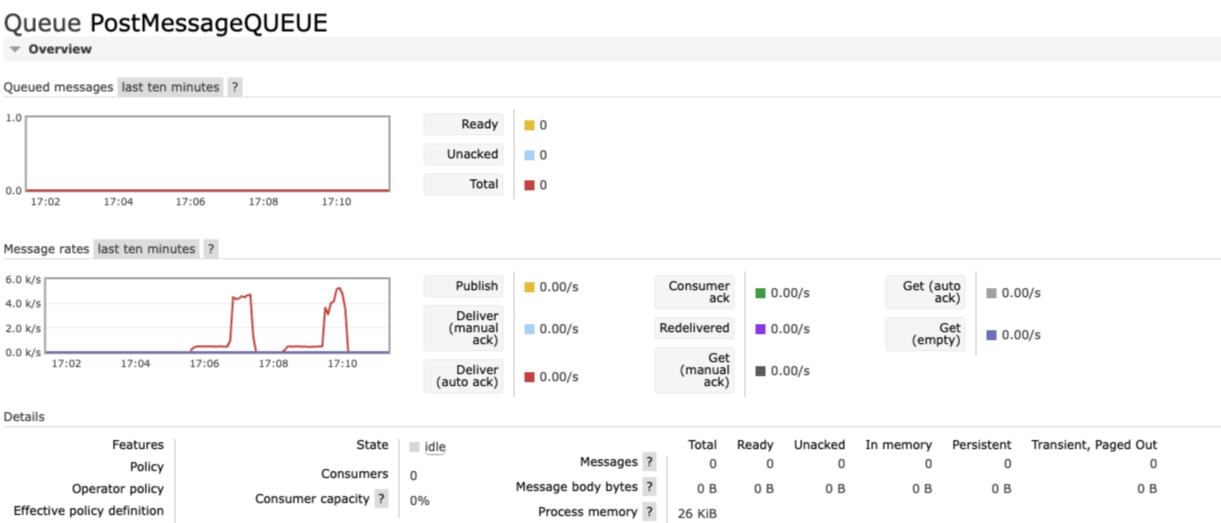
```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 300
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 121547
Mean latency= 82.717205 ms
Median latency= 74.0 ms
P99 latency= 219.0 ms
Throughout: 1652 requests/second
Min latency= 53
Max latency= 1210
-------------------------------------
Multi threads consumer test end
```

Client number of threads in phase 2: 400



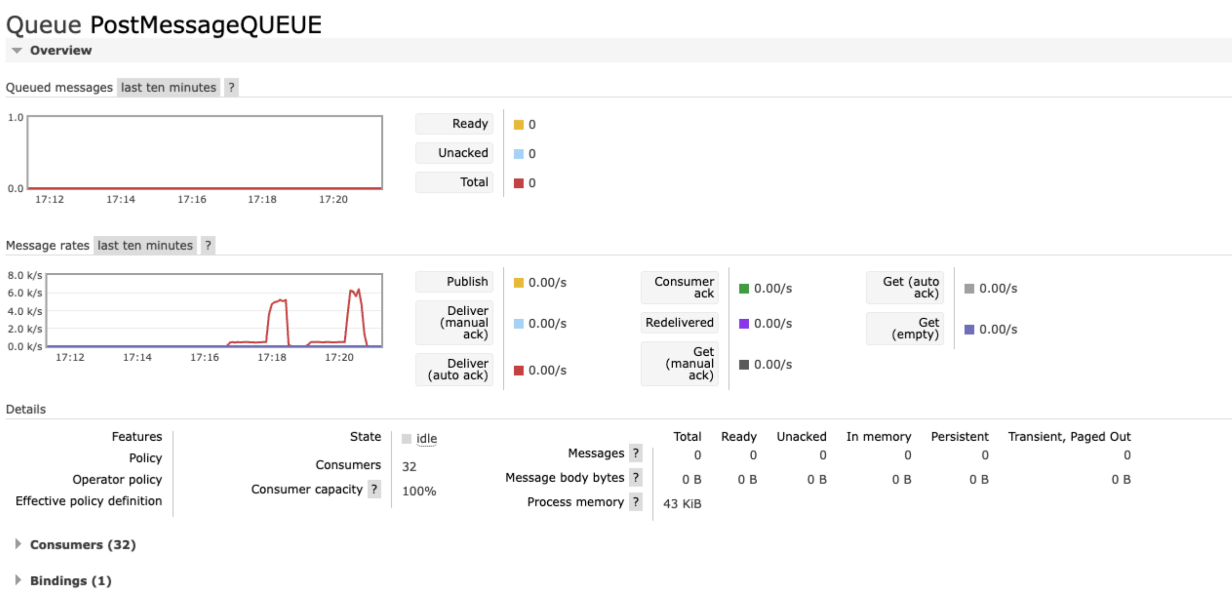
```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 400
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 105496
Mean latency= 84.971395 ms
Median latency= 76.0 ms
P99 latency= 231.0 ms
Throughout: 1904 requests/second
Min latency= 52
Max latency= 1173
-------------------------------------
Multi threads consumer test end
```

Client number of threads in phase 2: 500



```
Multi threads consumer test start
------------------------------------
Number of threads in phase 2: 500
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 106436
Mean latency= 103.18645 ms
Median latency= 88.0 ms
P99 latency= 418.0 ms
Throughout: 1886 requests/second
Min latency= 52
Max latency= 1488
------------------------------------
Multi threads consumer test end
```

Client number of threads in phase 2: 800



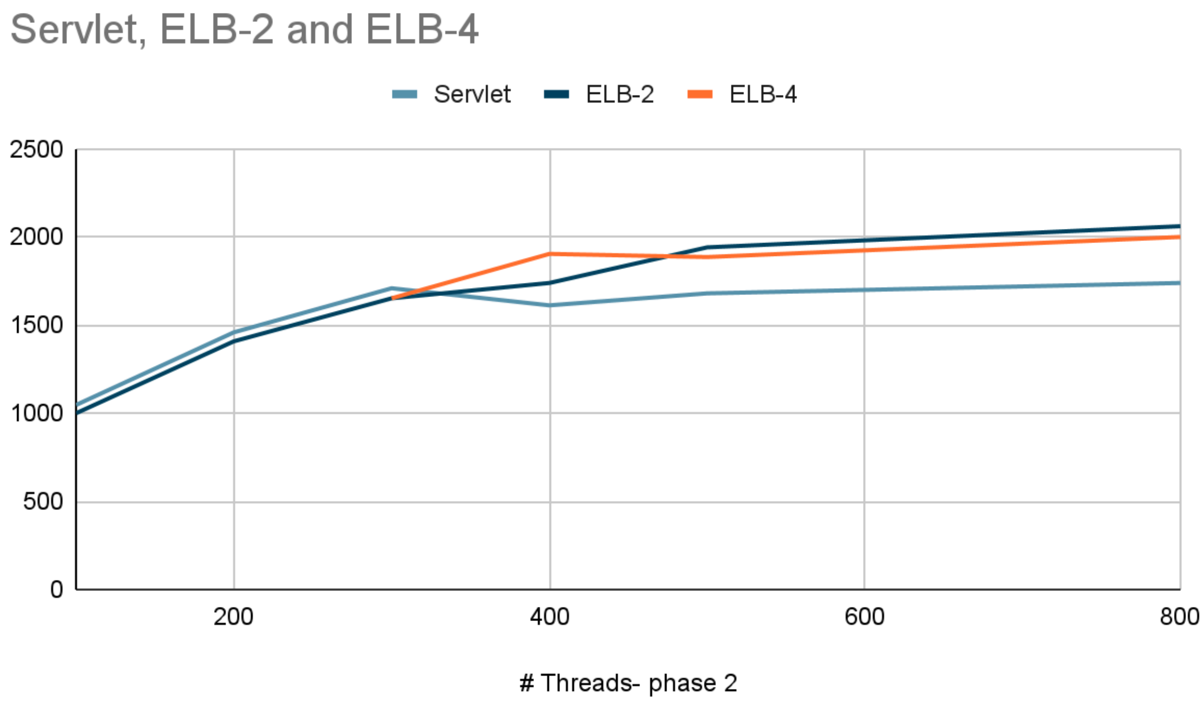
```
Multi threads consumer test start
-------------------------------------
Number of threads in phase 2: 800
Number of successful requests: 200000
Number of fail requests: 0
Wall Time: 100383
Mean latency= 129.456845 ms
Median latency= 106.0 ms
P99 latency= 533.0 ms
Throughout: 2000 requests/second
Min latency= 52
Max latency= 4040
-------------------------------------
Multi threads consumer test end
-------------------------------------
```

All the load tests done made sure that the messages in rabbitMQ were less than 1000.

Throughputs Comparison

All tests are launched under this condition for the Consumer:
Number of threads: 32
basicQos = 1

# Threads- phase 2	Servlet	ELB-2	ELB-4
100	1047	1000	-
200	1459	1408	-
300	1709	1652	1652
400	1612	1739	1904
500	1680	1941	1886
800	1739	2061	2000

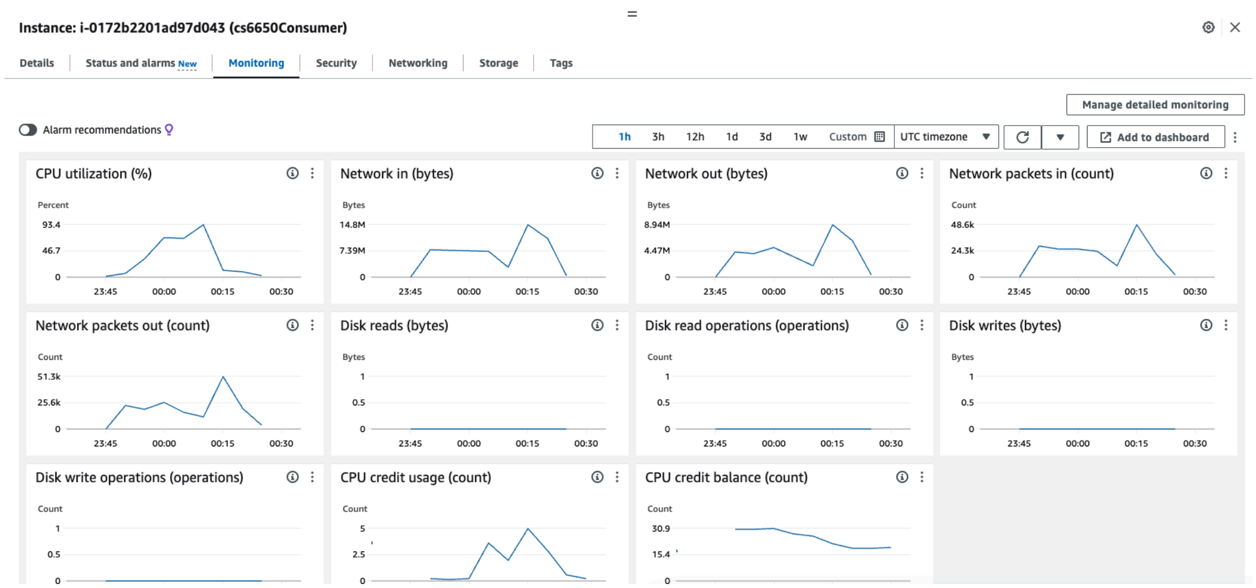


We can conclude from the figure above that using elastic load balance to distribute requests gives us greater throughputs, hence improving the performance of our system. This improvement is particularly more significant when the number of threads on client side gets bigger.

The best throughputs without ELB is around 1700; the best throughputs with ELB is around 2060.

The improvement can be calculated as 2060/1700 = 1.2.

CPU usage for the Consumer



Here we can see that the pick CPU usage of the consumer reached 93.4%, which is way beyond the 60% (normal utilization rate).