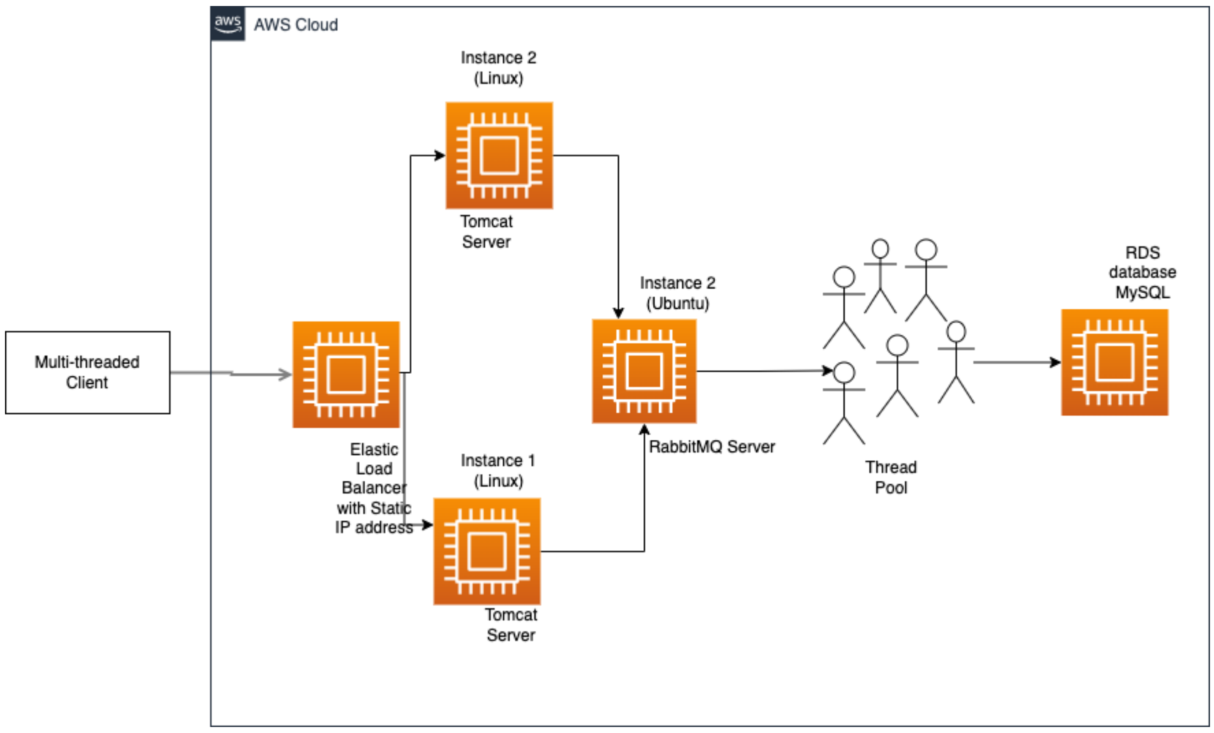
Github URL:

https://github.com/NabeelHR/CS6650-Milestone-2

Description:

The architecture

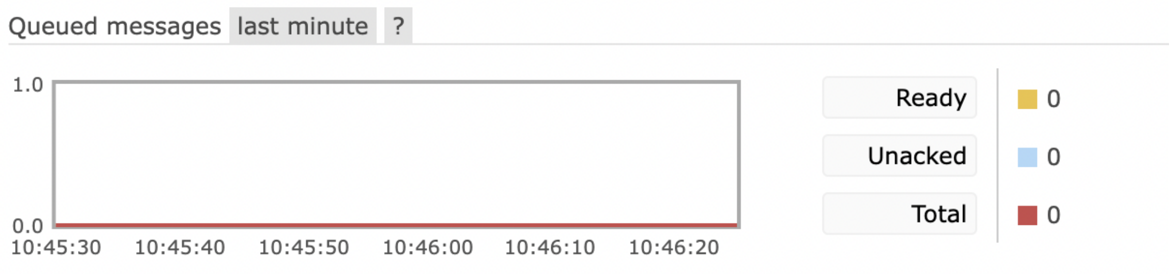


My architecture has the following main components:

1. A multi-threaded client (from lab 4) running locally that bombards the server with POST requests
2. A load balancer with a static IP address
3. 2 tomcat servers running on an EC2 instances that sends the appropriate messages on a rabbitMQ channel and responds to the client's queries
4. Another EC2 instance (Ubuntu version) that hosts the rabbitMQ server
5. A multi-threaded consumer also on EC2 that consumes the threads and commits the messages to a database (mySQL)

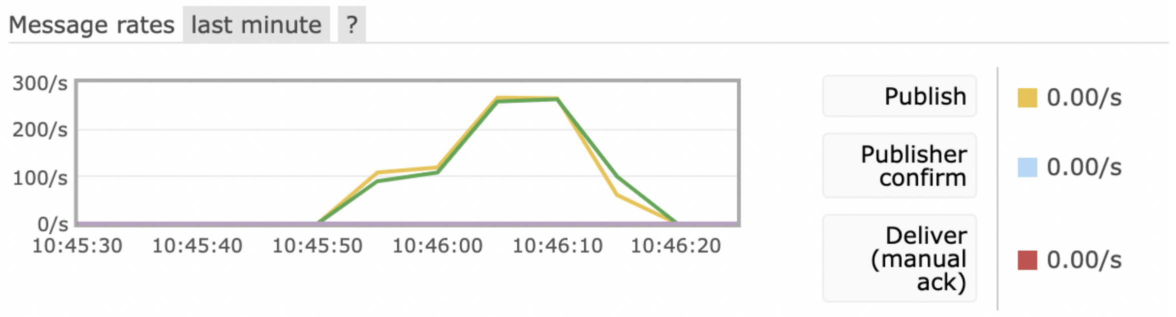
Results and Explanations

64 threads



Queued messages last minute ?

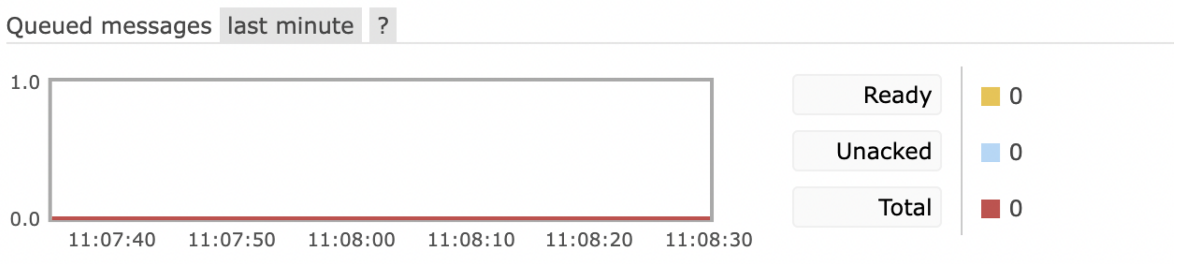
Ready	🟨 0
Unacked	🟦 0
Total	🟥 0



Message rates last minute ?

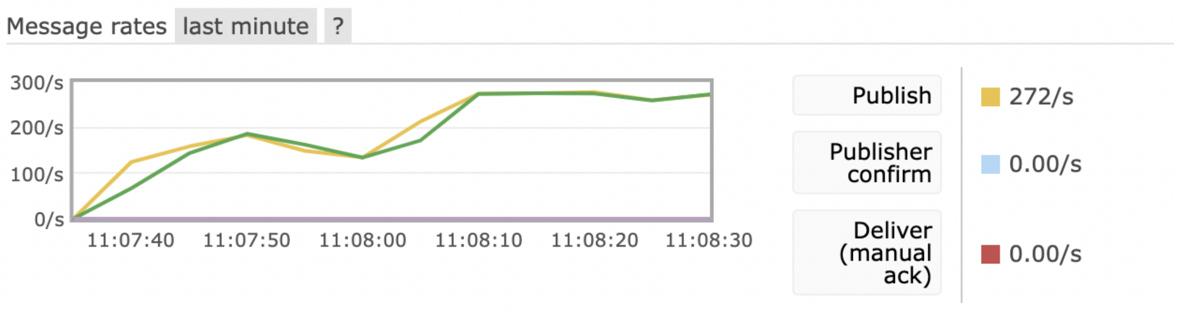
Publish	🟨 0.00/s
Publisher confirm	🟦 0.00/s
Deliver (manual ack)	🟥 0.00/s

128 threads



Queued messages last minute ?

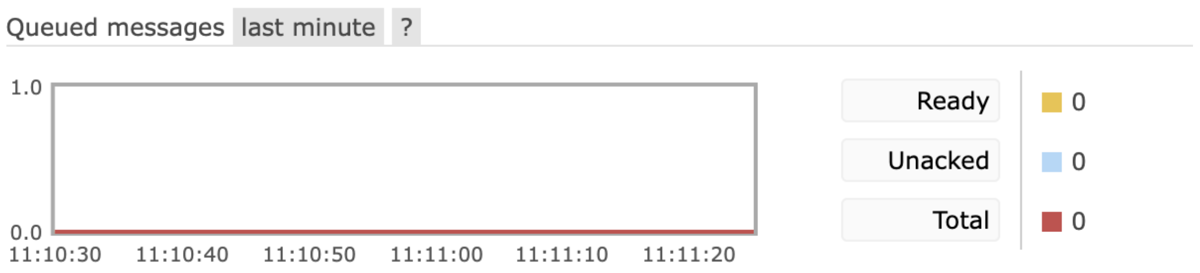
Ready	🟨 0
Unacked	🟦 0
Total	🟥 0



Message rates last minute ?

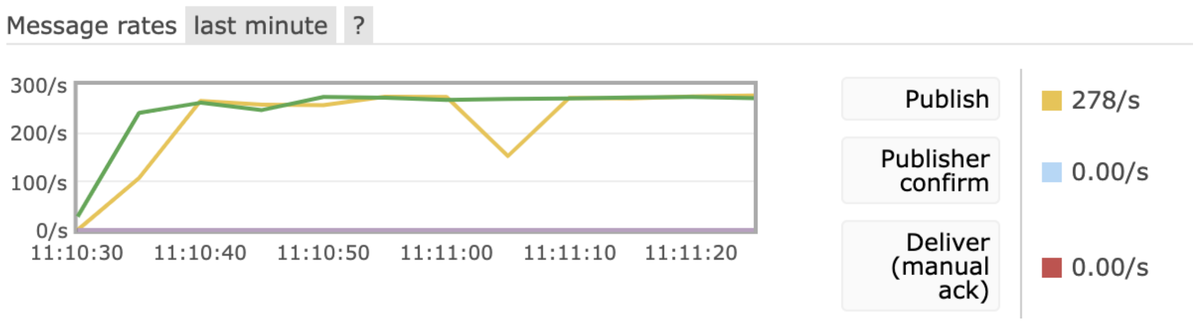
Publish	🟨 272/s
Publisher confirm	🟦 0.00/s
Deliver (manual ack)	🟥 0.00/s

256 threads



Queued messages last minute ?

Ready	🟨 0
Unacked	🟦 0
Total	🟥 0



Message rates last minute ?

Publish	🟨 278/s
Publisher confirm	🟦 0.00/s
Deliver (manual ack)	🟥 0.00/s

All clients were run with a constant number of skiers ie 1024

Key points:

- The queue never gets piled up
- Rate of messages incoming and messages ACKed is fairly constant across increasing client threads