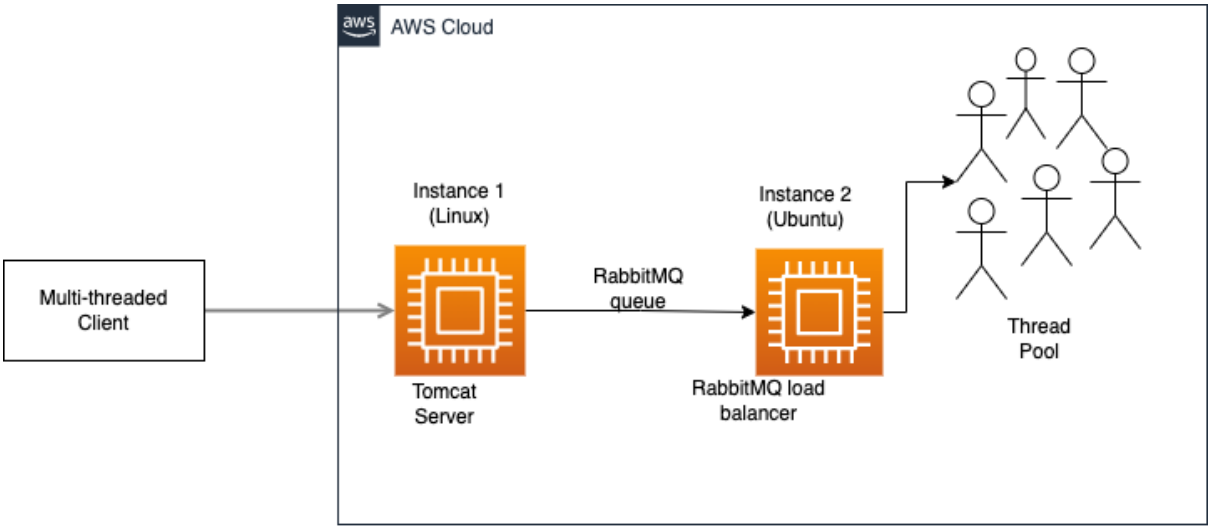
Github URL:

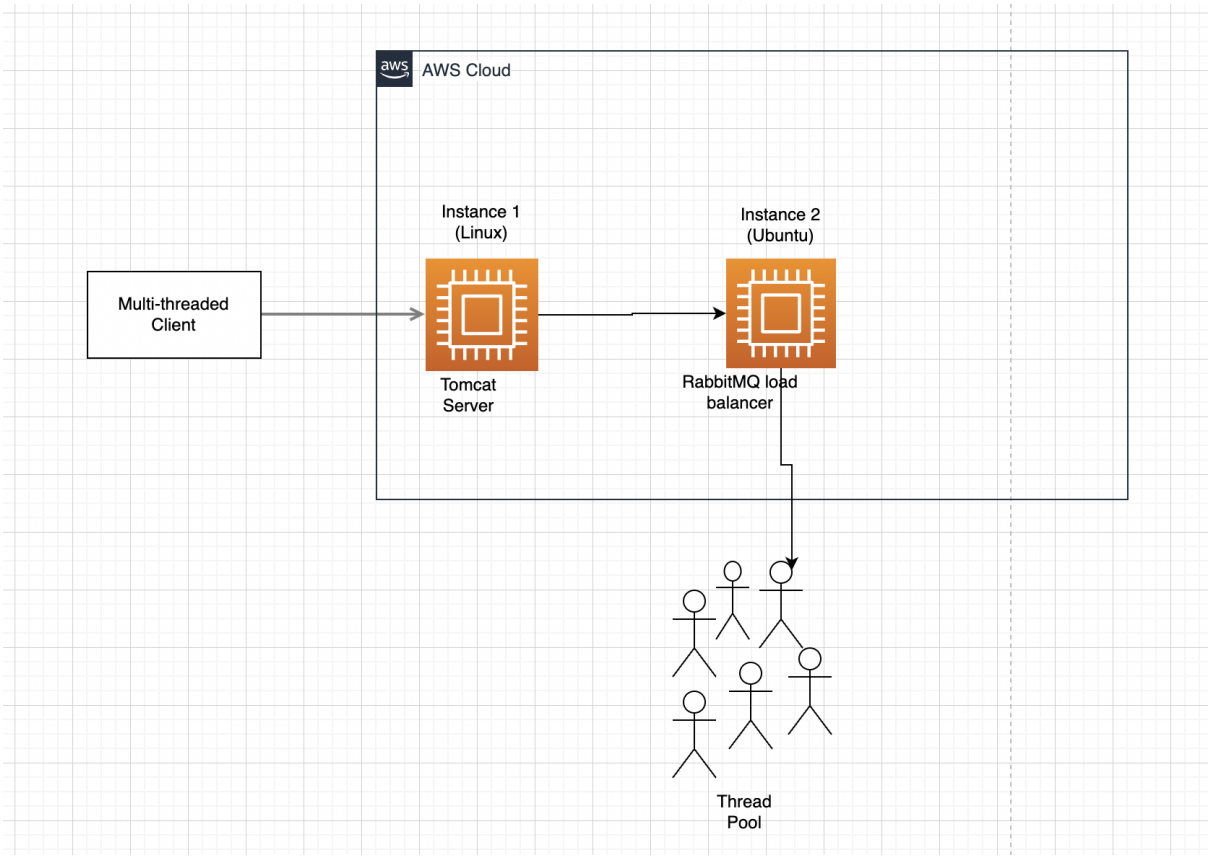
https://github.com/NabeelHR/CS-6650-Building-Scalable-Distributed-Systems

Description:

What the architecture should be



The architecture right now



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 tomcat server running on an EC2 instance that sends the appropriate messages on a rabbitMQ channel and responds to the client's queries
3. Another EC2 instance (Ubuntu version) that hosts the rabbitMQ server
4. A multi-threaded load balancer or 'thread pool' that runs locally and consumes messages from the rabbitMQ channel

Results and Explanations

The key limitation in my system right now is that the consumer is running locally and not on an EC2 instance on the cloud. This incurs latency and severely limits my server's capabilities.

With a constant number of around 2400 requests I measure the wall-time it takes my client to have all the requests addressed. My total wall time across 6 test runs stayed the same. I calibrated my consumer to have from 1 to 80 threads but the response time for the client stayed the same (2100ms) as it didn't take into account the actual time taken for all the messages to be consumed. The server would return the request once it was sent down a channel and not once the was actually consumed.

Future Work:

I need to have my consumers or worker threads also on an EC2 instance close to the tomcat and rabbitMQ server. And I need to measure & compare the time it takes for all the consumer threads to consume all the messages. I also need to use the load balancing features offered by AWS to see how it can manage the varying number of consumers for me. That would be truly making use of the 'elastic' in Elastic Cloud Computing.

With that I would be able to better instrument how scaling up works in a system that's being overloaded. It would help me explore to what extent and at what thresholds scaling up to multiple threads or consumers becomes beneficial.