

1. URL for repo:

<https://github.com/edellhou/CS6650/tree/main/Assignment2>

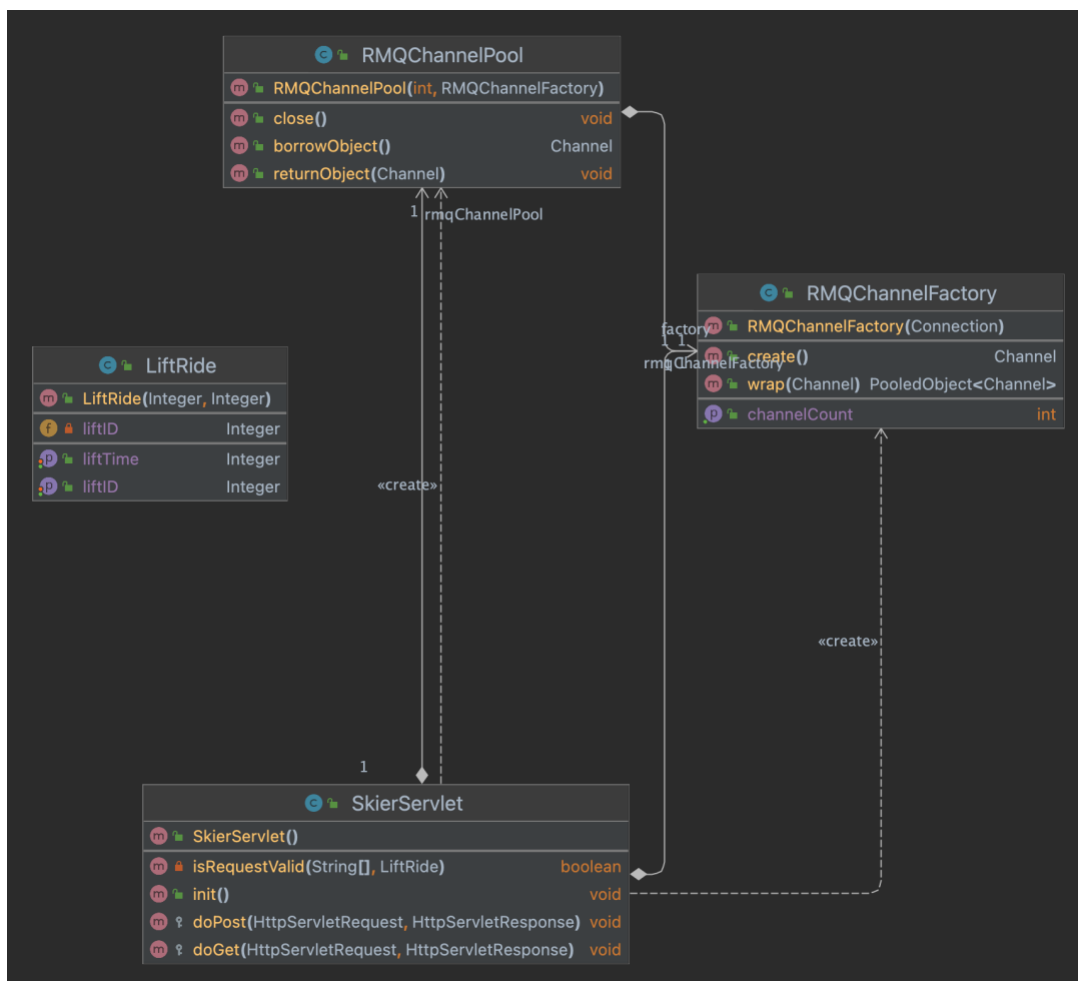
2. Server Design:

Here is the UML showing all the classes on the server side(publisher) and their relationship. The packages used are Gson and RBMQ.

LifeRide is a utility(helper) class that take the request body of liftRide and store it as LiftRide class.

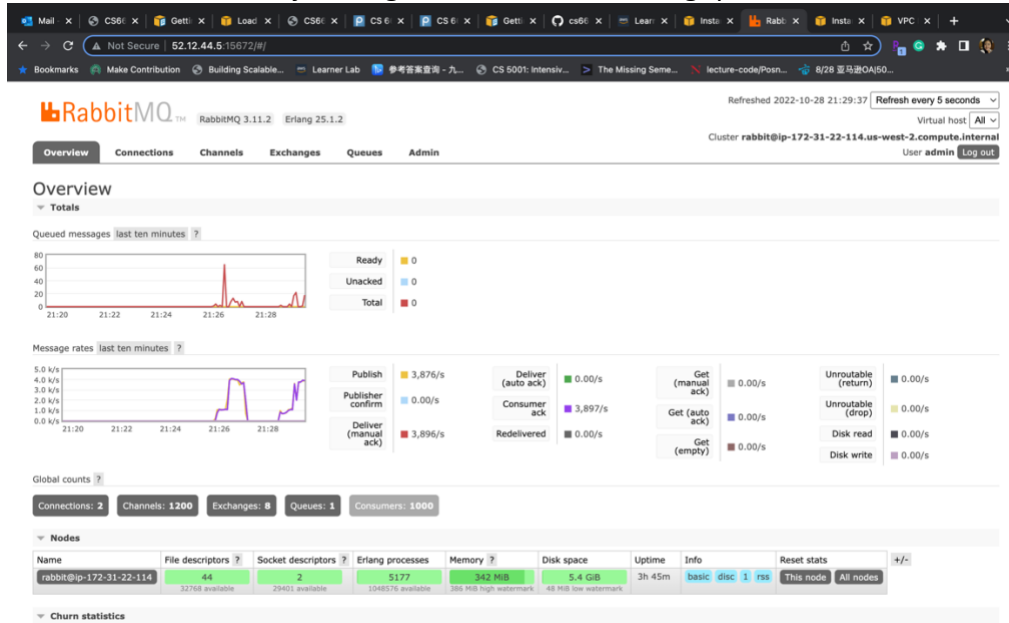
The server sent messages to a message queue use RBMQ library, the exchange method used is fanout.

In the main class SkierServlet, connection is created in the init() method, to only create time consuming connection once. The channel is created in a channel pool, and channels can be recycled to send messages to queue. The channel and channel pool utility classes are RMQChannelFactor and RMQChannelPool.



Consumer side is pretty simple, it only has one class called Recv, and it poll the message from the queue, and stored info in a Hashmap(key is the skier id, and value is a string that contain all the lift ride information). The exchange method used is fanout, aligned with publisher.

3. Here is the stats for just single instance test, throughput is 2390.



The image shows the IntelliJ IDE interface. The 'Maven' tab is open, displaying the 'dependencies' section of the 'pom.xml' file. The dependencies listed are: `<artifactId>javax.servlet-api</artifactId>`, `<version>4.0.1</version>`, `<scope>provided</scope>`, and `<dependency>` with `<groupId>io.swagger</groupId>`. The 'Run' tab is also open, showing the output of the 'PartTwo.Main' test. The output includes: 'finished producing', 'Single request mean response time in millisecond: 45', 'Single request Median response time in millisecond: 40', 'Single request p99 response time in millisecond: 120', 'Single request Max response time in millisecond: 1180', 'total successful requests: 200000', 'total failed requests: 0', 'total time used in millisecond: 83672', 'total throughput in requests per second: 2390.2858781910313', and 'Process finished with exit code 0'.

4. Here is the stats for load balanced test, throughput increased to 2745

The screenshot displays the RabbitMQ Overview page in a web browser, showing system statistics and a Java IDE with test results.

RabbitMQ Overview Page:

- Overview:** Shows a line graph for 'Queued messages' and 'Message rates' over the last ten minutes. The 'Message rates' graph shows a peak around 22:14.
- Totals:** Ready: 0, Unacked: 1, Total: 1.
- Message rates:** Publish: 4,361/s, Publisher confirm: 0.00/s, Deliver (manual ack): 4,322/s, Deliver (auto ack): 0.00/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: 3, Channels: 1400, Exchanges: 8, Queues: 1, Consumers: 1000.
- Nodes:** rabbit@ip-172-31-22-114. File descriptors: 44 (32768 available), Socket descriptors: 3 (29401 available), Erlang processes: 5982 (1048576 available), Memory: 339 MiB (386 MiB high watermark, 48 MiB low watermark), Disk space: 5.4 GiB (48 MiB low watermark), Uptime: 31m 8s, Info: basic, disc, 1, rss. Reset stats: This node, All nodes.
- Churn statistics:** (Empty)

IDE Screenshot:

- Project:** Upic - PostRunner.java, Consumer - Recv.java, UpicServlet - SkierServlet.java.
- Code:** Shows the `PostRunner` class with methods `numRequest`, `localBasePath`, `remoteBasePath`, and `PostRunner`.
- Run:** Shows the output of the `PartTwo.Main` class, including statistics for single request response times, total successful requests (200000), total failed requests (0), total time used (72848 ms), and total throughput (2745.442565341533 requests per second).

