distributed-systems-work

Assignments Repo for CS6650 Building Scalable Distributed Systems 23fall@neu

Ayan Mao 2023 Dec

https://gortonator.github.io/bsds-6650/assignments-2022/Assignment-1

https://gortonator.github.io/bsds-6650/assignments-2022/Assignment-2

https://gortonator.github.io/bsds-6650/assignments-2022/Assignment-3

Github link: https://github.com/98may/distributed-systems-work/

1: Overview

1.1: milestones

1) update Java server to support album review - like and dislike

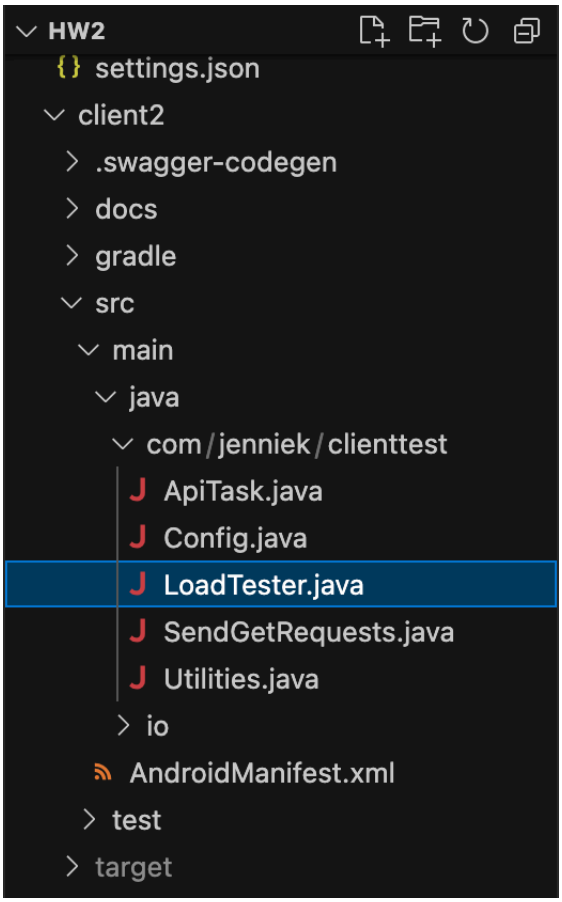
2) update client to send posts (4 post per thread = 1 new album + 2 like + 1 dislike)

3) baseline tests without MQ

4) update server with RabbitMQ

5) real tests with RabbitMQ

1.2: client code design



loadTester.java

loadTester is the main java class to handle tests,
`parse` to set up args from `Usage: LoadTester <threadGroupSize> <numThreadGroups> <delay> [java|go]`,

and then `mainLoad` send out the major requests.

```
    public static void main(String[] args) {
        LoadTester loadTester = new LoadTester();
        loadTester.parse(args);
        loadTester.mainLoad();
    }
```

ApiTask.java

implements `runnable`

as a single thread unit to send 4 post requests:

// 4 requests per thread: POST a new album and image + POST two likes and one dislike for the album. (4 = 1 album + 2 like + 1dislike)

Utilities.java

mainly the Utilitiy functions for logging and numbers

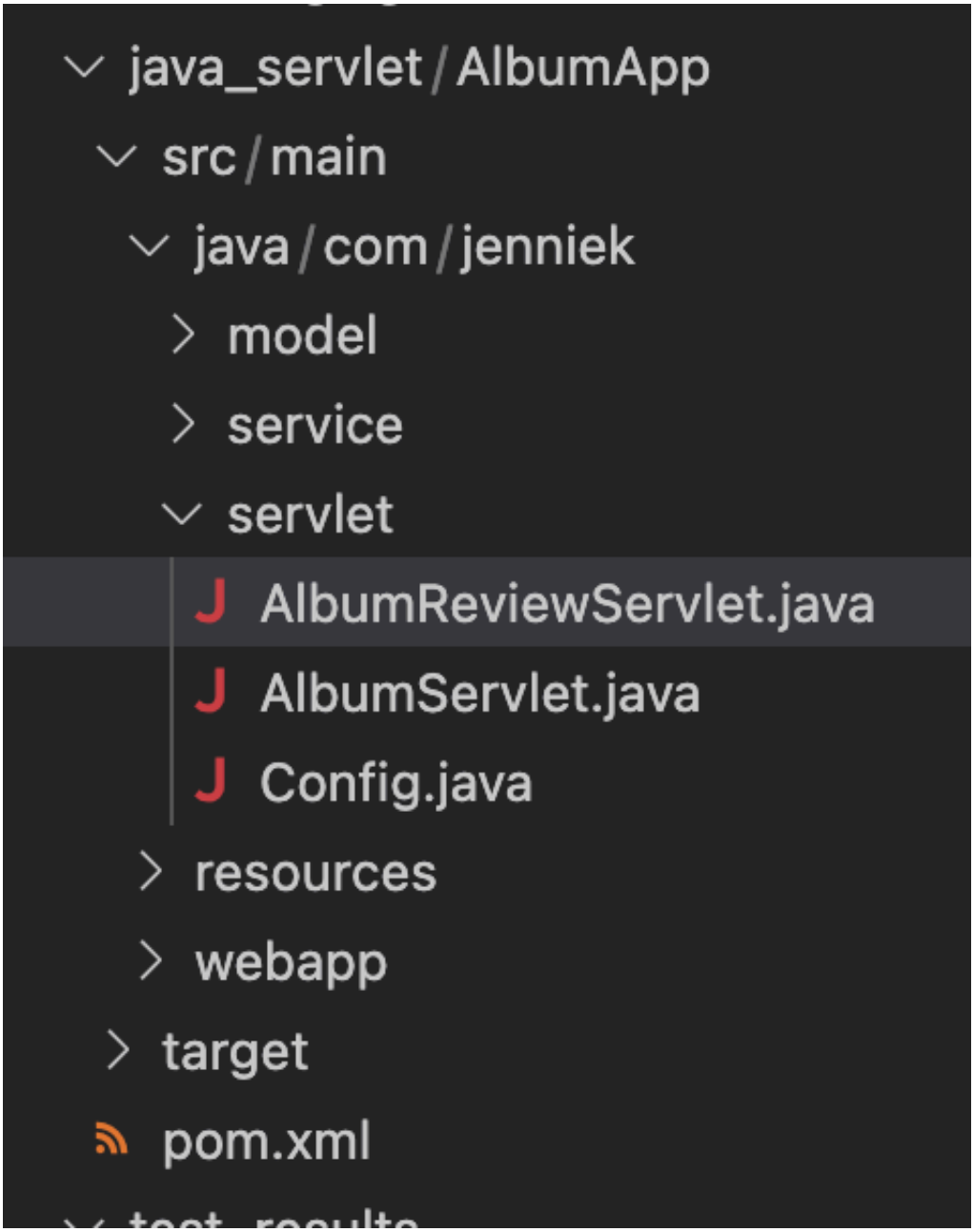
Config.java

set up the public static final arguments like `INITIAL_THREAD_COUNT`, `CLIENT_LOG_PATH` and `javaServletAddress`

Change this from 1000 to 100 `public static final int LOAD_TEST_REQUESTS_PER_THREAD = 100;` as this hw3 says "To reduce the new album write load, Just write 100 new albums per thread iteration instead of 1000".

1.3: server design with RabbitMQ

1) add review servlet

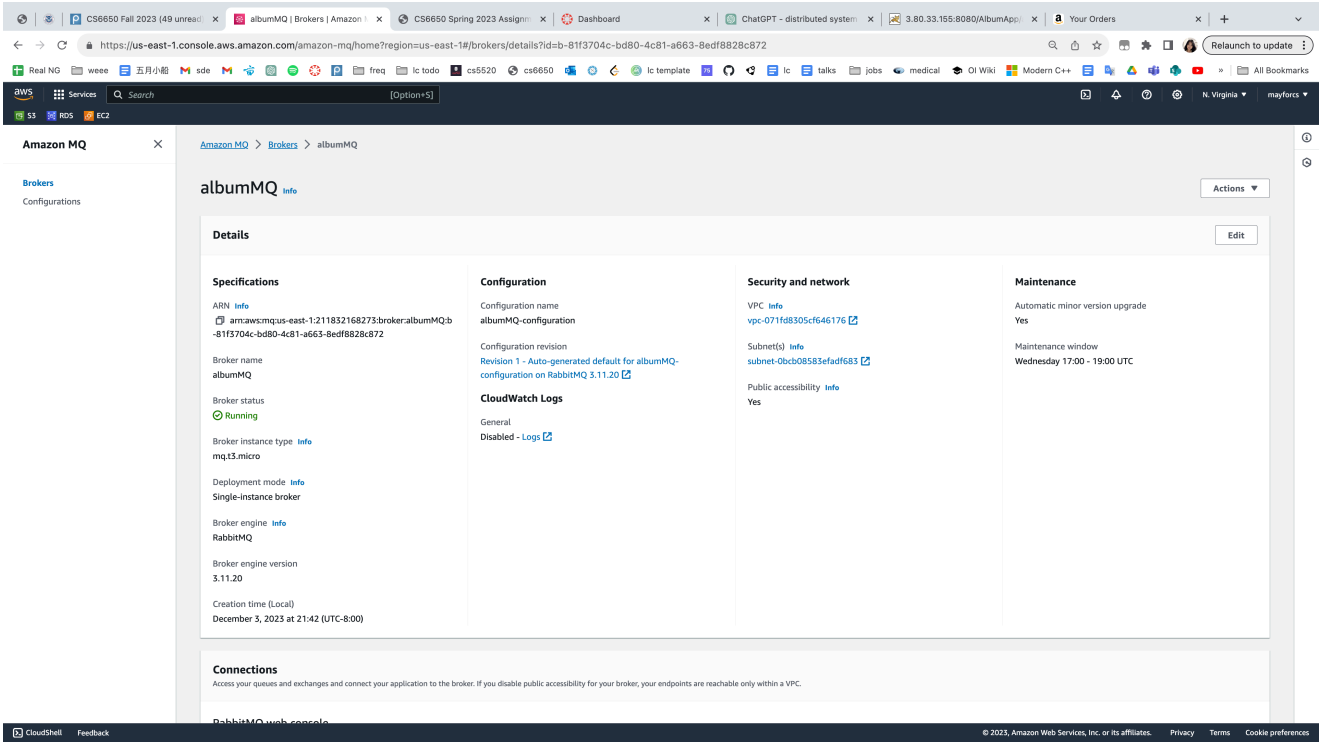


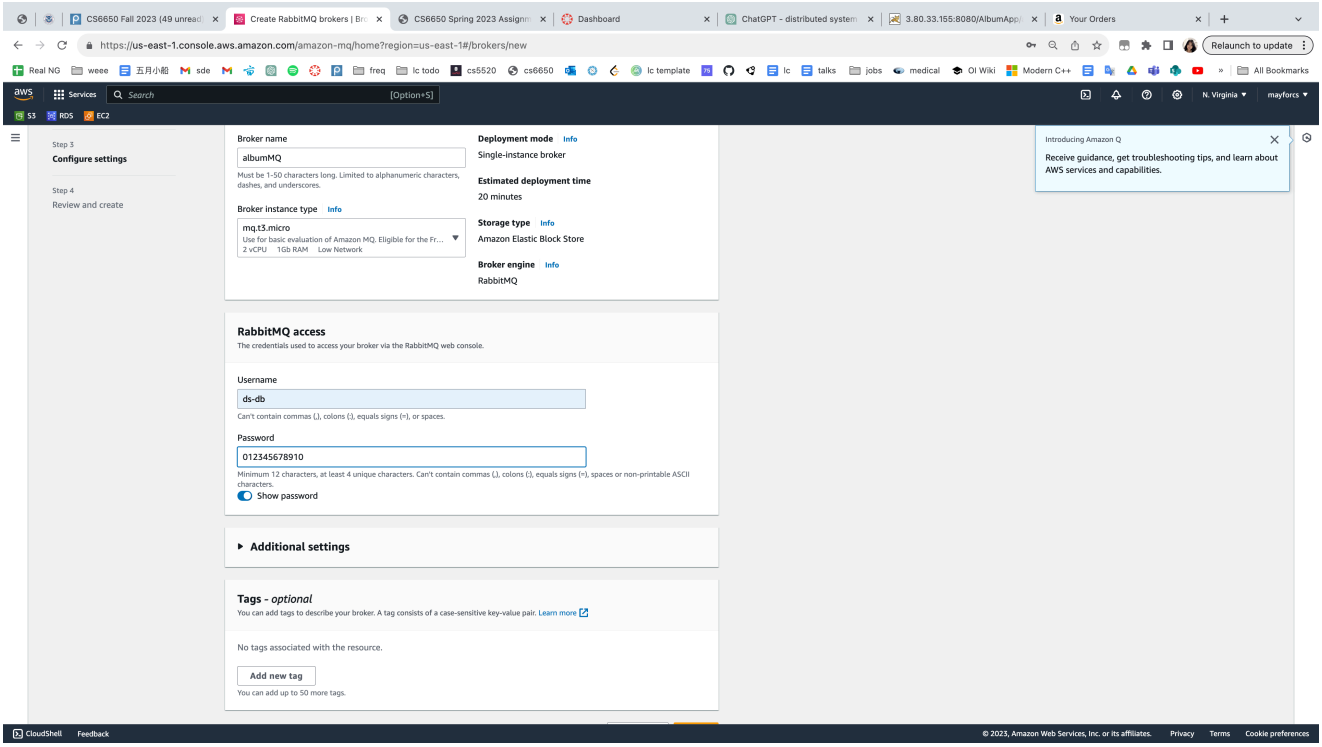

```
// AlbumReviewServlet.java
@WebServlet("/review/*")
public class AlbumReviewServlet extends HttpServlet {
    // implementation for handling like/dislike

// AlbumServlet.java
@WebServlet("/albums/*")
@MultipartConfig
public class AlbumServlet extends HttpServlet {
```

2) enable RabbitMQ

use producer and consumer

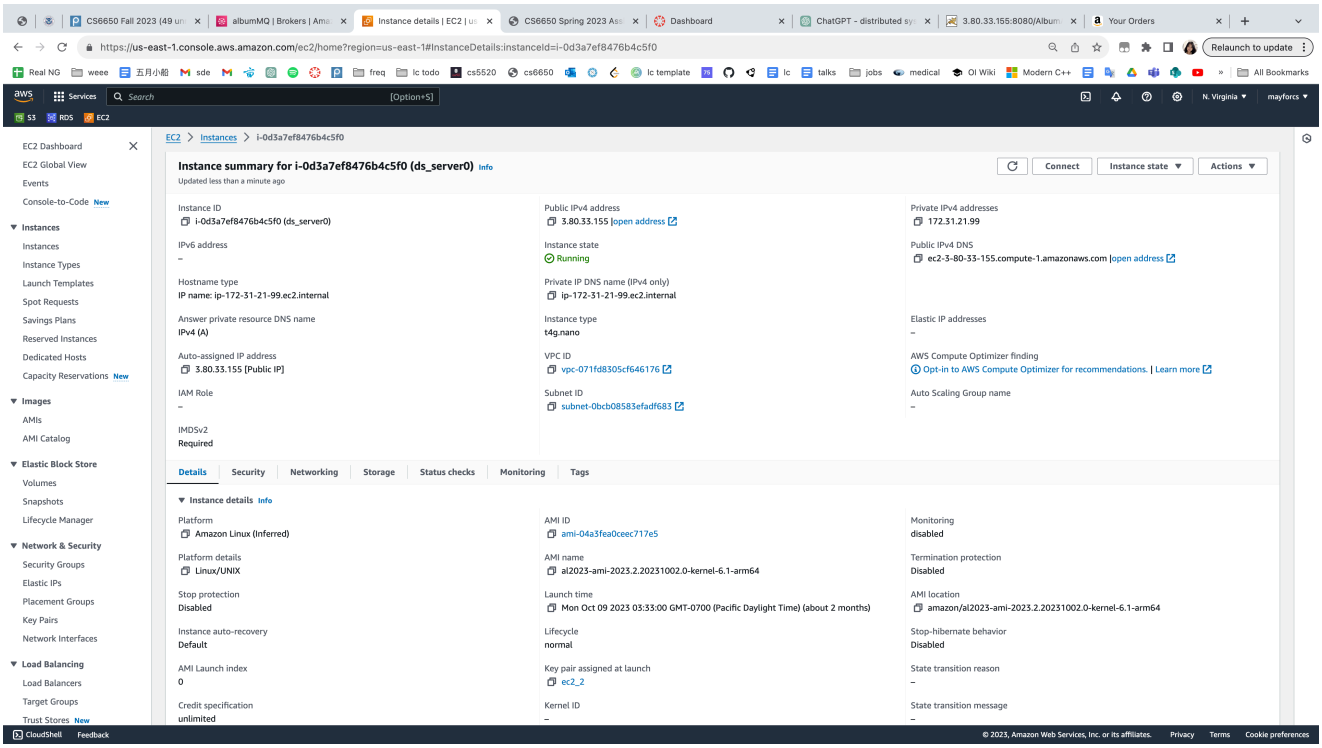




1.4: key info

AWS EC2

delete load balancer from hw2, so only use one AWS EC2



server address

```
public static final String javaServletAddress =
"http://3.80.33.155:8080/AlbumApp";
```

2: Database

2.1: choice

AWS RDS + MySQL 8.0.33

2.2: data model

alter the albums table in hw2 and clear it in the beginning for better performance

table: albums

field: album_i d(PK), name, artist, release_year, image, **likes, dislikes**

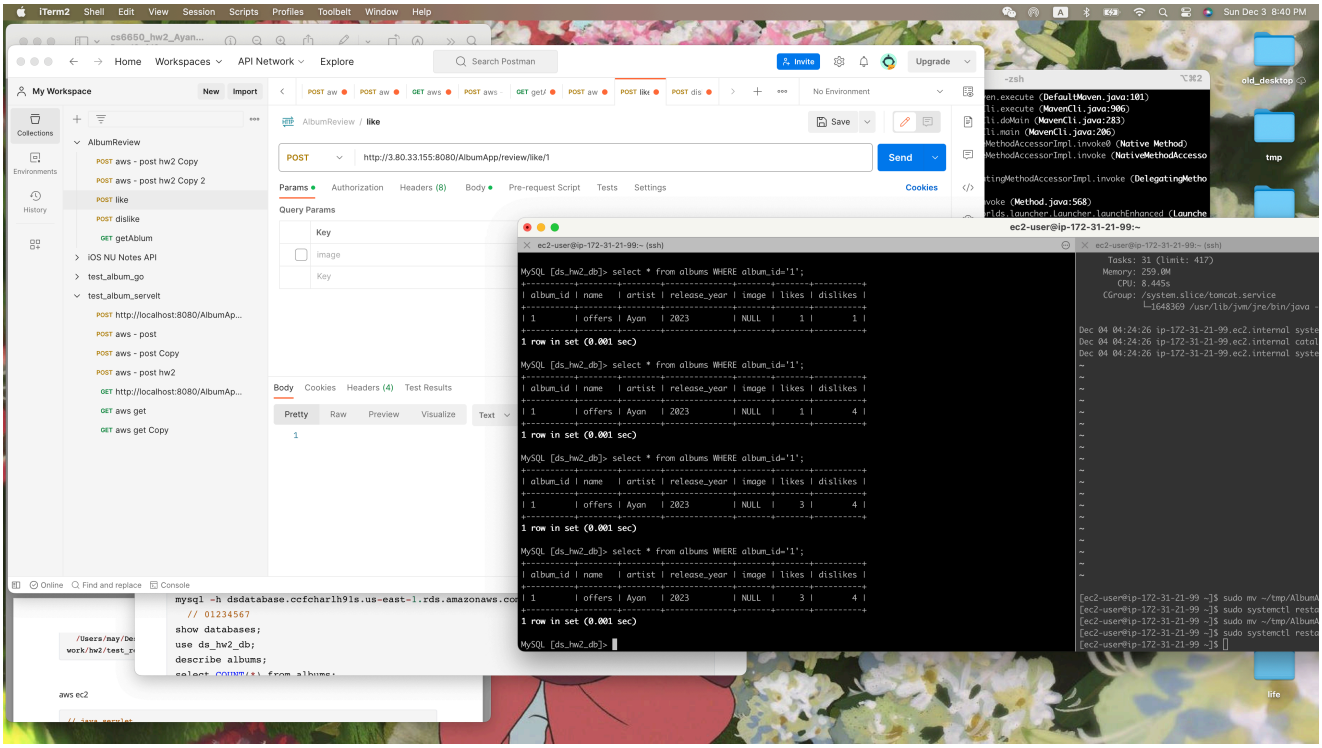
```
MySQL [ds_hw2_db]> describe albums;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| album_id     | varchar(50)  | NO   | PRI | NULL    |       |
| name         | varchar(100) | NO   |     | NULL    |       |
| artist       | varchar(100) | NO   |     | NULL    |       |
| release_year | varchar(4)   | YES  |     | NULL    |       |
| image        | varchar(255) | YES  |     | NULL    |       |
| likes        | int          | YES  |     | 0       |       |
| dislikes     | int          | YES  |     | 0       |       |
+--------------+--------------+------+-----+---------+-------+
7 rows in set (0.058 sec)
```

init albums table example:

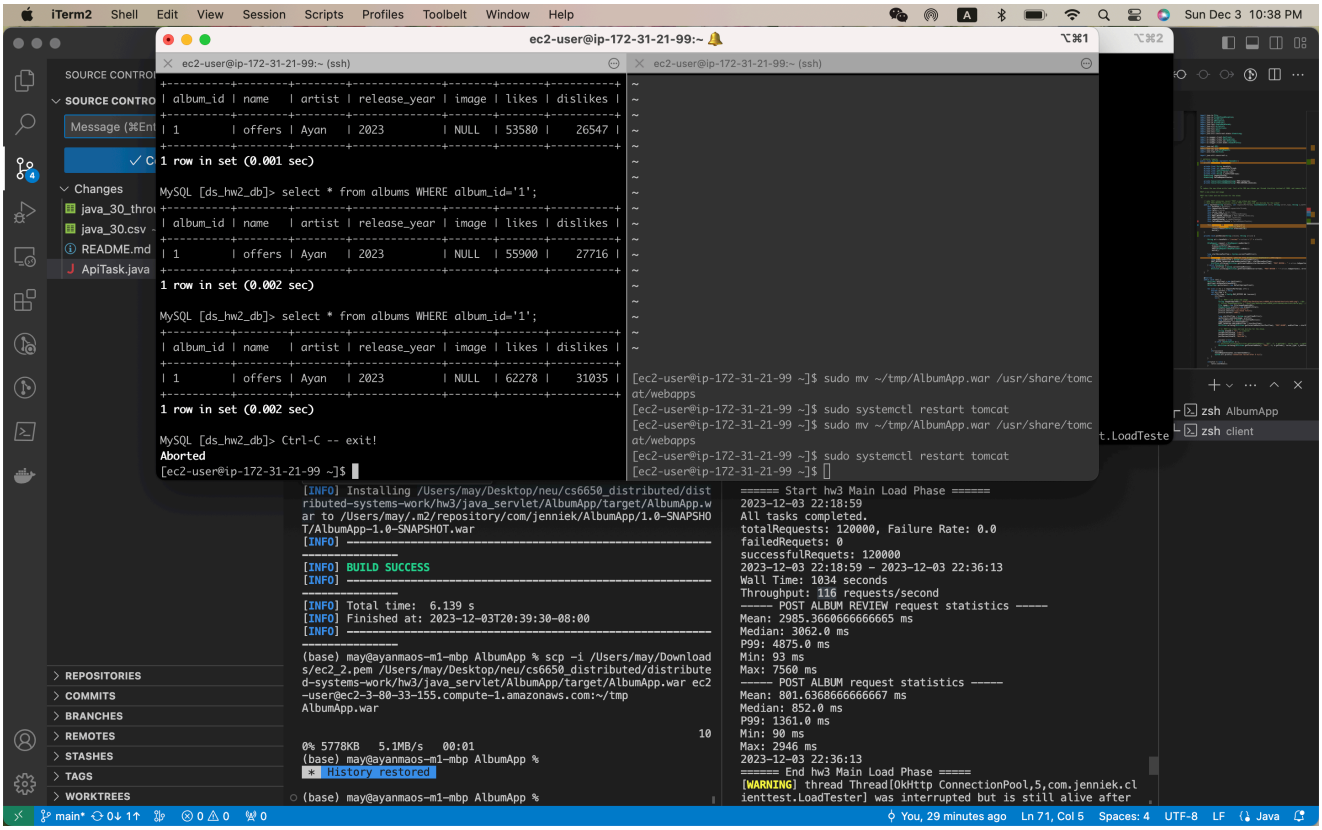
```
MySQL [ds_hw2_db]> select * from albums;
+--------------------------------------+---------+----------------+--------------+----------+-------+----------+
| album_id                             | name    | artist         | release_year | image    | likes | dislikes |
+--------------------------------------+---------+----------------+--------------+----------+-------+----------+
| 1                                    | offers  | Ayan           | 2023         | NULL     |     0 |        0 |
| 2                                    | offers  | Ayan           | 2023         | nmtb.jpg |   100 |        0 |
| 28ef6e80-9259-11ee-8d93-16914fa7a575 | 1       | Unknown Artist | 1998         | NULL     |     0 |        0 |
| 3                                    | offers3 | Ayan3          | 2023         | nmtb.jpg |   300 |        1 |
+--------------------------------------+---------+----------------+--------------+----------+-------+----------+
4 rows in set (0.001 sec)
```

2.3: results

simple postman test succeeded:

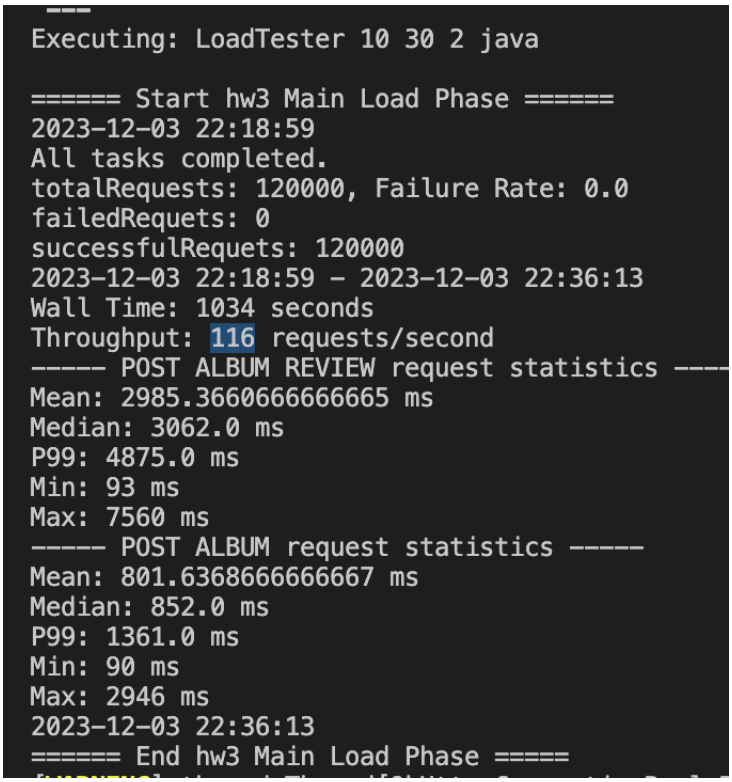


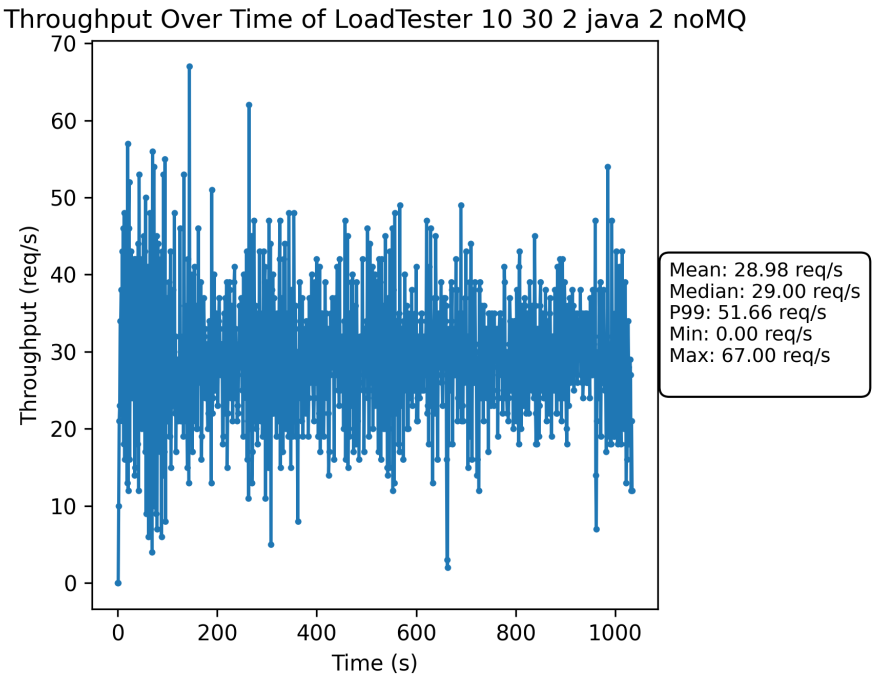
As you can see in the graph, it could have hundreds of thousands of records in the albums table.



3: test results

baseline test result without MQ





real test result with RabbitMQ enabled

```
(base) may@ayanmaos-m1-mbp client % mvn clean install && mvn exec:java -
Dexec.mainClass="com.jenniek.clienttest.LoadTester" -Dexec.args="10 30 2 java"
```

4: Appendix of Command

Please login as the user "ec2-user" rather than the user "root".

local

```
// ssh to aws ec2
```

```
/Users/may/Desktop/neu/cs6650_distributed/shortcuts/ssh_ec2.sh

  // chmod 400 ec2_2.pem
  // ssh -i ec2_2.pem ec2-user@ec2-3-80-33-155.compute-1.amazonaws.com
  ssh -i ec2_2.pem ec2-user@ec2-54-221-189-103.compute-1.amazonaws.com


// mysql to aws rdb
mysql -h dsdatabase.ccfcharlh91s.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
  // pwd: 01234567

  /* java servlet */
// local build java servlet in Maven project to generate the AlbumApp.war file
mvn clean install
// scp java servlet - AlbumApp.war to aws ec2
scp -i /Users/may/Downloads/ec2_2.pem
/Users/may/Desktop/neu/cs6650_distributed/distributed-systems-
work/hw3/java_servlet/AlbumApp/target/AlbumApp.war ec2-user@ec2-3-80-33-
155.compute-1.amazonaws.com:~/tmp


  /* Go server */
// local run go server(macOS + m1 chip) for local tests
GOOS=darwin GOARCH=arm64 go run ./main.go
// local build go server for aws ec2(linux + arm64)
GOOS=linux GOARCH=arm64 go build -o server main.go
// scp go server - server to ec2 ~/emp to start
scp -i /Users/may/Downloads/ec2_2.pem
/Users/may/Desktop/neu/cs6650_distributed/distributed-systems-
work/hw1/go_server/go-server-server-generated/server ec2-user@ec2-3-80-33-
155.compute-1.amazonaws.com:~/tmp

  /* client - LoadTester */
`Usage: LoadTester <threadGroupSize> <numThreadGroups> <delay> [java|go]`
// e.g.
mvn clean install && mvn exec:java -
Dexec.mainClass="com.jenniek.clienttest.LoadTester" -Dexec.args="10 30 2 java"

mvn exec:java -Dexec.mainClass="com.jenniek.clienttest.LoadTester" -Dexec.args="10
30 2 java"


  // plot throughput
  conda activate base
  /Users/may/anaconda3/bin/python
```

```
  /Users/may/Desktop/neu/cs6650_distributed/distributed-systems-
work/hw2/test_results/plot.py
```

aws ec2

```
// java servlet
sudo mv ~/tmp/AlbumApp.war /usr/share/tomcat/webapps && sudo systemctl restart
tomcat && systemctl status tomcat

sudo mv ~/tmp/AlbumApp.war /usr/share/tomcat/webapps

sudo systemctl restart tomcat
sudo systemctl stop tomcat
systemctl status tomcat

sudo vim /usr/share/tomcat/logs/catalina.out
sudo tail /usr/share/tomcat/logs/catalina.out -n 200

// see log to debug java servlet
sudo bash -c 'echo > /usr/share/tomcat/logs/catalina.out'
sudo grep "album" /usr/share/tomcat/logs/catalina.out

// go server
~/tmp/server


// mysql to aws rdb
mysql -h dsdatabase.ccfcharlh91s.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
  // 01234567
show databases;
use ds_hw2_db;
describe albums;
select COUNT(*) from albums;
SELECT * FROM albums ORDER BY RAND() LIMIT 20;

ALTER TABLE albums
ADD COLUMN likes INT DEFAULT 0,
ADD COLUMN dislikes INT DEFAULT 0;

INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('1', 'offers', 'Ayan', 2023, NULL, 0, 0);
```

```sql
INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('2', 'offers', 'Ayan', 2023, 'nmtb.jpg', 100, 0);

INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('3', 'offers3', 'Ayan3', 2023, 'nmtb.jpg', 300, 1);

INSERT INTO albums (album_id, name, artist, release_year, image, likes, dislikes)
VALUES ('1230', 'offers1230', 'Ayan1230', 1998, 'nmtb.jpg', 12300, 50);
```