Assignment 3 write up

1. URL for your code repo

Github link: https://github.com/kabraambika/CS6650Assignment3

2. Output windows for the 3 client configuration tests run against a single server/DB

Configuration used : 3 EC2 Instances in total used

1. RabbitMQ : EC2 named MyFirstServer
2. MongoDB : EC2 named DatabaseServer
3. Server(tomcat): EC2 named Server2Ambika

Instructions to setup environment for mongodb setup on EC2

Reference : https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-amazon/

- Deploy a Compute Engine Instance
- Using the EC2 service, launch an instance named DatabaseServer. Choose the free tier Amazon Linux image on a t2.micro instance type.
- Connect to the Instance and Install MongoDB
- Connect using the private key.
- Create the repo file in /etc/yum.repos.d/mongodb-org-5.0.repo with the following content:

```
[mongodb-org-5.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/amazon/2/mongodb-org/5.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-5.0.asc
```

- Install the mongodb-org mongodb-mongosh packages.
- Start the mongod service.
- Check the status of the mongod service.
- Connect via mongosh.
- Follow this link to allow connections from elsewhere in your Amazon VPC, https://docs.aws.amazon.com/dms/latest/sbs/chap-mongodb2documentdb.02.html

RabbitMQ installation for mac

Configuration used https://www.rabbitmq.com/install-homebrew.html

Install RabbitMQ server with : brew install rabbitmq
Accessible from : /usr/local/sbin

RabbitMQ in project in pom.xml:

```
<!-- https://mvnrepository.com/artifact/com.rabbitmq/amqp-client -->
  <dependency>
    <groupId>com.rabbitmq</groupId>
    <artifactId>amqp-client</artifactId>
    <version>5.18.0</version>
  </dependency>

  <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>2.0.7</version>
  </dependency>

  <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>2.0.5</version>
    <scope>test</scope>
  </dependency>
```

To start a node in the background, use brew services start rabbitmq
To stop a running node, use: brew services stop rabbitmq

In case that directory is not in PATH, it is recommended to append it:

```
# for macOS Intel
export PATH=$PATH:/usr/local/sbin
```

Add the above export to the shell profile (such as ~/.bashrc for bash or ~/.zshrc for zsh) to have PATH updated for every new shell, including OS restarts.

Instructions to setup environment for RabbitMQ setup on EC2

- Using the EC2 service, launch an instance named MyFirstServer. Choose the free tier Amazon Linux 2 image on a t2.micro instance type.
- Connect to the Instance using private key
- Install Epel-release : sudo amazon-linux-extras install epel
- Install erlang : sudo yum install erlang
- Install RabbitMQ server : sudo yum install rabbitmq-server
- Check all the plugins : sudo rabbitmq-plugins list
- Enable RabbitMQ management : sudo rabbitmq-plugins enable rabbitmq_management
- Recheck : sudo rabbitmq-plugins list
- Command to enable: sudo systemctl enable rabbitmq-server
- Command to start rabbitmq: sudo systemctl start rabbitmq-server
- Command to stop rabbitmq: sudo systemctl stop rabbitmq-server
- Note: custom TCP with 15672 and 5672 port should be added as inbound in security group

Instructions to setup environment for tomcat and deploying war file of server on EC2

1. Deploy a EC2
    a. Using the EC2 service, launch an instance. Choose the free tier Amazon Linux image on a t2.micro instance type.
2. Connect to the Instance and Install Java and tomcat
    a. Connect using the private key
    b. Install java 17
        i. sudo dnf install java-17-amazon-corretto-devel
    c. Create tomcat user and group
        i. sudo groupadd --system tomcat
        ii. sudo useradd -d /usr/share/tomcat -r -s /bin/false -g tomcat tomcat
    d. Install Tomcat 9 on Amazon Linux 2
        i. sudo yum -y install wget
        ii. export VER="9.0.80"
        iii. wget https://archive.apache.org/dist/tomcat/tomcat-9/v${VER}/bin/apache-tomcat-${VER}.tar.gz
        iv. sudo tar xvf apache-tomcat-${VER}.tar.gz -C /usr/share/
        v. sudo ln -s /usr/share/apache-tomcat-$VER/ /usr/share/tomcat
        vi. sudo chown -R tomcat:tomcat /usr/share/tomcat
        vii. sudo chown -R tomcat:tomcat /usr/share/apache-tomcat-$VER/
        viii. Create Tomcat Systemd service:

```
sudo tee /etc/systemd/system/tomcat.service<<EOF
[Unit]
Description=Tomcat Server
After=syslog.target network.target

[Service]
Type=forking
User=tomcat
Group=tomcat

Environment=JAVA_HOME=/usr/lib/jvm/jre
Environment='JAVA_OPTS=-Djava.awt.headless=true'
Environment=CATALINA_HOME=/usr/share/tomcat
Environment=CATALINA_BASE=/usr/share/tomcat
Environment=CATALINA_PID=/usr/share/tomcat/temp/tomcat.pid
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M'
ExecStart=/usr/share/tomcat/bin/catalina.sh start
ExecStop=/usr/share/tomcat/bin/catalina.sh stop

[Install]
WantedBy=multi-user.target
EOF
```

        ix. Enable and start tomcat service:

```
sudo systemctl daemon-reload
sudo systemctl start tomcat
sudo systemctl enable tomcat
sudo systemctl status tomcat
```

        x. Configure Apache web server as a proxy for Tomcat server. First install httpd package.

sudo yum -y install httpd

xi. sudo vim /etc/httpd/conf.d/tomcat_manager.conf

```
<VirtualHost *:80>
   ServerAdmin root@localhost
   ServerName tomcat.example.com
   DefaultType text/html
   ProxyRequests off
   ProxyPreserveHost On
   ProxyPass / http://localhost:8080/
   ProxyPassReverse / http://localhost:8080/
</VirtualHost>
```

xii. sudo systemctl restart httpd

xiii. sudo systemctl enable httpd

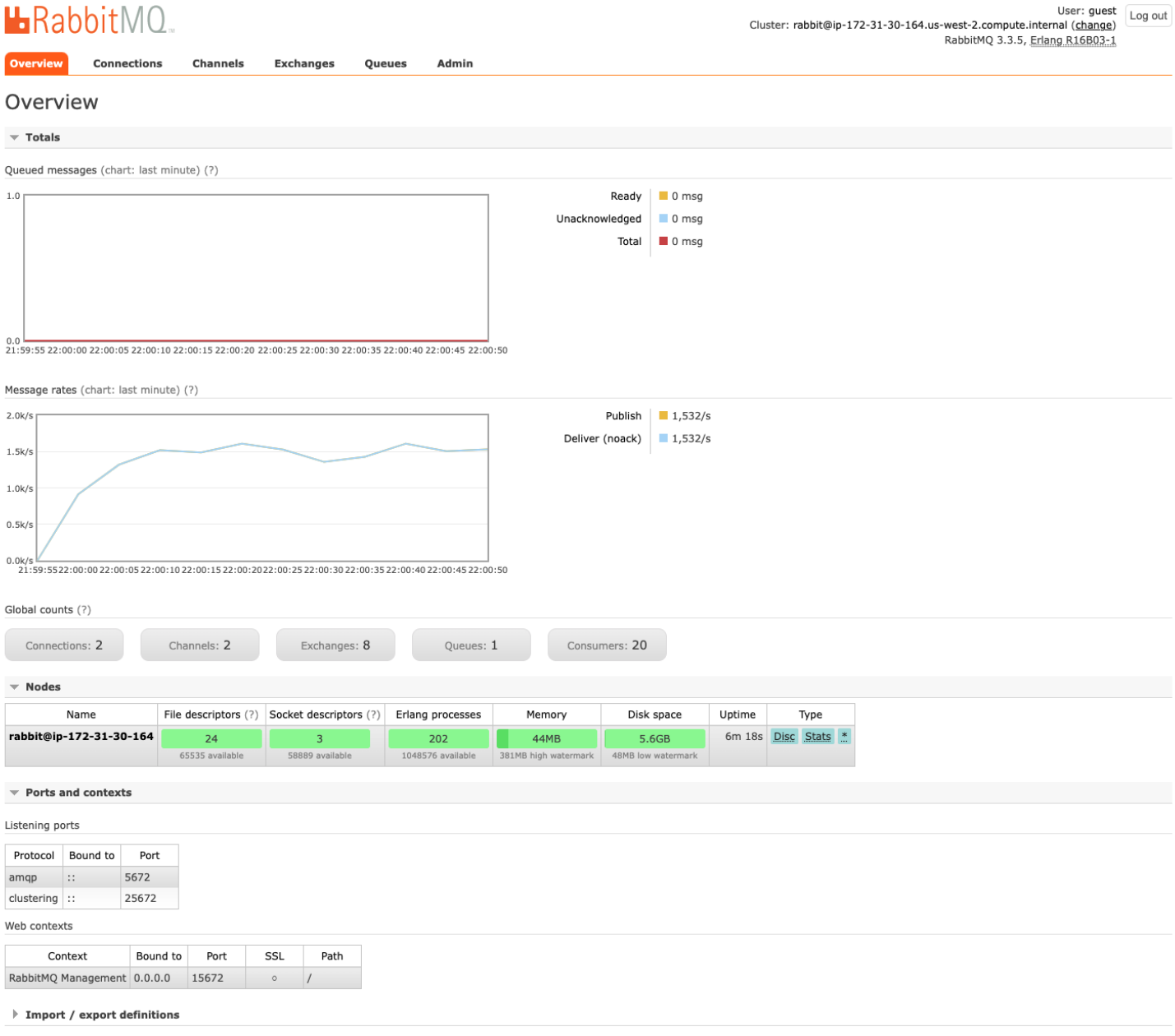
3. Give permission to add war application
   cd /usr/share/tomcat
   sudo chmod -R 777 webapps/

4. sudo scp -i /path/to/pem/file /local/path/to/war/file
   ec2-user@EC2_IP_ADDR:/remote/path/to/tomcat_webapp/directory

5. Visit http://{YOUR_REMOTE_INSTANCE_IP}:8080/[WEB_APP]/albums and ready
6. Note: custom TCP with 8080 port should be added as inbound in security group

Results

Load test the servers: threadGroupSize = 10, numThreadGroups = 30, delay = 2



RabbitMQ

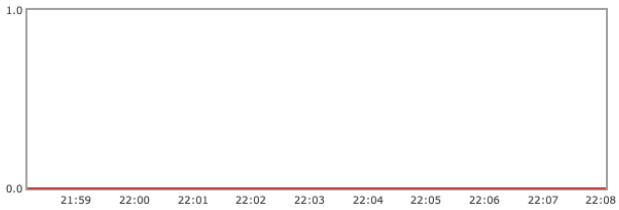
User: guest
Cluster: rabbit@ip-172-31-30-164.us-west-2.compute.internal (change)
RabbitMQ 3.3.5, Erlang R16B03-1
Log out

Overview | Connections | Channels | Exchanges | Queues | Admin

Overview

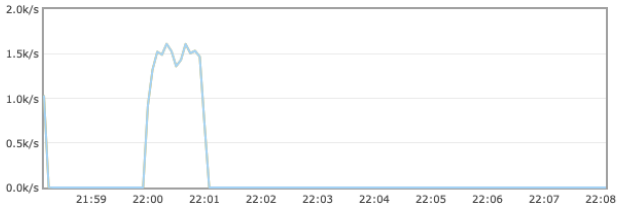
Totals

Queued messages (chart: last ten minutes) (?)



Ready	0 msg
Unacknowledged	0 msg
Total	0 msg

Message rates (chart: last ten minutes) (?)



Publish	0.00/s
Deliver (noack)	0.00/s

Global counts (?)

Connections: 2 | Channels: 2 | Exchanges: 8 | Queues: 1 | Consumers: 20

Nodes

Name	File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Uptime	Type
rabbit@ip-172-31-30-164	21 (65535 available)	3 (58889 available)	199 (1048576 available)	44MB (381MB high watermark)	5.6GB (48MB low watermark)	13m 29s	Disc Stats *

Ports and contexts

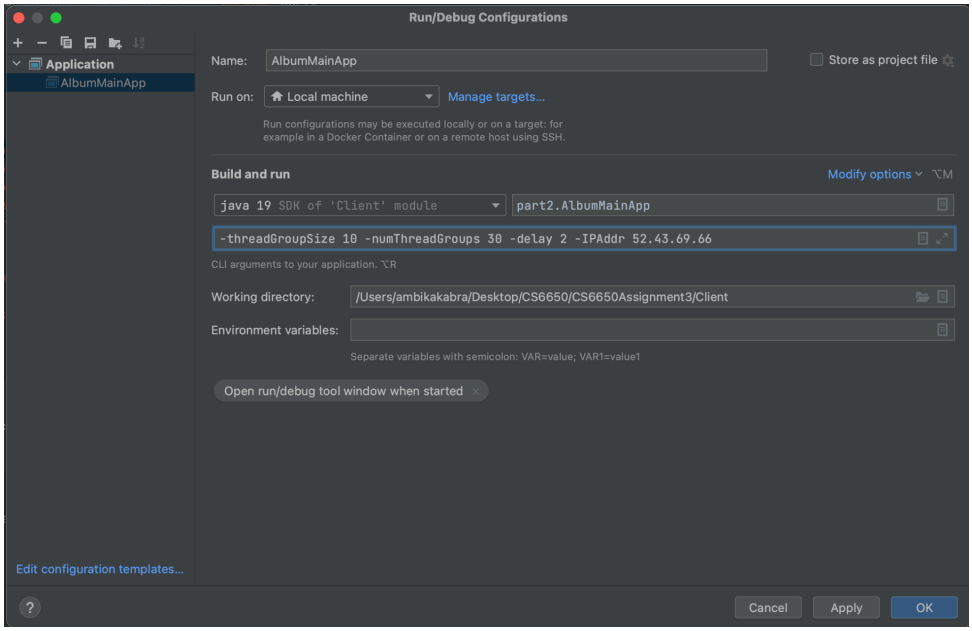
Listening ports

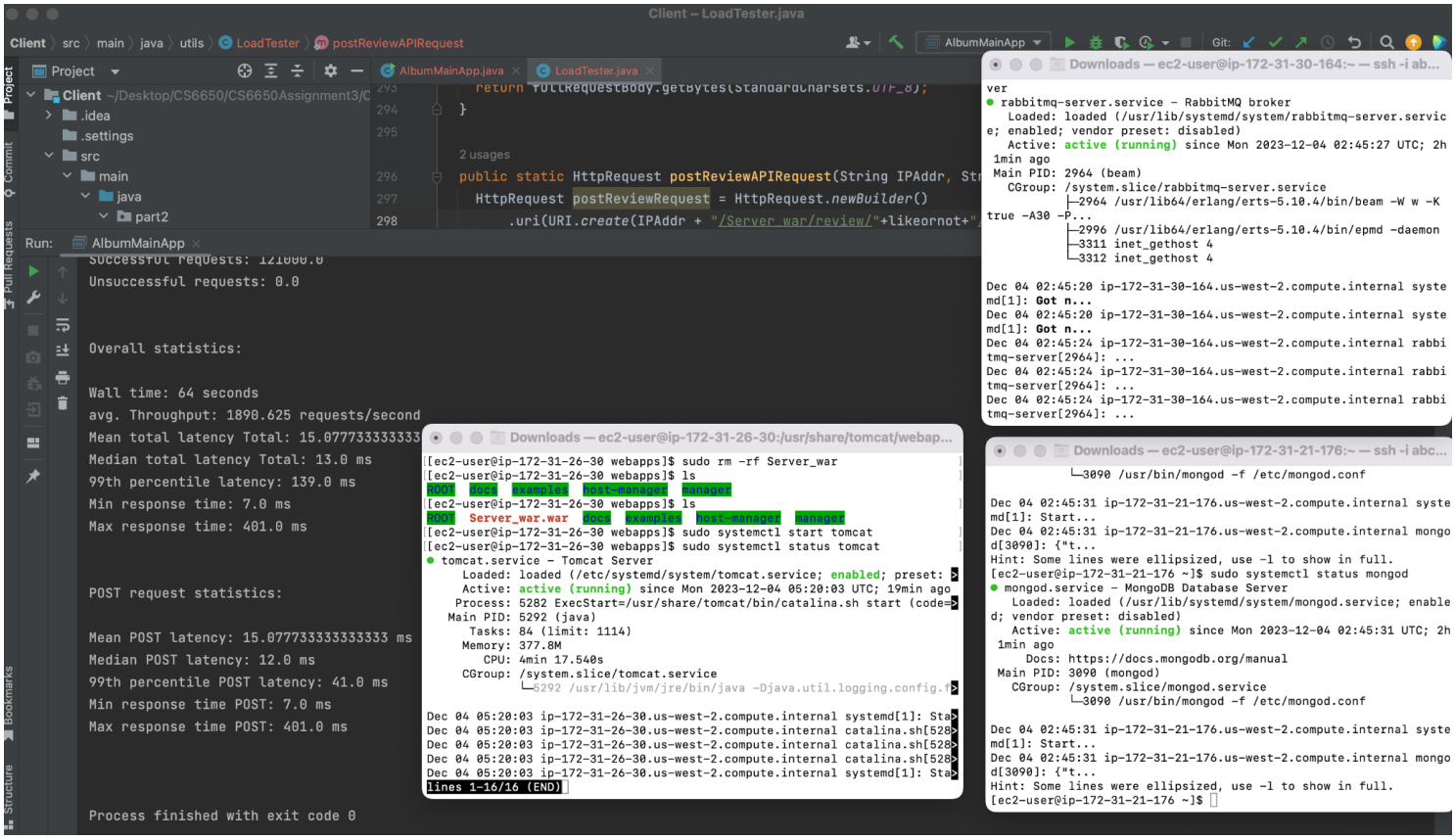
Protocol	Bound to	Port
amqp	::	5672
clustering	::	25672

Web contexts

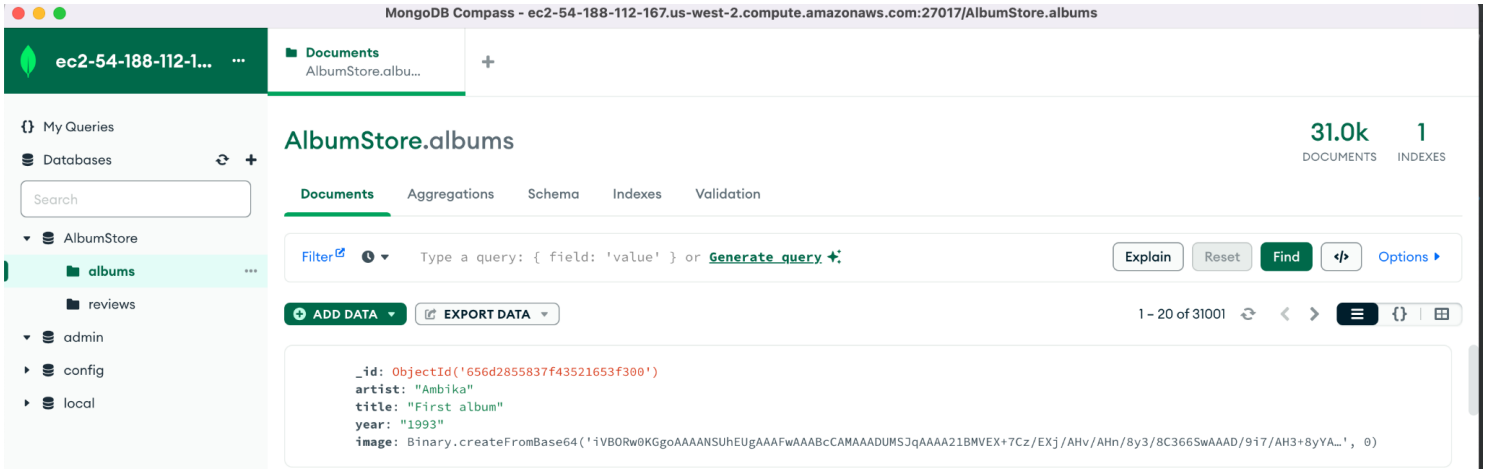
Context	Bound to	Port	SSL	Path
RabbitMQ Management	0.0.0.0	15672	○	/

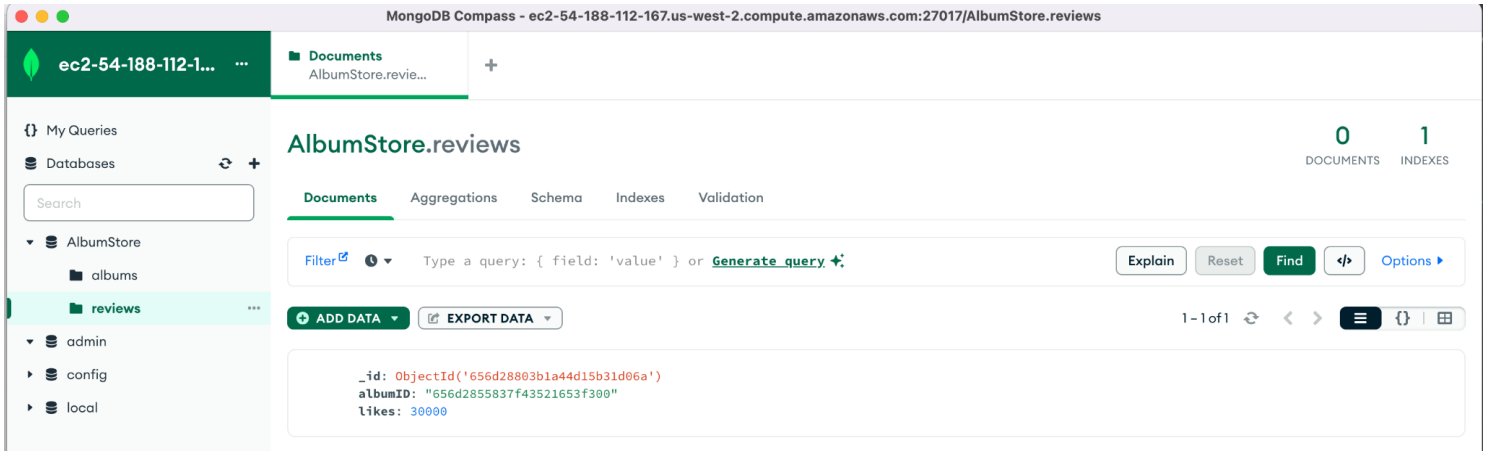
Import / export definitions



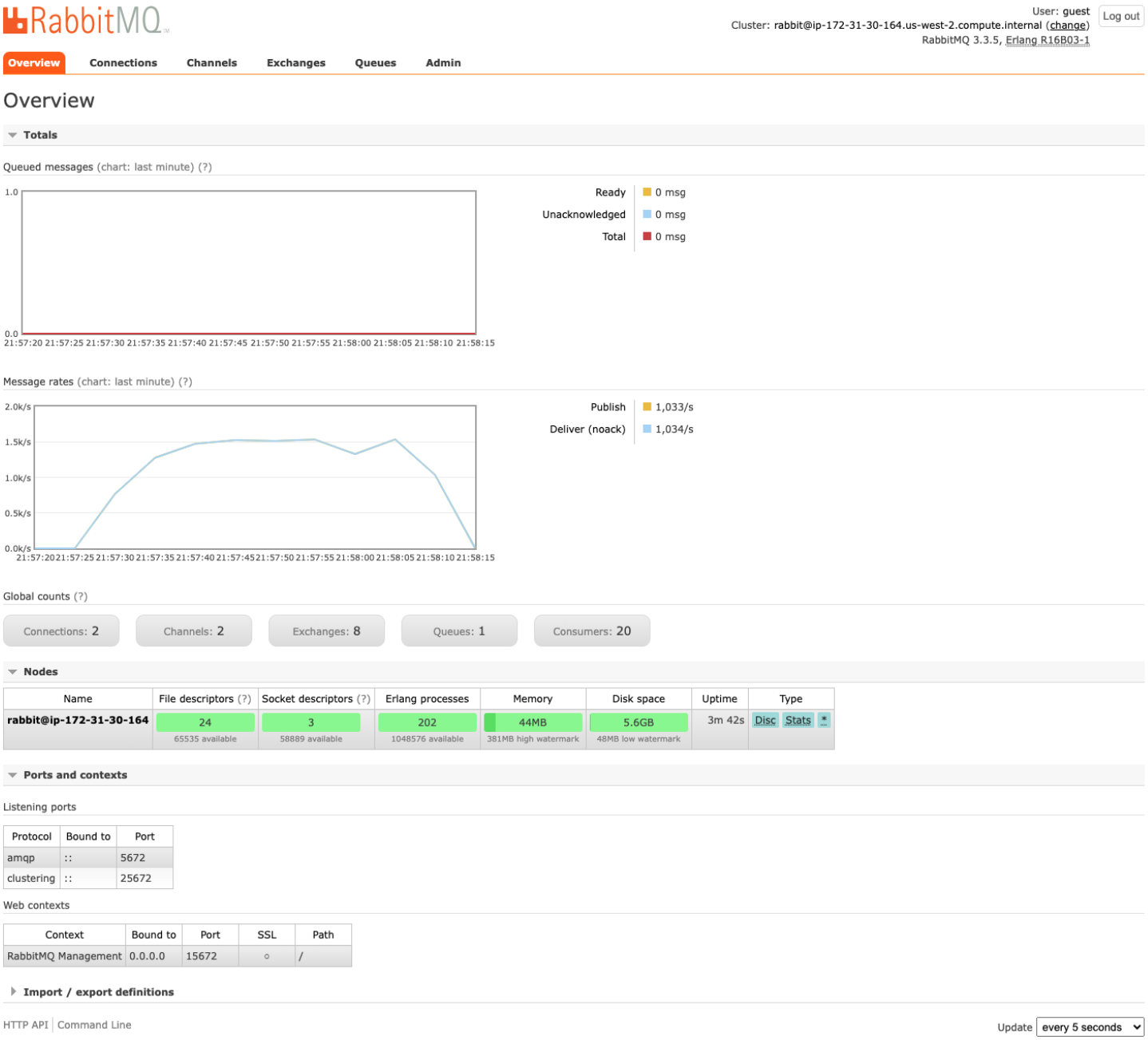


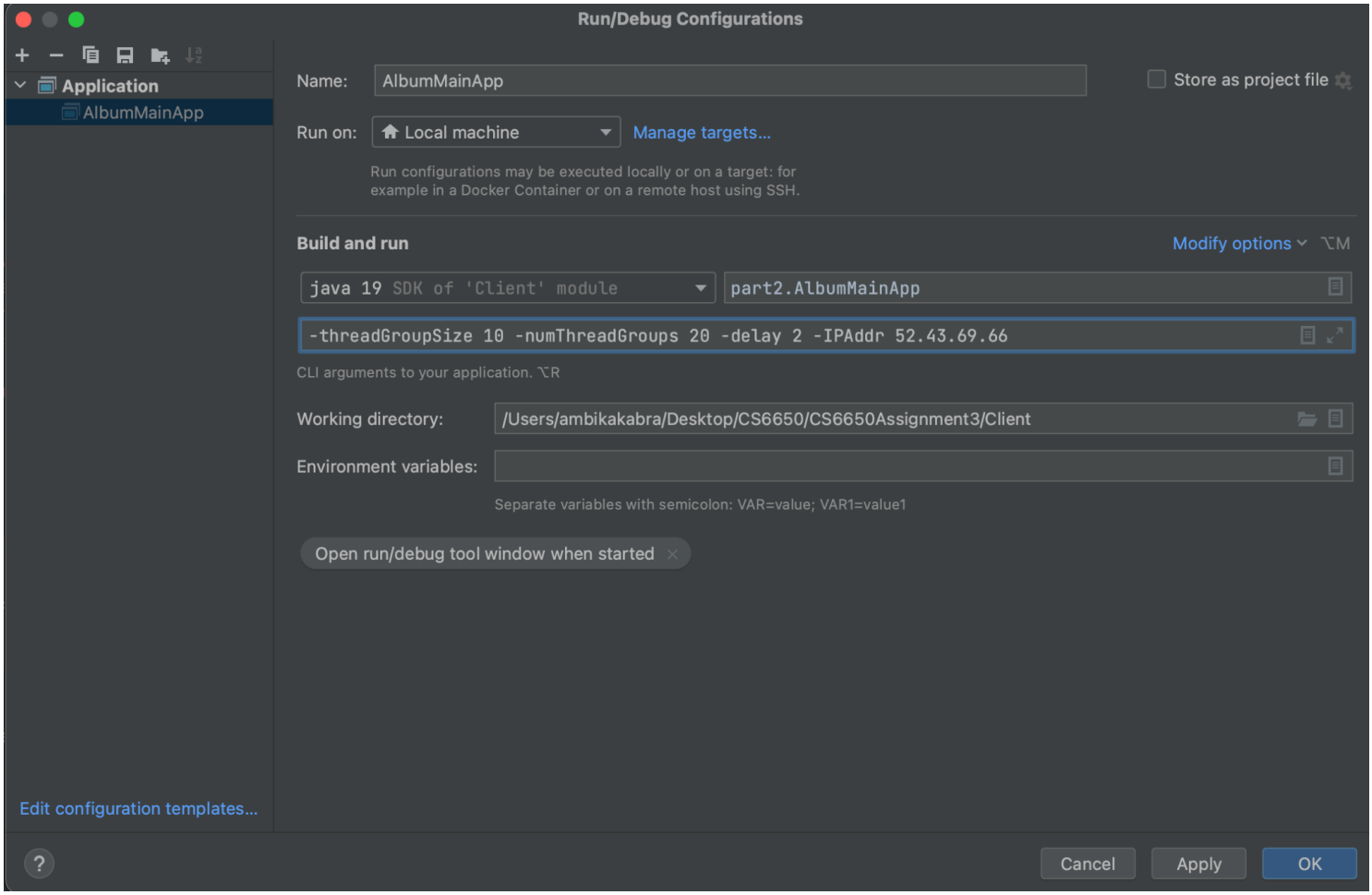
Database output result:

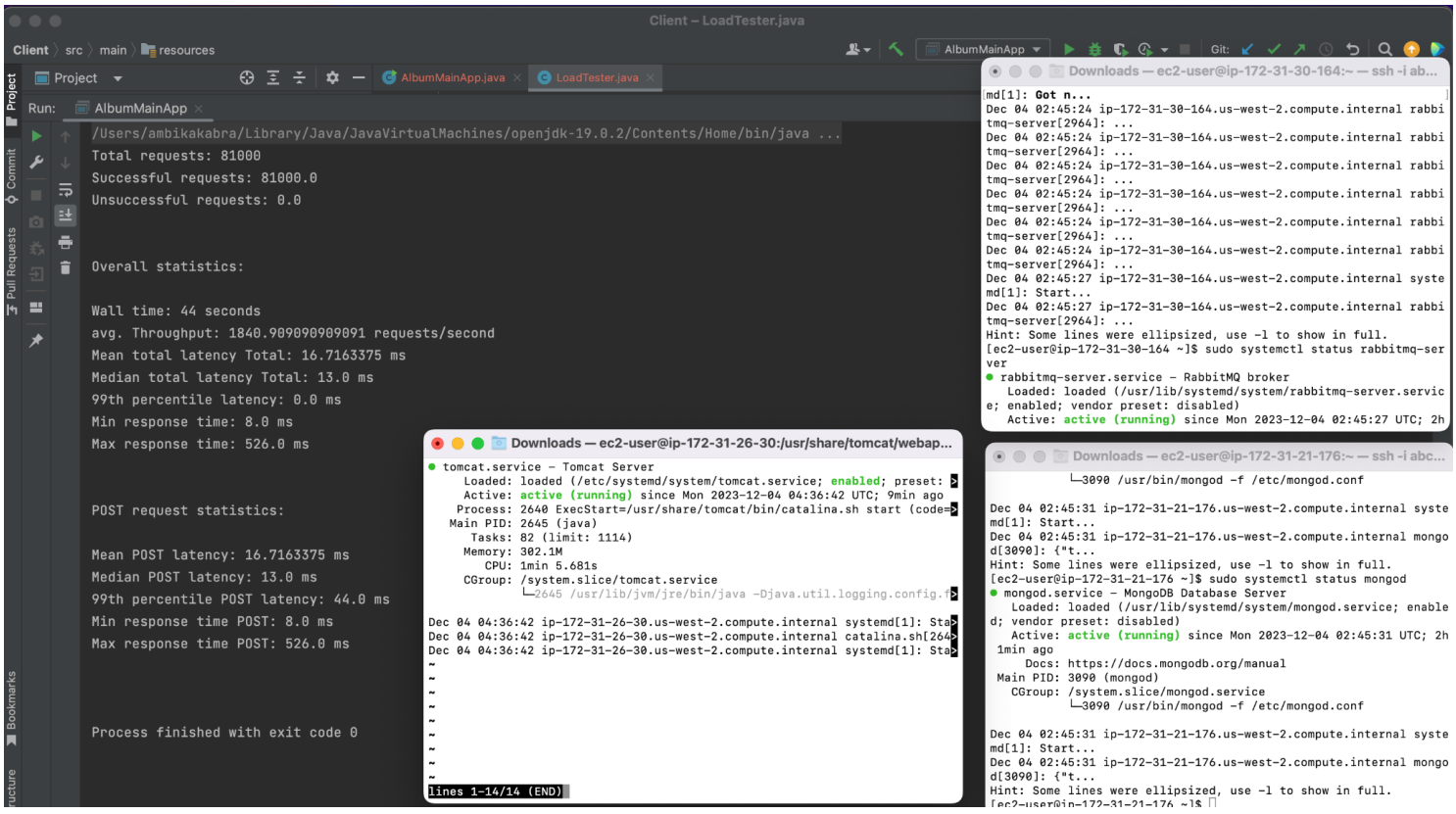


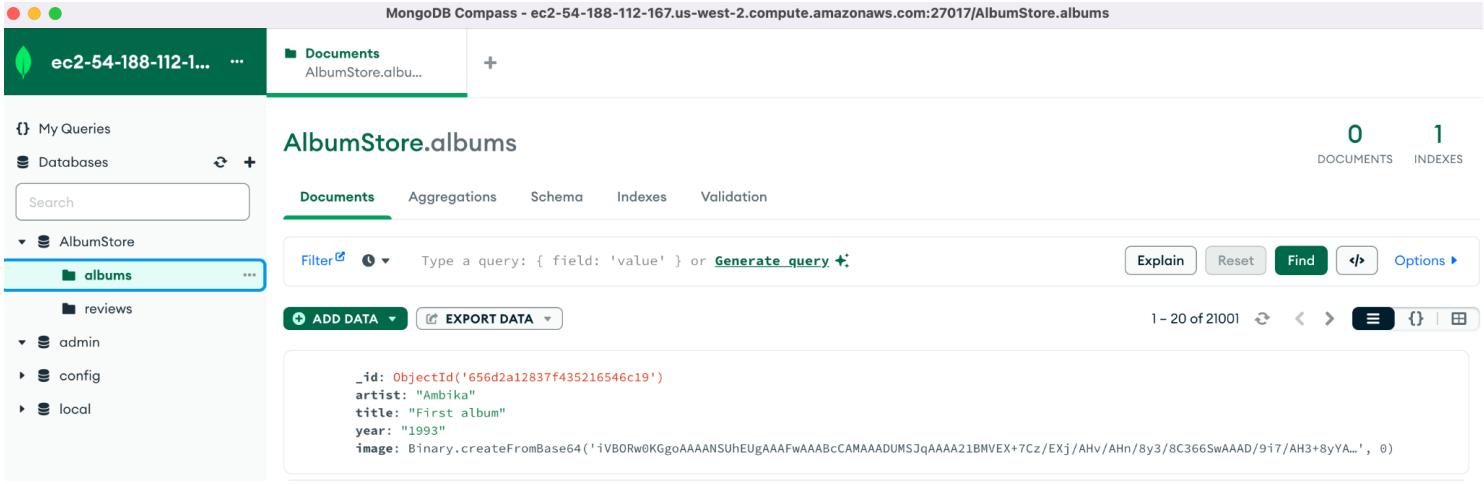


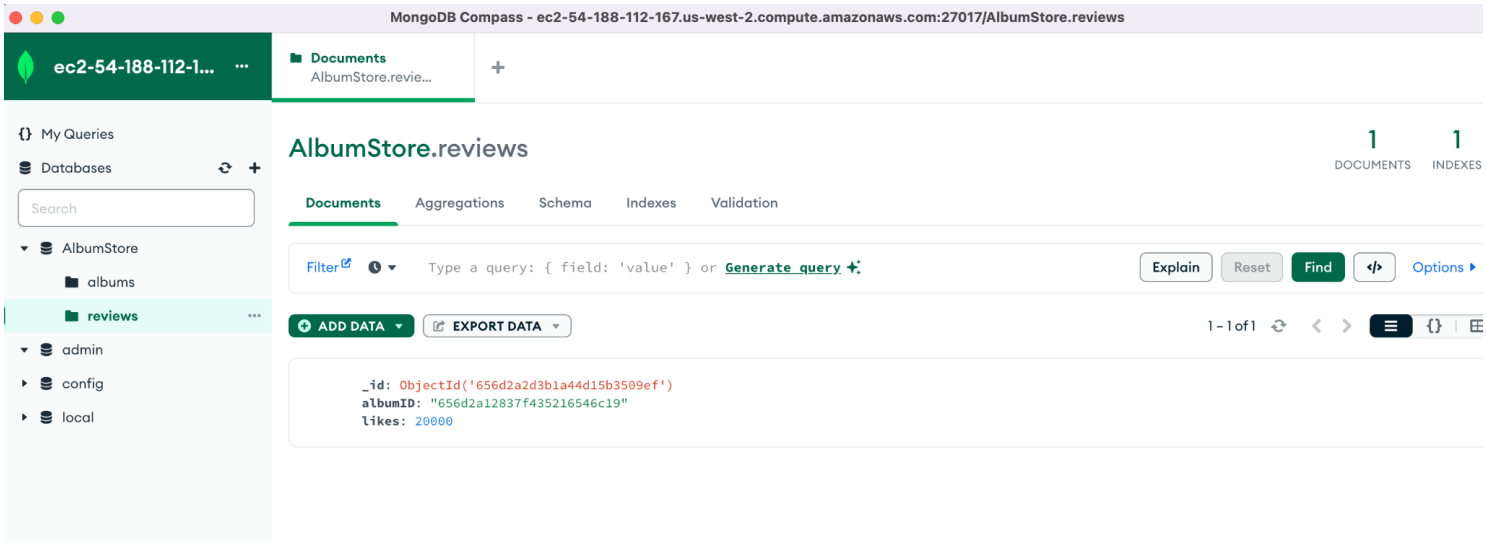
Load test the servers: threadGroupSize = 10, numThreadGroups = 20, delay = 2




Run/Debug Configurations
Application
AlbumMainApp
Name: AlbumMainApp
Store as project file
Run on: Local machine
Manage targets...
Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.
Build and run
Modify options ⌥M
java 19 SDK of 'Client' module
part2.AlbumMainApp
-threadGroupSize 10 -numThreadGroups 20 -delay 2 -IPAddr 52.43.69.66
CLI arguments to your application. ⌥R
Working directory: /Users/ambikakabra/Desktop/CS6650/CS6650Assignment3/Client
Environment variables:
Separate variables with semicolon: VAR=value; VAR1=value1
Open run/debug tool window when started
Edit configuration templates...
Cancel
Apply
OK


Client – LoadTester.java
Run: AlbumMainApp
/Users/ambikakabra/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/bin/java ...
Total requests: 81000
Successful requests: 81000.0
Unsuccessful requests: 0.0
Overall statistics:
Wall time: 44 seconds
avg. Throughput: 1840.909090909091 requests/second
Mean total latency Total: 16.7163375 ms
Median total latency Total: 13.0 ms
99th percentile latency: 0.0 ms
Min response time: 8.0 ms
Max response time: 526.0 ms
POST request statistics:
Mean POST latency: 16.7163375 ms
Median POST latency: 13.0 ms
99th percentile POST latency: 44.0 ms
Min response time POST: 8.0 ms
Max response time POST: 526.0 ms
Process finished with exit code 0
Downloads — ec2-user@ip-172-31-26-30:/usr/share/tomcat/webap...
tomcat.service - Tomcat Server
Loaded: loaded (/etc/systemd/system/tomcat.service; enabled; preset:
Active: active (running) since Mon 2023-12-04 04:36:42 UTC; 9min ago
Process: 2640 ExecStart=/usr/share/tomcat/bin/catalina.sh start (code=
Main PID: 2645 (java)
Tasks: 82 (limit: 1114)
Memory: 302.1M
CPU: 1min 5.681s
CGroup: /system.slice/tomcat.service
└2645 /usr/lib/jvm/jre/bin/java -Djava.util.logging.config.f
Dec 04 04:36:42 ip-172-31-26-30.us-west-2.compute.internal systemd[1]: Sta
Dec 04 04:36:42 ip-172-31-26-30.us-west-2.compute.internal catalina.sh[264
Dec 04 04:36:42 ip-172-31-26-30.us-west-2.compute.internal systemd[1]: Sta
lines 1-14/14 (END)
Downloads — ec2-user@ip-172-31-30-164:~ — ssh -i ab...
Dec 04 02:45:24 ip-172-31-30-164.us-west-2.compute.internal rabbitmq-server[2964]: ...
Dec 04 02:45:27 ip-172-31-30-164.us-west-2.compute.internal systemd[1]: Start...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-30-164 ~]$ sudo systemctl status rabbitmq-server
rabbitmq-server.service - RabbitMQ broker
Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2023-12-04 02:45:27 UTC; 2h
Downloads — ec2-user@ip-172-31-21-176:~ — ssh -i abc...
└3090 /usr/bin/mongod -f /etc/mongod.conf
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal systemd[1]: Start...
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal mongod[3090]: {"t...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-21-176 ~]$ sudo systemctl status mongod
mongod.service - MongoDB Database Server
Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2023-12-04 02:45:31 UTC; 2h 1min ago
Docs: https://docs.mongodb.org/manual
Main PID: 3090 (mongod)
CGroup: /system.slice/mongod.service
└3090 /usr/bin/mongod -f /etc/mongod.conf
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal systemd[1]: Start...
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal mongod[3090]: {"t...
Hint: Some lines were ellipsized, use -l to show in full.


MongoDB Compass - ec2-54-188-112-167.us-west-2.compute.amazonaws.com:27017/AlbumStore.albums
ec2-54-188-112-1...
Documents
AlbumStore.albu...
My Queries
Databases
Search
AlbumStore
albums
reviews
admin
config
local
AlbumStore.albums
0
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 20 of 21001
_id: ObjectId('656d2a12837f435216546c19')
artist: "Ambika"
title: "First album"
year: "1993"
image: Binary.createFromBase64('iVBORw0KGgoAAAANSUhEUgAAAFwAAABcCAMAAADUMSJqAAAA21BMVEX+7Cz/EXj/AHv/AHn/8y3/8C366SwAAAD/9i7/AH3+8yYA…', 0)


MongoDB Compass - ec2-54-188-112-167.us-west-2.compute.amazonaws.com:27017/AlbumStore.reviews
ec2-54-188-112-1...
Documents
AlbumStore.revie...
My Queries
Databases
Search
AlbumStore
albums
reviews
admin
config
local
AlbumStore.reviews
1
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 1 of 1
_id: ObjectId('656d2a2d3b1a44d15b3509ef')
albumID: "656d2a12837f435216546c19"
likes: 20000

Load test the servers: threadGroupSize = 10, numThreadGroups = 10, delay = 2

RabbitMQ

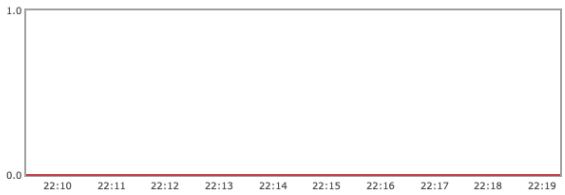
User: guest
Cluster: rabbit@ip-172-31-30-164.us-west-2.compute.internal (change)
RabbitMQ 3.3.5, Erlang R16B03-1
Log out

Overview | Connections | Channels | Exchanges | Queues | Admin

Overview

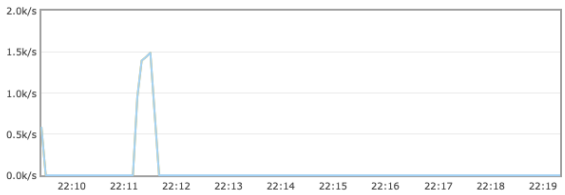
Totals

Queued messages (chart: last ten minutes) (?)



Ready 0 msg
Unacknowledged 0 msg
Total 0 msg

Message rates (chart: last ten minutes) (?)



Publish 0.00/s
Deliver (noack) 0.00/s

Global counts (?)

Connections: 2 | Channels: 2 | Exchanges: 8 | Queues: 1 | Consumers: 20

Nodes

Name	File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Uptime	Type
rabbit@ip-172-31-30-164	28 (65535 available)	3 (58889 available)	204 (1048576 available)	44MB (381MB high watermark)	5.6GB (48MB low watermark)	24m 47s	Disc Stats

Ports and contexts

Listening ports

Protocol	Bound to	Port
amqp	::	5672
clustering	::	25672

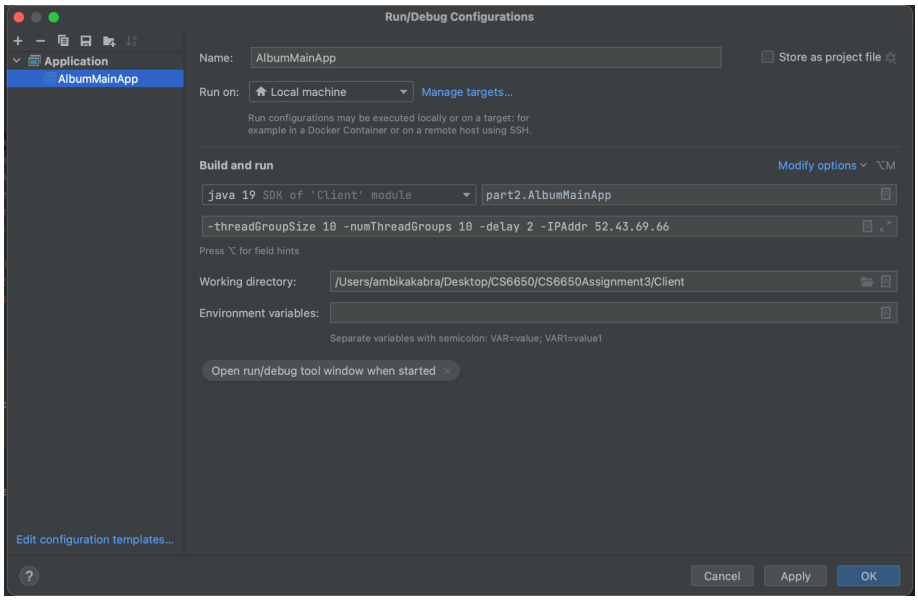
Web contexts

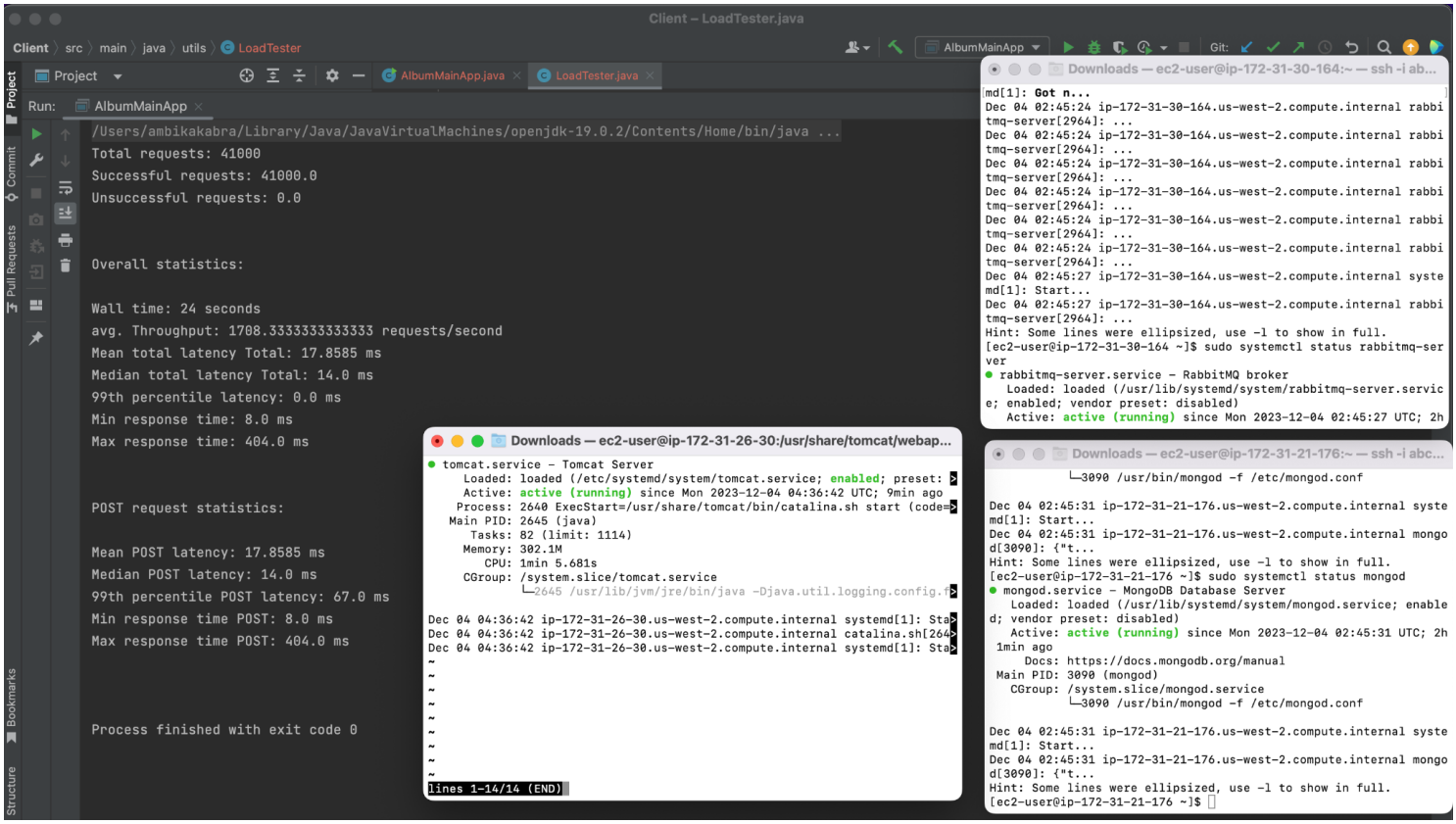
Context	Bound to	Port	SSL	Path
RabbitMQ Management	0.0.0.0	15672	○	/

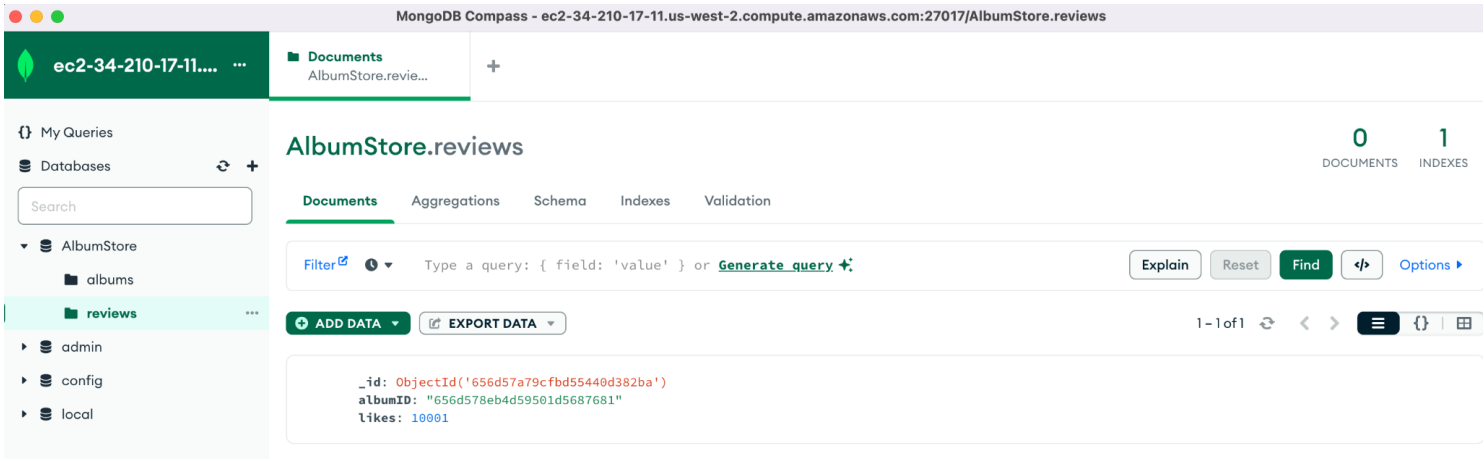
Import / export definitions

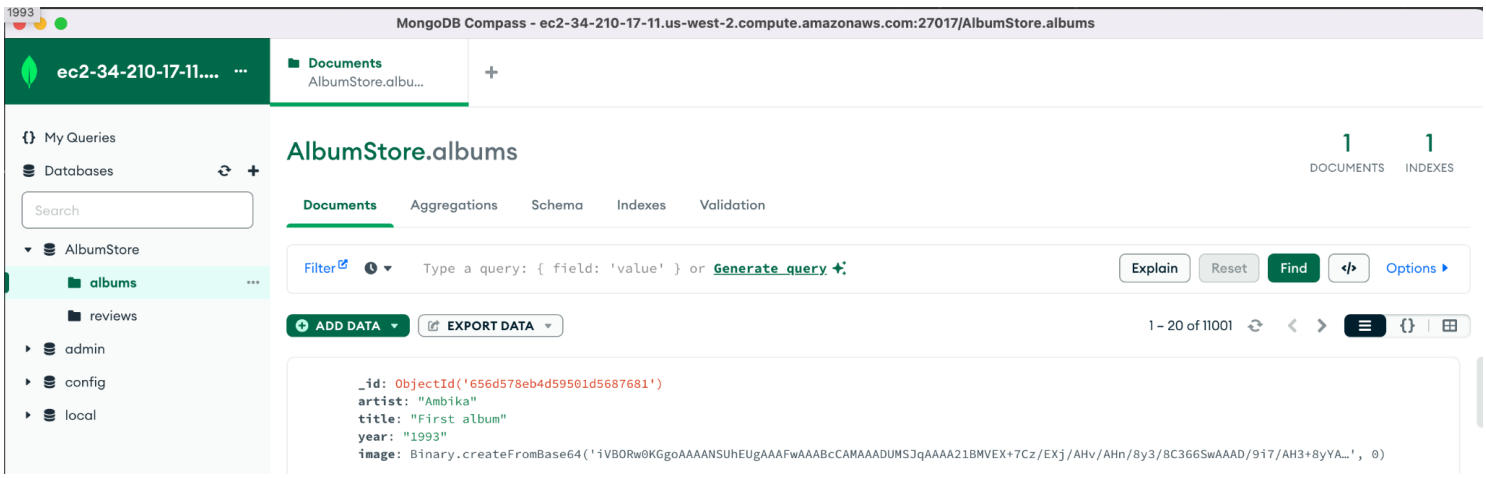
HTTP API | Command Line

Update every 5 seconds

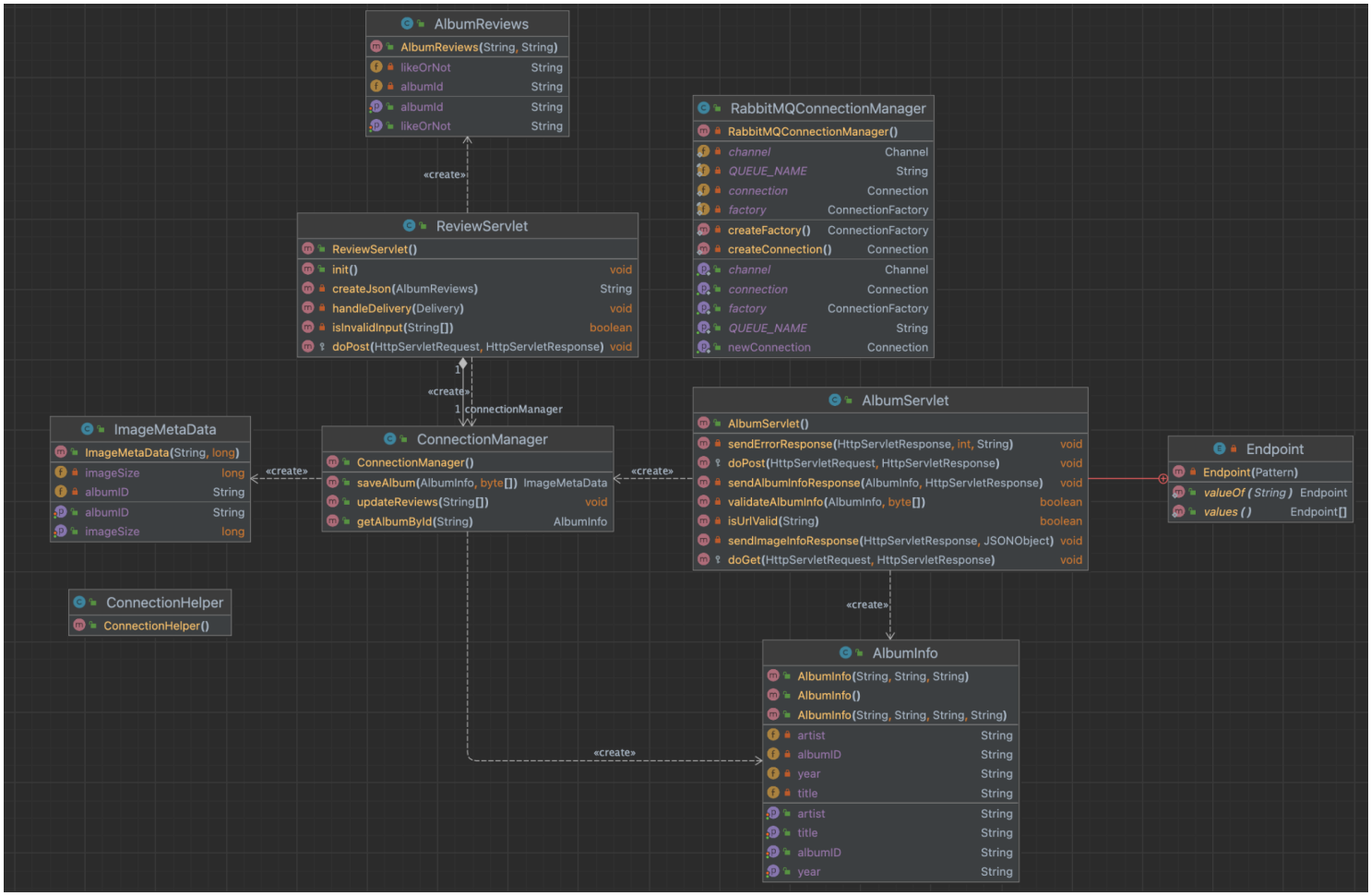



Client – LoadTester.java
Run: AlbumMainApp
/Users/ambikakabra/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/bin/java ...
Total requests: 41000
Successful requests: 41000.0
Unsuccessful requests: 0.0
Overall statistics:
Wall time: 24 seconds
avg. Throughput: 1708.3333333333333 requests/second
Mean total latency Total: 17.8585 ms
Median total latency Total: 14.0 ms
99th percentile latency: 0.0 ms
Min response time: 8.0 ms
Max response time: 404.0 ms
POST request statistics:
Mean POST latency: 17.8585 ms
Median POST latency: 14.0 ms
99th percentile POST latency: 67.0 ms
Min response time POST: 8.0 ms
Max response time POST: 404.0 ms
Process finished with exit code 0
Downloads — ec2-user@ip-172-31-26-30:/usr/share/tomcat/webap...
tomcat.service - Tomcat Server
Loaded: loaded (/etc/systemd/system/tomcat.service; enabled; preset:
Active: active (running) since Mon 2023-12-04 04:36:42 UTC; 9min ago
Process: 2640 ExecStart=/usr/share/tomcat/bin/catalina.sh start (code=
Main PID: 2645 (java)
Tasks: 82 (limit: 1114)
Memory: 302.1M
CPU: 1min 5.681s
CGroup: /system.slice/tomcat.service
└2645 /usr/lib/jvm/jre/bin/java -Djava.util.logging.config.f
Dec 04 04:36:42 ip-172-31-26-30.us-west-2.compute.internal systemd[1]: Sta
Dec 04 04:36:42 ip-172-31-26-30.us-west-2.compute.internal catalina.sh[264
Dec 04 04:36:42 ip-172-31-26-30.us-west-2.compute.internal systemd[1]: Sta
lines 1-14/14 (END)
Downloads — ec2-user@ip-172-31-30-164:~ — ssh -i ab...
Dec 04 02:45:24 ip-172-31-30-164.us-west-2.compute.internal rabbitmq-server[2964]: ...
Dec 04 02:45:27 ip-172-31-30-164.us-west-2.compute.internal systemd[1]: Start...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-30-164 ~]$ sudo systemctl status rabbitmq-server
rabbitmq-server.service - RabbitMQ broker
Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2023-12-04 02:45:27 UTC; 2h
Downloads — ec2-user@ip-172-31-21-176:~ — ssh -i abc...
└3090 /usr/bin/mongod -f /etc/mongod.conf
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal systemd[1]: Start...
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal mongod[3090]: {"t...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-21-176 ~]$ sudo systemctl status mongod
mongod.service - MongoDB Database Server
Loaded: loaded (/usr/lib/systemd/system/mongod.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2023-12-04 02:45:31 UTC; 2h 1min ago
Docs: https://docs.mongodb.org/manual
Main PID: 3090 (mongod)
CGroup: /system.slice/mongod.service
└3090 /usr/bin/mongod -f /etc/mongod.conf
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal systemd[1]: Start...
Dec 04 02:45:31 ip-172-31-21-176.us-west-2.compute.internal mongod[3090]: {"t...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-21-176 ~]$


MongoDB Compass - ec2-34-210-17-11.us-west-2.compute.amazonaws.com:27017/AlbumStore.reviews
ec2-34-210-17-11....
Documents
AlbumStore.revie...
My Queries
Databases
Search
AlbumStore
albums
reviews
admin
config
local
AlbumStore.reviews
0 DOCUMENTS
1 INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 1 of 1
_id: ObjectId('656d57a79cfbd55440d382ba')
albumID: "656d578eb4d59501d5687681"
likes: 10001


1993
MongoDB Compass - ec2-34-210-17-11.us-west-2.compute.amazonaws.com:27017/AlbumStore.albums
ec2-34-210-17-11....
Documents
AlbumStore.albu...
My Queries
Databases
Search
AlbumStore
albums
reviews
admin
config
local
AlbumStore.albums
1 DOCUMENTS
1 INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 20 of 11001
_id: ObjectId('656d578eb4d59501d5687681')
artist: "Ambika"
title: "First album"
year: "1993"
image: Binary.createFromBase64('iVBORw0KGgoAAAANSUhEUgAAAFwAAABcCAMAAADUMSJqAAAA21BMVEX+7Cz/EXj/AHv/AHn/8y3/8C366SwAAAD/9i7/AH3+8yYA…', 0)

Class Diagram of Server implementation



Factory folder contains:

1. ConnectionHelper.java: The ConnectionHelper class is responsible for establishing a connection to a MongoDB database and providing access to collections.
2. ConnectionManager.java: The ConnectionManager class is responsible for interacting with a MongoDB database to fetch album information and save new albums and its reviews.
3. RabbitMQConnectionManager.java: Manages RabbitMQ connections and channels for communication.

Model folder contains:

1. AlbumInfo.java: The AlbumInfo class represents information about an album, including the artist, title, and release year.
2. AlbumReviews.java: Represents album reviews information, including the album ID and whether the user liked or disliked the album.
3. ImageMetaData.java: The ImageMetaData class represents metadata information for an image, including album ID and image size.

Servlet folder contains:

1. AlbumServlet.java: The AlbumServlet class is a servlet for handling HTTP requests related to albums, including retrieving album information and uploading new albums with images.
2. ReviewServlet.java: Servlet for handling album reviews using RabbitMQ for communication.

Changes that will be needed to run in server before deploying war on ec2:

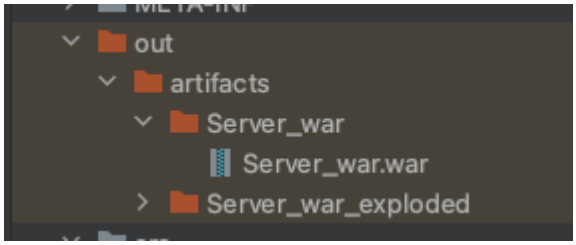
1. Once EC2 for mongodb is up, then change the ec2 public DNS as shown below in ConnectionHelper.connectionString

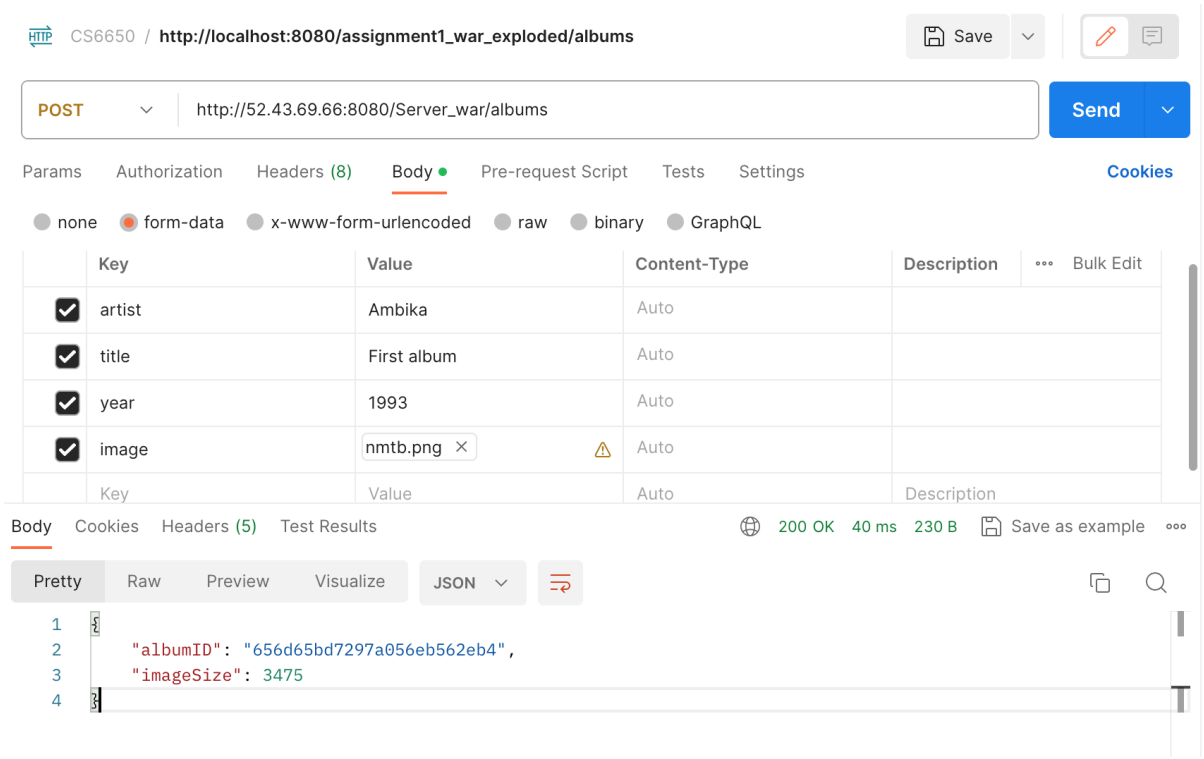
```
3 usages
public class ConnectionHelper {
    1 usage
    private static String connectionString = "mongodb://ec2-34-210-17-11.us-west-2.compute.amazonaws.com:27017/?directConnection=true&s
```

2. Another EC2 for RabbitMQ is up , then change then EC2 public DNS as shown below in RabbitMQConnectionManager.HOST_NAME

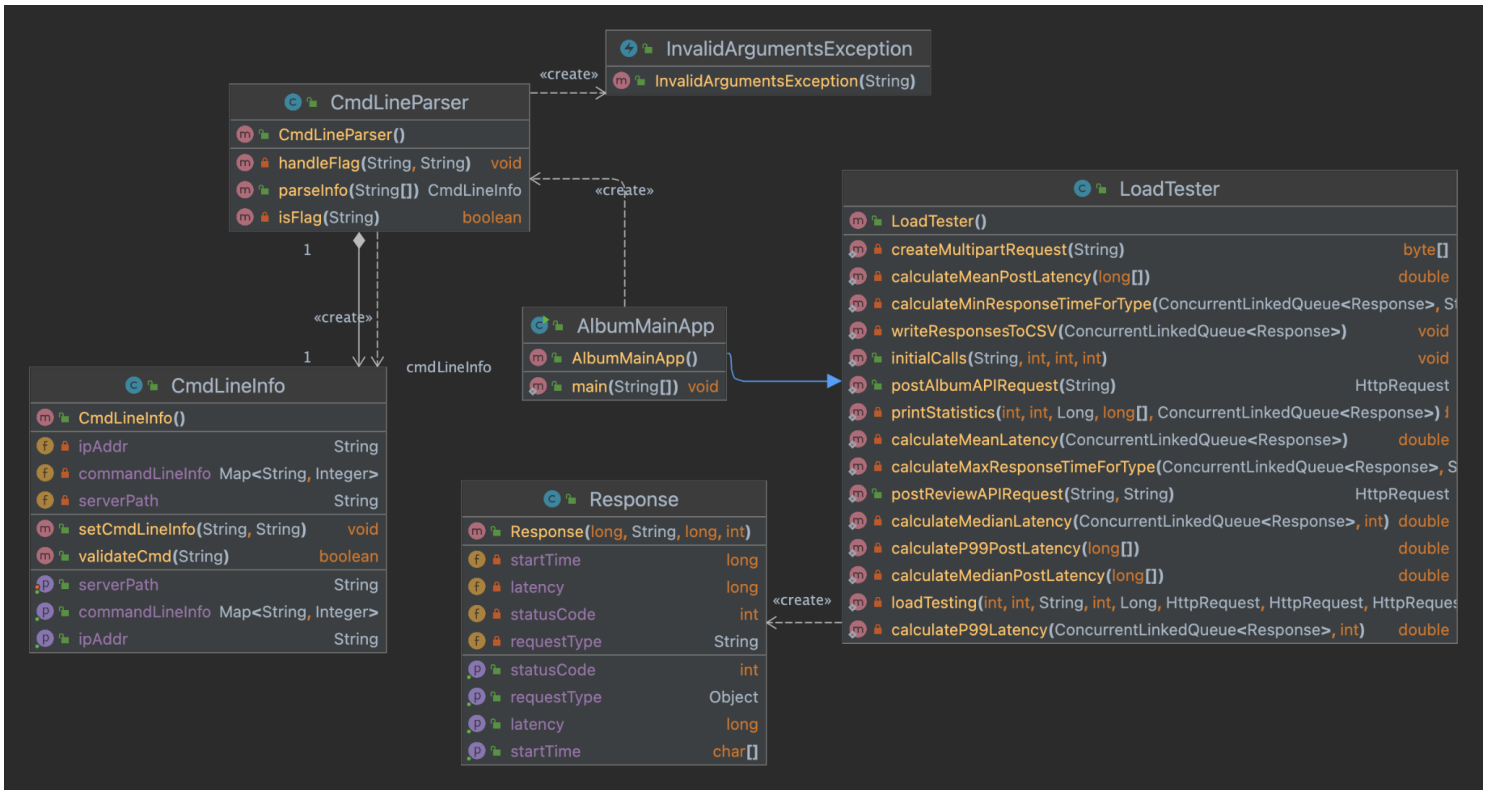
```
/** The default RabbitMQ server host name. */
1 usage
private static final String HOST_NAME = "ec2-54-69-76-218.us-west-2.compute.amazonaws.com";
```

3. Create a Database in your database named “AlbumStore” and a collection named “albums” which is required.
4. Now, Build project and create Artifact “Server_war.war” and copy Server_war.war on the third EC2 to deploy. Now, copy public IP4 address of this EC2 which will be like this:





Class diagram for Client Implementation



Changes that needs to be done in Client before running the load:

- Save an album via postman first before starting for load and copy the albumID and provide that valid albumID here in LoadTester.java in function of `postReviewAPIRequest`

LoadTester.java

```
277            "Content-Disposition: form-data; name=\"title\"" + CRLF +
278            CRLF +
279            "first album" + CRLF +
280            boundaryLine + CRLF +
281            "Content-Disposition: form-data; name=\"year\"" + CRLF +
282            CRLF +
283            "1993" + CRLF +
284            boundaryLine + CRLF +
285            "Content-Disposition: form-data; name=\"image\"; filename=\"nmtb.png\"" + CRLF +
286            "Content-Type: image/png" + CRLF +
287            CRLF;
288
289        String endBoundary = CRLF + boundaryLine + "--" + CRLF;
290        String fullRequestBody = requestBody + "FileContentHere" + endBoundary;
291
292        return fullRequestBody.getBytes(StandardCharsets.UTF_8);
293    }
294
            2 usages   Kabra, Ambika
295    public static HttpRequest postReviewAPIRequest(String IPAddr, String likeornot) {
296        HttpRequest postReviewRequest = HttpRequest.newBuilder()
297            .uri(URI.create(IPAddr + "/Server_war/review/"+likeornot+"/656d65bd7297a056eb562eb4"))
298            .POST(BodyPublishers.noBody())
299            .build();
300
301        return postReviewRequest;
302    }
303 }
304
```

AlbumMainApp.java: This is a main entry point of the application. To run successfully provide all the args.

A short description of data model

Database: **MongoDB** is used as the database to store the album information.
Database Name: The database is named "AlbumStore"
Collections:

- The album data is stored in a collection named "albums".
  The document in this collection has the following fields:
  - _id: This field is a unique identifier for the document. For example, the _id field has the value new ObjectId("654585d35b2fb608340e0afd")
  - artist: The "artist" field contains the name of the artist. For example, it is "Ambika."
  - title: The "title" field contains the title of the album. The album's title is "First album."
  - year: The "year" field contains the year of release for the album. For example, the album was released in "1993."
  - image: The "image" field appears to store binary data representing an image. The image data is provided in Base64 format. The actual image size can be determined by decoding the Base64 string to obtain the binary image data. The image size will depend on the original image file that was encoded as Base64.

```
_id: ObjectId('656ccf98e249e713f31baadc')
artist: "Ambika"
title: "First album"
year: "1993"
image: Binary.createFromBase64('iVBORw0KGgoAAAANSUhEUgAAAFwAAABcCAMAAADUMSJqAAAA21BMVEX+7…
```

- Reviews data is stored in a collection named “reviews”.
  The document in this collection has the following fields:
  a. _id: This field is a unique identifier for the document. For example, the _id field has the value new ObjectId("654585d35b2fb608340e0afd")
  b. albumID: album id mentioned in albums collection
  c. Likes: Number of likes

```
1  _id: ObjectId('656c32f28579229beb6f599c')        ObjectId
2  albumID: "656c32df8ba5eb4622f4d5bd"              String
3  likes: 10000                                     Int32
```