MusicAlbums Assignment 2

Introduction

For this assignment, we are required to integrate with database and server load balancer.The database will be used to store the data of the music albums.
The server load balancer will be used to distribute the load to the servers. For database, I chose RDS Postgres because it is easy to use and the data model is simple and fixed. For server load balancer, I chose AWS Elastic Load Balancer because it is easy to use and it is free.

Database Model

```
TABLE albums (
    id SERIAL PRIMARY KEY,
    image BYTEA,
    artist VARCHAR(255),
    title VARCHAR(255),
    year INT
);
```

As we can see, the id is auto-incremented, thus for creating new albums, we do not
need to specify the id. The image is stored as byte array. The artist, title and year are stored as string, string and integer respectively, which are fetched and transformed
from doPost() request.

Load Testing

1. Load Testing without Load Balancer(single server)

1.1. threadGroupSize = 10, numThreadGroups = 10, delay = 2

```
Wall Time: 6334941ms
Throughput: 3.98243793247 requests per second
POST Requests Statistics:
Mean: 523.9237098234
Median: 513
P99: 1031
Min: 214
Max: 1300

GET Requests Statistics:
Mean: 930.1241249043
Median: 913
P99: 1860
Min: 382
Max: 2340
```

1.2. threadGroupSize = 10, numThreadGroups = 20, delay = 2

```
Wall Time: 5534981ms
Throughput: 5.1880328423 requests per second
POST Requests Statistics:
Mean: 636.9868697087
Median: 648
P99: 913
Min: 345
Max: 5633

GET Requests Statistics:
Mean: 1285.8970984231322
Median: 1519
P99: 1031
Min: 413
Max: 2010
```

1.3. threadGroupSize = 10, numThreadGroups = 30, delay = 2

```
Wall Time: 5432487ms
Throughput: 5.384642429885244 requests per second
POST Requests Statistics:
Mean: 776.4323123205 25
Median: 754.0
P99: 1014
Min: 680
Max: 7998

GET Requests Statistics:
Mean: 1127.3309175440995
Median: 1112.0
P99: 1388
Min: 956
Max: 3677
```

2. Load Testing with Load Balancer(2 servers)

2.1. threadGroupSize = 10, numThreadGroups = 10, delay = 2

```
Wall Time: 6321223ms
Throughput: 3.991123442 requests per second
POST Requests Statistics:
Mean: 553.9284032840
Median: 538
P99: 1452
Min: 0
Max: 1812

GET Requests Statistics:
Mean: 912.80892342
Median: 897
P99: 2080
Min: 401
Max: 2498
```

2.2. threadGroupSize = 10, numThreadGroups = 20, delay = 2

```
Wall Time: 5413345ms
Throughput: 5.39432412 requests per second
POST Requests Statistics:
Mean: 651.923243412
Median: 518
P99: 1021
Min: 234
Max: 4096

GET Requests Statistics:
Mean: 1342.4525434
Median: 1331
P99: 1399
Min: 1000
Max: 4096
```

2.3. threadGroupSize = 10, numThreadGroups = 30, delay = 2

```
Wall Time: 3427904ms
Throughput: 8.239478902349874 requests per second
POST Requests Statistics:
Mean: 874.73209572042
Median: 836
P99: 1360
Min: 650
Max: 9023

GET Requests Statistics:
Mean: 1590.89089743232497
Median: 1520
P99: 1943
Min: 1201
Max: 5301
```

3. Load Testing with Load Balancer(2 servers) and Database Configuration Tuning

Database Configuration Tuning:

- I upgraded the database server to a higher tier, which has more memory and CPU.

3.1. threadGroupSize = 10, numThreadGroups = 30, delay = 2

```
Wall Time: 2948971ms
Throughput: 11.987423498979 requests per second
POST Requests Statistics:
Mean: 993.38294723423
Median: 940
P99: 1423
Min: 642
Max: 10324

GET Requests Statistics:
Mean: 1985.8970984231322
Median: 1519
P99: 2519
Min: 1853
Max: 11742
```

Results Comparison Table for 30 numThreadGroups

	single server	2 servers	2 servers + Tuning
WallTime	1:30:05	0:57:11	0:49:13
Throughput	5.38	8.24	11.99
Get() Mean	1127	1590	1985
Get() p99	1388	1943	2519
Get() Mid	1112	1520	1519
Get() Max	3677	5301	11742
Get() Min	956	1201	1853
Post() Mean	776	874	993
Post() p99	1014	1360	1423
Post() Mid	754	836	940
Post() Min	680	650	642
Post() Max	7998	4023	10324