Task 1

Connection Pooling

We changed the content in context.xml (picture 1) and web.xml (picture 2) under WebContent folder as follows to enable connection pooling. So the servlet that needs to communicate with database will get connection using information from this data source to extract data.

```
15      <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
16                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
17                password="mypassword" driverClassName="com.mysql.jdbc.Driver"
18                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
```

```
 9       <resource-ref>
10         <description>
11             Resource reference to a factory for java.sql.Connection
12             instances that may be used for talking to a particular
13             database that
14             is configured in the server.xml file.
15         </description>
16         <res-ref-name>jdbc/moviedb</res-ref-name>
17         <res-type>javax.sql.DataSource</res-type>
18         <res-auth>Container</res-auth>
19      </resource-ref>
```

Below is a simple example that we modify our implementation to use connection pooling. This is mainPage/searchHelper.java. We also modify all other java classes that need to contact the database in a similar way. All changed java classes include:

checkout/CheckoutServlet.java	68-75
checkout/ConfirmationServlet.java	43-50
employee/DashboardServlet.java	79-86
employee/EmployeeLoginServlet.java	76-83
login/LoginVerifyUtils.java	25-32
mainPage/AdvancedSearchServlet.java	54-61
mainPage/BrowsingServlet.java	54-61
mainPage/IndexServlet.java	62-69
mainPage/SearchHelper.java	26-33
mainPage/SearchSuggetion.java	50-57
mainPage/SingleMoviePage.java	55-62
mainPage/SingleStarPage.java	48-55

```
25          try {
26              Context initCtx = new InitialContext();
27
28              Context envCtx = (Context) initCtx.lookup("java:comp/env");
29
30              // Look up our data source
31              DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
32
33              Connection dbcon = ds.getConnection();
34
```

Prepared Statement

We started to use prepared statements from project 3 and the screenshot below is one example from login/LoginVerifyUtils.java. You can check all classes with prepared statements in Connection Pooling section. Those classes use prepared statements.

```
34              String query = "SELECT id, email, password FROM customers WHERE email=?";
35              PreparedStatement statement = dbcon.prepareStatement(query);
36              statement.setString(1, username);
37              // execute query
38              ResultSet resultSet = statement.executeQuery();
```

Task 2

AWS address (All following public ip addresses are subject to change)
Instance 1: 52.53.170.122
Master: 18.144.18.87
Slave: 13.57.185.152

Google Cloud: 10.142.0.2 (internal)

- Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port? **Yes.**

Besides the snapshot provided in Task 1, we added another data source to context.xml (picture 1) and web.xml (picture 2) under WebContent folder to enable connection pooling with two data source. All read requests are using the one provided in Task 1 (jdbc/moviedb), and all write requests are using the following one (master/moviedb). We set the jdbc/moviedb to localhost, and master/moviedb to the public ip of master. This makes sure that all write requests will go to the master, and read requests are distributed between master and slave.

```
20      <Resource name="master/moviedb" auth="Container" type="javax.sql.DataSource"
21                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
22                password="mypassword" driverClassName="com.mysql.jdbc.Driver"
23                url="jdbc:mysql://18.144.18.87:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
```

```
21    <resource-ref>
22        <description>
23            Resource reference to a factory for java.sql.Connection
24            instances that may be used for talking to a particular
25            database that
26            is configured in the server.xml file.
27        </description>
28        <res-ref-name>master/moviedb</res-ref-name>
29        <res-type>javax.sql.DataSource</res-type>
30        <res-auth>Container</res-auth>
31    </resource-ref>
```

Here is one example that makes sure all write requests are sent to master (employee/DashboardServlet.java)

```
79              Context initCtx = new InitialContext();
80
81              Context envCtx = (Context) initCtx.lookup("java:comp/env");
82
83              // Look up our data source
84              DataSource ds = (DataSource) envCtx.lookup("master/moviedb");
85
86              Connection dbcon = ds.getConnection();
```

Read requests are just as before through localhost. Master and slave will serve in round robin order.

```
25          try {
26              Context initCtx = new InitialContext();
27
28              Context envCtx = (Context) initCtx.lookup("java:comp/env");
29
30              // Look up our data source
31              DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
32
33              Connection dbcon = ds.getConnection();
34
```

Below is the setting of routing for load balancer and enabling sticky session to serve all requests from one user on a single server.

```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED

<Proxy "balancer://Fabflix_balancer">
    BalancerMember "http://172.31.29.199:8080/cs122b-spring18-team-90/" route=1
    BalancerMember "http://172.31.30.108:8080/cs122b-spring18-team-90/" route=2
ProxySet stickysession=ROUTEID
</Proxy>
```

Task 3

- **Have you uploaded the log file to Github? Where is it located?**

Yes. Under Project5_Log_Graph directory. All of log files are separated into single-instance case and scaled-version case. File names are self-explanatory.

- **Have you uploaded the HTML file to Github? Where is it located?**

Yes. It's at the root directory on Github and the file name is jmeter_report.html

- **Have you uploaded the script to Github? Where is it located?**

Yes. It's at the root directory on Github and the file name is parse.py. Remember to revise the log file path.

- **Have you uploaded the WAR file and README to Github? Where is it located?**

  Yes. It's It's at the root directory on Github