Hanjiao Qiu
CSCIE63 Big Data Analytics
Assignment 07

Assignment 07 Solution

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used as separate files. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. Please, submit to the class drop box. For issues and comments visit the class Discussion Board. The following problems are formulated in Java, however you can solve the following problems using any language of your choice that is supported by Cassandra Client API-s. You are not obliged to use Eclipse. You are welcome to use any IDE of your choice.

Problem 1) Install Cassandra server on your Cloudera VM. Use one of the methods described in notes. Use Cassandra SQL Client, `cqlsh`, to create and populate table `person`. Let every `person` by described by his or her first and last name, and city where he or she lives. Let every person possess up to three cell phones. Populate your table with three individuals using `cqlsh` client. Demonstrate that you can select the content of your table `person`.

Solution

1. Install Cassandra server on my Cloudera VM. First check the version of current Operating System. The OS on Cloudera VM is CentOS 6.4. Since Cassandra server should be run by Linux user cassandra, we create this account, give it sudo permission and setup the environment variables.

```
root@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ cat /etc/issue
CentOS release 6.4 (Final)
Kernel \r on an \m

[cloudera@quickstart ~]$ cat /etc/centos-release
CentOS release 6.4 (Final)
```

```
[cloudera@quickstart ~]$ su -
Password:
[root@quickstart ~]# groupadd cassandra
[root@quickstart ~]# useradd cassandra -g cassandra
[root@quickstart ~]# passwd cassandra
Changing password for user cassandra.


[root@quickstart ~]# chmod +w /etc/sudoers
[root@quickstart ~]# visudo -f /etc/sudoers
```

```
## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
cloudera ALL=(ALL) NOPASSWD: ALL
root ALL=(ALL) NOPASSWD: ALL
cassandra ALL=(ALL) NOPASSWD:ALL

Defaults env_keep += "LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY"
Defaults env_keep += "JAVA_HOME PATH"
-- INSERT --
```

We can use the binary tarball to install Cassandra. We can download Cassandra 2.1.13 from http://cassandra.apache.org/download/. Untar the tar.gz file and store the path into file "bash_profile". Source "bash_profile" again after edit.

Apache Cassandra 2.1 is supported until November 2016 with critical fixes only. The latest release is 2.1.13, released on 2016-02-08.

- apache-cassandra-2.1.13-bin.tar.gz [PGP] [MD5] [SHA1]
- Debian installation instructions

```
apache-cassandra-2.1.13/tools/bin/sstablerepairedset
apache-cassandra-2.1.13/tools/bin/sstablesplit
apache-cassandra-2.1.13/tools/bin/sstablesplit.bat
apache-cassandra-2.1.13/tools/bin/token-generator
apache-cassandra-2.1.13/tools/bin/token-generator.bat
[cassandra@quickstart ~]$ mv apache-cassandra-2.1.13 cassandra-2.1.13
[cassandra@quickstart ~]$ vi ~/.bash_profile
[cassandra@quickstart ~]$ source .bash_profile
```

```
# User specific environment and startup programs

PATH=$PATH:$HOME/bin

CASSANDRA_PATH=/home/cassandra/cassandra-2.1.13
export CASSANDRA_PATH

PATH=$PATH:$CASSANDRA_PATH/bin
export PATH
```

2. To start Cassandra server, run as Linux user cassandra. Since we have already source the path "cassandra-2.1.13/bin", we can directly call "cassandra -f" to start Cassandra server. The switch "-f" tells Cassandra to run in foreground. We can see all the server logs being printed out to the terminal window.

```
[cloudera@quickstart ~]$ su - cassandra
Password:
[cloudera@quickstart ~]$ cassandra -f
```

```
[cloudera@quickstart ~]$ su - cassandra
Password:
```

```
[cassandra@quickstart ~]$ cassandra -f
CompilerOracle: inline org/apache/cassandra/db/AbstractNativeCell.compareTo (Lorg/apache/cassa
ndra/db/composites/Composite;)I
CompilerOracle: inline org/apache/cassandra/db/composites/AbstractSimpleCellNameType.compareUn
signed (Lorg/apache/cassandra/db/composites/Composite;Lorg/apache/cassandra/db/composites/Comp
osite;)I
CompilerOracle: inline org/apache/cassandra/io/util/Memory.checkBounds (JJ)V
CompilerOracle: inline org/apache/cassandra/io/util/SafeMemory.checkBounds (JJ)V
CompilerOracle: inline org/apache/cassandra/utils/ByteBufferUtil.compare (Ljava/nio/ByteBuffer
;[B)I
CompilerOracle: inline org/apache/cassandra/utils/ByteBufferUtil.compare ([BLjava/nio/ByteBuff
er;)I
```

```
em/local-7ad54392bcdd35a684174e047860b377/system-local-ka-5,].  5,916 bytes to 5,748 (~97% of
original) in 457ms = 0.011995MB/s.  4 total partitions merged to 1.  Partition merge counts we
re {4:1, }
INFO  01:37:52 Using Netty Version: [netty-buffer=netty-buffer-4.0.23.Final.208198c, netty-cod
ec=netty-codec-4.0.23.Final.208198c, netty-codec-http=netty-codec-http-4.0.23.Final.208198c, n
etty-codec-socks=netty-codec-socks-4.0.23.Final.208198c, netty-common=netty-common-4.0.23.Fina
l.208198c, netty-handler=netty-handler-4.0.23.Final.208198c, netty-transport=netty-transport-4
.0.23.Final.208198c, netty-transport-rxtx=netty-transport-rxtx-4.0.23.Final.208198c, netty-tra
nsport-sctp=netty-transport-sctp-4.0.23.Final.208198c, netty-transport-udt=netty-transport-udt
-4.0.23.Final.208198c]
INFO  01:37:52 Starting listening for CQL clients on localhost/127.0.0.1:9042...
INFO  01:37:52 Binding thrift service to localhost/127.0.0.1:9160
INFO  01:37:52 Listening for thrift clients...
```

3. Start the Cassandra command line interface to examine data in Cassandra server. Run "./cqlsh" to connect to the local Cassandra instance.

```
[cloudera@quickstart ~]$ su - cassandra
Password:
[cassandra@quickstart ~]$ echo $CASSANDRA_PATH
/home/cassandra/cassandra-2.1.13
[cassandra@quickstart ~]$ cd $CASSANDRA_PATH/bin

[cassandra@quickstart bin]$ ./cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.1.13 | CQL spec 3.2.1 | Native protocol v3]
Use HELP for help.

[cassandra@quickstart ~]$ echo $CASSANDRA_PATH
/home/cassandra/cassandra-2.1.13
[cassandra@quickstart ~]$ cd $CASSANDRA_PATH/bin
[cassandra@quickstart bin]$ ls
cassandra          cassandra.in.sh  debug-cql.bat    sstablekeys.bat    sstableupgrade
cassandra.bat      cassandra.ps1    nodetool         sstableloader      sstableupgrade.bat
cassandra-cli      cqlsh            nodetool.bat     sstableloader.bat  stop-server
cassandra-cli.bat  cqlsh.bat        source-conf.ps1  sstablescrub       stop-server.bat
cassandra.in.bat   debug-cql        sstablekeys      sstablescrub.bat   stop-server.ps1
[cassandra@quickstart bin]$ ./cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.1.13 | CQL spec 3.2.1 | Native protocol v3]
Use HELP for help.
cqlsh>
```

4. Create a keyspace "mykeyspace" for the tables. Create and populate table "person" into the keyspace.

```
cqlsh> CREATE KEYSPACE mykeyspace WITH REPLICATION = { 'class' :
'SimpleStrategy', 'replication_factor' : 1};

cqlsh> USE mykeyspace;

cqlsh:mykeyspace> CREATE TABLE person (
              ... id uuid PRIMARY KEY,
              ... fname text,
              ... lname text,
              ... city text,
              ... phone1 text,
              ... phone2 text,
              ... phone3 text);

cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2,
phone3) VALUES (de305d54-75b4-431b-adb2-eb6b9e546001, 'Nick', 'Smith',
'Boston', '(781)397-7171', '(781)393-5303', '(781)286-8925');

cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2,
phone3) VALUES (de305d54-75b4-431b-adb2-eb6b9e546002, 'Joe', 'White',
'Cambridge', '(617)321-1805', '(617)346-1273', '(617)319-1835');

cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2,
phone3) VALUES (de305d54-75b4-431b-adb2-eb6b9e546003, 'Mary', 'Earl', 'Newton',
'(854)317-3715', '(854)652-2482', '(854)592-1753');

cqlsh:mykeyspace> SELECT * FROM person;
```

```
[cassandra@quickstart bin]$ ./cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.1.13 | CQL spec 3.2.1 | Native protocol v3]
Use HELP for help.
cqlsh> CREATE KEYSPACE mykeyspace WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replicati
on_factor' : 1};
cqlsh> USE mykeyspace;
cqlsh:mykeyspace> CREATE TABLE person (
              ... id uuid PRIMARY KEY,
              ... fname text,
              ... lname text,
              ... city text,
              ... phone1 text,
              ... phone2 text,
              ... phone3 text);
cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2, phone3) VALUES
(de305d54-75b4-431b-adb2-eb6b9e546001, 'Nick', 'Smith', 'Boston', '(781)397-7171', '(781)393-
5303', '(781)286-8925');
cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2, phone3) VALUES
(de305d54-75b4-431b-adb2-eb6b9e546002, 'Joe', 'White', 'Cambridge', '(617)321-1805', '(617)34
6-1273', '(617)319-1835');
cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2, phone3) VALUES
(de305d54-75b4-431b-adb2-eb6b9e546003, 'Mary', 'Earl', 'Newton', '(854)317-3715', '(854)652-2
482', '(854)592-1753');
```

```
cqlsh:mykeyspace> SELECT * FROM person;

 id                                   | city      | fname | lname | phone1         | phone2         | phone3
--------------------------------------+-----------+-------+-------+----------------+----------------+----------------
 de305d54-75b4-431b-adb2-eb6b9e546003 |    Newton |  Mary |  Earl | (854)317-3715 | (854)652-2482 | (854)592-1753
 de305d54-75b4-431b-adb2-eb6b9e546002 | Cambridge |   Joe | White | (617)321-1805 | (617)346-1273 | (617)319-1835
 de305d54-75b4-431b-adb2-eb6b9e546001 |    Boston |  Nick | Smith | (781)397-7171 | (781)393-5303 | (781)286-8925

(3 rows)
```

Select the content from the table. The records are inserted correctly.

If we want to filter the data based on columns other than "id", we need to create an index on that column first. Otherwise the operation will fail because the operation would be very inefficient. Below is an example to create an index on "lname" and uses this index to pull out the results.

```
cqlsh:mykeyspace> CREATE INDEX ON person (lname);

cqlsh:mykeyspace> SELECT * FROM person WHERE lname = 'White';
```

```
cqlsh:mykeyspace> CREATE INDEX ON person (lname);
cqlsh:mykeyspace> SELECT * FROM person WHERE lname = 'White';

 id                                   | city      | fname | lname | phone1         | phone2         | phone3
--------------------------------------+-----------+-------+-------+----------------+----------------+----------------
 de305d54-75b4-431b-adb2-eb6b9e546002 | Cambridge |   Joe | White | (617)321-1805 | (617)346-1273 | (617)319-1835

(1 rows)
```

Insert another records and query again.

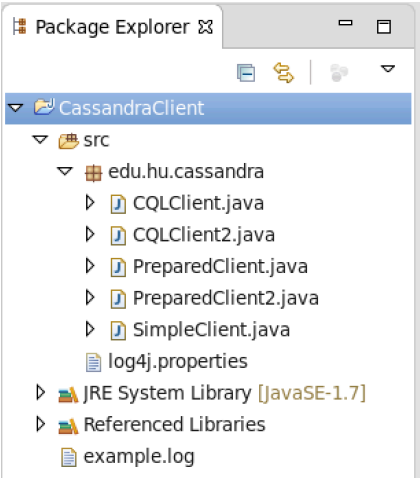
```
cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2, phone3) VALUES
(de305d54-75b4-431b-adb2-eb6b9e546004, 'Lucy', 'White', 'Boston', '(678)955-1345', '(678)213-
3141', '(678)131-1732');

cqlsh:mykeyspace> INSERT INTO person (id, fname, lname, city, phone1, phone2,
phone3) VALUES (de305d54-75b4-431b-adb2-eb6b9e546004, 'Lucy', 'White',
'Boston', '(678)955-1345', '(678)213-3141', '(678)131-1732');

cqlsh:mykeyspace> SELECT * FROM person;

 id                                   | city      | fname | lname | phone1         | phone2         | phone3
--------------------------------------+-----------+-------+-------+----------------+----------------+----------------
 de305d54-75b4-431b-adb2-eb6b9e546003 |    Newton |  Mary |  Earl | (854)317-3715 | (854)652-2482 | (854)592-1753
 de305d54-75b4-431b-adb2-eb6b9e546002 | Cambridge |   Joe | White | (617)321-1805 | (617)346-1273 | (617)319-1835
 de305d54-75b4-431b-adb2-eb6b9e546001 |    Boston |  Nick | Smith | (781)397-7171 | (781)393-5303 | (781)286-8925
 de305d54-75b4-431b-adb2-eb6b9e546004 |    Boston |  Lucy | White | (678)955-1345 | (678)213-3141 | (678)131-1732

(4 rows)
cqlsh:mykeyspace> SELECT * FROM person WHERE lname = 'White';

 id                                   | city      | fname | lname | phone1         | phone2         | phone3
--------------------------------------+-----------+-------+-------+----------------+----------------+----------------
 de305d54-75b4-431b-adb2-eb6b9e546002 | Cambridge |   Joe | White | (617)321-1805 | (617)346-1273 | (617)319-1835
 de305d54-75b4-431b-adb2-eb6b9e546004 |    Boston |  Lucy | White | (678)955-1345 | (678)213-3141 | (678)131-1732

(2 rows)
```

Problem 2) Create an Eclipse project. Move attached class `SimpleClient` into the project. Place attached `log4j.properties` file in the `src` directory of your project. Properly set the `Build Path` of your project. Make sure that Cassandra is started. Run your `SimpleClient` class as a Java Application. Capture console output. It should

basically say that you are running a single machine Cassandra cluster on the host 127.0.0.1.

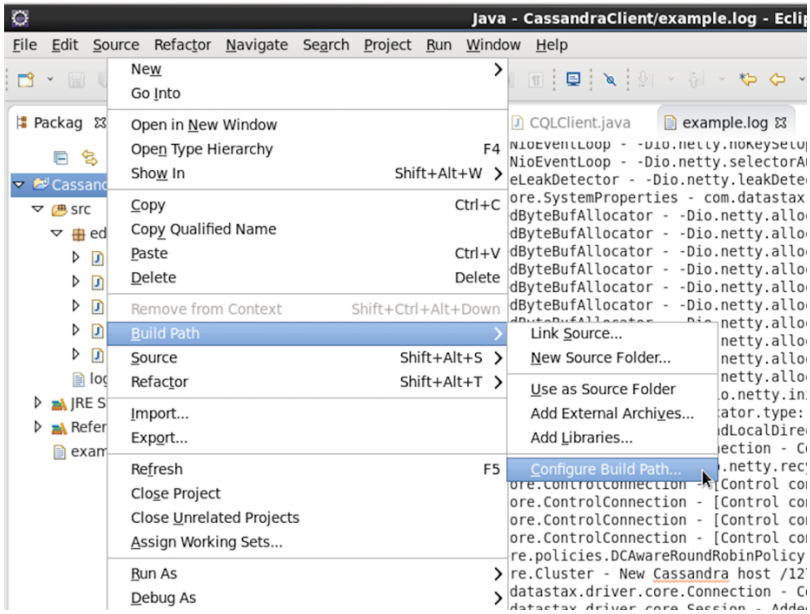
Solution

1. Create a Java project. The file structure is like the following figure. Put the source file "SimpleClient.java" into package "edu.hu.cassandra" and "log4j.properties" into "src" directory.

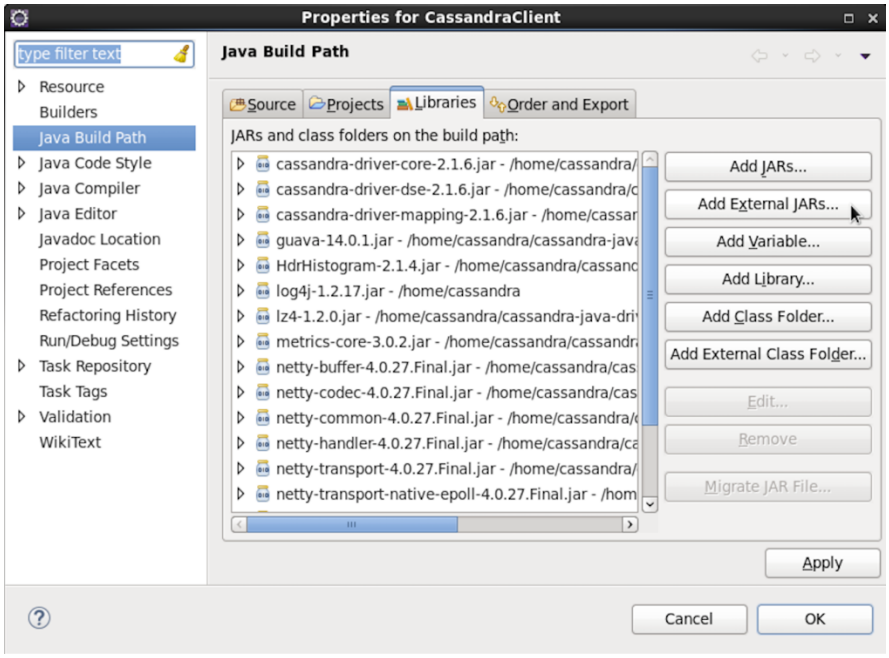


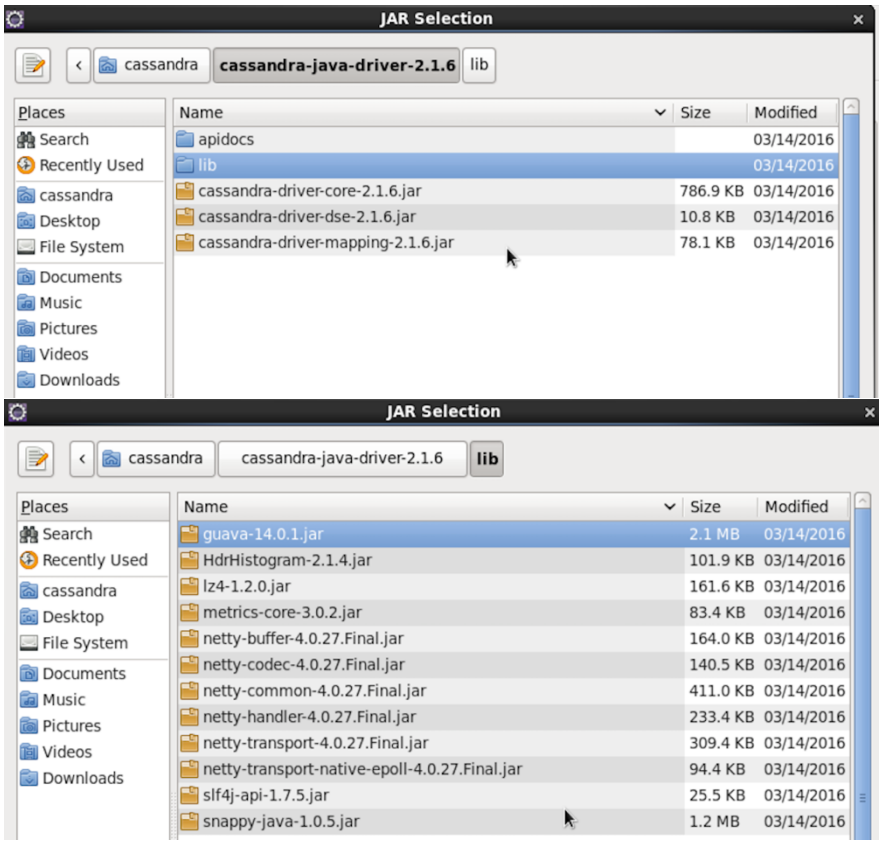
2. Download the binary drive and import the jar files.

Download **Cassandra java driver** from:
http://downloads.datastax.com/java-driver/cassandra-java-driver-2.1.6.tar.gz



Import the jar files from the folder “cassandra-java-driver”. Right click the project name and choose “Build Path -> Configure Build Path”. Then in the pop-up window, choose “Libraries -> Add External JARs”. Choose “**cassandra-driver-core-2.1.6.jar**” and “**cassandra-driver-dse-2.1.6.jar**”. Also, import additional jars from subdirectory “lib”.

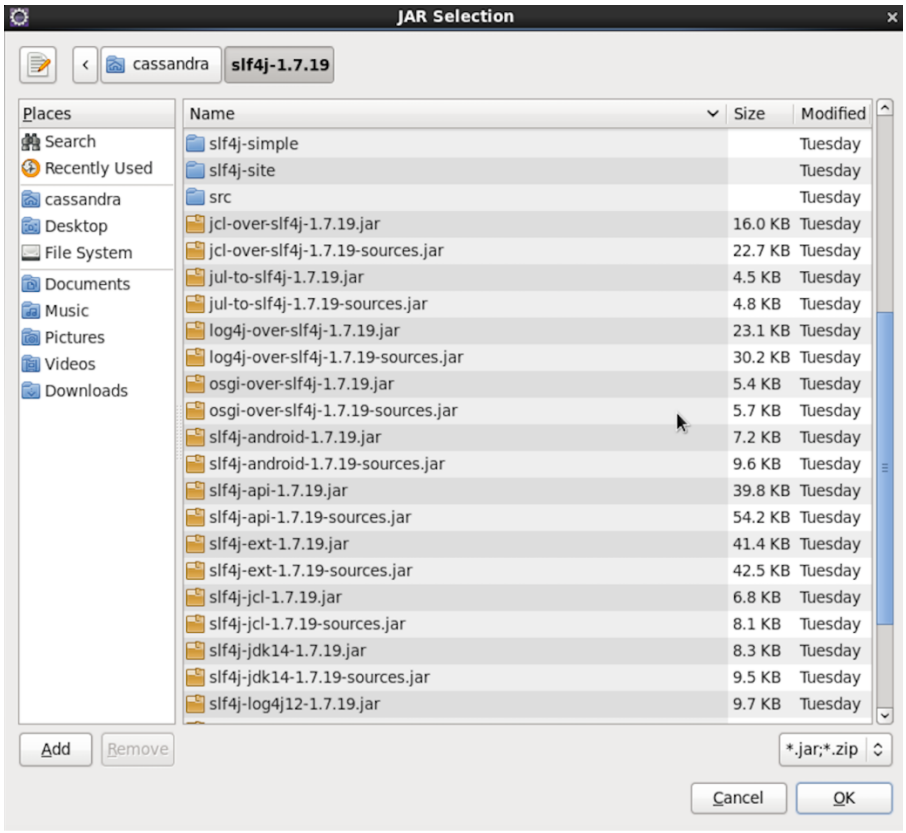


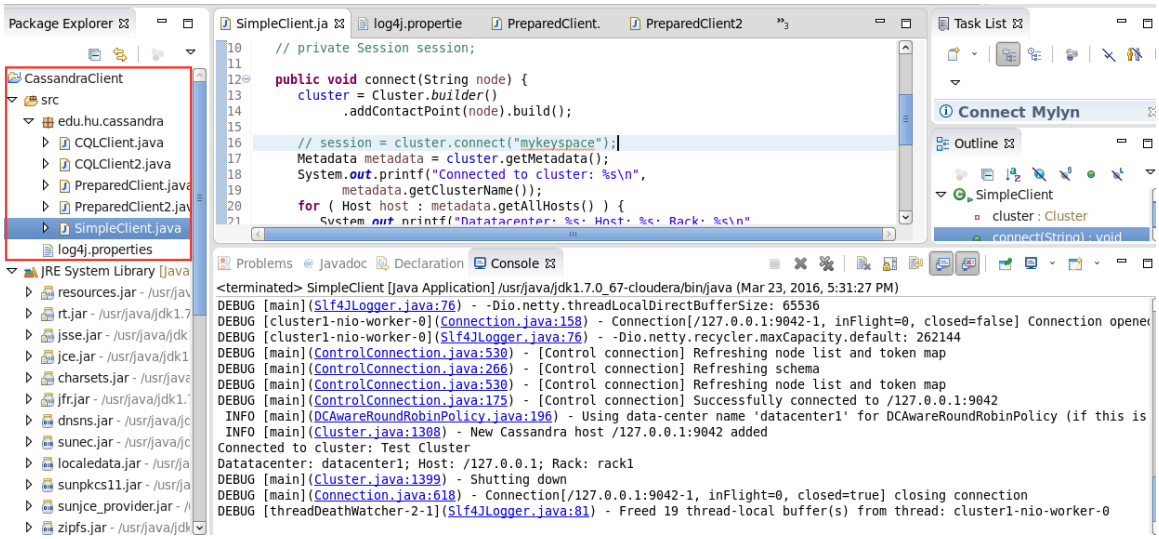


To make the log4j work properly, we also need to download:

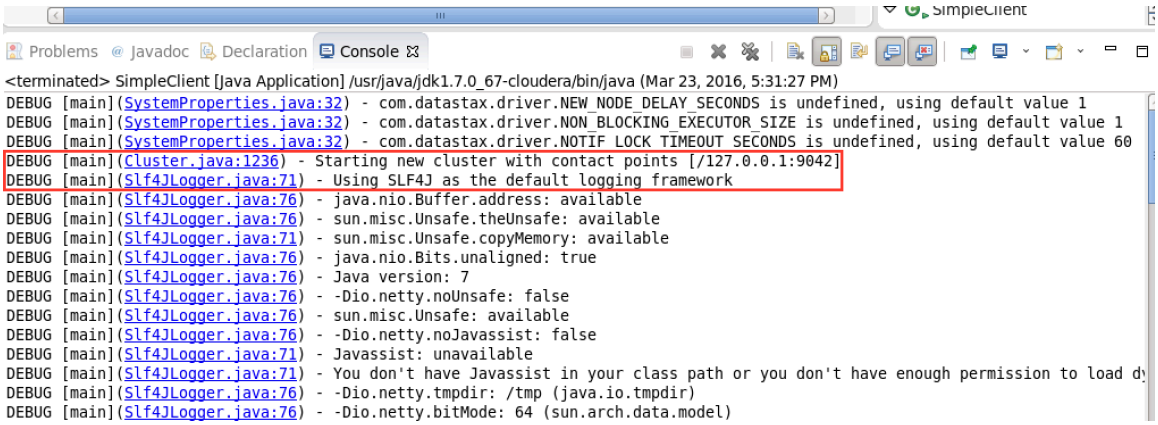
log4j-1.2.17.jar: https://logging.apache.org/log4j/1.2/download.html
slf4j-log4j12-1.7.19.jar: http://www.slf4j.org/download.html

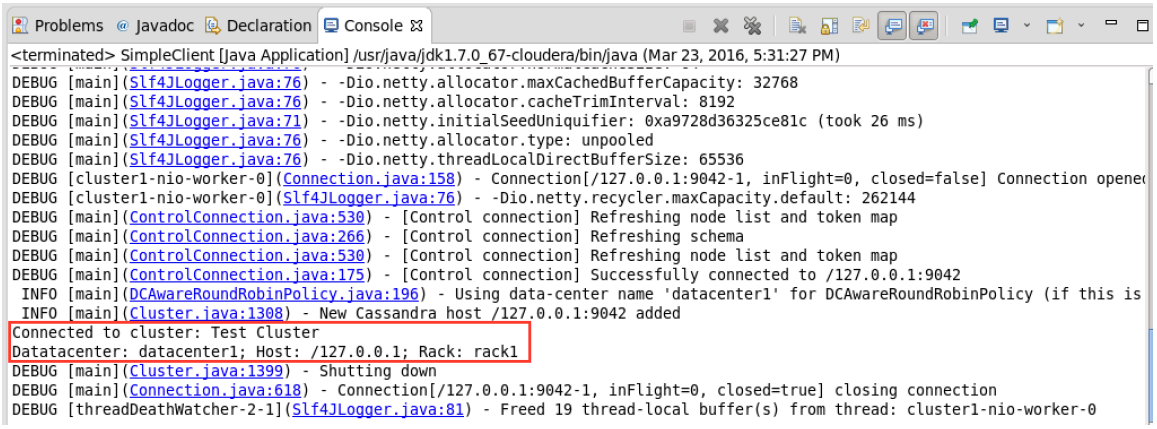
The **slf4j-log4j12-1.7.19.jar** is the binding for log4j version 1.2. It serves as a simple facade or abstraction for various logging frameworks, such as java.util.logging, logback and log4j.





3. Run the Java application. Use current "log4j.properties", all the debug messages will be output to the console. We can see the detailed steps logged by class "Slf4JLogger". The messages show that we can connect to the local Cassandra test cluster.





Problem 3) Write a simple Java client starting from the attached Java class `CQLClient` to your Java project. As you can see this class performs basic CQL operations on your Cassandara database. It opens a session to Cassandra cluster, creates a keyspace, creates new table, inserts and queries some rows in that table. Modify that class so that it creates, populates and queries table `person` introduced in Problem 1. You might want to run this problem in a Cassandra keyspace different from the one created in Problem 1. Modify your `log4j.properties` to stop DEBUG lines from being printed out. Capture all the steps, working code and resulting console outputs. Submit modified `log4j.properties` file, as well.

Solution

1. Open a connection to Cassandra cluster. Create a Cluster object and connect to the instance using the "Cluster.builder()" method. It will add a contact point and build a cluster instance. Get a session from the cluster, connecting to the keyspace. If we already have a keyspace, we can set it to the existing one by passing the

keyspace name in string format. We will create the keyspace later, so just call connect() directly for now.

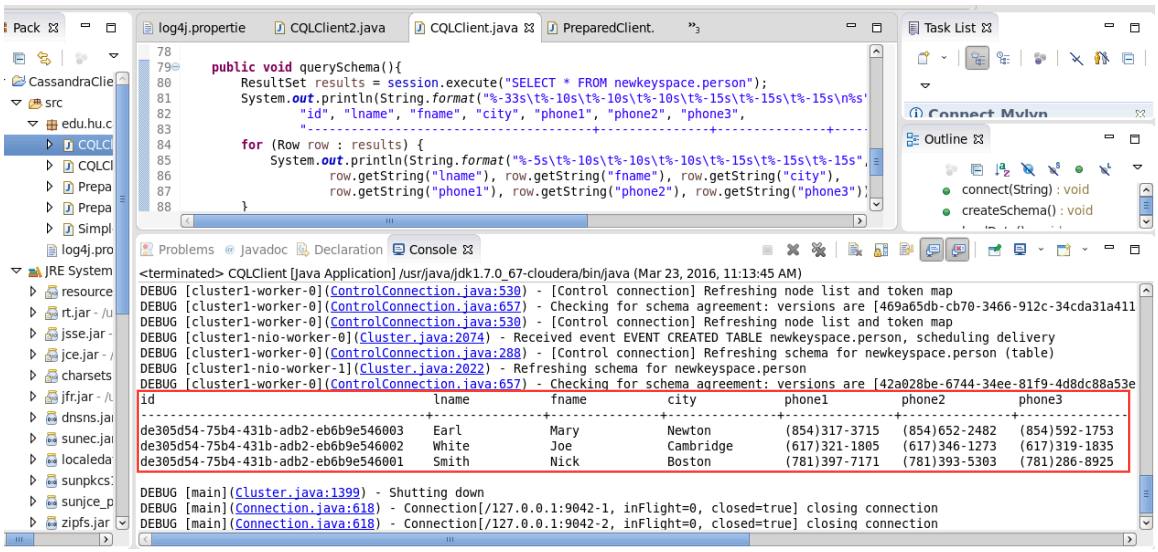
```
public void connect(String node) {
    cluster = Cluster.builder()
            .addContactPoint(node).build();
    session = cluster.connect();
    Metadata metadata = cluster.getMetadata();
    System.out.printf("Connected to cluster: %s\n",
            metadata.getClusterName());
    for ( Host host : metadata.getAllHosts() ) {
        System.out.printf("Datatacenter: %s; Host: %s; Rack: %s\n",
                host.getDatacenter(), host.getAddress(), host.getRack());
    }
}
```

2. Execute query. Create keyspace, table and populate data into the table.

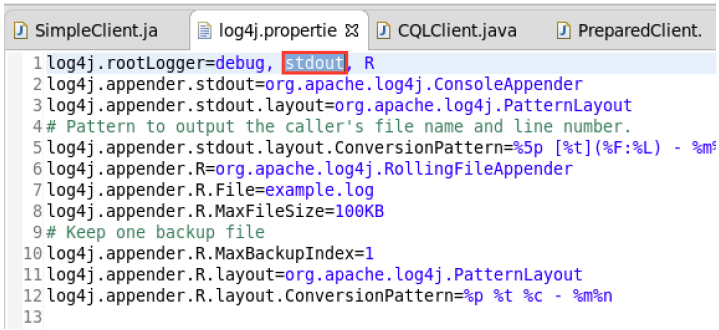
```
public void createSchema() {
    session.execute("CREATE KEYSPACE newkeyspace WITH replication " +
            "= {'class':'SimpleStrategy', 'replication_factor':1};");

    session.execute(
            "CREATE TABLE newkeyspace.person (" +
                    "id uuid PRIMARY KEY," +
                    "fname text," +
                    "lname text," +
                    "city text," +
                    "phone1 text," +
                    "phone2 text," +
                    "phone3 text" +
            ");");
}

public void loadData() {
    session.execute(
            "INSERT INTO newkeyspace.person (id, fname, lname, city, phone1,
phone2, phone3) " +
                    "VALUES (" +
                    "de305d54-75b4-431b-adb2-eb6b9e546001," +
                    "'Nick'," +
                    "'Smith'," +
                    "'Boston'," +
                    "'(781)397-7171'," +
                    "'(781)393-5303'," +
                    "'(781)286-8925'" +
            ");");
    session.execute(
            "INSERT INTO newkeyspace.person (id, fname, lname, city, phone1,
phone2, phone3) " +
                    "VALUES (" +
                    "de305d54-75b4-431b-adb2-eb6b9e546002," +
                    "'Joe'," +
```

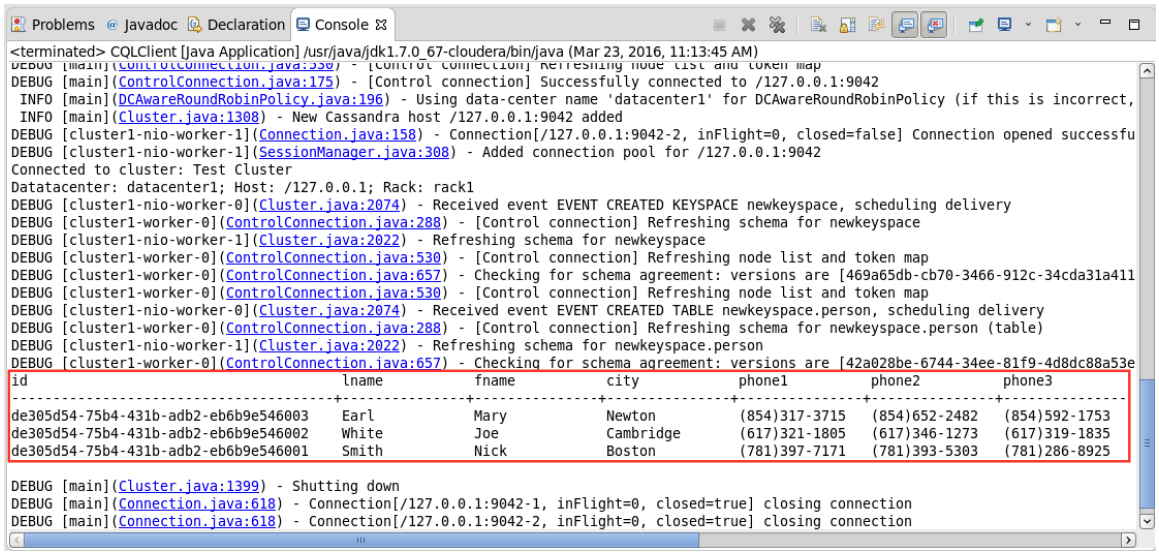
```
                    "'White'," +
                    "'Cambridge'," +
                    "'(617)321-1805'," +
                    "'(617)346-1273'," +
                    "'(617)319-1835'" +
            ");");
    session.execute(
            "INSERT INTO newkeyspace.person (id, fname, lname, city, phone1,
phone2, phone3) " +
                    "VALUES (" +
                    "de305d54-75b4-431b-adb2-eb6b9e546003," +
                    "'Mary'," +
                    "'Earl'," +
                    "'Newton'," +
                    "'(854)317-3715'," +
                    "'(854)652-2482'," +
                    "'(854)592-1753'" +
            ");");
}

public void querySchema(){
    ResultSet results = session.execute("SELECT * FROM newkeyspace.person");
    System.out.println(String.format("%-33s\t%-10s\t%-10s\t%-10s\t%-15s\t%-
15s\t%-15s\n%s",
            "id", "lname", "fname", "city", "phone1", "phone2", "phone3",
            "-------------------------------------+---------------+----------
-----+---------------+---------------+---------------+---------------"));
    for (Row row : results) {
        System.out.println(String.format("%-5s\t%-10s\t%-10s\t%-10s\t%-15s\t%-
15s\t%-15s", row.getUUID("id").toString(),
                row.getString("lname"), row.getString("fname"),
row.getString("city"),
                row.getString("phone1"), row.getString("phone2"),
row.getString("phone3")));
    }
    System.out.println();
}
```

3. Run the Java application. We can see that we can successfully create the keyspace, table and populate data. We can also select the whole content from the table.



Using the original **log4j.properties** file, we can see all the DEBUG messages being printed out to the console. We will suppress the output to console in step 4.





4. Modify “log4j.properties” and re-run the code.

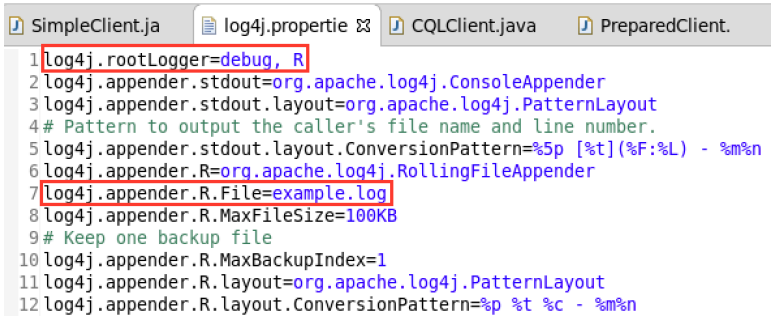
In our **log4j.properties**, the level of the root logger is defined as DEBUG, the DEBUG attached the appender named **stdout, R** to it. It then set the layout and pattern for the appender R.

We can add an Appender object to a Logger like this:

log4j.logger.[logger-name]=level, appender1,appender..n

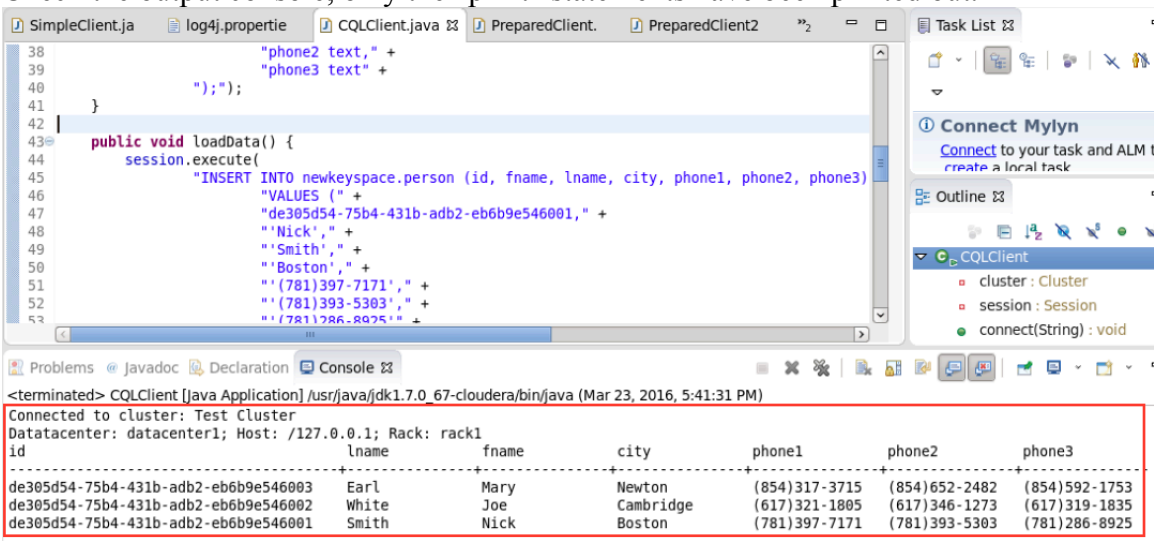
So here, the log will be appended to **stdout** and **R**. The appender **R** will writes to a file named **example.log**. If we want to stop DEBUG lines printing out to the console, we just need to delete **stdout** in **log4j.rootLogger**.

In log4j, a log request of level **p** in a logger with level **q** is enabled if **p >= q**. It assumes that levels are ordered. For the standard levels, we have **ALL < DEBUG < INFO < WARN < ERROR < FATAL < OFF**. So when we set the debug level to **“DEBUG”**, all the levels after that will be logged. We can also control the debug level by changing first parameter **level** in **log4j.rootLogger**.



Check **example.log**, all the debug messages have been logged here.

Check the output console, only the “print” statements have been printed out.



Problem 4) Placing hard-coded values inside your CQL (SQL) statements, as we did in the previous problem, is considered a bad programming practice. For all kind of reasons, including application security, code reuse and application performance, you want to be able to write generic CQL (SQL) statements which have placeholders for values and then assign concrete values at the moment when you want to perform database operations. In the class `CQLClient` we executed such hard coded (CQL) SQL statements using method `execute()` on the `Session` object. A better way is to create objects of `PreparedStatement` type. Those objects will contain CQL statements and bind values (place-holders). Prepared statements will only need to be parsed once by Cassandra cluster. We will bind values to the variables and execute the bound statements when we want to read or write data from or to Cassandra’s tables.

In your project, create a new class called `PerparedClient` by copying the content of `CQLClient`. Next, modify `loadData()` method . Add code to your client for:

- creating a prepared statement
- creating a bound statement from the prepared statement and binding values to its variables
- executing the bound statement to insert data

Add code to prepare an INSERT statement. You get a prepared statement by calling the prepare method on your session.

```
PreparedStatement statement = getSession().prepare(
      "INSERT INTO mykeyspace.songs " +
      "(id, title, album, artist) " +
      "VALUES (?, ?, ?, ?);");
```

Add code to bind values to the prepared statement's variables and then execute the statement. You create a bound statement by calling its constructor and passing in the prepared statement. Use the bind method to bind values and execute the bound statement on your session.

```
BoundStatement boundStatement = new BoundStatement(statement);
getSession().execute(boundStatement.bind(
      UUID.fromString("756716f7-2e54-4715-9f00-91dcbea6cf50"),
      "La Petite Tonkinoise'",
      "Bye Bye Blackbird'",
      "Joséphine Baker" ) );
```

Note that you cannot pass in string representations of UUIDs or sets as you did in the previous `loadData()` method.
Add code to create a new bound statement for inserting data into the `simplex.playlists` table.

```
statement = getSession().prepare(
      "INSERT INTO simplex.playlists " +
      "(id, song_id, title, album, artist) " +
      "VALUES (?, ?, ?, ?, ?);");
boundStatement = new BoundStatement(statement);
getSession().execute(boundStatement.bind(
      UUID.fromString("2cc9ccb7-6221-4ccb-8387-f22b6a1b354d"),
      UUID.fromString("756716f7-2e54-4715-9f00-91dcbea6cf50"),
      "La Petite Tonkinoise",
      "Bye Bye Blackbird",
      "Joséphine Baker") );
```

Review the `main()` method of your class.

```
public static void main(String[] args) {
   PreparedClient client = new PreparedClient();
   client.connect("127.0.0.1");
```

```
client.createSchema();
client.loadData();
client.querySchema();
client.close();
```

Of course, in the above, replace the `keyspace name, table names` and `column names` with names you used in your version of `CQLClient` class. Before running this new class go to the `cqlsh` prompt and drop your existing tables and the existing keyspaces if they overlap with ones in this problem. Otherwise, you might get an error telling you that a `keyspace (tables)` already exist.
Submit the working code and all console outputs.

Solution

1. Load data in three steps. First, create a prepared statement. Replace the real values with question mark. The same prepared statement should be prepared only once and will be parsed only once. Second, create a bound statement by calling the constructor and pass in the prepared statement. Third, bind real values to the bound statement and execute the code.

Since "session" is a private class object, we will create a method to get the instance.

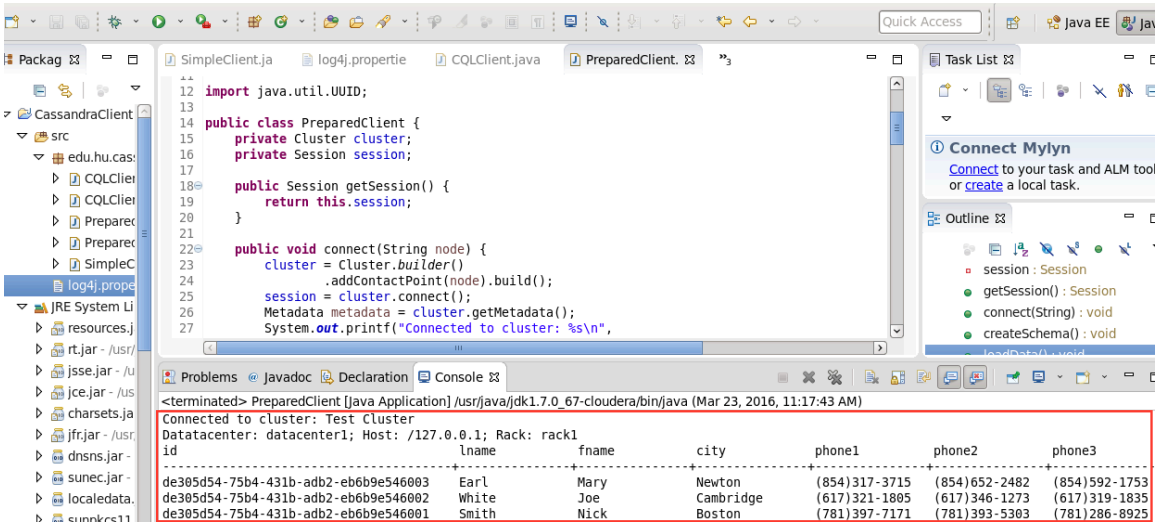
```
public Session getSession() {
    return this.session;
}

public void loadData() {
    PreparedStatement statement = getSession().prepare(
        "INSERT INTO newkeyspace.person " +
        "(id, fname, lname, city, phone1, phone2, phone3) " +
        "VALUES (?, ?, ?, ?, ?, ?);");

    BoundStatement boundStatement = new BoundStatement(statement);

    getSession().execute(boundStatement.bind(
        UUID.fromString("de305d54-75b4-431b-adb2-eb6b9e546001"),
        "Nick",
        "Smith",
        "Boston",
        "(781)397-7171",
        "(781)393-5303",
        "(781)286-8925") );

    getSession().execute(boundStatement.bind(
        UUID.fromString("de305d54-75b4-431b-adb2-eb6b9e546002"),
        "Joe",
        "White",
        "Cambridge",
        "(617)321-1805",
        "(617)346-1273",
```

```
        "(617)319-1835") );

    getSession().execute(boundStatement.bind(
        UUID.fromString("de305d54-75b4-431b-adb2-eb6b9e546003"),
        "Mary",
        "Earl",
        "Newton",
        "(854)317-3715",
        "(854)652-2482",
        "(854)592-1753") );
}
```

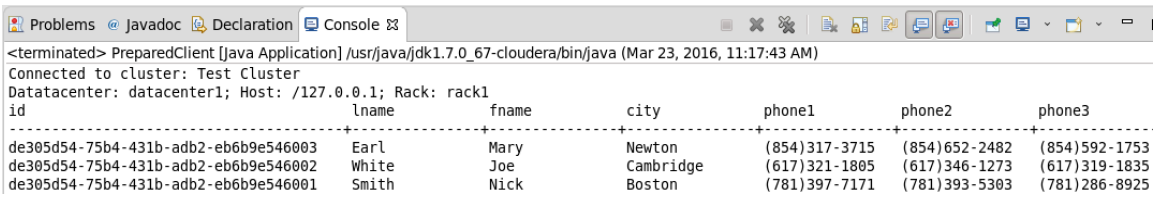
2. Since we use the same keyspace as before. Drop the existing keyspace first before executing the code. Like Problem 3, we can set the debug message only being printed out to the log file in "log4j.properties".

```
cqlsh> DROP KEYSPACE newkeyspace;
```

```
log4j.rootLogger=debug, R
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
# Pattern to output the caller's file name and line number.
log4j.appender.stdout.layout.ConversionPattern=%5p [%t](%F:%L) - %m%n
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=example.log
log4j.appender.R.MaxFileSize=100KB
# Keep one backup file
log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%p %t %c - %m%n
```



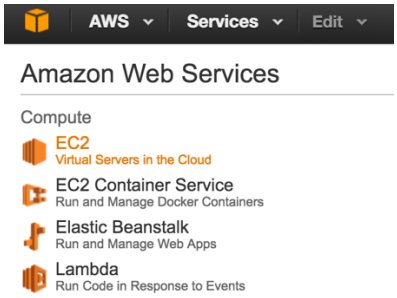
The results are correct!



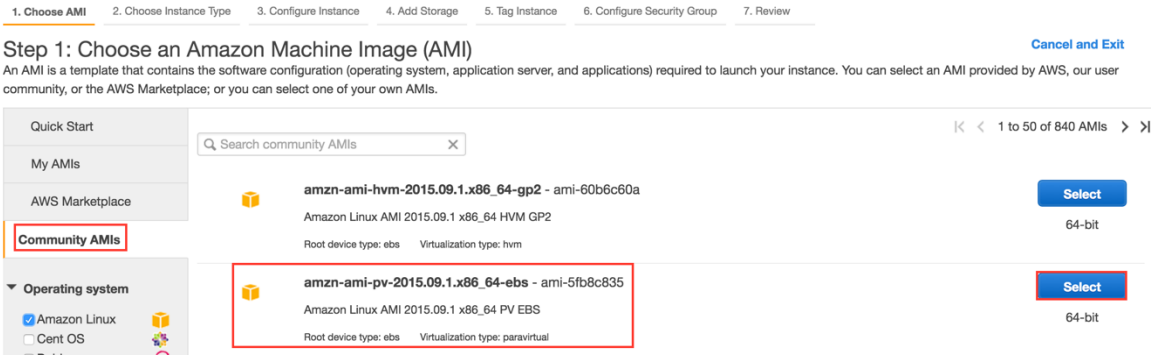
Problem 5) Instantiate a micro Amazon Linux instance in AWS Cloud. Download a Tomcat 8 distribution to your local machine and then transfer the file to the AWS instance using an `scp` command. Install Tomcat on your remote instance. Verify that port 8080 is set properly in Tomcat's `server.xml` configuration file. Start Tomcat on remote machine. Demonstrate that you can use your browser to open the Welcome page of the remote Tomcat.

Solution

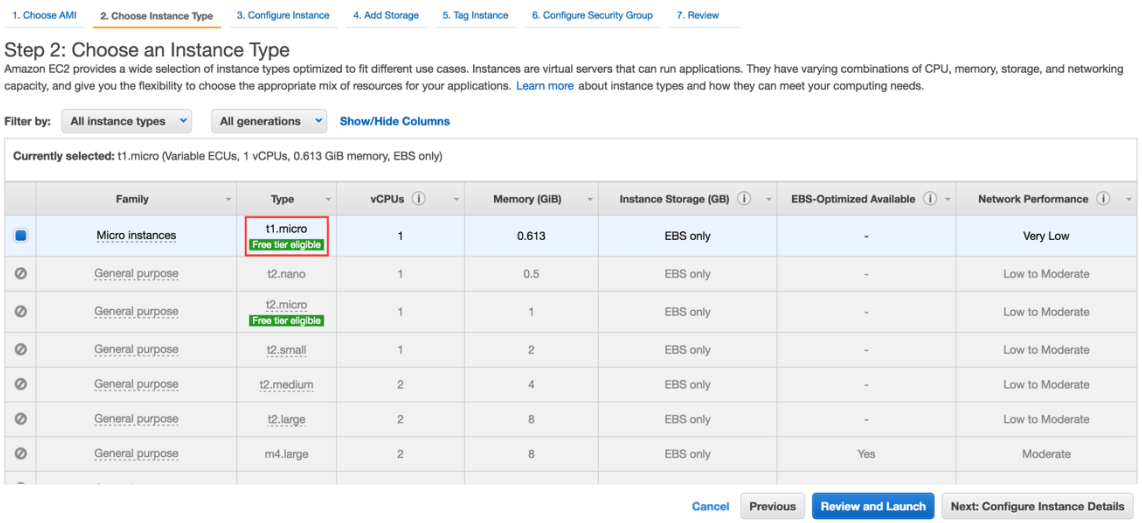
1. Create an Amazon AWS account. Download the credential information. Select EC2 service and launch an instance through it. Select the right region. Mine is "N.Virginia".



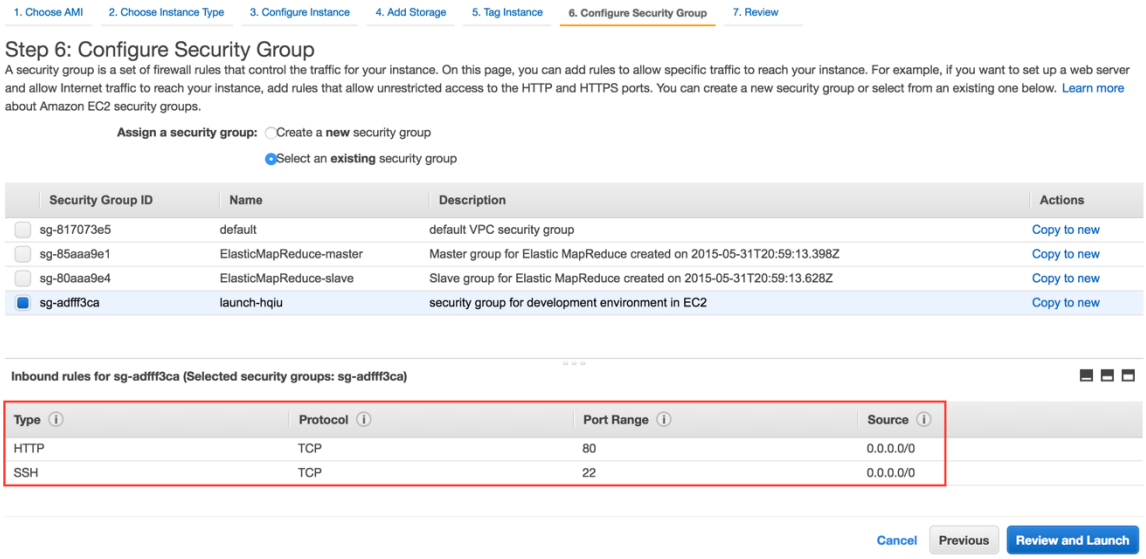
Choose any AMI from "Community AMIs". Here I choose ami-5fb8c835, which Tomcat is not available as default. We will install Tomcat by ourselves.



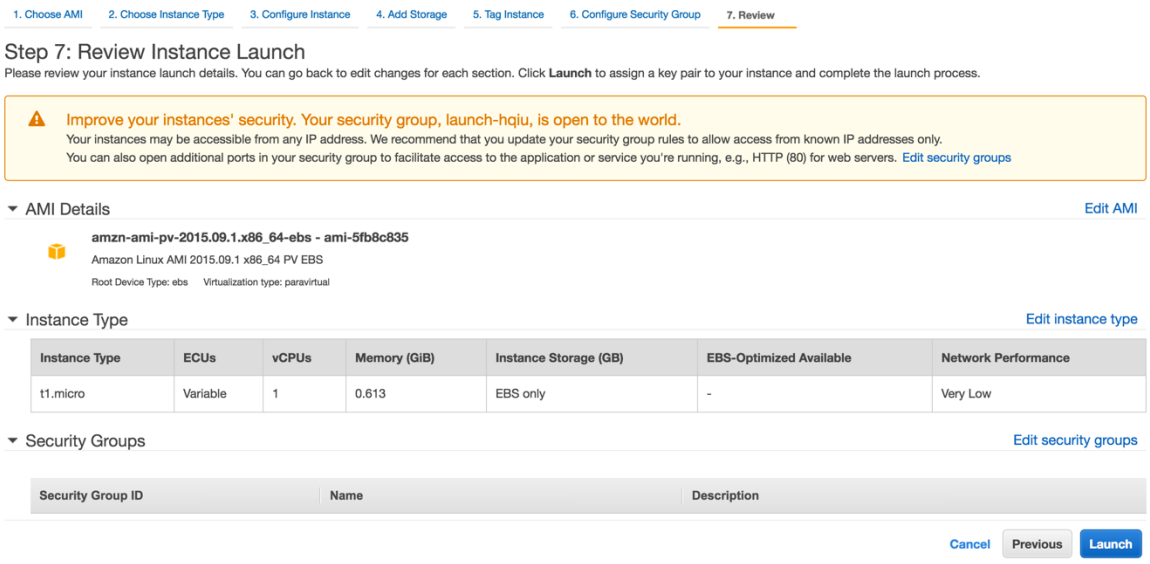
Choose “t1.micro” as the instance type since it’s free.



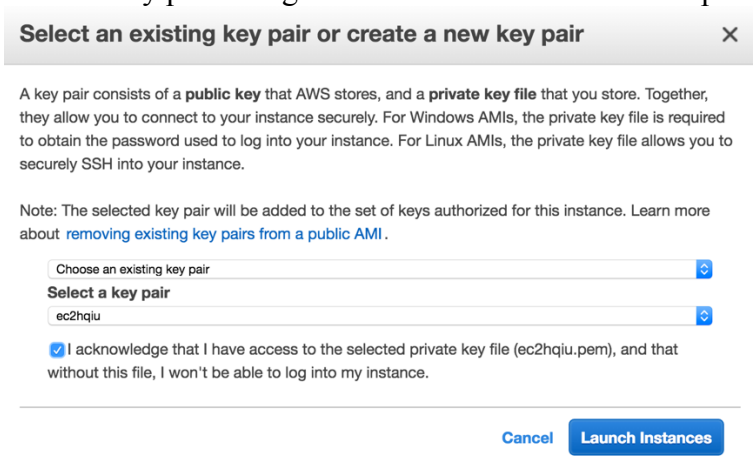
Select the “Security Group”. We can either select the existing security group or create a new one. Click the specific security group, we can see the containing rules. Here I open port 22 and port 80. New rules can be added after the EC2 instance has been launched.



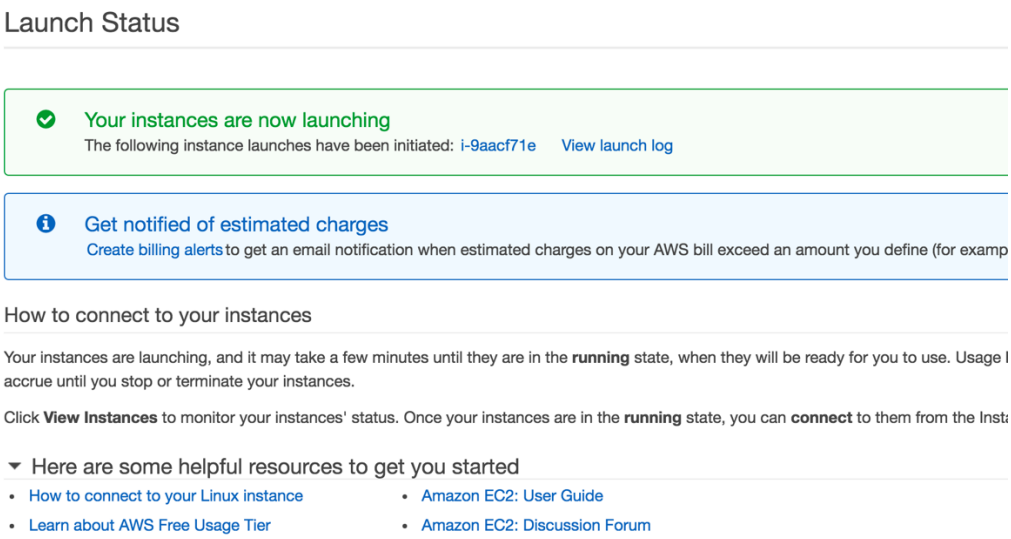
Review and launch the instance.



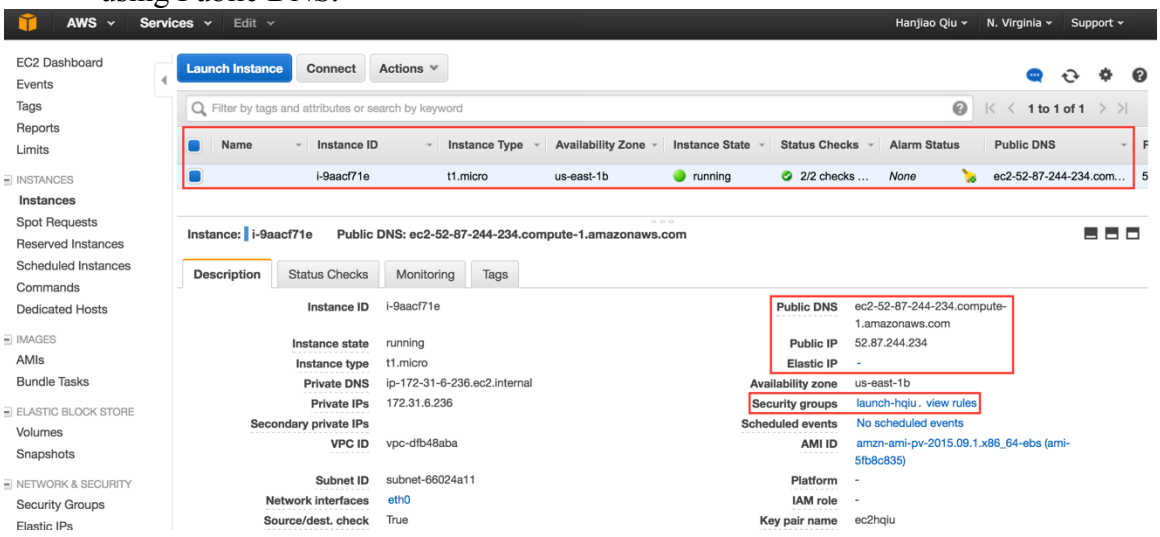
Select a key pair to log in to the instance as the last step.



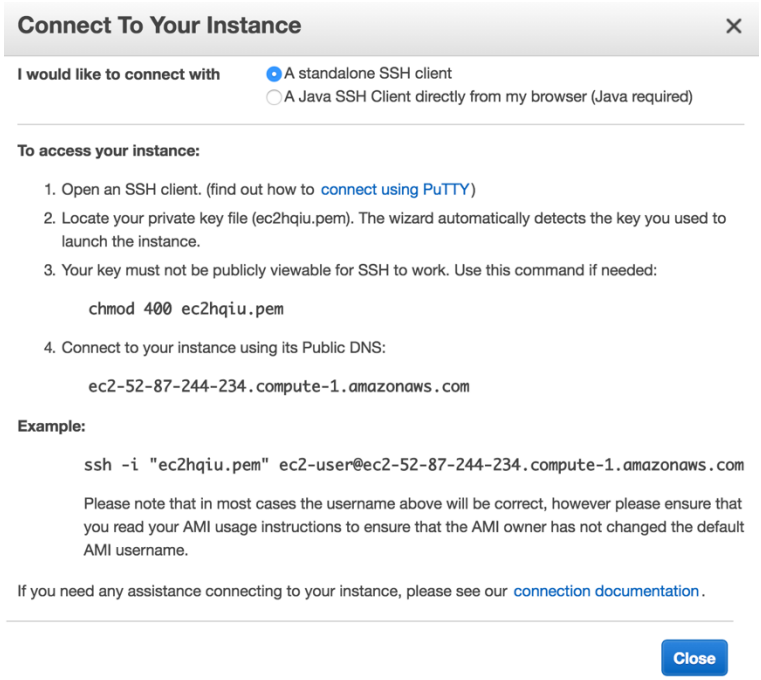
The EC2 instance has been successfully launched!



2. Check the running status of the instance. The public DNS as well as other information is listed in the description. The Public DNS of our instance is "ec2-52-87-244-234.compute-1.amazonaws.com". We can connect to our instance using Public DNS.



Click on the "Connect" button next to "Launch Instance". It shows how to connect to the instance.



Follow the instructions and connect to the instance.

```
hqiu@bos-mp9cx>> sudo chmod 400 ec2hqiu.pem

hqiu@bos-mp9cx>> ssh -i "ec2hqiu.pem" ec2-user@ec2-52-87-244-234.compute-
1.amazonaws.com
```

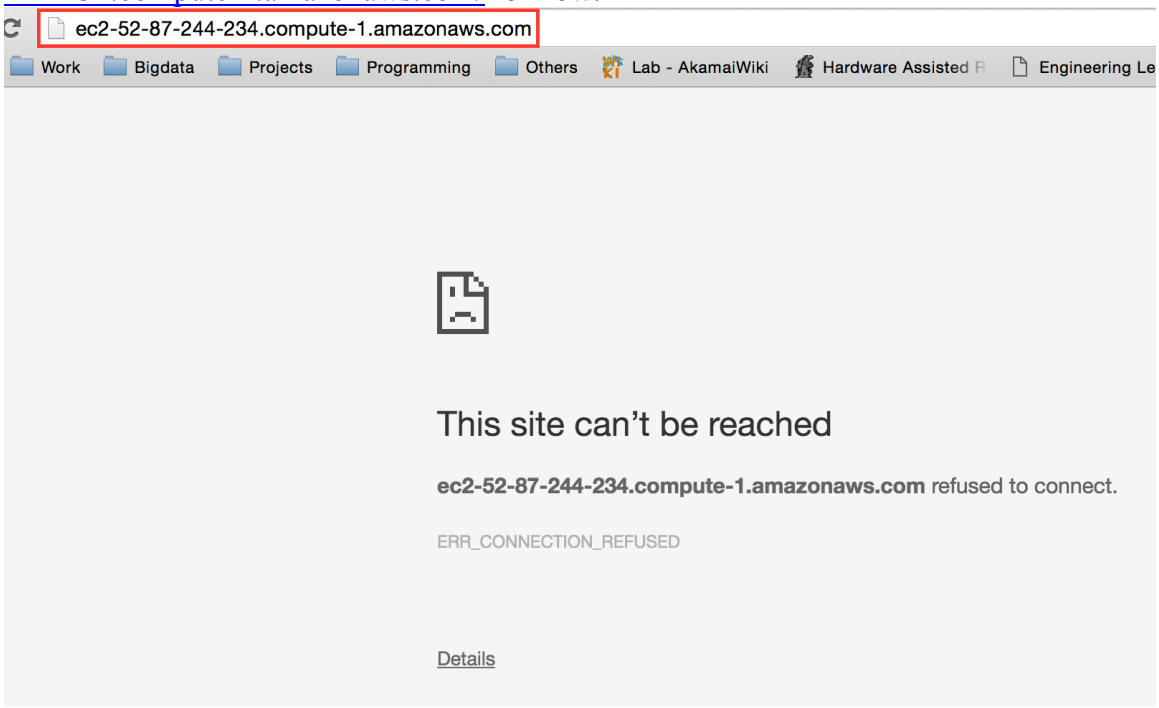
```
AWS — ec2-user@ip-172-31-6-236:~ — ssh — 95×27
ec2-user@ip-172-31-6-236:~
Last login: Tue Mar 22 19:34:34 on ttys014
hqiu@bos-mp9cx>> cd ~/Documents/Harvard_Extension/Big_Data_Analytics/AWS/
hqiu@bos-mp9cx>> sudo chmod 400 ec2hqiu.pem
Password:
hqiu@bos-mp9cx>> ssh -i "ec2hqiu.pem" ec2-user@ec2-52-87-244-234.compute-1.amazonaws.com
The authenticity of host 'ec2-52-87-244-234.compute-1.amazonaws.com (52.87.244.234)' can't be e
stablished.
RSA key fingerprint is 76:32:28:bd:c1:fa:ca:84:4b:51:49:63:8a:ce:db:a5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-52-87-244-234.compute-1.amazonaws.com,52.87.244.234' (RSA) to t
he list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2015.09-release-notes/
23 package(s) needed for security, out of 60 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2016.03 is available.
[ec2-user@ip-172-31-6-236 ~]$
```

```
[ec2-user@ip-172-31-6-236 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-6-236 ~]$ sudo yum update
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package aws-cli.noarch 0:1.9.1-1.29.amzn1 will be updated
---> Package aws-cli.noarch 0:1.10.8-1.37.amzn1 will be an update
--> Processing Dependency: /etc/mime.types for package: aws-cli-1.10.8-1.37.amzn1.noarch
---> Package bash.x86_64 0:4.2.46-12.34.amzn1 will be updated
---> Package bash.x86_64 0:4.2.46-19.35.amzn1 will be an update
---> Package binutils.x86_64 0:2.23.52.0.1-30.64.amzn1 will be updated
```

Check the ports listening on the Linux box for incoming calls. Since this box doesn't have "httpd" service running as default, we won't see it listening on port 80.

```
[ec2-user@ip-172-31-6-236 ~]$ netstat -an | grep LISTEN
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:52387               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
tcp        0      0 :::111                      :::*                        LISTEN
tcp        0      0 :::33938                    :::*                        LISTEN
unix  2      [ ACC ]     STREAM     LISTENING     10169  /var/run/rpcbind.sock
unix  2      [ ACC ]     STREAM     LISTENING     8750   /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     7451   @/com/ubuntu/upstart
unix  2      [ ACC ]     SEQPACKET  LISTENING     7659   @/org/kernel/udev/udevd
```

Since the Apache HTTP server is running, we couldn't get access to http://ec2-52-87-244-234.compute-1.amazonaws.com/ for now.



3. This is not required for running Tomcat. However, we will show how to get access to the web page through both port 80 and port 8080 here.

Install Apache HTTP service “httpd” and start the service.

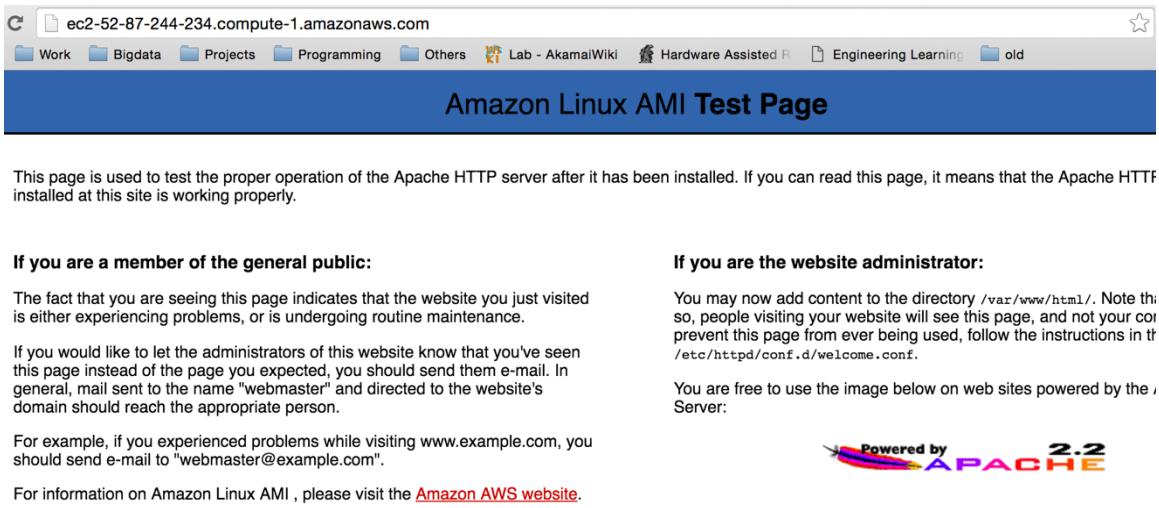
```
[ec2-user@ip-172-31-6-236 ~]$ sudo yum install httpd
```

```
[ec2-user@ip-172-31-6-236 ~]$ sudo yum install httpd
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.2.31-1.7.amzn1 will be installed
--> Processing Dependency: httpd-tools = 2.2.31-1.7.amzn1 for package: httpd-2.2.31-1.7.amzn1.x86_64
--> Processing Dependency: apr-util-ldap for package: httpd-2.2.31-1.7.amzn1.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.2.31-1.7.amzn1.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.2.31-1.7.amzn1.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.5.1-1.12.amzn1 will be installed
---> Package apr-util.x86_64 0:1.4.1-4.17.amzn1 will be installed
---> Package apr-util-ldap.x86_64 0:1.4.1-4.17.amzn1 will be installed
---> Package httpd-tools.x86_64 0:2.2.31-1.7.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

```
[ec2-user@ip-172-31-6-236 ~]$ sudo /etc/init.d/httpd start
```

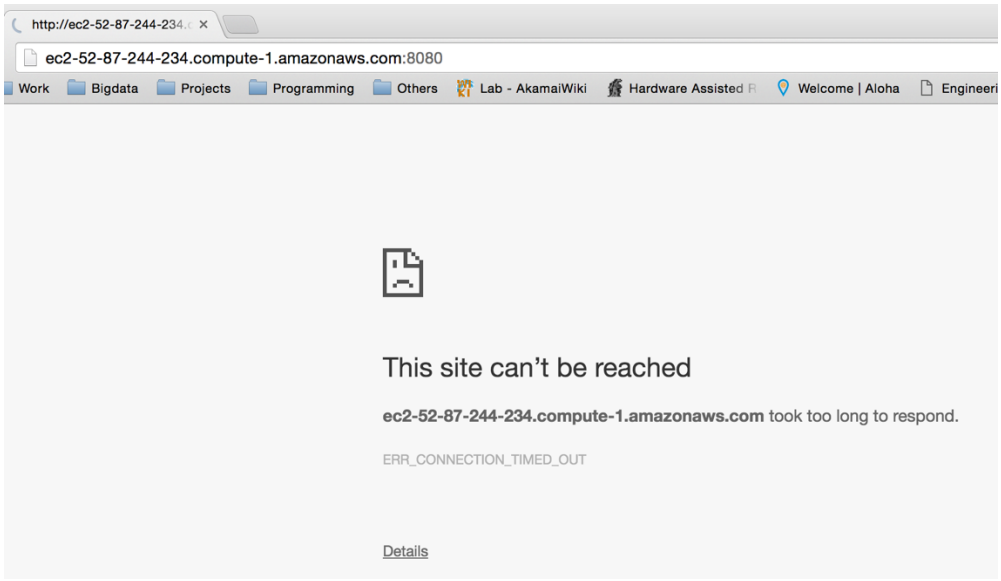
```
[ec2-user@ip-172-31-6-236 ~]$ ps -ef | grep httpd
```

```
[ec2-user@ip-172-31-6-236 ~]$ sudo /etc/init.d/httpd start
Starting httpd:                                            [  OK  ]
[ec2-user@ip-172-31-6-236 ~]$ ps -ef | grep httpd
root      6139     1  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6141  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6142  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6143  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6144  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6145  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6146  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6147  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
apache    6148  6139  0 01:28 ?        00:00:00 /usr/sbin/httpd
ec2-user  6150  1481  0 01:29 pts/0    00:00:00 grep --color=auto httpd
[ec2-user@ip-172-31-6-236 ~]$ netstat -an | grep LISTEN
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:52387               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
tcp        0      0 :::111                      :::*                        LISTEN
tcp        0      0 :::80                       :::*                        LISTEN
tcp        0      0 :::33938                    :::*                        LISTEN
unix  2      [ ACC ]     STREAM     LISTENING     10169  /var/run/rpcbind.sock
unix  2      [ ACC ]     STREAM     LISTENING     8750   /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     7451   @/com/ubuntu/upstart
unix  2      [ ACC ]     SEQPACKET  LISTENING     7659   @/org/kernel/udev/udevd
```



We can see that "httpd" daemon is running and listening on port 80. Refresh the page, we can see the Amazon Linux AMI Test Page now.

4. Tomcat is not visible as default. We will download Tomcat 8 package from https://tomcat.apache.org/download-80.cgi to our local machine and transfer it to the Linux box using "scp" command.




```
hqiu@bos-mp9cx>> scp -i "ec2hqiu.pem" ~/Downloads/apache-tomcat-8.0.32.tar.gz
ec2-user@ec2-52-87-244-234.compute-1.amazonaws.com:/home/ec2-user
```

```
hqiu@bos-mp9cx>> cd ~/Documents/Harvard_Extension/Big_Data_Analytics/AWS/
hqiu@bos-mp9cx>> scp -i "ec2hqiu.pem" ~/Downloads/apache-tomcat-8.0.32.tar.gz ec2-user@ec2-52-87-244
-234.compute-1.amazonaws.com:/home/ec2-user
apache-tomcat-8.0.32.tar.gz                                   100% 8954KB 895.4KB/s   00:10
```

```
[ec2-user@ip-172-31-6-236 ~]$ tar xzf apache-tomcat-8.0.32.tar.gz
[ec2-user@ip-172-31-6-236 ~]$ sudo vi apache-tomcat-8.0.32/conf/server.xml
```

```
[ec2-user@ip-172-31-6-236 ~]$ ls
apache-tomcat-8.0.32.tar.gz
[ec2-user@ip-172-31-6-236 ~]$ tar xzf apache-tomcat-8.0.32.tar.gz
[ec2-user@ip-172-31-6-236 ~]$ sudo vi apache-tomcat-8.0.32/conf/server.xml
```

Uncomment the part related to connector 8080 in “conf/server.xml” file and start Tomcat service.

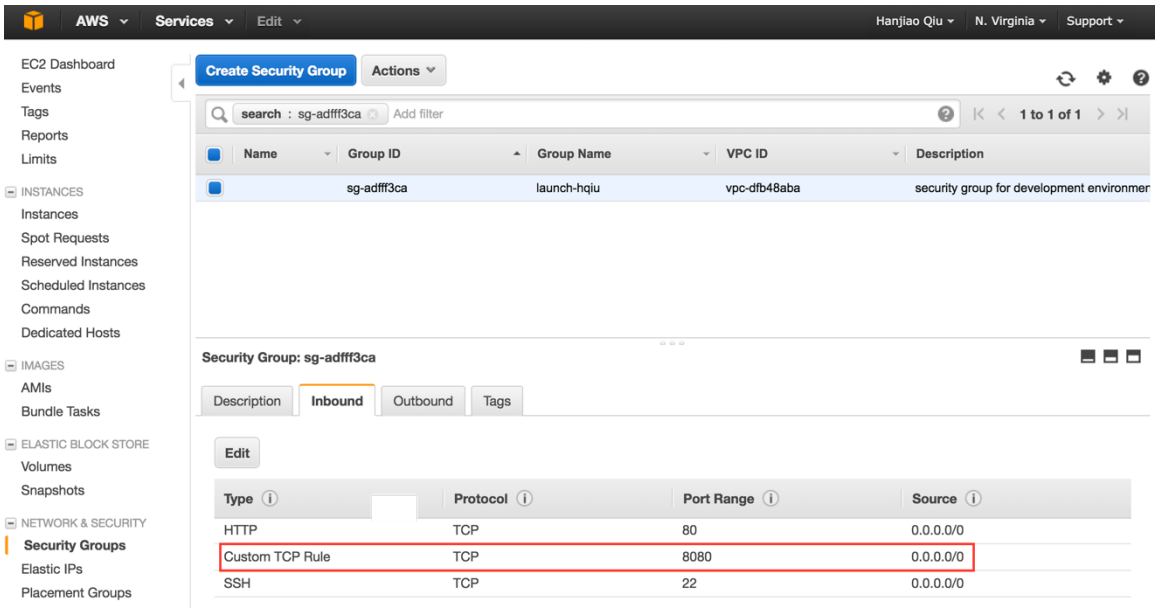
```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
-->
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->

<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

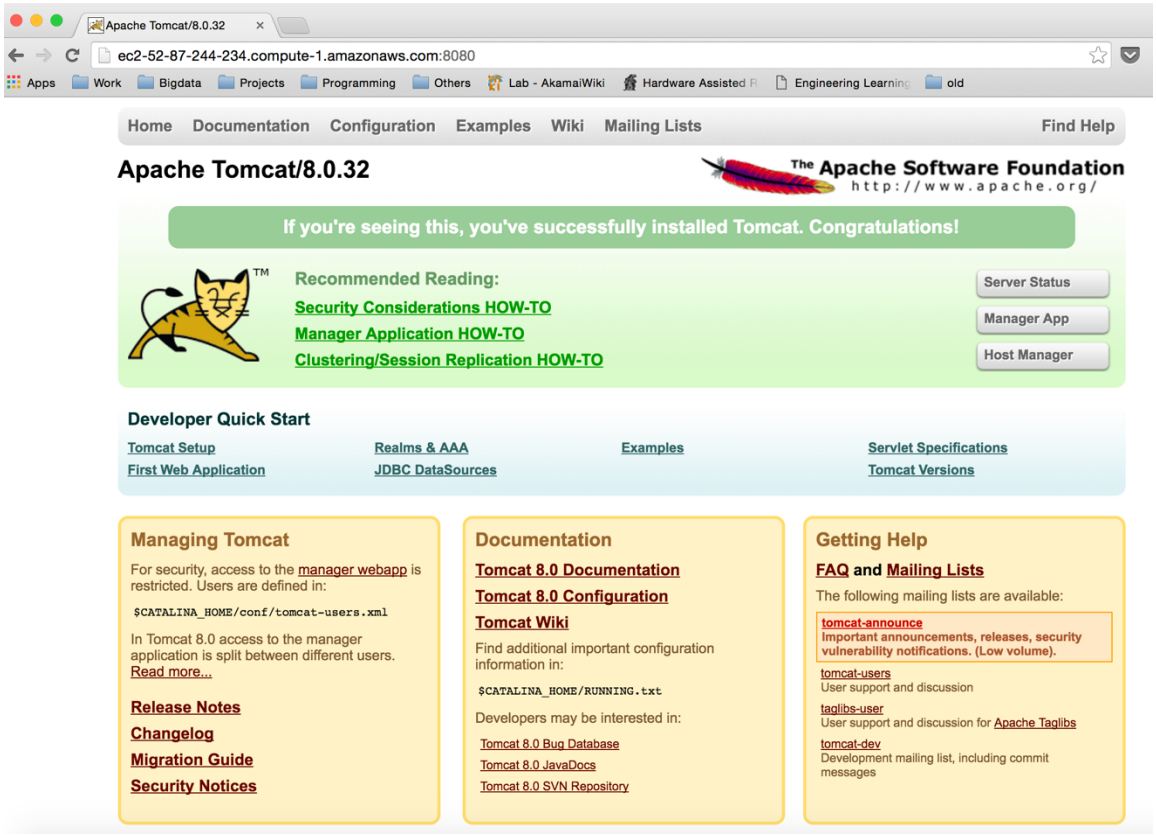
```
[ec2-user@ip-172-31-6-236 ~]$ sudo ./apache-tomcat-8.0.32/bin/startup.sh

[ec2-user@ip-172-31-6-236 ~]$ ps -ef | grep tomcat

[ec2-user@ip-172-31-6-236 ~]$ netstat -an | grep LISTEN
```

```
[ec2-user@ip-172-31-6-236 ~]$ netstat -an | grep LISTEN
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:52387               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
tcp        0      0 ::ffff:127.0.0.1:8005       :::*                        LISTEN
tcp        0      0 :::8009                     :::*                        LISTEN
tcp        0      0 :::111                      :::*                        LISTEN
tcp        0      0 :::8080                     :::*                        LISTEN
tcp        0      0 :::80                       :::*                        LISTEN
tcp        0      0 :::33938                    :::*                        LISTEN
unix  2      [ ACC ]     STREAM     LISTENING     10169  /var/run/rpcbind.sock
unix  2      [ ACC ]     STREAM     LISTENING     8750   /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     7451   @/com/ubuntu/upstart
unix  2      [ ACC ]     SEQPACKET  LISTENING     7659   @/org/kernel/udev/udevd
```

Port 8080 is active and Tomcat is listening on it now.

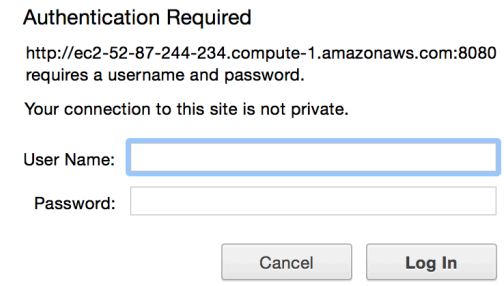
Besides that, we need to allow incoming traffic from port 8080 on our Linux box. We need to edit our security group and add inbound traffic from port 8080.

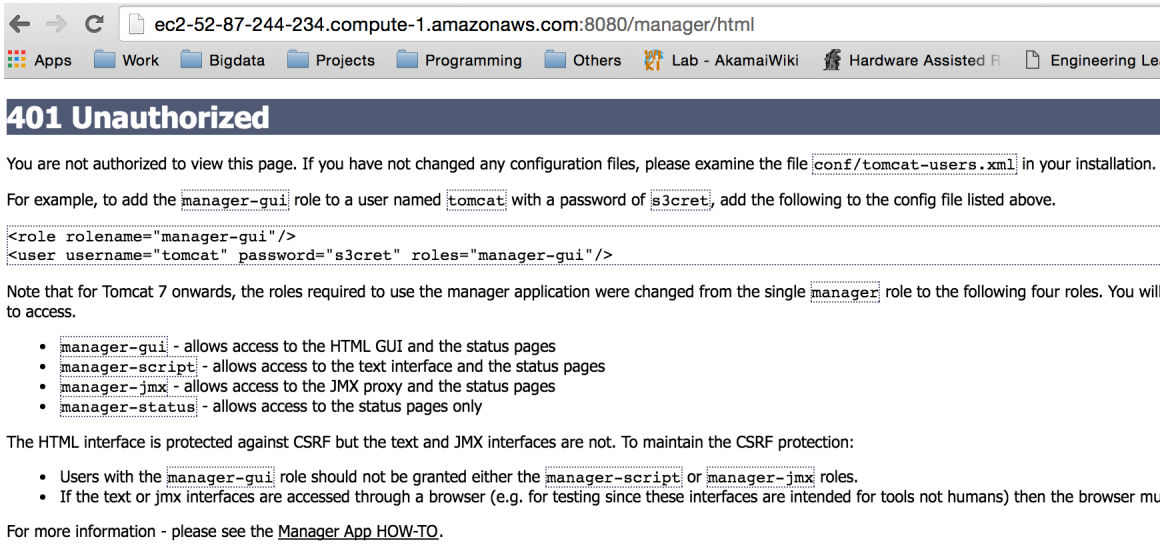




Now we can get access to http://ec2-52-87-244-234.compute-1.amazonaws.com:8080 now!

5. Try clicking on “Manager App”. We will see some authentication problem.





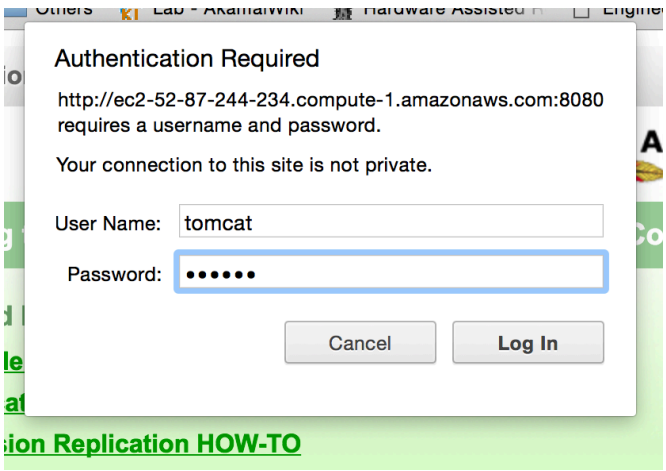
Customize the Tomcat installation to enable it. Add “manager-gui” role and user to “tomcat-users.xml” from my local apache-tomcat. Restart the Tomcat service to apply the change.

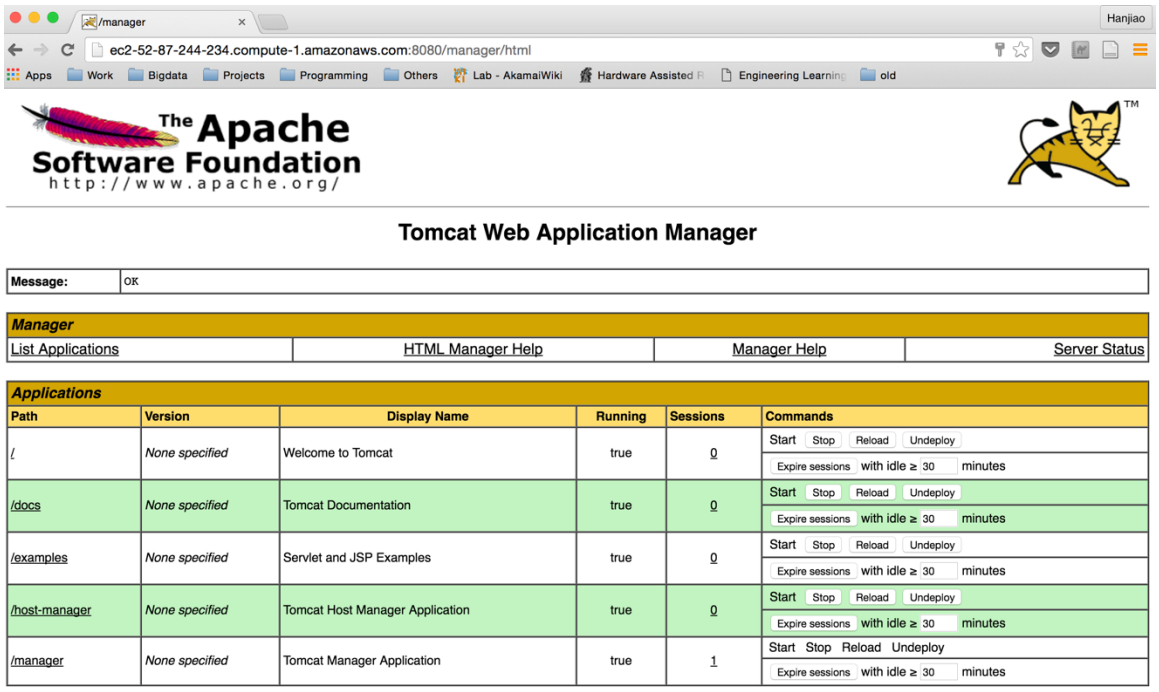
```
[ec2-user@ip-172-31-6-236 ~]$ sudo vi apache-tomcat-8.0.32/conf/tomcat-
users.xml

[ec2-user@ip-172-31-6-236 ~]$ sudo ./apache-tomcat-8.0.32/bin/shutdown.sh
[ec2-user@ip-172-31-6-236 ~]$ sudo ./apache-tomcat-8.0.32/bin/startup.sh
```

```
<!--
  NOTE:  The sample user and role entries below are wrapped in a comment
  and thus are ignored when reading this file. Do not forget to remove
  <!.. ..> that surrounds them.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
-->

<role rolename="manager-gui"/>
<user username="tomcat" password="tomcat" roles="manager-gui"/>

</tomcat-users>
-- INSERT --
```

```
[ec2-user@ip-172-31-6-236 ~]$ sudo ./apache-tomcat-8.0.32/bin/shutdown.sh
Using CATALINA_BASE:   /home/ec2-user/apache-tomcat-8.0.32
Using CATALINA_HOME:   /home/ec2-user/apache-tomcat-8.0.32
Using CATALINA_TMPDIR: /home/ec2-user/apache-tomcat-8.0.32/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/ec2-user/apache-tomcat-8.0.32/bin/bootstrap.jar:/home/ec2-user/apache-t
omcat-8.0.32/bin/tomcat-juli.jar
[ec2-user@ip-172-31-6-236 ~]$ sudo ./apache-tomcat-8.0.32/bin/startup.sh
Using CATALINA_BASE:   /home/ec2-user/apache-tomcat-8.0.32
Using CATALINA_HOME:   /home/ec2-user/apache-tomcat-8.0.32
Using CATALINA_TMPDIR: /home/ec2-user/apache-tomcat-8.0.32/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/ec2-user/apache-tomcat-8.0.32/bin/bootstrap.jar:/home/ec2-user/apache-t
omcat-8.0.32/bin/tomcat-juli.jar
Tomcat started.
```

Refresh the page and login with username “tomcat” and password “tomcat”. Now we can get into the Manager App!





6. Don't forget to terminate the instance after all work done.