

## 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` be 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/cassandra/db/composites/Composite;)I
CompilerOracle: inline org/apache/cassandra/db/composites/AbstractSimpleCellNameType.compareUnsigned (Lorg/apache/cassandra/db/composites/Composite;Lorg/apache/cassandra/db/composites/Composite;)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/ByteBuffer;I)

em/local-7ad54392bcd35a684174e047860b377/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 were {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...
```

- 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>
```

- 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', '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;



| <b>id</b>                            | <b>city</b> | <b>fname</b> | <b>lname</b> | <b>phone1</b> | <b>phone2</b> | <b>phone3</b> |
|--------------------------------------|-------------|--------------|--------------|---------------|---------------|---------------|
| 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';



| <b>id</b>                            | <b>city</b> | <b>fname</b> | <b>lname</b> | <b>phone1</b> | <b>phone2</b> | <b>phone3</b> |
|--------------------------------------|-------------|--------------|--------------|---------------|---------------|---------------|
| 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;
```

<b>id</b>	<b>city</b>	<b>fname</b>	<b>lname</b>	<b>phone1</b>	<b>phone2</b>	<b>phone3</b>
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';



| <b>id</b>                            | <b>city</b> | <b>fname</b> | <b>lname</b> | <b>phone1</b> | <b>phone2</b> | <b>phone3</b> |
|--------------------------------------|-------------|--------------|--------------|---------------|---------------|---------------|
| 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)

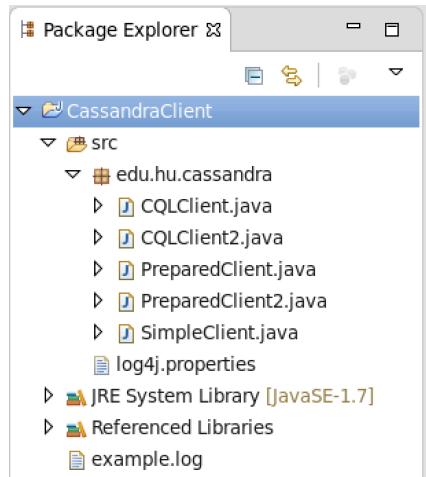

```

**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

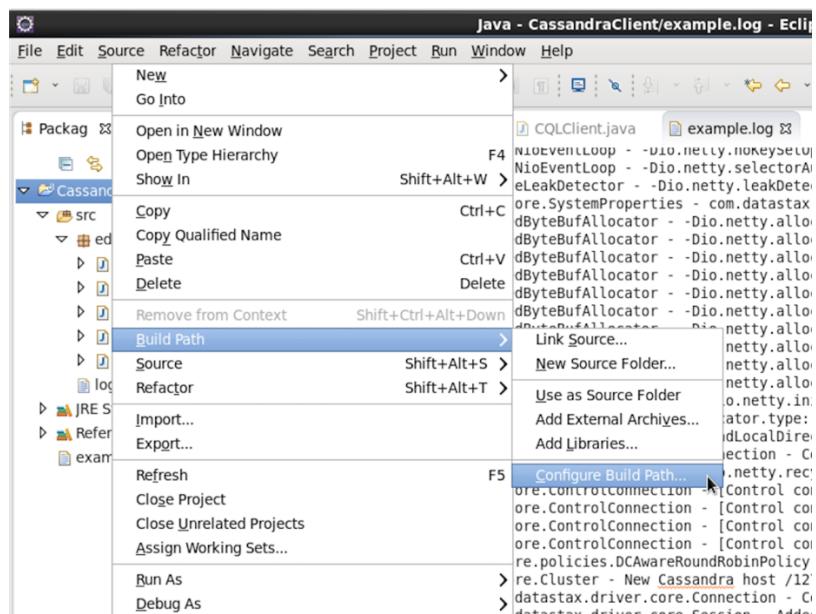
1. Create a Java project. The file structure is like the following figure. Put the source file “SimpleClient.java” into package “edu.hu.cassandra” and “log4j.properties” into “src” directory.



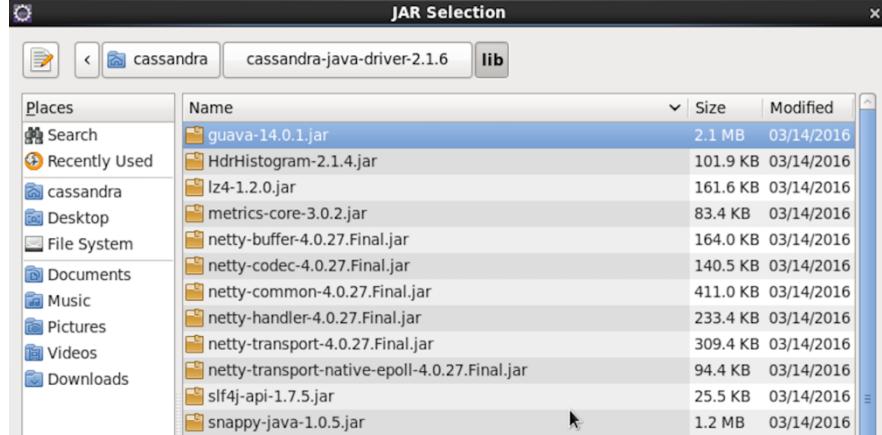
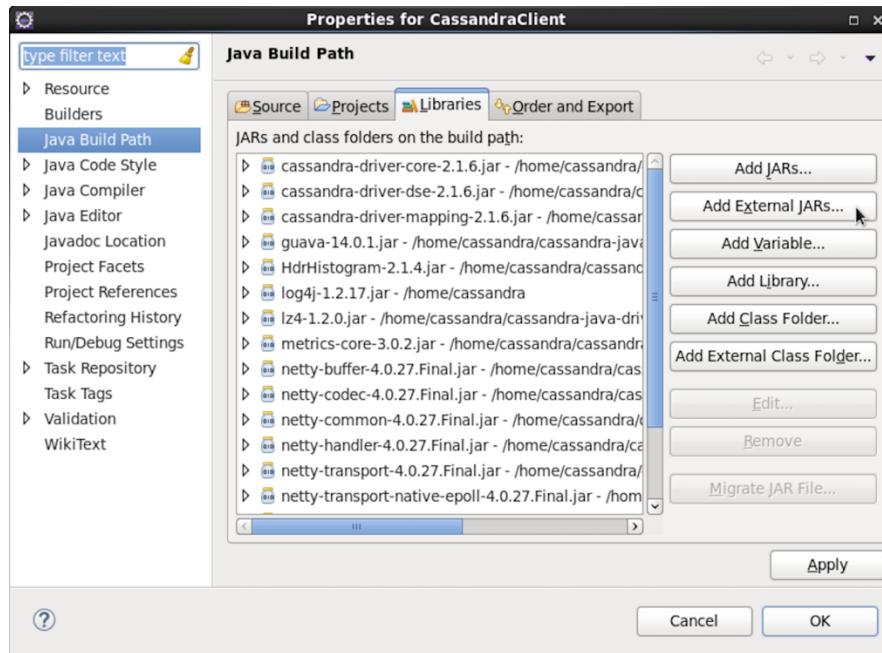
2. Download the binary drive and import the jar files.

Download **Cassandra java driver** from:

<http://downloads.datastax.com/java-driver/cassandra-java-driver-2.1.6.tar.gz>



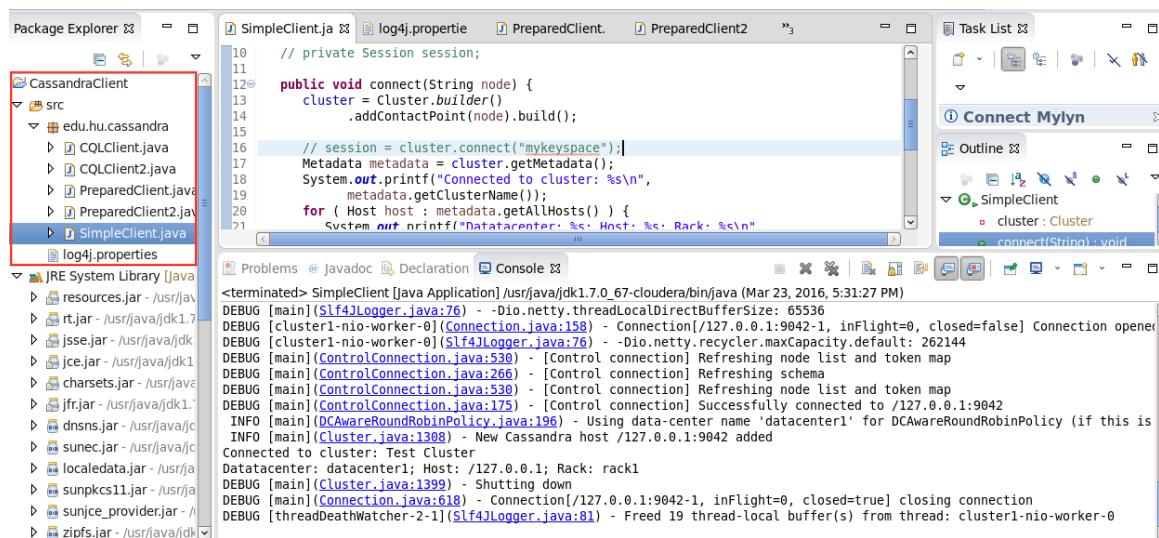
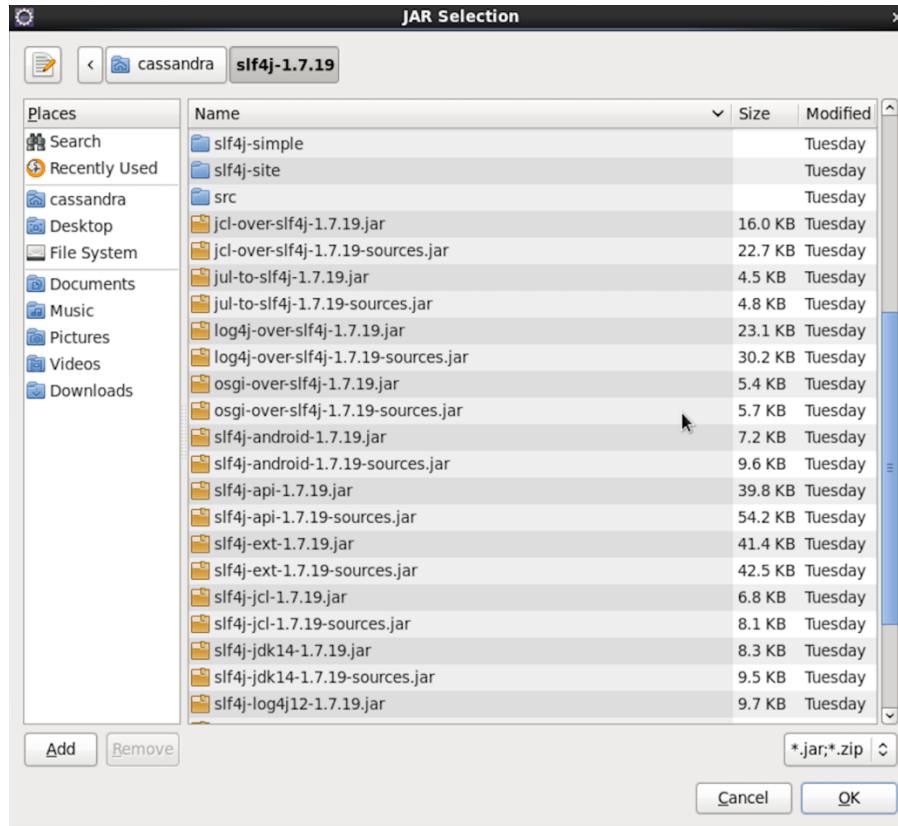
Import the jar files from the folder “cassandra-java-driver”. Right click the project name and choose “Build Path -> Configure Build Path”. Then in the pop-up window, choose “Libraries -> Add External JARs”. Choose “**cassandra-driver-core-2.1.6.jar**” and “**cassandra-driver-dse-2.1.6.jar**”. Also,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

```
<terminated> SimpleClient [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Mar 23, 2016, 5:31:27 PM)
DEBUG [main] (SystemProperties.java:32) - com.datastax.driver.NEW_NODE_DELAY_SECONDS is undefined, using default value 1
DEBUG [main] (SystemProperties.java:32) - com.datastax.driver.NON_BLOCKING_EXECUTOR_SIZE is undefined, using default value 1
DEBUG [main] (SystemProperties.java:32) - com.datastax.driver.NOTIFY_LOCK_TIMEOUT_SECONDS is undefined, using default value 60
DEBUG [main] (Cluster.java:1236) - Starting new cluster with contact points [/127.0.0.1:9042]
DEBUG [main] (Slf4JLogger.java:71) - Using SLF4J as the default logging framework
DEBUG [main] (Slf4JLogger.java:76) - java.nio.Buffer.address: available
DEBUG [main] (Slf4JLogger.java:76) - sun.misc.Unsafe.theUnsafe: available
DEBUG [main] (Slf4JLogger.java:71) - sun.misc.Unsafe.copyMemory: available
DEBUG [main] (Slf4JLogger.java:76) - java.nio.Bits.unaligned: true
DEBUG [main] (Slf4JLogger.java:76) - Java version: 7
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.noUnsafe: false
DEBUG [main] (Slf4JLogger.java:76) - sun.misc.Unsafe: available
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.noJavassist: false
DEBUG [main] (Slf4JLogger.java:71) - Javassist: unavailable
DEBUG [main] (Slf4JLogger.java:71) - You don't have Javassist in your class path or you don't have enough permission to load d
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.tmpdir: /tmp (java.io.tmpdir)
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.bitMode: 64 (sun.arch.data.model)

<terminated> SimpleClient [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Mar 23, 2016, 5:31:27 PM)
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.allocator.maxCachedBufferCapacity: 32768
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.allocator.cacheTrimInterval: 8192
DEBUG [main] (Slf4JLogger.java:71) - -Dio.netty.initialSeedUniquifier: 0xa9728d36325ce81c (took 26 ms)
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.allocator.type: unpooled
DEBUG [main] (Slf4JLogger.java:76) - -Dio.netty.threadLocalDirectBufferSize: 65536
DEBUG [cluster1-nio-worker-0] (Connection.java:158) - Connection[/127.0.0.1:9042-1, inFlight=0, closed=false] Connection opened
DEBUG [cluster1-nio-worker-0] (Slf4JLogger.java:76) - -Dio.netty.recycler.maxCapacity.default: 262144
DEBUG [main] (ControlConnection.java:530) - [Control connection] Refreshing node list and token map
DEBUG [main] (ControlConnection.java:266) - [Control connection] Refreshing schema
DEBUG [main] (ControlConnection.java:530) - [Control connection] Refreshing node list and token map
DEBUG [main] (ControlConnection.java:175) - [Control connection] Successfully connected to /127.0.0.1:9042
INFO [main] (DCAwareRoundRobinPolicy.java:196) - Using data-center name 'datacenter1' for DCAwareRoundRobinPolicy (if this is
INFO [main] (Cluster.java:1398) - New Cassandra host /127.0.0.1:9042 added
Connected to cluster: Test Cluster
Datacenter: datacenter1; Host: /127.0.0.1; Rack: rack1
DEBUG [main] (Cluster.java:1399) - Shutting down
DEBUG [main] (Connection.java:618) - Connection[/127.0.0.1:9042-1, inFlight=0, closed=true] closing connection
DEBUG [threadDeathWatcher-2-1] (Slf4JLogger.java:81) - Freed 19 thread-local buffer(s) from thread: cluster1-nio-worker-0
```

**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 Cassandra 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("Datacenter: %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.

The screenshot shows the Eclipse IDE interface with several open tabs and windows. The left sidebar displays the project structure for 'CassandraClient' with files like 'log4j.properties', 'CQLClient2.java', and 'PreparedClient.java'. The main editor window contains the 'CQLClient.java' code, which includes a 'querySchema()' method. The 'Console' tab in the bottom right shows the application's output. It includes DEBUG logs from the Cassandra driver and the application itself, such as schema creation and connection details. A red box highlights a portion of the console output where the application prints a table of person data.

```

78     public void querySchema(){
79         ResultSet results = session.execute("SELECT * FROM newkeyspace.person");
80         System.out.println(String.format("%-35t%-10s\\t%-10s\\t%-10s\\t%-15s\\t%-15s\\n%s",
81             "id", "lname", "fname", "city", "phone1", "phone2", "phone3",
82             "..."));
83         for (Row row : results) {
84             System.out.println(String.format("%-5s\\t%-10s\\t%-10s\\t%-10s\\t%-15s\\t%-15s\\n%s",
85                 row.getString("lname"), row.getString("fname"), row.getString("city"),
86                 row.getString("phone1"), row.getString("phone2"), row.getString("phone3")));
87         }
88     }

```

```

<terminated> CQLClient [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Mar 23, 2016, 11:13:45 AM)
DEBUG [cluster1-worker-0]([ControlConnection.java:530] - [Control connection] Refreshing node list and token map
DEBUG [cluster1-worker-0]([ControlConnection.java:657] - Checking for schema agreement: versions are [469a65db-cb70-3466-912c-34cd31a411
DEBUG [cluster1-worker-0]([ControlConnection.java:530] - [Control connection] Refreshing node list and token map
DEBUG [cluster1-nio-worker-0]([Cluster.java:2074] - Received event EVENT CREATED TABLE newkeyspace.person, scheduling delivery
DEBUG [cluster1-worker-0]([ControlConnection.java:288] - [Control connection] Refreshing schema for newkeyspace.person
DEBUG [cluster1-nio-worker-0]([Cluster.java:2022] - Refreshing schema for newkeyspace.person
DEBUG [cluster1-worker-0]([ControlConnection.java:657] - Checking for schema agreement: versions are [42a028be-6744-34ee-81f9-4d8dc88a53e
id          lname        fname       city      phone1    phone2    phone3
de305d54-75b4-431b-adb2-eb6b9e546003 Earl        Mary        Newton    (854)317-3715 (854)652-2482 (854)592-1753
de305d54-75b4-431b-adb2-eb6b9e546002 White       Joe        Cambridge (617)321-1805 (617)346-1273 (617)319-1835
de305d54-75b4-431b-adb2-eb6b9e546001 Smith       Nick       Boston    (781)397-7171 (781)393-5303 (781)286-8925

```

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.

The screenshot shows the Eclipse IDE interface with the 'SimpleClient.java' file selected. The code is identical to the one in the previous screenshot, but the 'log4j.properties' file has been modified. Line 1 now contains 'log4j.rootLogger=debug, stdout, R'. The 'Console' tab shows the application's output, which is significantly cleaner as it only contains INFO and higher-level logs, indicating that DEBUG output is no longer being printed.

```

1 log4j.rootLogger=debug, stdout, R
2 log4j.appender.stdout=org.apache.log4j.ConsoleAppender
3 log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
4 # Pattern to output the caller's file name and line number.
5 log4j.appender.stdout.layout.ConversionPattern=%5p [%t] (%F:%L) - %m%
6 log4j.appender.R=org.apache.log4j.RollingFileAppender
7 log4j.appender.R.File=example.log
8 log4j.appender.R.MaxFileSize=100KB
9 # Keep one backup file
10 log4j.appender.R.MaxBackupIndex=1
11 log4j.appender.R.layout=org.apache.log4j.PatternLayout
12 log4j.appender.R.layout.ConversionPattern=%p %t %c - %m%n
13

```

The screenshot shows the Eclipse IDE interface with the 'SimpleClient.java' file selected. The code is identical to the one in the previous screenshots. The 'Console' tab shows the application's output, which is now much cleaner, only displaying INFO and higher-level logs. The DEBUG logs from the Cassandra driver and application are completely absent, as expected from the modified log4j.properties file.

```

<terminated> CQLClient [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Mar 23, 2016, 11:13:45 AM)
DEBUG [main]([ControlConnection.java:530] - [Control connection] Refreshing node list and token map
DEBUG [main]([ControlConnection.java:175] - [Control connection] Successfully connected to /127.0.0.1:9042
INFO [main]([DCAwareRoundRobinPolicy.java:196] - Using data-center name 'datacenter1' for DCAwareRoundRobinPolicy (if this is incorrect,
INFO [main]([Cluster.java:1308] - New Cassandra host /127.0.0.1:9042 added
DEBUG [cluster1-nio-worker-1]([Connection.java:158] - Connection[/127.0.0.1:9042-2, inFlight=0, closed=false] Connection opened successfully
DEBUG [cluster1-nio-worker-1]([SessionManager.java:308] - Added connection pool for /127.0.0.1:9042
Connected to cluster: Test Cluster
Datacenter: datacenter1; Host: /127.0.0.1; Rack: rack1
DEBUG [cluster1-nio-worker-0]([Cluster.java:2074] - Received event EVENT CREATED KEYSPACE newkeyspace, scheduling delivery
DEBUG [cluster1-worker-0]([ControlConnection.java:288] - [Control connection] Refreshing schema for newkeyspace
DEBUG [cluster1-nio-worker-1]([Cluster.java:2022] - Refreshing schema for newkeyspace
DEBUG [cluster1-worker-0]([ControlConnection.java:530] - [Control connection] Refreshing node list and token map
DEBUG [cluster1-worker-0]([ControlConnection.java:657] - Checking for schema agreement: versions are [469a65db-cb70-3466-912c-34cd31a411
DEBUG [cluster1-worker-0]([ControlConnection.java:530] - [Control connection] Refreshing node list and token map
DEBUG [cluster1-nio-worker-0]([Cluster.java:2074] - Received event EVENT CREATED TABLE newkeyspace.person, scheduling delivery
DEBUG [cluster1-worker-0]([ControlConnection.java:288] - [Control connection] Refreshing schema for newkeyspace.person
DEBUG [cluster1-nio-worker-1]([Cluster.java:2022] - Refreshing schema for newkeyspace.person
DEBUG [cluster1-worker-0]([ControlConnection.java:657] - Checking for schema agreement: versions are [42a028be-6744-34ee-81f9-4d8dc88a53e
id          lname        fname       city      phone1    phone2    phone3
de305d54-75b4-431b-adb2-eb6b9e546003 Earl        Mary        Newton    (854)317-3715 (854)652-2482 (854)592-1753
de305d54-75b4-431b-adb2-eb6b9e546002 White       Joe        Cambridge (617)321-1805 (617)346-1273 (617)319-1835
de305d54-75b4-431b-adb2-eb6b9e546001 Smith       Nick       Boston    (781)397-7171 (781)393-5303 (781)286-8925

```

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

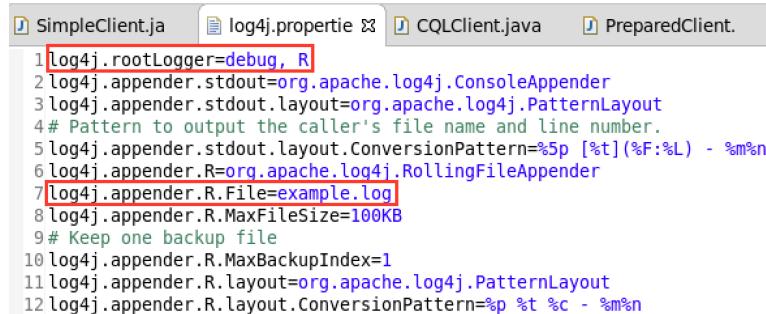
In our **log4j.properties**, the level of the root logger is defined as DEBUG, the DEBUG attached the appender named **stdout, R** to it. It then set the layout and pattern for the appender R.

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**.



```
SimpleClient.java  log4j.properties  CQLClient.java  PreparedClient.java
1 log4j.rootLogger=debug, R
2 log4j.appender.stdout=org.apache.log4j.ConsoleAppender
3 log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
4 # Pattern to output the caller's file name and line number.
5 log4j.appender.stdout.layout.ConversionPattern=%5p [%t] (%F:%L) - %m%n
6 log4j.appender.R=org.apache.log4j.RollingFileAppender
7 log4j.appender.R.File=example.log
8 log4j.appender.R.MaxFileSize=100KB
9 # Keep one backup file
10 log4j.appender.R.MaxBackupIndex=1
11 log4j.appender.R.layout=org.apache.log4j.PatternLayout
12 log4j.appender.R.layout.ConversionPattern=%p %t %c - %m%n
```

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

```

121 DEBUG main io.netty.channel.nio.NioEventLoop - -Dio.netty.noKeySetOptimization: false
122 DEBUG main io.netty.channel.nio.NioEventLoop - -Dio.netty.selectorAutoRebuildThreshold: 512
123 DEBUG main io.netty.util.ResourceLeakDetector - -Dio.netty.leakDetectionLevel: simple
124 DEBUG main com.datastax.driver.core.SystemProperties - com.datastax.driver.DISABLE_COALESCING
125 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.numHeapArenas: 1
126 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.numDirectArenas: 1
127 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.pageSize: 8192
128 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.maxOrder: 11
129 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.chunkSize: 16777216
130 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.tinyCacheSize: 512
131 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.smallCacheSize: 256
132 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.normalCacheSize: 64
133 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.maxCachedBufferCapacity: 1024
134 DEBUG main io.netty.buffer.PooledByteBufAllocator - -Dio.netty.allocator.cacheTrimInterval: 8
135 DEBUG main io.netty.util.internal.ThreadLocalRandom - -Dio.netty.initialSeedUniquifier: 0x6dbd
136 DEBUG main io.netty.buffer.ByteBufUtil - -Dio.netty.allocator.type: unpooled
137 DEBUG main io.netty.buffer.ByteBufUtil - -Dio.netty.threadLocalDirectBufferSize: 65536
138 DEBUG cluster1-nio-worker-0 com.datastax.driver.core.Connection - Connection[/127.0.0.1:9042]
139 DEBUG cluster1-nio-worker-0 io.netty.util.Recycler - -Dio.netty.recycler.maxCapacity.default: 10000
140 DEBUG main com.datastax.driver.core.ControlConnection - [Control connection] Refreshing node
141 DEBUG main com.datastax.driver.core.ControlConnection - [Control connection] Refreshing schema
142 DEBUG main com.datastax.driver.core.ControlConnection - [Control connection] Refreshing node
143 DEBUG main com.datastax.driver.core.ControlConnection - [Control connection] Successfully connected to /127.0.0.1:9042
144 INFO main com.datastax.driver.core.policies.DCAwareRoundRobinPolicy - Using data-center name
145 INFO main com.datastax.driver.core.Cluster - New Cassandra host /127.0.0.1:9042 added
146 DEBUG cluster1-nio-worker-1 com.datastax.driver.core.Connection - Connection[/127.0.0.1:9042]
147 DEBUG cluster1-nio-worker-1 com.datastax.driver.core.Session - Added connection pool for /127.0.0.1:9042
148

```

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

```

38             "phone2 text," +
39             "phone3 text" +
40         ");";
41     }
42
43     public void loadData() {
44         session.execute(
45             "INSERT INTO newkeyspace.person (id, fname, lname, city, phone1, phone2, phone3)
46             VALUES (" +
47             "de305d54-75b4-431b-adb2-eb6b9e546001," +
48             "'Nick','" +
49             "'Smith','" +
50             "'Boston','" +
51             "'(781)397-7171','" +
52             "'(781)393-5303','" +
53             "'(781)286.8875'" +

```

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

**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 `PreparedClient` 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",

```

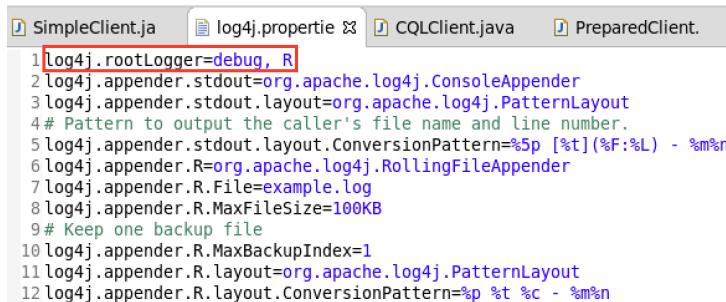
```

        "(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”.

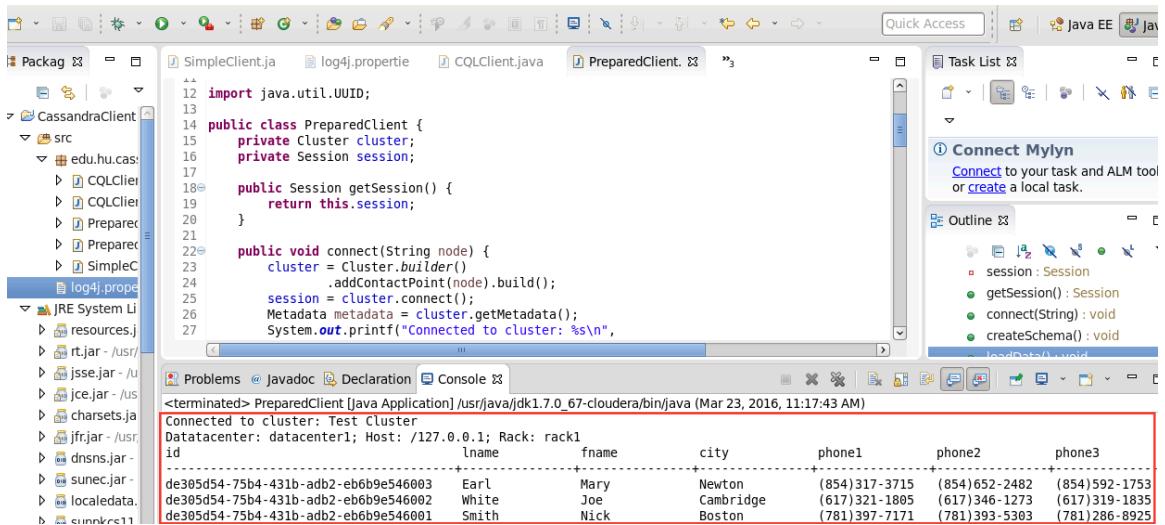
```
cqlsh> DROP KEYSPACE newkeyspace;
```



```

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!

```

<terminated> PreparedClient [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Mar 23, 2016, 11:17:43 AM)
Connected to cluster: Test Cluster
Datacenter: datacenter1; Host: /127.0.0.1; Rack: rack1
id          lname   fname    city      phone1   phone2   phone3
de305d54-75b4-431b-adb2-eb6b9e546003 Earl     Mary     Newton    (854)317-3715 (854)652-2482 (854)592-1753
de305d54-75b4-431b-adb2-eb6b9e546002 White   Joe     Cambridge (617)321-1805 (617)346-1273 (617)319-1835
de305d54-75b4-431b-adb2-eb6b9e546001 Smith   Nick     Boston    (781)397-7171 (781)393-5303 (781)286-8925

```

**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”.

### Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage c  
To learn more about the types of AWS credentials and how they're used, see

- + Password
- + Multi-Factor Authentication (MFA)
- + Access Keys (Access Key ID and Secret Access Key)
- + CloudFront Key Pairs
- + X.509 Certificates
- + Account Identifiers

AWS Services

Compute

- EC2** Virtual Servers in the Cloud
- EC2 Container Service Run and Manage Docker Containers
- Elastic Beanstalk Run and Manage Web Apps
- Lambda Run Code in Response to Events

### Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

**Launch Instance**

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

1. Choose AMI   2. Choose Instance Type   3. Configure Instance   4. Add Storage   5. Tag Instance   6. Configure Security Group   7. Review

**Step 1: Choose an Amazon Machine Image (AMI)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start      Cancel and Exit

My AMIs

AWS Marketplace

**Community AMIs**  

Operating system

Amazon Linux    Cent OS

amzn-ami-hvm-2015.09.1.x86\_64-gp2 - ami-60b6c60a   Select  
Amazon Linux AMI 2015.09.1 x86\_64 HVM GP2  
Root device type: ebs   Virtualization type: hvm

amzn-ami-pv-2015.09.1.x86\_64-ebs - ami-5fb8c835   Select  
Amazon Linux AMI 2015.09.1 x86\_64 PV EBS  
Root device type: ebs   Virtualization type: paravirtual

1 to 50 of 840 AMIs

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

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types ▾ All generations ▾ Show/Hide Columns

Currently selected: t1.micro (Variable ECUs, 1 vCPUs, 0.613 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
Micro instances	t1.micro Free tier eligible	1	0.613	EBS only	-	Very Low
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
General purpose	t2.small	1	2	EBS only	-	Low to Moderate
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
General purpose	t2.large	2	8	EBS only	-	Low to Moderate
General purpose	m4.large	2	8	EBS only	Yes	Moderate

Cancel Previous Review and Launch Next: Configure Instance Details

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.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  Select an existing security group

Security Group ID	Name	Description	Actions
sg-817073e5	default	default VPC security group	<a href="#">Copy to new</a>
sg-85aaa9e1	ElasticMapReduce-master	Master group for Elastic MapReduce created on 2015-05-31T20:59:13.398Z	<a href="#">Copy to new</a>
sg-80aaa9e4	ElasticMapReduce-slave	Slave group for Elastic MapReduce created on 2015-05-31T20:59:13.628Z	<a href="#">Copy to new</a>
sg-adffff3ca	launch-hqui	security group for development environment in EC2	<a href="#">Copy to new</a>

Inbound rules for sg-adffff3ca (Selected security groups: sg-adffff3ca)

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0

Cancel Previous Review and Launch

Review and launch the instance.

### Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

**⚠ Improve your instances' security. Your security group, launch-hqiu, is open to the world.**

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

**AMI Details** [Edit AMI](#)

 amzn-ami-pv-2015.09.1.x86_64-ebs - ami-5fb8c835
Amazon Linux AMI 2015.09.1 x86_64 PV EBS
Root Device Type: ebs Virtualization type: paravirtual

**Instance Type** [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t1.micro	Variable	1	0.613	EBS only	-	Very Low

**Security Groups** [Edit security groups](#)

Security Group ID	Name	Description

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

**Select an existing key pair or create a new key pair** ×

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

**Select a key pair**

I acknowledge that I have access to the selected private key file (ec2hqiu.pem), and that without this file, I won't be able to log into my instance.

[Cancel](#) Launch Instances

The EC2 instance has been successfully launched!

## Launch Status

>Your instances are now launching  
The following instance launches have been initiated: i-9aacf71e [View launch log](#)

Get notified of estimated charges  
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, \$100).

### How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage will accrue until you stop or terminate your instances.

Click [View Instances](#) to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances page.

#### Here are some helpful resources to get you started

- [How to connect to your Linux instance](#)
- [Amazon EC2: User Guide](#)
- [Learn about AWS Free Usage Tier](#)
- [Amazon EC2: Discussion Forum](#)

- 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.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Spot Requests, Reserved Instances, Scheduled Instances, Commands, Dedicated Hosts, Images, AMIs, Bundle Tasks, Elastic Block Store, Volumes, Snapshots, and Network & Security. The main area has tabs for Launch Instance, Connect, and Actions. A search bar at the top says "Filter by tags and attributes or search by keyword". Below it is a table with columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, and Public DNS. One row is highlighted for the instance with ID i-9aacf71e. The "Description" tab is selected, showing detailed information for the instance. Key details include:

Instance ID	Public DNS
i-9aacf71e	ec2-52-87-244-234.compute-1.amazonaws.com

Other visible details in the description tab include:

- Instance state: running
- Instance type: t1.micro
- Private DNS: ip-172-31-6-236.ec2.internal
- Private IPs: 172.31.6.236
- Secondary private IPs: vpc-dfb48aba
- VPC ID: vpc-dfb48aba
- Subnet ID: subnet-66024a11
- Network interfaces: eth0
- Source/dest. check: True
- Platform: -
- IAM role: -
- Key pair name: ec2hqi

On the right side of the description tab, there are sections for Public DNS (with a red box around it), Public IP, Elastic IP, Availability zone, Security groups, Scheduled events, AMI ID, Platform, IAM role, and Key pair name.

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

**Connect To Your Instance**

I would like to connect with  A standalone SSH client  A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to [connect using PuTTY](#))
2. Locate your private key file (ec2hqiui.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:  
chmod 400 ec2hqiui.pem
4. Connect to your instance using its Public DNS:  
ec2-52-87-244-234.compute-1.amazonaws.com

Example:

```
ssh -i "ec2hqiui.pem" ec2-user@ec2-52-87-244-234.compute-1.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

[Close](#)

Follow the instructions and connect to the instance.

```
hqiu@bos-mp9cx>> sudo chmod 400 ec2hqiui.pem

hqiu@bos-mp9cx>> ssh -i "ec2hqiui.pem" ec2-user@ec2-52-87-244-234.compute-1.amazonaws.com
AWS - ec2-user@ip-172-31-6-236:~ - ssh - 95x27
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 ec2hqiui.pem
Password:
hqiu@bos-mp9cx>> ssh -i "ec2hqiui.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 established.
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 the list of known hosts.

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

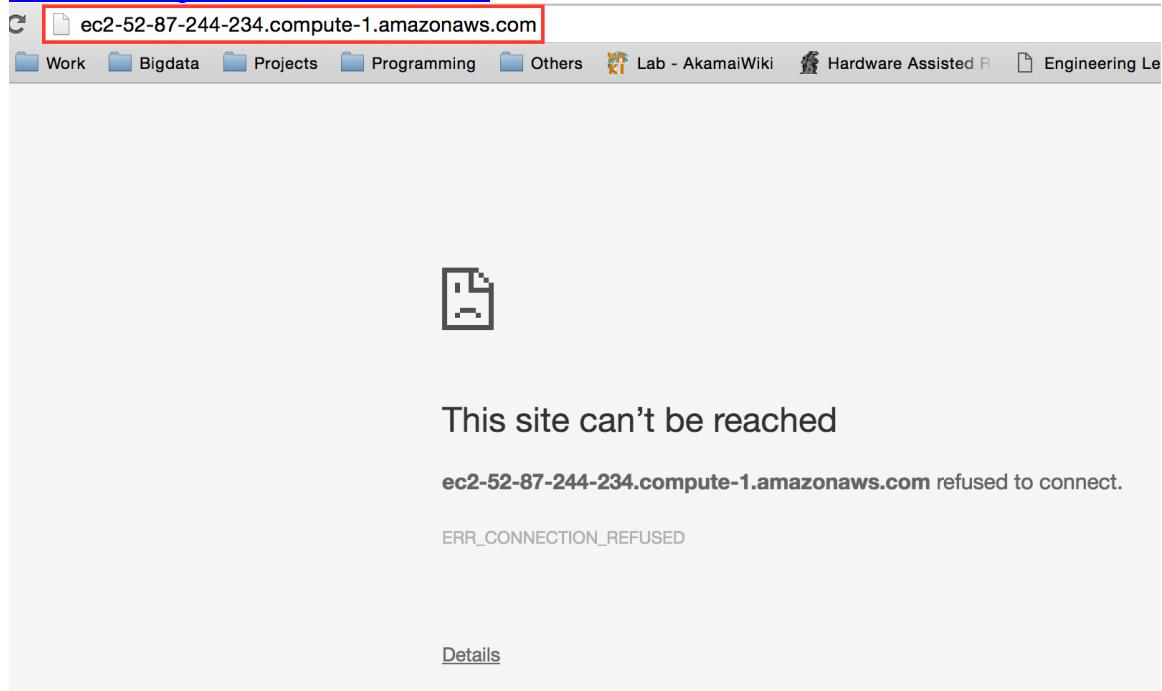
https://aws.amazon.com/amazon-linux-ami/2015.09-release-notes/
23 package(s) needed for security, out of 60 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2016.03 is available.
[ec2-user@ip-172-31-6-236 ~]$
```

```
[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:*
tcp        0      0 127.0.0.1:25             0.0.0.0:*
tcp        0      0 0.0.0.0:52387            0.0.0.0:*
tcp        0      0 0.0.0.0:111              0.0.0.0:*
tcp        0      0 ::::22                  ::::*
tcp        0      0 ::::111                ::::*
tcp        0      0 ::::33938              ::::*
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.*                  LISTEN
tcp        0      0 127.0.0.1:25        0.0.0.*                  LISTEN
tcp        0      0 0.0.0.0:52387       0.0.0.*                  LISTEN
tcp        0      0 0.0.0.0:111         0.0.0.*                  LISTEN
tcp        0      0 ::1:22              ::*:                    LISTEN
tcp        0      0 ::1:111             ::*:                    LISTEN
tcp        0      0 0:::80              ::*:                    LISTEN
tcp        0      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
```



This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP Server at this site is working properly.

#### If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

For information on Amazon Linux AMI , please visit the [Amazon AWS website](#).

#### If you are the website administrator:

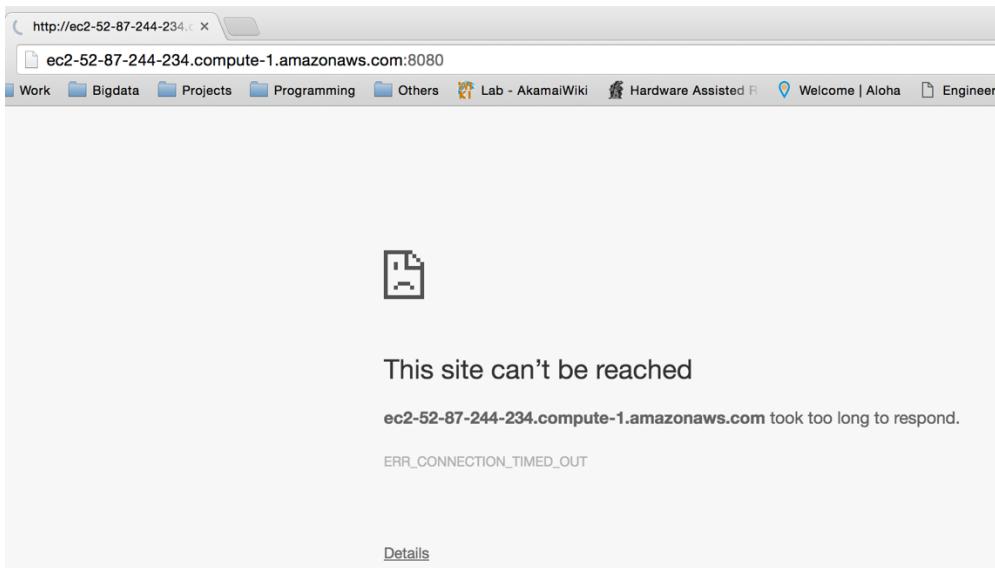
You may now add content to the directory /var/www/html/. Note that so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the /etc/httpd/conf.d/welcome.conf.

You are free to use the image below on web sites powered by the Apache Server:



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 "ec2hqiuy.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 "ec2hqiuy.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      0.0.0.0:22              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 ::1:22                ::*:              LISTEN
tcp        0      0 ::ffff:127.0.0.1:8005  ::*:              LISTEN
tcp        0      0 ::1:8009             ::*:              LISTEN
tcp        0      0 ::1:111              ::*:              LISTEN
tcp        0      0 ::1:8080             ::*:              LISTEN
tcp        0      0 ::1:80               ::*:              LISTEN
tcp        0      0 ::1:33938            ::*:              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.

The screenshot shows the AWS EC2 Dashboard with the 'Security Groups' section selected. A search bar at the top right shows 'sg-adfff3ca'. Below it is a table with columns: Name, Group ID, Group Name, VPC ID, and Description. One row is visible: sg-adfff3ca, sg-adfff3ca, launch-hqiu, vpc-dfb48aba, security group for development environment. Below the table, a detailed view for 'Security Group: sg-adfff3ca' is shown. The 'Inbound' tab is selected. A table lists rules: 
 

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
Custom TCP Rule	TCP	8080	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0

The screenshot shows a web browser window for 'Apache Tomcat/8.0.32'. The address bar shows 'ec2-52-87-244-234.compute-1.amazonaws.com:8080'. The page content includes:
 

- Apache Tomcat/8.0.32**
- If you're seeing this, you've successfully installed Tomcat. Congratulations!**
- Recommended Reading:**
  - [Security Considerations HOW-TO](#)
  - [Manager Application HOW-TO](#)
  - [Clustering/Session Replication HOW-TO](#)
- Developer Quick Start** (with links to Tomcat Setup, First Web Application, Realms & AAA, JDBC DataSources, Examples, Servlet Specifications, and Tomcat Versions).
- Managing Tomcat** (with links to Release Notes, Changelog, Migration Guide, Security Notices, and a note about restricted access to the manager webapp).
- Documentation** (with links to Tomcat 8.0 Documentation, Tomcat 8.0 Configuration, and Tomcat Wiki).
- Getting Help** (with links to FAQ and Mailing Lists, and information about available mailing lists: tomcat-announce, tomcat-users, taglibs-user, tomcat-dev, and tomcat-svn).

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.

Authentication Required

http://ec2-52-87-244-234.compute-1.amazonaws.com:8080  
requires a username and password.

Your connection to this site is not private.

User Name:

Password:

← → C ec2-52-87-244-234.compute-1.amazonaws.com:8080/manager/html

Apps Work Bigdata Projects Programming Others Lab - AkamaiWiki Hardware Assisted Engineering Le

## 401 Unauthorized

You are not authorized to view this page. If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation.

For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

```
<role rolename="manager-gui"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
```

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to access:

- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or jmx interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must

For more information - please see the [Manager App HOW-TO](#).

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 --

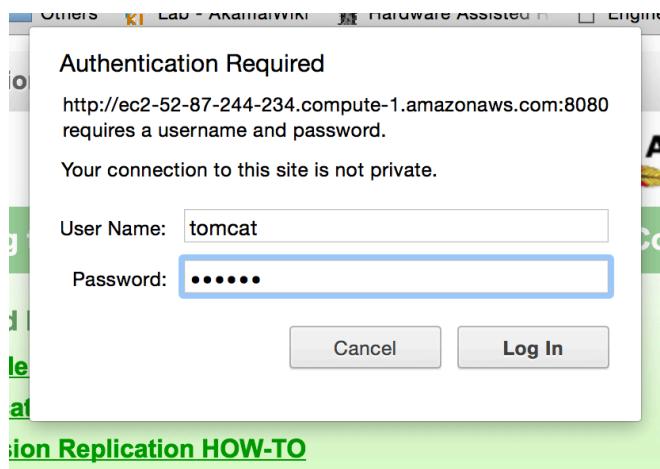
```

```

[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-tomcat-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-tomcat-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!



The screenshot shows the Tomcat Web Application Manager interface. At the top, there's a banner for 'The Apache Software Foundation' with the URL <http://www.apache.org/>. To the right of the banner is a cartoon cat icon. Below the banner, the title 'Tomcat Web Application Manager' is displayed. A message box at the top left says 'Message: OK'. The main content area has tabs for 'Manager', 'List Applications', 'HTML Manager Help', 'Manager Help', and 'Server Status'. Under the 'Manager' tab, there's a table titled 'Applications' listing various Tomcat applications with their paths, versions, display names, running status, session counts, and command buttons (Start, Stop, Reload, Undeploy, Expire sessions).

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

This screenshot shows the AWS Management Console for an EC2 instance named 'i-9aacf71e'. The 'Actions' dropdown menu is open, and the 'Terminate' option is highlighted. The instance status is shown as 'running' with 2/2 checks passing. The instance ID is 'i-9aacf71e' and the public DNS is 'ec2-52-87-244-24.compute-1.amazonaws.com'.

The screenshot shows a confirmation dialog box titled 'Terminate Instances'. It contains a warning message: 'On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost.' Below the warning, a question asks 'Are you sure you want to terminate these instances?' followed by a list of instances: 'i-9aacf71e (ec2-52-87-244-24.compute-1.amazonaws.com)'. At the bottom are 'Cancel' and 'Yes, Terminate' buttons.