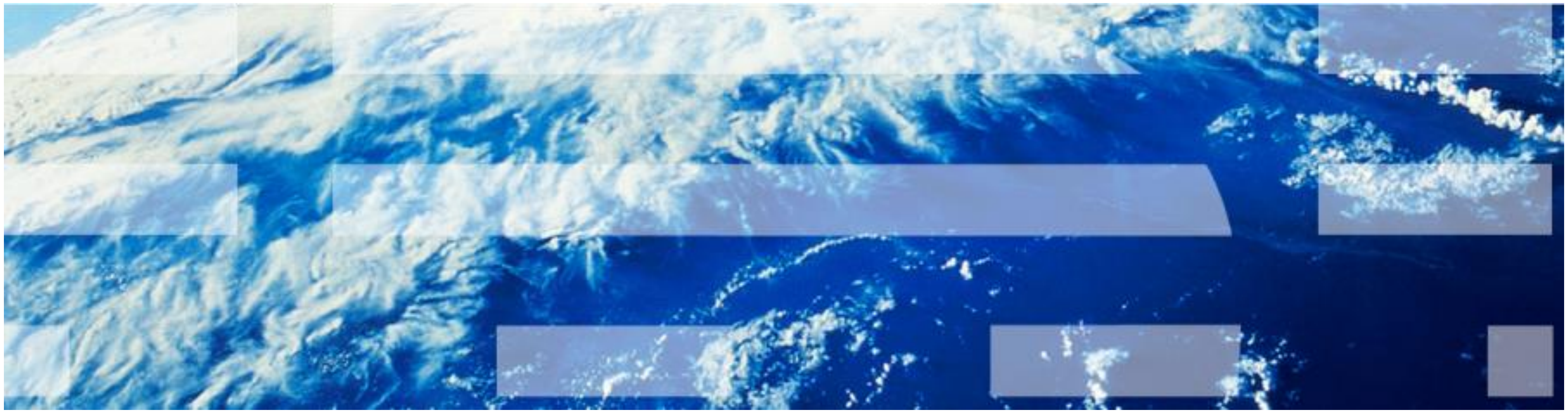
E6893 Big Data Analytics:

Demo Session for HW I

Ruichi Yu, Shuguan Yang, Jen-Chieh Huang

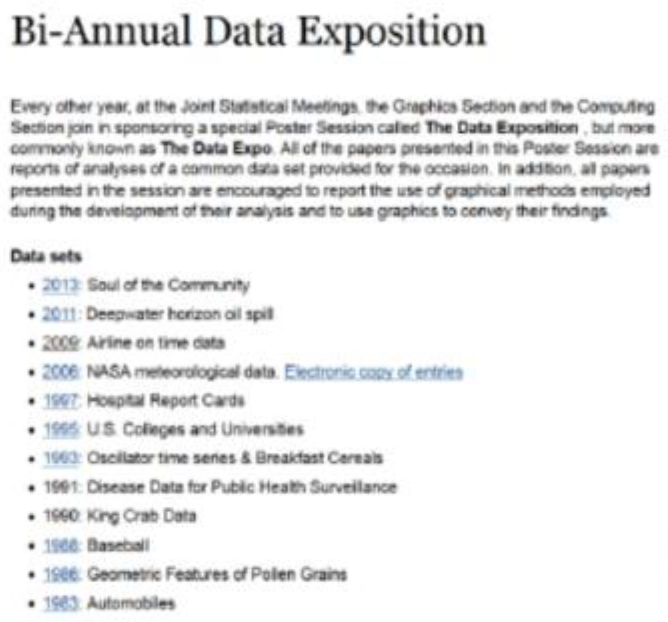
Meng-Yi Hsu, Weizhen Wang, Lin Haung



Oct 2, 2014

Homework #1

1. Install Hadoop
2. Download Airline Data and one your own selected dataset from Stat-Computing.org
3. Learn to use PIG. You can try the example in the reference
4. Use Oozie to schedule a few jobs
5. Try HBase. Use your own example
6. Try Hive. Use your own example


Bi-Annual Data Exposition

Every other year, at the Joint Statistical Meetings, the Graphics Section and the Computing Section join in sponsoring a special Poster Session called **The Data Exposition** , but more commonly known as **The Data Expo**. All of the papers presented in this Poster Session are reports of analyses of a common data set provided for the occasion. In addition, all papers presented in the session are encouraged to report the use of graphical methods employed during the development of their analysis and to use graphics to convey their findings.

Data sets

- 2013: Soul of the Community
- 2011: Deepwater horizon oil spill
- 2009: Airline on time data
- 2006: NASA meteorological data. Electronic copy of entries
- 1997: Hospital Report Cards
- 1995: U.S. Colleges and Universities
- 1993: Oscillator time series & Breakfast Cereals
- 1991: Disease Data for Public Health Surveillance
- 1990: King Crab Data
- 1988: Baseball
- 1986: Geometric Features of Pollen Grains
- 1983: Automobiles

http://stat-computing.org/dataexpo/

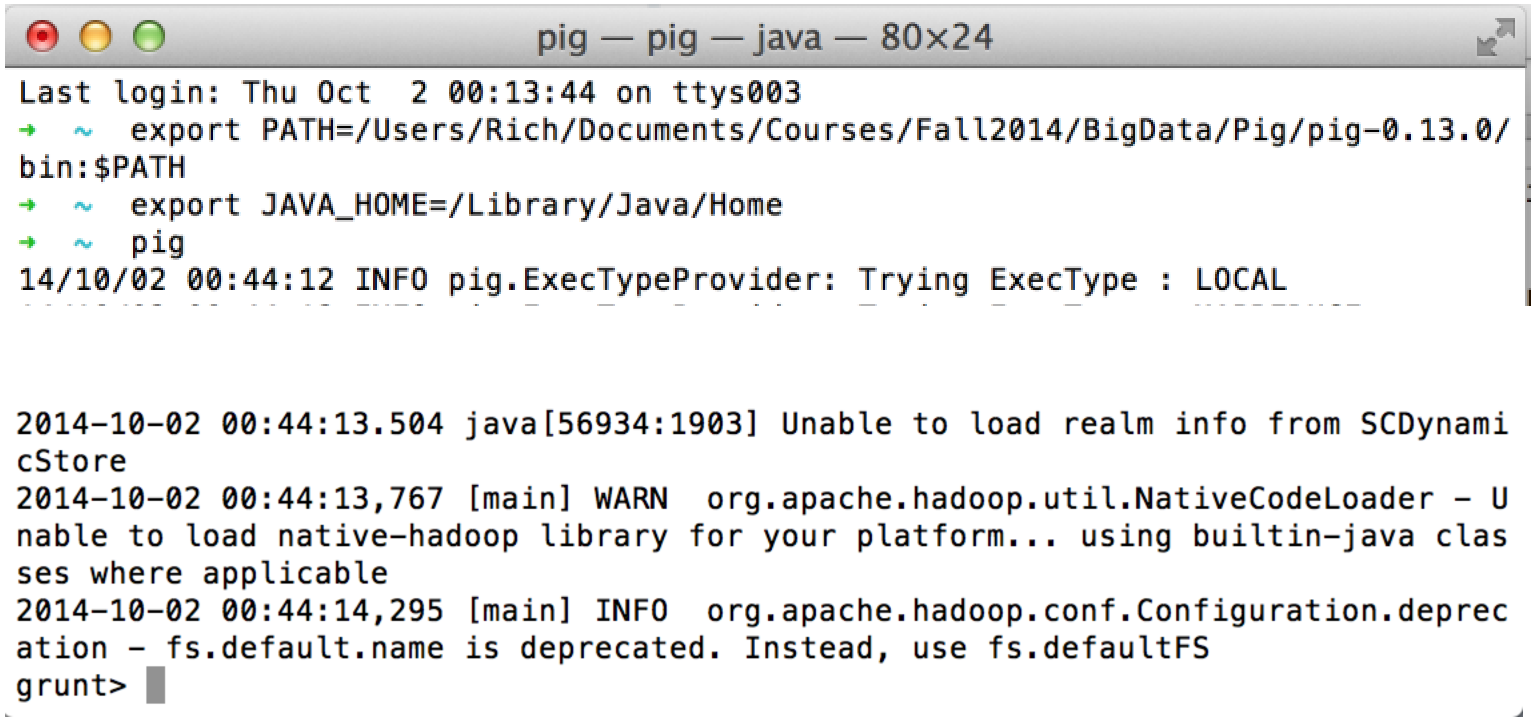
Part I: Pig installation and Demo

Pig is a platform for **analyzing large data sets** that consists of a **high-level language** for expressing data analysis programs, coupled with infrastructure for evaluating these programs.

1. Installation of Pig:
https://pig.apache.org/docs/r0.7.0/setup.html

Download pig and run following sentence:

```
export PATH=/<my-path-to-pig>/pig-n.n.n/bin:$PATH
```



pig — pig — java — 80×24

```
Last login: Thu Oct  2 00:13:44 on ttys003
➜  ~  export PATH=/Users/Rich/Documents/Courses/Fall2014/BigData/Pig/pig-0.13.0/
bin:$PATH
➜  ~  export JAVA_HOME=/Library/Java/Home
➜  ~  pig
14/10/02 00:44:12 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
```

```
2014-10-02 00:44:13.504 java[56934:1903] Unable to load realm info from SCDynami
cStore
2014-10-02 00:44:13,767 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - U
nable to load native-hadoop library for your platform... using builtin-java clas
ses where applicable
2014-10-02 00:44:14,295 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> 
```

2. Running pig in local mode:

```
pig -x local
```

```
movies = LOAD '/Users/Rich/Documents/Courses/Fall2014/BigData/Pig/movies_data.csv' USING PigStorage(',') as (id,name,year,rating,duration);
```

```
DUMP movies
```

Filter:

List the movies that were released between 1950 and 1960

```
movies_1950_1960 = FILTER movies BY (float)year>1949 and (float)year<1961;

store movies_1950_1960 into '/Users/Rich/Desktop/Demo/movies_1950_1960';
```

Foreach Generate:

List movie names and their duration (in minutes)

```
movies_name_duration = foreach movies
generate name, (float)duration/3600;

store movies_name_duration into
'/Users/Rich/Desktop/Demo/movies_name_duration';
```

Order:

List all movies in descending order of year

movies_year_sort =**order** movies by year **desc**;

store movies_year_sort into '/Users/Rich/Desktop/Demo/movies_year_sort';

3. Running pig with HDFS:

```
pig
```

Run HDFS first:

```
ssh localhost

cd /usr/local/Cellar/hadoop/2.5.0

sbin/start-dfs.sh
```

3. Upload file to HDFS:

Make a directory:

```
bin/hdfs dfs -mkdir /PigSource
```

Upload a file:

```
bin/hdfs dfs -put
/Users/Rich/Documents/Courses/Spring201
4/CloudComputing/HW/MINI5/movies_dat
a.csv /PigSource
```

4. Run pig in grunt with HDFS:

```
   pig

movies = LOAD
'/PigSource/movies_data.csv' USING
PigStorage(',') as
(id,name,year,rating,duration);

DUMP movies
```

5. Run .pig file with HDFS:

pig

pig /Users/Rich/Documents/Courses/Fall2014/BigData/Pig/run1.pig

Part II: Hbase installation and Demo

HBase is an open source, non-relational, distributed database modeled after Google's Big Table and written in Java.

It runs on top of HDFS, and can serve as input and output for MapReduce jobs run in Hadoop.

Access through Java API and Pig.

Hbase Configuration:

Download and configure Hbase by following the instruction online.

To run Hbase, under your Hbase path:

```
$bin/start-hbase.sh
```

Enter shell

```
$bin/hbase shell
```

Also visit localhost:60010 to check Hbase webUI

Hbase Command:

Create:

```
hbase>create ‘table’, ‘cf’
hbase>put ’table', ’r1', ’cf:c1', 'value1’
hbase>scan ‘table’
```

Also we can do count, get, delete and drop. etc much like other DB systems.

Part III: Hive installation and Demo

Simple demo for Hive

Install steps:

Mac: you can first install brew

```
$ brew install Hive
```

Linux: cd ~/Downloads

```
wget
http://mirror.cc.columbia.edu/pub/software/apache/hive/
hive-0.13.1/apache-hive-0.13.1-bin.tar.gz
cp apache-hive-0.13.1-bin.tar.gz ~
cd ~
tar zxvf apache-hive-0.13.1-bin.tar.gz
mv apache-hive-0.13.1-bin hive
cd hive
cd bin
./hive
```

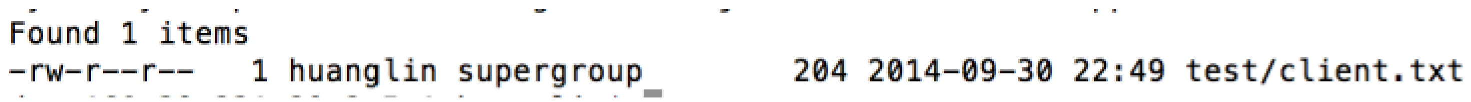
Simple demo for Hive

Step 1. start Hadoop, create a file under /user/yourname

```
hadoop fs -mkdir test
```

Step 2. put your dataset into hdfs

```
hadoop fs -put (your dataset path) test
hadoop fs -ls test
```

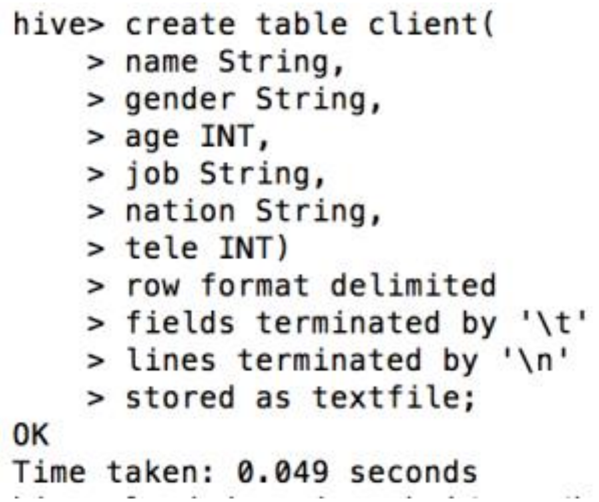


```
Found 1 items
-rw-r--r--   1 huanglin supergroup        204 2014-09-30 22:49 test/client.txt
```

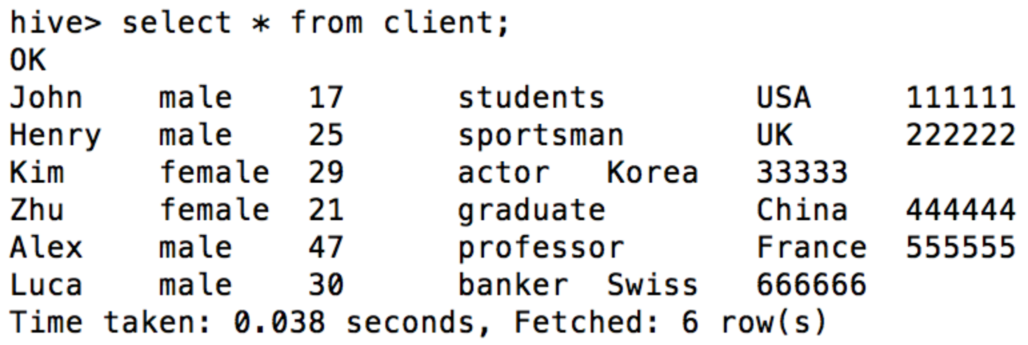
Simple demo for Hive

Step 3. create the table (SQL) in Hive shell

```
hive
```



```
hive> create table client(
    > name String,
    > gender String,
    > age INT,
    > job String,
    > nation String,
    > tele INT)
    > row format delimited
    > fields terminated by '\t'
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.049 seconds
```



```
hive> select * from client;
OK
John    male    17      students        USA     111111
Henry   male    25      sportsman       UK      222222
Kim     female  29      actor   Korea   33333
Zhu     female  21      graduate        China   444444
Alex    male    47      professor       France  555555
Luca    male    30      banker  Swiss   666666
Time taken: 0.038 seconds, Fetched: 6 row(s)
```

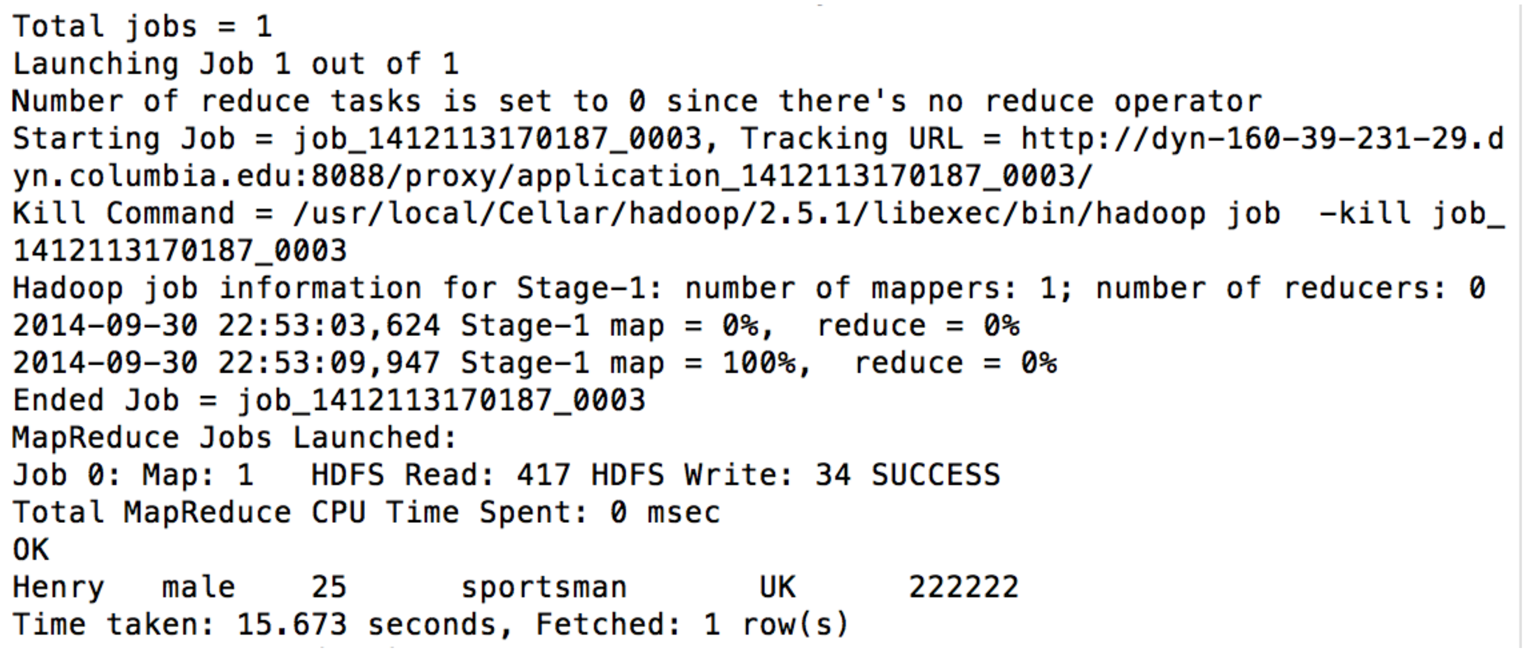
Step 4. store the data into table

```
    load data inpath '/user/yourname/test' into table
client;
```

You can see your data has already been put into the table

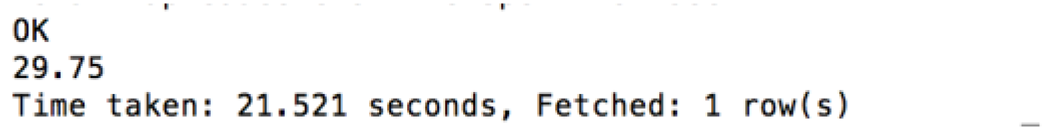
Simple demo for Hive

1. select * from client where name='Henry';



```
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1412113170187_0003, Tracking URL = http://dyn-160-39-231-29.d
yn.columbia.edu:8088/proxy/application_1412113170187_0003/
Kill Command = /usr/local/Cellar/hadoop/2.5.1/libexec/bin/hadoop job  -kill job_
1412113170187_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2014-09-30 22:53:03,624 Stage-1 map = 0%,  reduce = 0%
2014-09-30 22:53:09,947 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_1412113170187_0003
MapReduce Jobs Launched:
Job 0: Map: 1   HDFS Read: 417 HDFS Write: 34 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Henry   male    25      sportsman       UK      222222
Time taken: 15.673 seconds, Fetched: 1 row(s)
```

2. select avg(age) from client where gender='male';

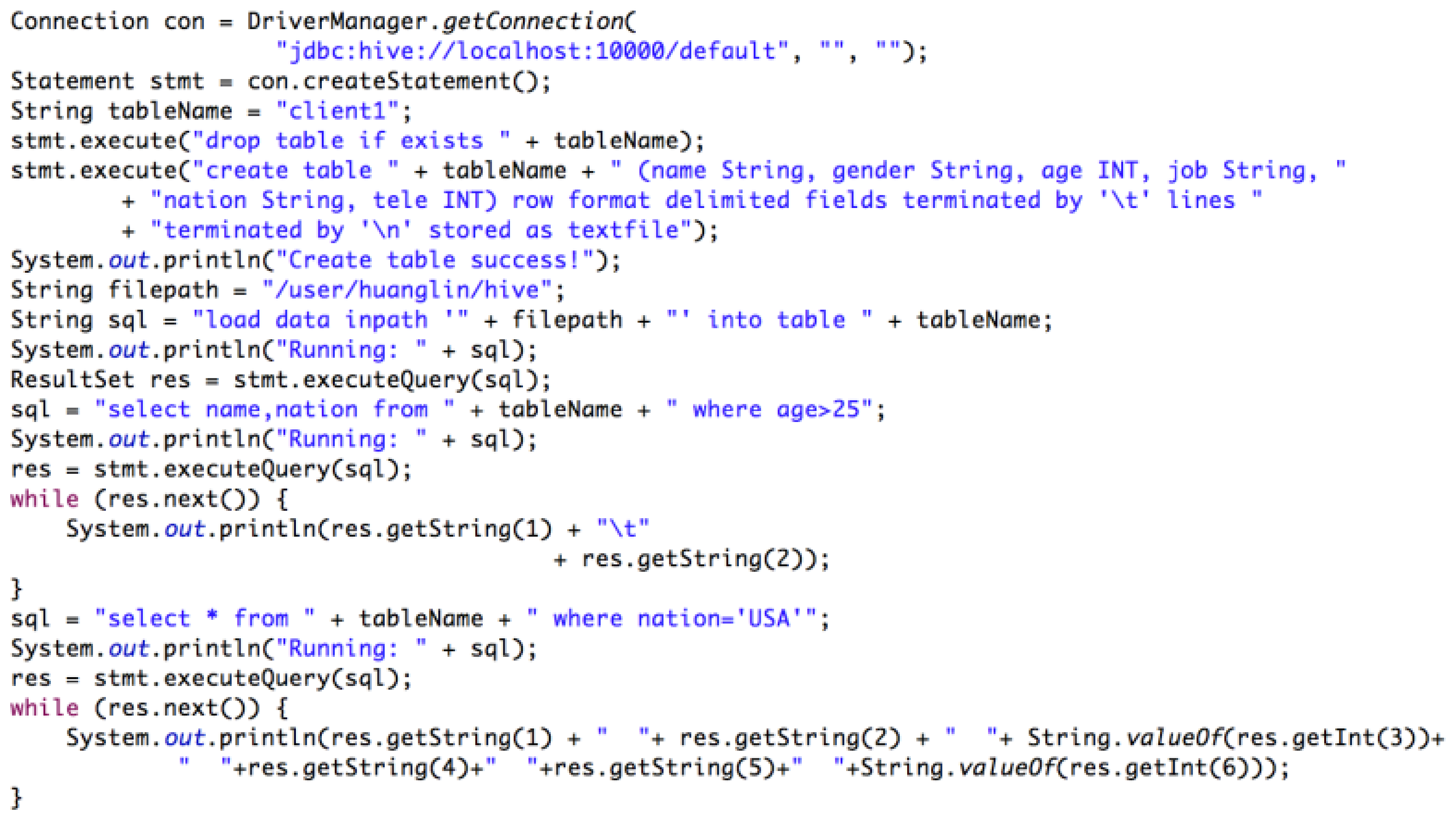


```
OK
29.75
Time taken: 21.521 seconds, Fetched: 1 row(s)
```

Simple demo for Hive

Use JDBC(JAVA) to access data in Hive Server

Other choice: Python, PHP etc.



```
Connection con = DriverManager.getConnection(
                   "jdbc:hive://localhost:10000/default", "", "");
Statement stmt = con.createStatement();
String tableName = "client1";
stmt.execute("drop table if exists " + tableName);
stmt.execute("create table " + tableName + " (name String, gender String, age INT, job String, "
        + "nation String, tele INT) row format delimited fields terminated by '\t' lines "
        + "terminated by '\n' stored as textfile");
System.out.println("Create table success!");
String filepath = "/user/huanglin/hive";
String sql = "load data inpath '" + filepath + "' into table " + tableName;
System.out.println("Running: " + sql);
ResultSet res = stmt.executeQuery(sql);
sql = "select name,nation from " + tableName + " where age>25";
System.out.println("Running: " + sql);
res = stmt.executeQuery(sql);
while (res.next()) {
    System.out.println(res.getString(1) + "\t"
                                   + res.getString(2));
}
sql = "select * from " + tableName + " where nation='USA'";
System.out.println("Running: " + sql);
res = stmt.executeQuery(sql);
while (res.next()) {
    System.out.println(res.getString(1) + "   "+ res.getString(2) + "   "+ String.valueOf(res.getInt(3))+
            "   "+res.getString(4)+"   "+res.getString(5)+"   "+String.valueOf(res.getInt(6)));
}
```

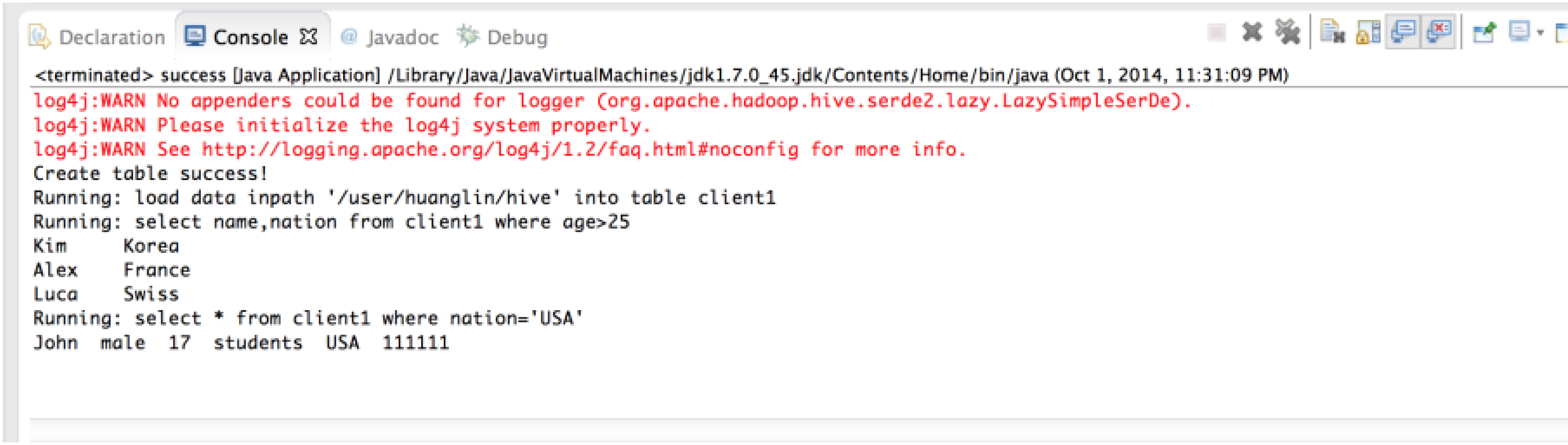
Simple demo for Hive

Open the Hive (Starting Hive Thrift Server)

```
serverhive --service hiveserver -p 10000
```

URL: jdbc:hive://localhost:10000/default

Result in the console

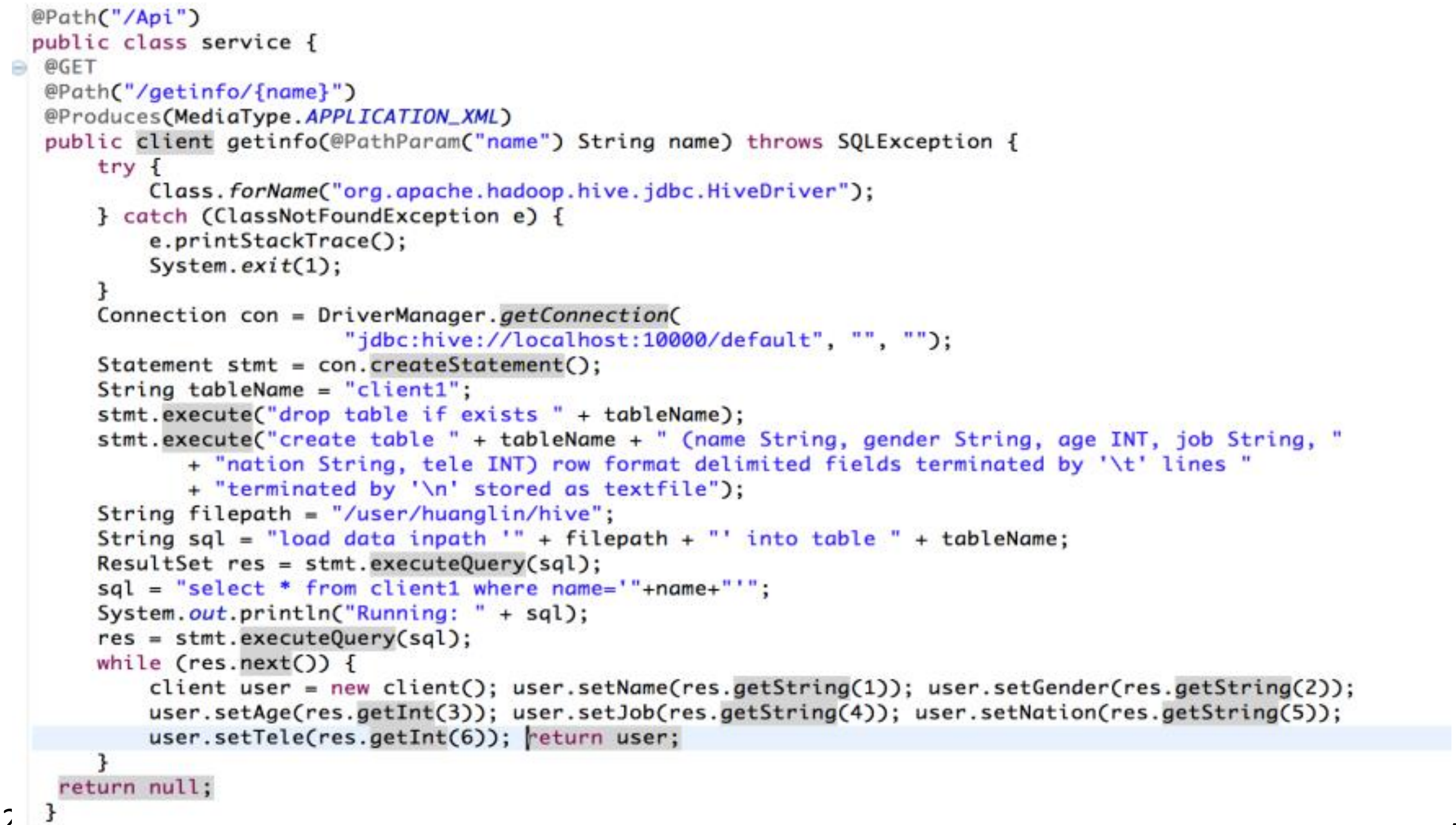


```
<terminated> success [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_45.jdk/Contents/Home/bin/java (Oct 1, 2014, 11:31:09 PM)
log4j:WARN No appenders could be found for logger (org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Create table success!
Running: load data inpath '/user/huanglin/hive' into table client1
Running: select name,nation from client1 where age>25
Kim     Korea
Alex    France
Luca    Swiss
Running: select * from client1 where nation='USA'
John  male  17  students  USA  111111
```

Simple demo for Hive

Example. Jetty Server with Jersey, restful web service

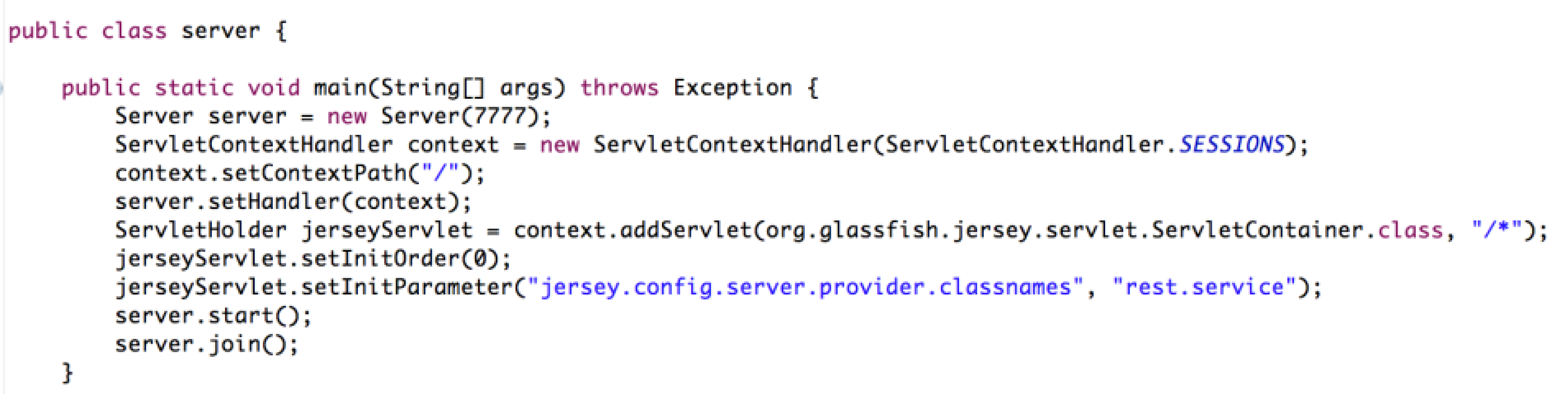
Create simple API (Http request, XML/JSON format output)



```
@Path("/Api")
public class service {
  @GET
  @Path("/getinfo/{name}")
  @Produces(MediaType.APPLICATION_XML)
  public client getinfo(@PathParam("name") String name) throws SQLException {
      try {
          Class.forName("org.apache.hadoop.hive.jdbc.HiveDriver");
      } catch (ClassNotFoundException e) {
          e.printStackTrace();
          System.exit(1);
      }
      Connection con = DriverManager.getConnection(
                          "jdbc:hive://localhost:10000/default", "", "");
      Statement stmt = con.createStatement();
      String tableName = "client1";
      stmt.execute("drop table if exists " + tableName);
      stmt.execute("create table " + tableName + " (name String, gender String, age INT, job String, "
              + "nation String, tele INT) row format delimited fields terminated by '\t' lines "
              + "terminated by '\n' stored as textfile");
      String filepath = "/user/huanglin/hive";
      String sql = "load data inpath '" + filepath + "' into table " + tableName;
      ResultSet res = stmt.executeQuery(sql);
      sql = "select * from client1 where name='"+name+"'";
      System.out.println("Running: " + sql);
      res = stmt.executeQuery(sql);
      while (res.next()) {
          client user = new client(); user.setName(res.getString(1)); user.setGender(res.getString(2));
          user.setAge(res.getInt(3)); user.setJob(res.getString(4)); user.setNation(res.getString(5));
          user.setTele(res.getInt(6)); return user;
      }
    return null;
  }
```

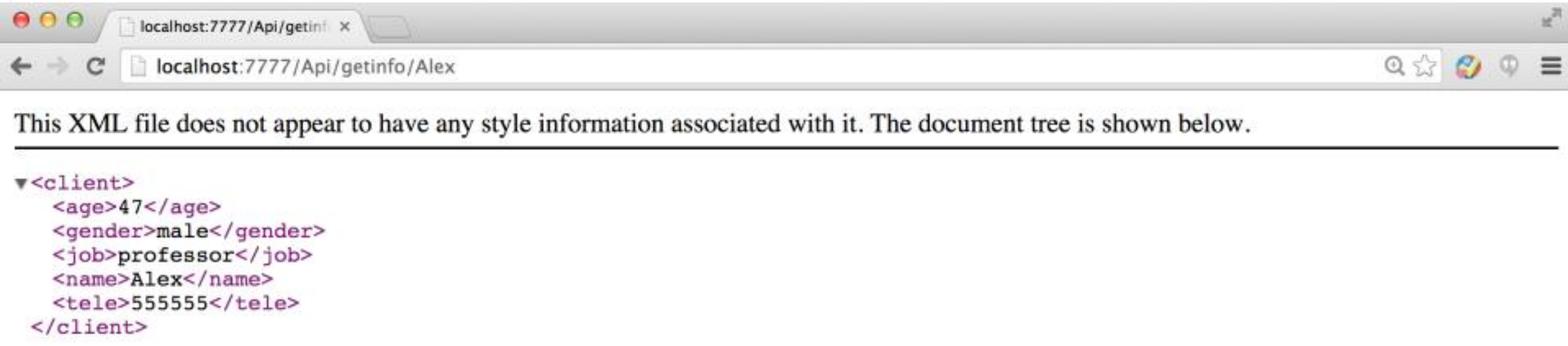
Simple demo for Hive

Example. Jetty Server with Jersey, restful web service



```
public class server {

    public static void main(String[] args) throws Exception {
        Server server = new Server(7777);
        ServletContextHandler context = new ServletContextHandler(ServletContextHandler.SESSIONS);
        context.setContextPath("/");
        server.setHandler(context);
        ServletHolder jerseyServlet = context.addServlet(org.glassfish.jersey.servlet.ServletContainer.class, "/*");
        jerseyServlet.setInitOrder(0);
        jerseyServlet.setInitParameter("jersey.config.server.provider.classnames", "rest.service");
        server.start();
        server.join();
    }
```



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<client>
   <age>47</age>
   <gender>male</gender>
   <job>professor</job>
   <name>Alex</name>
   <tele>555555</tele>
 </client>
```

Part IV: Oozie installation and Demo

- Before Getting Started
- How to solve
- Virtual Machine Environment Spec
- Run and Test

Before Getting Started

- Oozie isn’t compatible with Hadoop 2.
- BigTop came for rescue.
- However, BigTop doesn’t supported Hadoop 2 now.

SO HOW DO WE SUPPOSE TO DO ?

Don’t worry, we found the solution

- We have setup a virtual machine environment for you, which already includes Pig, Hadoop and Oozie installed and configured.
- We have already verified that Pig, Hadoop and Oozie can work with each other in our provided virtual environment with no conflict.

Spec

- OS: Linux OS (Ubuntu 14.04)
- Hadoop: 2.5.0 (locate in /usr/local)
- Ooze: 4.0.1 (locate in /usr/local)
- Maven: 3.0.5
- Pig: 0.13 (locate in /usr/local)
- Java: 1.6

NOTE: JAVA_HOME and HADOOP_PREFIX has already setup

How to Run Oozie (1)

(1) <u>(IMPORTANT) SSH to localhost and start HDFS</u>

```
$ ssh localhost
$ cd /usr/local/hadoop-2.5.0
$ ./sbin/start-dfs.sh
$ ./sbin/start-yarn.sh
```

How to Run Oozie (1)

(2) <u>6 Nodes should be running shown as below:</u>

```
$ jps
8633 Jps
5118 NameNode
5238 DataNode
5411 SecondaryNameNode
5625 ResourceManager
5750 NodeManager
```

How to Run Oozie (2)

(3) Start Oozie

```
$ cd /usr/local/oozie-4.0.1
$ ./bin/oozied.sh start
```

(4) Check Oozie running status

```
$ cd /usr/local/oozie-4.0.1
$ ./bin/oozie admin -oozie
http://localhost:11000/oozie -status
NORMAL
```

How to Run Oozie (3)

(5) Untar Oozie example

```
$ cd /usr/local/oozie-4.0.1
$ tar –zxvf oozie-examples.tar.gz
```

(6-1) Change Namenode port number from 8020 to 9000

```
$ cd /usr/local/oozie-4.0.1/examples
$ find ./ -type f -exec sed -i -e 's/8020/9000/g' {} \;
```

(6-2) Change Jobtracker port number from 8021 to 8088

```
$ cd /usr/local/oozie-4.0.1/examples
$ find ./ -type f -exec sed -i -e 's/8021/8088/g' {} \;
```

How to Run Oozie (4)

(7) Submit a job to Oozie

```
$ cd /usr/local/oozie-4.0.1
$ ./oozie job -oozie http://localhost:11000/oozie -config
    examples/apps/map-reduce/job.properties -run
[YOUR_JOB_ID] will return here
```

(8) Check the job status

```
$ cd /usr/local/oozie-4.0.1
$ ./oozie job -oozie http://localhost:11000/oozie -info
    [YOUR_JOB_ID]
```

How to Test Oozie

(1) Check the Oozie log file logs/oozie.log to ensure Oozie started properly.

(2) Using the Oozie command line tool check the status of Oozie:

```
$ cd /usr/local/oozie-4.0.1
$ ./bin/oozie admin –oozie
    http://localhost:11000/oozie  -status
```

(3) Using a browser go to the oozie web console Oozie status should be NORMAL

Questions?