Big Data using Hadoop

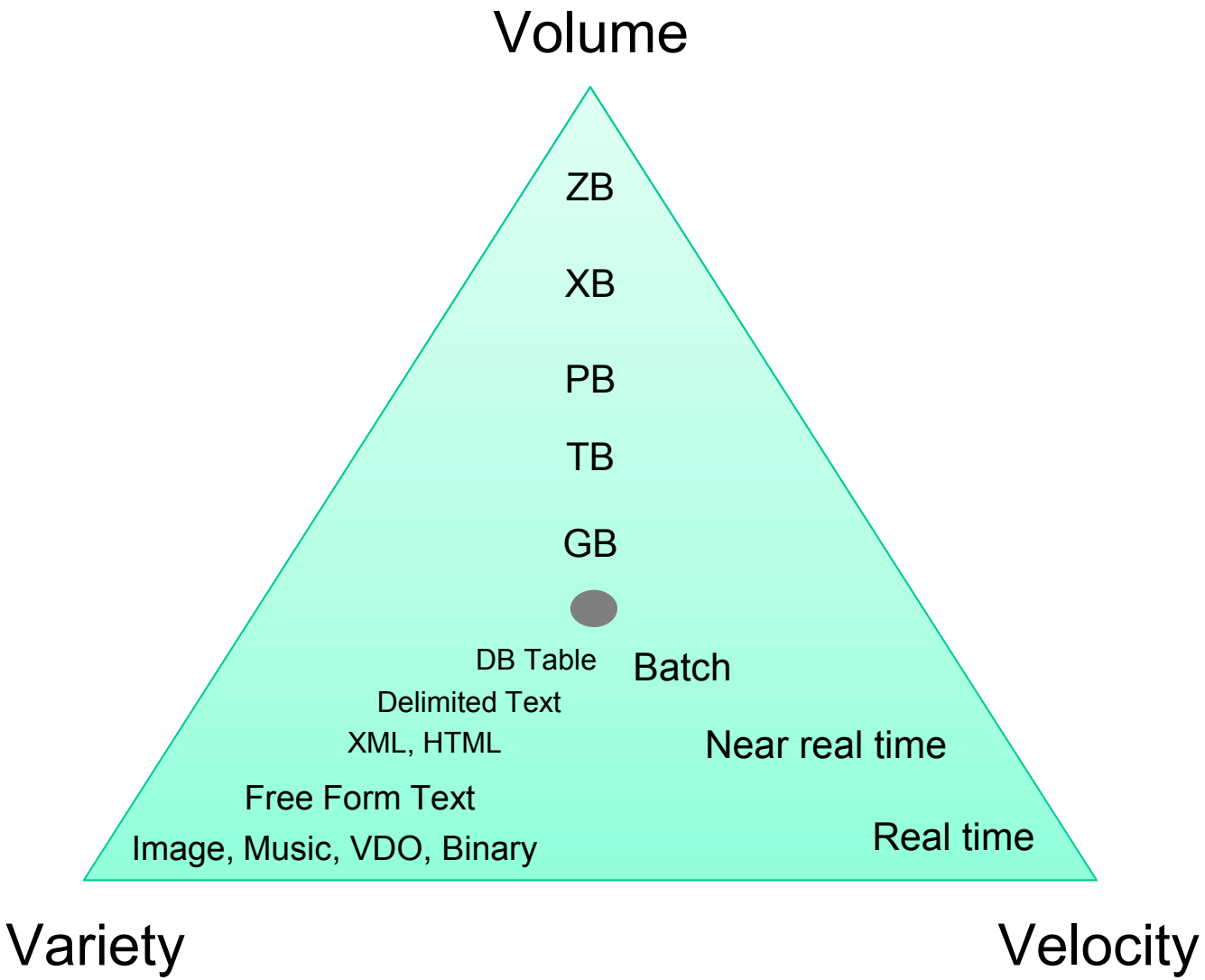
Hands On Workshop

Dr.Thanachart Numnonda
Certified Java Programmer
thanachart@imcinstitute.com

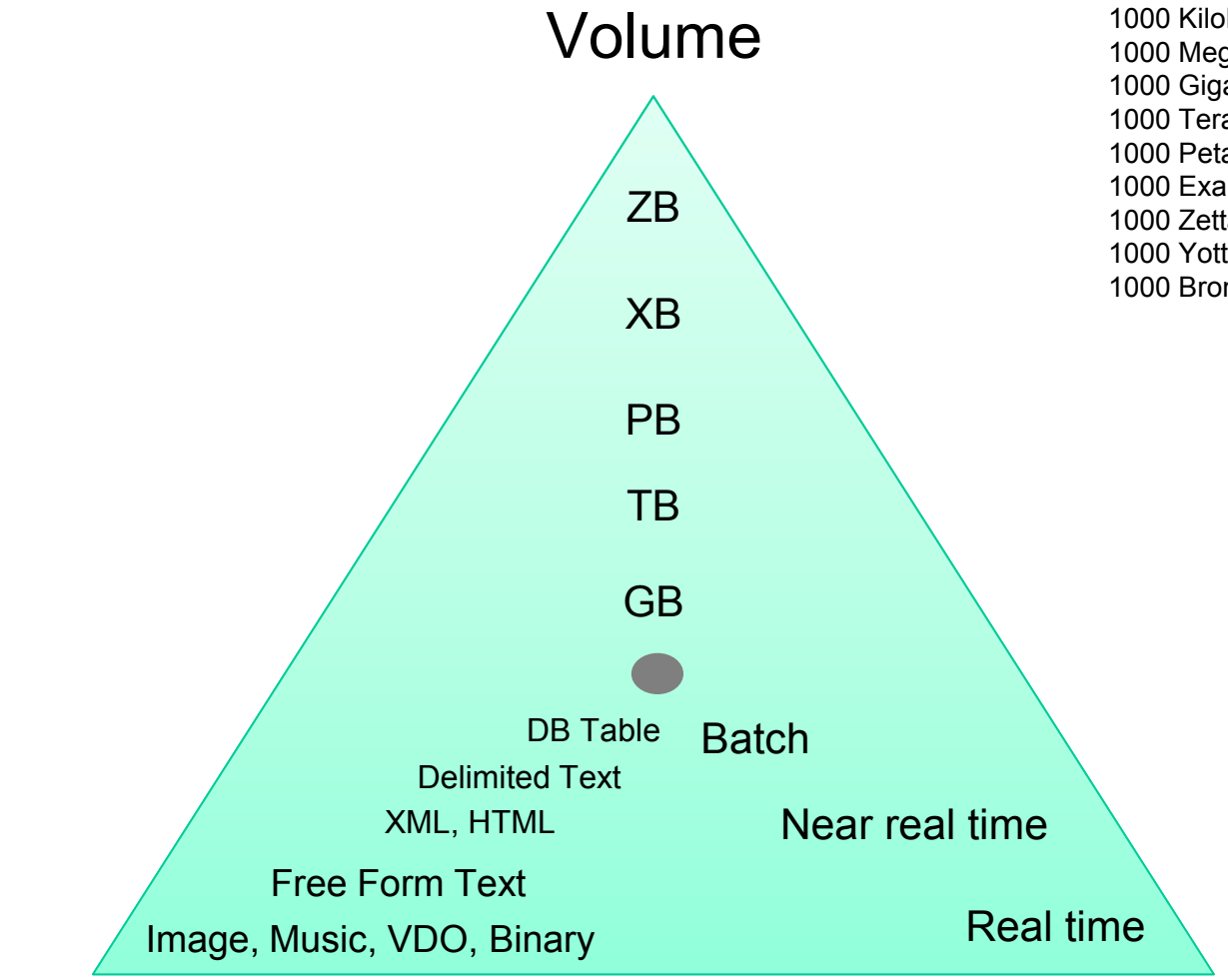
Danairat T.
Certified Java Programmer, TOGAF – Silver
danairat@gmail.com, +66-81-559-1446

YARN

Big Data Introduction

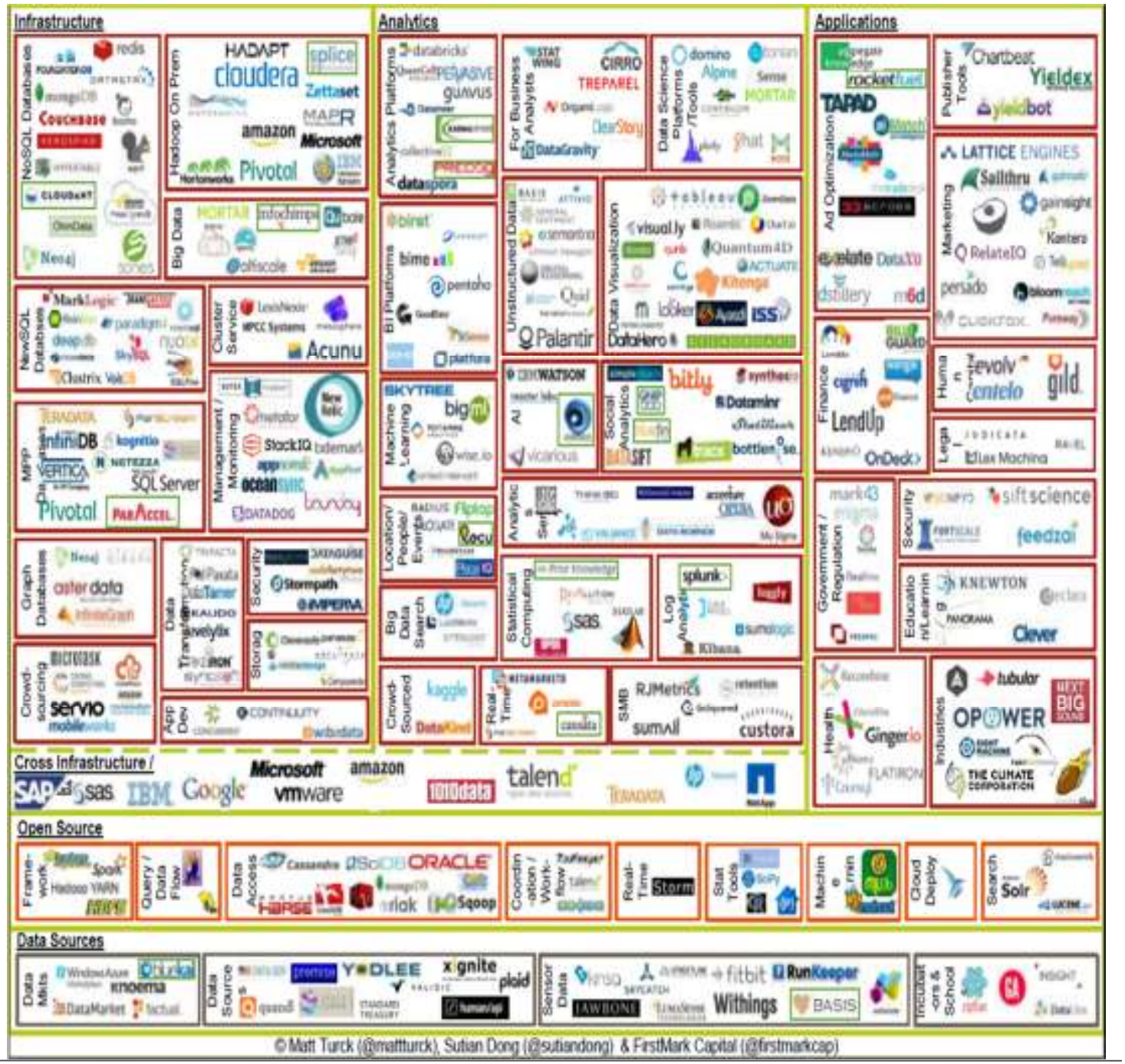


Big Data Introduction

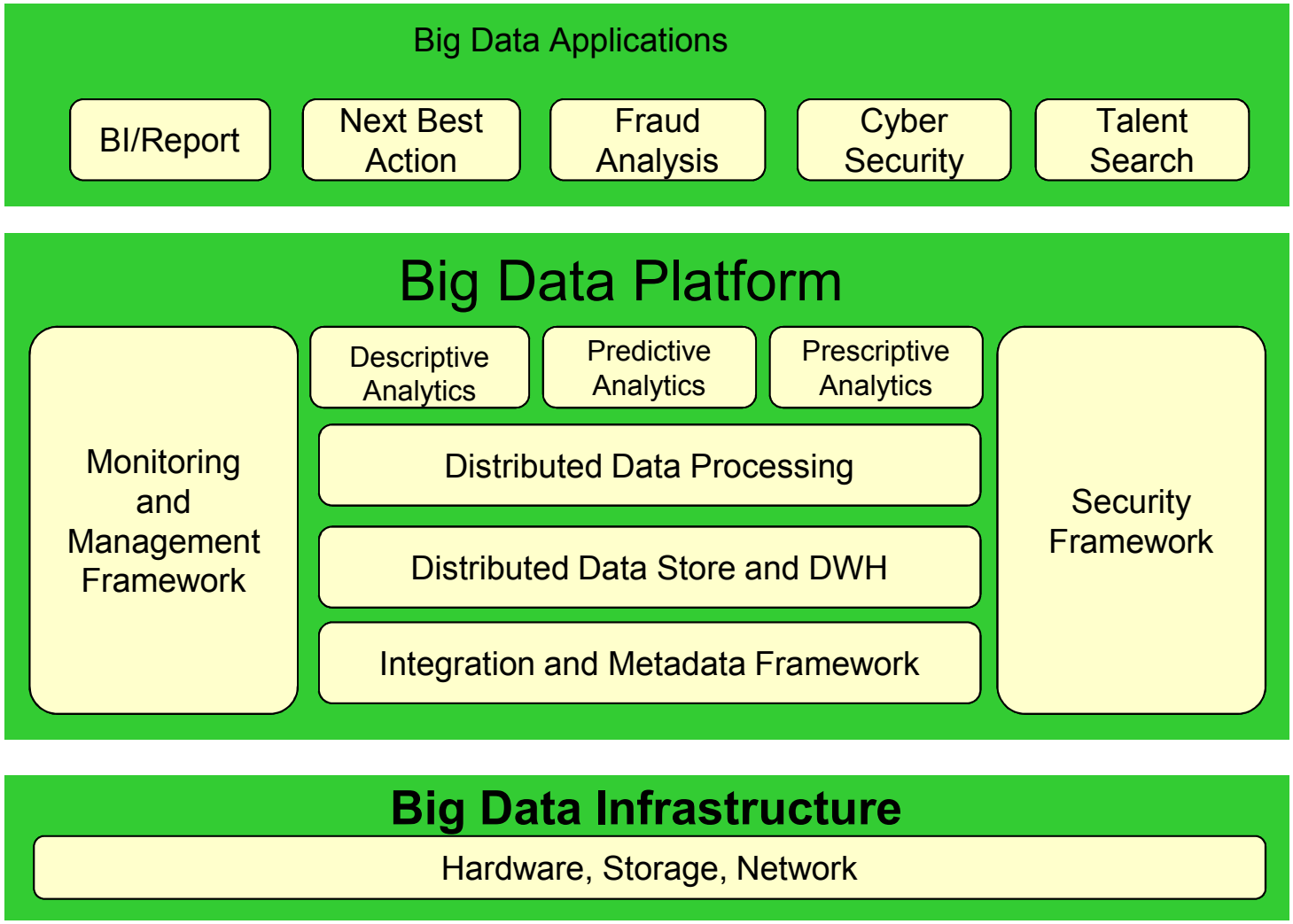


1000 Kilobytes = 1 Megabyte
1000 Megabytes = 1 Gigabyte
1000 Gigabytes = 1 Terabyte
1000 Terabytes = 1 Petabyte
1000 Petabytes = 1 Exabyte
1000 Exabytes = 1 Zettabyte
1000 Zettabytes = 1 Yottabyte
1000 Yottabytes = 1 Brontobyte
1000 Brontobytes = 1 Geopbyte

Big Data Players



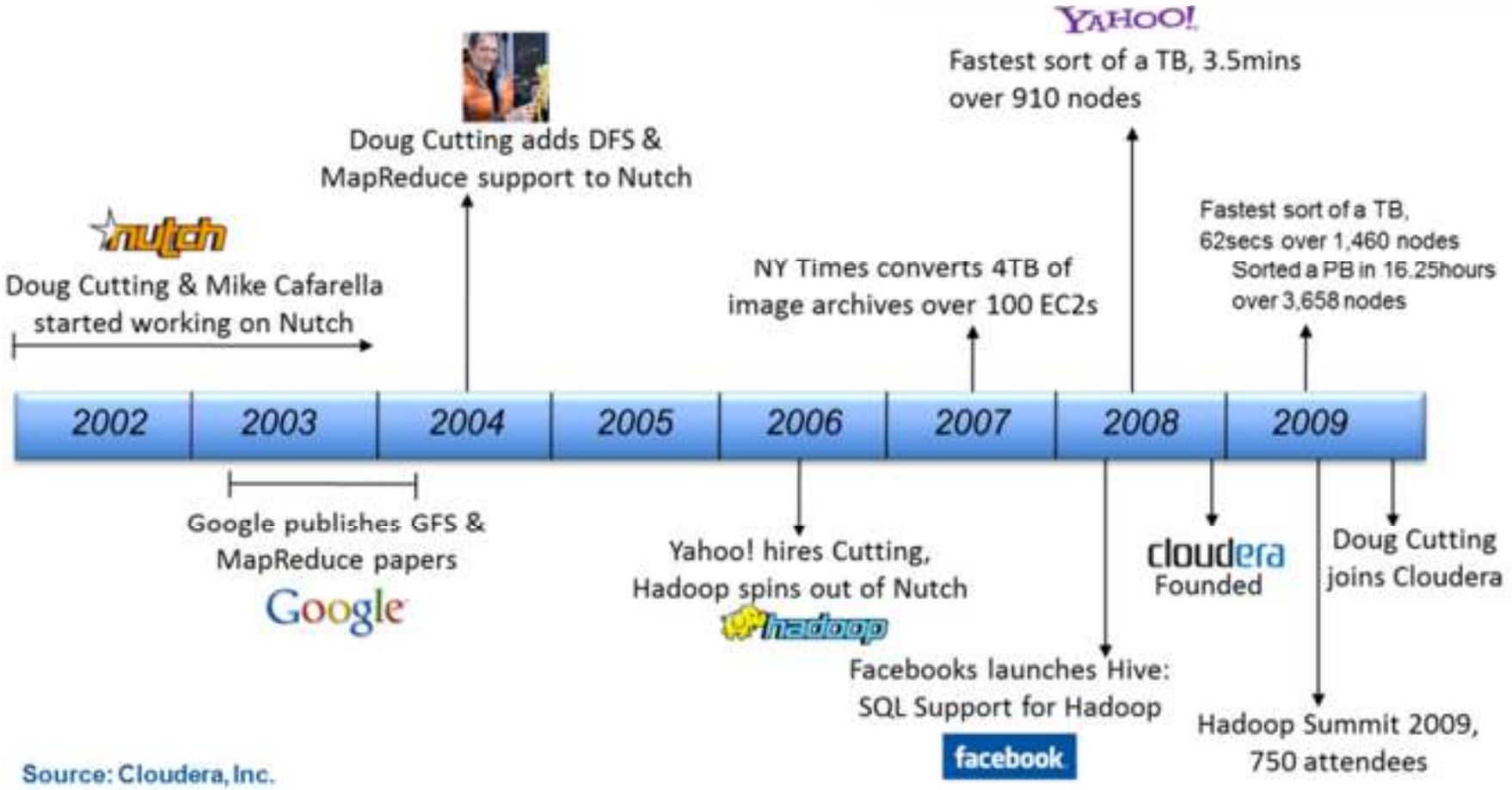
Big Data Architecture



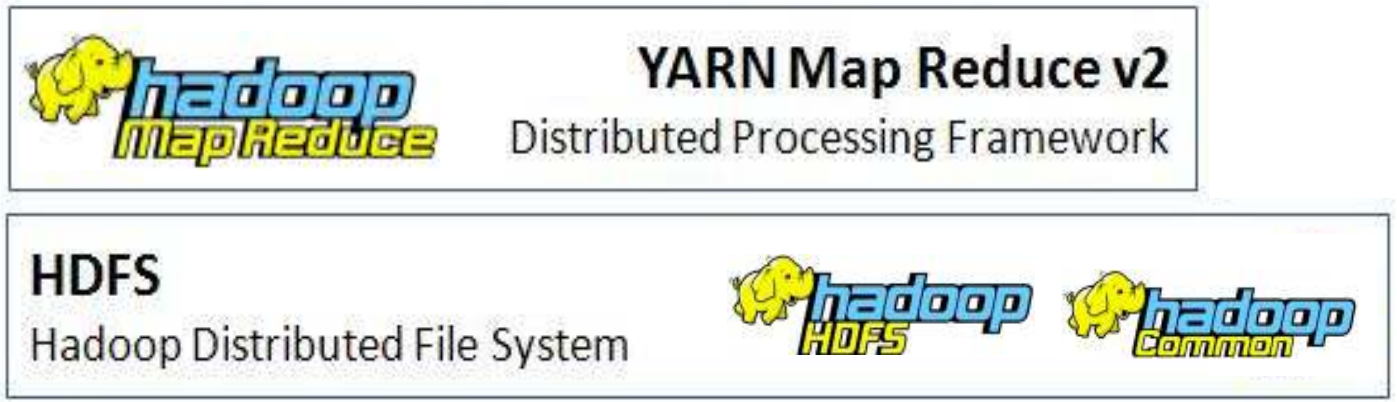
Lecture: Hadoop

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

Hadoop Timeline

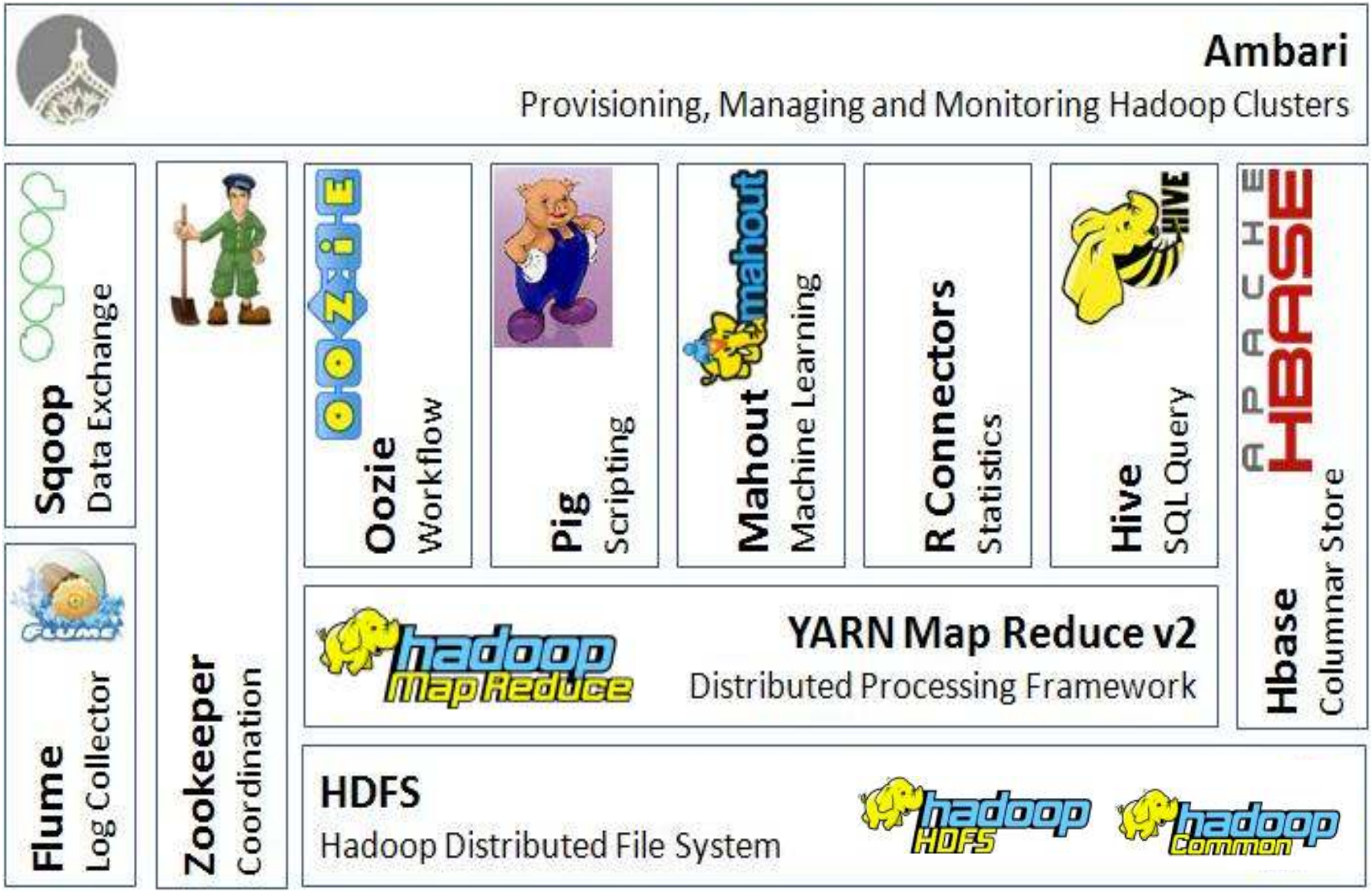


Apache Hadoop Core Technology



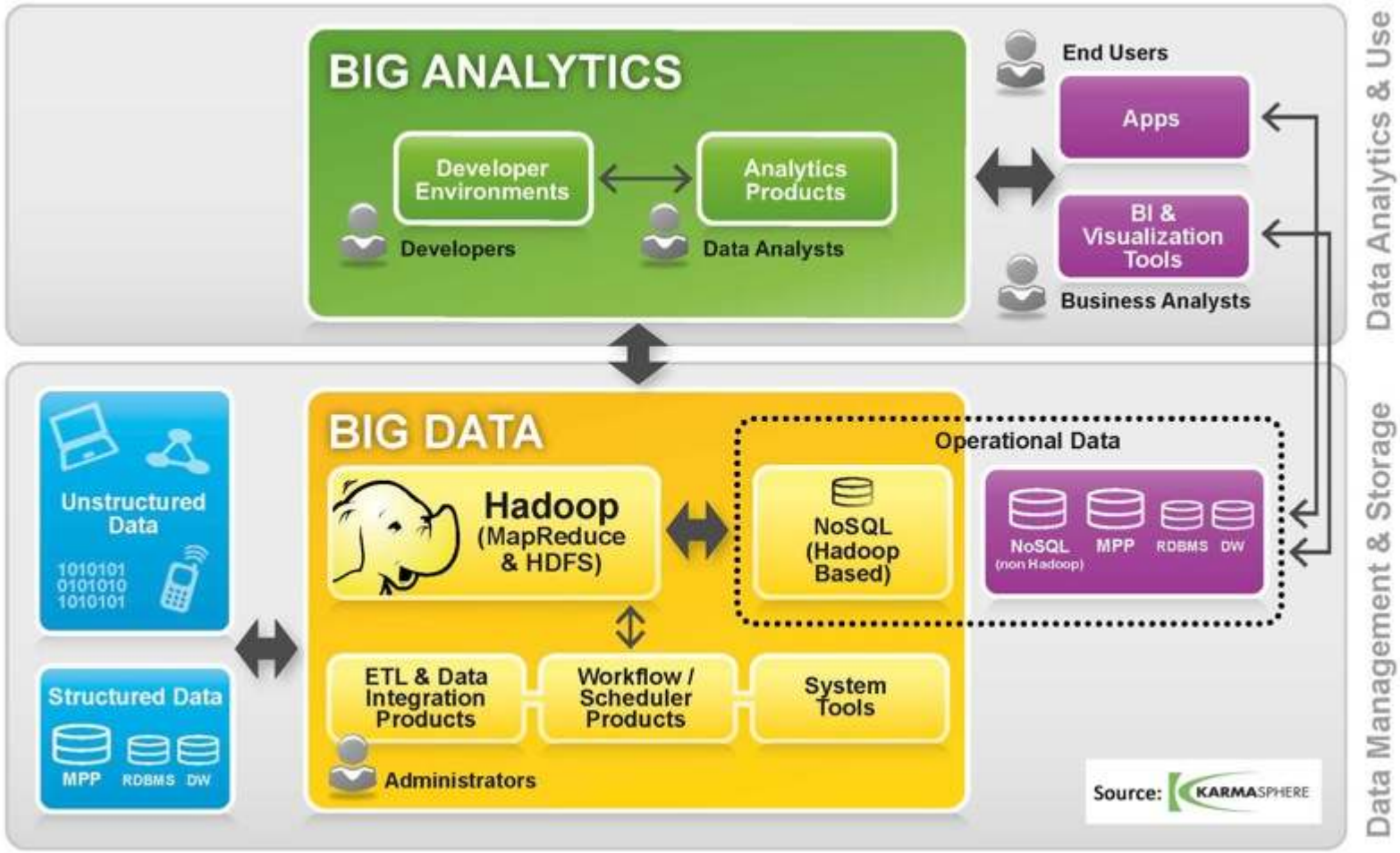
j2eedev.org/ecosystem-hadoop

Apache Hadoop Ecosystem

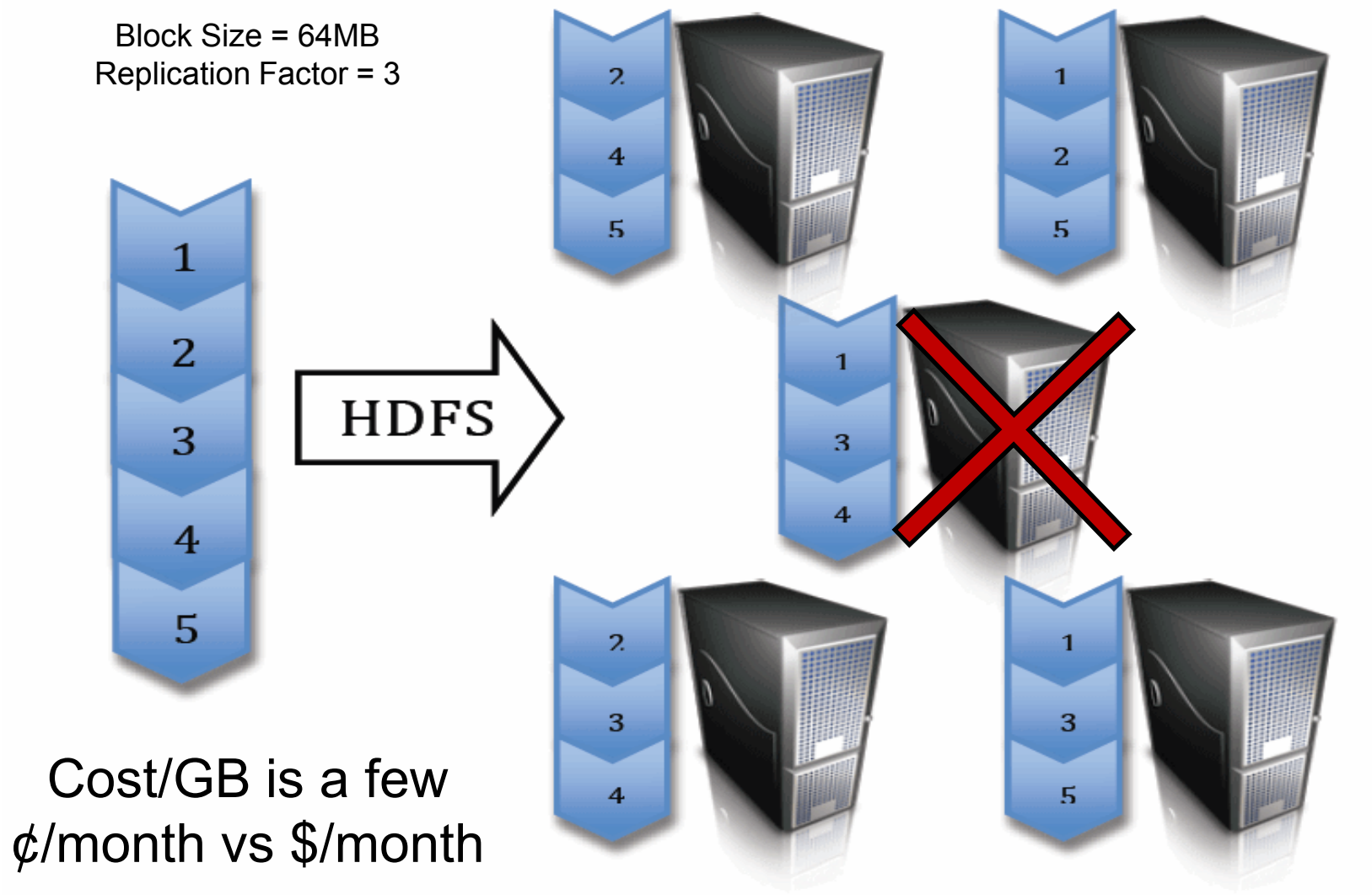


j2eedev.org/ecosystem-hadoop

Big Data Platform & Big Data Analytics
Hadoop Technology



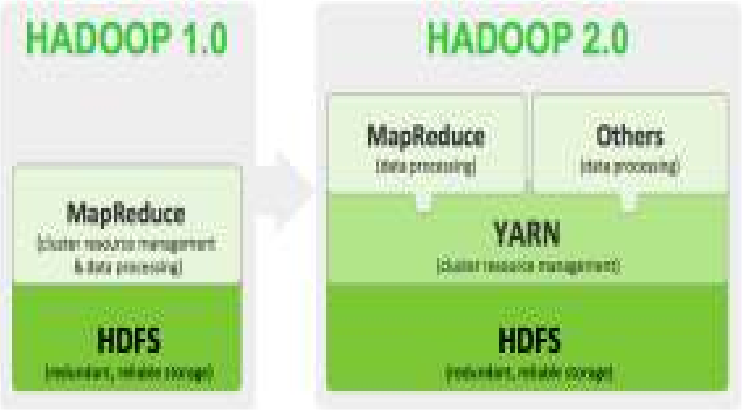
HDFS: Hadoop Distributed File System



Cost/GB is a few ¢/month vs $/month

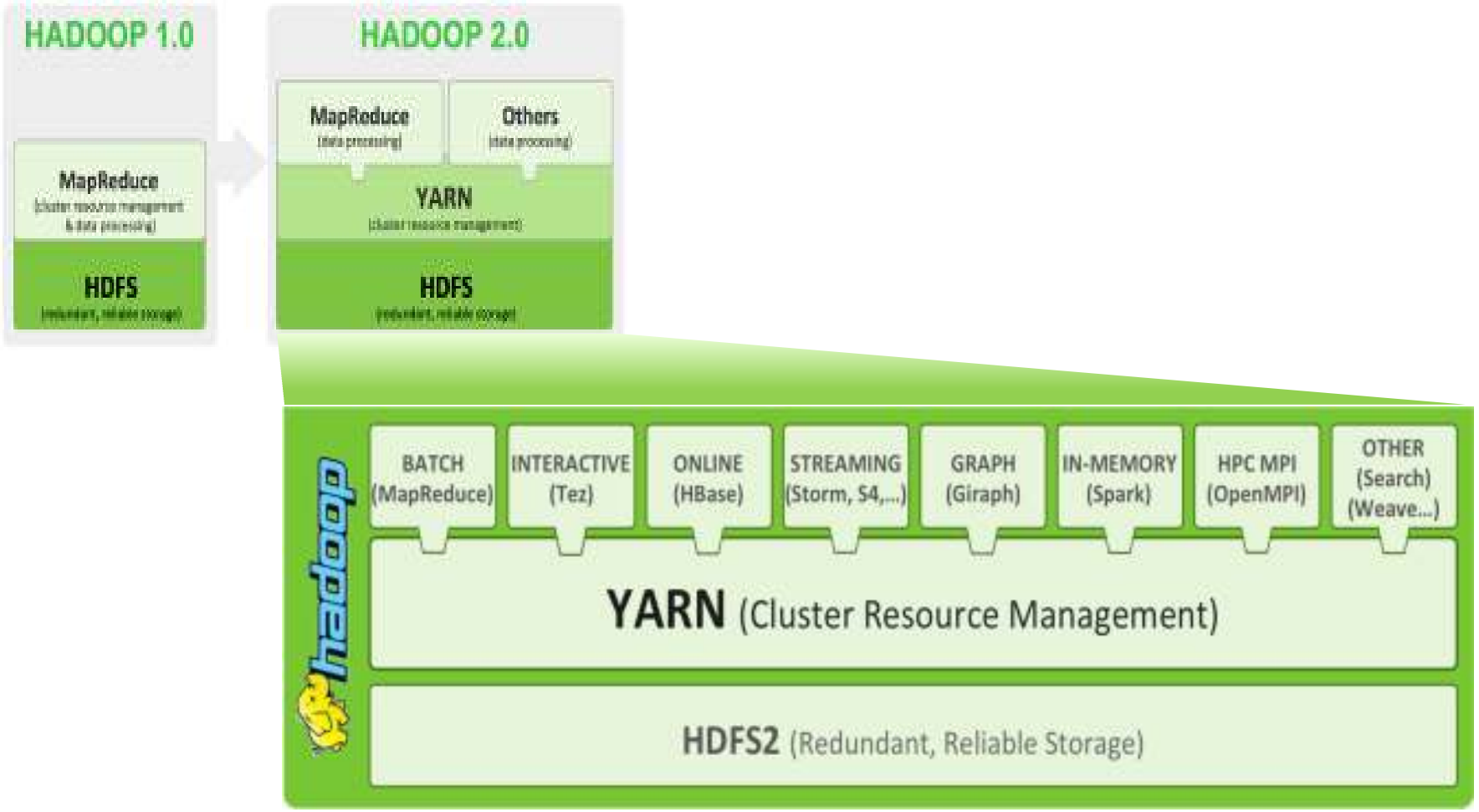
apache.org/hadoop/

Hadoop 1.0 vs Hadoop 2.0



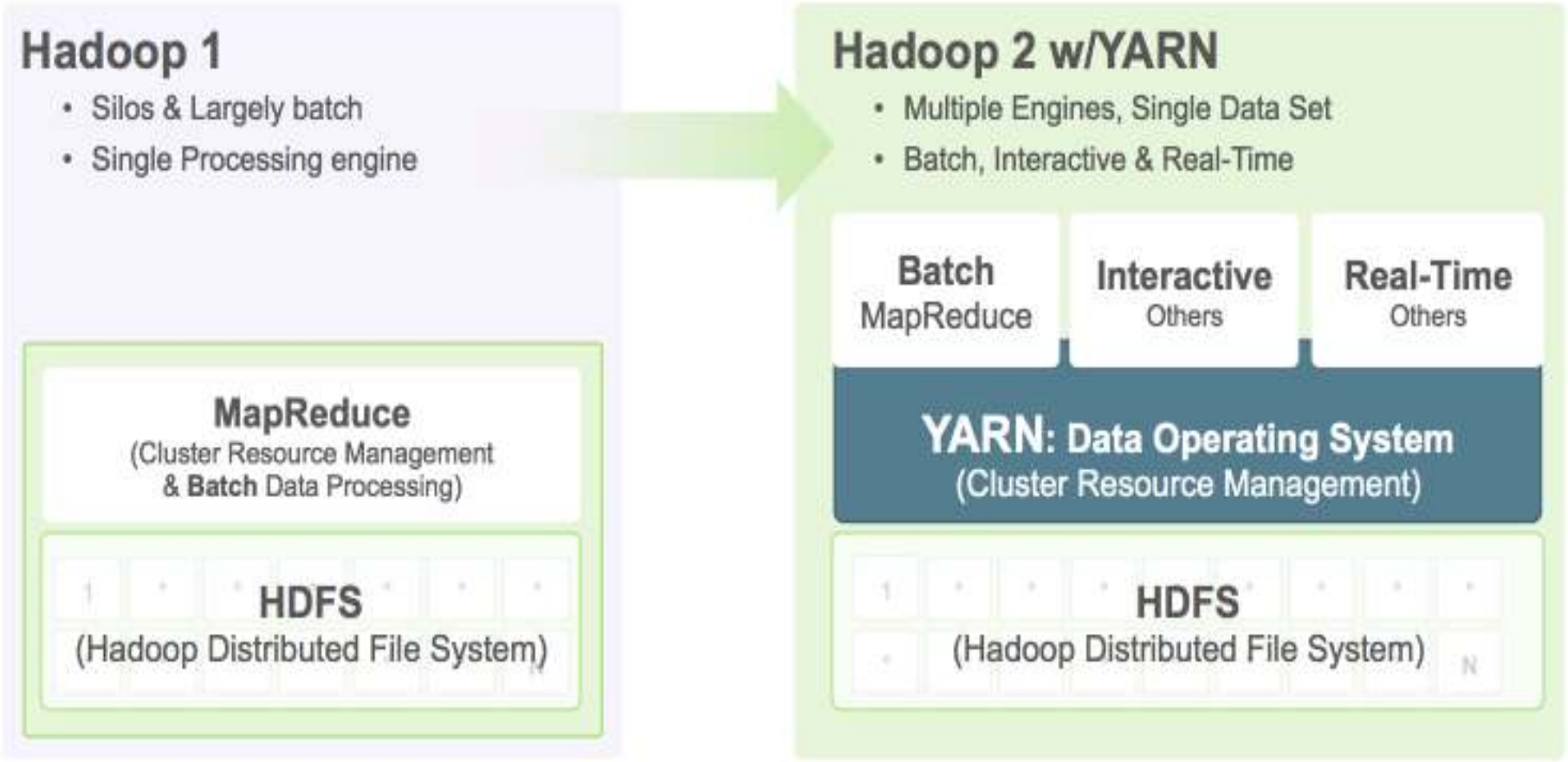
Hortonwork.com

Hadoop 1.0 vs Hadoop 2.0



Hortonwork.com

Hadoop 2



Hortonworks.com

Hadoop Symbols and Reasons Behind

Hadoop Components and Sub projects	Comparing with	Reason for the specific animal	Image
1.Hadoop distributed file system (HDFS)	Elephant	Memory of an elephant is compared with huge data storage of HDFS	
2.MapReduce	Mammoth	Mammoth means enormous, huge, massive and immense. A Mammoth's task is compared with a programming model for performing the tasks with huge volumes of data	
3.Hive	Honey Bee	Storage area of the honey in wax honeycombs inside the beehive is compared with data warehouse for storing the data in the format of table.	
4.Hbase	Horse	Running Speed of the horse indicates the real time read/write access from HBase	
5.Pig	Pig	Pigs are omnivores animals which means they can consume both plants and animals. This PIG consumes any type of data whether structured or unstructured or any other machine data and helps processing the same.	
6.Hue	Elephant foot prints	Elephant foot print is compared with "TREAD ON" Hadoop and explore it.	
7.Beeswax	Beeswax	Data storage happens in a structured way as honey stored in Beeswax	

Opportunity and Market Outlook

SECTOR	EXAMPLE APPLICATIONS	MAJOR DRIVER
Smart buildings	Automated monitoring of heating, ventilation and cooling	Reduced energy costs
Smart cities	Street lights that dim when roads are empty	Cost savings
Automotive	Emergency calling and accident alerts	
Leisure	Leisure vehicle and boat tracking	Safety and security
Consumer electronics	Connected satellite navigation devices to monitor traffic jams	Production innovation
Health	Remote monitoring of patients and personal health monitoring	Cheaper, home-based care
Utilities	Smart meters and energy demand response	Regulatory requirement
Transportation and logistics	Fleet optimization and supply-chain tracking and tracing	Cost savings
Retail	Wireless payments	Retail innovation
Manufacturing	Predictive maintenance through improved system monitoring	Reduced maintenance costs
Construction	Monitoring usage of equipment to improve efficiency and cut fuel usage	Cost savings
Agriculture and extraction	Remote monitoring of farm or mining operations and equipment	Proactive maintenance
Emergency services and national security	Disaster response and critical infrastructure protection	Faster response times

Source: Machina Research

Hands-On: Launch a virtual server on EC2 Amazon Web Services


AWS
Services
Edit
IMC Institute
Oregon
Support
Amazon Web Services
Compute
EC2
Virtual Servers in the Cloud
Lambda PREVIEW
Run Code in Response to Events
Storage & Content Delivery
S3
Scalable Storage in the Cloud
Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage
Glacier
Archive Storage in the Cloud
CloudFront
Global Content Delivery Network
Database
RDS
MySQL, Postgres, Oracle, SQL Server, and Amazon Aurora
DynamoDB
Predictable and Scalable NoSQL Data Store
ElastiCache
In-Memory Cache
Redshift
Managed Petabyte-Scale Data Warehouse Service
Administration & Security
Directory Service
Managed Directories in the Cloud
Identity & Access Management
Access Control and Key Management
Trusted Advisor
AWS Cloud Optimization Expert
CloudTrail
User Activity and Change Tracking
Config
Resource Configurations and Inventory
CloudWatch
Resource and Application Monitoring
Deployment & Management
Elastic Beanstalk
OpsWorks
DevOps Application Management Service
CloudFormation
Templated AWS Resource Creation
CodeDeploy
Automated Deployments
Analytics
EMR
Managed Hadoop Framework
Application Services
SQS
Message Queue Service
SWF
Workflow Service for Coordinating Application Components
AppStream
Low Latency Application Streaming
Elastic Transcoder
Easy-to-use Scalable Media Transcoding
SES
Email Sending Service
CloudSearch
Managed Search Service
Mobile Services
Cognito
User Identity and App Data Synchronization
Mobile Analytics
Understand App Usage Data at Scale
SNS
Push Notification Service
Enterprise Applications
WorkSpaces
Desktops in the Cloud
WorkDocs
Resource Groups
A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.
Create a Group
Tag Editor
Additional Resources
Getting Started
See our documentation to get started and learn more about how to use our services.
AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.
AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.
Service Health

Hadoop Installation

Hadoop provides three installation choices:

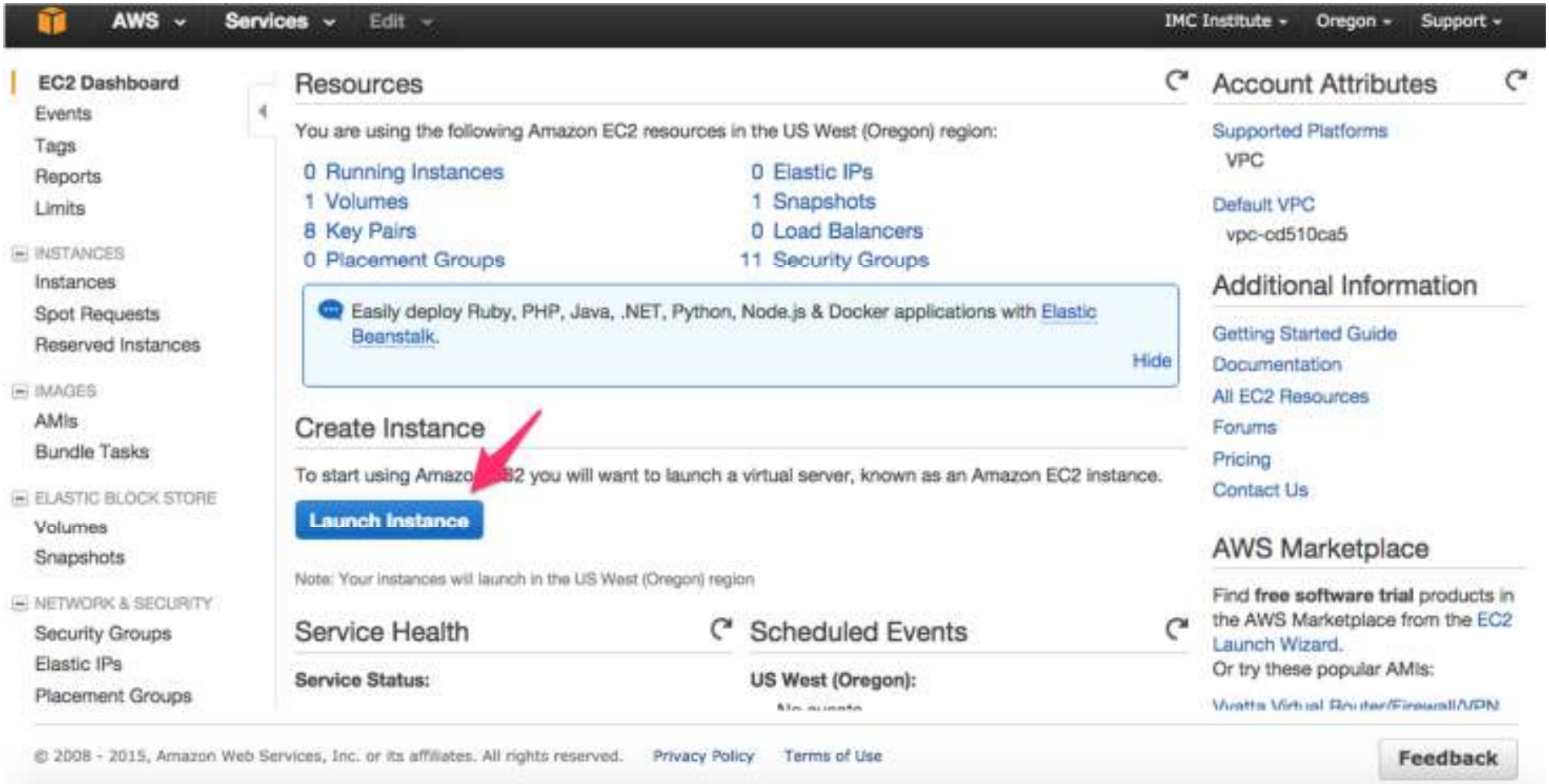
1. **Local mode:** This is an unzip and run mode to get you started right away where allparts of Hadoop run within the same JVM
2. **Pseudo distributed mode**: This mode will be run on different parts of Hadoop as different Java processors, but within a single machine
3. **Distributed mode:** This is the real setup that spans multiple machines

Virtual Server

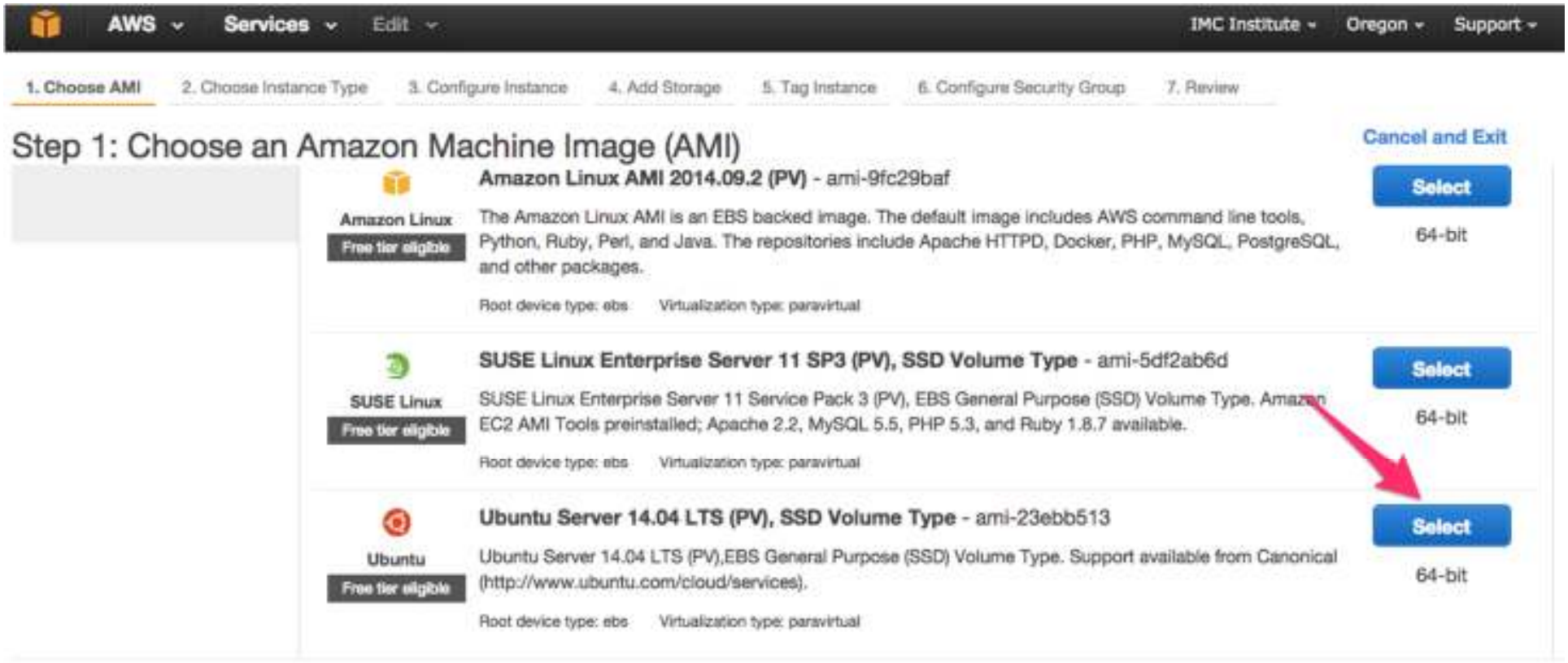
This lab will use a EC2 virtual server to install a Hadoop server using the following features:

- Ubuntu Server 14.04 LTS
- m3.mediun 1vCPU, 3.75 GB memory
- Security group: default
- Keypair: imchadoop

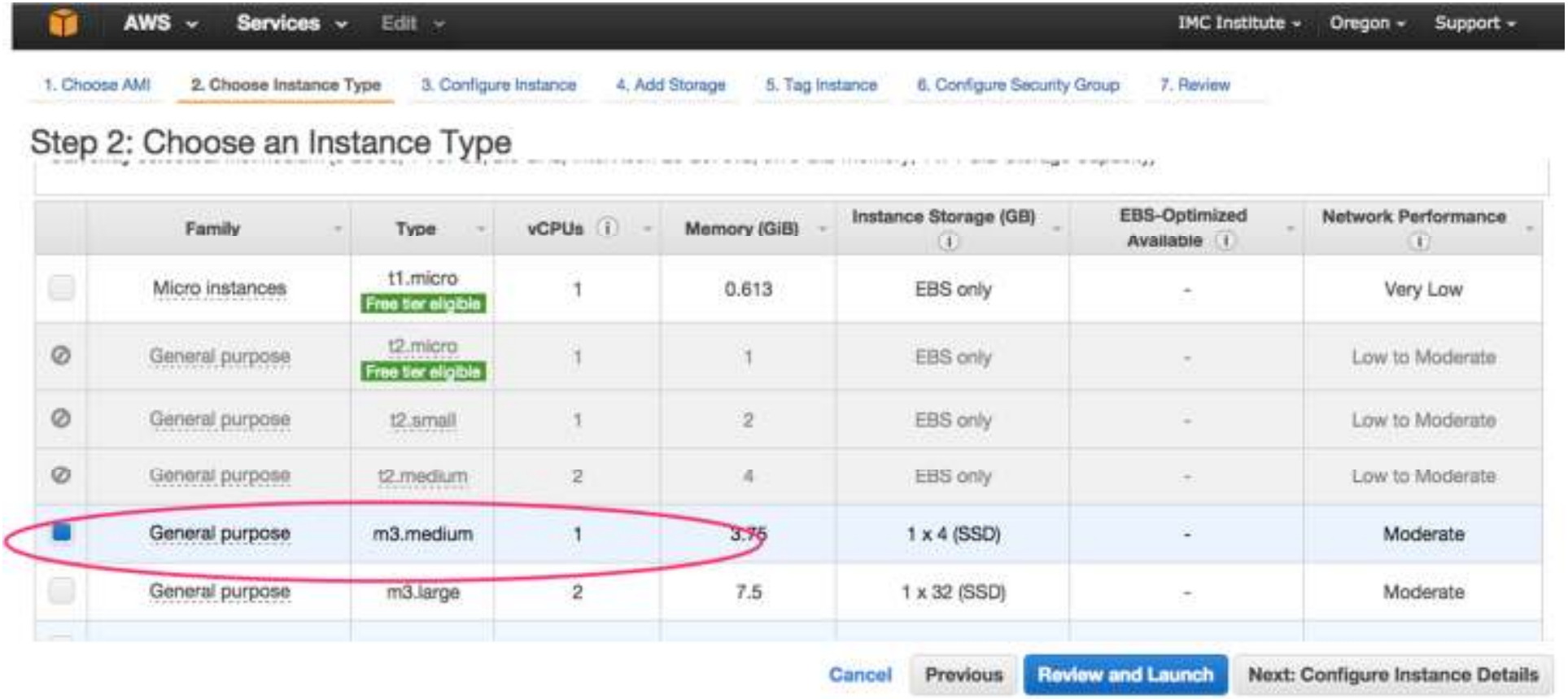
Select a EC2 service and click on Lunch Instance



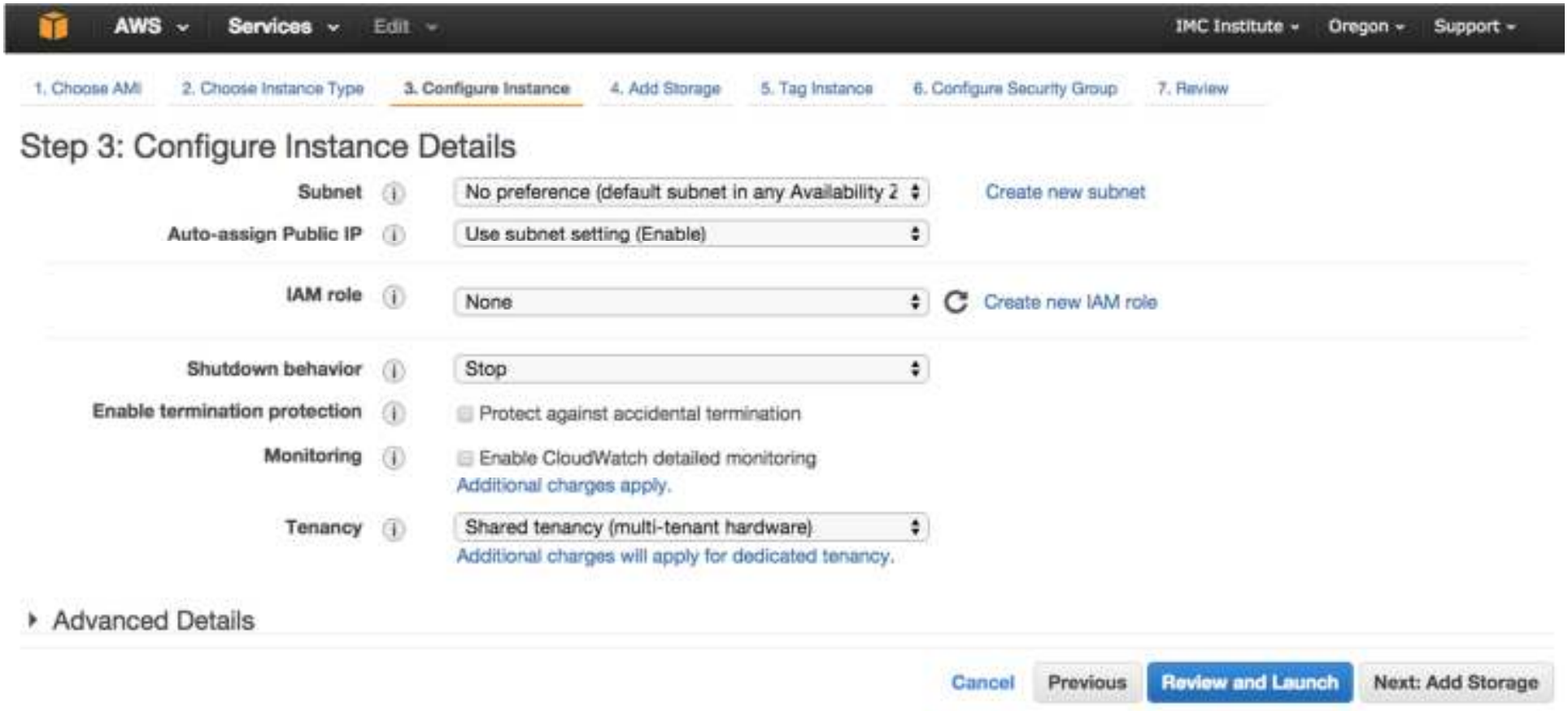
Select an Amazon Machine Image (AMI) and Ubuntu Server 14.04 LTS (PV)



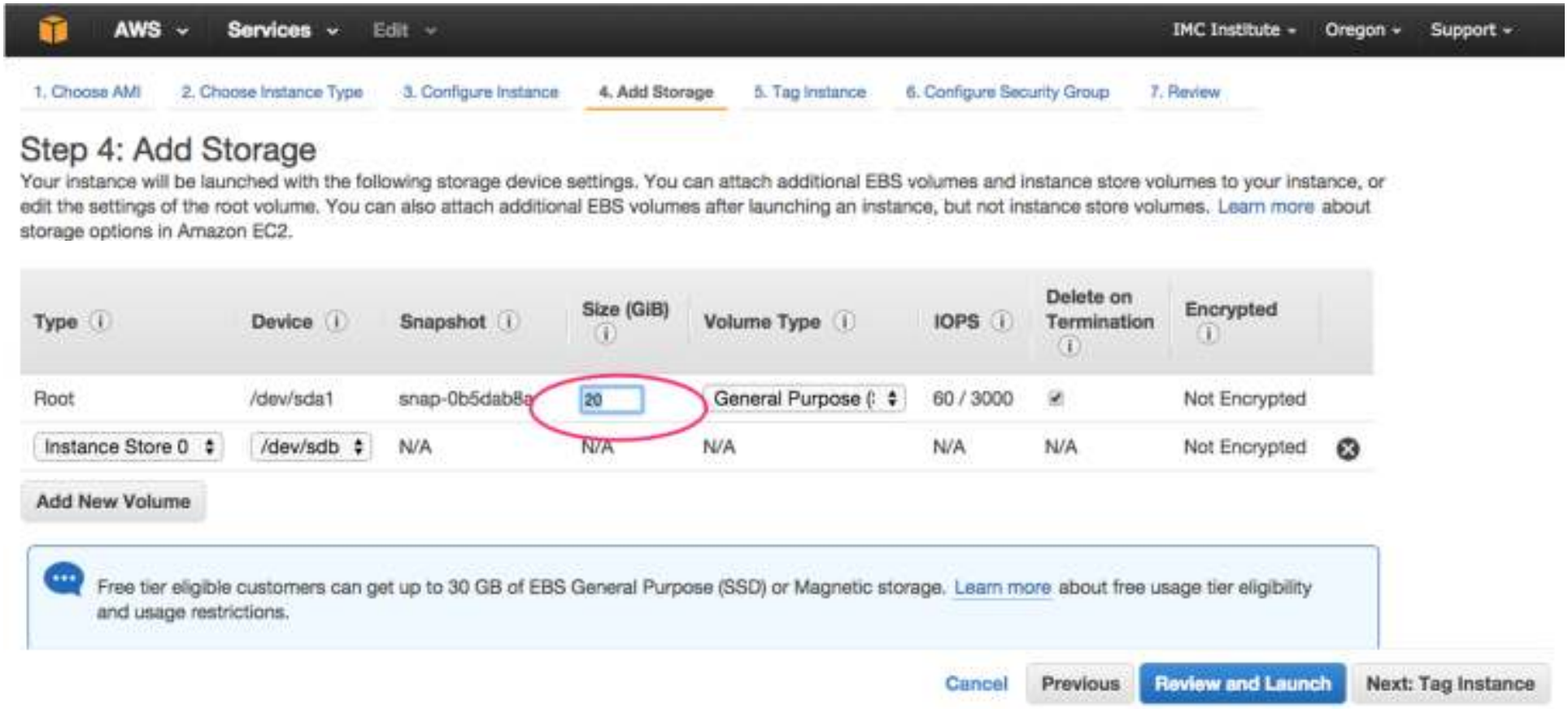
Choose m3.medium Type virtual server



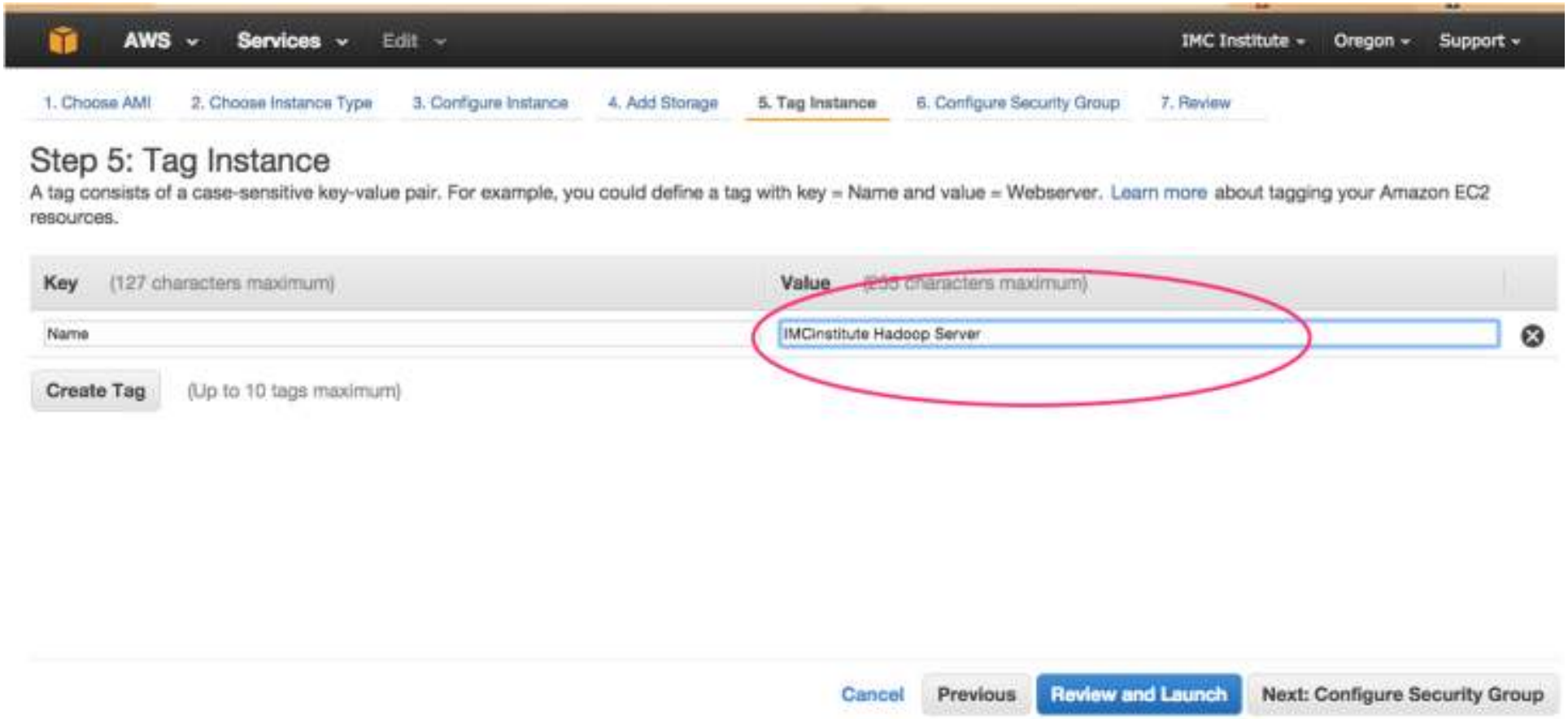
Leave configuration details as default



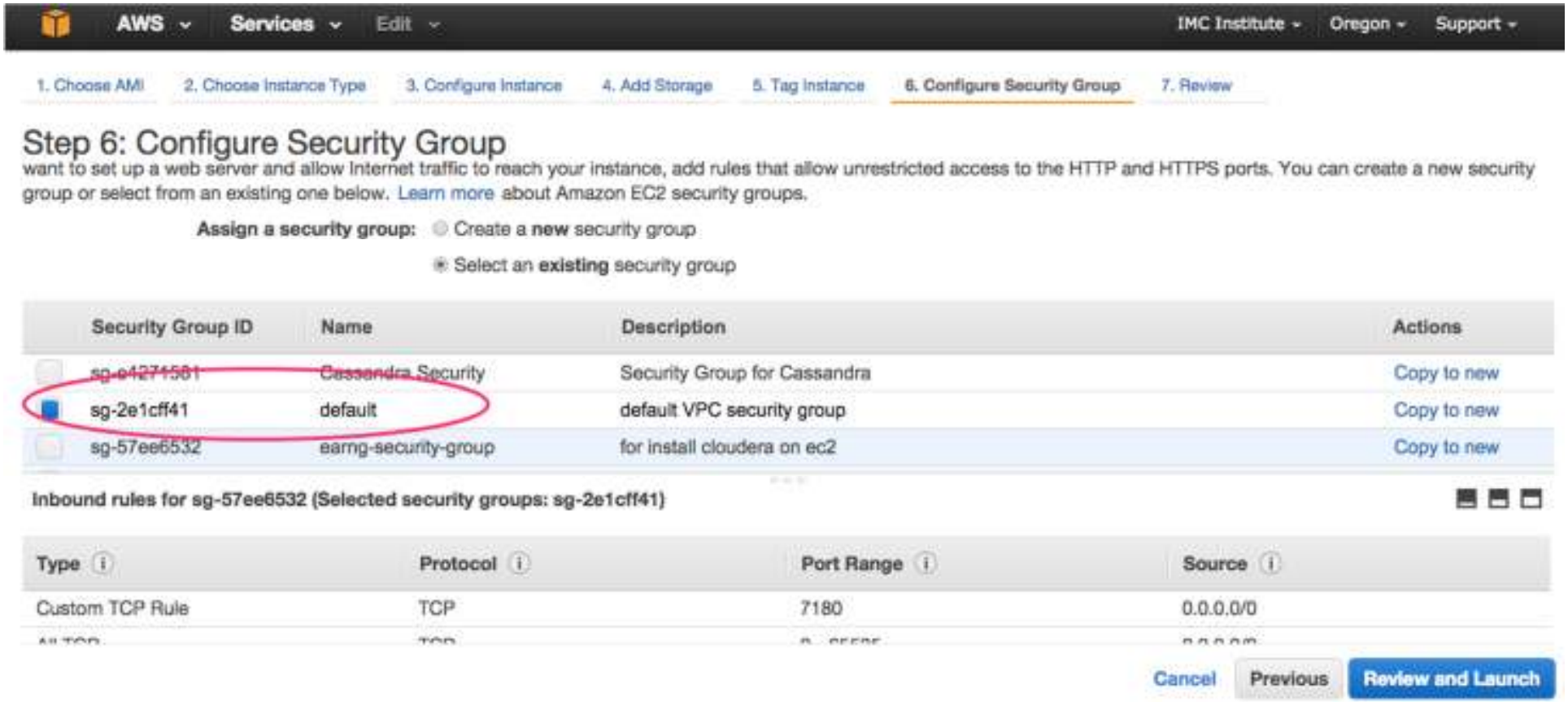
Add Storage: 20 GB



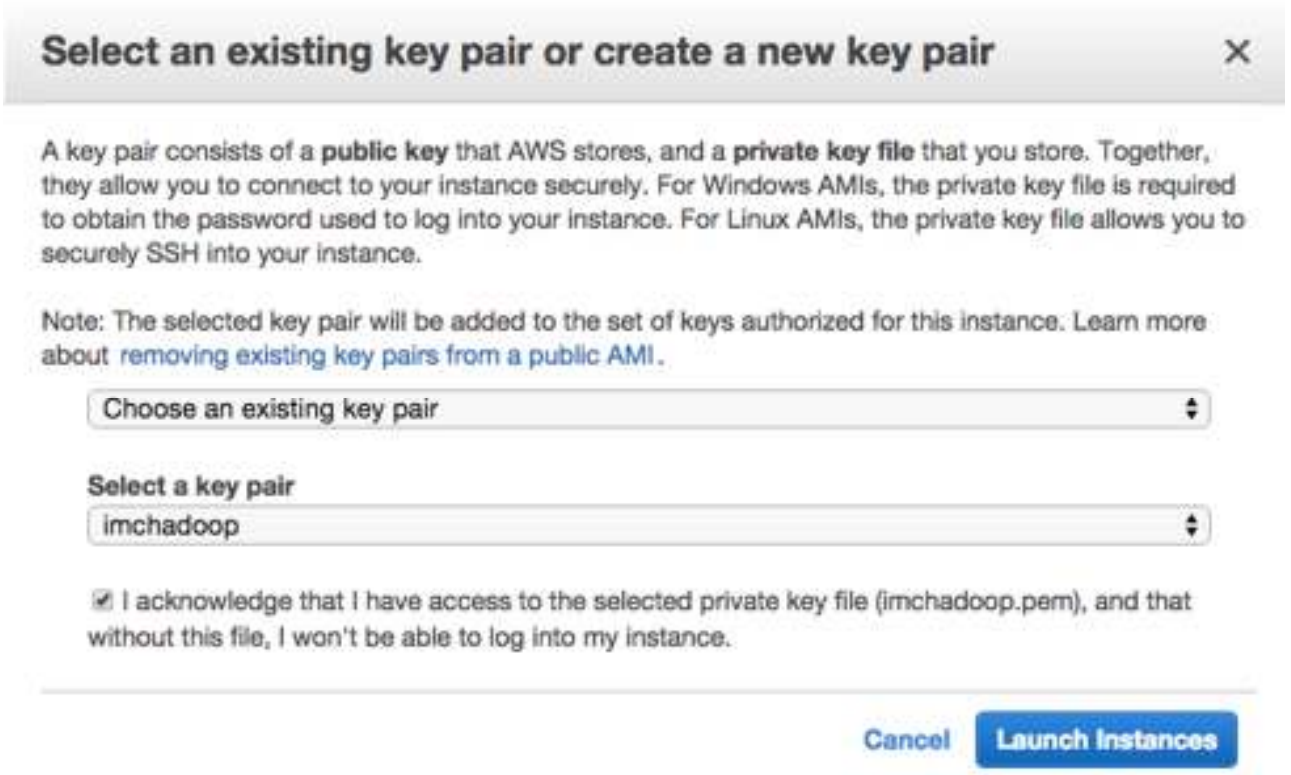
Name the instance



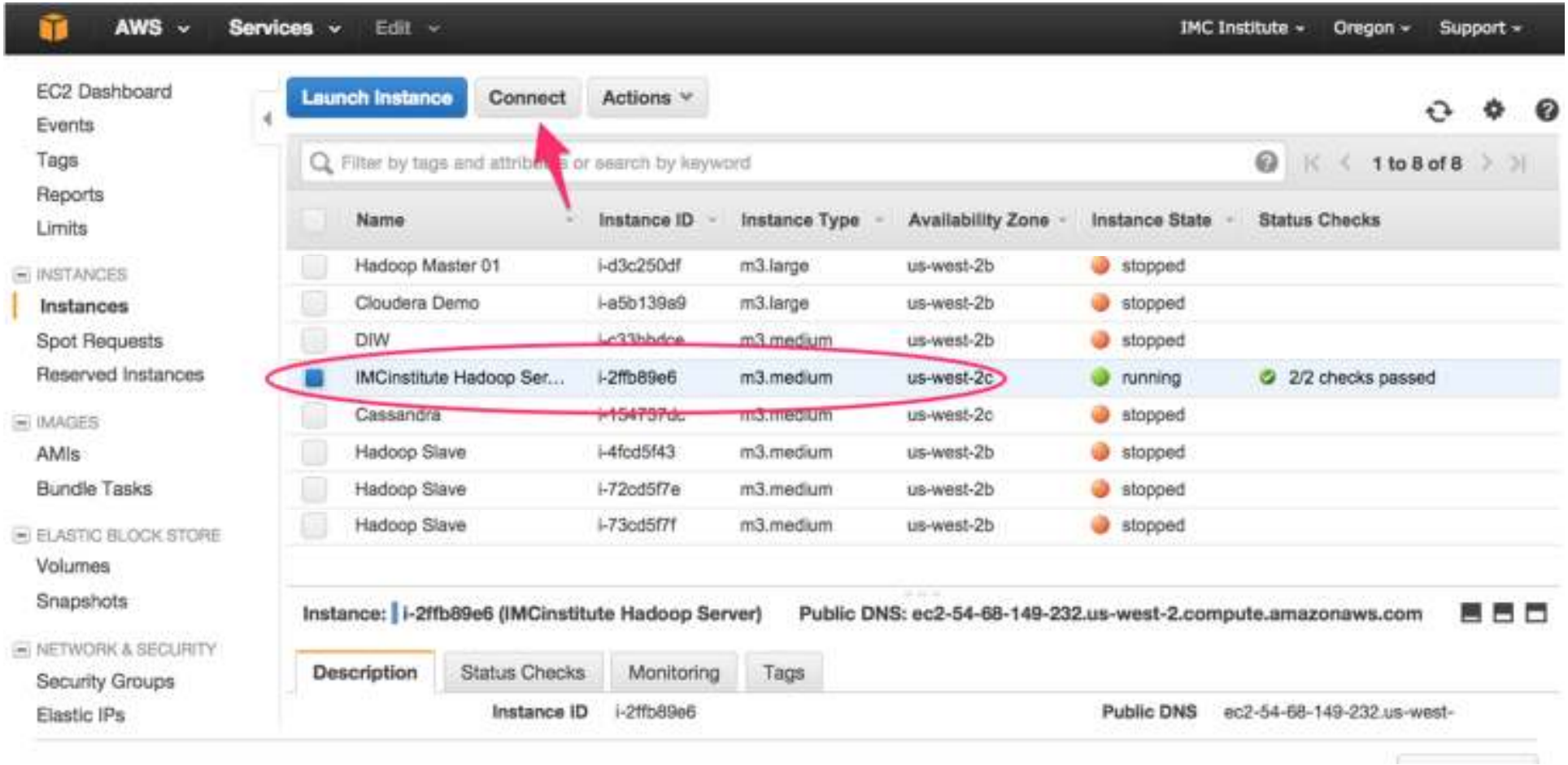
Select an existing security group > Select Security Group Name: default



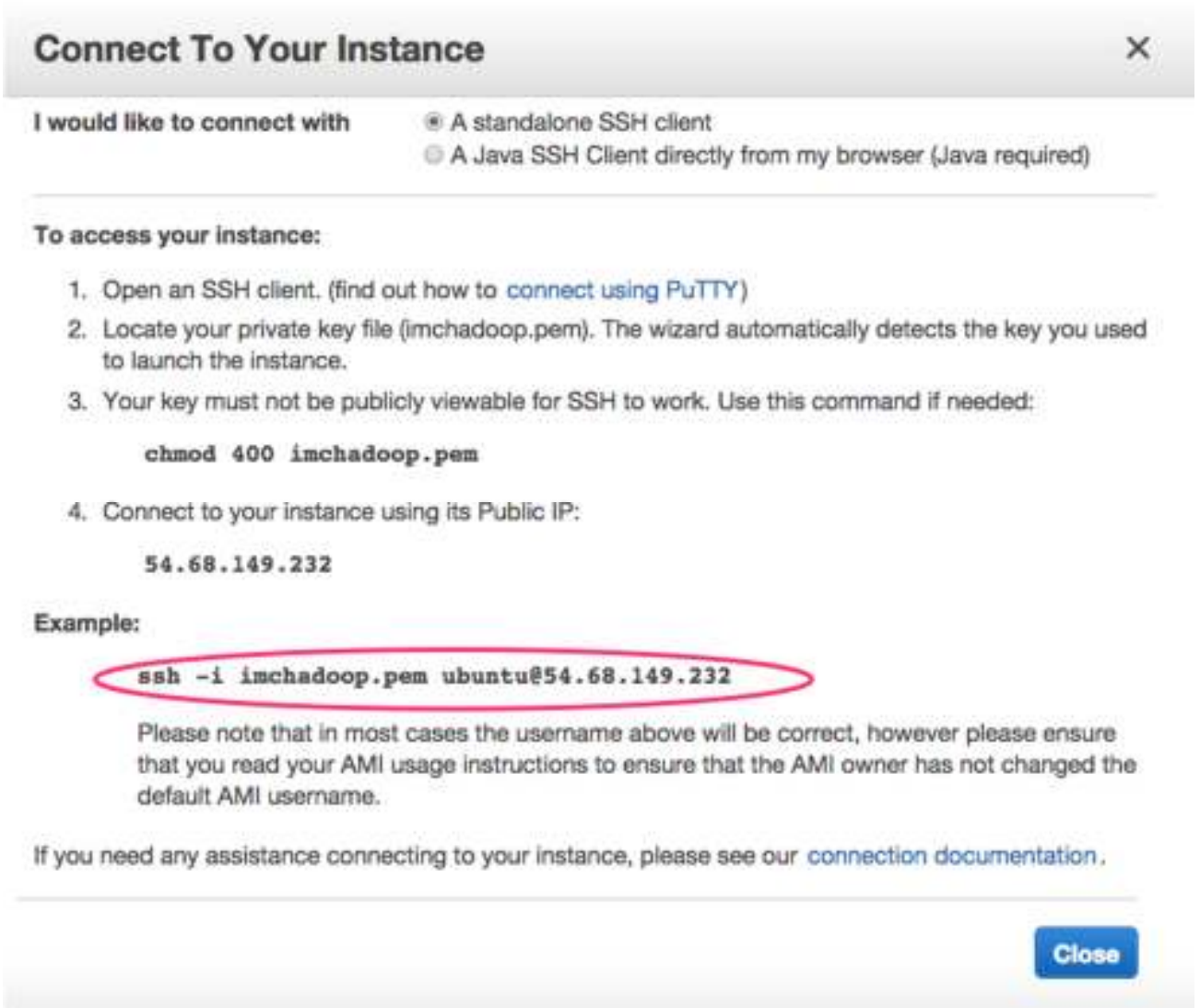
Click Launch and choose imchadoop as a key pair



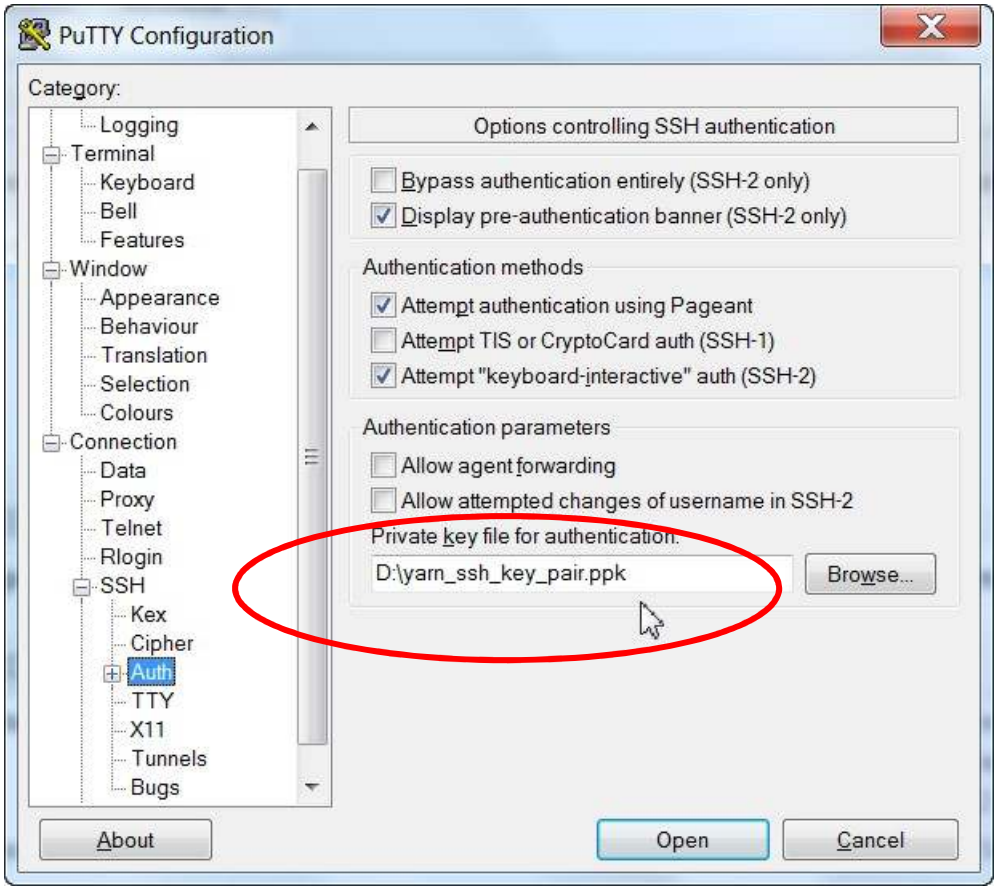
Review an instance / click Connect for an instruction to connect to the instance



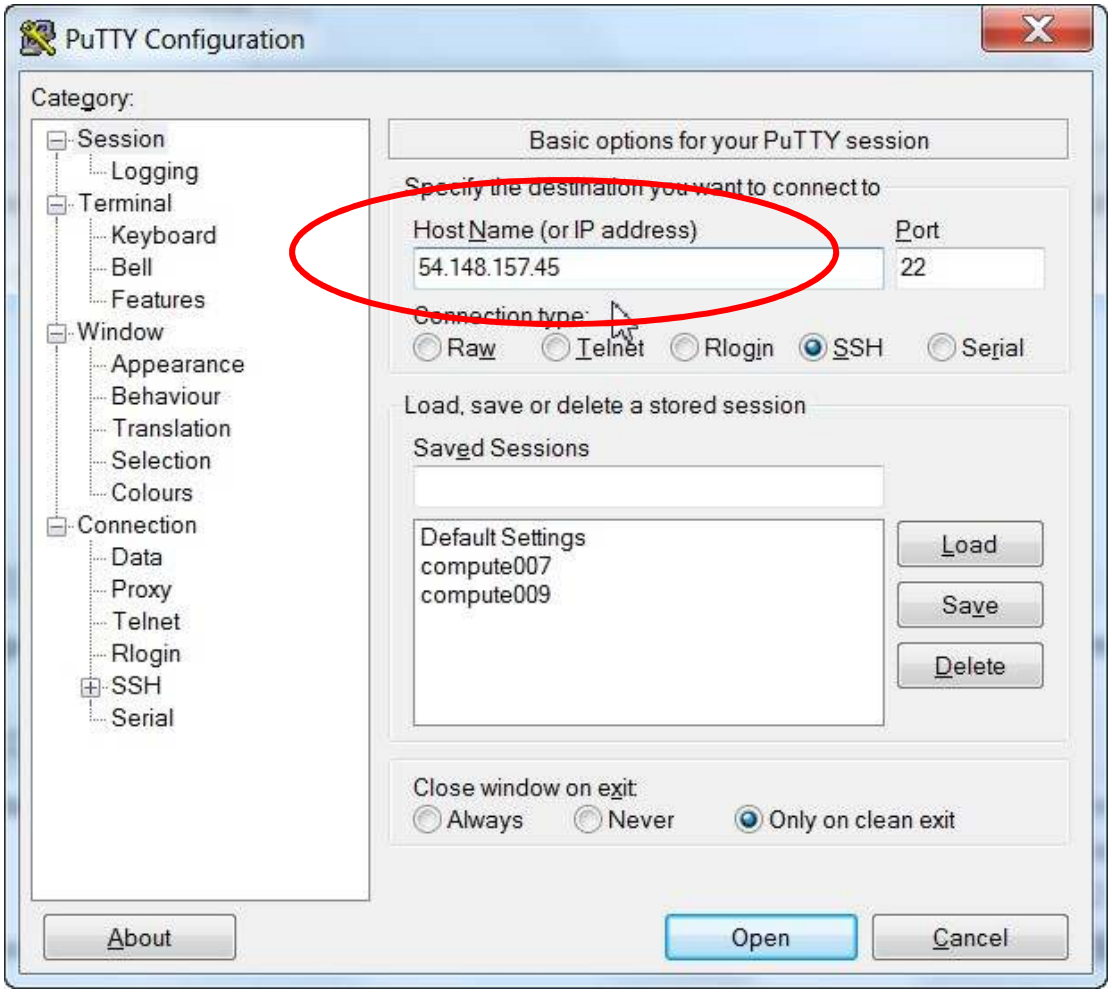
Connect to an instance from Mac/Linux



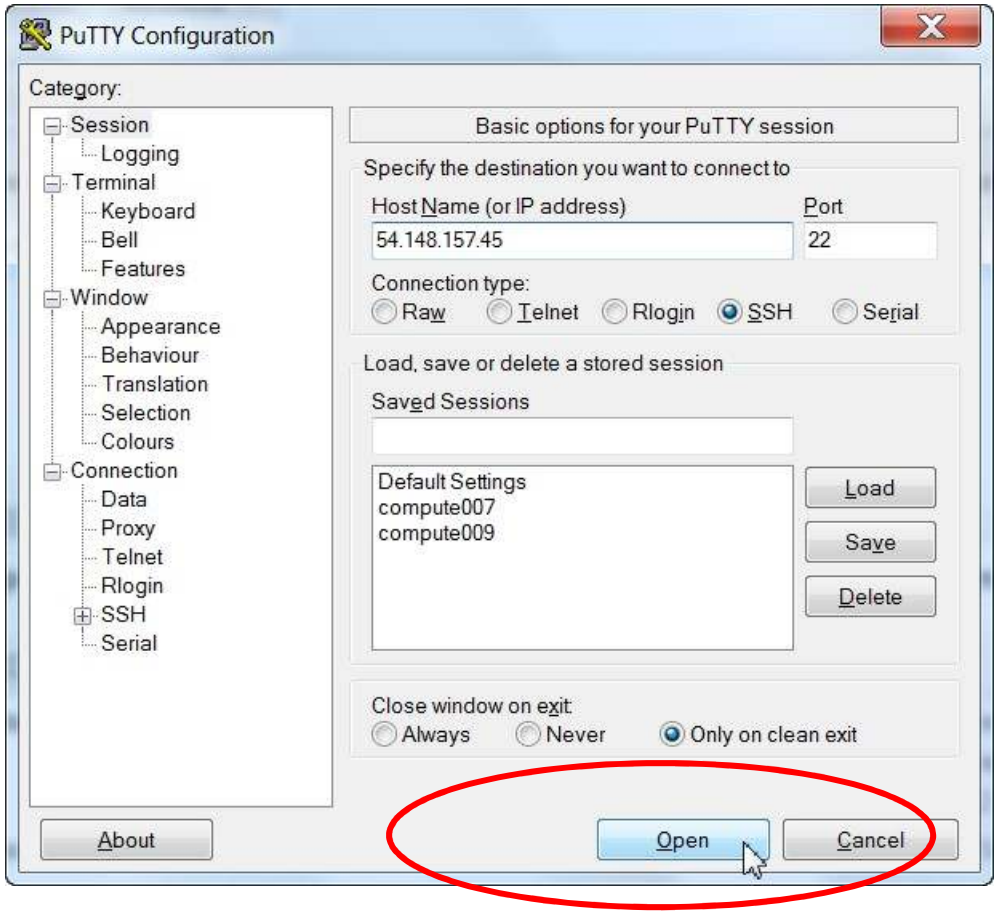
Connect to an instance from Windows using Putty



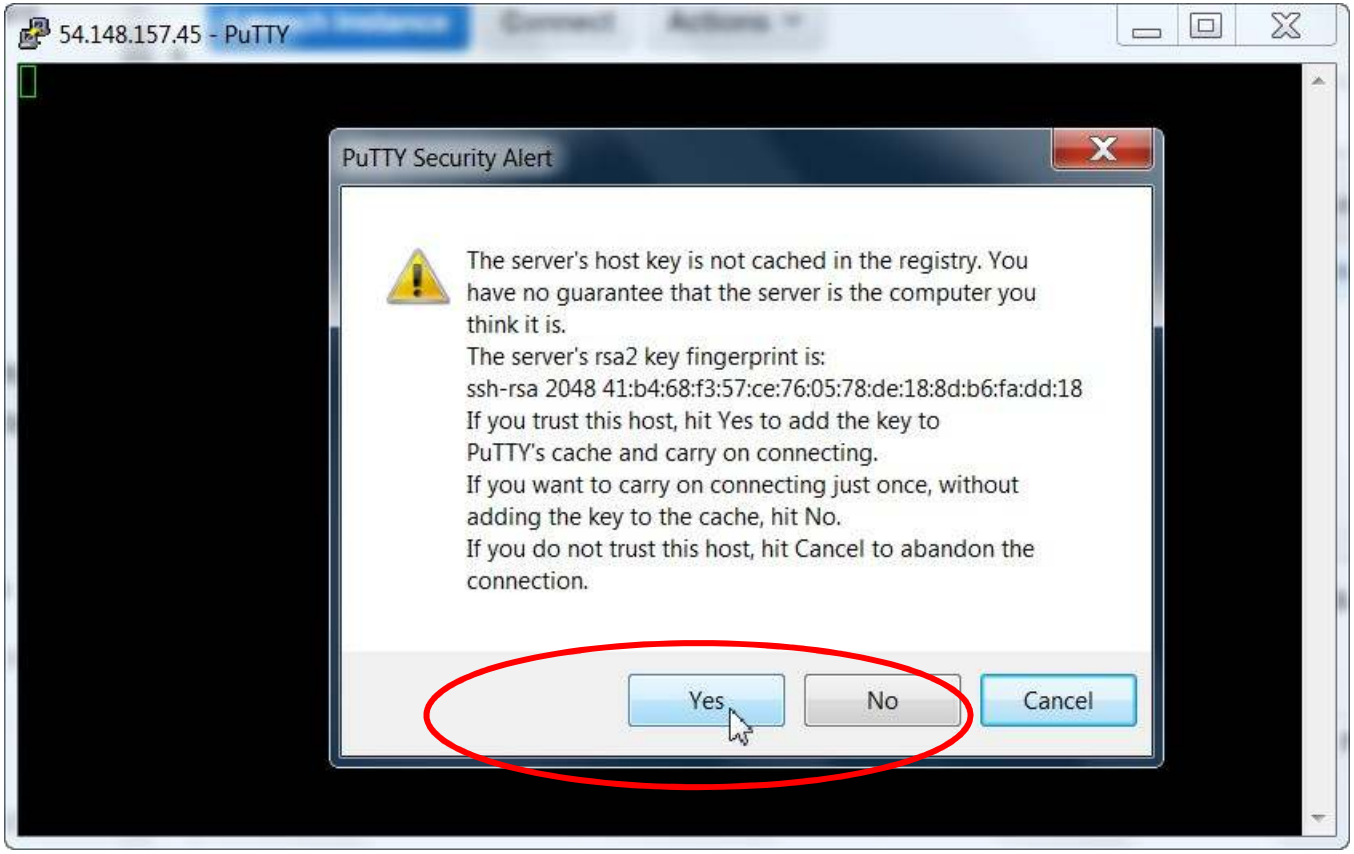
Connect to an instance from Windows using Putty



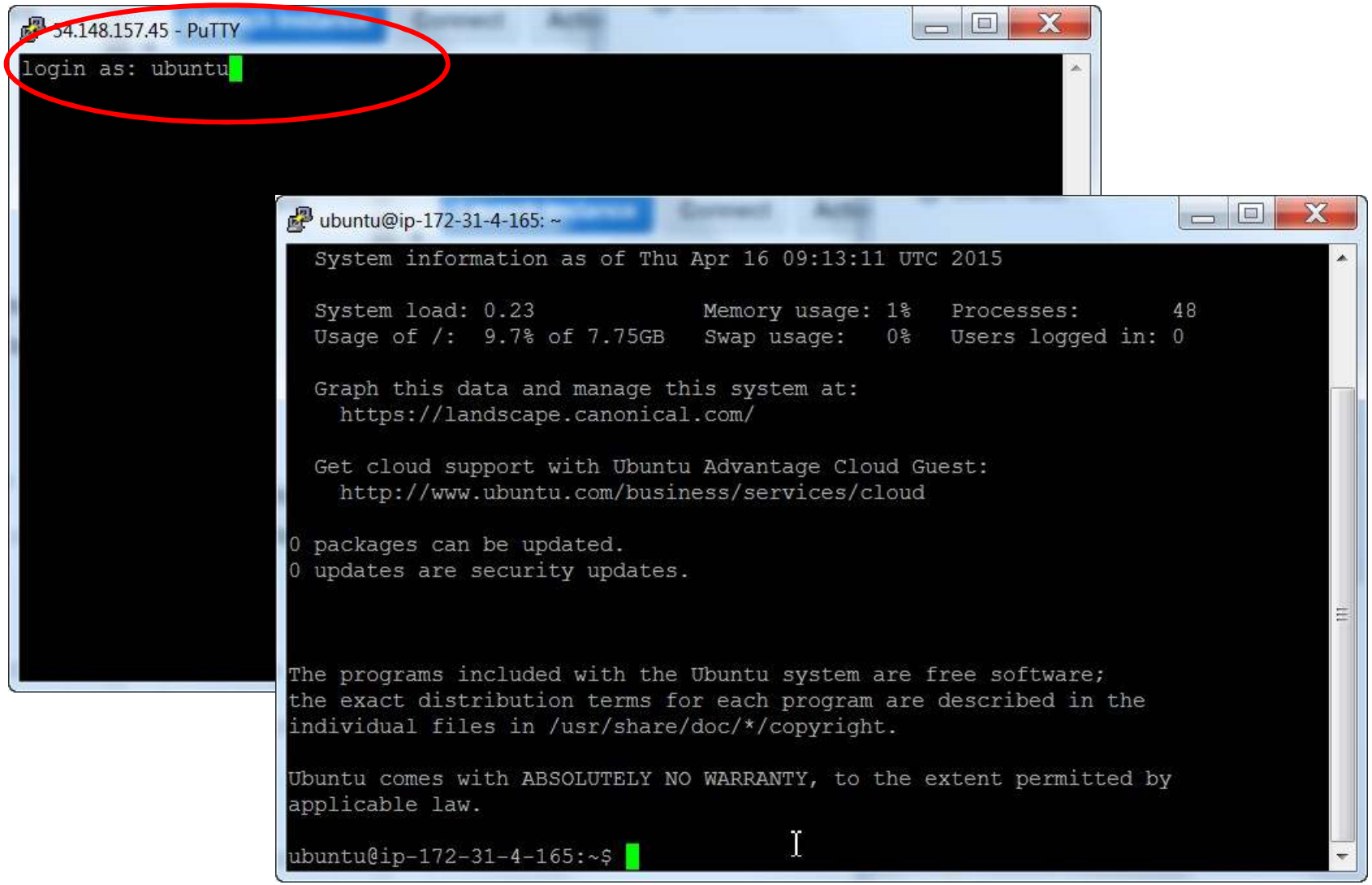
Connect to an instance from Windows using Putty



Connect to an instance from Windows using Putty



Login as: ubuntu



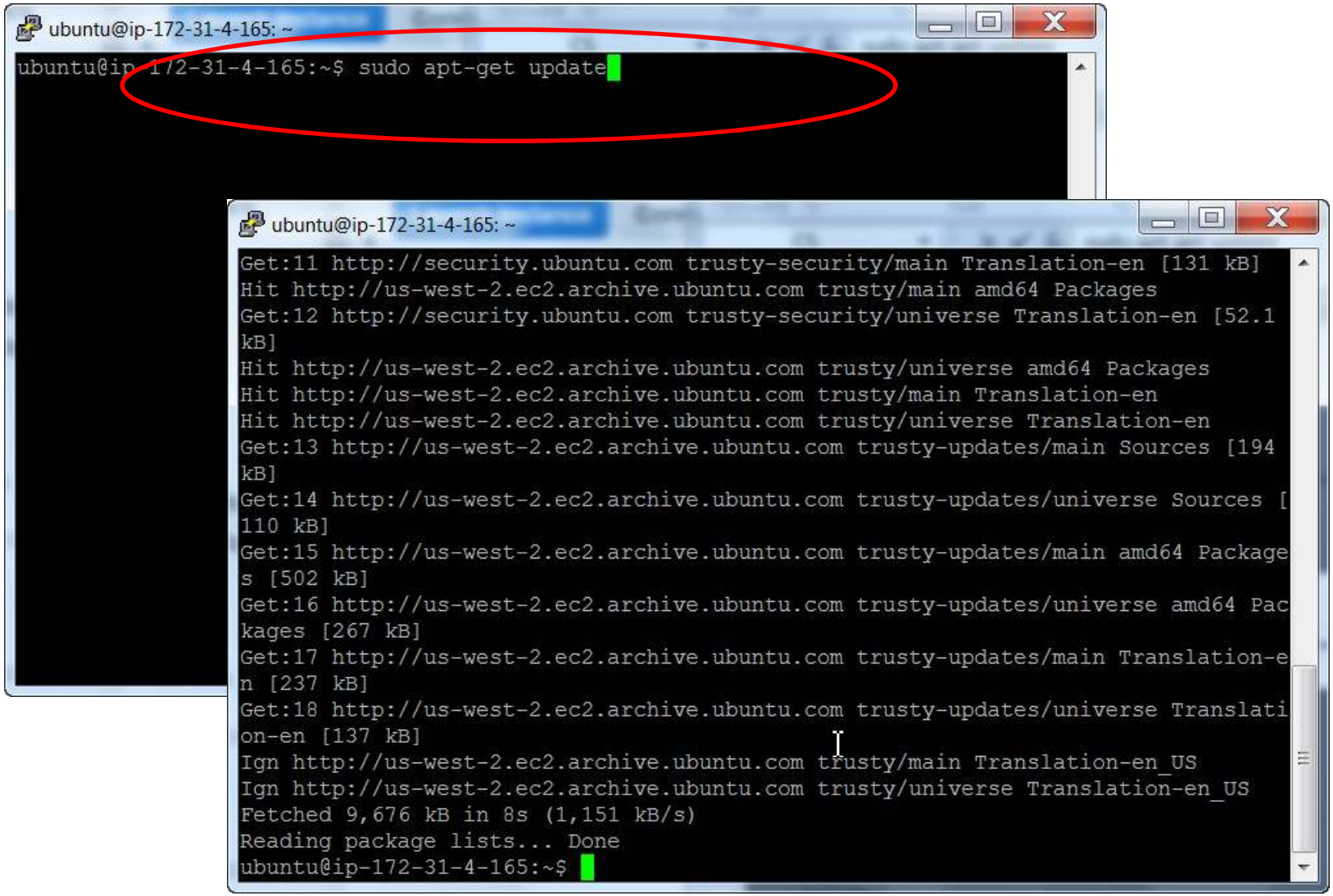
Hands-On: Installing Hadoop

Installing Hadoop and Ecosystem

1. Update System Software Repository
2. Configuring SSH
3. Installing Java
4. Download/Extract Hadoop
5. Installing Hadoop
6. Configure Hadoop
7. Formatting Namenode
8. Starting Hadoop
9. Accessing Hadoop Web Console
10. Stopping Hadoop

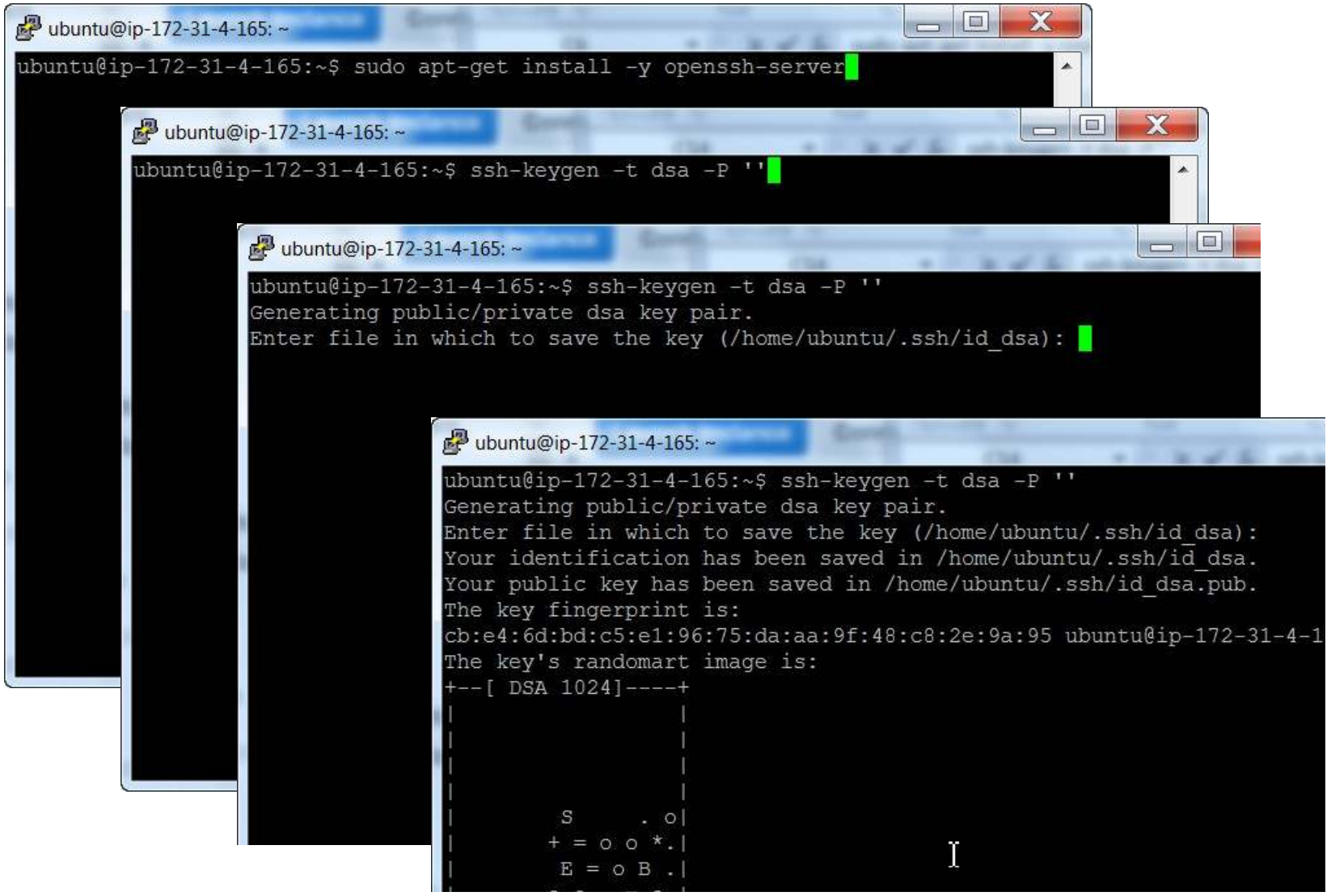
1. Update System Software Repository

sudo apt-get update

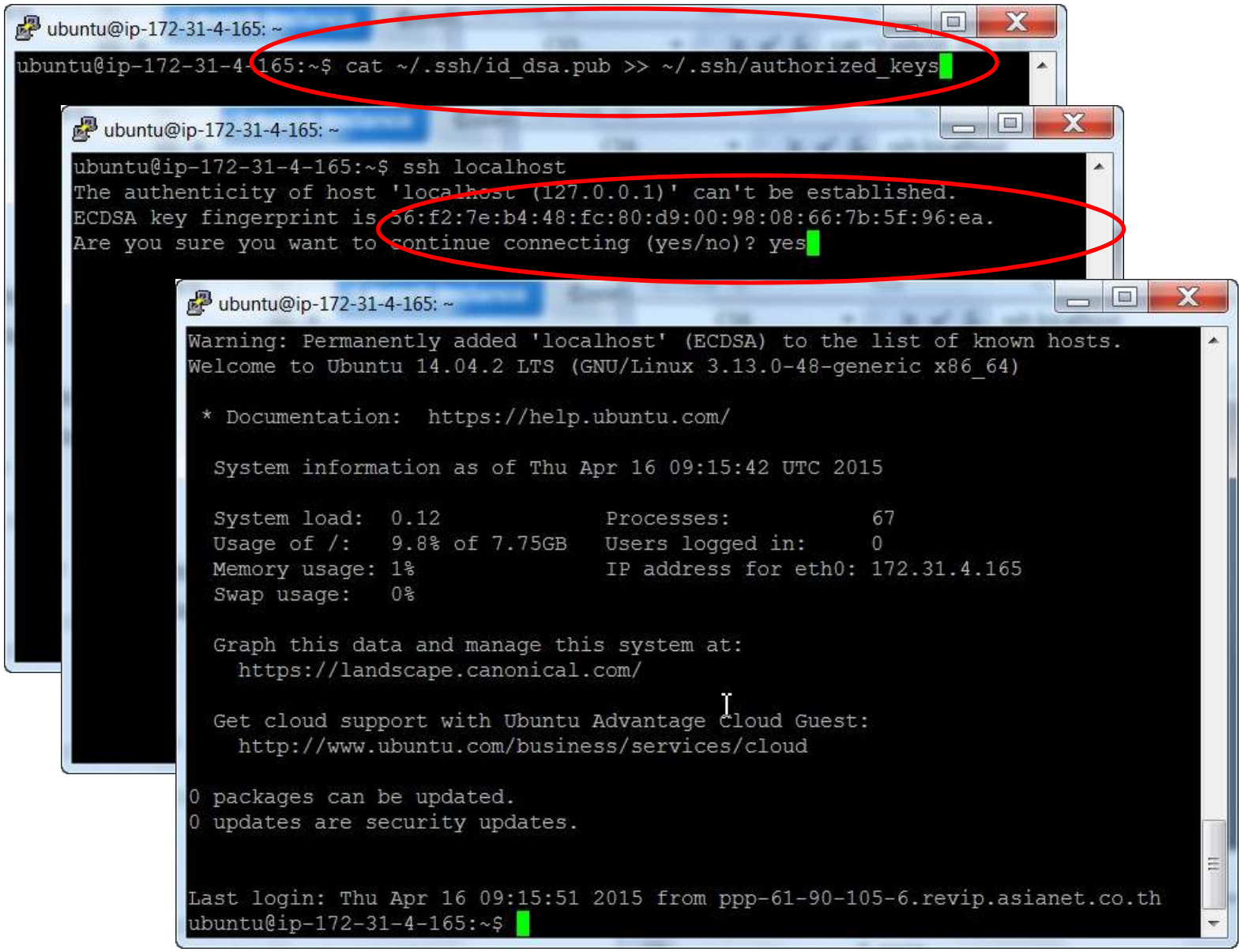


2. Configuring SSH

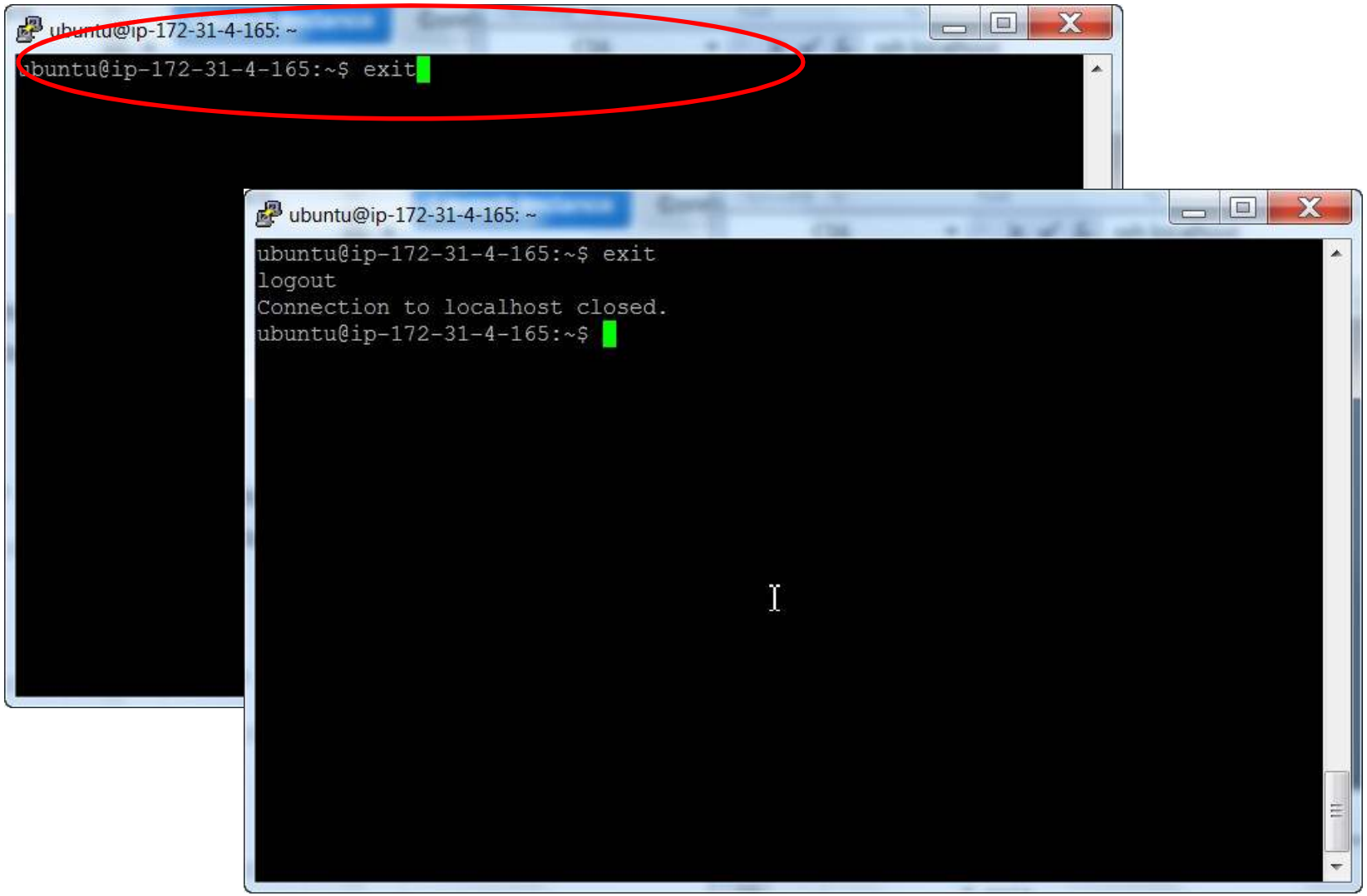
Install SSH, Create ssh-key



cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys

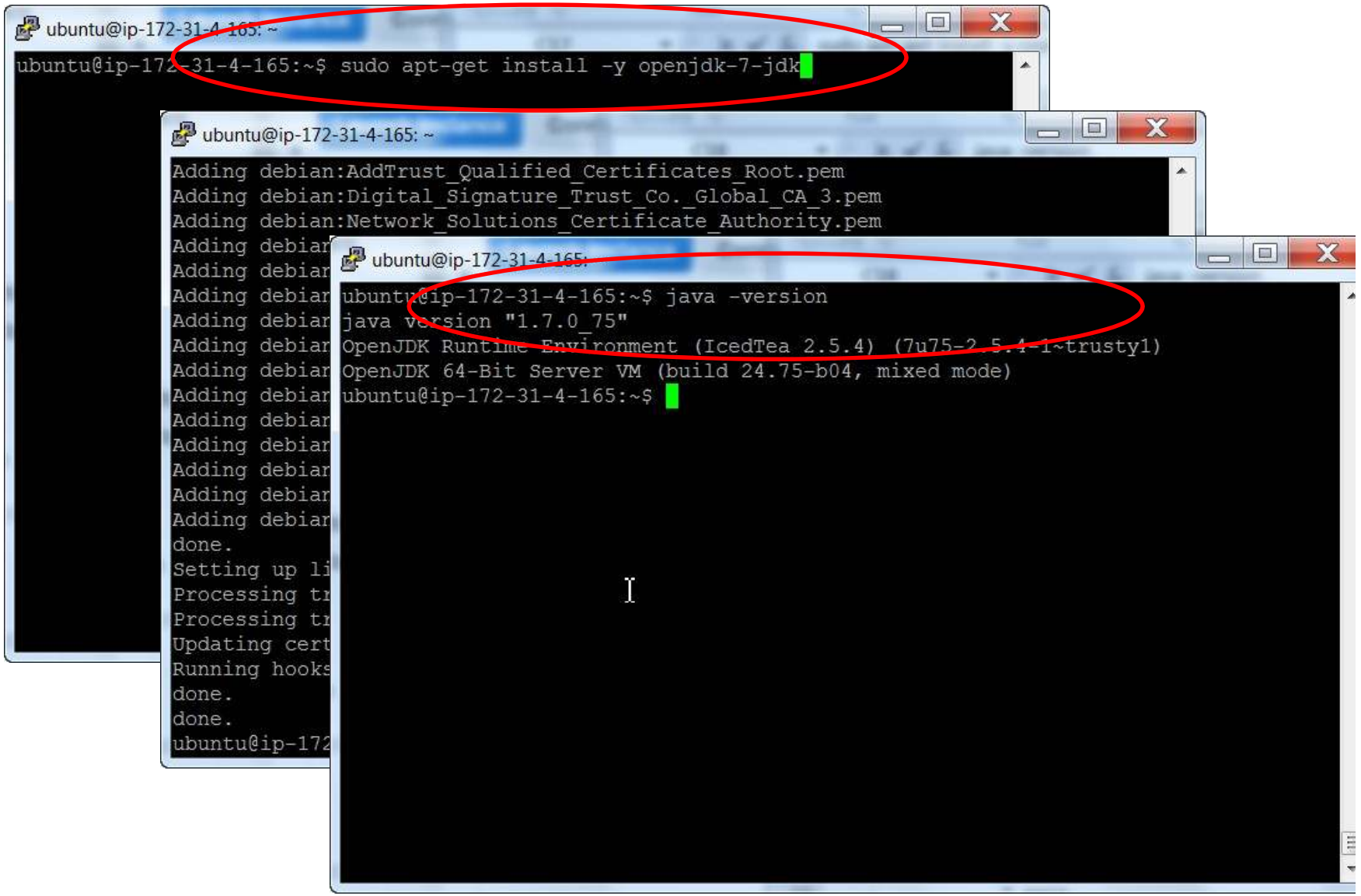


Exit from ssh connection



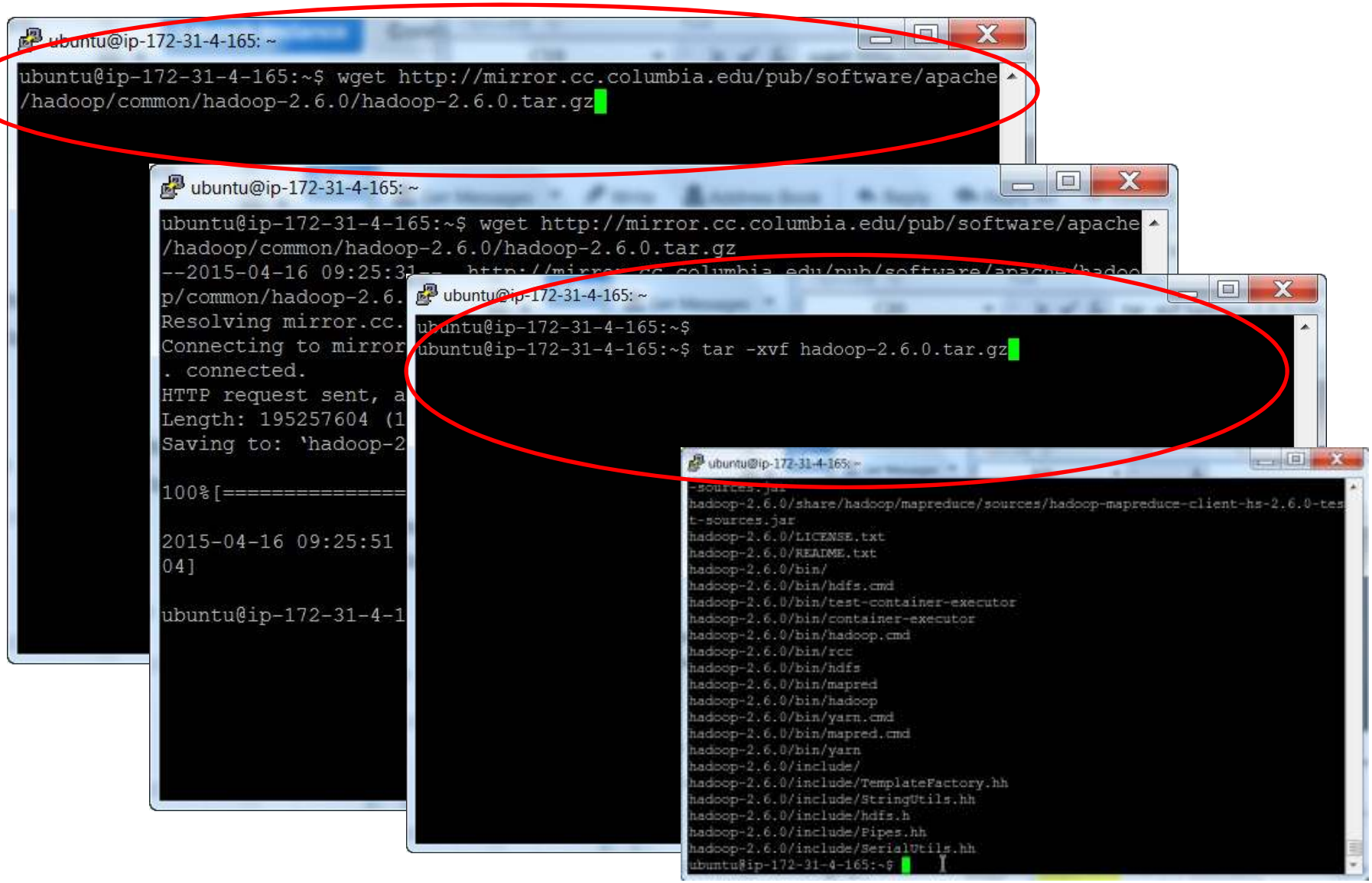
3. Installing Java

Installing Java



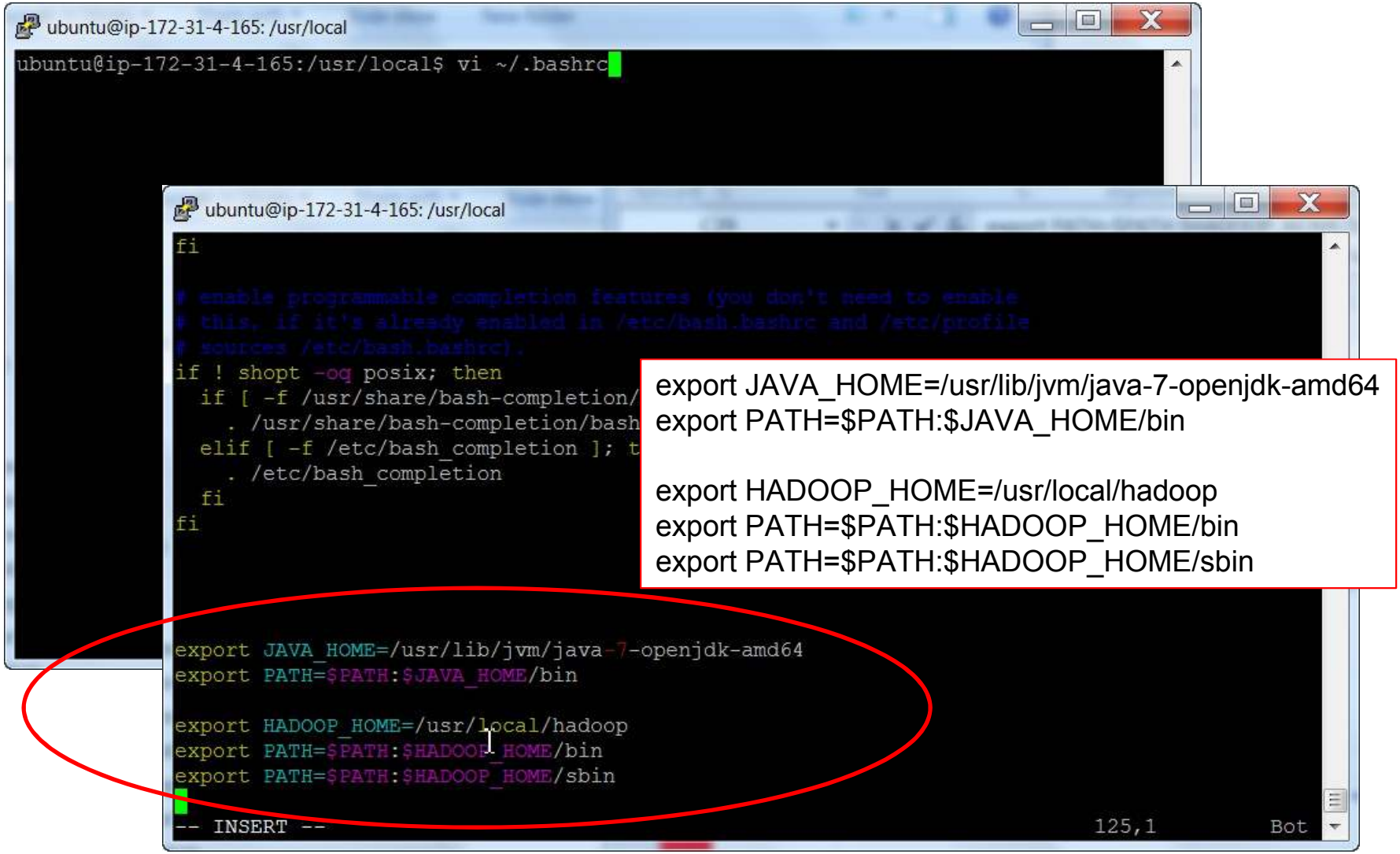
4. Download/Extract Hadoop

Download/Extract Hadoop

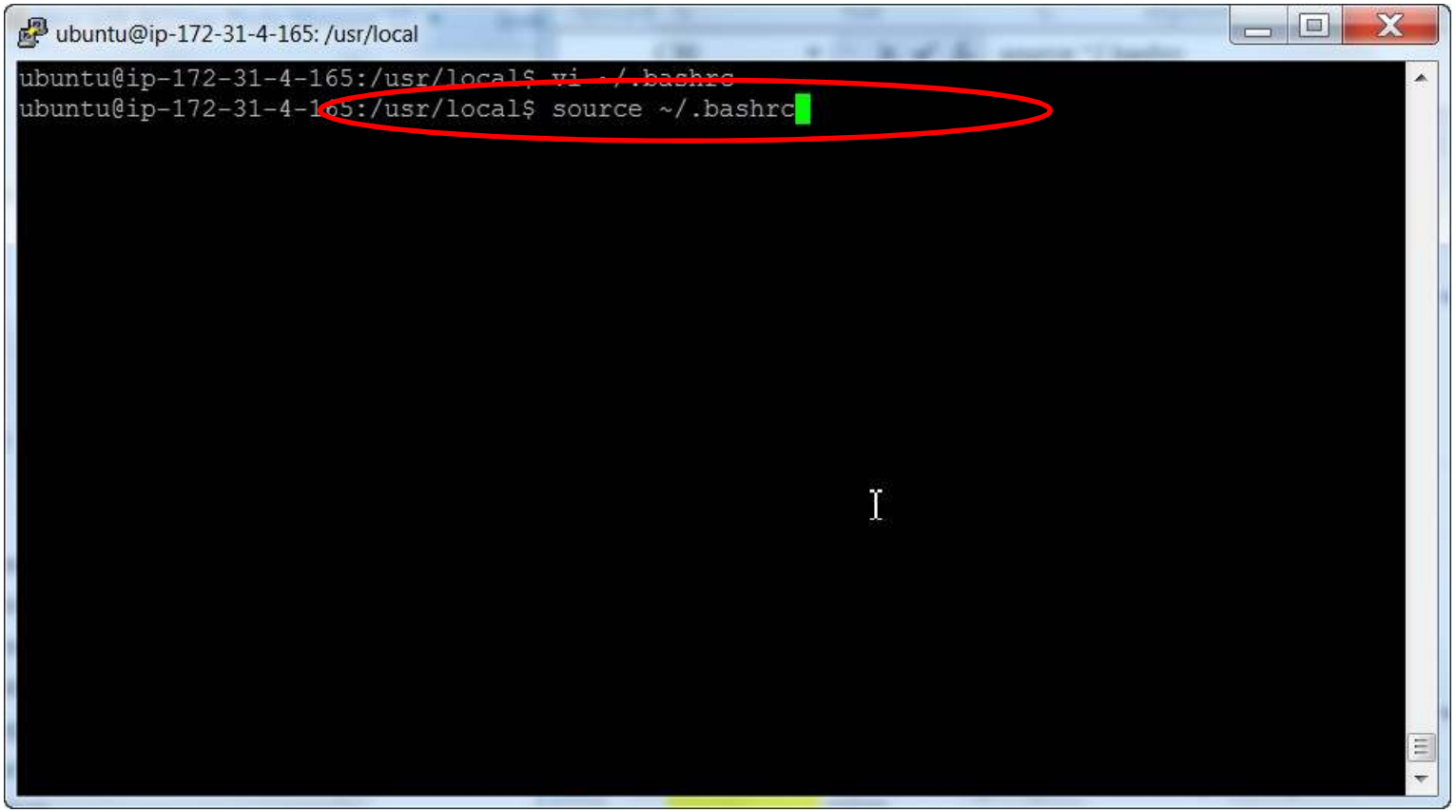


5. Installing Hadoop

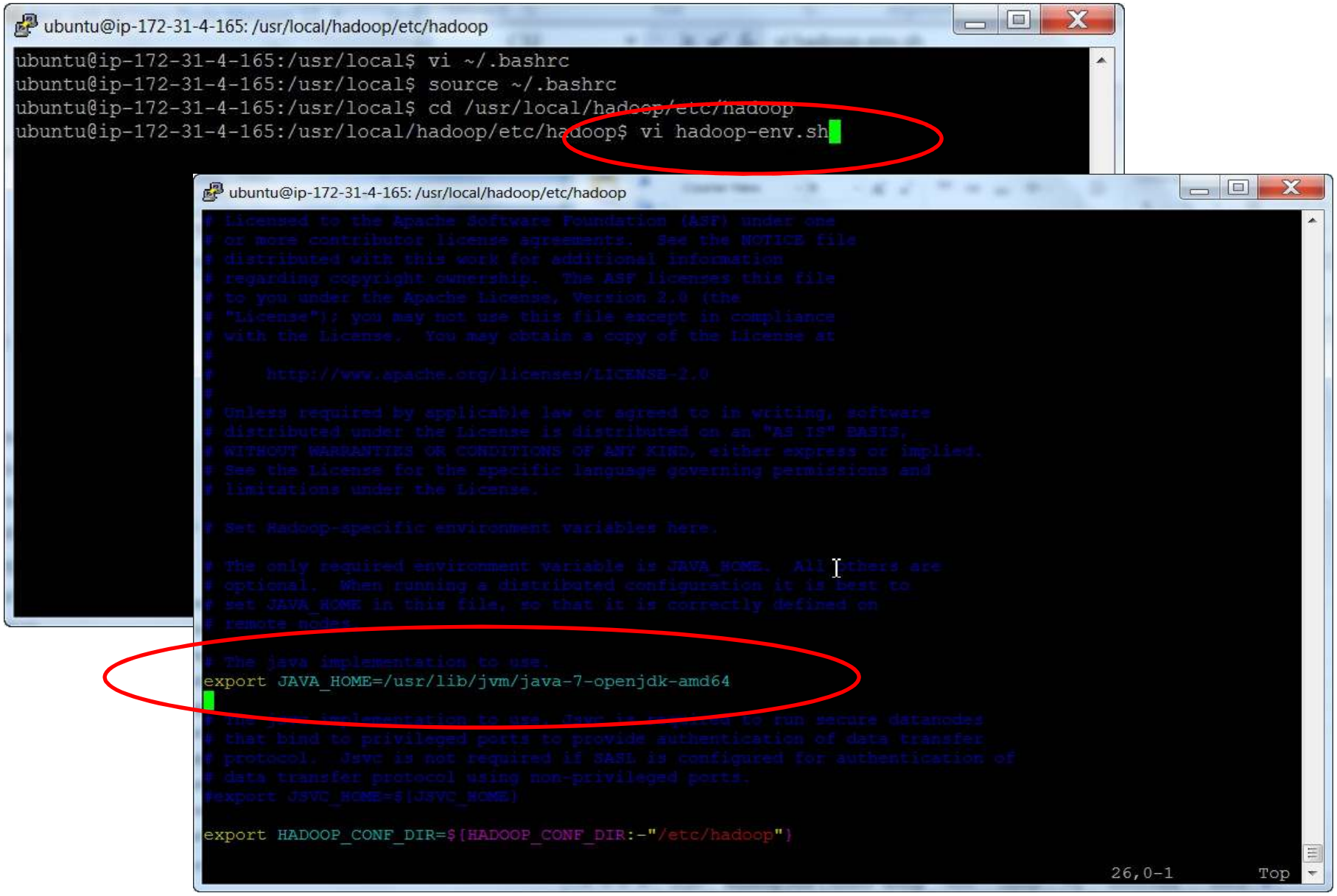
Installing Hadoop



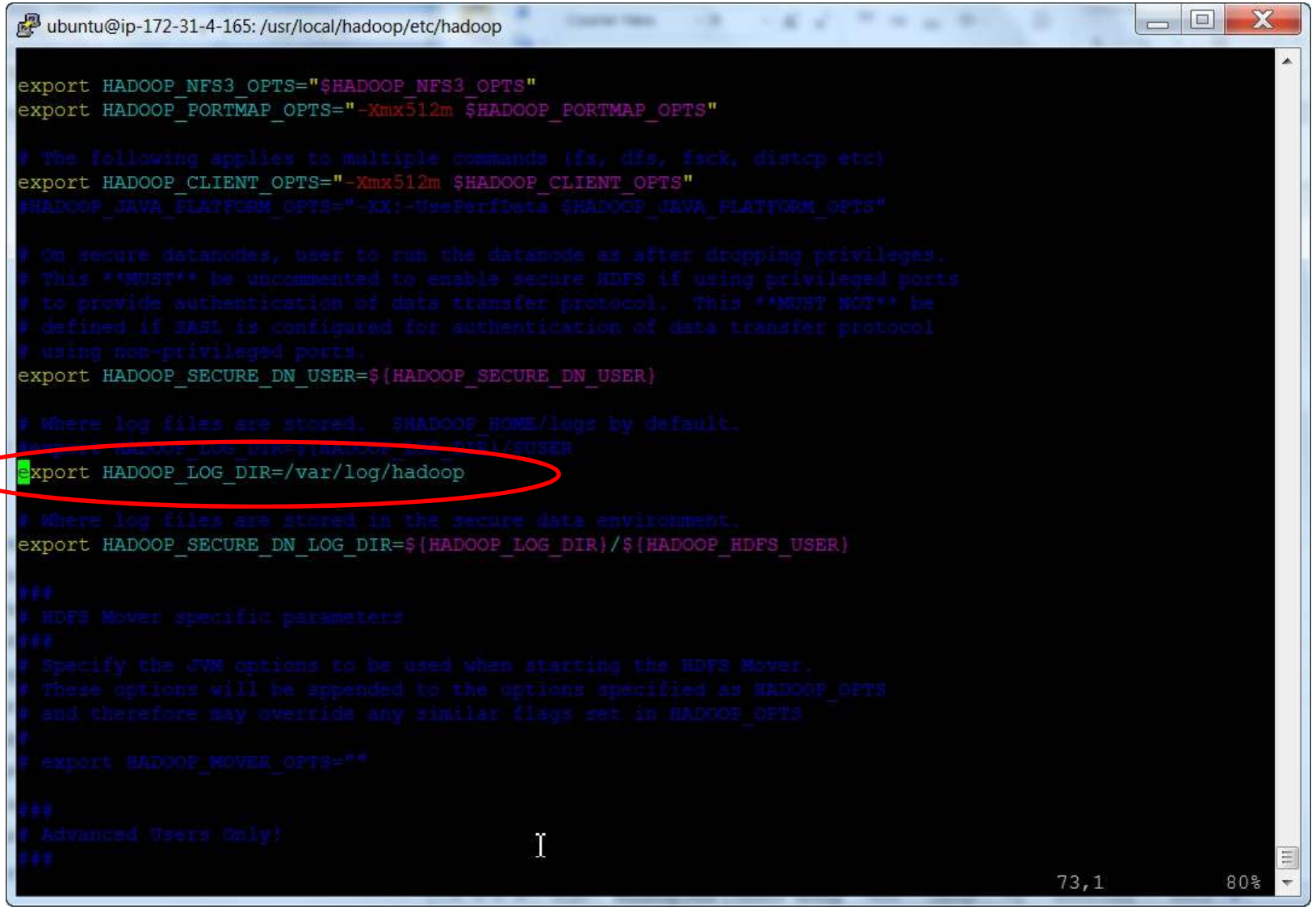
Execute environment variables: source ~/.bashrc



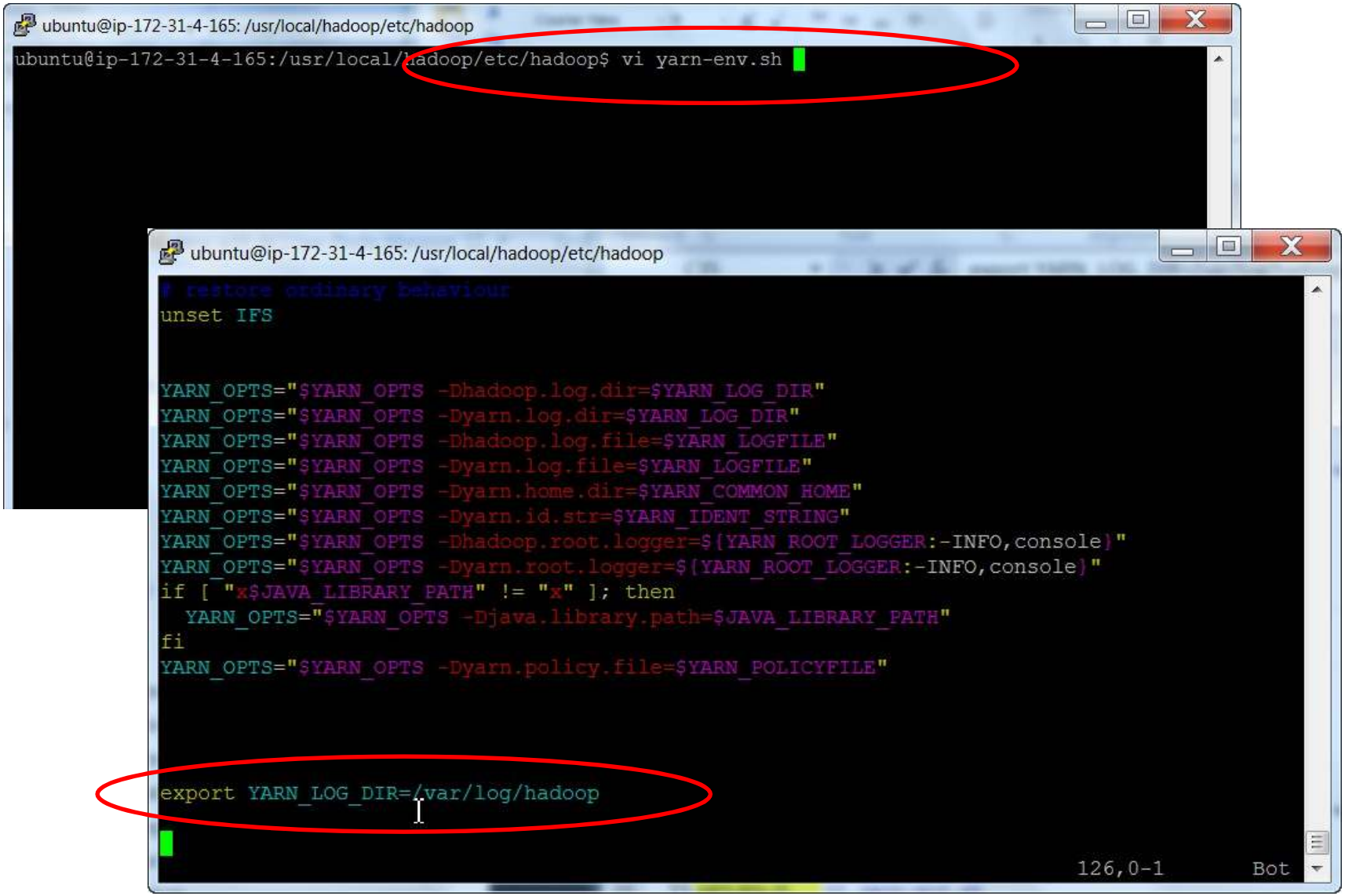
Edit hadoop shell script



Edit hadoop shell script – log location to another directory

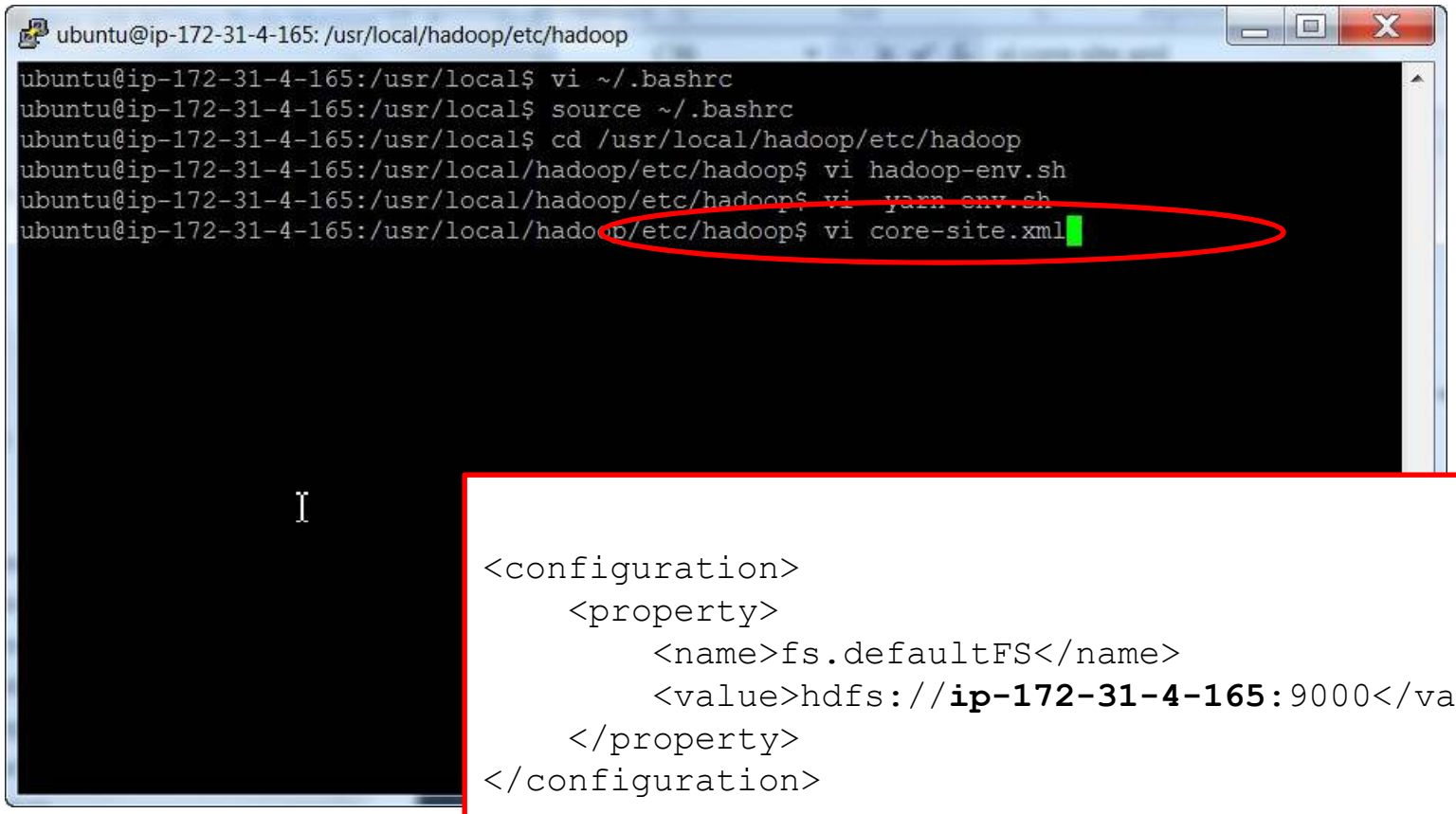


Edit Yarn shell script – log location to another directory



6. Configuring Hadoop

Edit hadoop - core-site.xml




```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://ip-172-31-4-165:9000</value>
    </property>
</configuration>
```


Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

Creating directories for namenode and datanode

```
sudo mkdir -p /var/hadoop_data/namenode
sudo mkdir -p /var/hadoop_data/datanode
sudo chown ubuntu:ubuntu -R /var/hadoop_data
```

Edit hdfs-site.xml

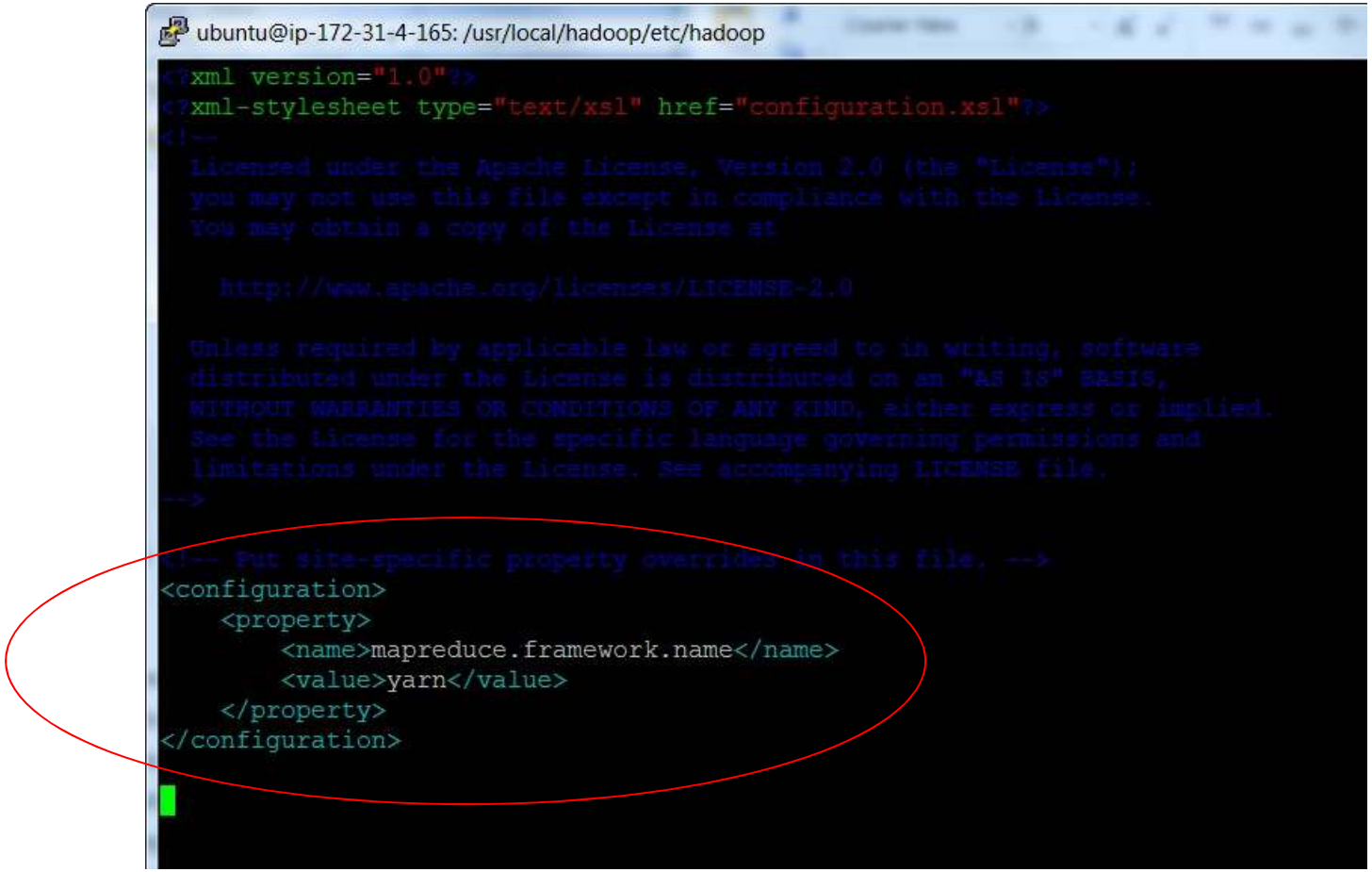
Edit yarn-site.xml

```
ubuntu@ip-172-31-4-165:/usr/local/hadoop/etc/hadoop$ vi yarn-site.xml
```

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>ip-172-31-4-165</value>
    </property>
    <property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>ip-172-31-4-165:8030</value>
    </property>
    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>ip-172-31-4-165:8031</value>
    </property>
    <property>
        <name>yarn.resourcemanager.address</name>
        <value>ip-172-31-4-165:8032</value>
    </property>
    <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>ip-172-31-4-165:8033</value>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>ip-172-31-4-165:8088</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
</configuration>
```

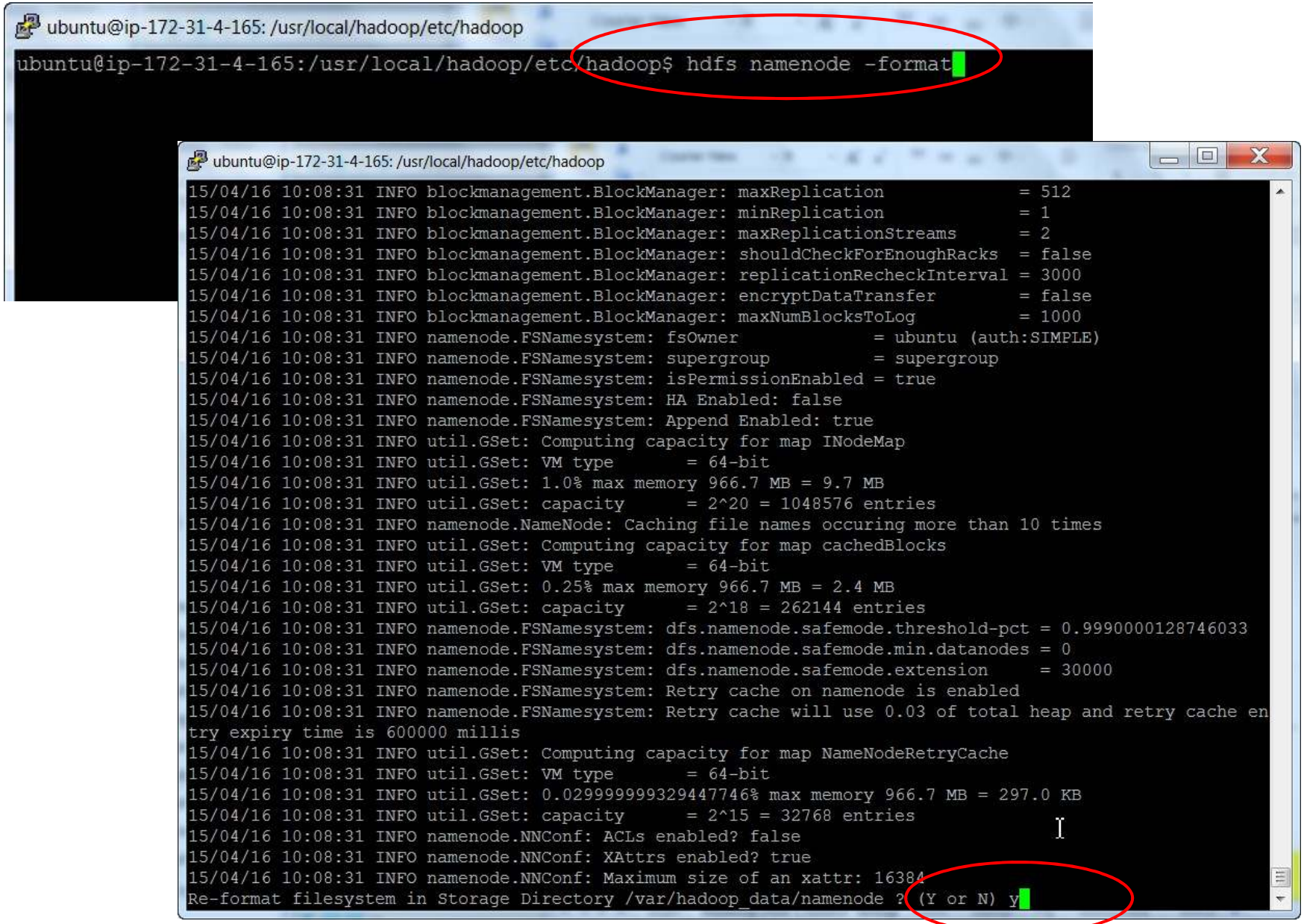
Edit mapred-site.xml

```
cp mapred-site.xml.template mapred-site.xml
vi mapred-site.xml
```

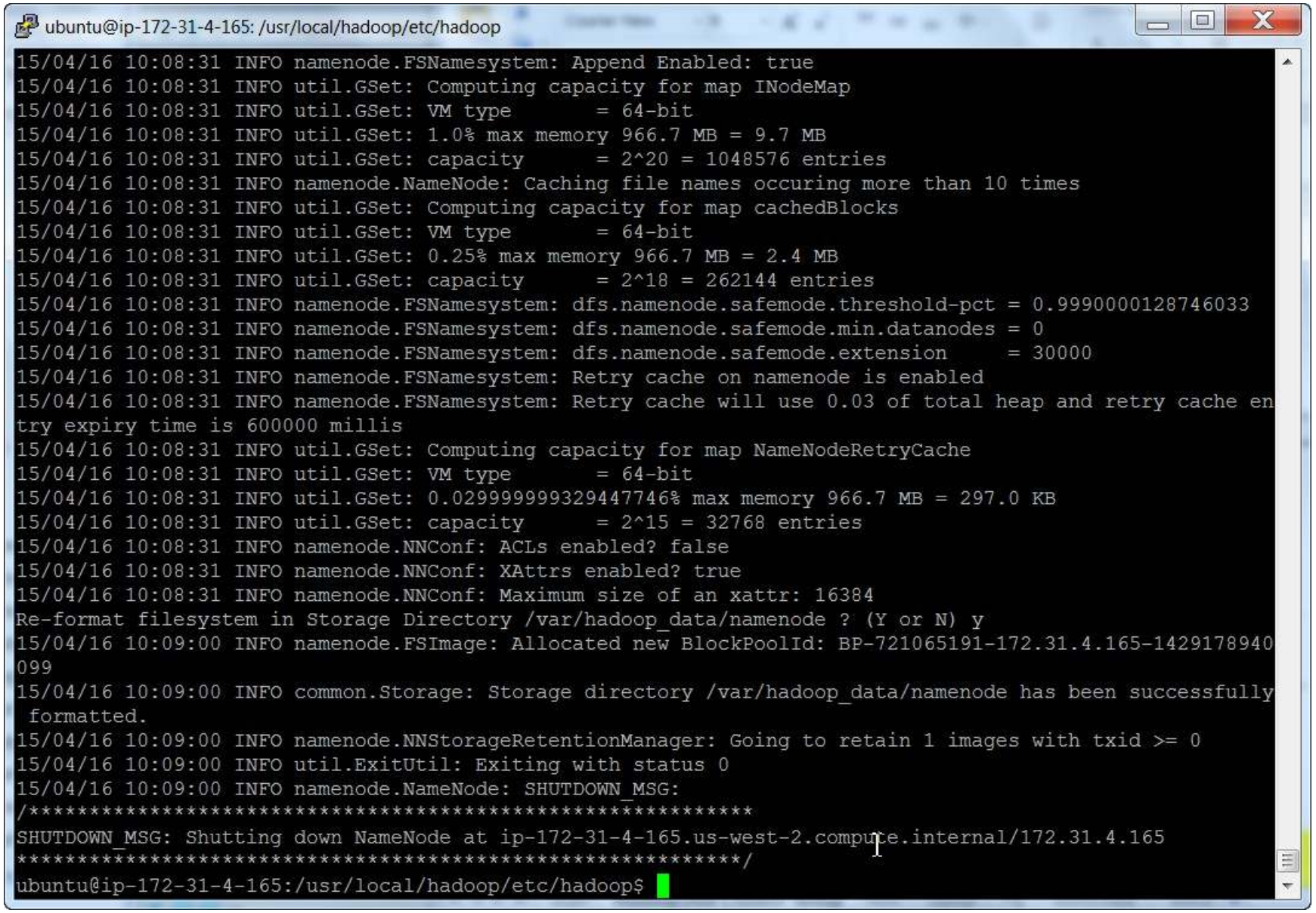


7. Formatting Namenode

Formatting Namenode

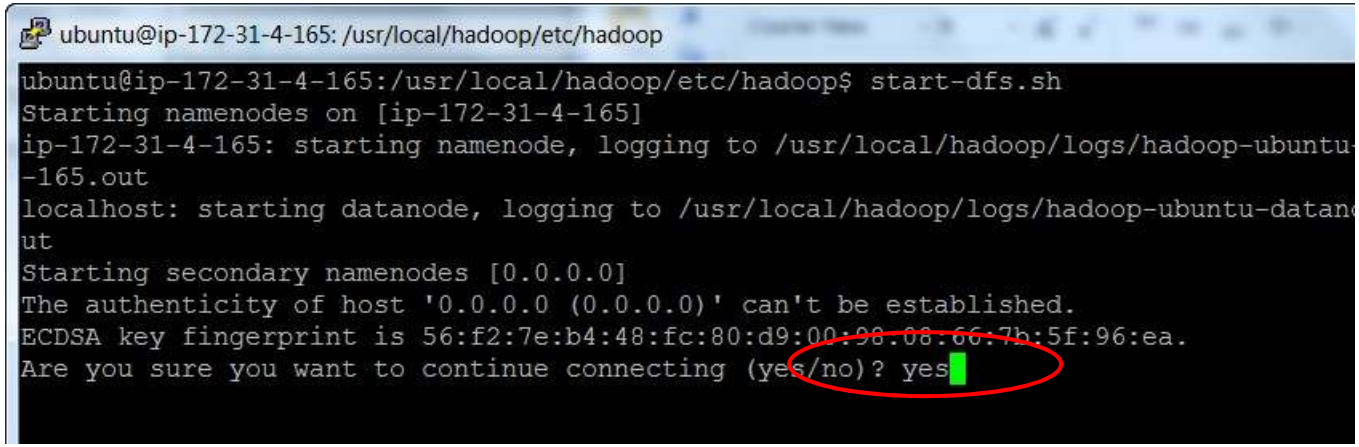


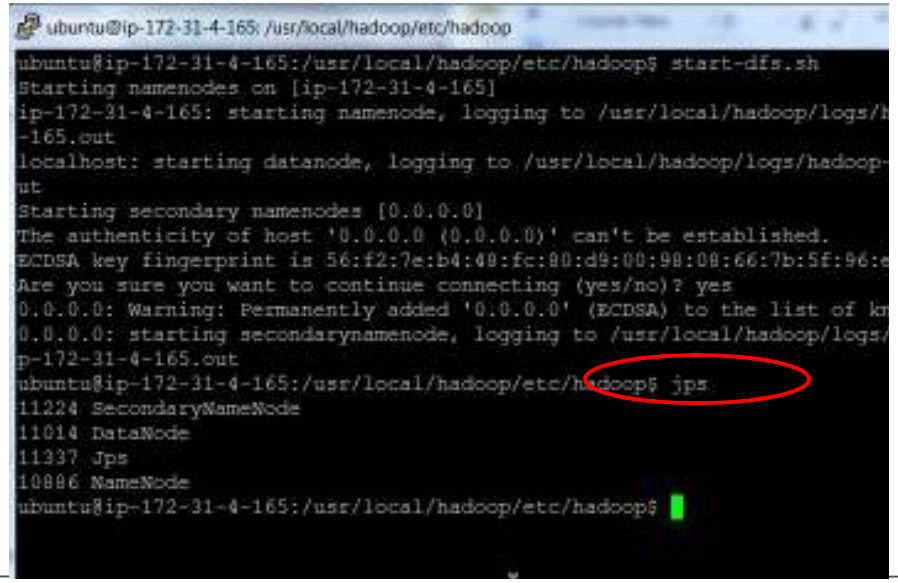
Formatting Namenode



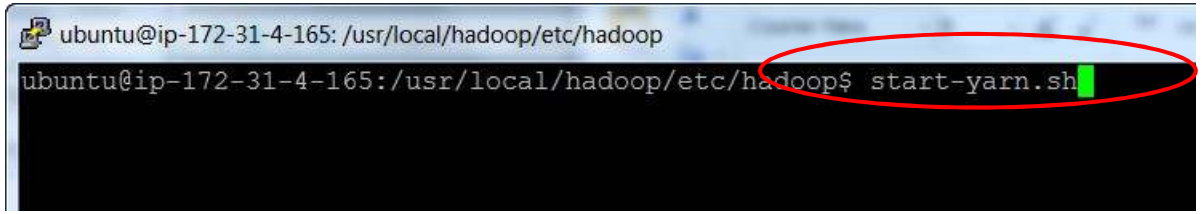
8. Starting Hadoop

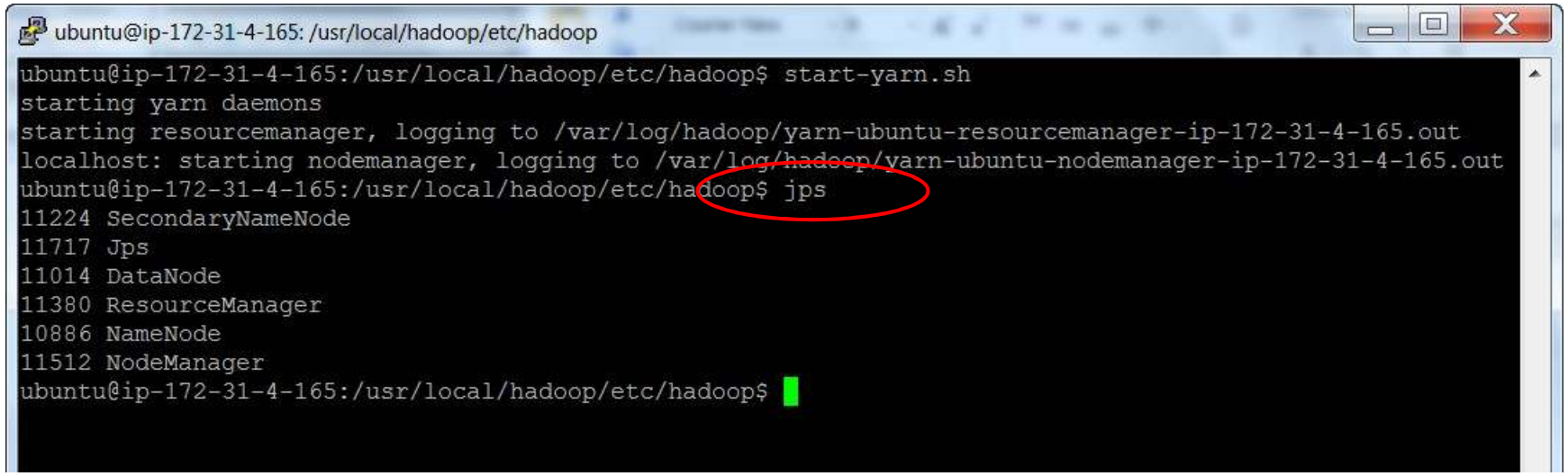
Starting Namenode and Datanode





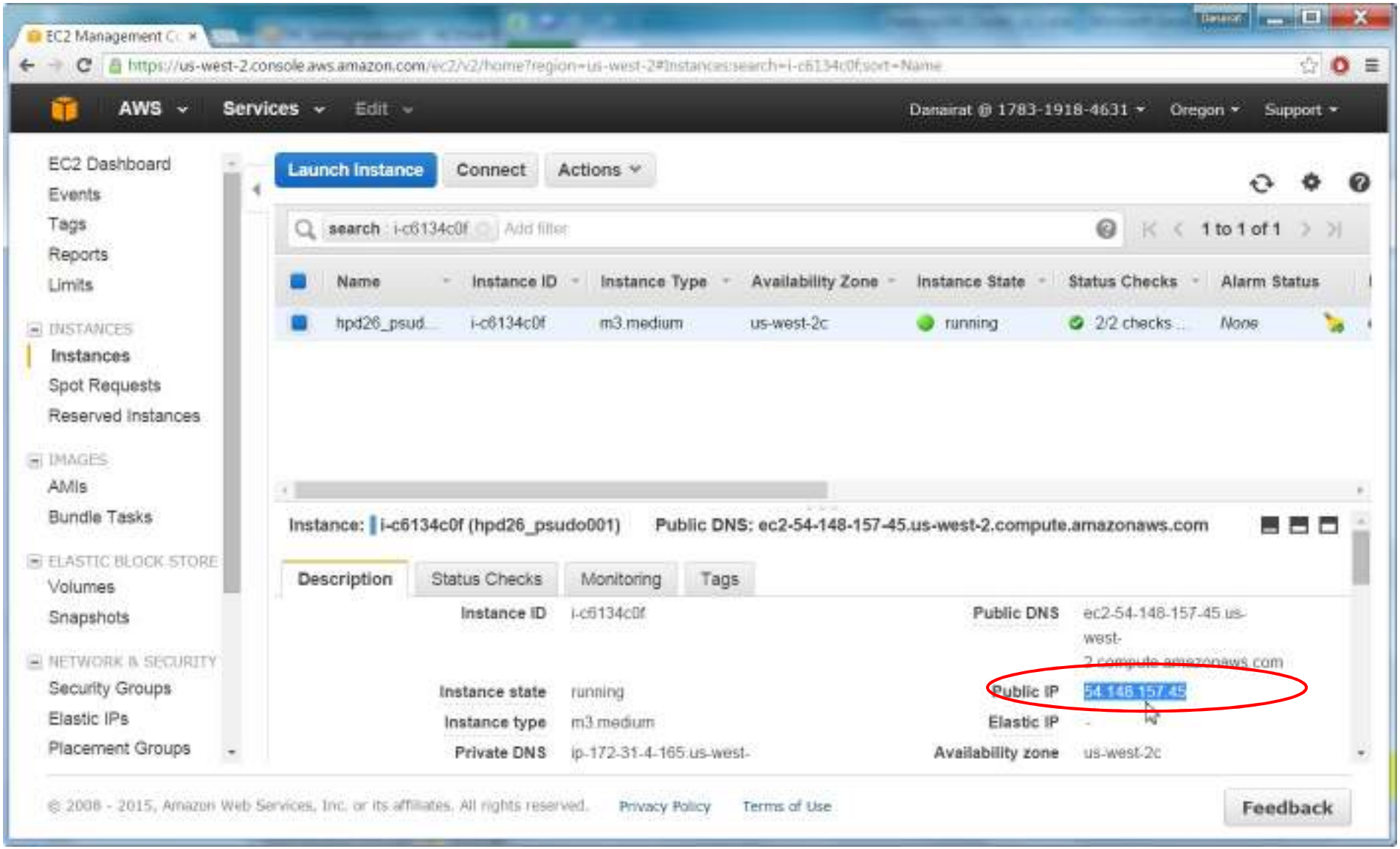
Starting Yarn



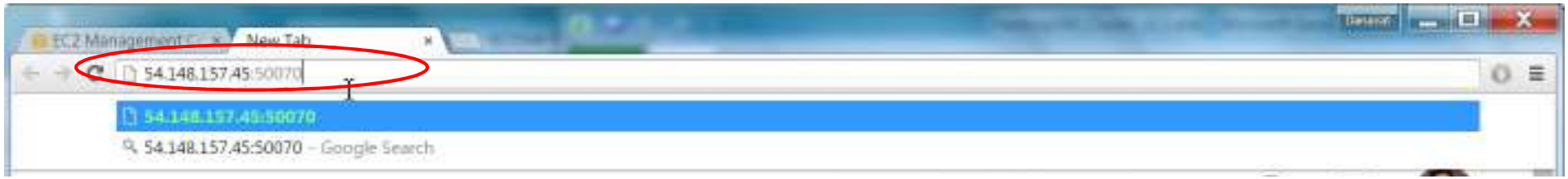


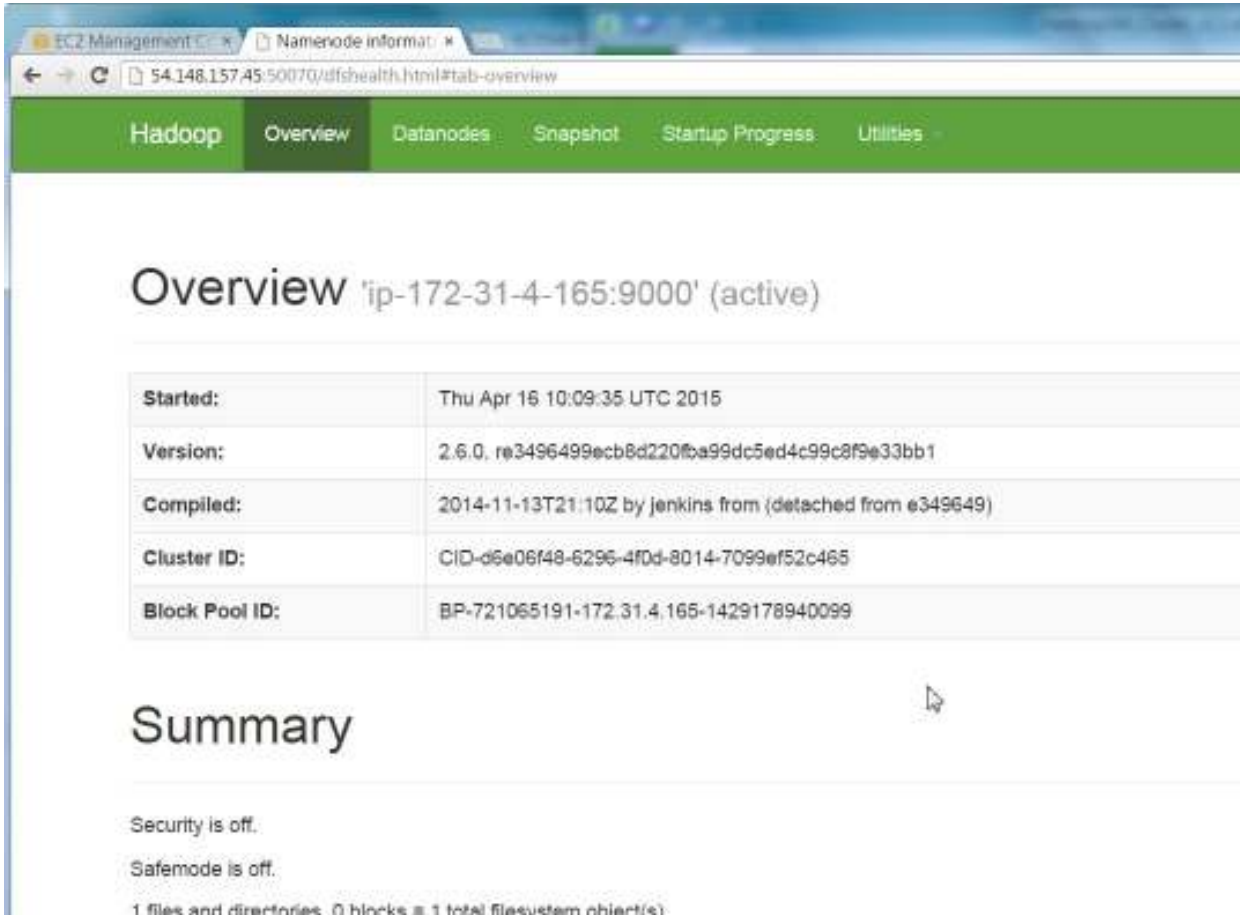
9. Accessing Hadoop Web Console

Checking Public IT Address



Accessing Hadoop Web Console



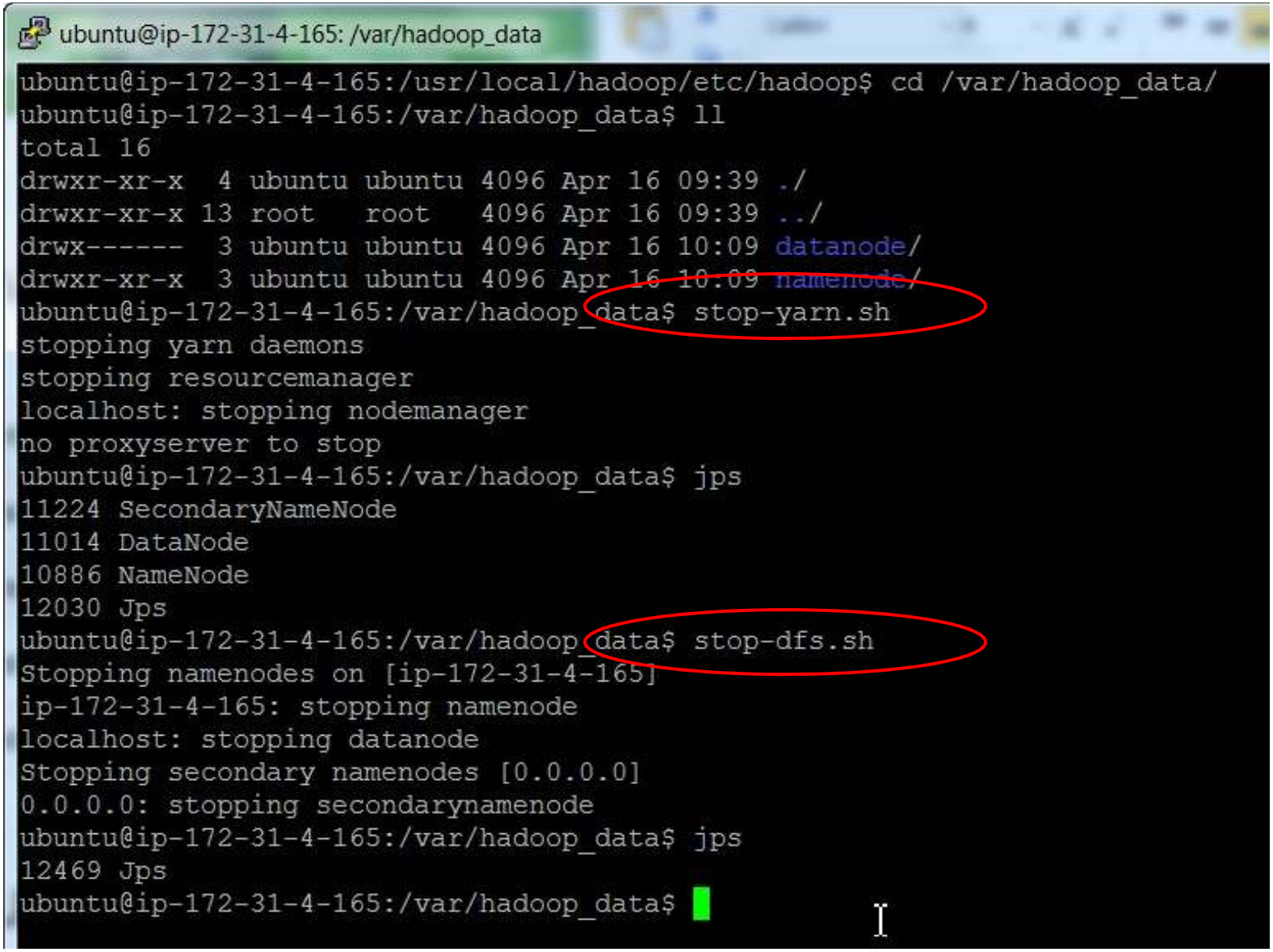


Hadoop Port Numbers

	Daemon	Default Port	Configuration Parameter in conf/*-site.xml
HDFS	Namenode Web	50070	dfs.http.address
	Datanodes	50075	dfs.datanode.http.address
	Secondarynamenode	50090	dfs.secondary.http.address
Yarn	ResourceManager Web	8088	yarn.resourcemanager.webapp.address

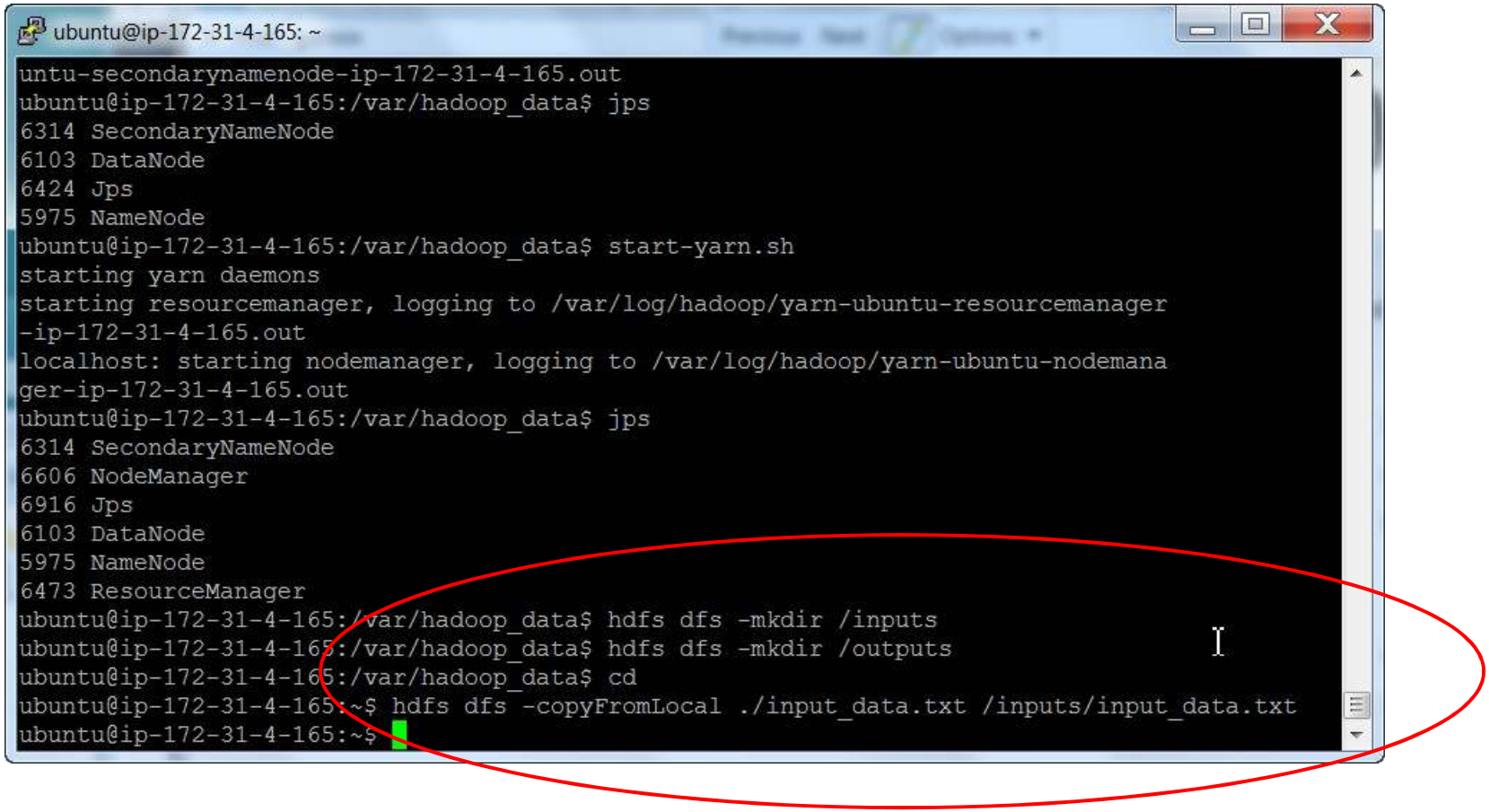
10. Stopping Hadoop

Stop Hadoop Yarn and HDFS



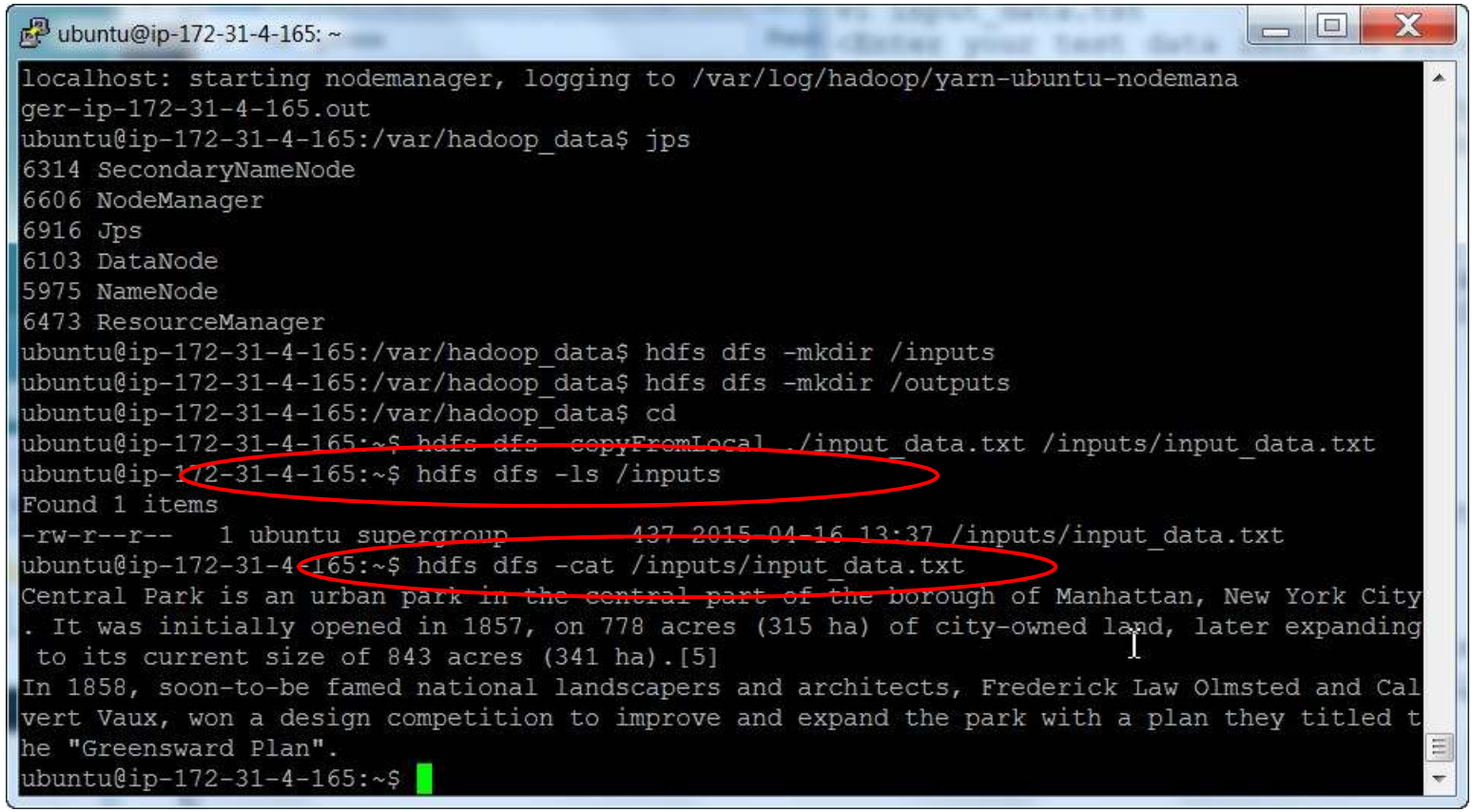
Hands-On: Importing Data to HDFS using Hadoop Command Line

Creating Hadoop HDFS Directories and Importing file to Hadoop



Hands-On: Traversing, Retrieving Data from HDFS

Review data in Hadoop HDFS



Deleting file from HDFS

rm

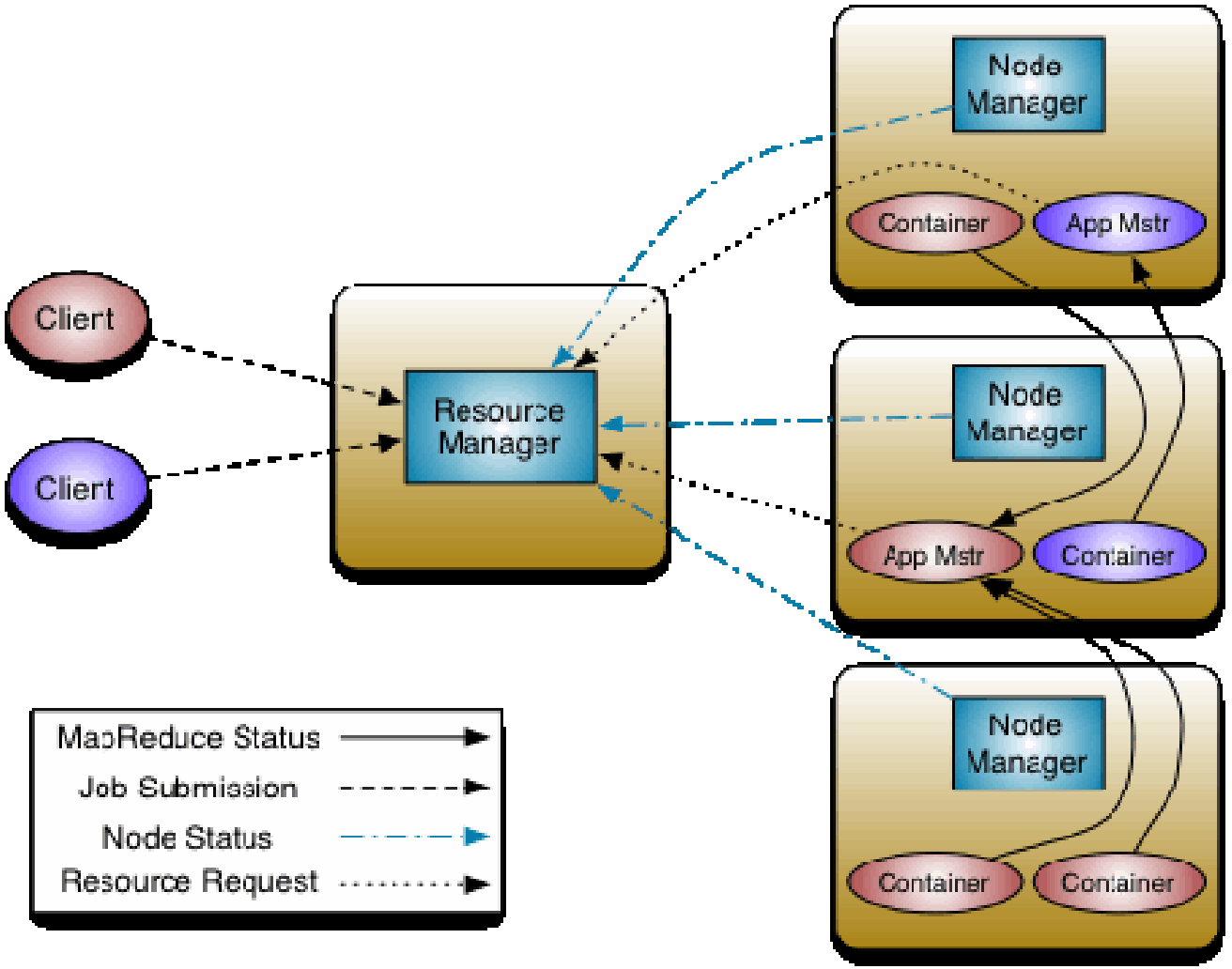
```
Usage: hdfs dfs -rm [-f] [-r|-R] [-skipTrash] URI [URI ...]
Delete files specified as args.
Options:
The -f option will not display a diagnostic message or modify the
exit status to reflect an error if the file does not exist.
The -R option deletes the directory and any content under it
recursively.
The -r option is equivalent to -R.
The -skipTrash option will bypass trash, if enabled, and delete
the specified file(s) immediately. This can be useful when it is
necessary to delete files from an over-quota directory.
```

```
Example:
hdfs dfs -rm hdfs://nn.example.com/file /user/hadoop/mydir
```

Lecture: YARN

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

YARN: Yet Another Resource Negotiator



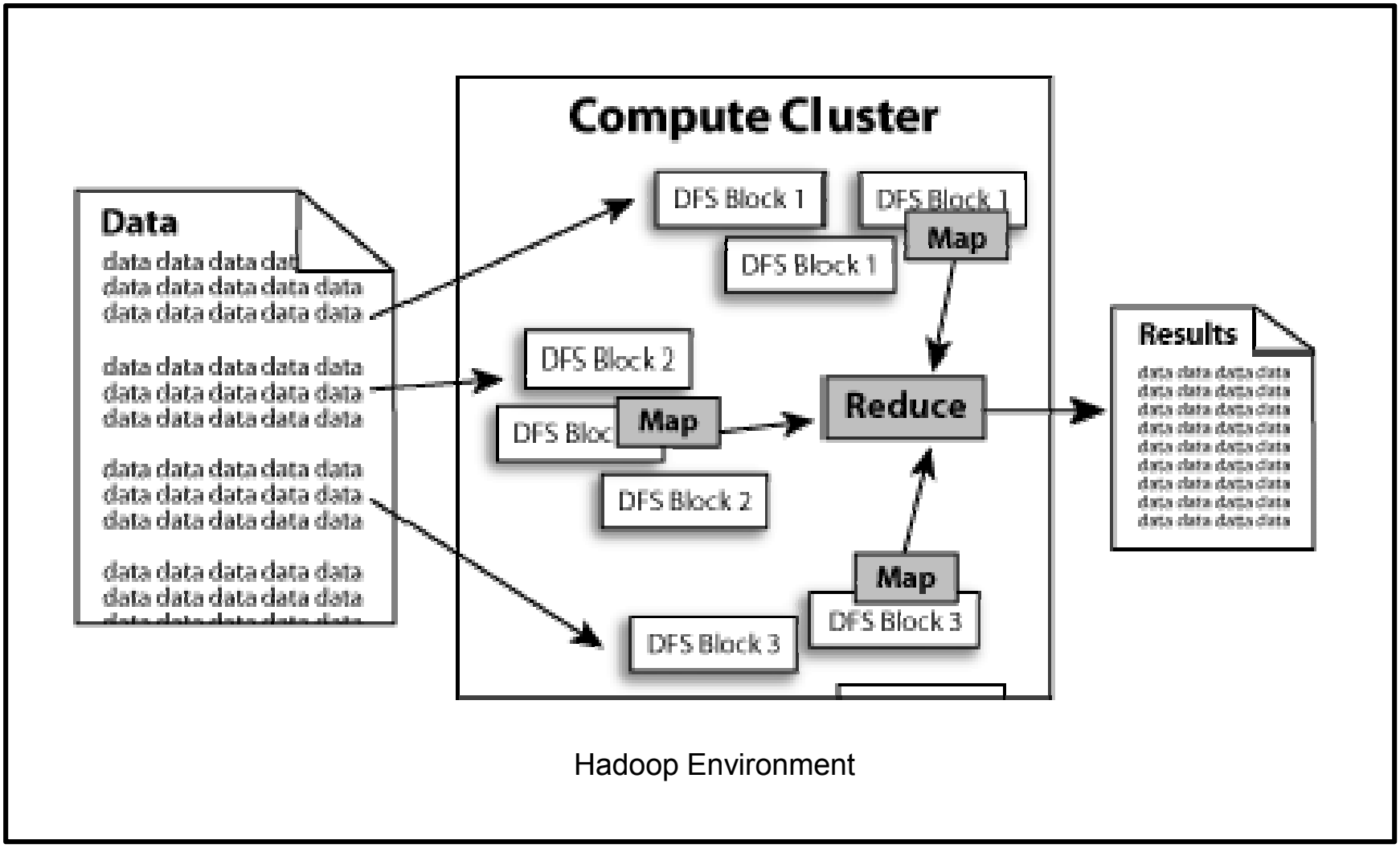
MRV2 maintains API compatibility with previous stable release (hadoop-1.x). This means that all Map-Reduce jobs should still run unchanged on top of MRv2 with just a recompile.

Hadoop.apache.org

Lecture: Map Reduce

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

MapReduce



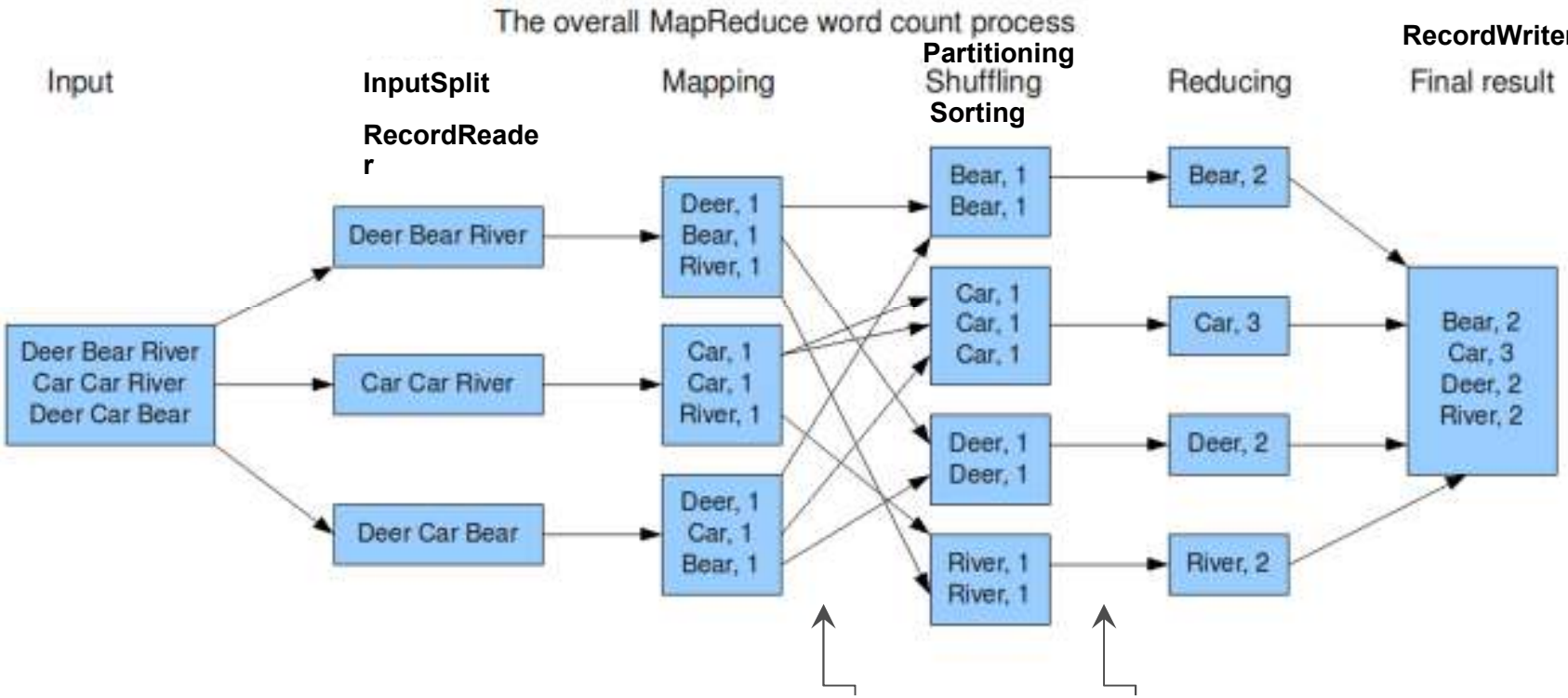
Hadoop Environment

Hadoop.apache.org

MapReduce Framework

```
map: (K1, V1) -> list(K2, V2))
reduce: (K2, list(V2)) -> list(K3, V3)
```

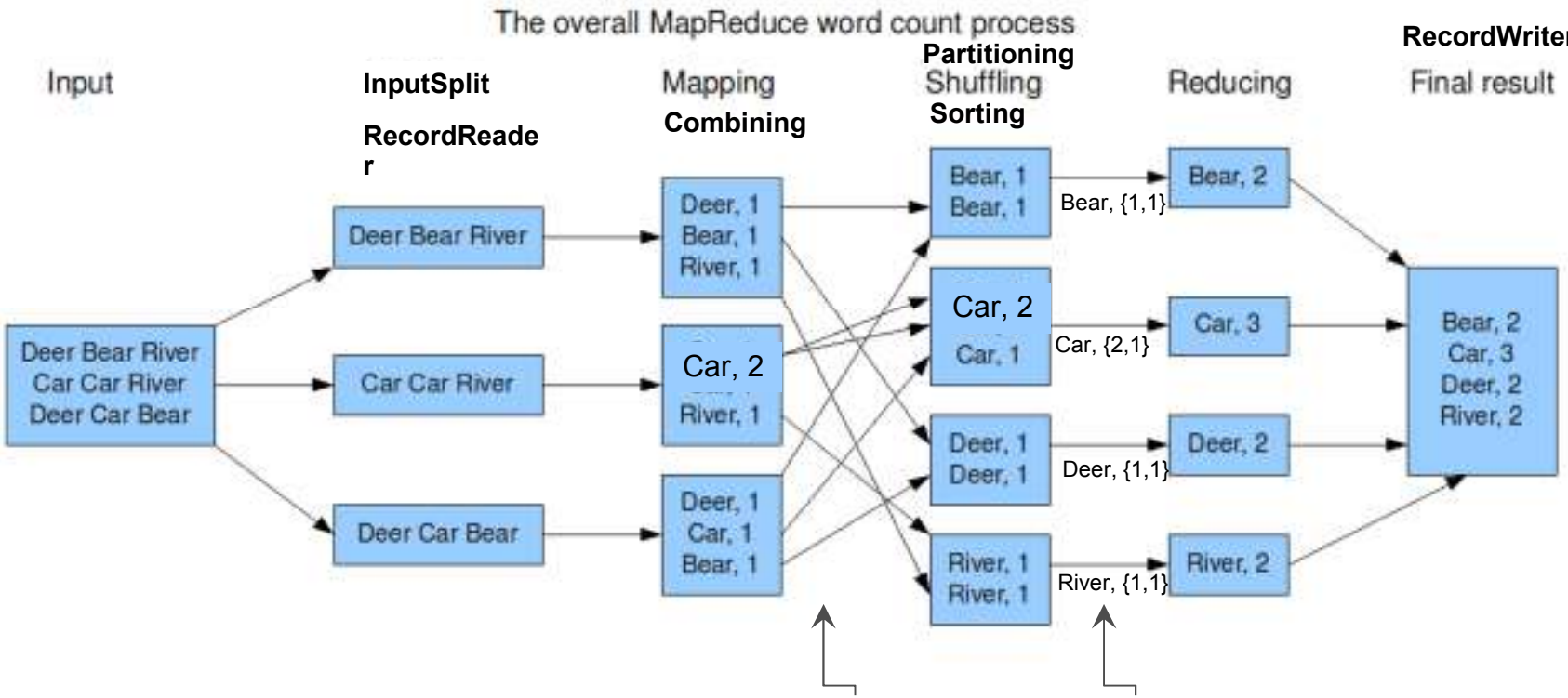
How does the MapReduce work?



Output in a list of (Key, Value) in the intermediate file

Output in a list of (Key, List of Values) in the intermediate file

How does the MapReduce work?



Output in a list of (Key, Value) in the intermediate file

Output in a list of (Key, List of Values) in the intermediate file

MapReduce Processing – The Data flow

1. InputFormat, InputSplits, RecordReader
2. Mapper - your focus is here
3. Partition, Shuffle & Sort
4. Reducer - your focus is here
5. OutputFormat, RecordWriter

InputFormat

TextInputFormat	**Default format; reads lines of text files**	**The byte offset of the line**	**The line contents**
KeyValueInputFormat	**Parses lines into key, val pairs**	**Everything up to the first tab character**	**The remainder of the line**
SequenceFileInputFormat	**A Hadoop-specific high-performance binary format**	**user-defined**	**user-defined**

InputSplit

An InputSplit describes a unit of work that comprises a single *map task.*

InputSplit presents a byte-oriented view of the input.

You can control this value by setting the mapred.min.split.size parameter in core-site.xml, or by overriding the parameter in the JobConf object used to submit a particular MapReduce job.

RecordReader

RecordReader reads <key, value> pairs from an InputSplit.

Typically the RecordReader converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented to the Mapper

Mapper

Mapper: The Mapper performs the user-defined logic to the input a key, value and emits (key, value) pair(s) which are forwarded to the Reducers.

Partition, Shuffle & Sort

After the first map tasks have completed, the nodes may still be performing several more map tasks each. But they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers.

Partitioner controls the partitioning of map-outputs to assign to reduce task . he total number of partitions is the same as the number of reduce tasks for the job

The set of intermediate keys on a single node is automatically sorted by internal Hadoop before they are presented to the Reducer

This process of moving map outputs to the reducers is known as shuffling.

Reducer

This is an instance of user-provided code that performs read each key, iterator of values in the partition assigned. The *OutputCollector* object in Reducer phase has a method named collect() which will collect a (key, value) output.

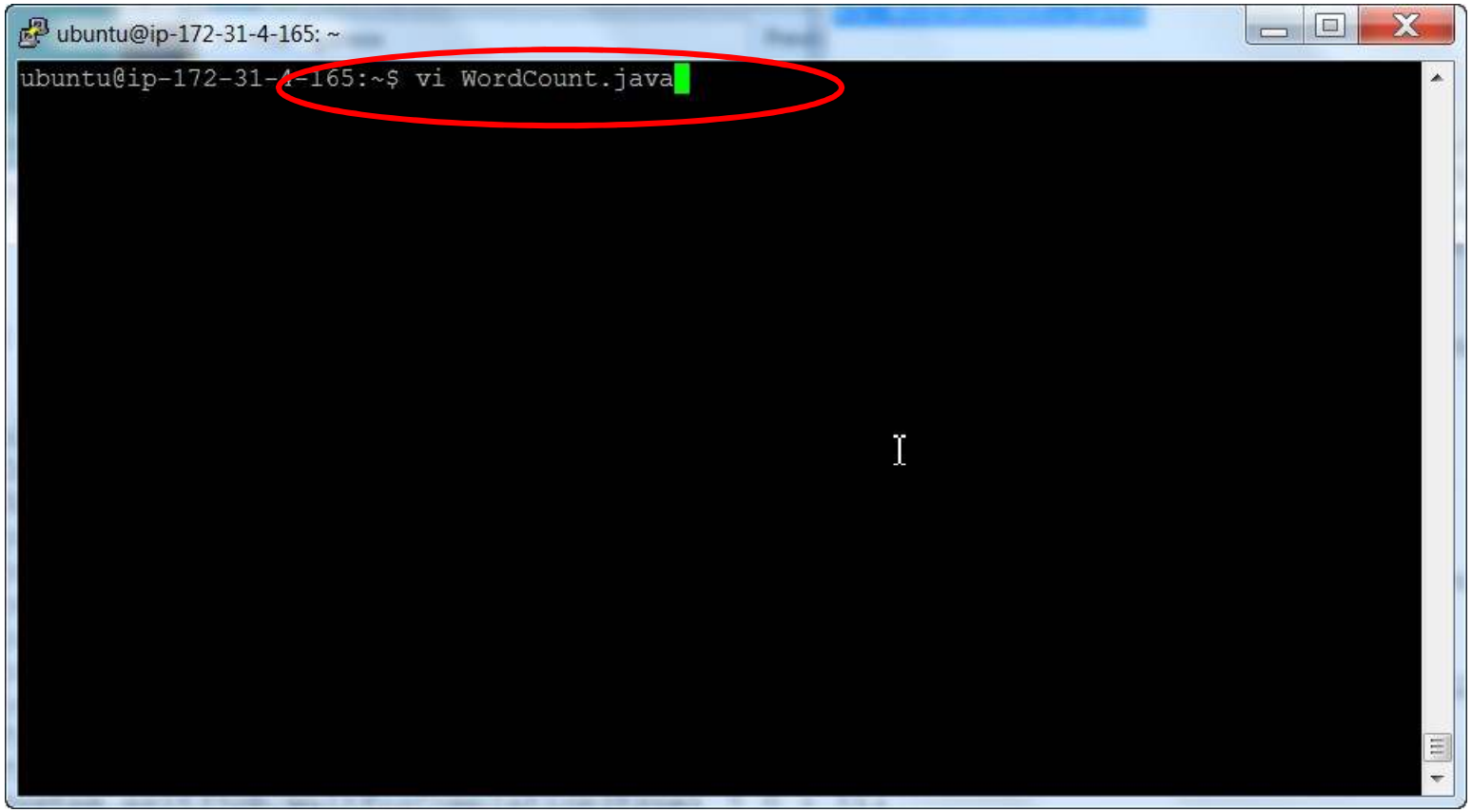
OutputFormat, Record Writer

OutputFormat governs the writing format in OutputCollector and RecordWriter writes output into HDFS.

TextOutputFormat	Default; writes lines in "key \t value" form
SequenceFileOutputFormat	Writes binary files suitable for reading into subsequent MapReduce jobs
NullOutputFormat	generates no output files

Hands-On: Writing Your Own Map Reduce

Writing Your Own Map Reduce



Writing Your Own Map Reduce

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

Continue (มีต่อ)..

Writing Your Own Map Reduce

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

Continue (มีต่อ)..

Writing Your Own Map Reduce

```
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Hands-On: Packaging Map Reduce and Deploying to Hadoop Runtime Environment

Packaging Map Reduce and Deploying to Hadoop Runtime Environment

Create java classes directory (สร้าง directory สำหรับ java class ที่ถูก compile แล้ว

```
mkdir wordcount_classes
```

Typing in one single line (พิมพต่อเนื่องเลย อย่ากดแป้นขึ้นบรรทัดใหม่)

```
javac -classpath /usr/local/hadoop/share/hadoop/common/hadoop-common-
2.6.0.jar:/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-
core-2.6.0.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-cli-1.2.jar -d
wordcount_classes WordCount.java
```

Create jar file (สร้าง jar file อย่าลืมใส่เครื่องหมายจุดด้านหลังด้วย)

```
jar -cvf ./wordcount.jar -C wordcount_classes/
```

Execute yarn (สั่งให้ yarn ทำงาน)

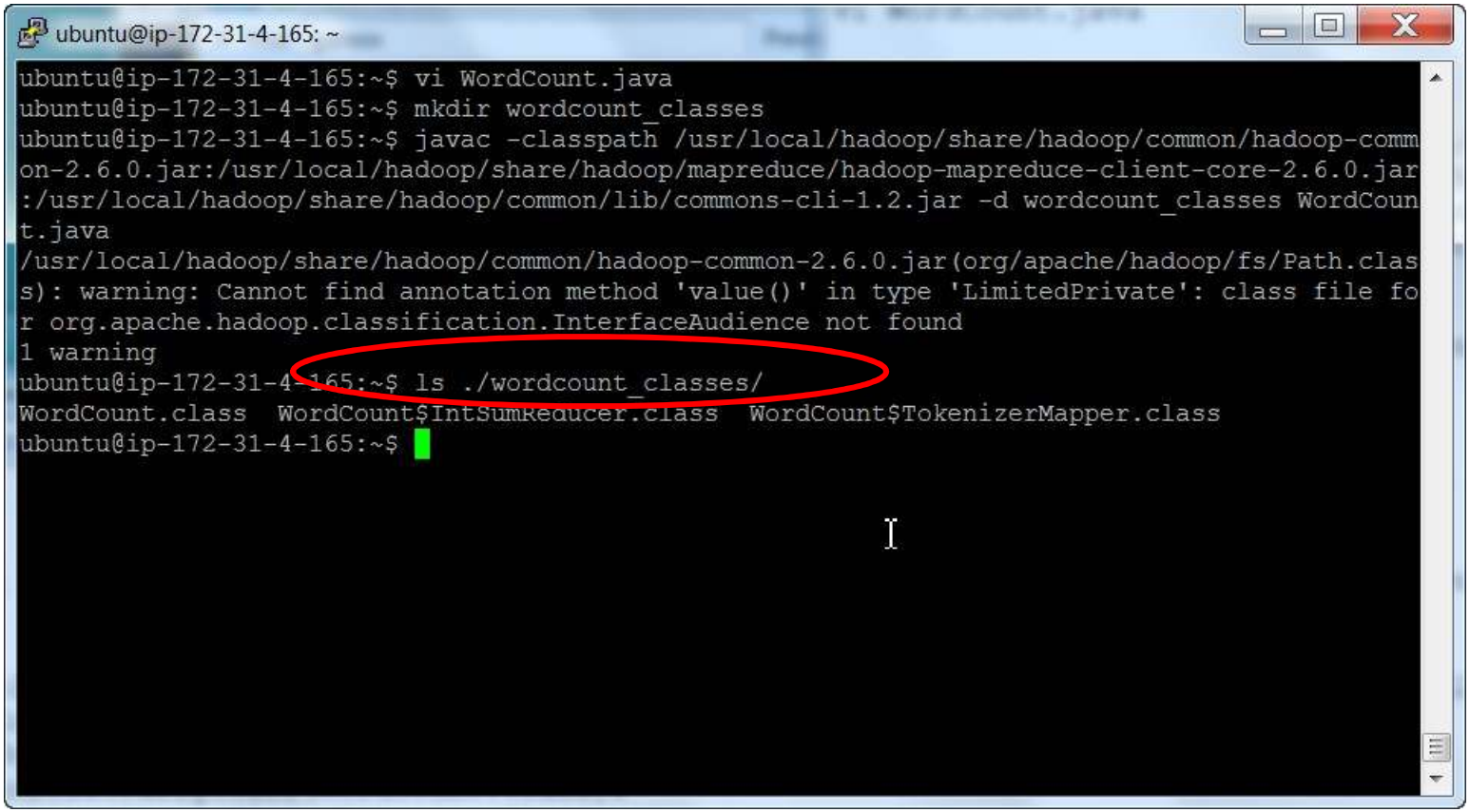
```
yarn jar ./wordcount.jar WordCount /inputs/* /outputs/wordcount_output_dir01
```

Review the result (สั่งให้พิมพ์ผลลัพท์ออกหน้าจอ)

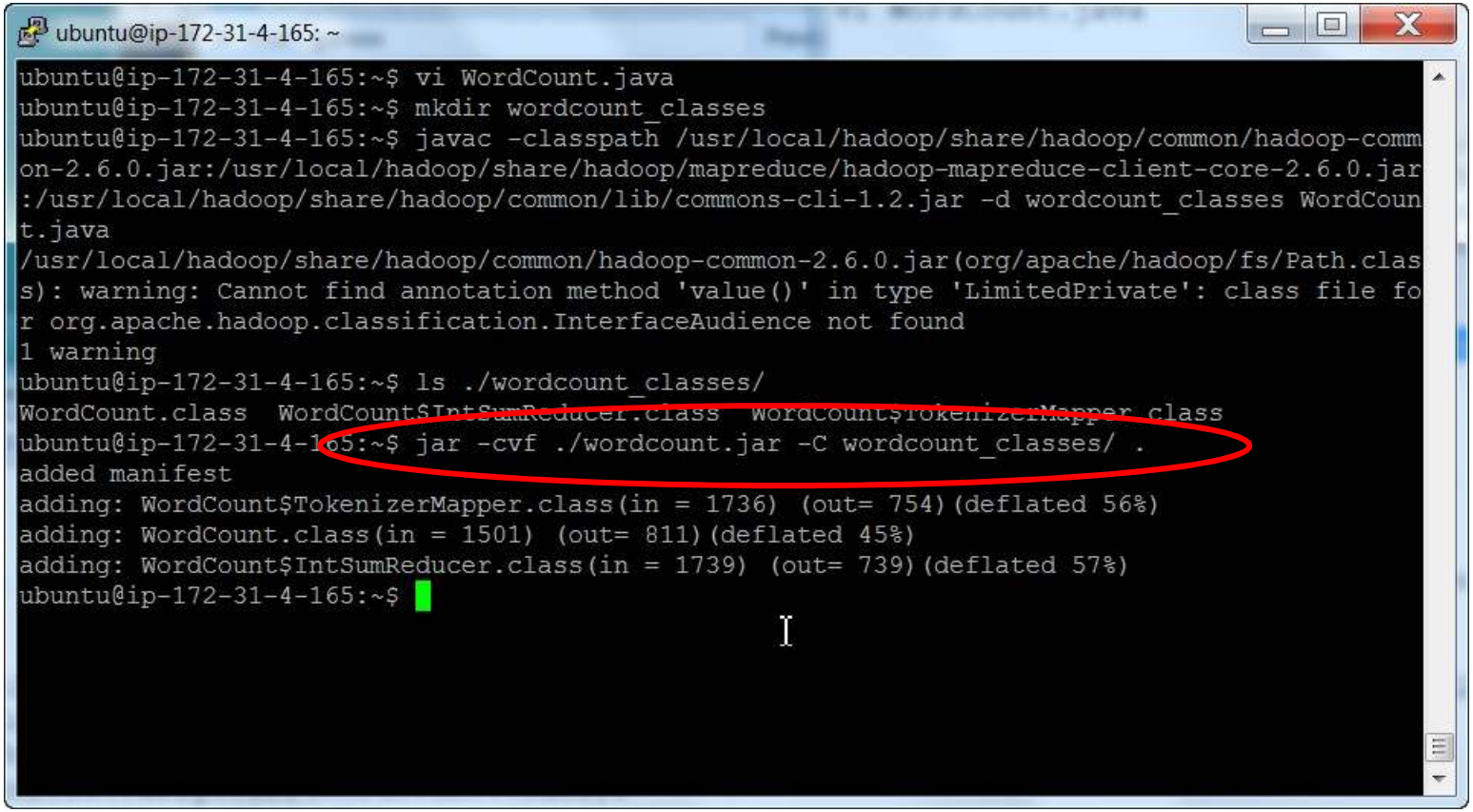
```
hdfs dfs -cat /outputs/wordcount_output_dir01/part-r-00000
```

Please find next page for screen step by step (โปรดดูหน้าถัดไปสำหรับหน้าจอในแต่ละขั้นตอน)...

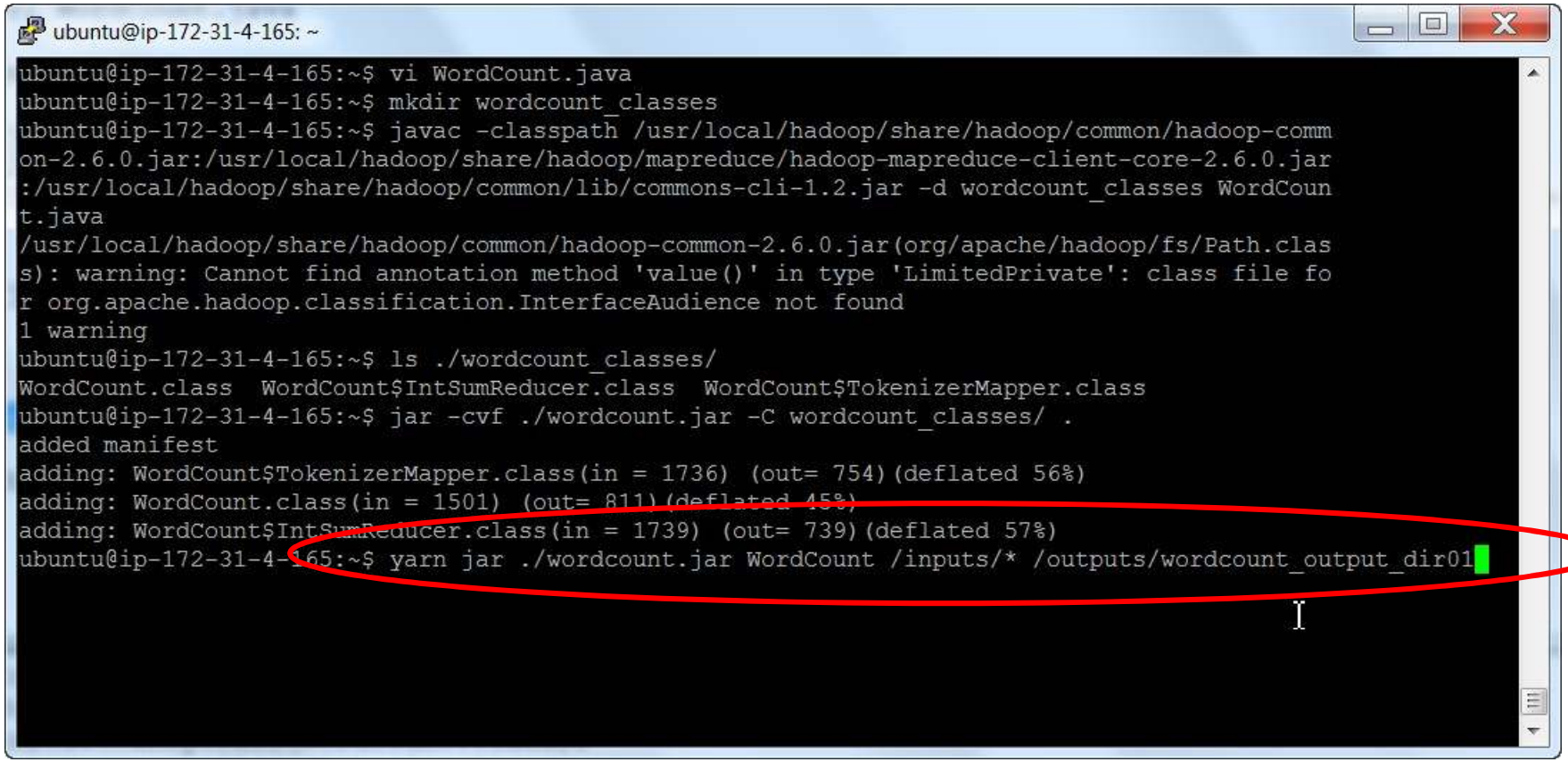
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



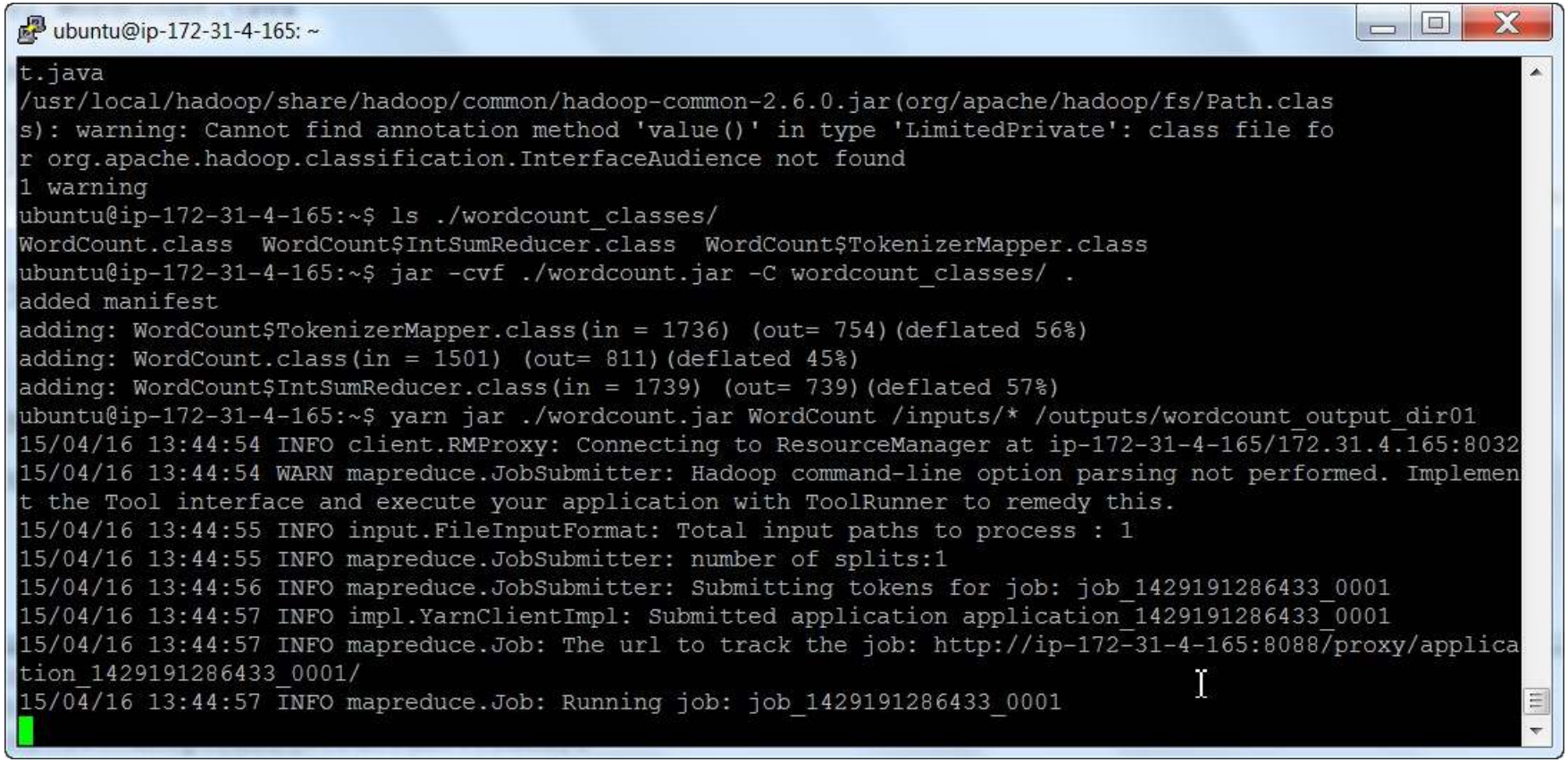
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



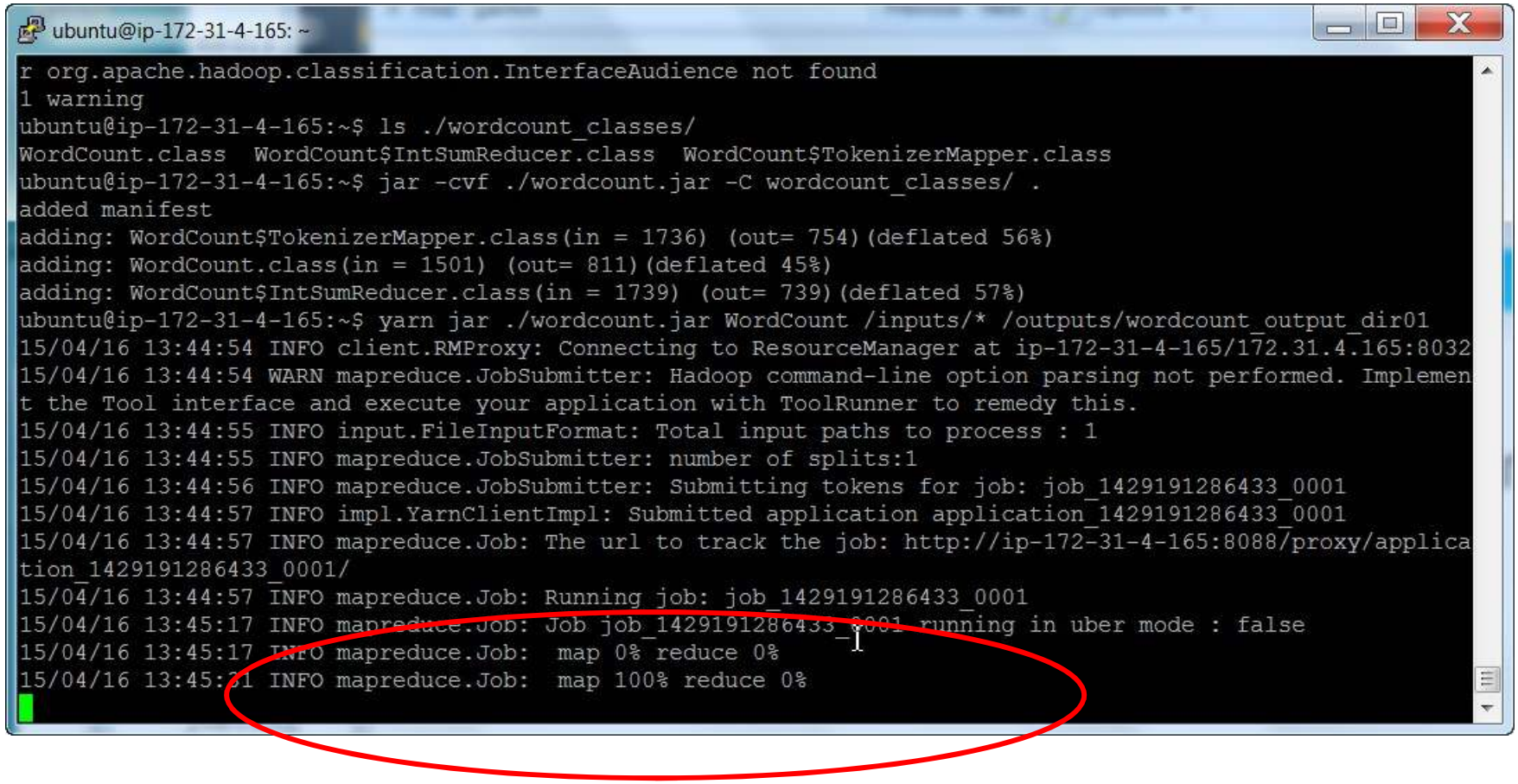
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



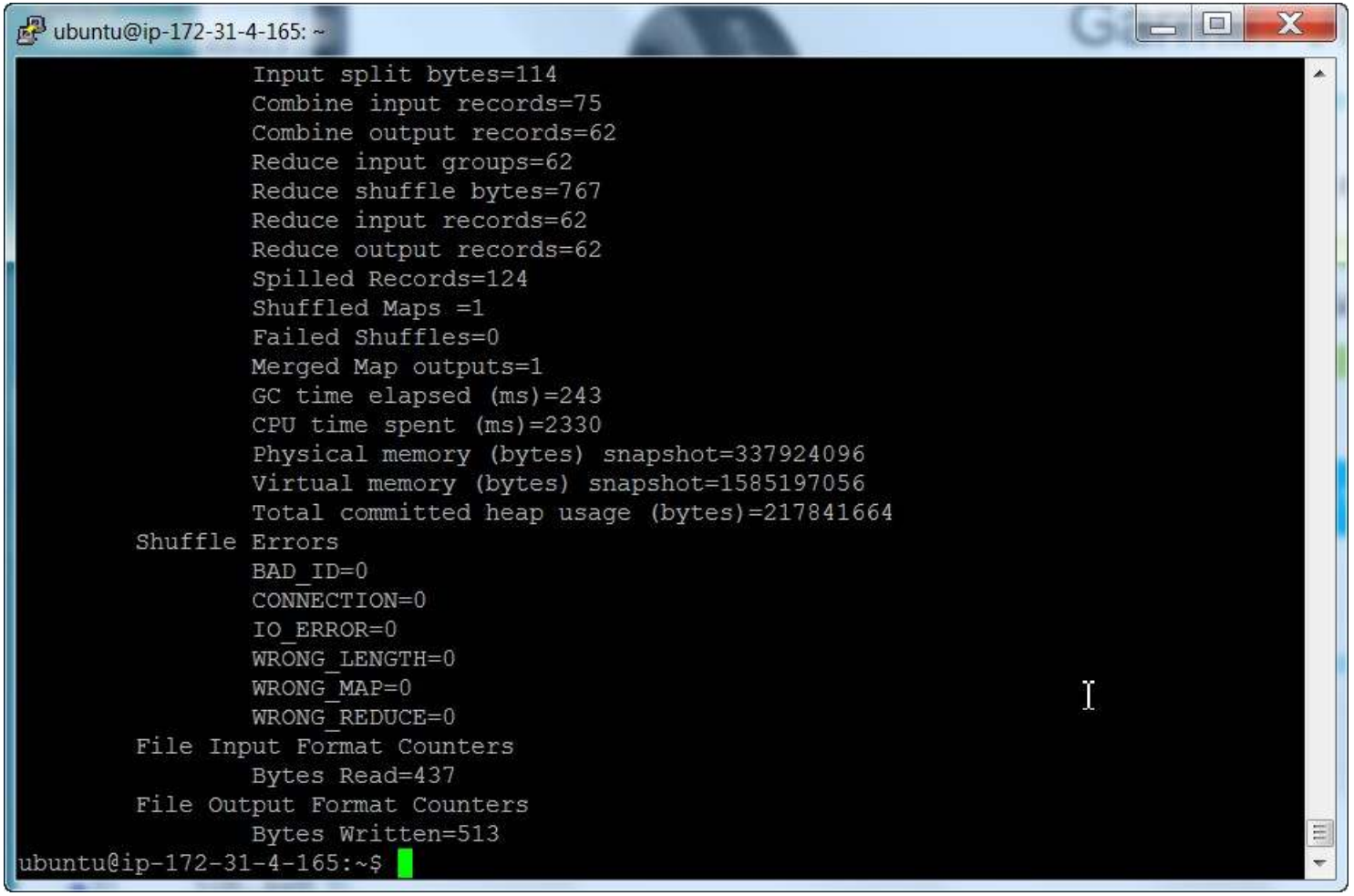
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



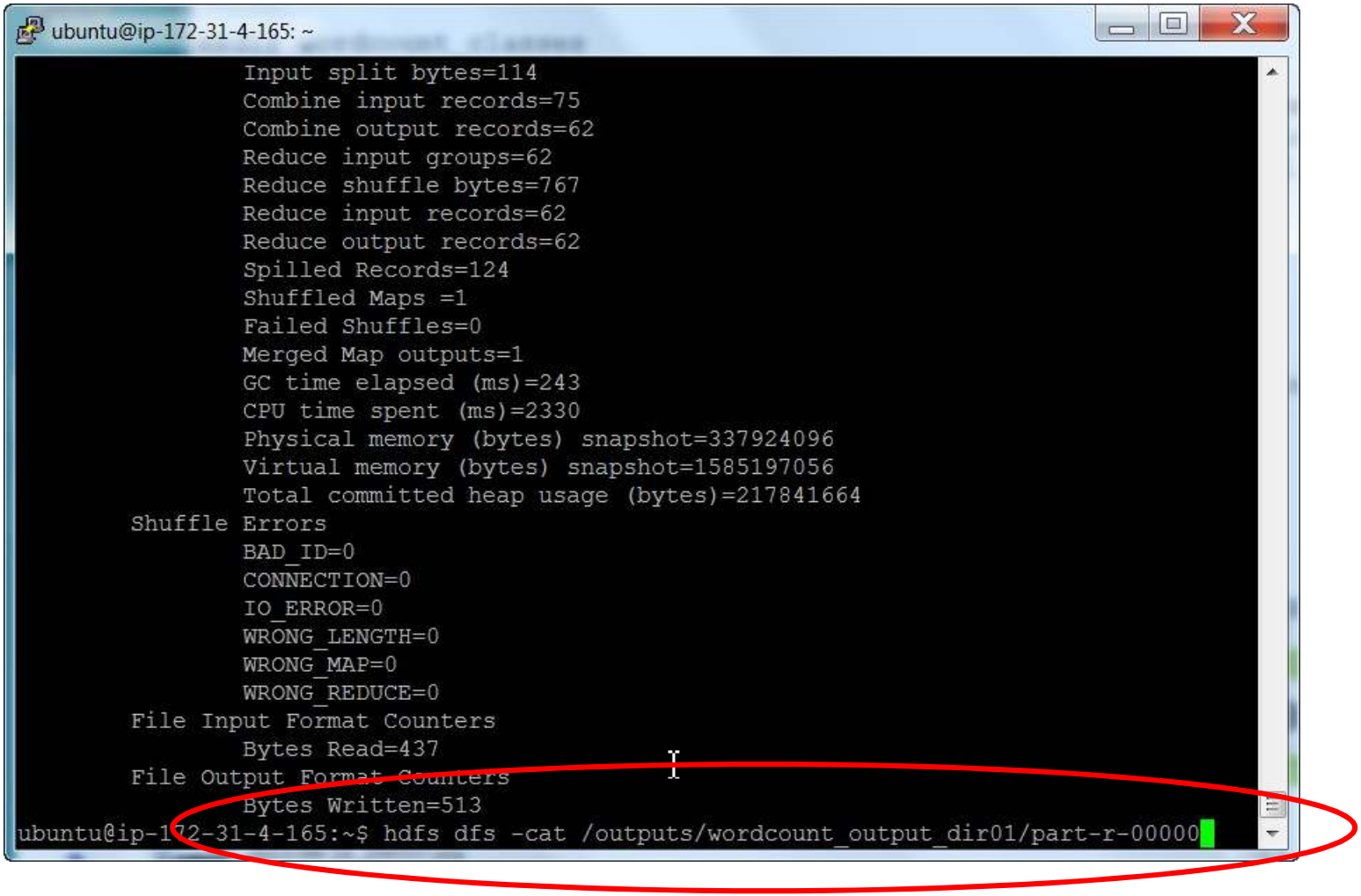
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



Packaging Map Reduce and Deploying to Hadoop Runtime Environment

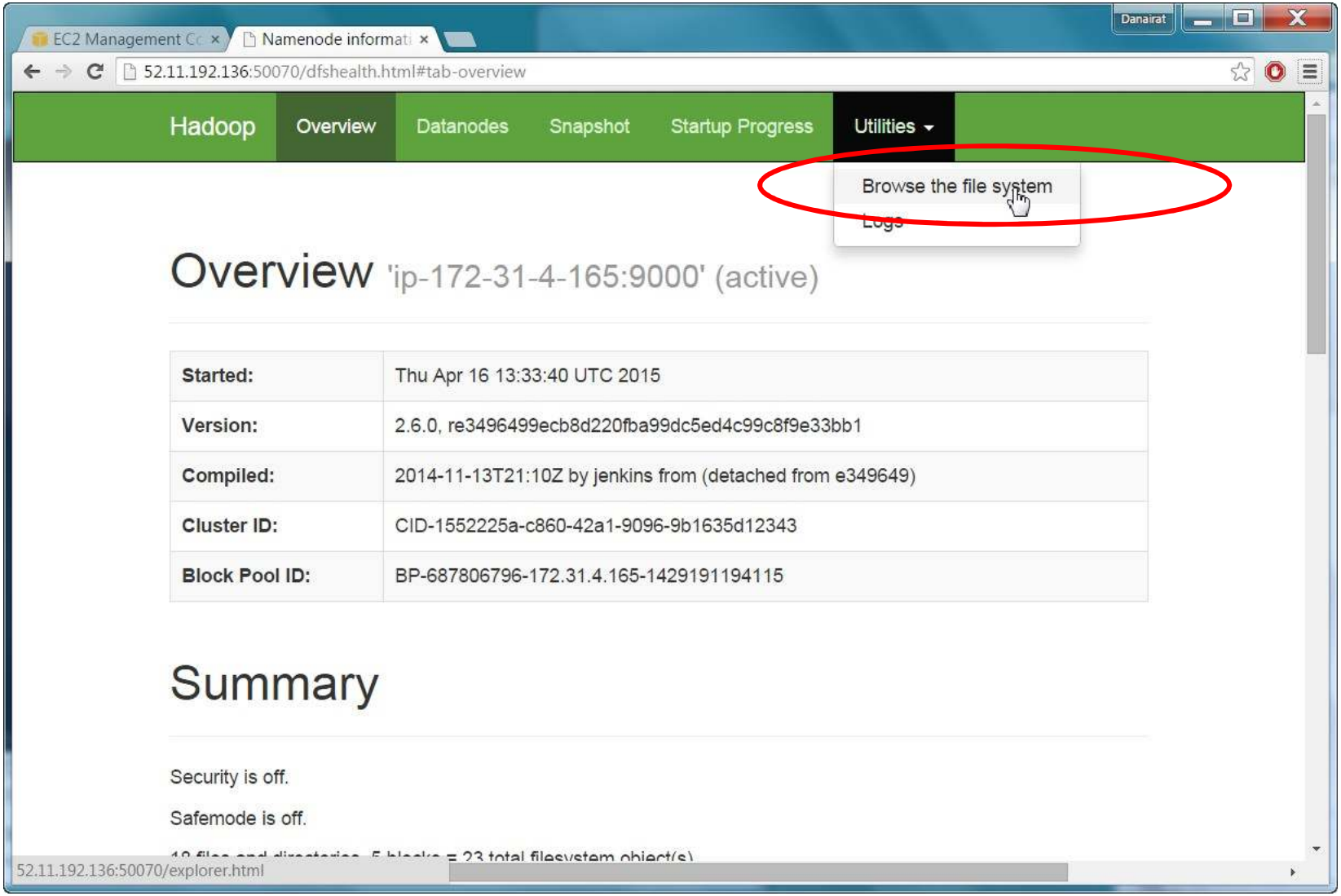


Packaging Map Reduce and Deploying to Hadoop Runtime Environment

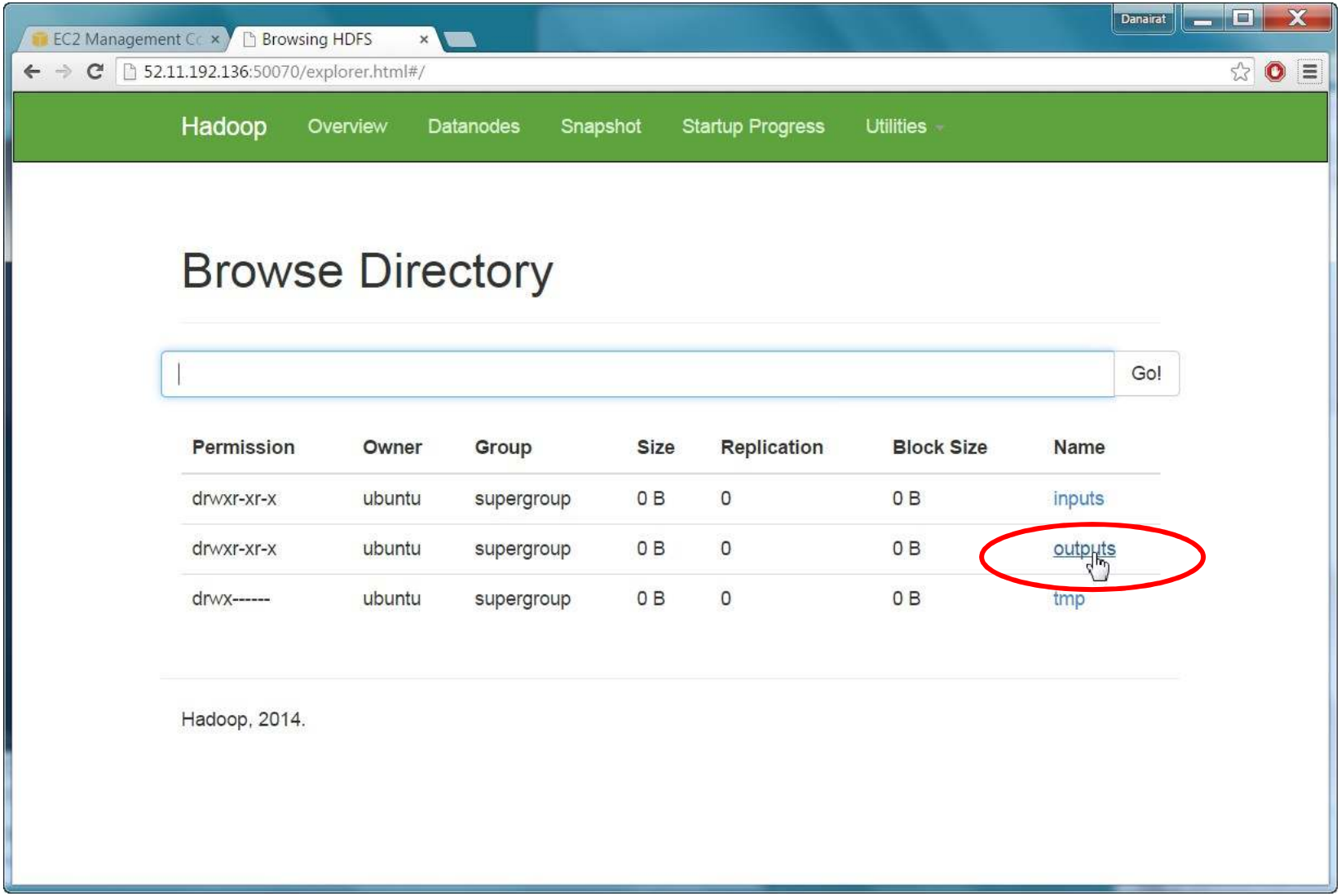


Packaging Map Reduce and Deploying to Hadoop Runtime Environment

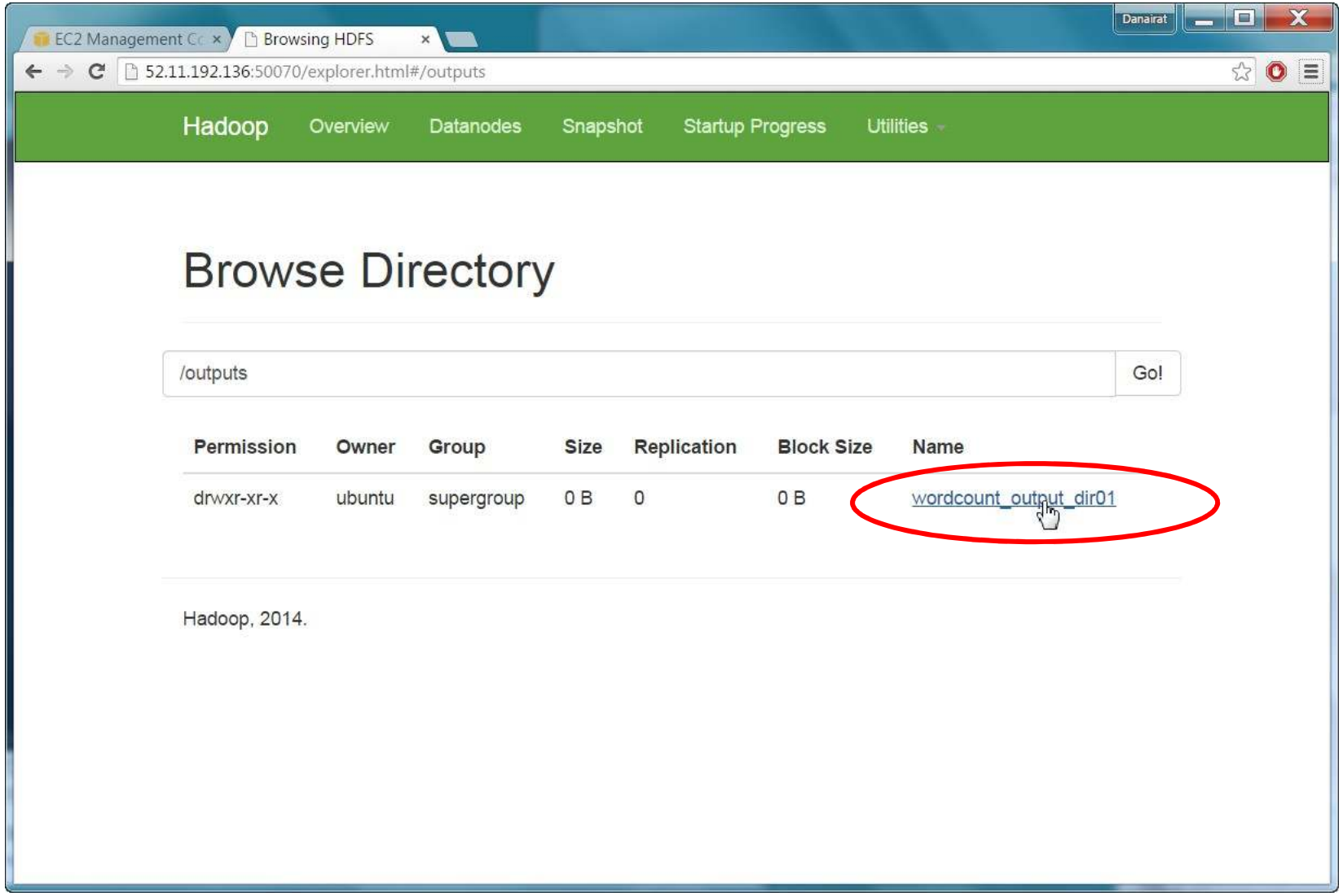
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



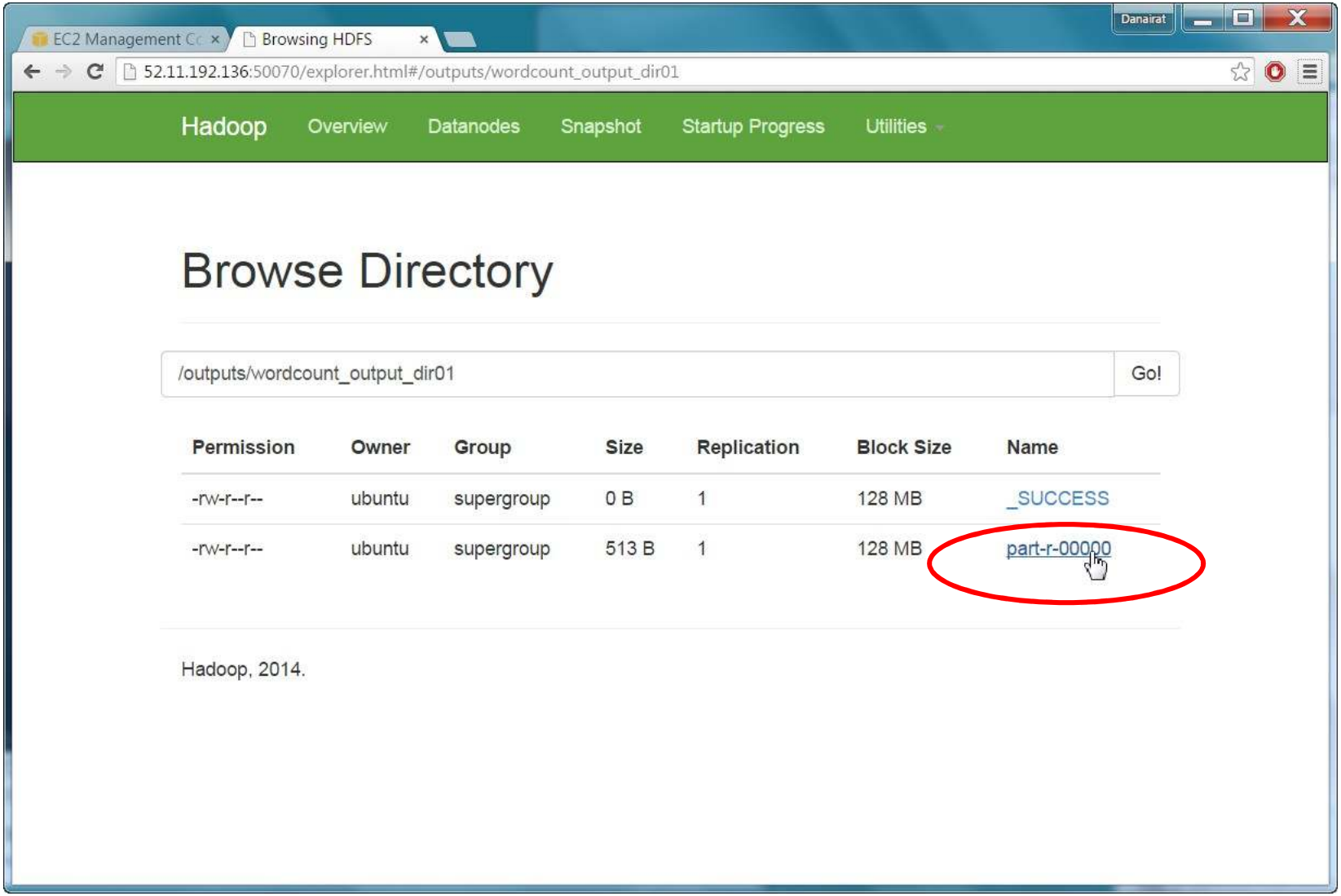
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



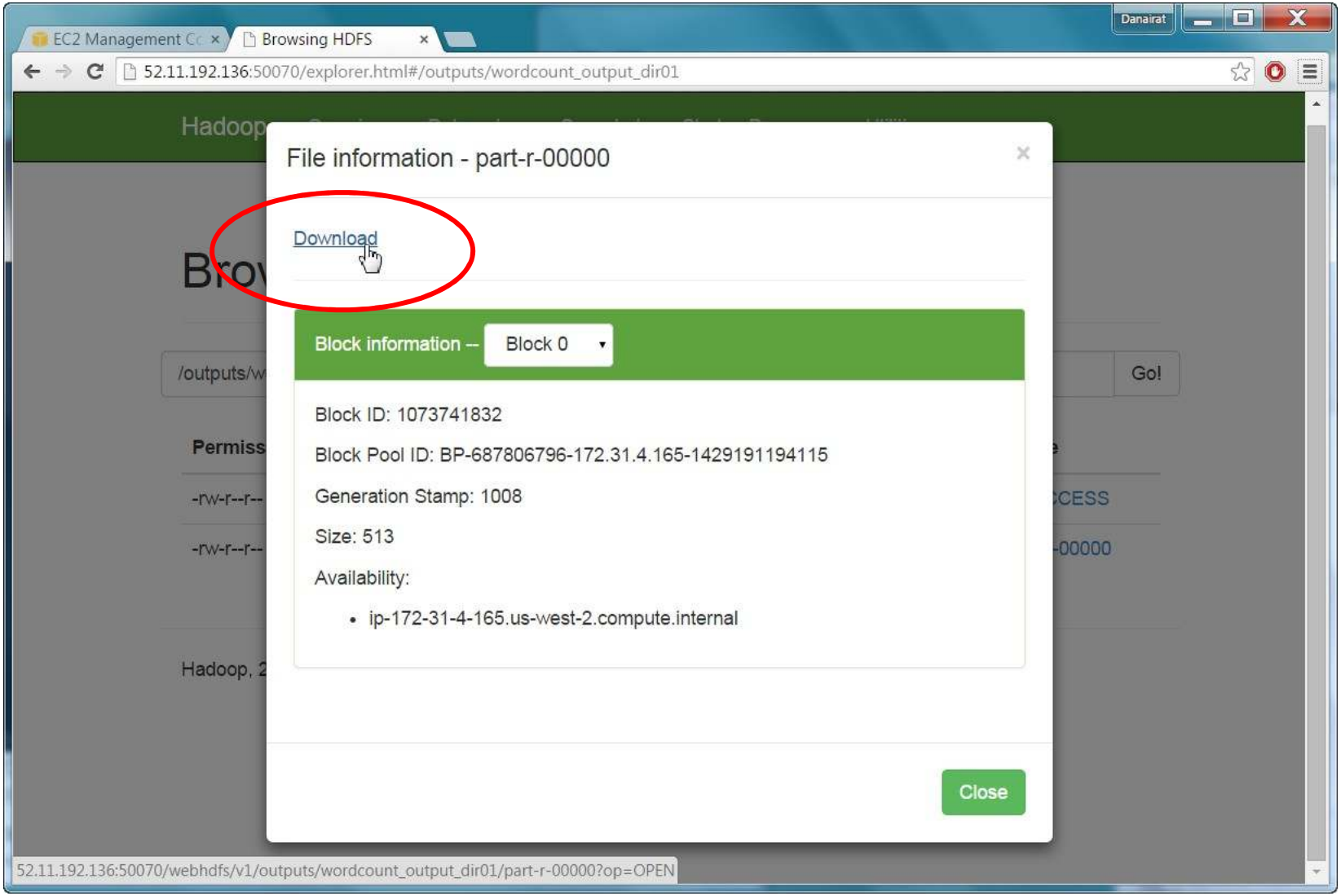
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



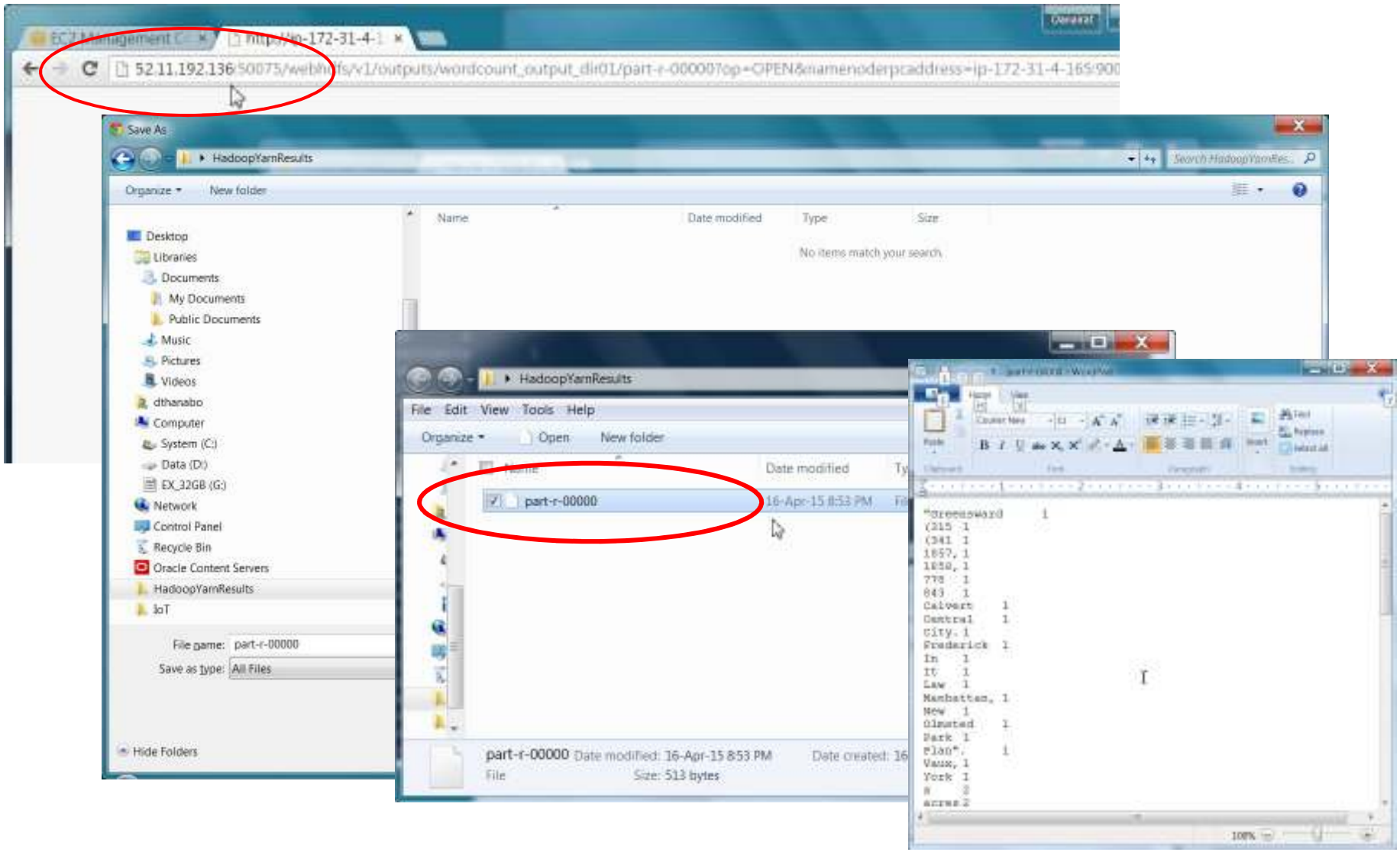
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



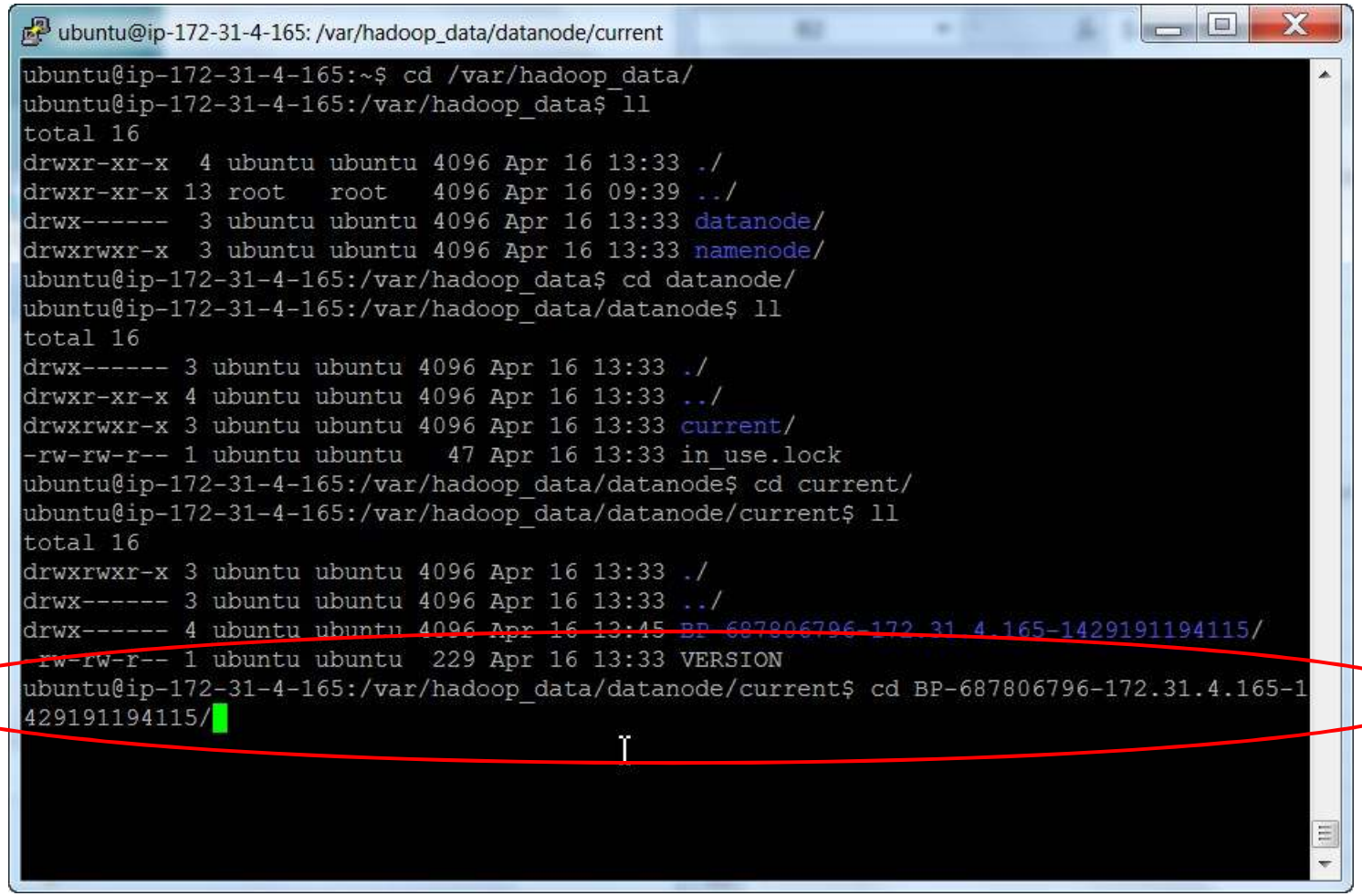
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



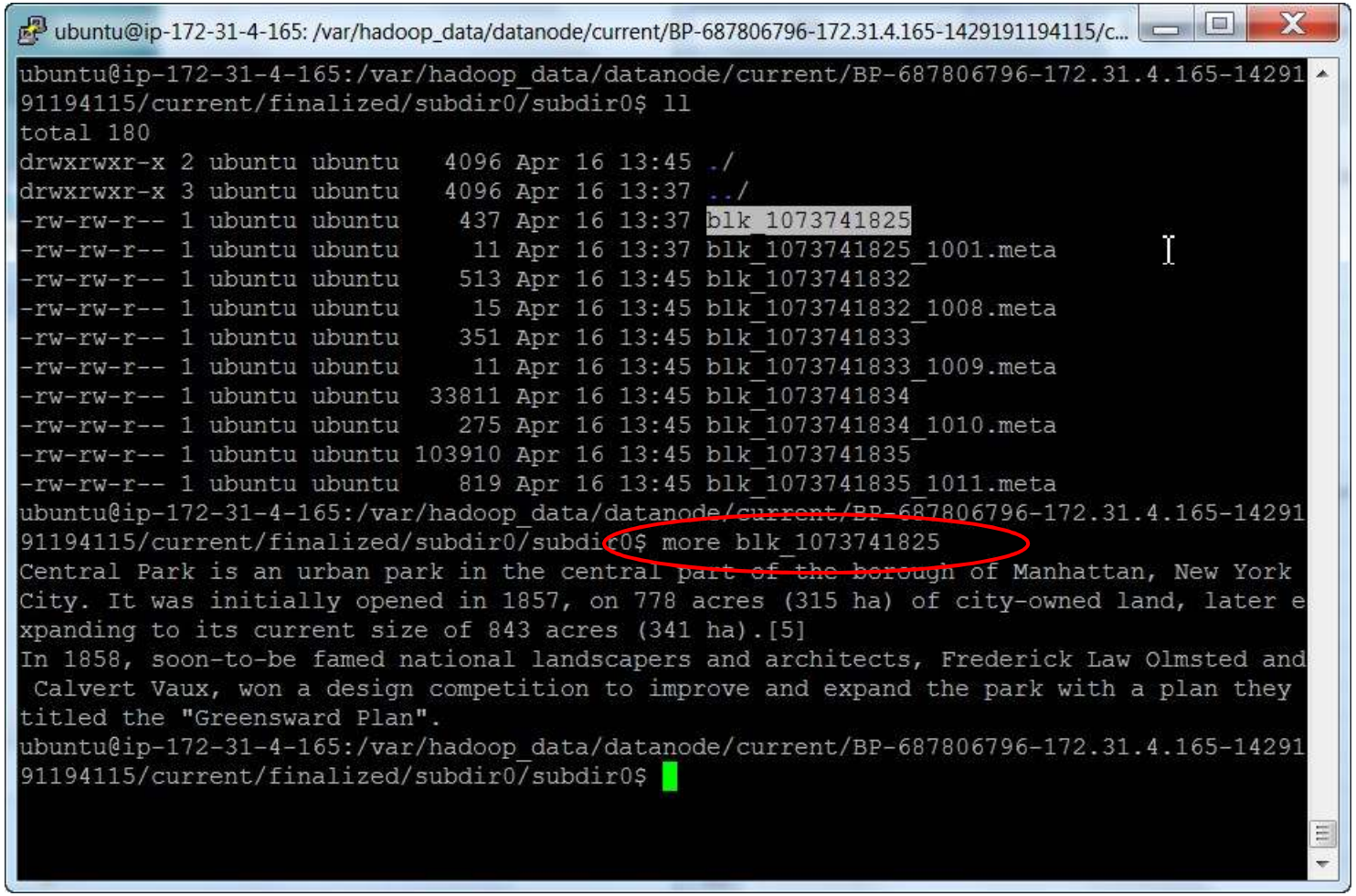
Packaging Map Reduce and Deploying to Hadoop Runtime Environment



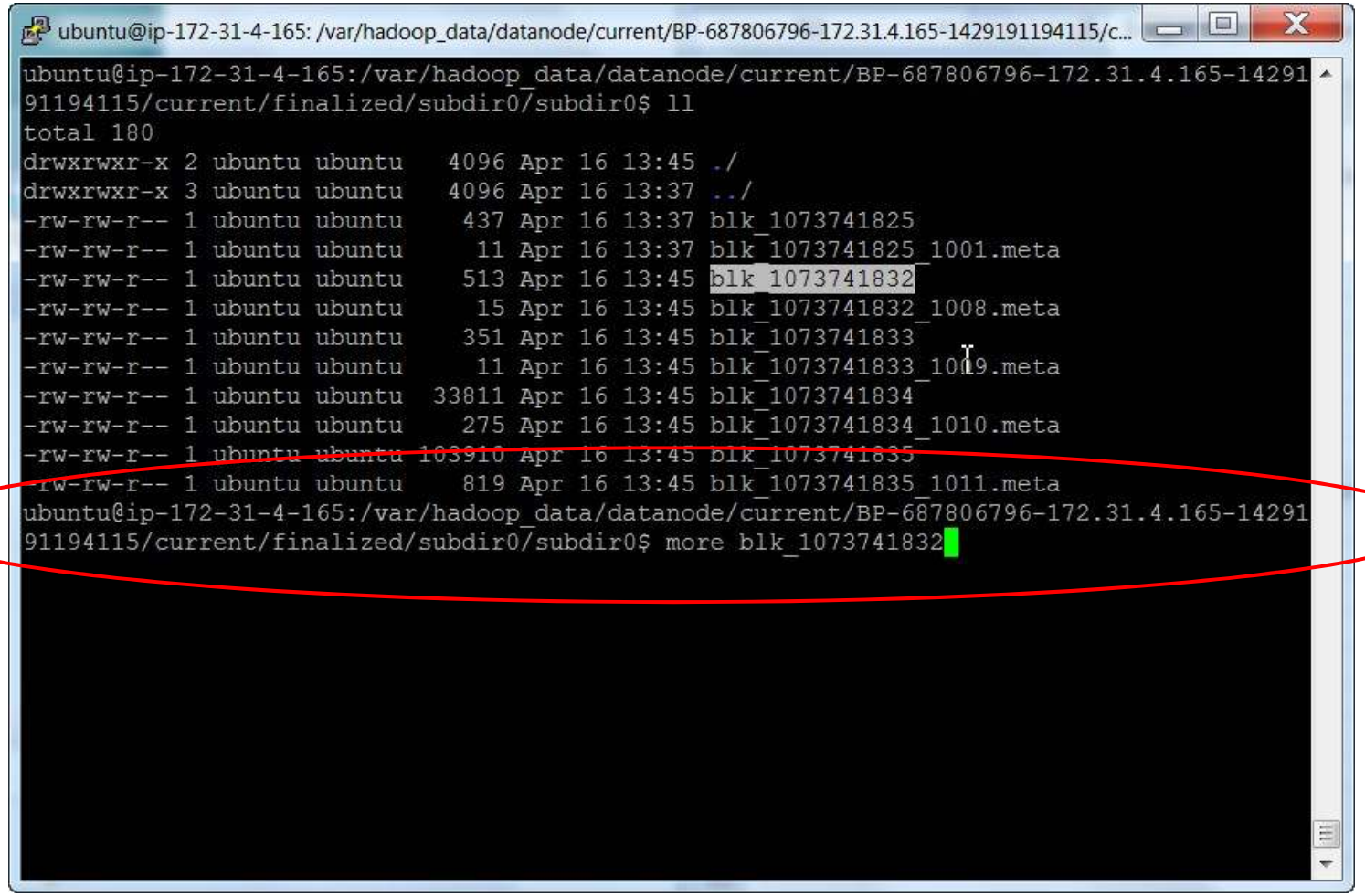
(Optional) Review OS File System



(Optional) Review OS File System



(Optional) Review OS File System



Lecture
Understanding Hive

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

Introduction

A Petabyte Scale Data Warehouse Using Hadoop

Hive is developed by Facebook, designed to enable easy data summarization, ad-hoc querying and analysis of large volumes of data. It provides a simple query language called Hive QL, which is based on SQL

What Hive is NOT

Hive is not designed for online transaction processing and does not offer real-time queries and row level updates. It is best used for batch jobs over large sets of immutable data (like web logs, etc.).

Sample HiveQL

The Query compiler uses the information stored in the metastore to convert SQL queries into a sequence of map/reduce jobs, e.g. the following query

SELECT * FROM t where t.c = 'xyz'

SELECT t1.c2 FROM t1 JOIN t2 ON (t1.c1 = t2.c1)

SELECT t1.c1, count(1) from t1 group by t1.c1

Hive.apache.org

Sample HiveQL

- **Aggregation**
  - Select count (DISTINCT category) from txnrecords;
- **Grouping**
  - Select category, sum(amount) from txnrecords group by category

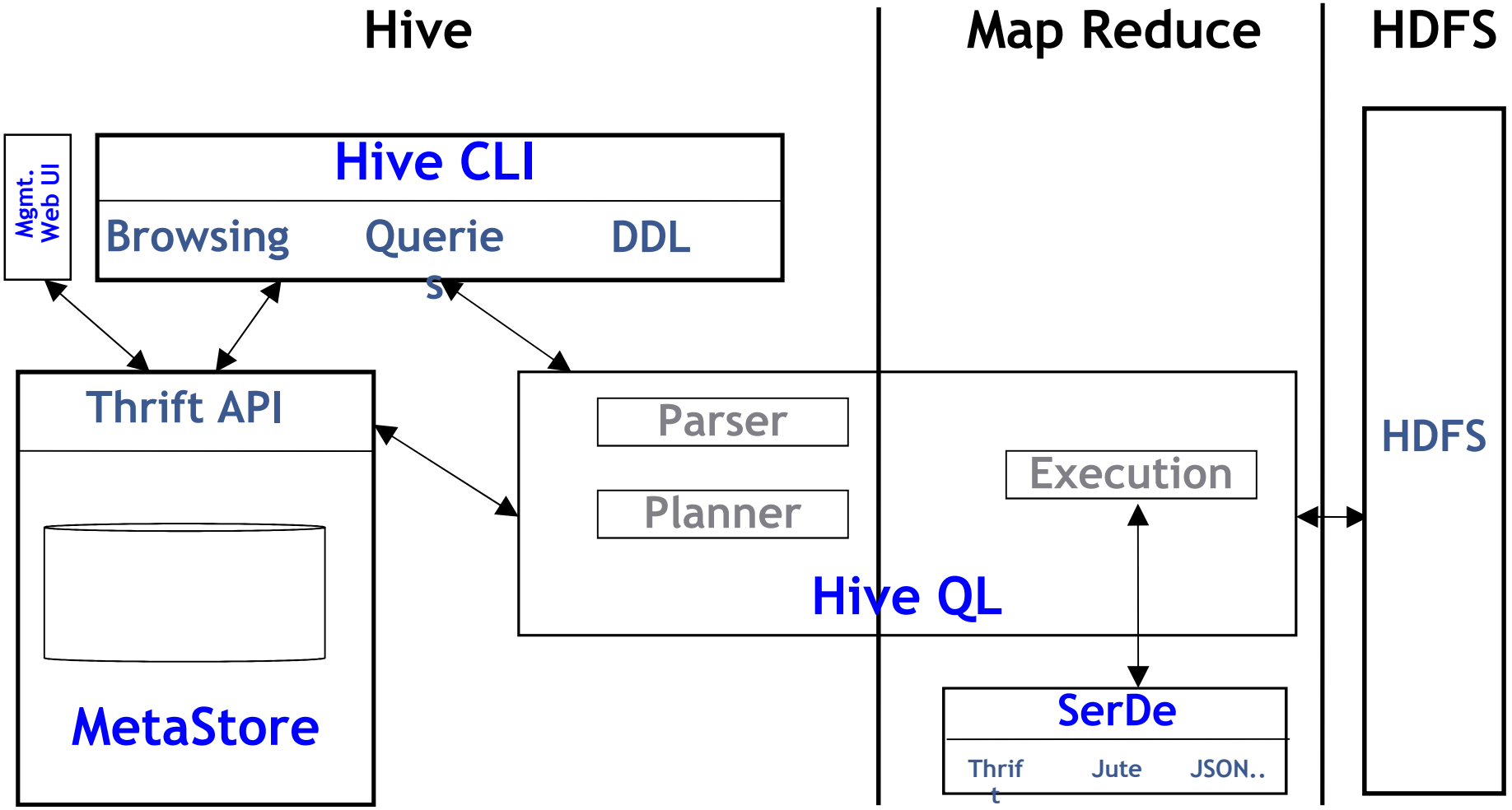
- **Inserting Output into another table**
  - INSERT OVERWRITE TABLE results (SELECT * from txnrecords)
- **Inserting into local file**
  - INSERT OVERWRITE LOCAL DIRECTORY'tmp/results' (SELECT * from txnrecords)

System Architecture and Components

- **Metastore: To store the meta data.**
- **Query compiler and execution engine: To convert SQL queries to a sequence of map/reduce jobs that are then executed on Hadoop.**
- **SerDe and ObjectInspectors: Programmable interfaces and implementations of common data formats and types.**
  A SerDe is a combination of a Serializer and a Deserializer (hence, Ser-De). The Deserializer interface takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate. The Serializer, however, will take a Java object that Hive has been working with, and turn it into something that Hive can write to HDFS or another supported system.
- **UDF and UDAF: Programmable interfaces and implementations for user defined functions (scalar and aggregate functions).**
- **Clients: Command line client similar to Mysql command line.**

hive.apache.org

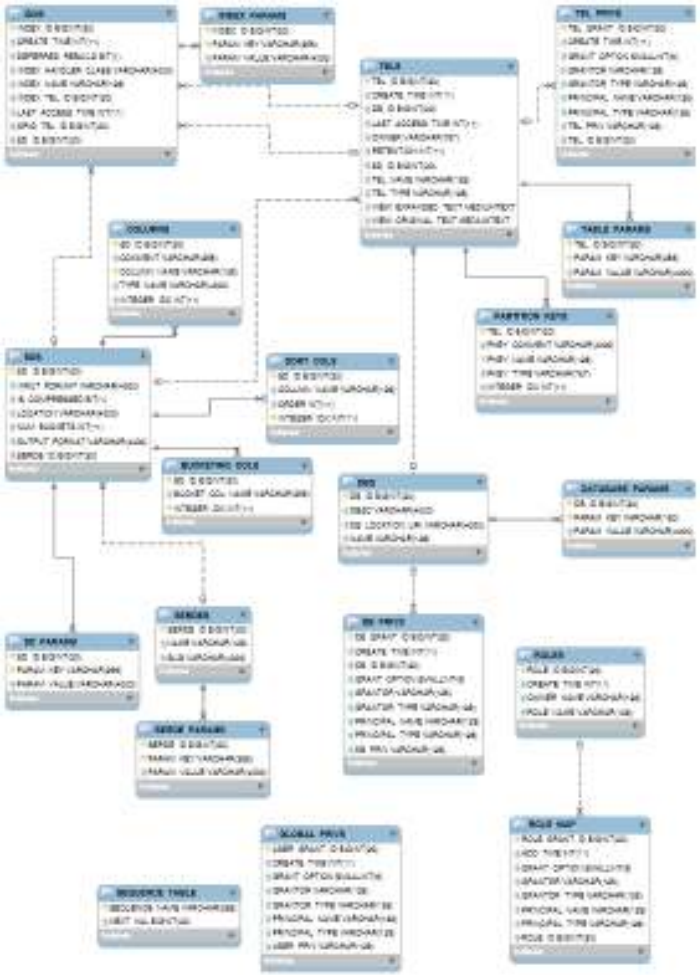
Architecture Overview



Hive.apache.org

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

Hive Metastore



Hive Metastore is a repository to keep all Hive metadata; Tables and Partitions definition.

By default, Hive will store its metadata in Derby DB

Hive Built in Functions

Return Type	Function Name (Signature)	Description
BIGINT	round(double a)	returns the rounded BIGINT value of the double
BIGINT	floor(double a)	returns the maximum BIGINT value that is equal or less than the double
BIGINT	ceil(double a)	returns the minimum BIGINT value that is equal or greater than the double
double	rand(), rand(int seed)	returns a random number (that changes from row to row). Specifiying the seed will make sure the generated random number sequence is deterministic.
string	concat(string A, string B,...)	returns the string resulting from concatenating B after A. For example, concat('foo', 'bar') results in 'foobar'. This function accepts arbitrary number of arguments and return the concatenation of all of them.
string	substr(string A, int start)	returns the substring of A starting from start position till the end of string A. For example, substr('foobar', 4) results in 'bar'
string	substr(string A, int start, int length)	returns the substring of A starting from start position with the given length e.g. substr('foobar', 4, 2) results in 'ba'
string	upper(string A)	returns the string resulting from converting all characters of A to upper case e.g. upper('fOoBaR') results in 'FOOBAR'
string	ucase(string A)	Same as upper
string	lower(string A)	returns the string resulting from converting all characters of B to lower case e.g. lower('fOoBaR') results in 'foobar'
string	lcase(string A)	Same as lower
string	trim(string A)	returns the string resulting from trimming spaces from both ends of A e.g. trim(' foobar ') results in 'foobar'
string	ltrim(string A)	returns the string resulting from trimming spaces from the beginning(left hand side) of A. For example, ltrim(' foobar ') results in 'foobar '
string	rtrim(string A)	returns the string resulting from trimming spaces from the end(right hand side) of A. For example, rtrim(' foobar ') results in ' foobar'
string	regexp_replace(string A, string B, string C)	returns the string resulting from replacing all substrings in B that match the Java regular expression syntax(See Java regular expressions syntax) with C. For example, regexp_replace('foobar', 'oo|ar',) returns 'fb'
string	from_unixtime(int unixtime)	convert the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00"
string	to_date(string timestamp)	Return the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01"
int	year(string date)	Return the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970
int	month(string date)	Return the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11
int	day(string date)	Return the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1
string	get_json_object(string json_string, string path)	Extract json object from a json string based on json path specified, and return json string of the extracted json object. It will return null if the input json string is invalid

hive.apache.org

Hive Aggregate Functions

Return Type	Aggregation Function Name (Signature)	Description
BIGINT	count(*), count(expr), count(DISTINCT expr[, expr_.])	count(*) - Returns the total number of retrieved rows, including rows containing NULL values; count(expr) - Returns the number of rows for which the supplied expression is non-NULL; count(DISTINCT expr[, expr]) - Returns the number of rows for which the supplied expression(s) are unique and non-NULL.
DOUBLE	sum(col), sum(DISTINCT col)	returns the sum of the elements in the group or the sum of the distinct values of the column in the group
DOUBLE	avg(col), avg(DISTINCT col)	returns the average of the elements in the group or the average of the distinct values of the column in the group
DOUBLE	min(col)	returns the minimum value of the column in the group
DOUBLE	max(col)	returns the maximum value of the column in the group

hive.apache.org

Running Hive

Hive Shell

- **Interactive**

  hive
- **Script**

  hive -f myscript
- **Inline**

  *hive -e 'SELECT * FROM mytable'*

Hive.apache.org

Hive Commands

Command Line

Function	Hive
Run query	`hive -e 'select a.col from tab1 a'`
Run query silent mode	`hive -S -e 'select a.col from tab1 a'`
Set hive config variables	`hive -e 'select a.col from tab1 a' -hiveconf hive.root.logger=DEBUG,console`
Use initialization script	`hive -i initialize.sql`
Run non-interactive script	`hive -f script.sql`

Hive Shell

Function	Hive
Run script inside shell	`source file_name`
Run ls (dfs) commands	`dfs -ls /user`
Run ls (bash command) from shell	`!ls`
Set configuration variables	`set mapred.reduce.tasks=32`
TAB auto completion	`set hive.<TAB>`
Show all variables starting with hive	`set`
Revert all variables	`reset`
Add jar to distributed cache	`add jar jar_path`
Show all jars in distributed cache	`list jars`
Delete jar from distributed cache	`delete jar jar_name`

Hive Tables

- Managed- CREATE TABLE
  - LOAD- File moved into Hive's data warehouse directory
  - DROP- Both data and metadata are deleted.
- External- CREATE EXTERNAL TABLE
  - LOAD- No file moved
  - DROP- Only metadata deleted
  - Use when sharing data between Hive and Hadoop applications or you want to use multiple schema on the same data

Hive External Table

```
– CREATE EXTERNAL TABLE external_Table (dummy STRING)
– LOCATION '/user/notroot/external_table';
```

Dropping External Table using Hive:-

- Hive will delete metadata from metastore
- Hive will NOT delete the HDFS file
- You need to manually delete the HDFS file

Java JDBC for Hive

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;

public class HiveJdbcClient {
  private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

  public static void main(String[] args) throws SQLException {
    try {
      Class.forName(driverName);
    } catch (ClassNotFoundException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
      System.exit(1);
    }
    Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/default", "", "");
    Statement stmt = con.createStatement();
    String tableName = "testHiveDriverTable";
    stmt.executeQuery("drop table " + tableName);
    ResultSet res = stmt.executeQuery("create table " + tableName + " (key int, value string)");
    // show tables
    String sql = "show tables '" + tableName + "'";
    System.out.println("Running: " + sql);
    res = stmt.executeQuery(sql);
    if (res.next()) {
      System.out.println(res.getString(1));
    }
    // describe table
    sql = "describe " + tableName;
    System.out.println("Running: " + sql);
    res = stmt.executeQuery(sql);
    while (res.next()) {
      System.out.println(res.getString(1) + "\t" + res.getString(2));
    }
```

Java JDBC for Hive

```
    // load data into table
    // NOTE: filepath has to be local to the hive server
    // NOTE: /tmp/a.txt is a ctrl-A separated file with two fields per line
    String filepath = "/tmp/a.txt";
    sql = "load data local inpath '" + filepath + "' into table " + tableName;
    System.out.println("Running: " + sql);
    res = stmt.executeQuery(sql);

    // select * query
    sql = "select * from " + tableName;
    System.out.println("Running: " + sql);
    res = stmt.executeQuery(sql);
    while (res.next()) {
      System.out.println(String.valueOf(res.getInt(1)) + "\t" + res.getString(2));
    }

    // regular hive query
    sql = "select count(1) from " + tableName;
    System.out.println("Running: " + sql);
    res = stmt.executeQuery(sql);
    while (res.next()) {
      System.out.println(res.getString(1));
    }
  }
}
```

HiveQL and MySQL Comparison

Metadata

Function	MySQL	HiveQL
Selecting a database	`USE database;`	`USE database;`
Listing databases	`SHOW DATABASES;`	`SHOW DATABASES;`
Listing tables in a database	`SHOW TABLES;`	`SHOW TABLES;`
Describing the format of a table	`DESCRIBE table;`	`DESCRIBE (FORMATTED|EXTENDED) table;`
Creating a database	`CREATE DATABASE db_name;`	`CREATE DATABASE db_name;`
Dropping a database	`DROP DATABASE db_name;`	`DROP DATABASE db_name (CASCADE);`

HiveQL and MySQL Query Comparison

Query

Function	MySQL	HiveQL
Retrieving information	`SELECT from_columns FROM table WHERE conditions;`	`SELECT from_columns FROM table WHERE conditions;`
All values	`SELECT * FROM table;`	`SELECT * FROM table;`
Some values	`SELECT * FROM table WHERE rec_name = "value";`	`SELECT * FROM table WHERE rec_name = "value";`
Multiple criteria	`SELECT * FROM table WHERE rec1="value1" AND rec2="value2";`	`SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";`
Selecting specific columns	`SELECT column_name FROM table;`	`SELECT column_name FROM table;`
Retrieving unique output records	`SELECT DISTINCT column_name FROM table;`	`SELECT DISTINCT column_name FROM table;`
Sorting	`SELECT col1, col2 FROM table ORDER BY col2;`	`SELECT col1, col2 FROM table ORDER BY col2;`
Sorting backward	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`
Counting rows	`SELECT COUNT(*) FROM table;`	`SELECT COUNT(*) FROM table;`
Grouping with counting	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`
Maximum value	`SELECT MAX(col_name) AS label FROM table;`	`SELECT MAX(col_name) AS label FROM table;`
Selecting from multiple tables (Join same table using alias w/"AS")	`SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;`	`SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);`

Hands-On: Creating Table and Retrieving Data using Hive

Hive Hands-On Labs

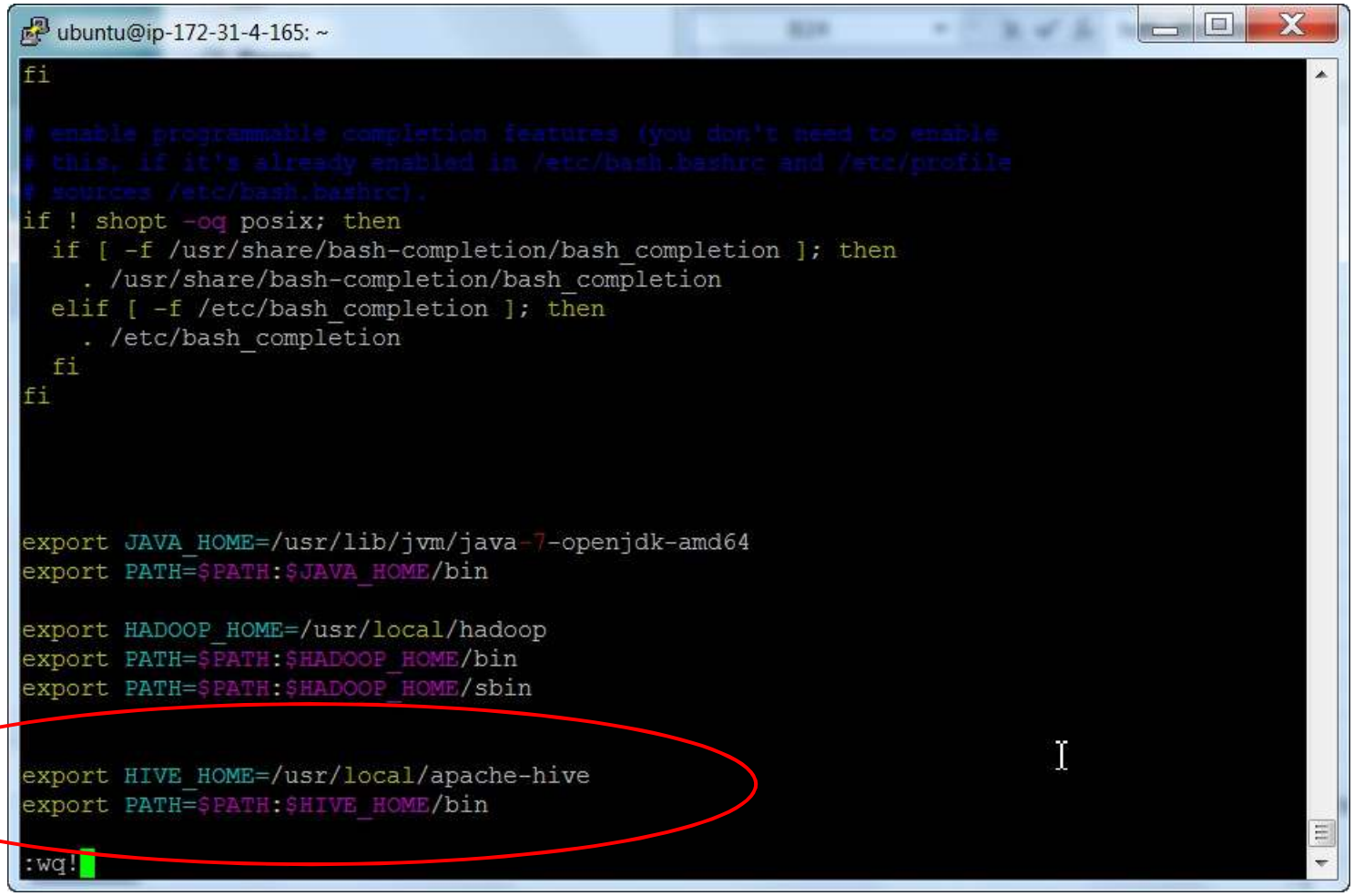
1. Installing Hive
2. Configuring / Starting Hive
3. Creating Hive Table
4. Reviewing Hive Table in HDFS
5. Alter and Drop Hive Table
6. Loading Data to Hive Table
7. Querying Data from Hive Table
8. Reviewing Hive Table Content from HDFS Command and WebUI

1. Installing Hive

```
$ wget http://www.us.apache.org/dist/hive/hive-
1.0.0/apache-hive-1.0.0-bin.tar.gz# tar -xvzf apache-hive-
1.1.0-bin.tar.gz

$ tar -xvf apache-hive-1.0.0-bin.tar.gz

$ mv apache-hive-1.0.0-bin apache-hive

$ sudo mv ./apache-hive /usr/local/

$ chown -R ubuntu:ubuntu /usr/local/apache-hive
```

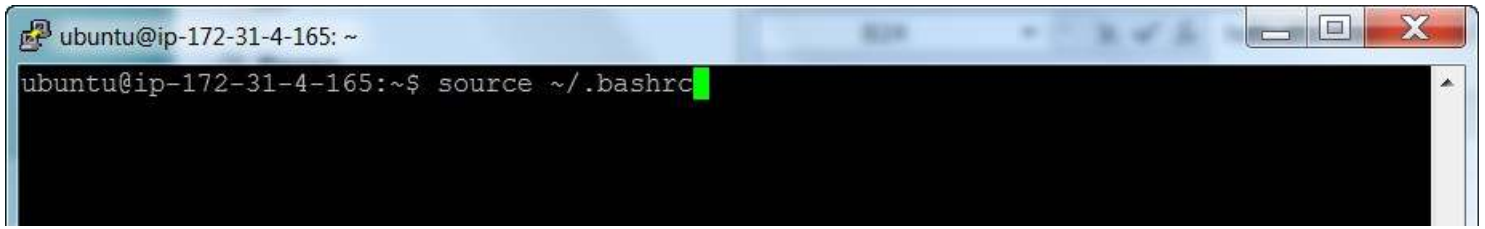
1. Installing Hive
Edit $HOME ./bashrc

```
$ vi /home/ubuntu/.bashrc
```

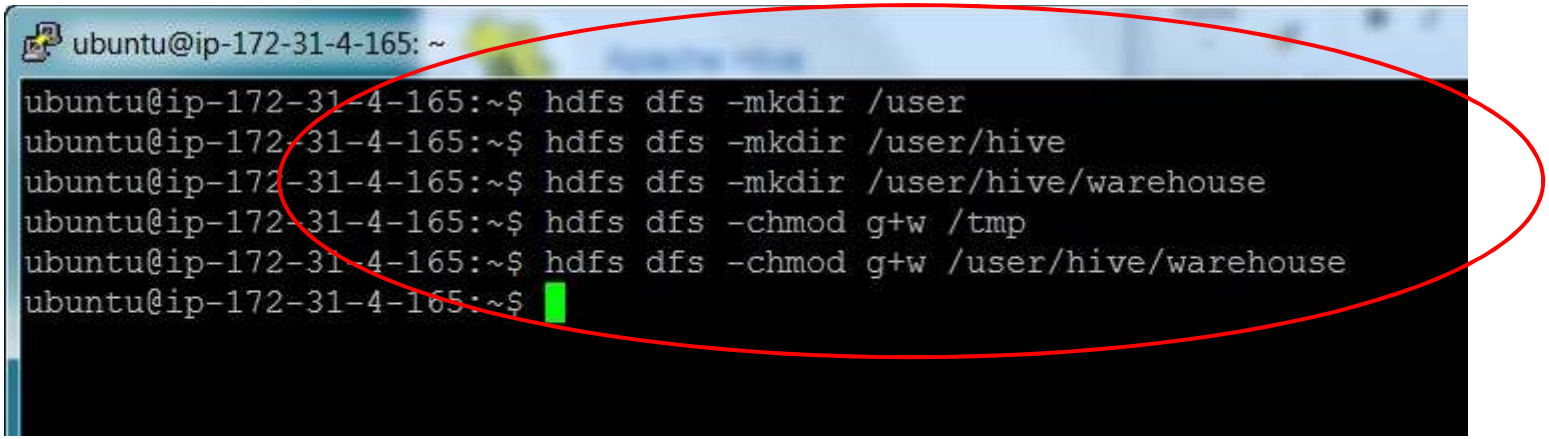


1. Installing Hive

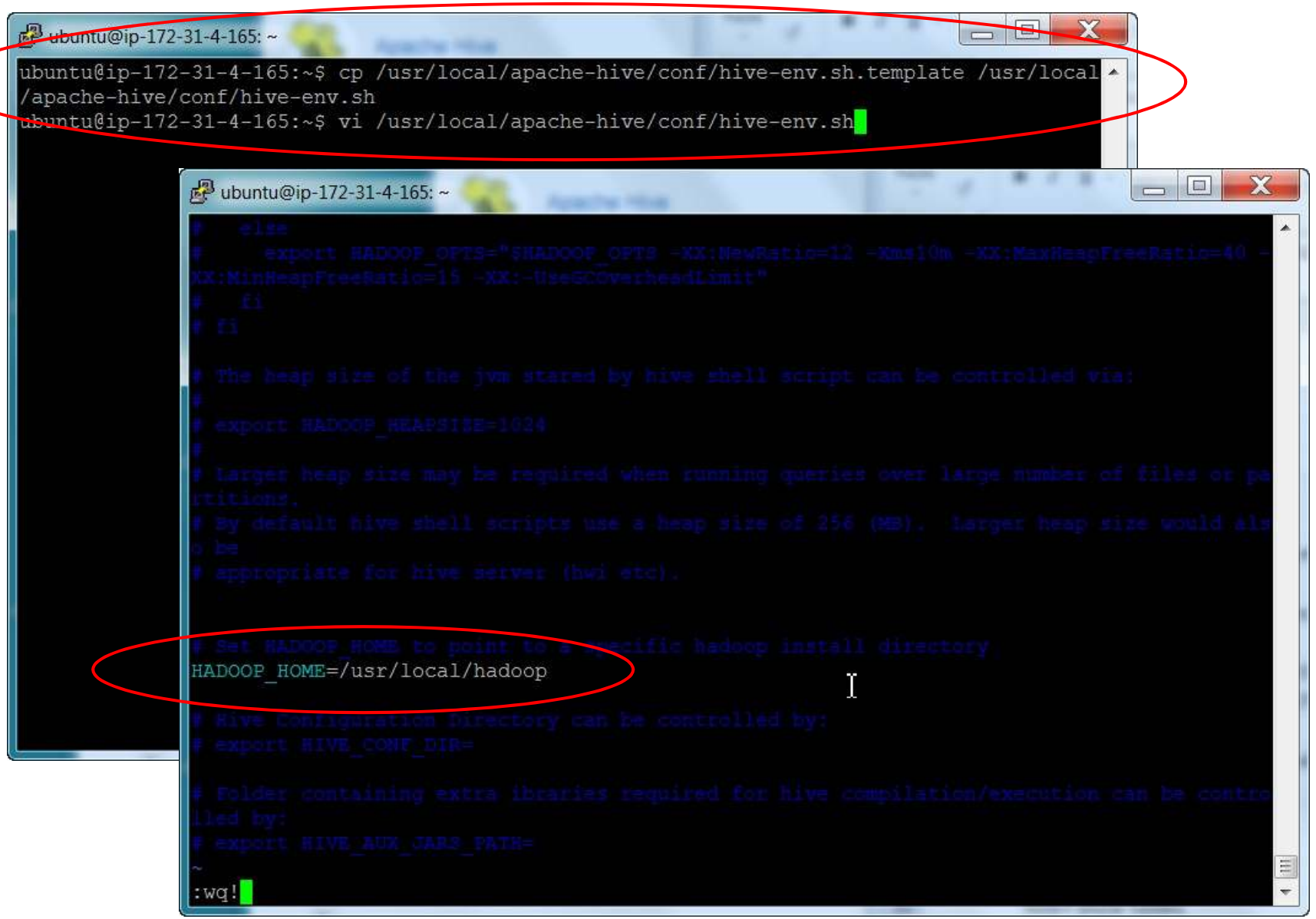
Execute environment variable



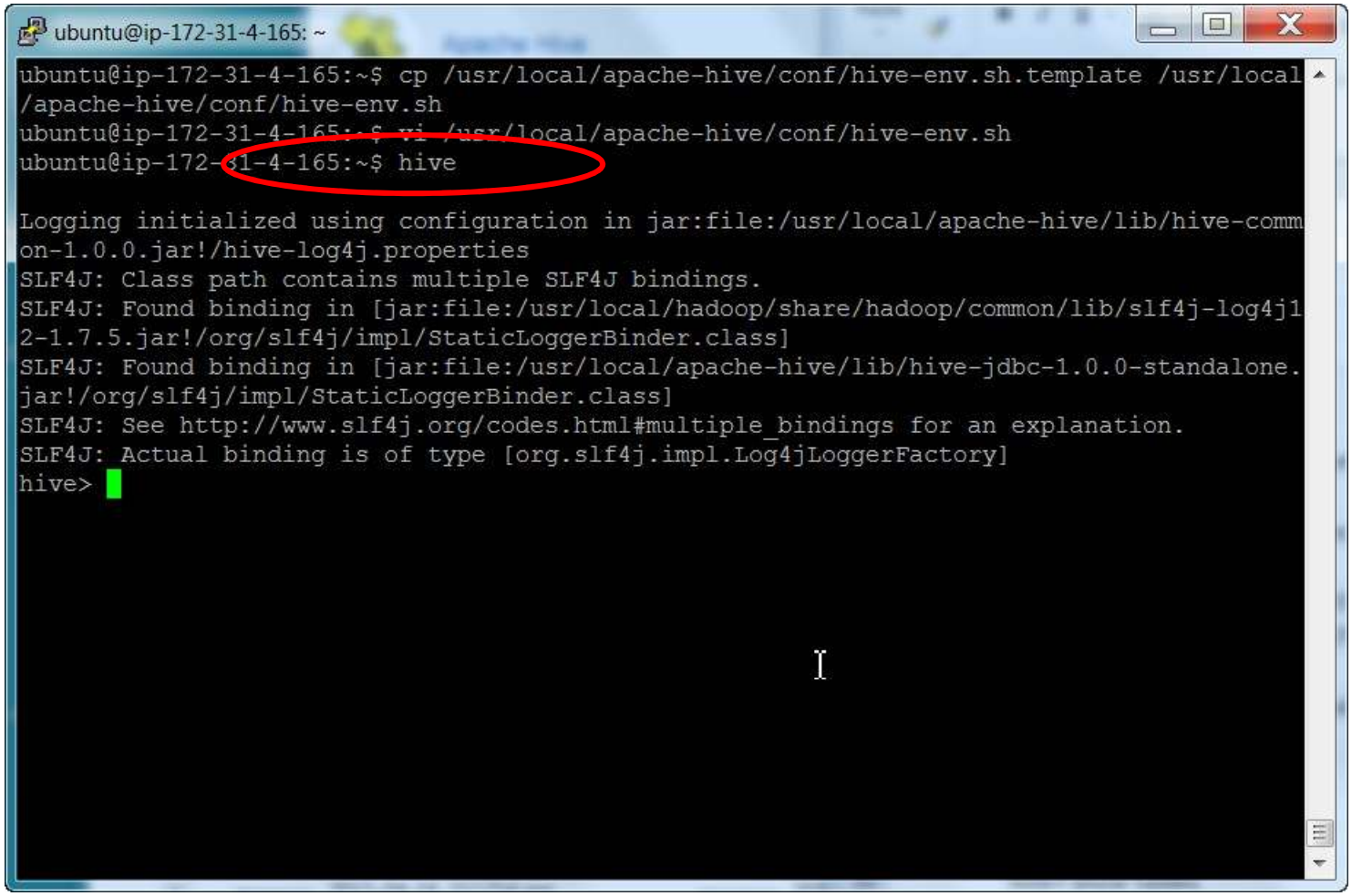
Create hive directories



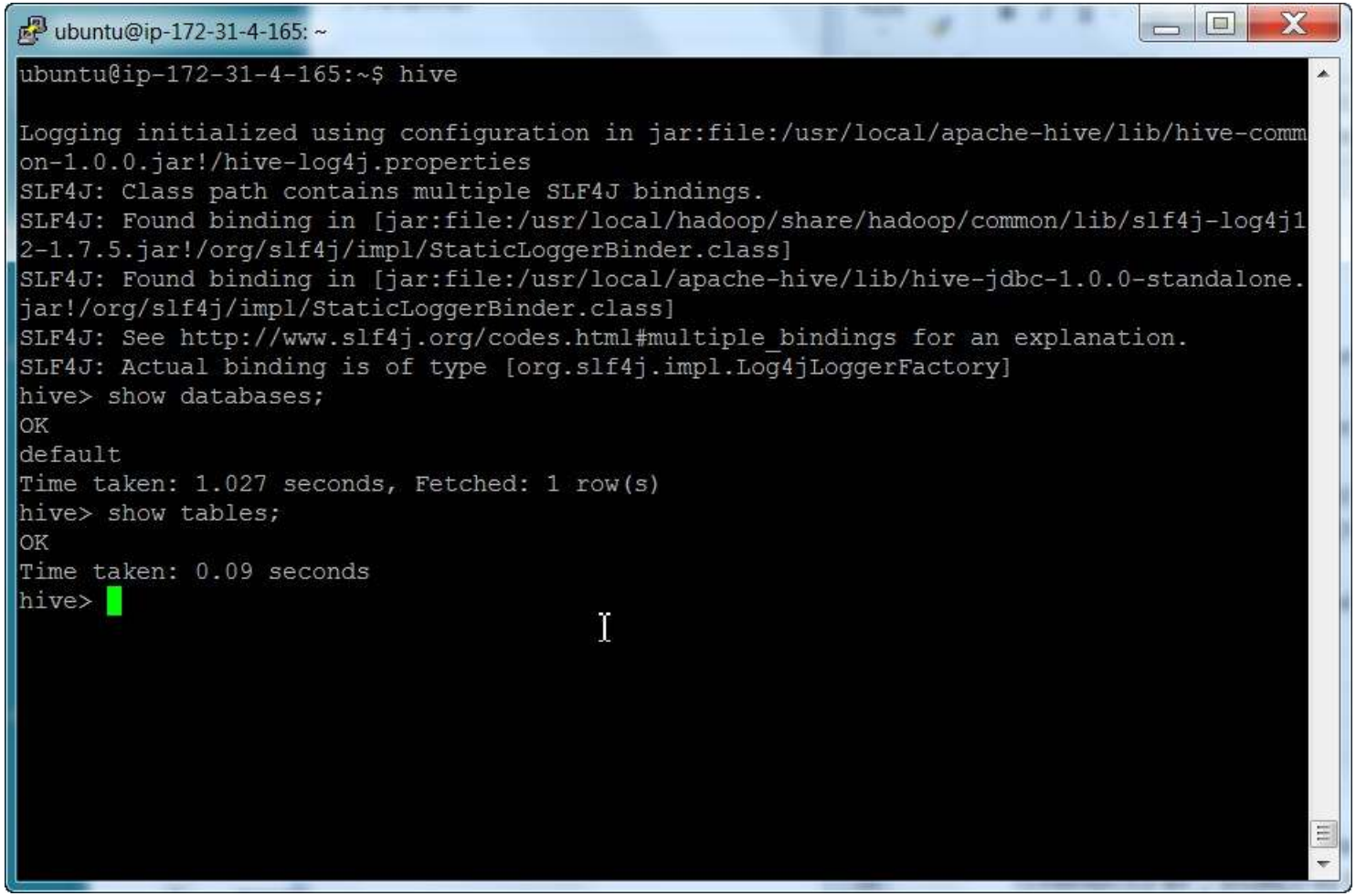
2. Configuring Hive



2. Starting Hive



Show Hive Table

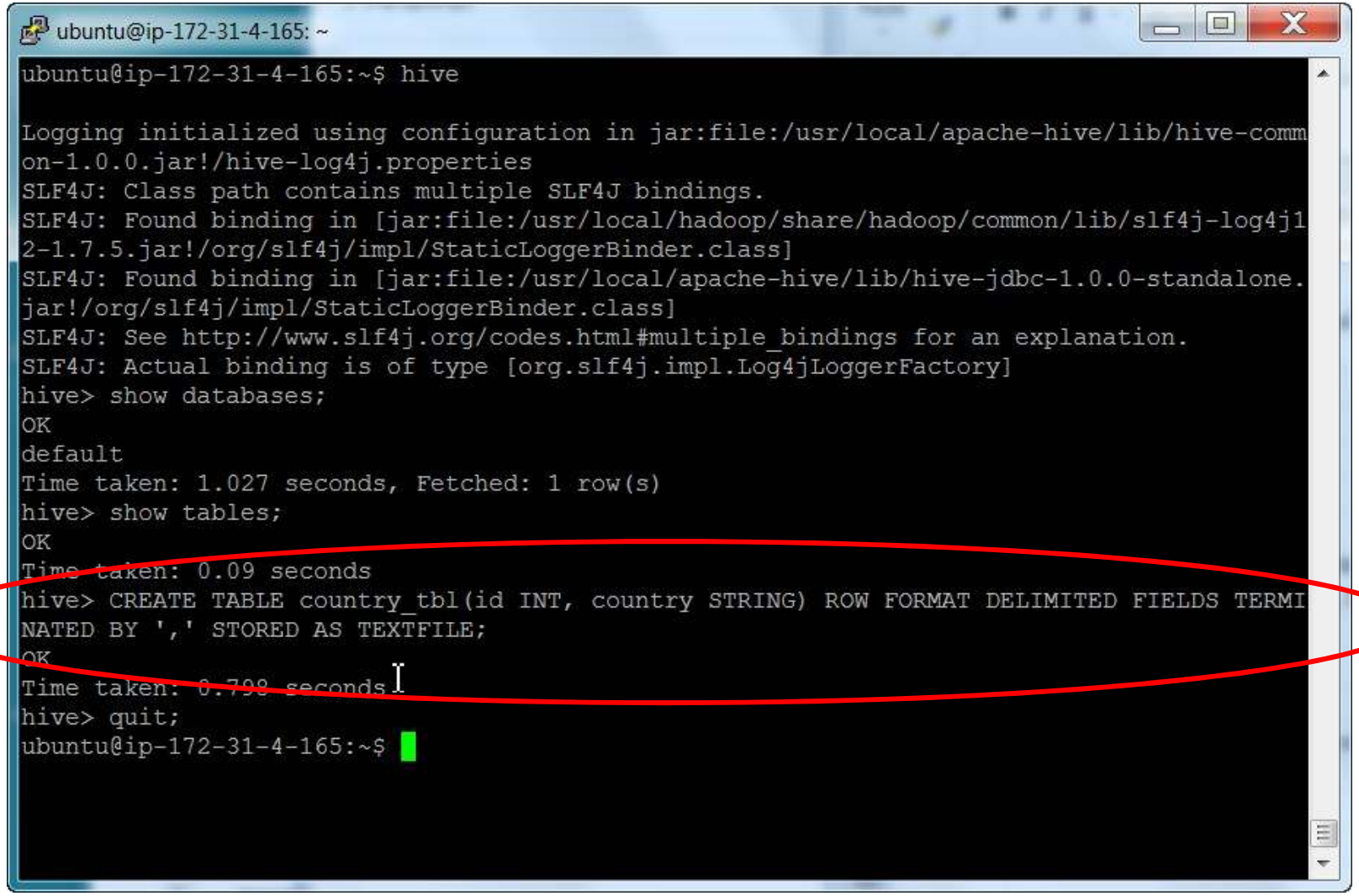


3. Creating Hive Table

```
hive (default)> CREATE TABLE test_tbl(id INT, country STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
OK
Time taken: 4.069 seconds
hive (default)> show tables;
OK
test_tbl
Time taken: 0.138 seconds
hive (default)> describe test_tbl;
OK
id       int
country  string
Time taken: 0.147 seconds
hive (default)>
```

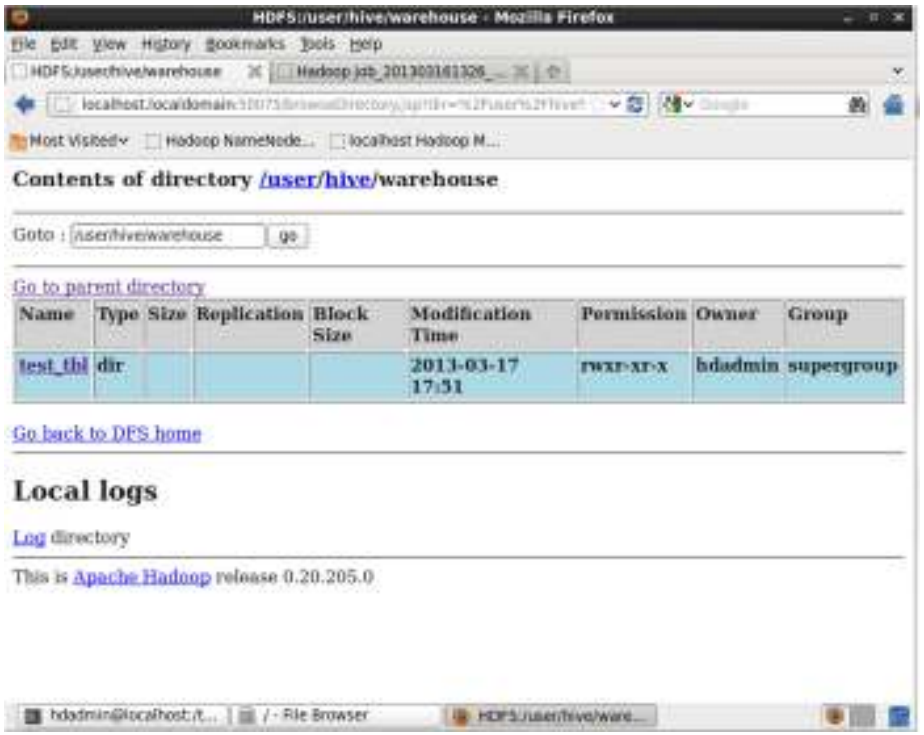
See also: https://cwiki.apache.org/Hive/languagemanual-ddl.html

3. Creating Hive Table



4. Reviewing Hive Table in HDFS

```
[hdadmin@localhost hdadmin]$ hadoop fs -ls /user/hive/warehouse
Found 1 items
drwxr-xr-x   - hdadmin supergroup          0 2013-03-17 17:51
/user/hive/warehouse/test_tbl
[hdadmin@localhost hdadmin]$
```



Review Hive Table from HDFS WebUI

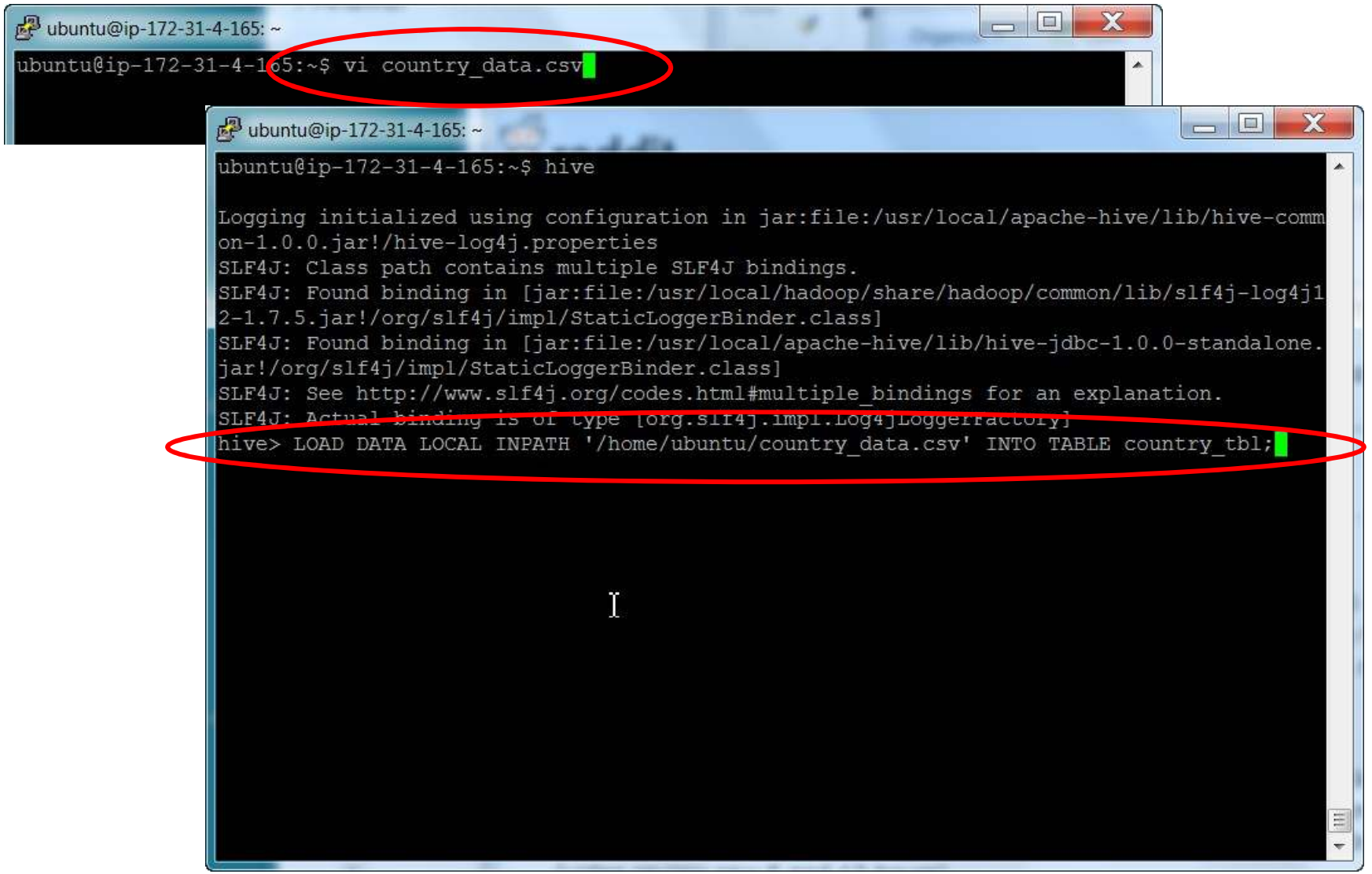
5. Alter and Drop Hive Table

```
hive (default)> alter table test_tbl add columns (remarks STRING);

hive (default)> describe test_tbl;
OK
id          int
country     string
remarks     string
Time taken: 0.077 seconds
hive (default)> drop table test_tbl;
OK
Time taken: 0.9 seconds
```

See also: https://cwiki.apache.org/Hive/adminmanual-metastoreadmin.html

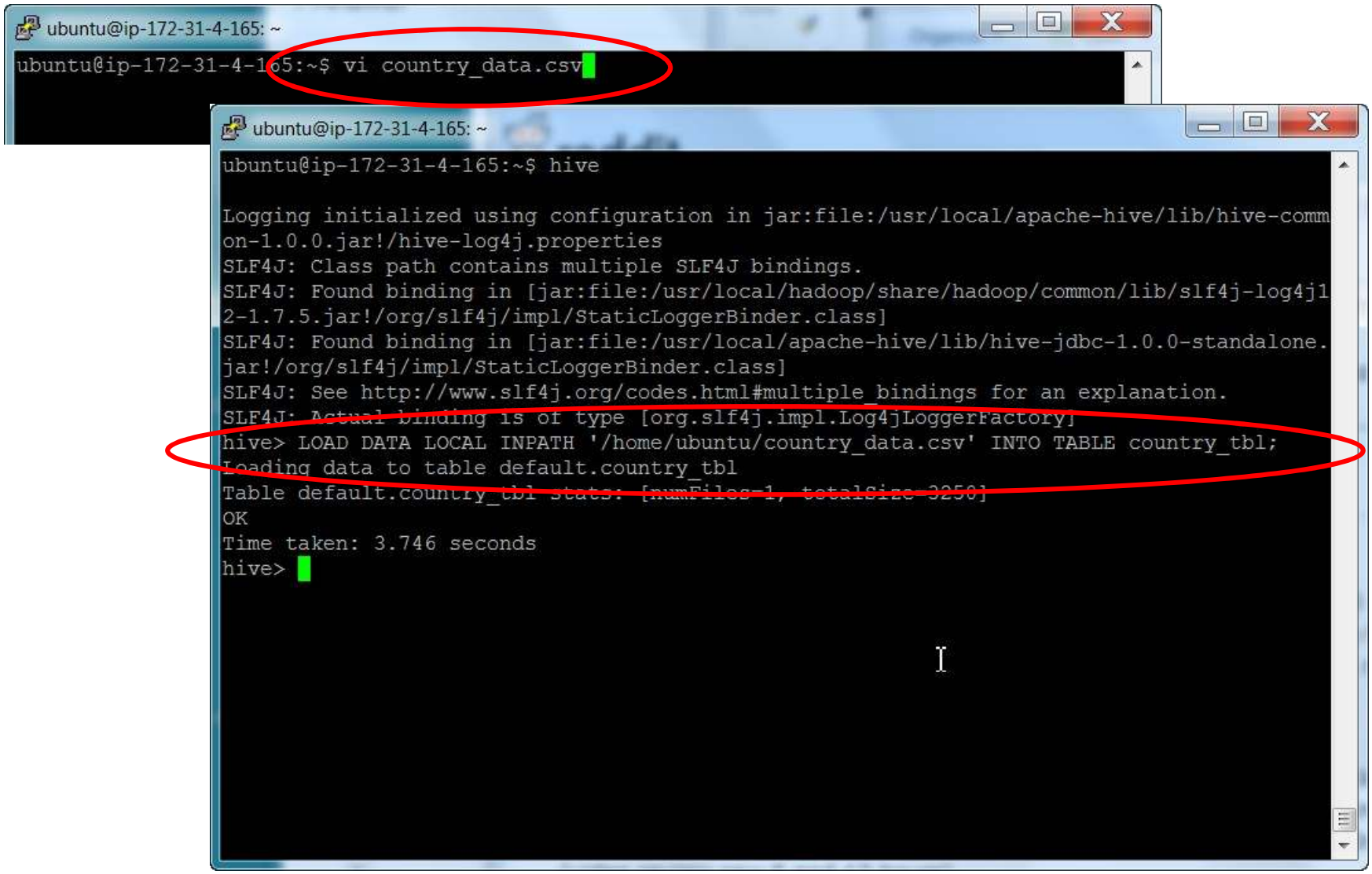
6. Loading Data to Hive Table

Please get the country_data.csv from your instructor.



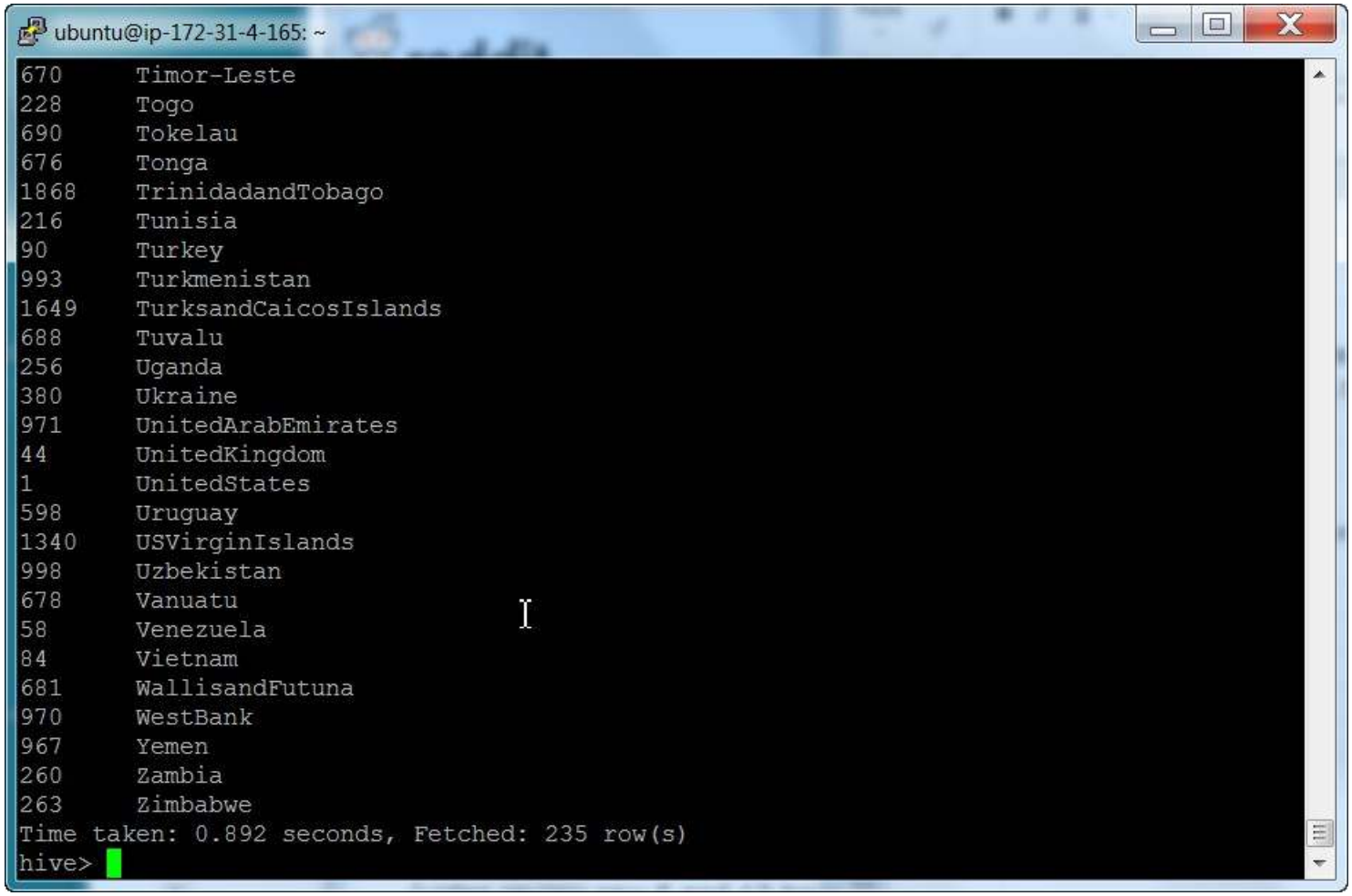
6. Loading Data to Hive Table

Please get the country_data.csv from your instructor.

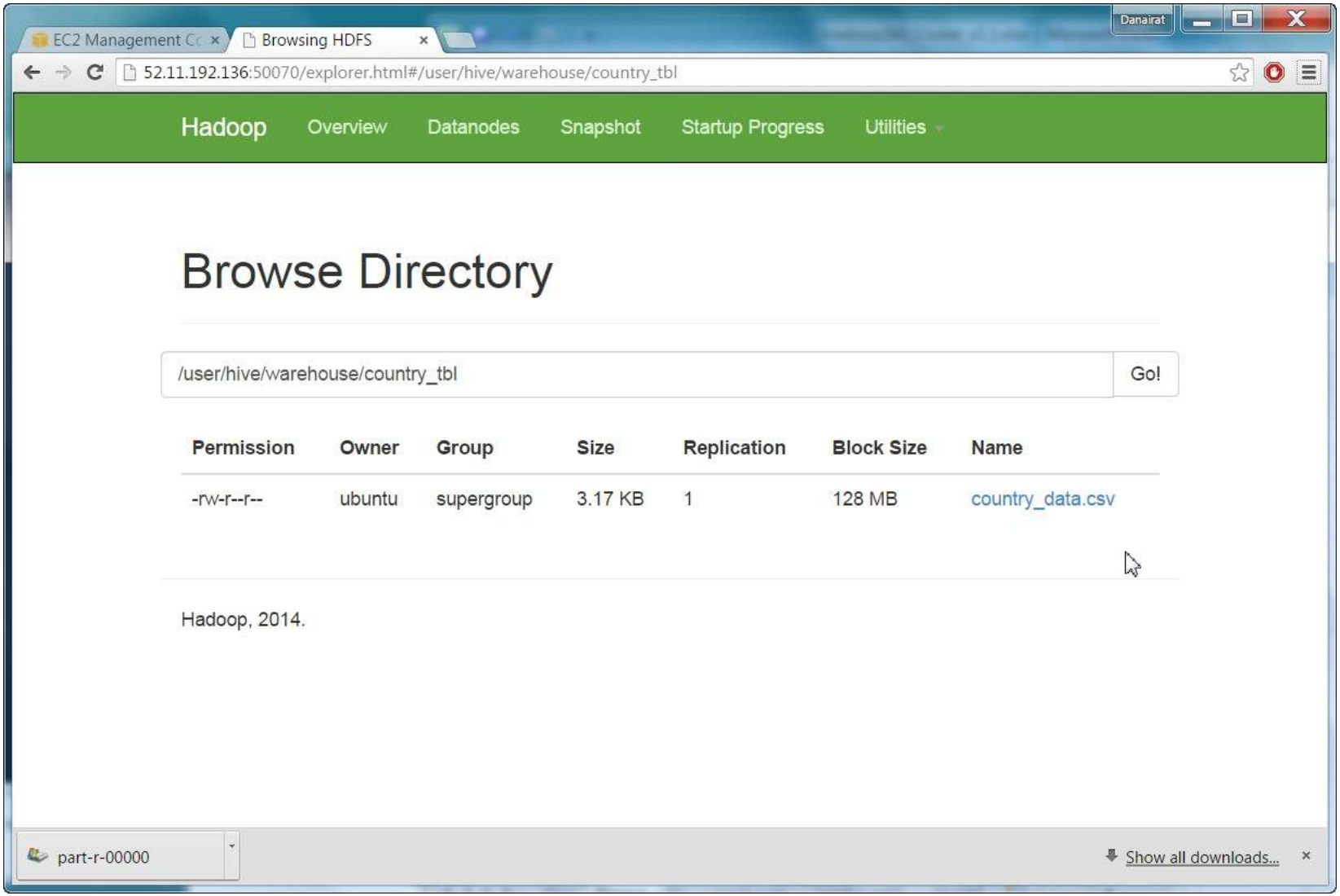


7. Querying Data from Hive Table

```
hive> select * from country_tbl;
```



8. Reviewing Hive Table Content from HDFS Command and WebUI



Lecture: Pig

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

Introduction

A high-level platform for creating MapReduce programs Using Hadoop

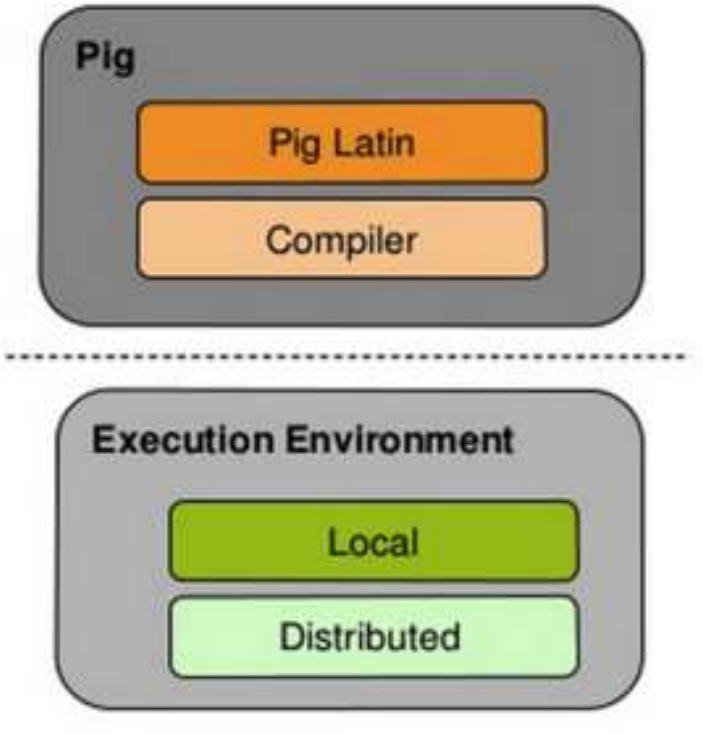
Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

Pig Components

- **Two Compnents**
  - Language (Pig Latin)
  - Compiler
- **Two Execution Environments**
  - **Local**

  pig -x local
  - **Distributed**

    pig -x mapreduce



Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

Running Pig

- **Script**

  pig myscript
- **Command line (Grunt)**

  pig
- **Embedded**

  Writing a java program

Pig Dataflow Organization

1. A LOAD statement reads data from the file system.
2. A series of "transformation" statements process the data.
3. A STORE statement writes output to the file system; or, a DUMP statement displays output to the screen.

Pig Latin

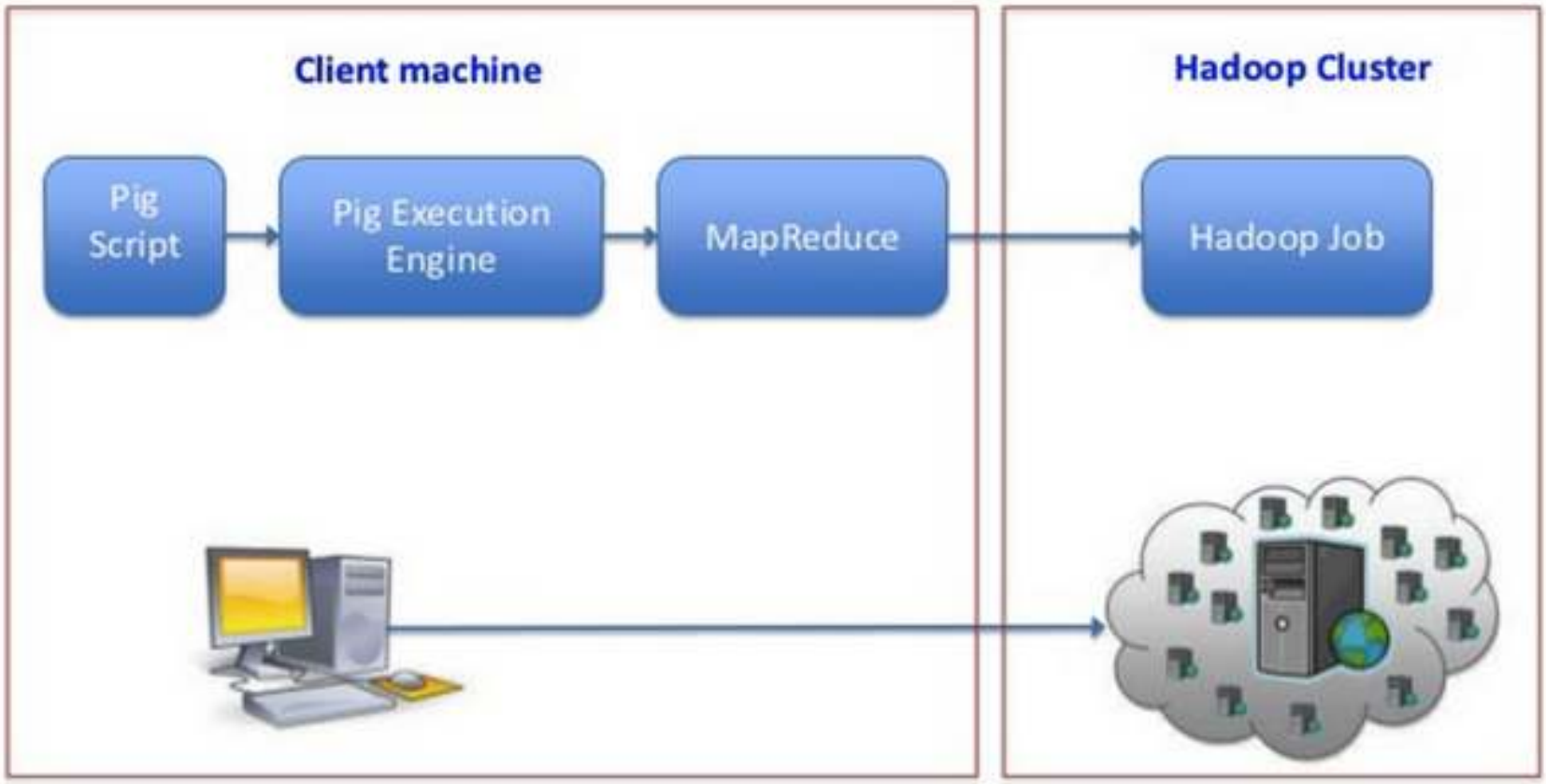
```
A = load 'hdi-data.csv' using PigStorage(',') AS
(id:int, country:chararray, hdi:float, lifeex:int,
mysch:int, eysch:int, gni:int);
B = FILTER A BY gni > 2000;
C = ORDER B BY gni;
dump C;
```

Pig Command Language

Pig Command	What it does
load	Read data from file system.
store	Write data to file system.
foreach	Apply expression to each record and output one or more records.
filter	Apply predicate and remove records that do not return true.
group/cogroup	Collect records with the same key from one or more inputs.
join	Join two or more inputs based on a key.
order	Sort records based on a key.
distinct	Remove duplicate records.
union	Merge two data sets.
split	Split data into 2 or more sets, based on filter conditions.
stream	Send all records through a user provided binary.
dump	Write output to stdout.
limit	Limit the number of records.

Cloudera, 2009

Pig Execution Stages



Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

Why Pig?

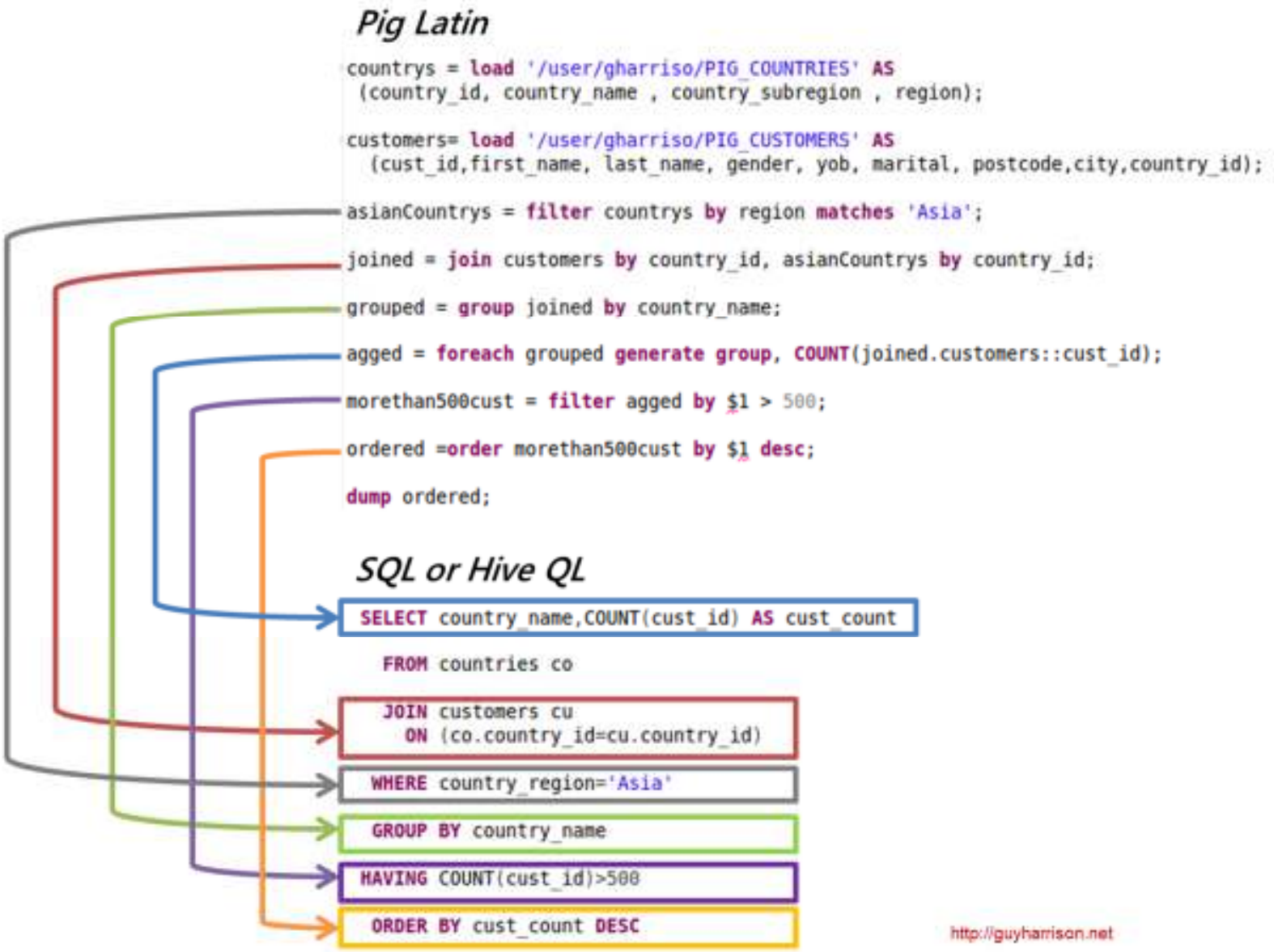
- **Makes writing Hadoop jobs easier**
  - You don't need to be a programmer to write Pig scripts
- **Provide major functionality required for DatawareHouse and Analytics**
  - *Load, Filter, Join, Group By, Order, Transform*
- **User can write custom UDFs (User Defined Function)**

Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

Pig v.s. Hive

Characteristic	Pig	Hive
Developed by	Yahoo!	Facebook
Language name	Pig Latin	HiveQL
Type of language	Data flow	Declarative (SQL dialect)
Data structures it operates on	Complex, nested	
Schema optional?	Yes	No, but data can have many schemas
Relational complete?	Yes	Yes
Turing complete?	Yes when extended with Java UDFs	Yes when extended with Java UDFs

Pig v.s. Hive


Pig Latin
countrys = load '/user/gharriso/PIG_COUNTRIES' AS
(country_id, country_name , country_subregion , region);
customers= load '/user/gharriso/PIG_CUSTOMERS' AS
(cust_id,first_name, last_name, gender, yob, marital, postcode,city,country_id);
asianCountrys = filter countrys by region matches 'Asia';
joined = join customers by country_id, asianCountrys by country_id;
grouped = group joined by country_name;
agged = foreach grouped generate group, COUNT(joined.customers::cust_id);
morethan500cust = filter agged by $1 > 500;
ordered =order morethan500cust by $1 desc;
dump ordered;
SQL or Hive QL
SELECT country_name,COUNT(cust_id) AS cust_count
FROM countries co
JOIN customers cu
ON (co.country_id=cu.country_id)
WHERE country_region='Asia'
GROUP BY country_name
HAVING COUNT(cust_id)>500
ORDER BY cust_count DESC
http://guyharrison.net

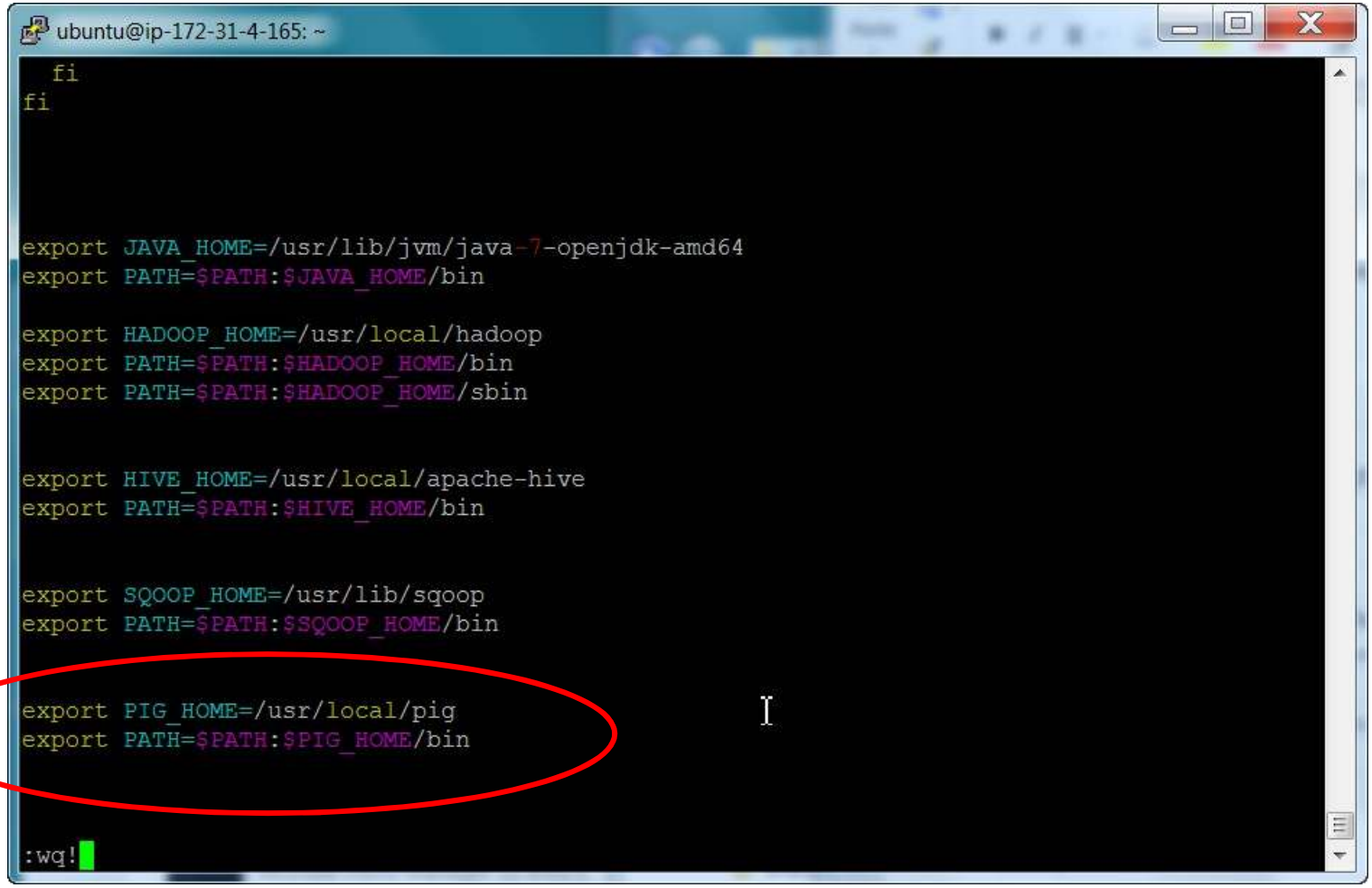
Hands-On: Running a Pig script

Installing Pig

```
$ wget http://apache.cs.utah.edu/pig/pig-0.14.0/pig-0.14.0.tar.gz

$ tar -xvf pig-0.14.0.tar.gz

$ mv pig-0.14.0 pig

$ sudo mv pig /usr/local/
```

Edit System Environment Variables

```
vi ./.bashrc
```

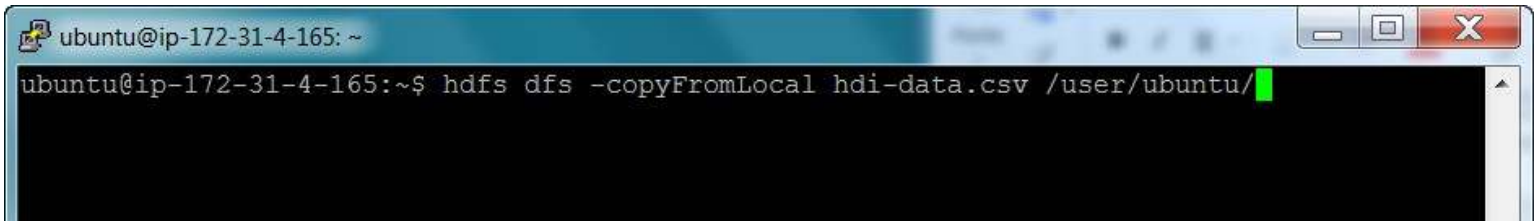


Execute System Environment

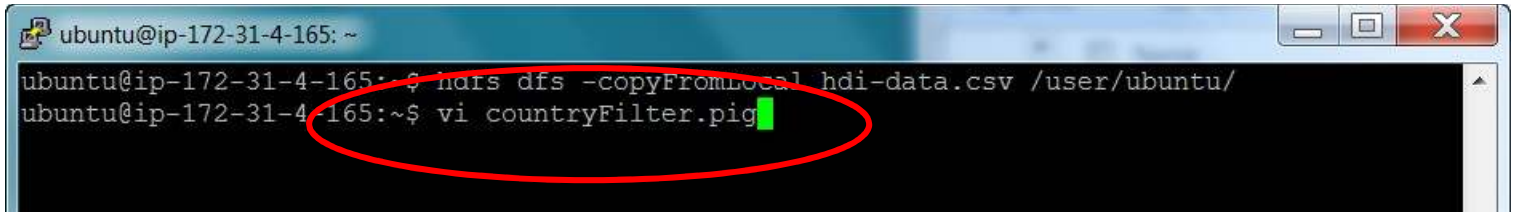
```
source ~/.bashrc
```

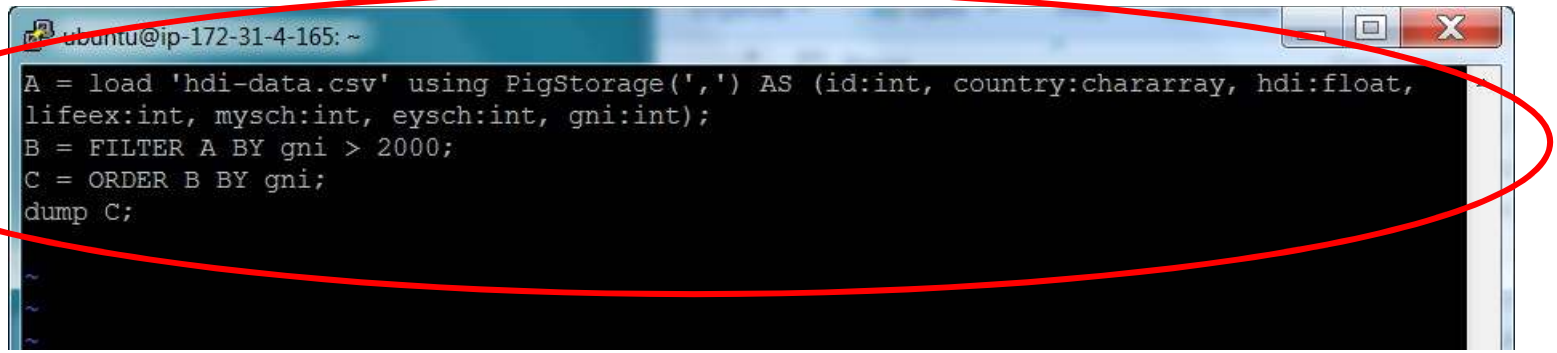
Download sample data

Import sample data to HDFS

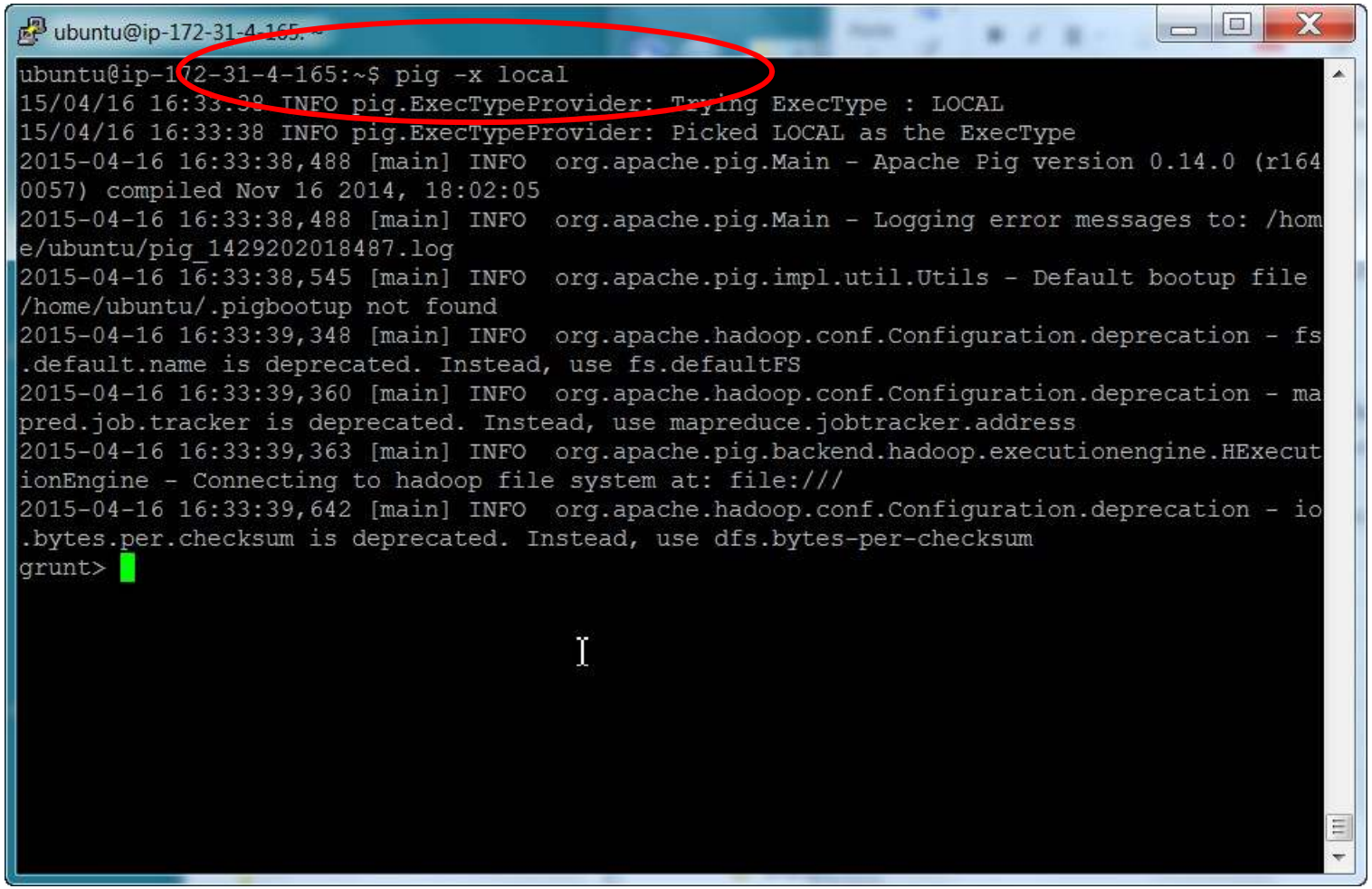


Write your own Pig script



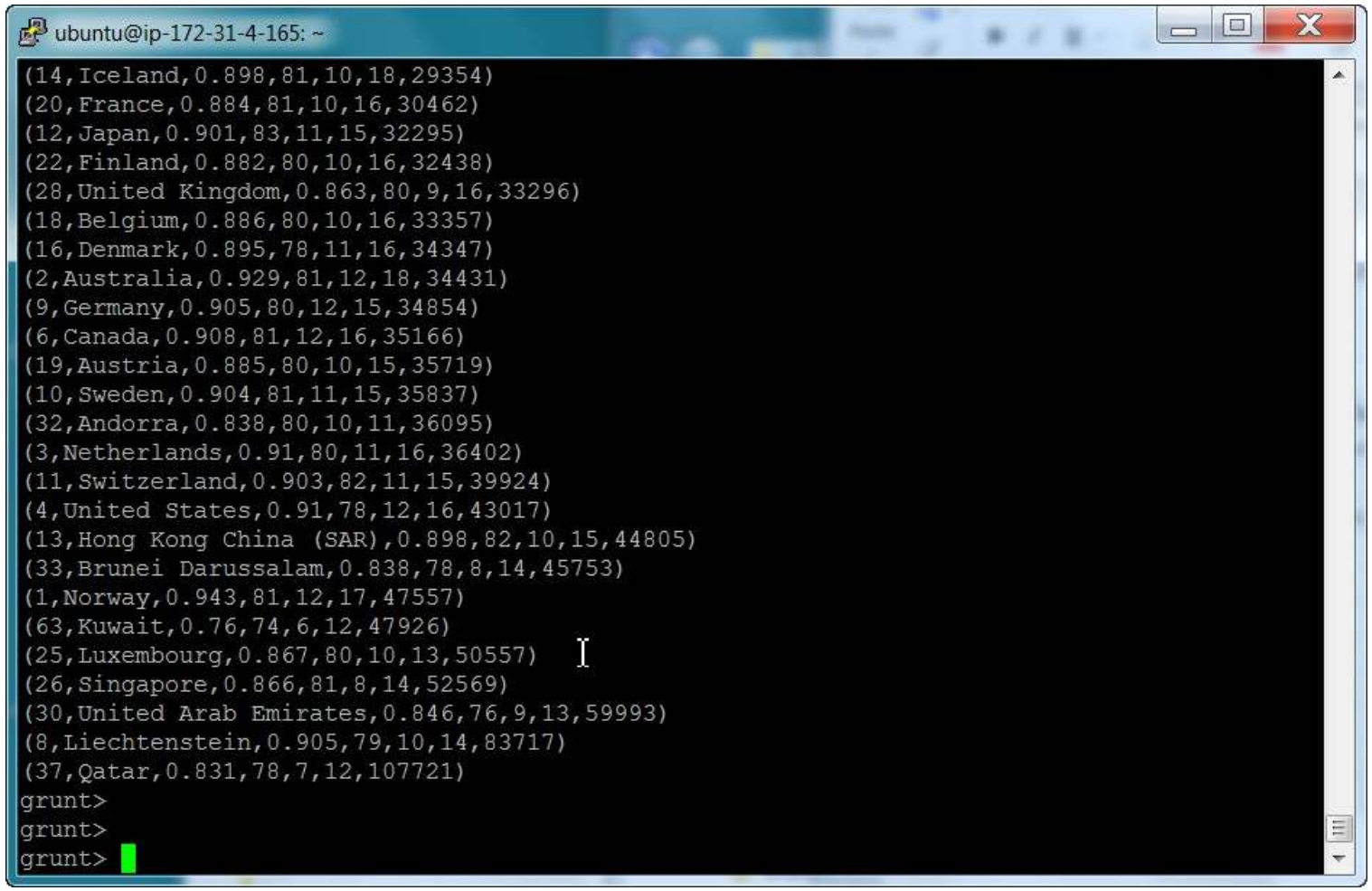


Running Pig shell

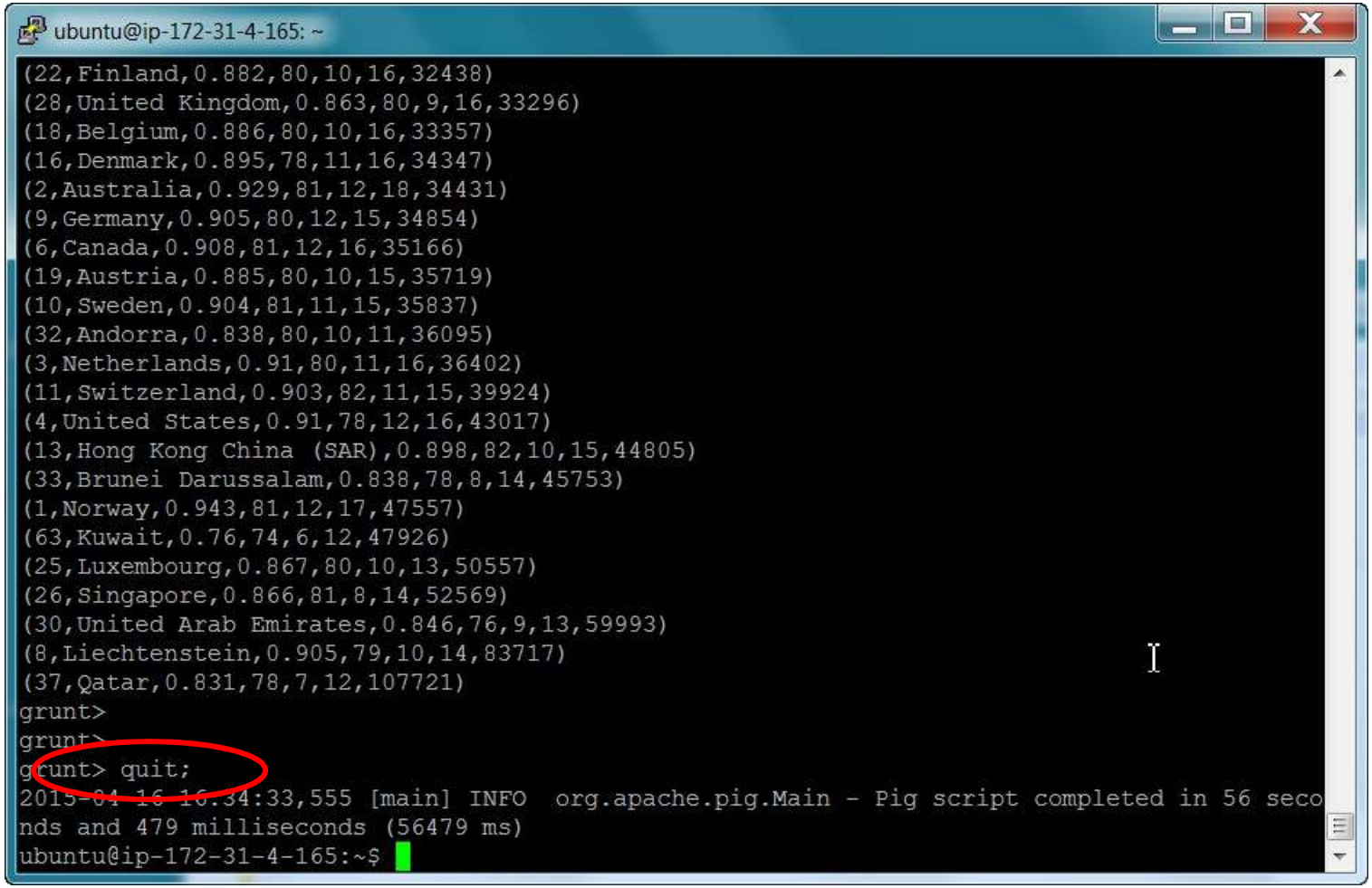


Execute Pig Script

```
run countryFilter.pig
```



Quit from Pig Shell



Lecture: Sqoop

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

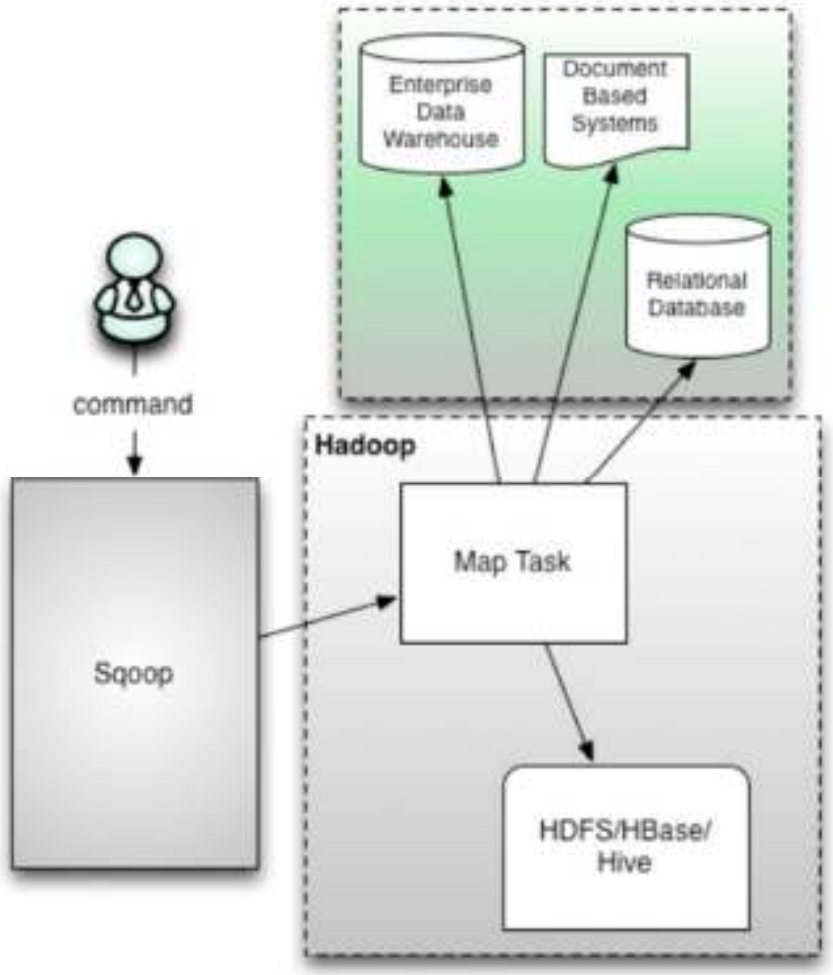
Introduction

Sqoop ("SQL-to-Hadoop") is a straightforward command-line tool with the following capabilities:

- **Imports individual tables or entire databases to files in HDFS**
- **Generates Java classes to allow you to interact with your imported data**
- **Provides the ability to import from SQL databases straight into your Hive data warehouse**

See also: http://sqoop.apache.org/docs/1.4.2/SqoopUserGuide.html

Architecture Overview



Hive.apache.org

Hands-On: Loading Data from DBMS to Hadoop HDFS

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com

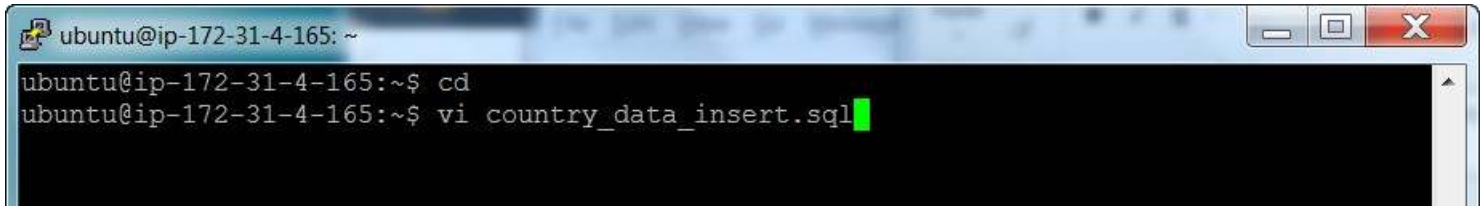
Sqoop Hands-On Labs

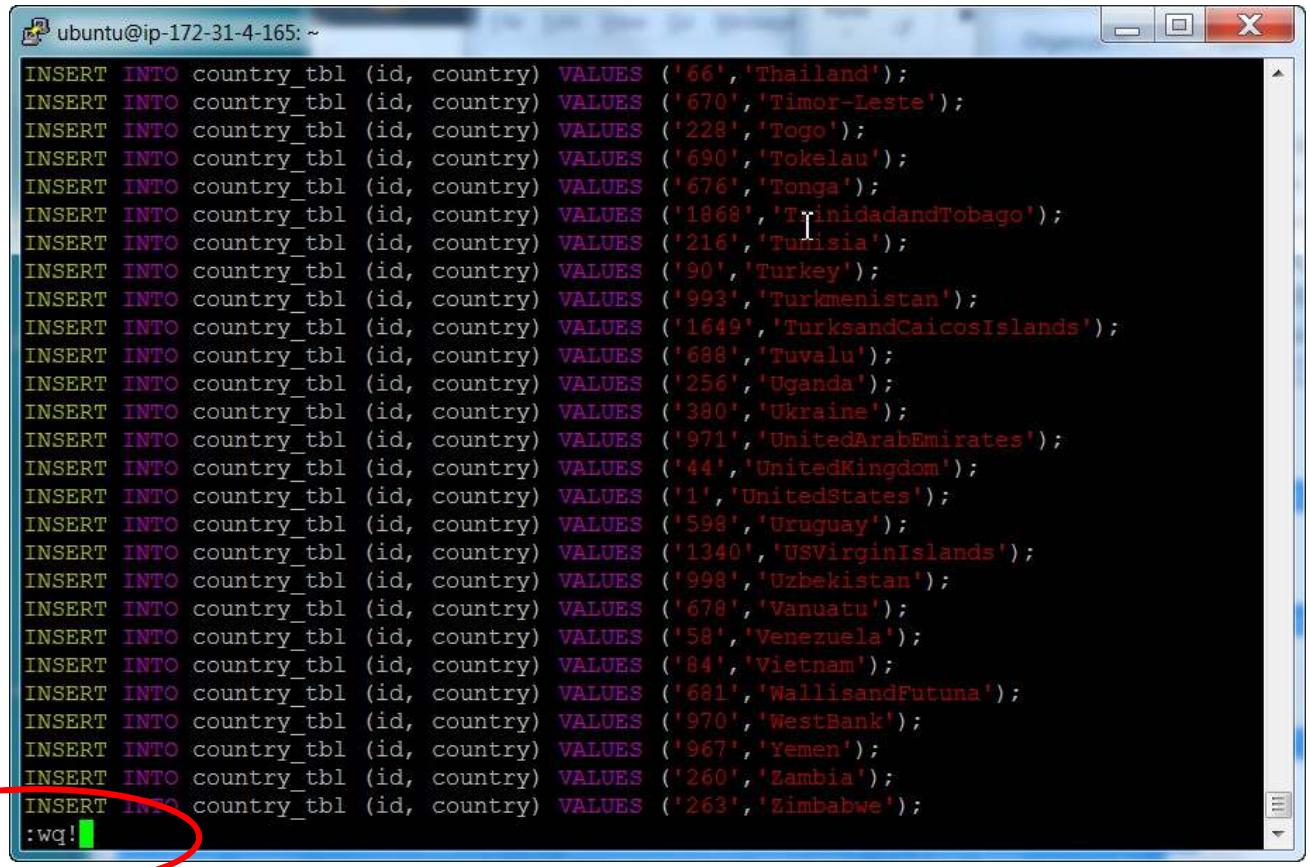
1. Loading Data into MySQL DB
2. Installing Sqoop
3. Configuring Sqoop
4. Installing DB driver for Sqoop
5. Importing data from MySQL to Hadoop
6. Reviewing HDFS data

1. Loading Data into MySQL DB

Create sample sql scripts

Please get the script from your instructor

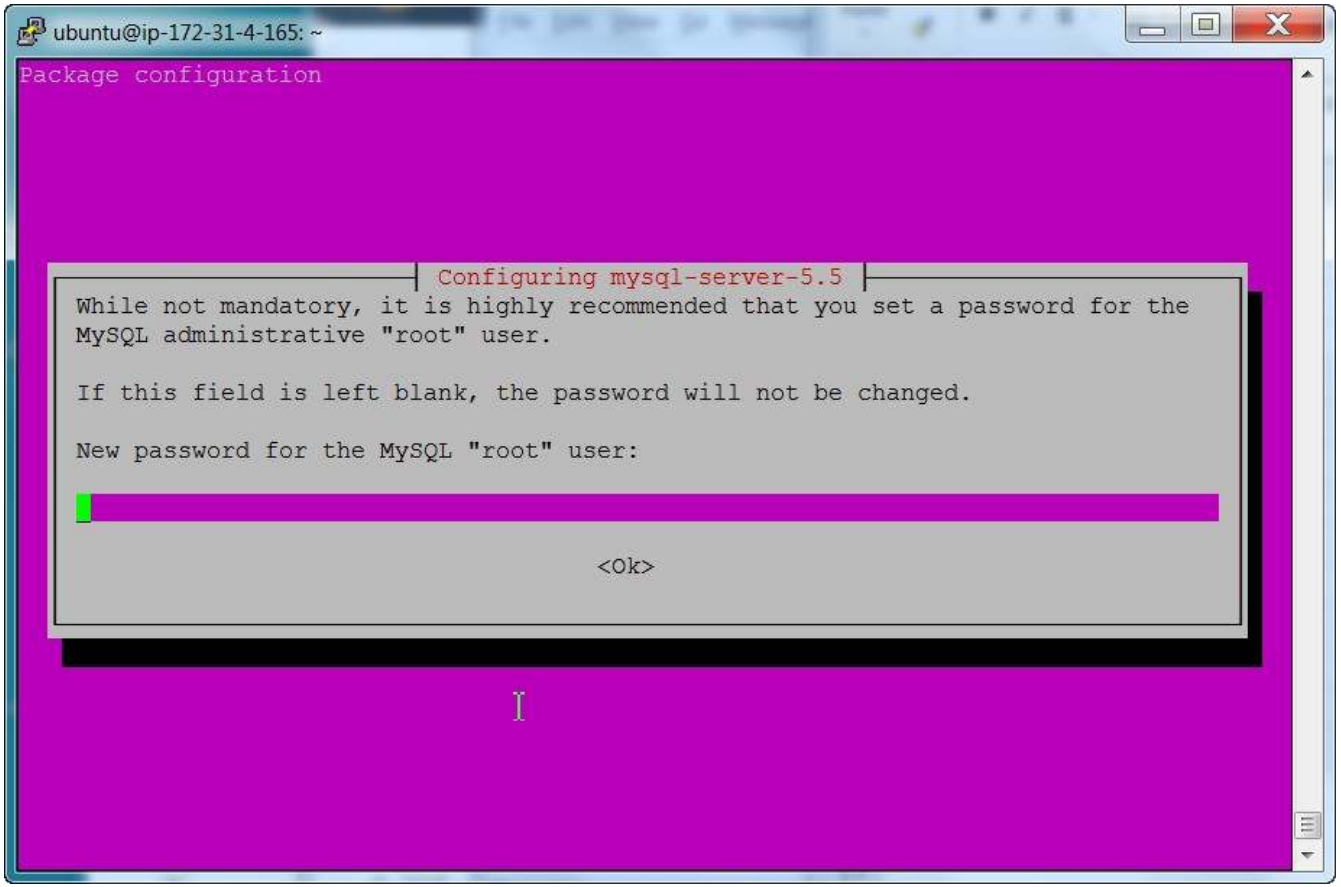




Install MySQL

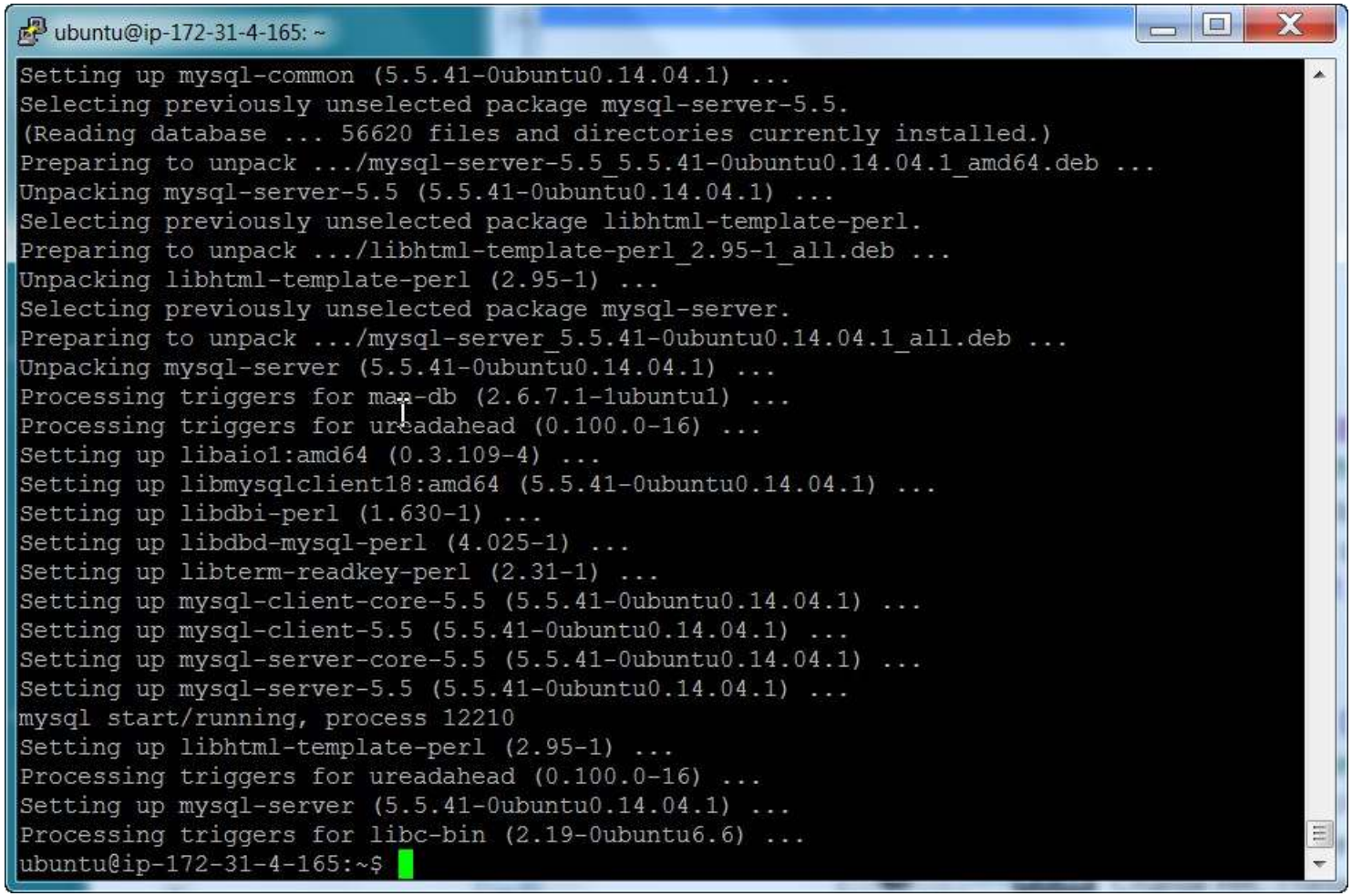
```
sudo apt-get install mysql-server
```

Leave black for password in our class room testing

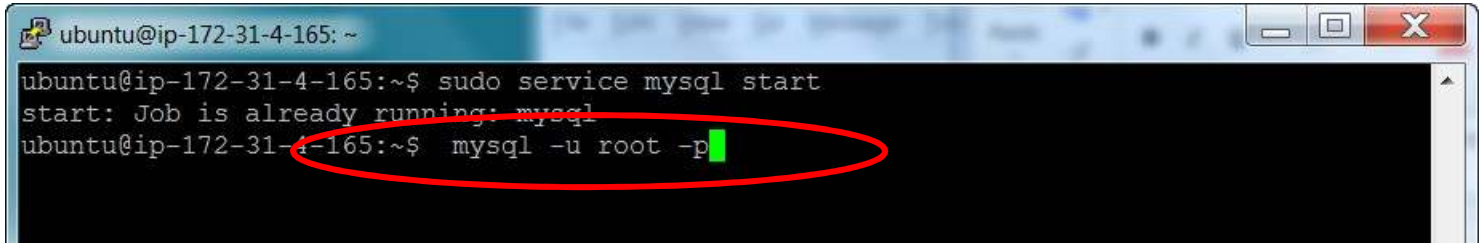


Install MySQL

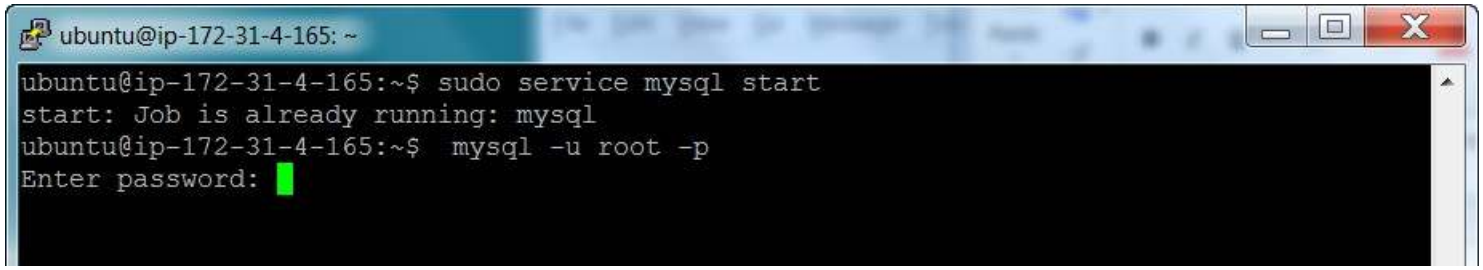
Done installation



Start MySQL service



Press enter for password



Create database name: countrydb

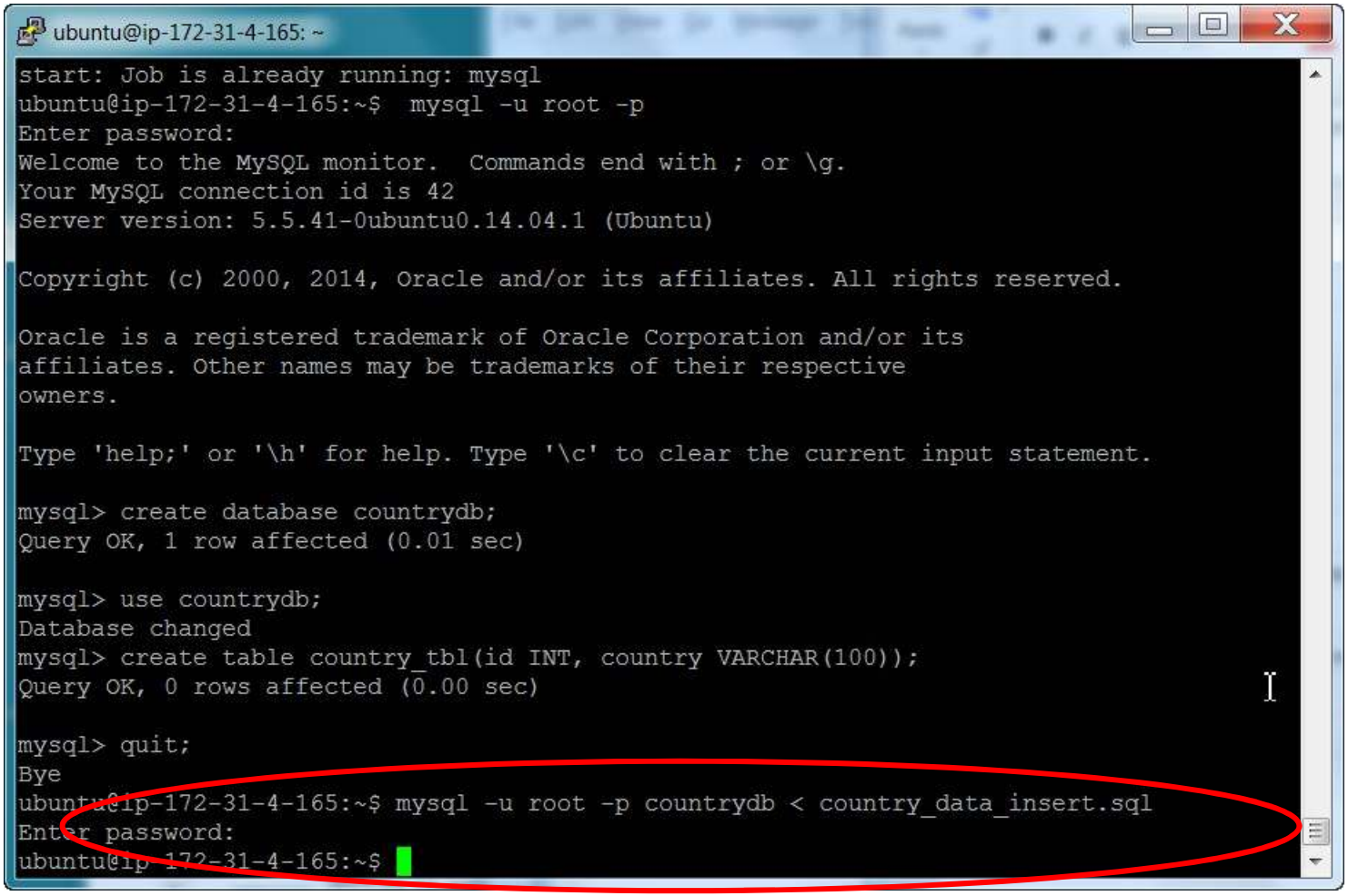
Select database name: countrydb

```
mysql> use countrydb;
Database changed
mysql>
```

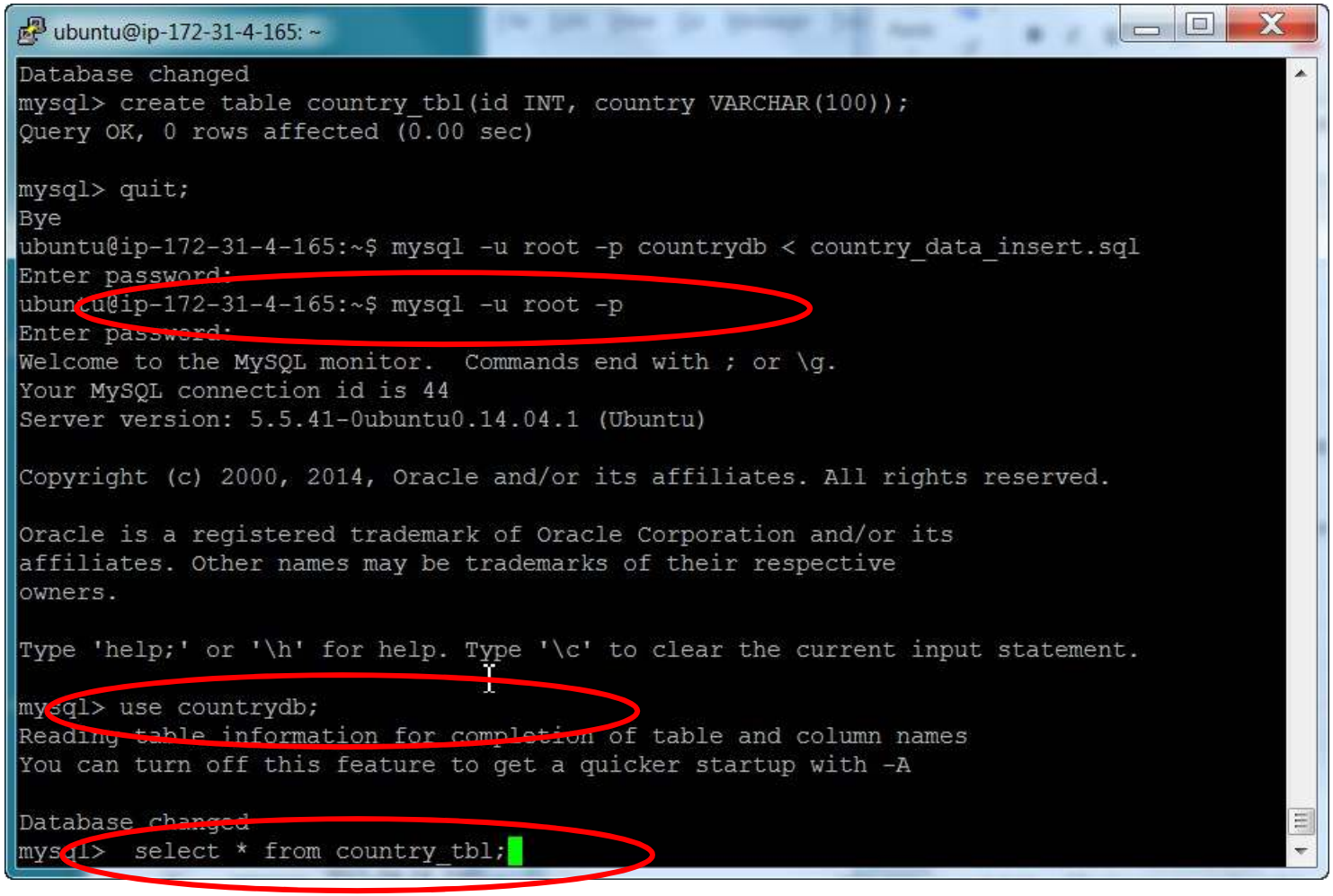
Create table name: country_tbl

```
mysql> create table country_tbl(id INT, country VARCHAR(100));
```

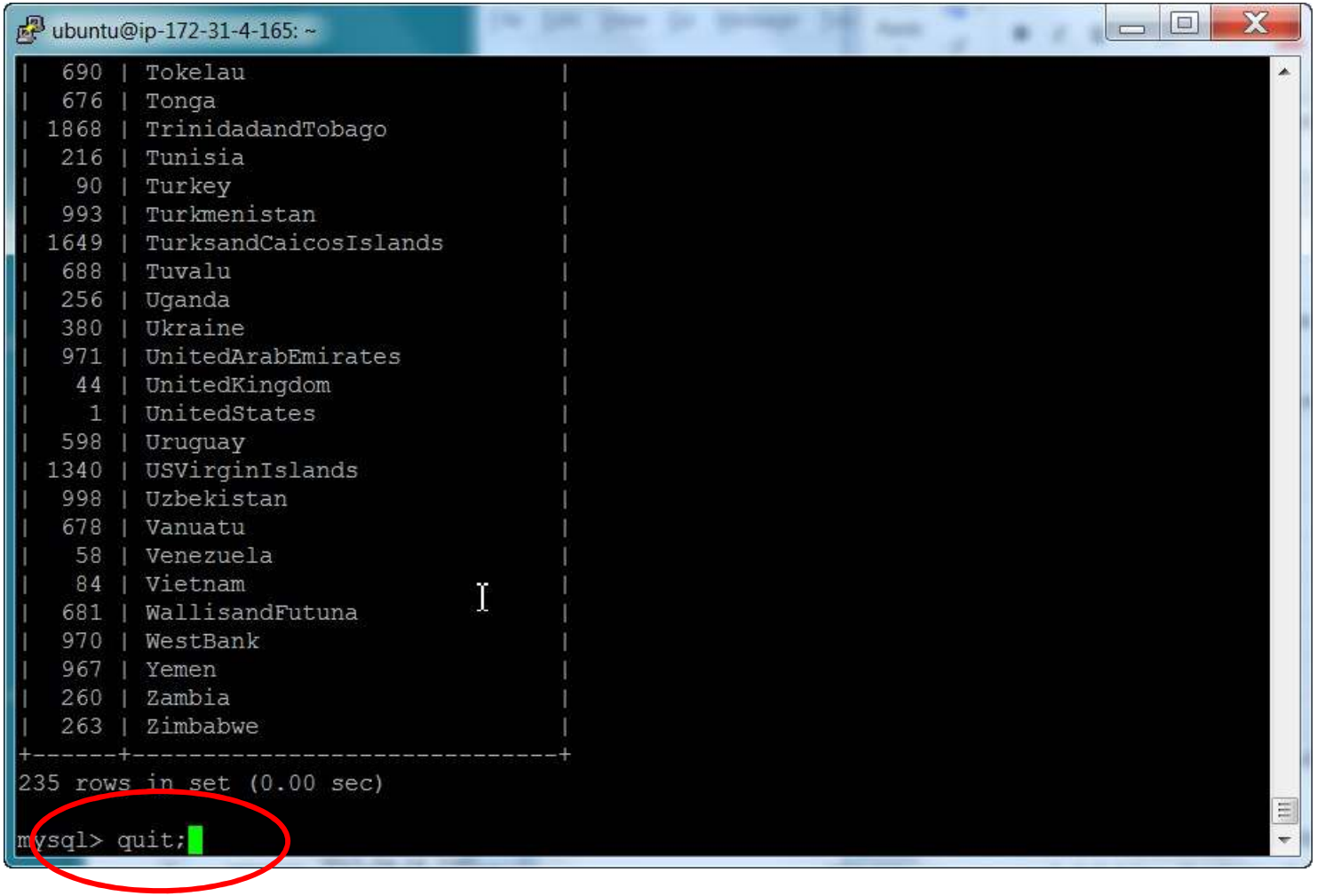
Import your sample data to MySQL



Review your sample data from MySQL



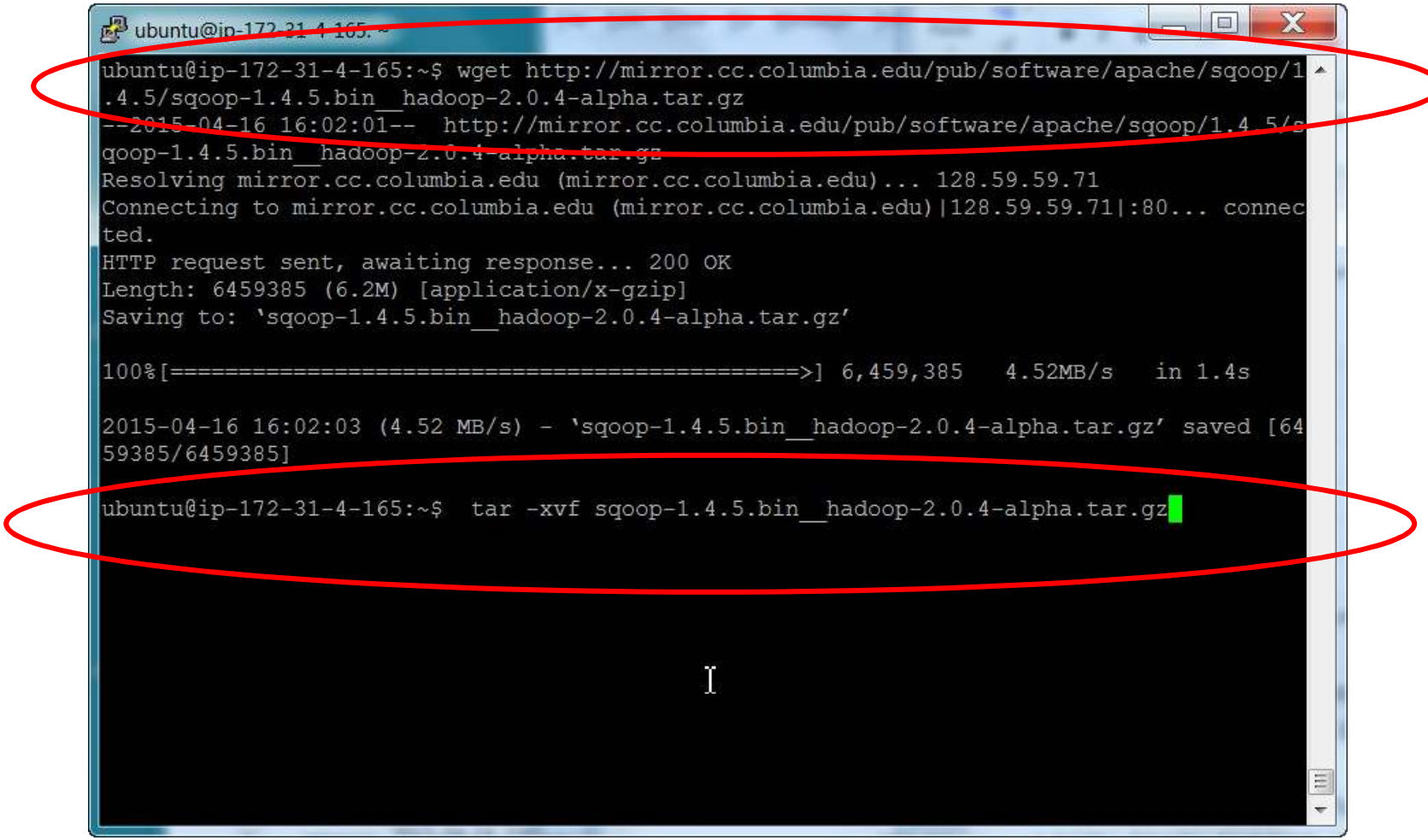
Review your sample data from MySQL



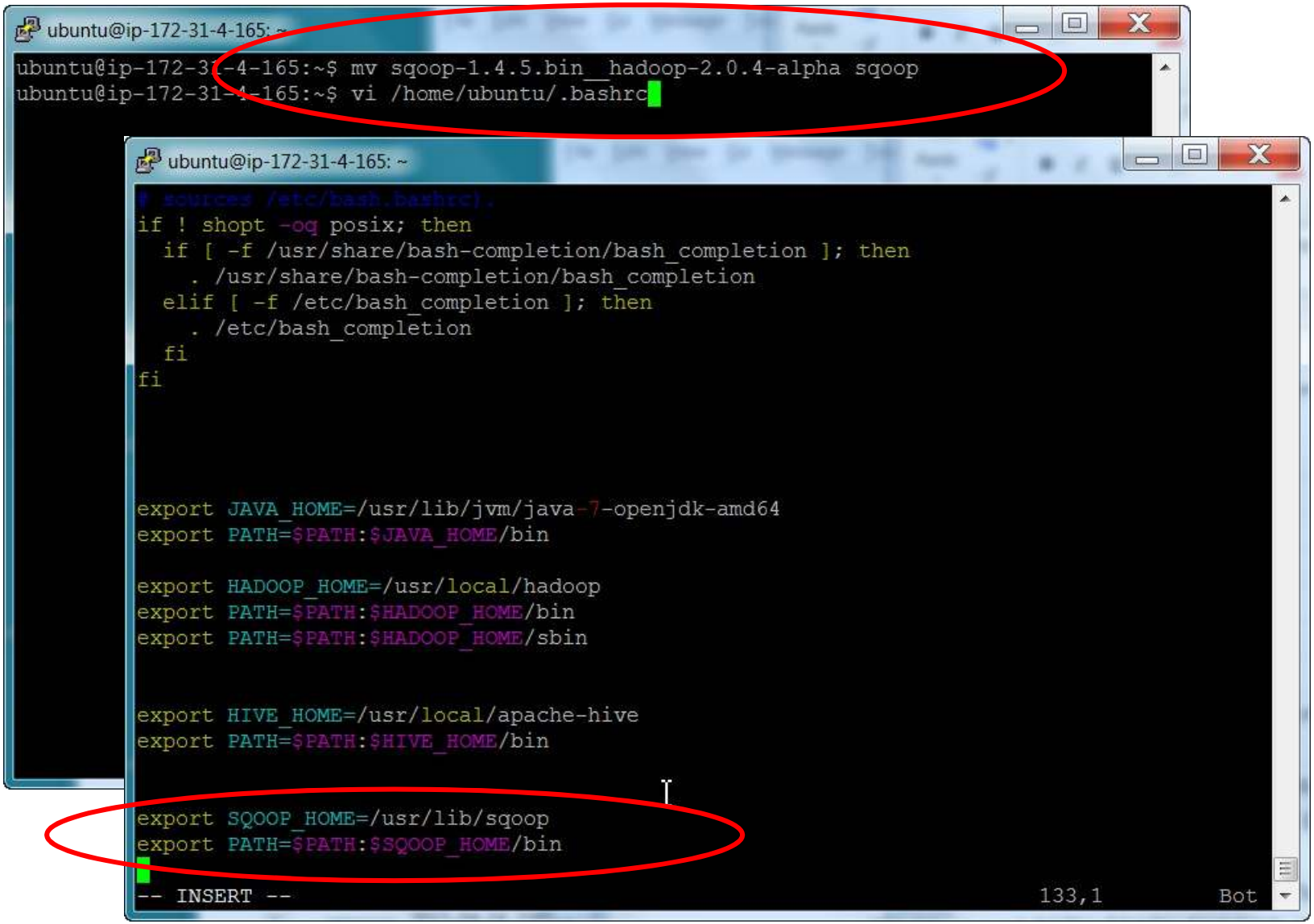
2. Installing Sqoop

Install Sqoop

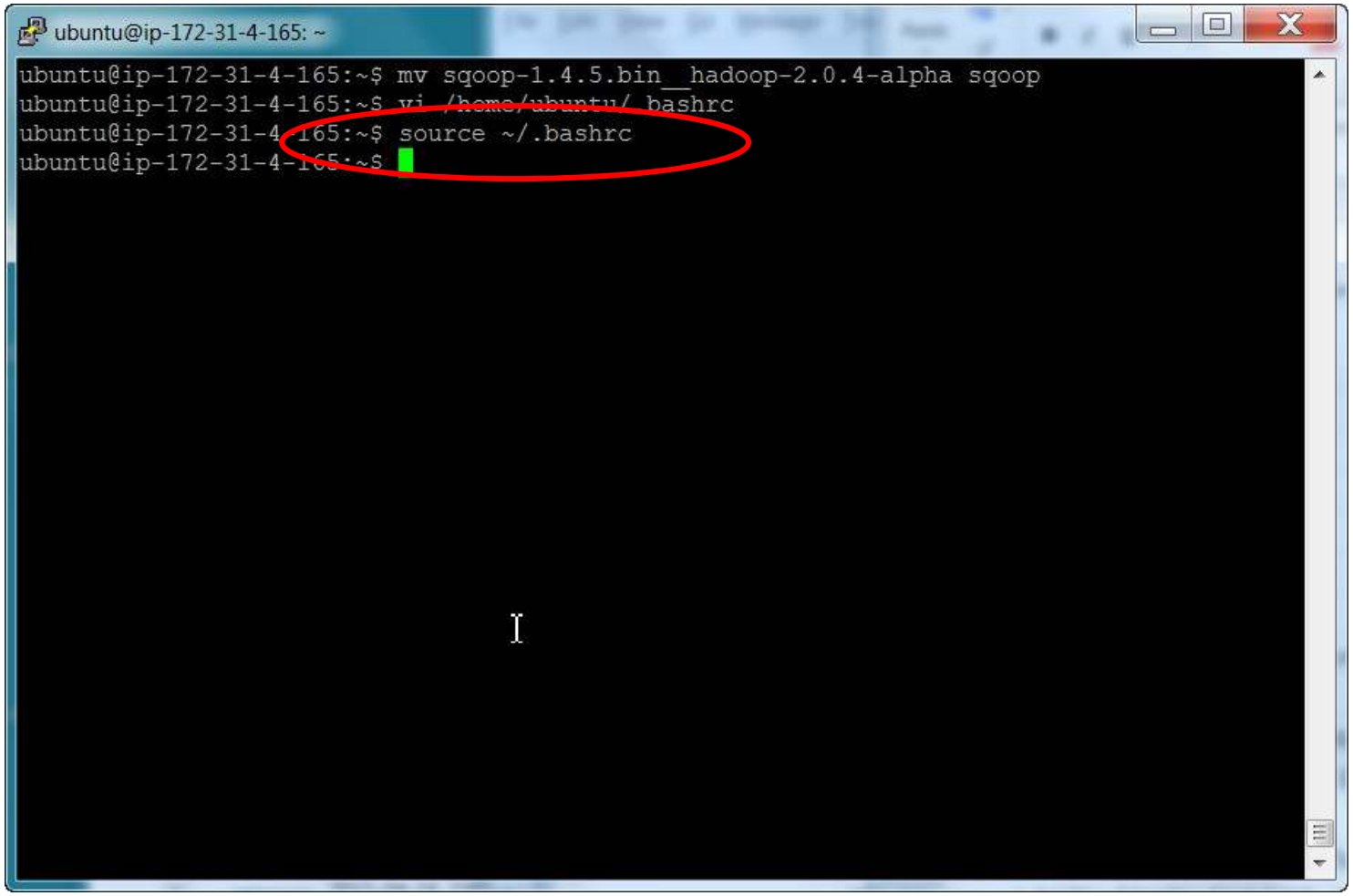
wget http://mirror.cc.columbia.edu/pub/software/apache/sqoop/1.4.5/sqoop-1.4.5.bin__hadoop-2.0.4-alpha.tar.gz



Edit System Environment Variables

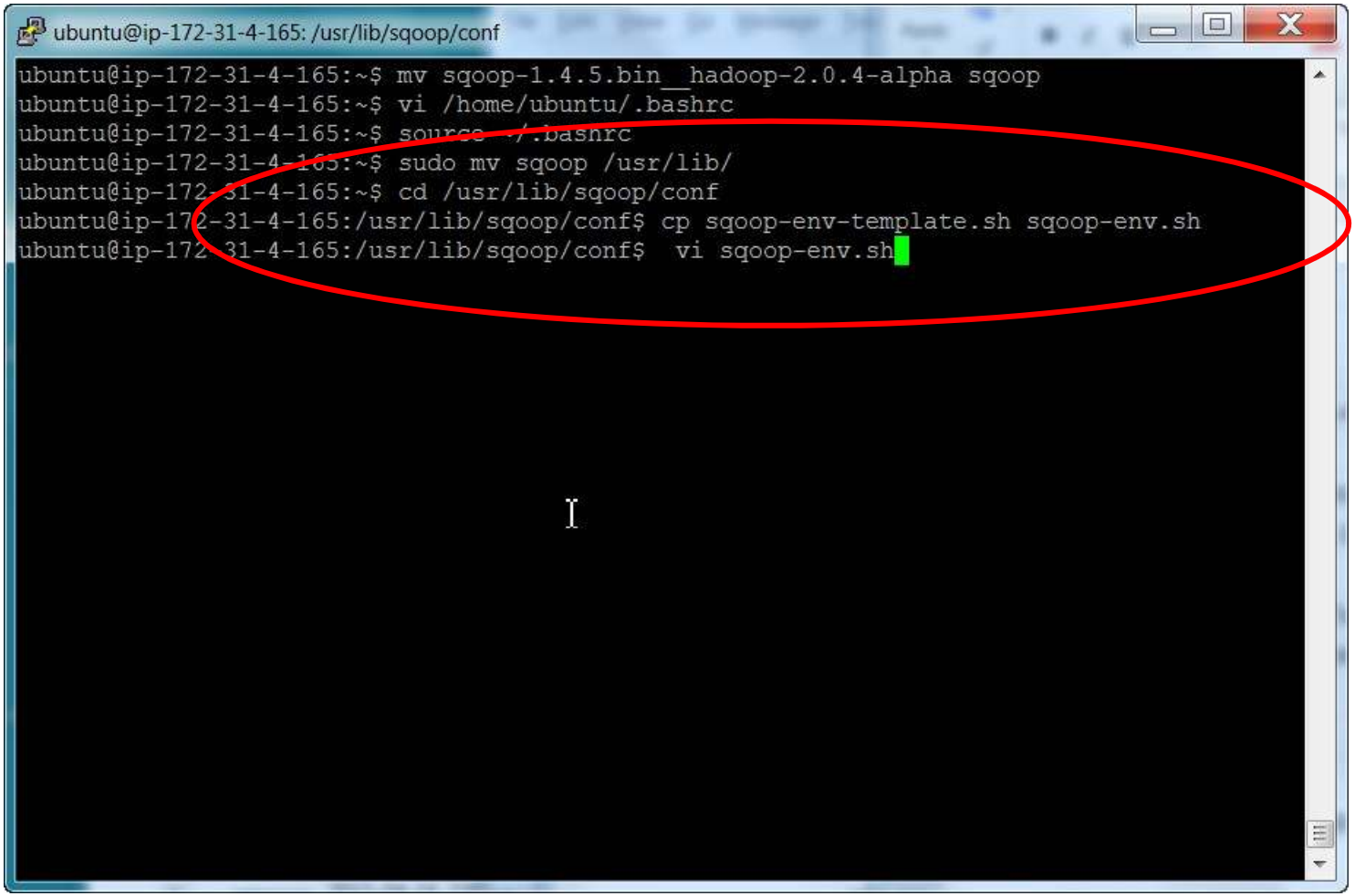


Execute System Environment Variables

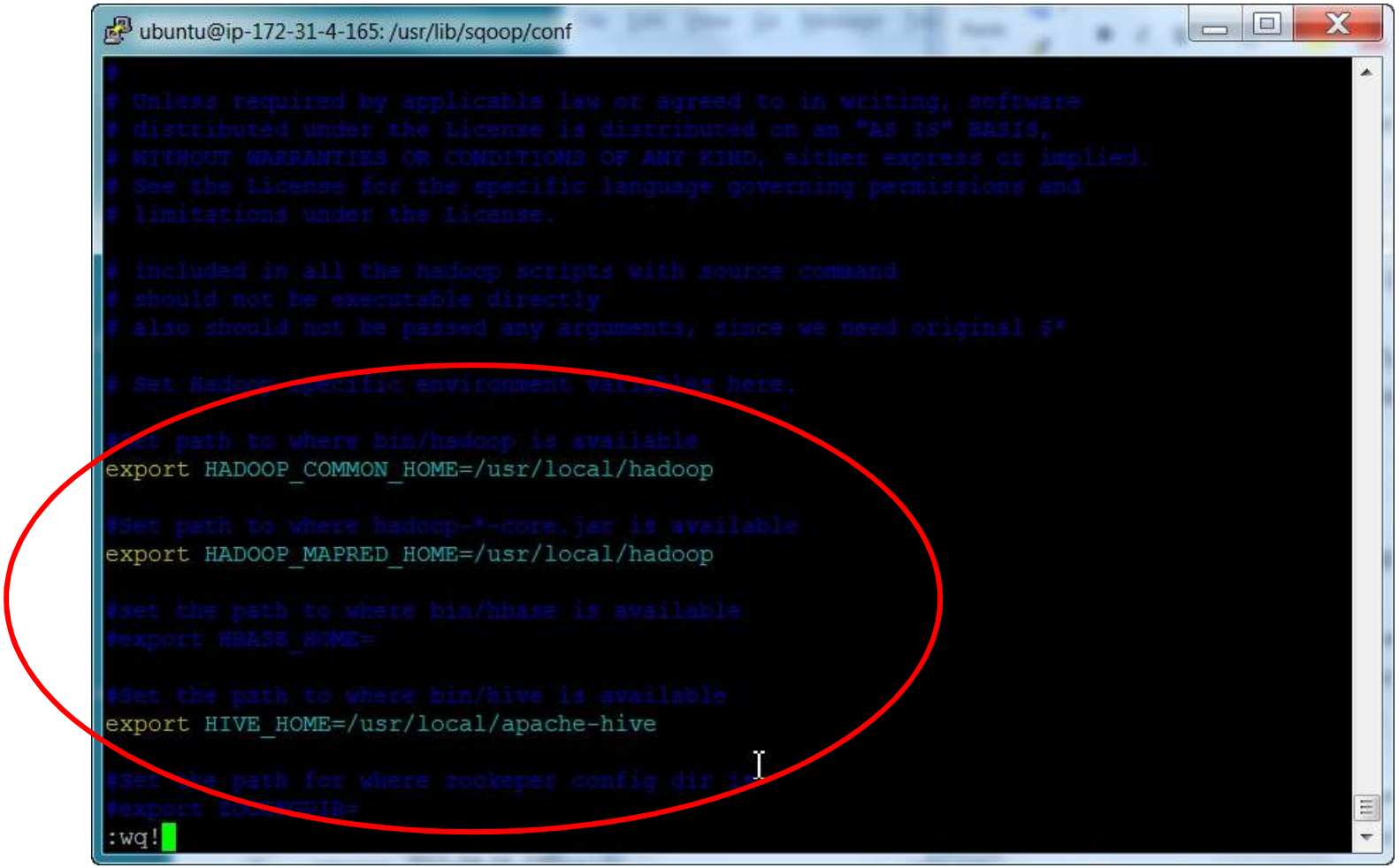


3. Configuring Sqoop

Configuring Sqoop



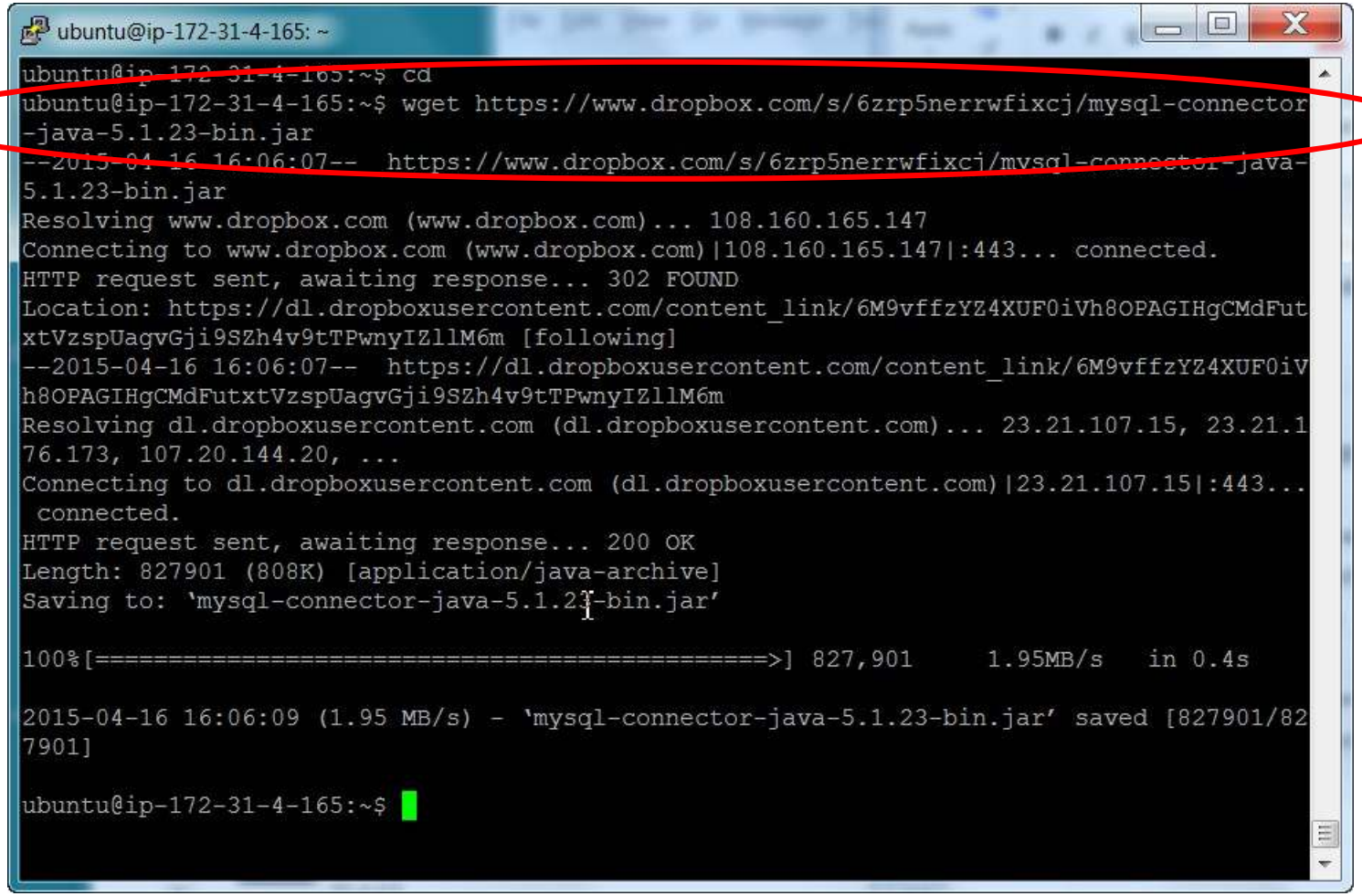
Configuring Sqoop



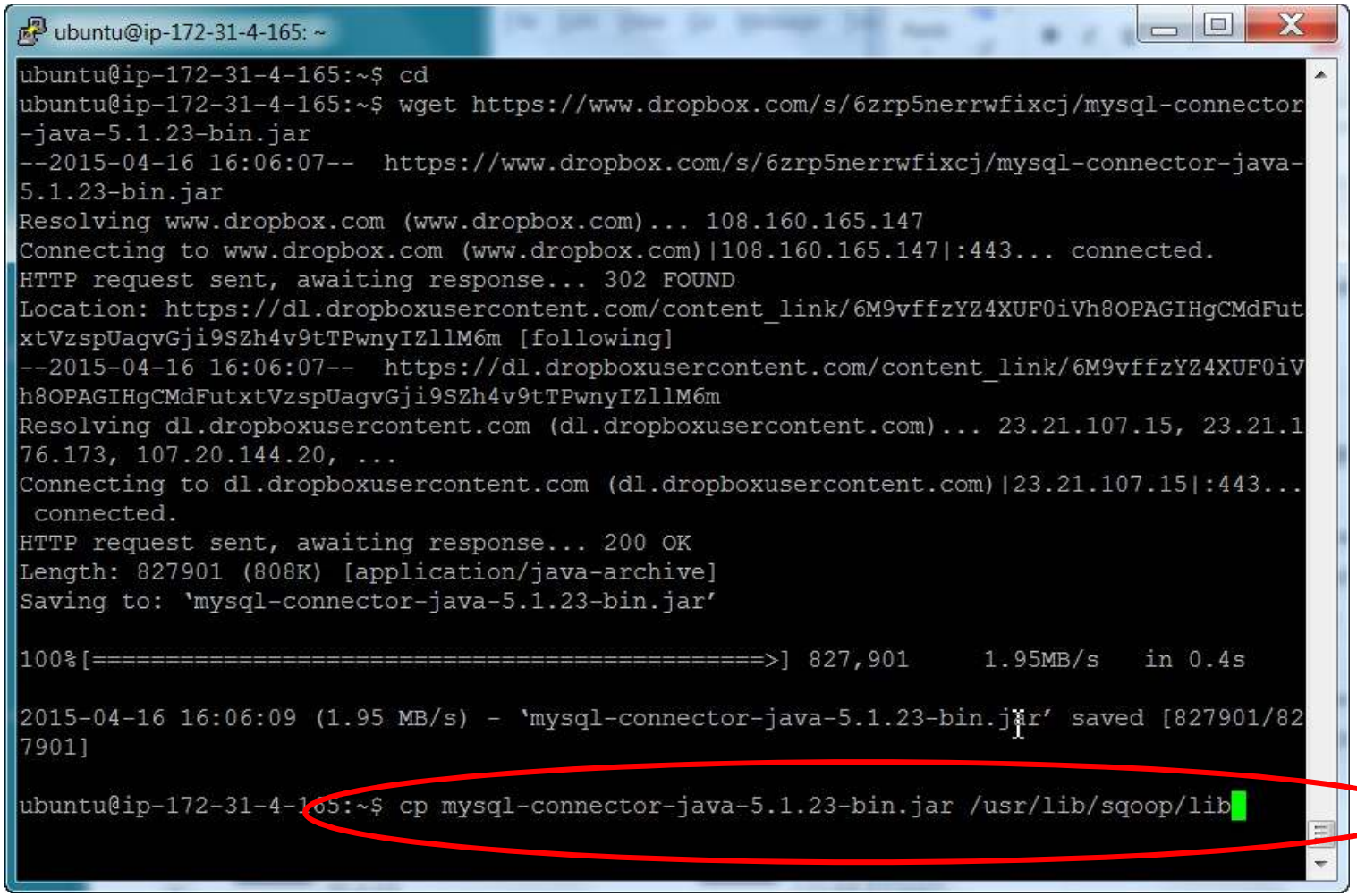
4. Installing DB driver for Sqoop

Install MySQL DB Driver

wget https://www.dropbox.com/s/6zrp5nerrwfixcj/mysql-connector-java-5.1.23-bin.jar



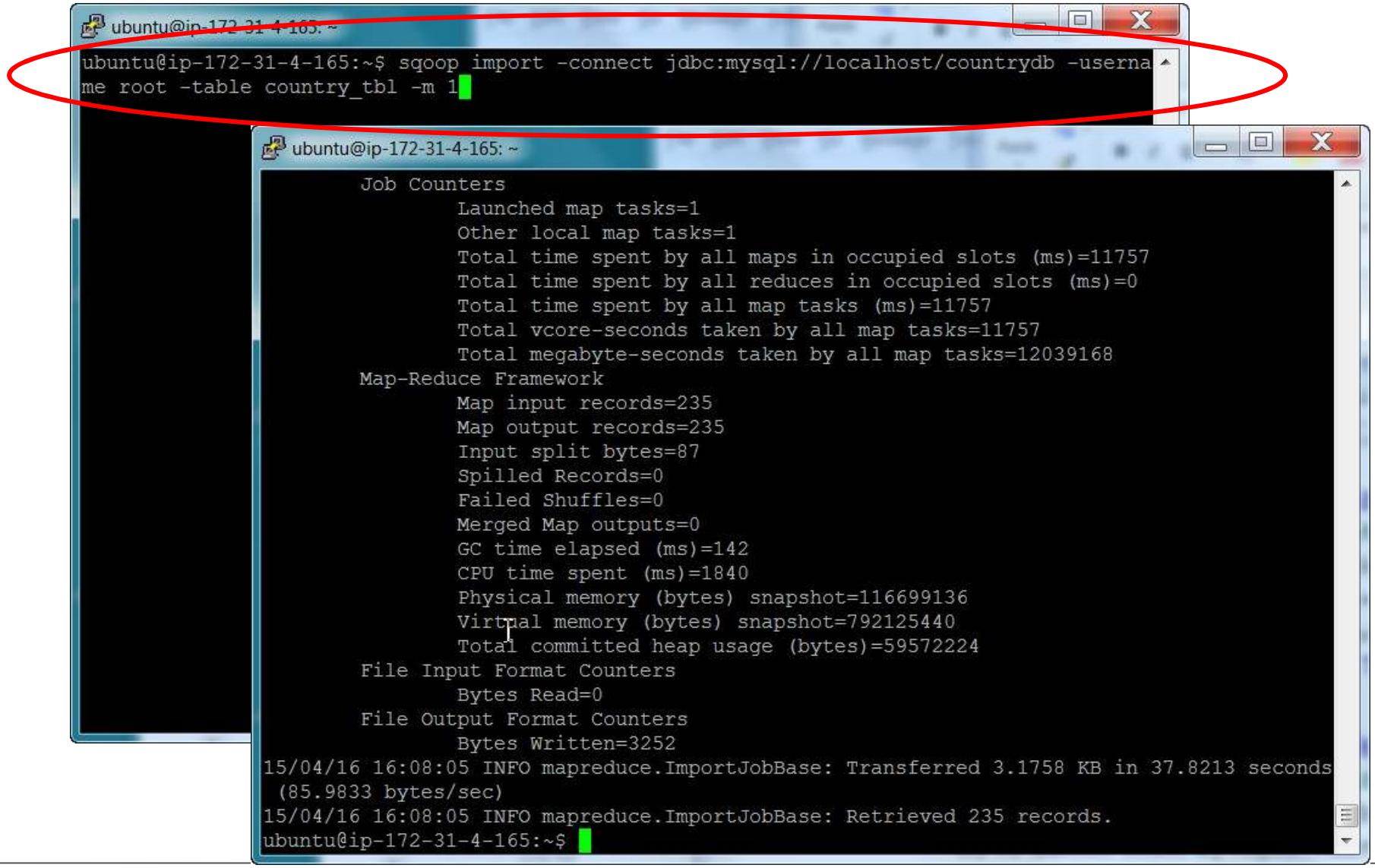
Install MySQL DB Driver



5. Importing data from MySQL to Hadoop

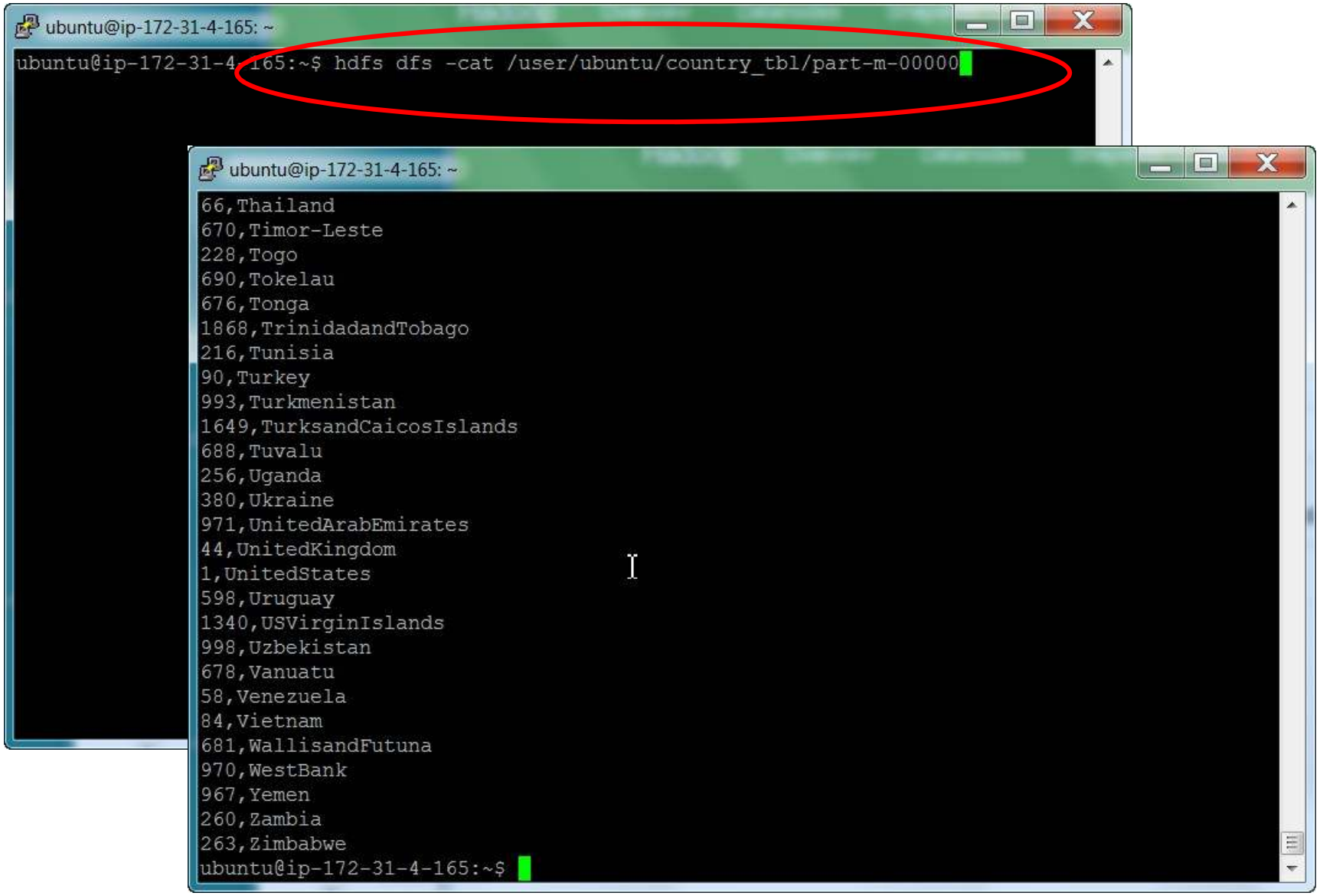
Sqoop Import Command

sqoop import -connect jdbc:mysql://localhost/countrydb -username root -table country_tbl -m 1

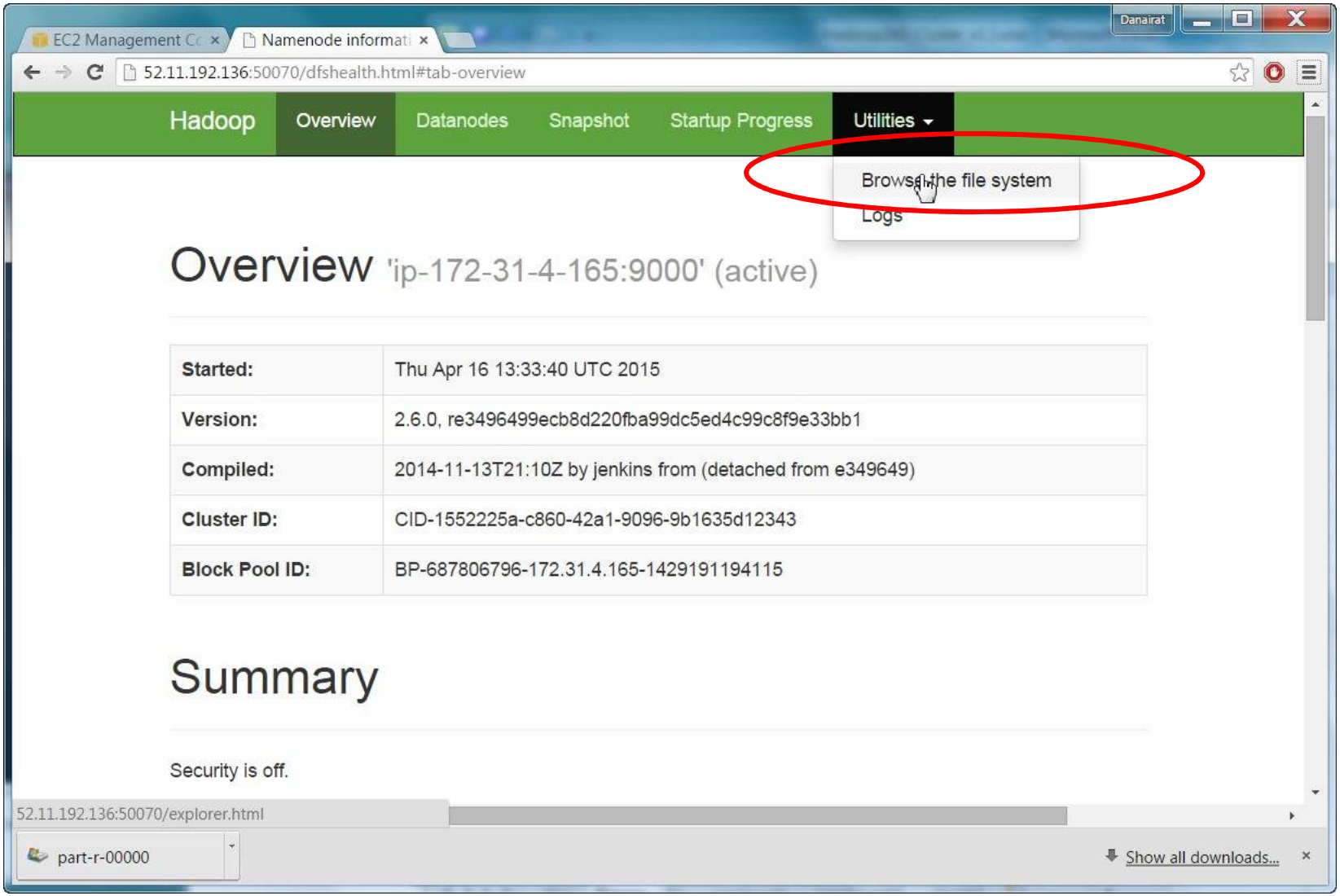


6. Reviewing HDFS data

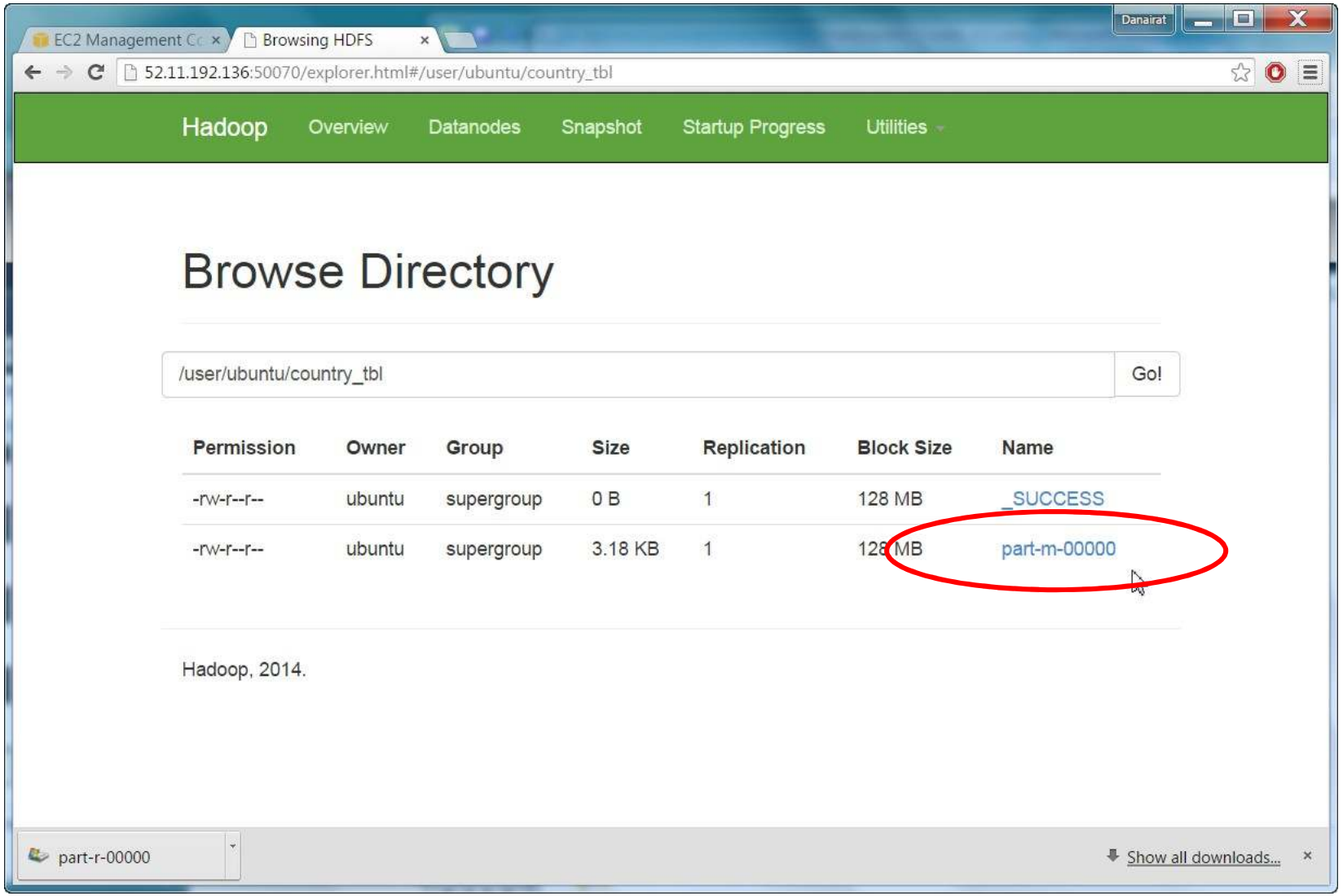
Review HDFS result data



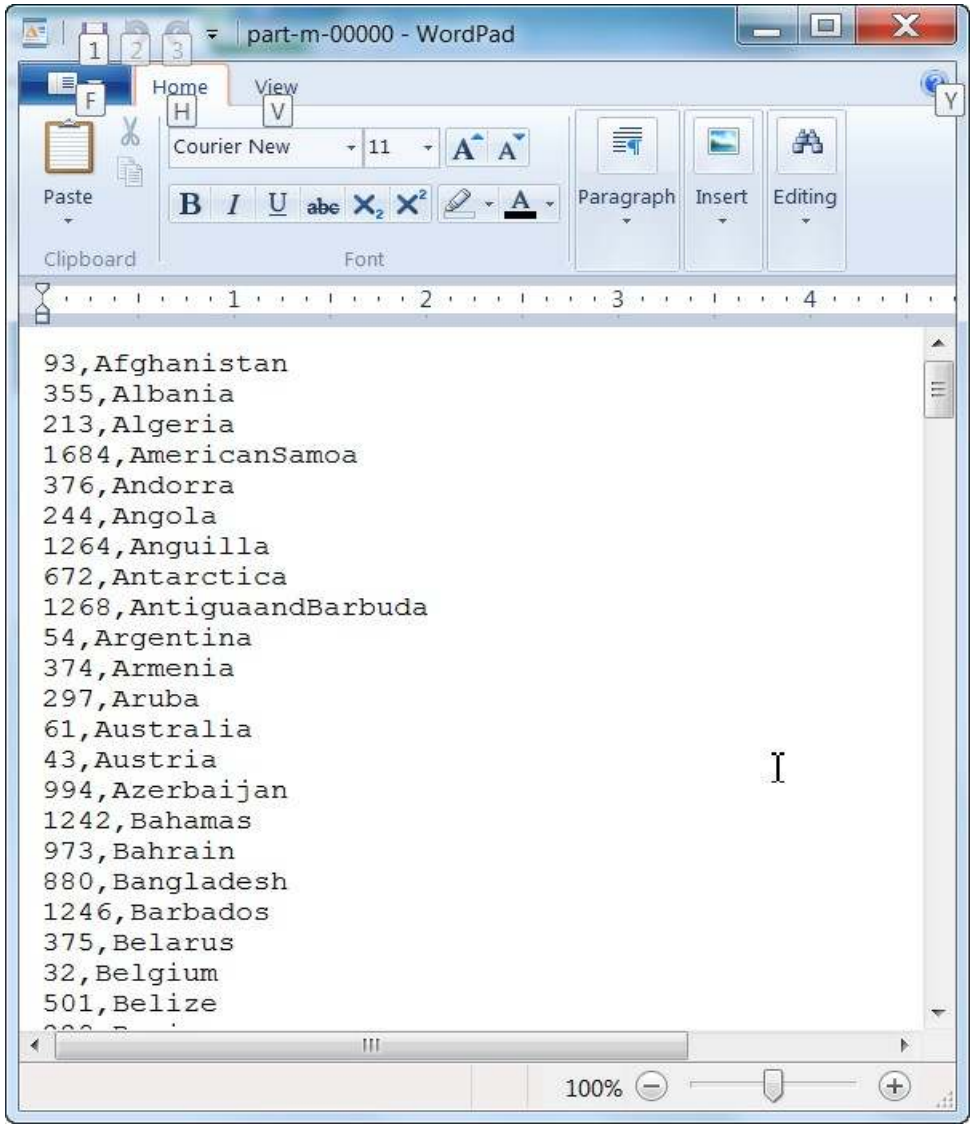
Review HDFS result data



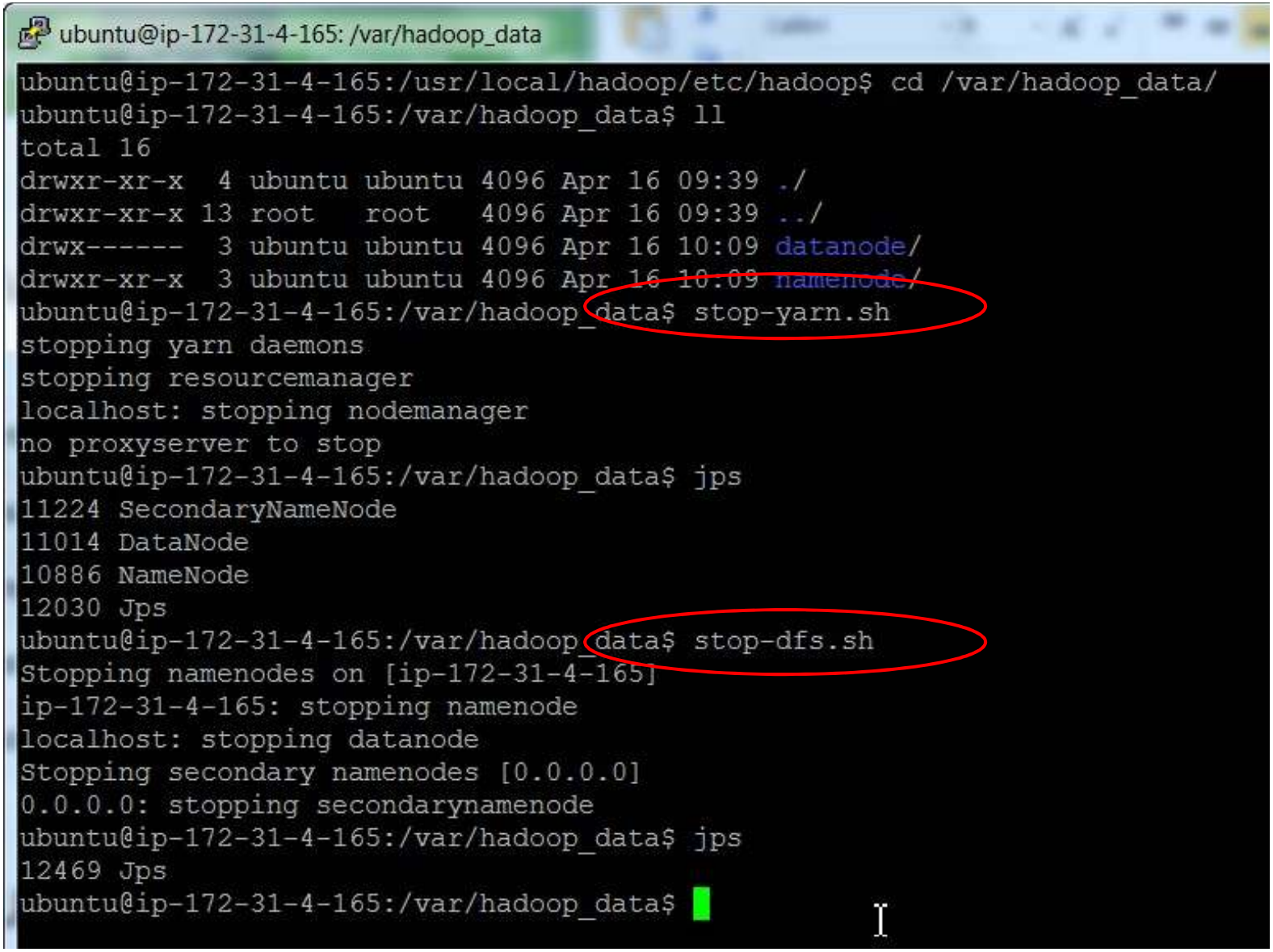
Review HDFS result data



Review HDFS result data



Stop Hadoop Yarn and HDFS



Recommendation to Further Study

Thank you very much