Big Data Processing Using Cloudera Quickstart with a Docker Container

June 2016

Dr.Thanachart Numnonda
IMC Institute
thanachart@imcinstitute.com

Modifiy from Original Version by Danairat T.
Certified Java Programmer, TOGAF – Silver
danairat@gmail.com

Outline

- Launch AWS EC2 Instance
- Install Docker on Ubuntu
- Pull Cloudera QuickStart to the docker
- HDFS
- Hive
- Pig
- Impala
- Spark
- Spark SQL
- Spark Streaming

Hive.apache.org

Cloudera VM

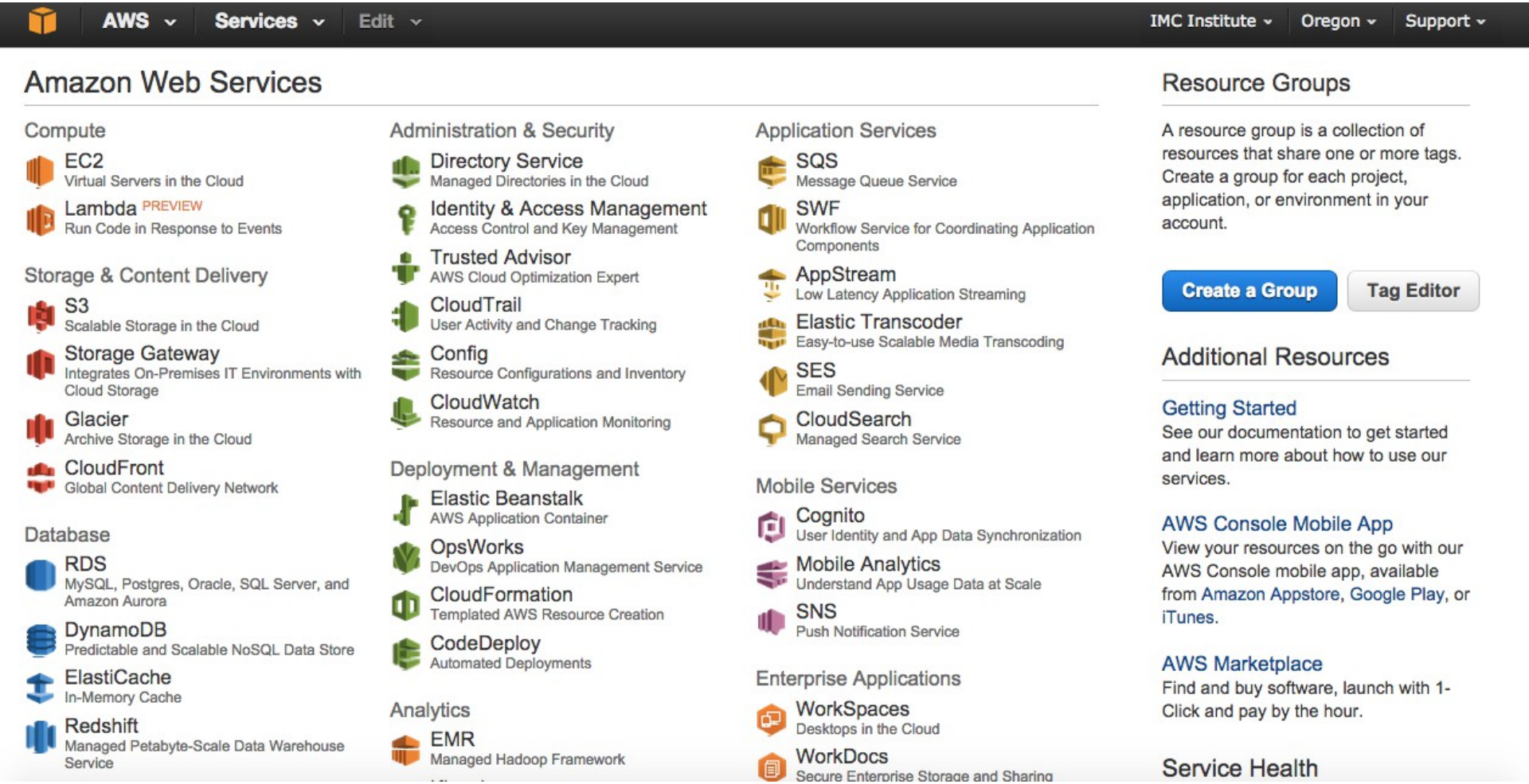
This lab will use a EC2 virtual server on AWS to install Cloudera, However, you can also use Cloudera QuickStart VM which can be downloaded from:

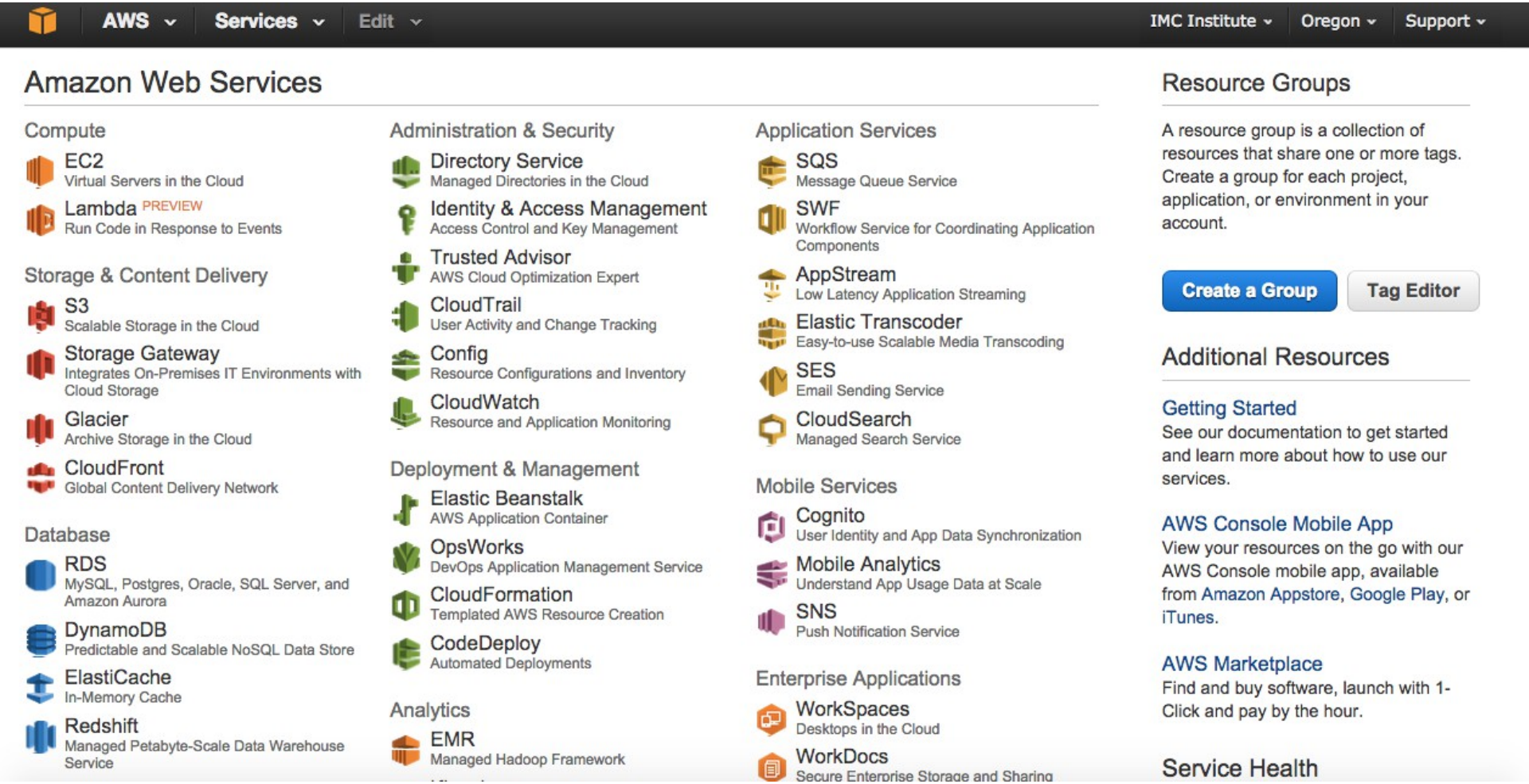
http://www.cloudera.com/content/www/en-us/downloads.html

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

(Note: You can skip this session if you use your own computer or another cloud service)

Thanachart Numnonda, thanachart@imcinstitute.com June 2016


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
AWS Application Container
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
Secure Enterprise Storage and Sharing
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

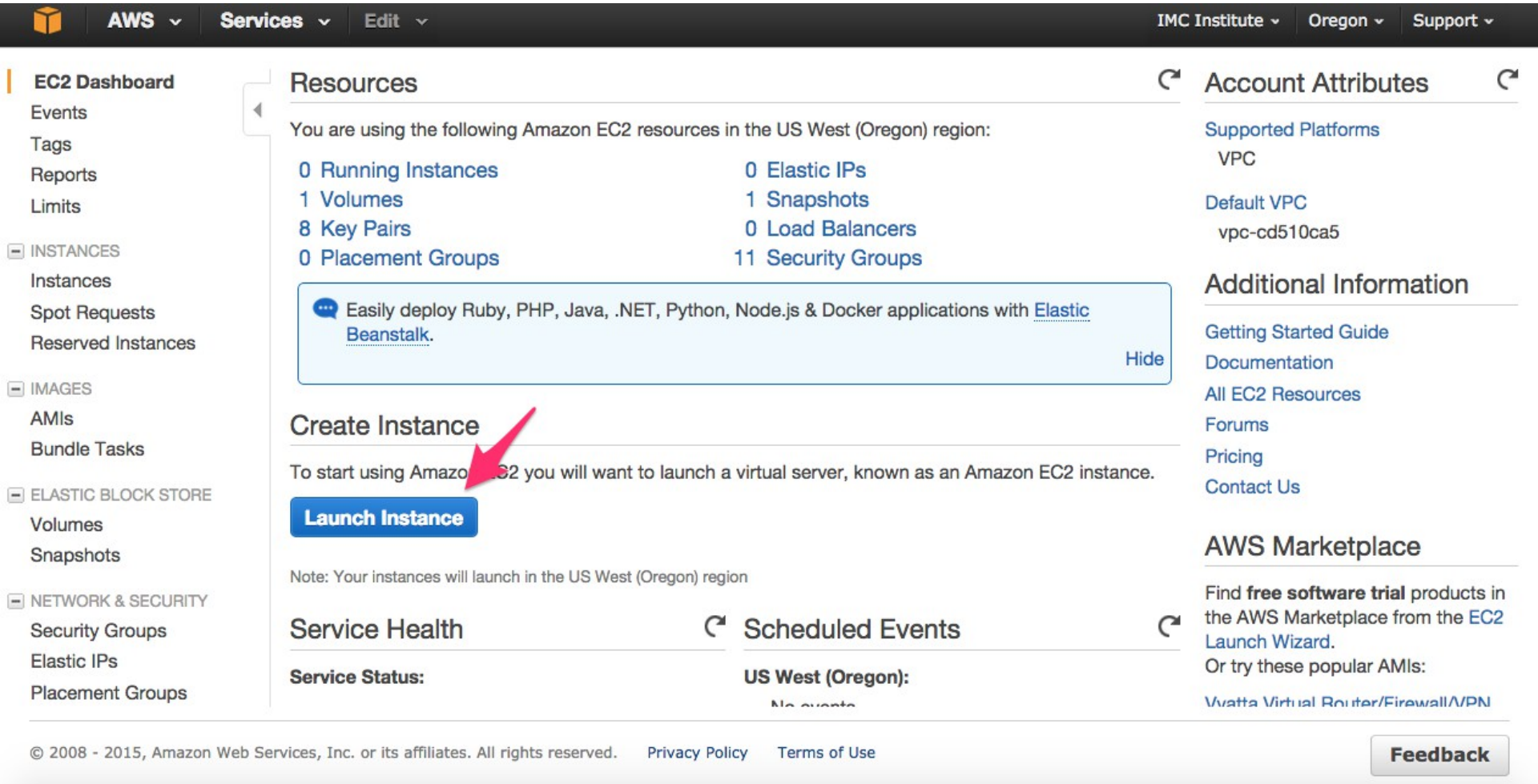

AWS
Services
Edit
IMC Institute
Oregon
Support
Amazon Web Services
Compute
EC2
Virtual Servers in the Cloud
Lambda PREVIEW
Run Code in Response to Events
Storage & Content Delivery
S3
Scalable Storage in the Cloud
Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage
Glacier
Archive 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
AWS Application Container
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
Secure Enterprise Storage and Sharing
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

Virtual Server

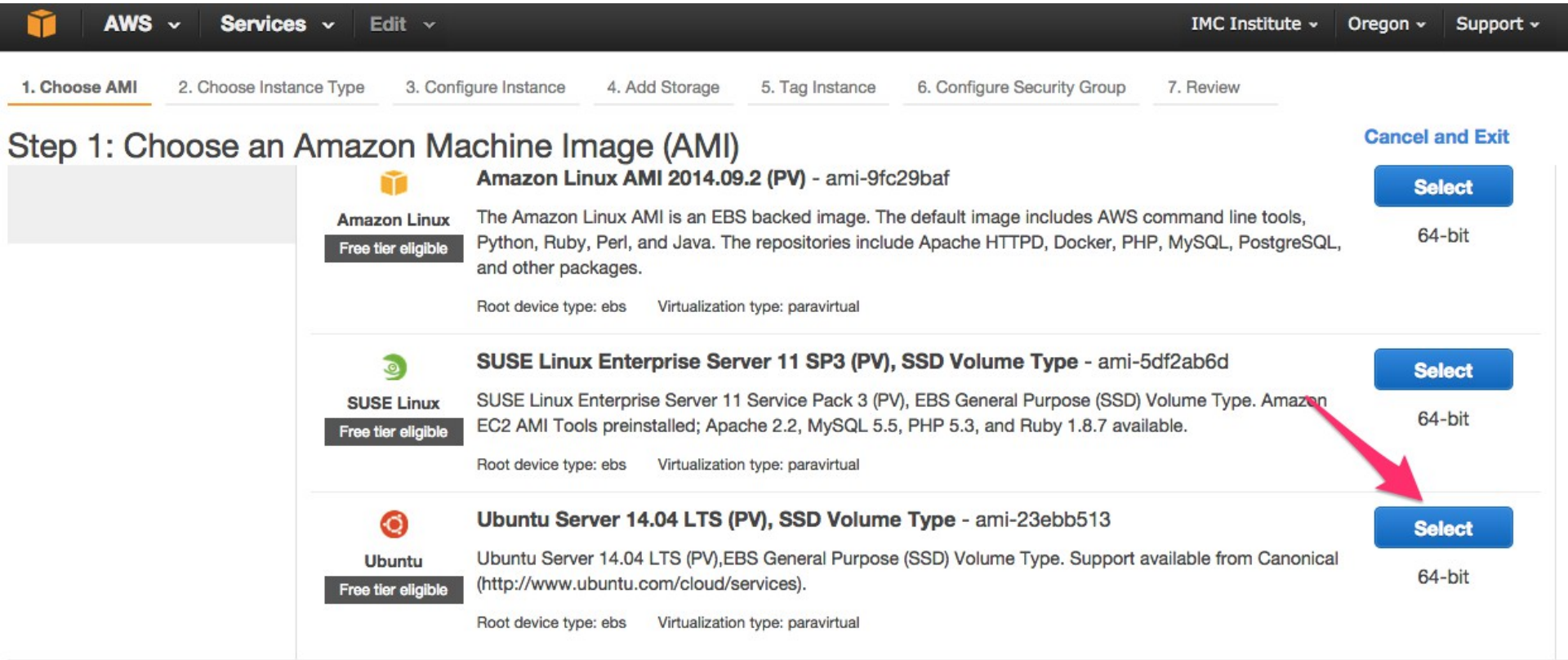
This lab will use a EC2 virtual server to install a Cloudera Cluster using the following features:

- Ubuntu Server 14.04 LTS
- Four m3.xLarge 4vCPU, 15 GB memory,80 GB SSD
- Security group: default
- Keypair: imchadoop

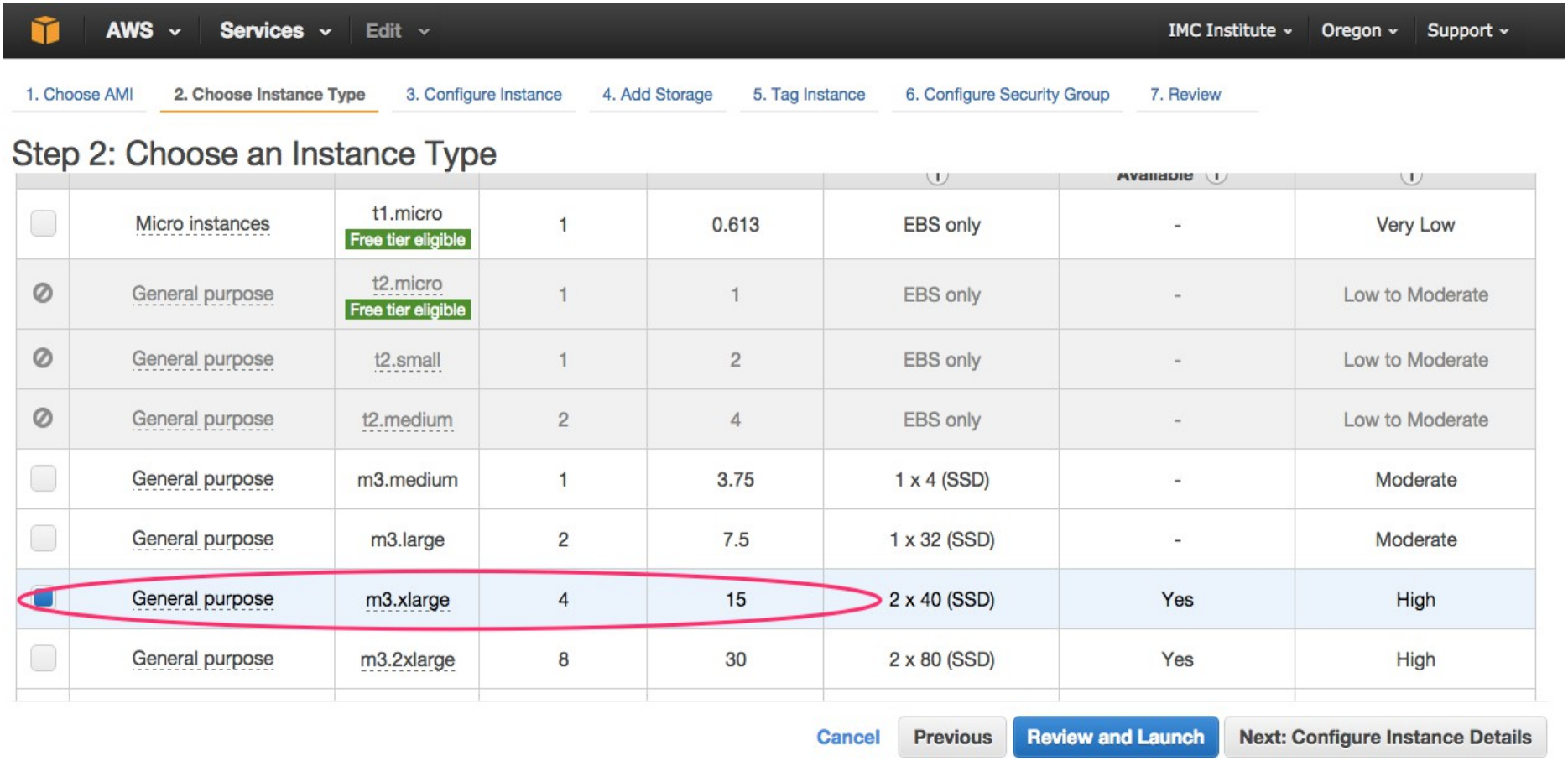
Select a EC2 service and click on Lunch Instance

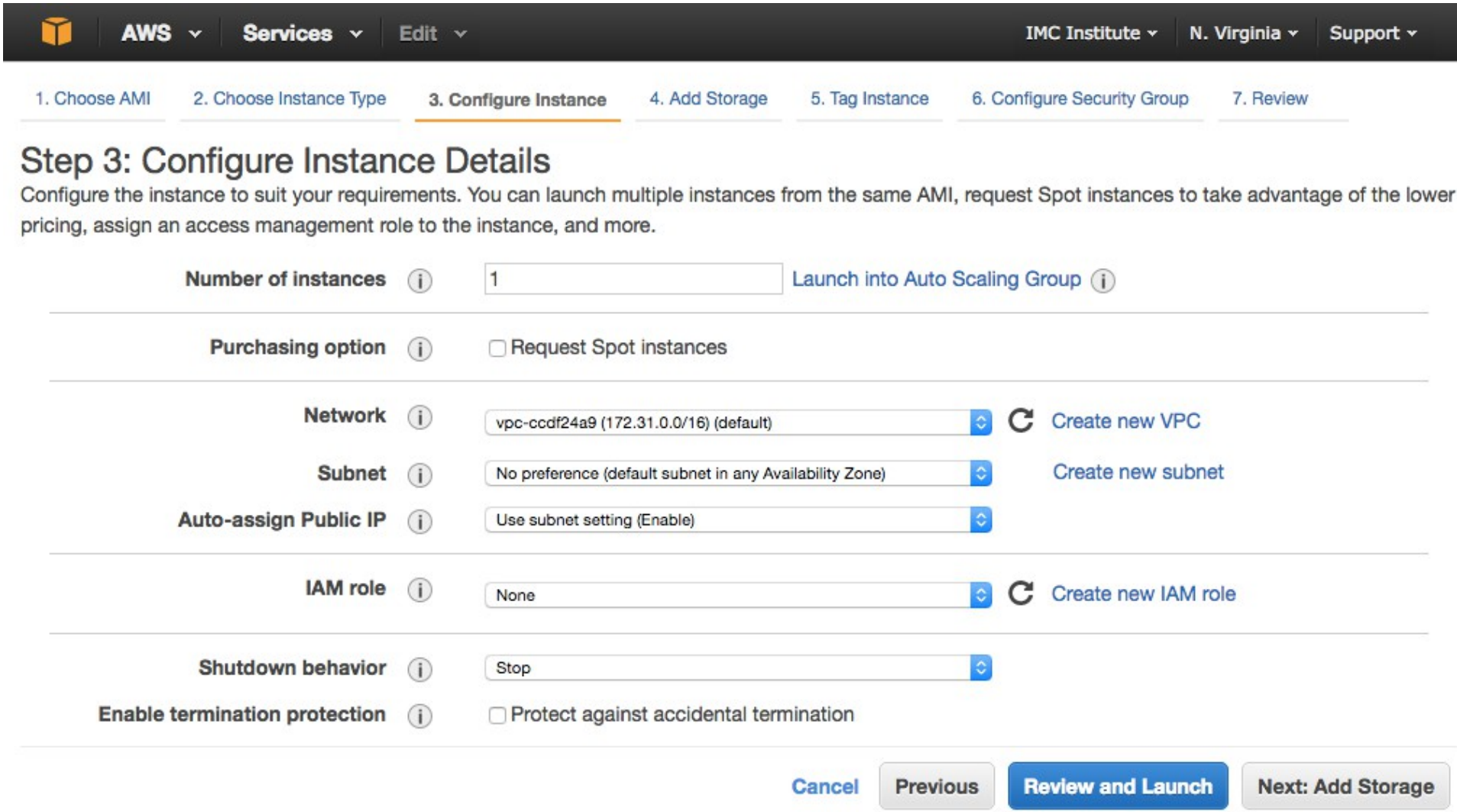


Select an Amazon Machine Image (AMI) and Ubuntu Server 14.04 LTS (PV)

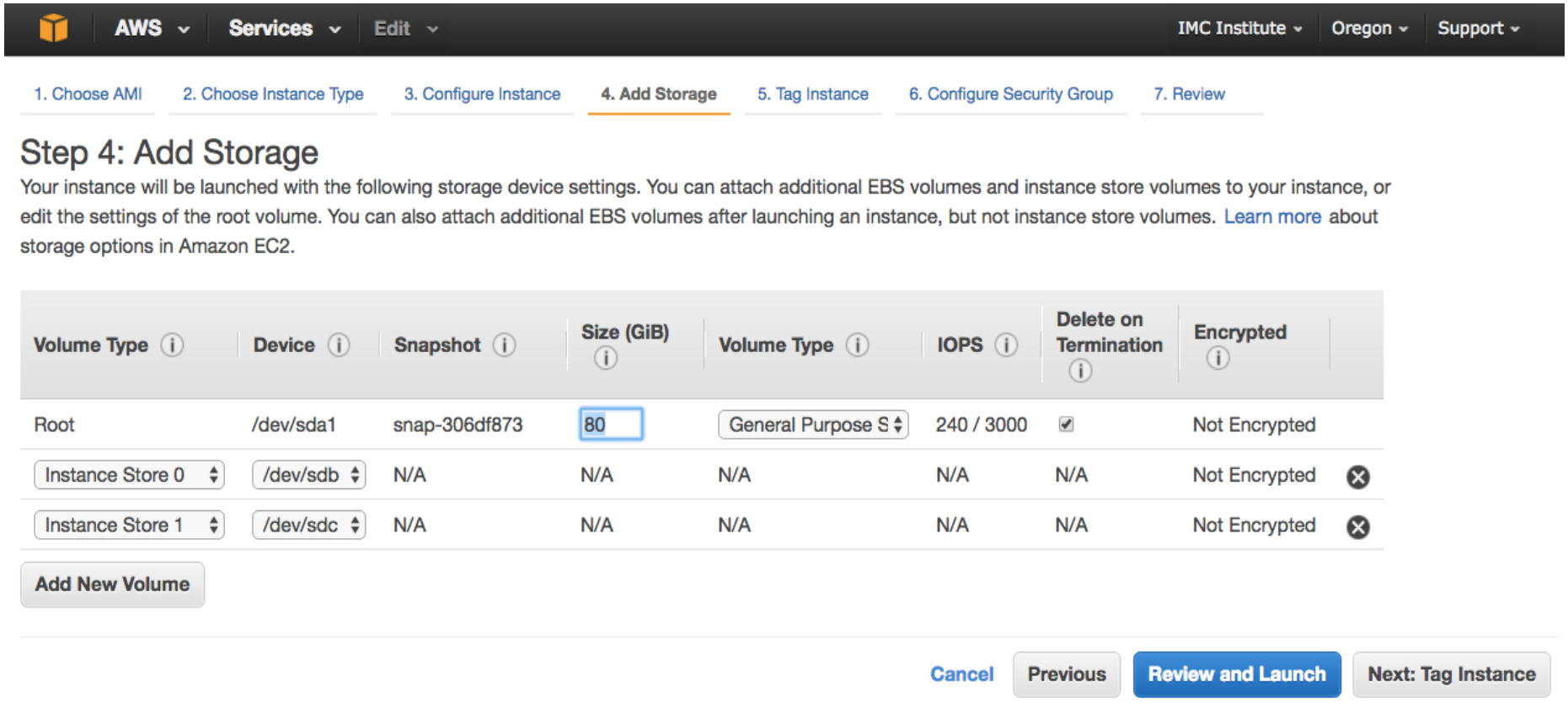


Choose m3.xlarge Type virtual server

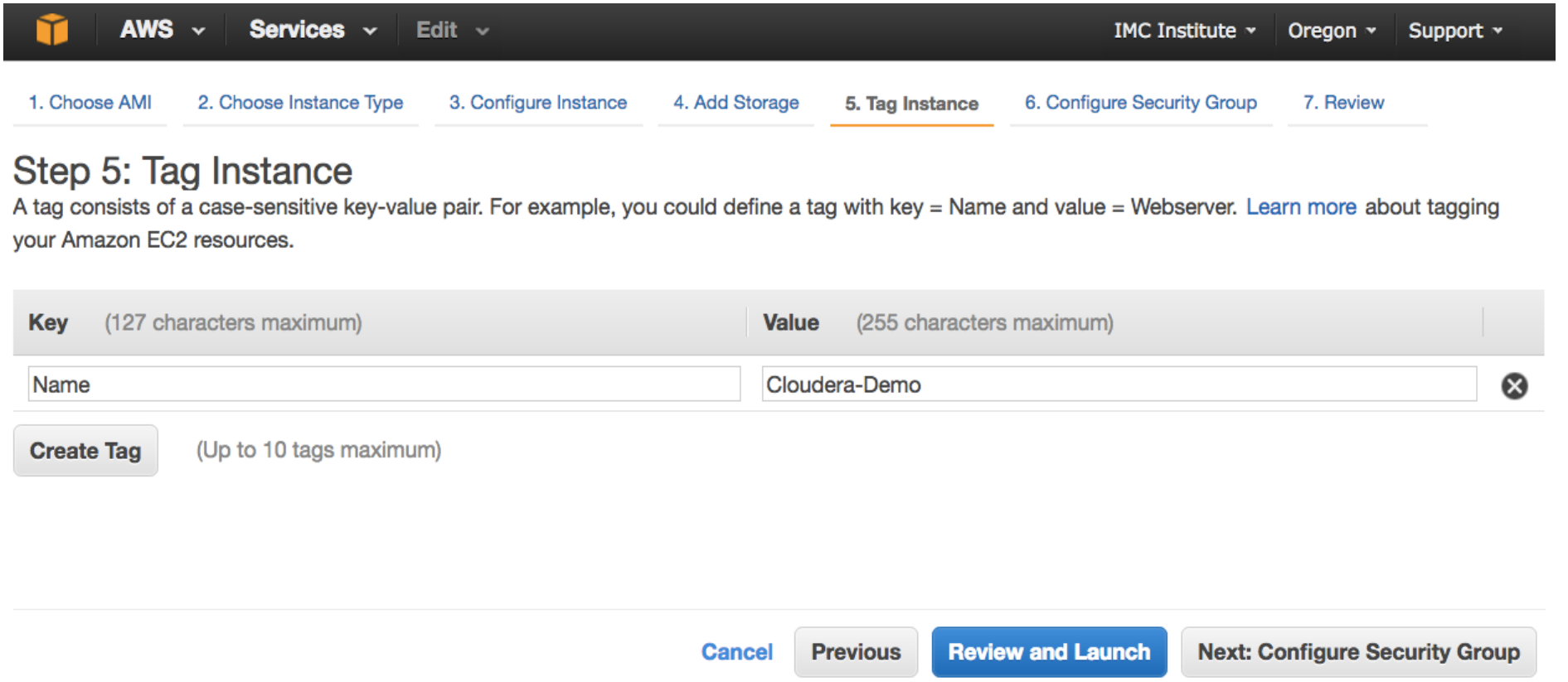



AWS
Services
Edit
IMC Institute
N. Virginia
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances
1
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Network
vpc-ccdf24a9 (172.31.0.0/16) (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zone)
Create new subnet
Auto-assign Public IP
Use subnet setting (Enable)
IAM role
None
Create new IAM role
Shutdown behavior
Stop
Enable termination protection
Protect against accidental termination
Cancel
Previous
Review and Launch
Next: Add Storage

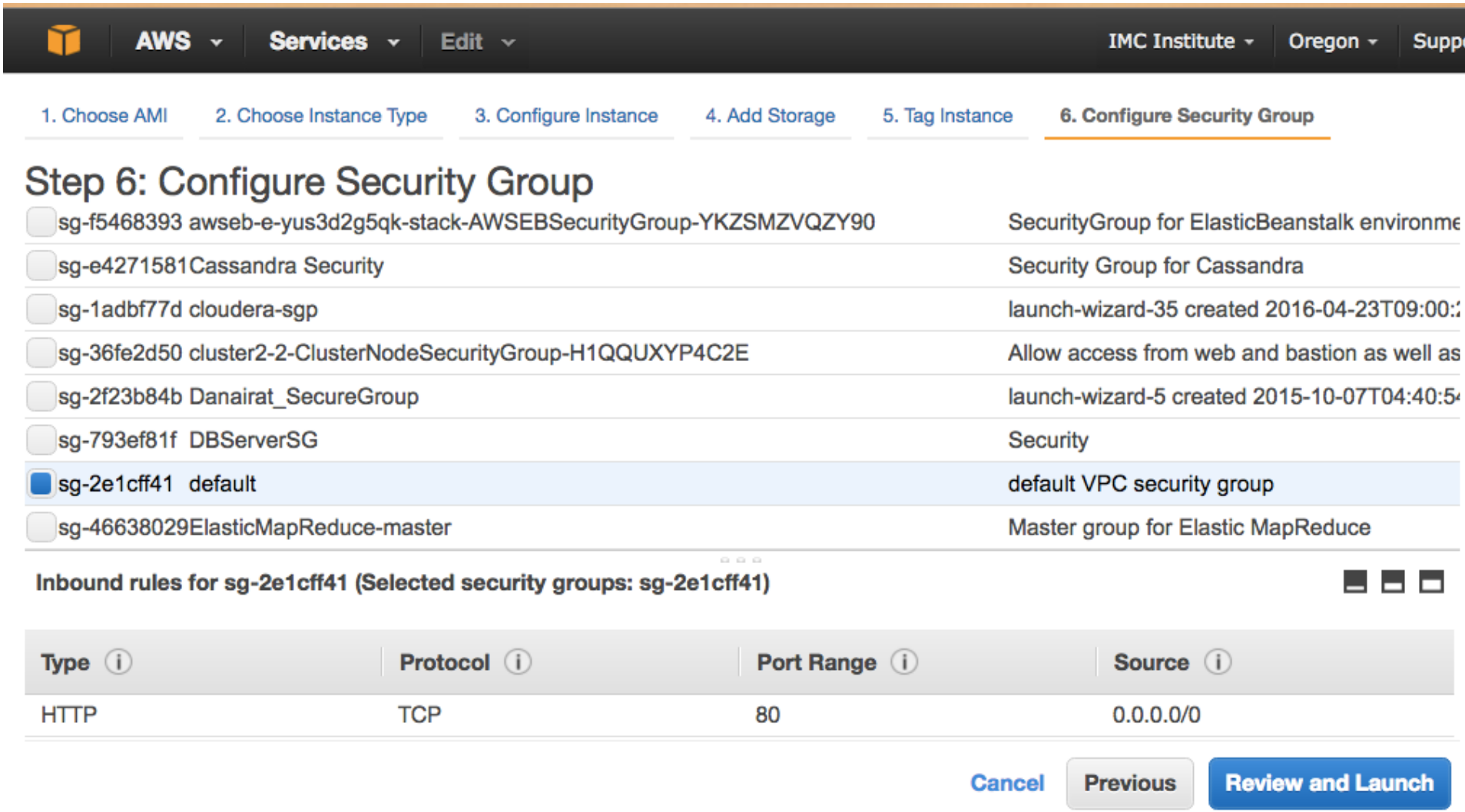
Add Storage: 80 GB



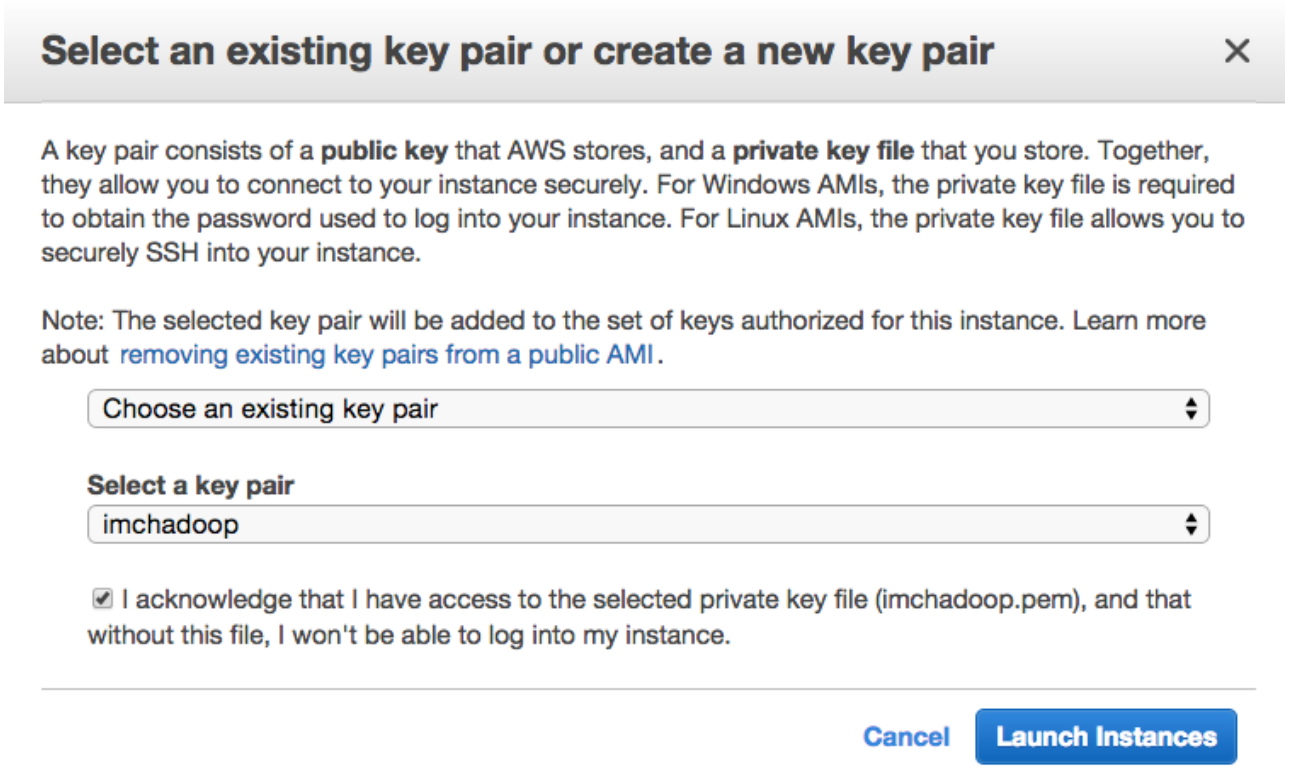
Name the instance



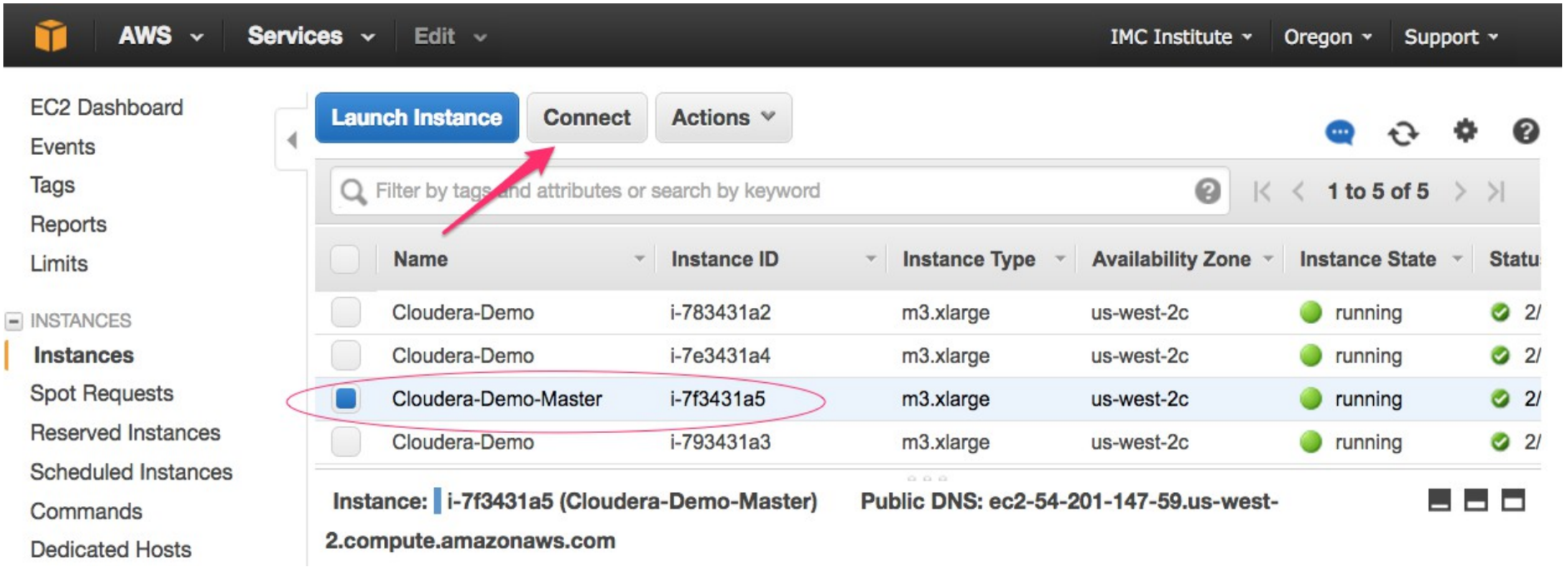
Select Create an existing security group > Default



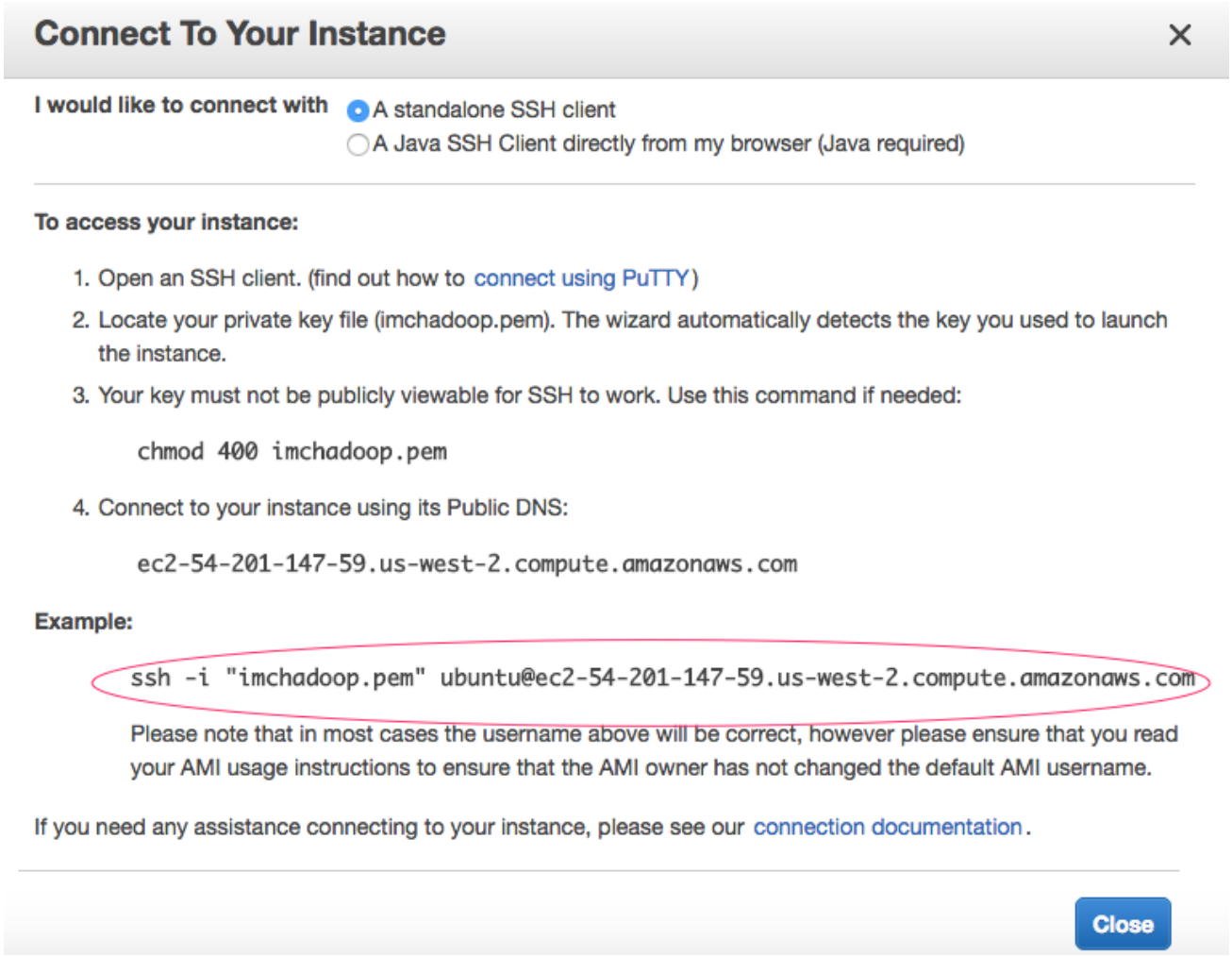
Click Launch and choose imchadoop as a key pair



Review an instance and rename one instance as a master / click Connect for an instruction to connect to the instance



Connect to an instance from Mac/Linux



Connect To Your Instance

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

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (imchadoop.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 imchadoop.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-54-201-147-59.us-west-2.compute.amazonaws.com
```

Example:

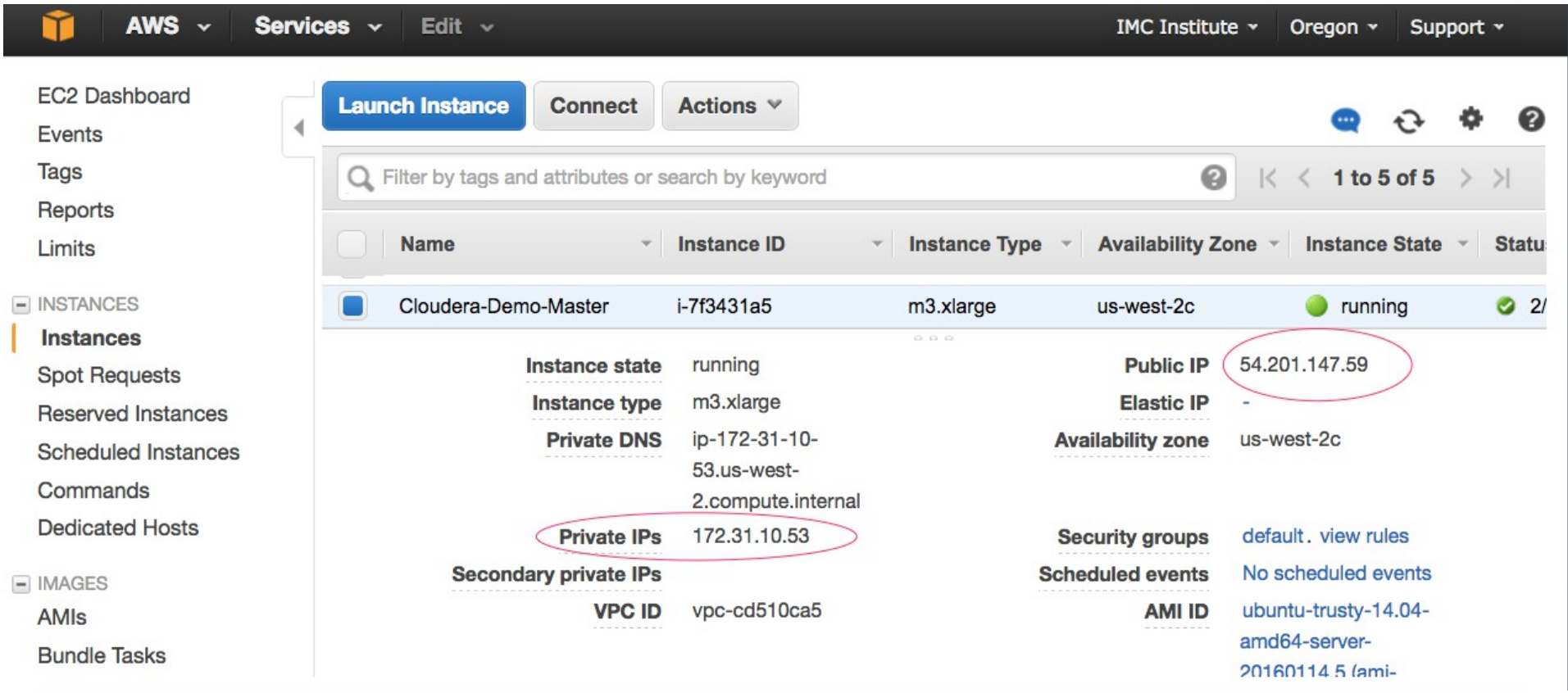
```
ssh -i "imchadoop.pem" ubuntu@ec2-54-201-147-59.us-west-2.compute.amazonaws.com
```

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

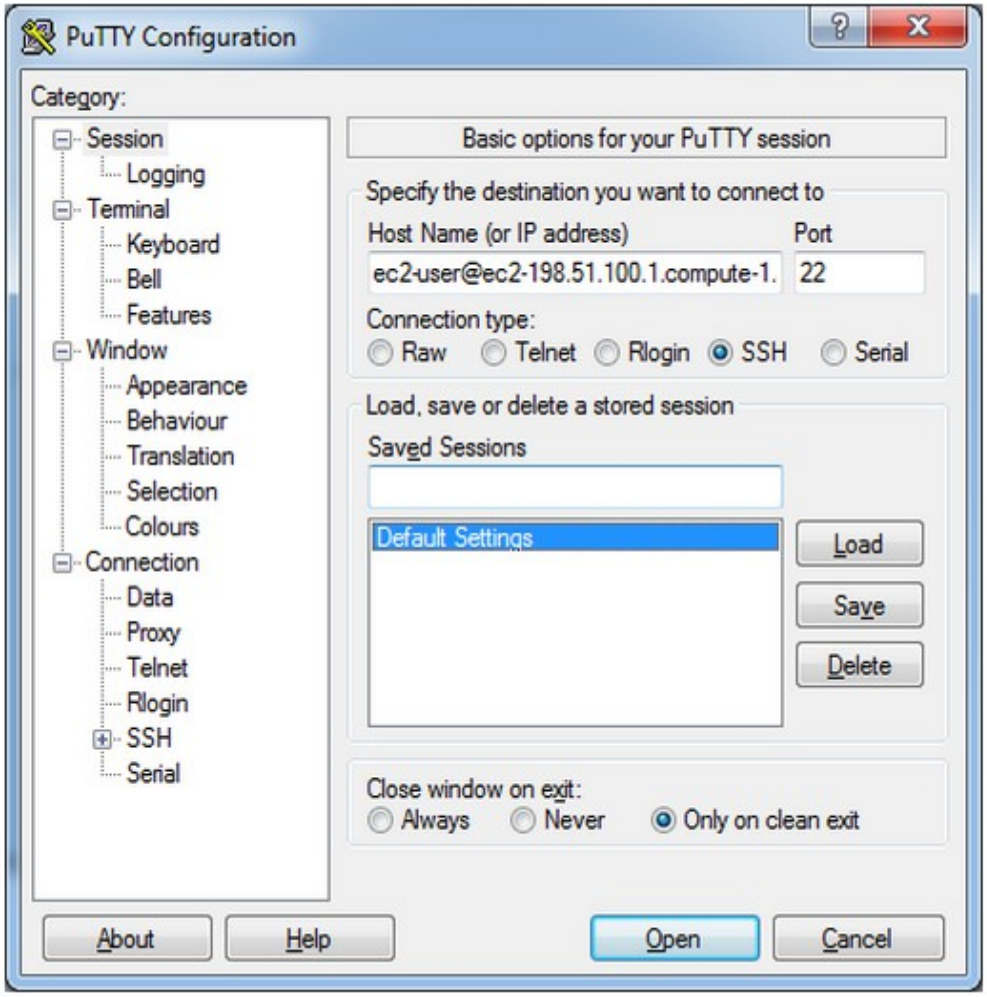
If you need any assistance connecting to your instance, please see our connection documentation.

Close

Can also view details of the instance such as Public IP and Private IP



Connect to an instance from Windows using Putty



Connect to the instance

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

_____________________________________________________________________
WARNING! Your environment specifies an invalid locale.
 This can affect your user experience significantly, including the
 ability to manage packages. You may install the locales by running:

   sudo apt-get install language-pack-UTF-8
     or
   sudo locale-gen UTF-8

To see all available language packs, run:
   apt-cache search "^language-pack-[a-z][a-z]$"
To disable this message for all users, run:
   sudo touch /var/lib/cloud/instance/locale-check.skip
_____________________________________________________________________

ubuntu@ip-172-31-1-242:~$
```

Hands-On: Installing Cloudera Quickstart on Docker Container

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Installation Steps

- Update OS
- Install Docker
- Pull Cloudera Quickstart
- Run Cloudera Quickstart
- Run Cloudera Manager

Hive.apache.org

Update OS (Ubuntu)

- Command: sudo apt-get update

```
ubuntu@ip-172-31-30-238:~$ sudo apt-get update
Ign http://us-east-1.ec2.archive.ubuntu.com trusty InRelease
Get:1 http://us-east-1.ec2.archive.ubuntu.com trusty-updates InRelease [65.9 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com trusty-backports InRelease [65.9 kB]
Hit http://us-east-1.ec2.archive.ubuntu.com trusty Release.gpg
Hit http://us-east-1.ec2.archive.ubuntu.com trusty Release
Get:3 http://security.ubuntu.com trusty-security InRelease [65.9 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/main Sources [277 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/restricted Sources [535
2 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/universe Sources [156 k
B]
Get:7 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/multiverse Sources [593
9 B]
Get:8 http://us-east-1.ec2.archive.ubuntu.com trusty-updates/main amd64 Packages [78
1 kB]
```

Docker Installation

- Command: sudo apt-get install docker.io

```
ubuntu@ip-172-31-30-238:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  aufs-tools cgroup-lite git git-man liberror-perl
Suggested packages:
  btrfs-tools debootstrap lxc rinse git-daemon-run git-daemon-sysvinit git-doc
  git-el git-email git-gui gitk gitweb git-arch git-bzr git-cvs git-mediawiki
  git-svn
The following NEW packages will be installed:
  aufs-tools cgroup-lite docker.io git git-man liberror-perl
0 upgraded, 6 newly installed, 0 to remove and 84 not upgraded.
Need to get 8150 kB of archives.
After this operation, 51.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/ trusty/universe aufs-tools amd
64 1:3.2+20130722-1.1 [92.3 kB]
```

Pull Cloudera Quickstart

- Command: sudo docker pull cloudera/quickstart:latest

```
ubuntu@ip-172-31-30-238:~$ sudo docker pull cloudera/quickstart:latest
latest: Pulling from cloudera/quickstart
2cda82941cb7: Already exists
Digest: sha256:f91bee4cdfa2c92ea3652929a22f729d4d13fc838b00f120e630f91c941acb63
Status: Downloaded newer image for cloudera/quickstart:latest
ubuntu@ip-172-31-30-238:~$
```

Show docker images

- Command: sudo docker images

```
ubuntu@ip-172-31-30-238:~$ sudo docker images
REPOSITORY            TAG                 IMAGE ID            CREATED
  VIRTUAL SIZE
cloudera/quickstart   latest              2cda82941cb7        9 weeks ago
  6.336 GB
```

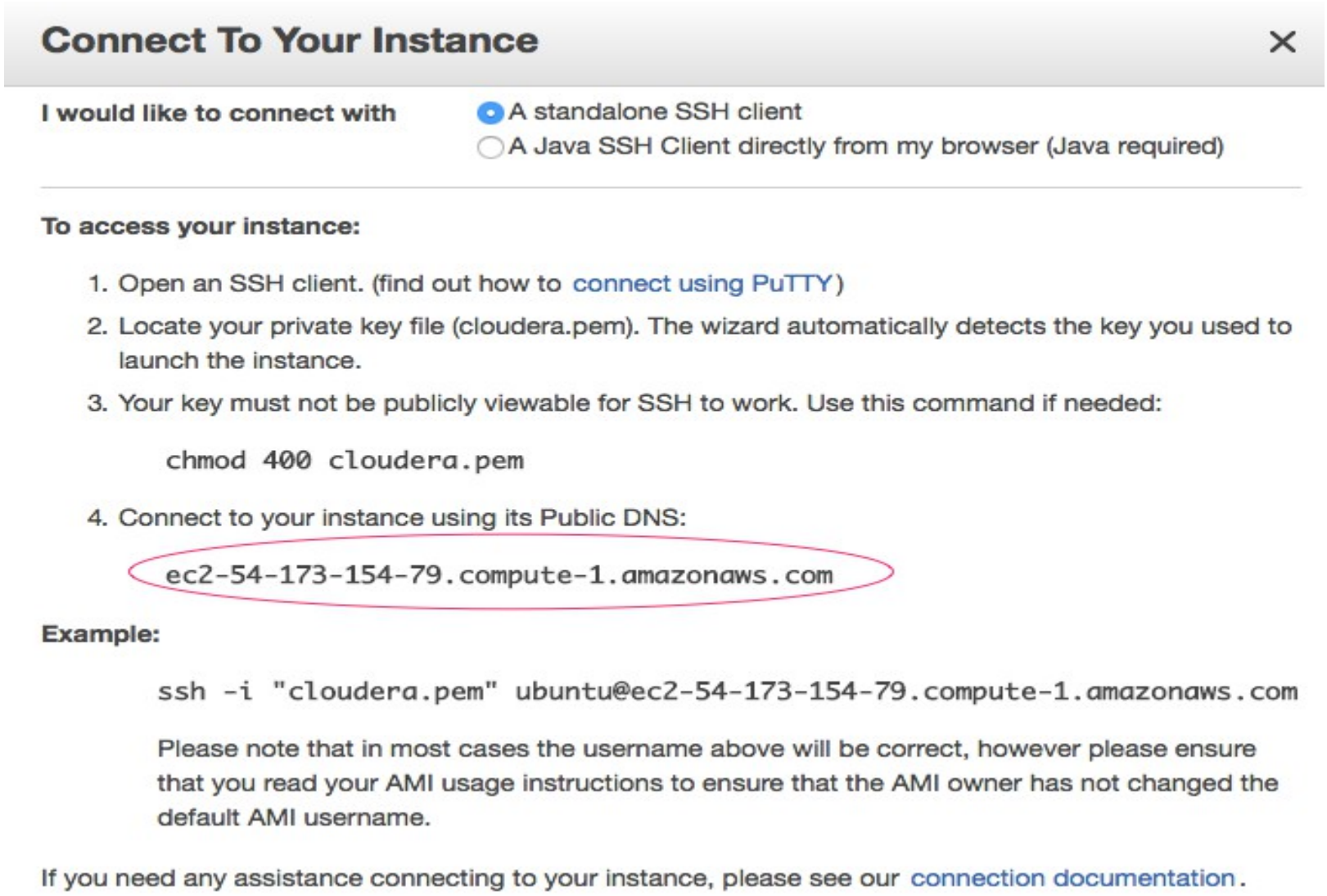
Run Cloudera quickstart

- Command: sudo docker run --hostname=quickstart.cloudera --privileged=true -t -i [OPTIONS] [IMAGE] /usr/bin/docker-quickstart

Example: sudo docker run --hostname=quickstart.cloudera --privileged=true -t -i -p 8888:8888 cloudera/quickstart /usr/bin/docker-quickstart

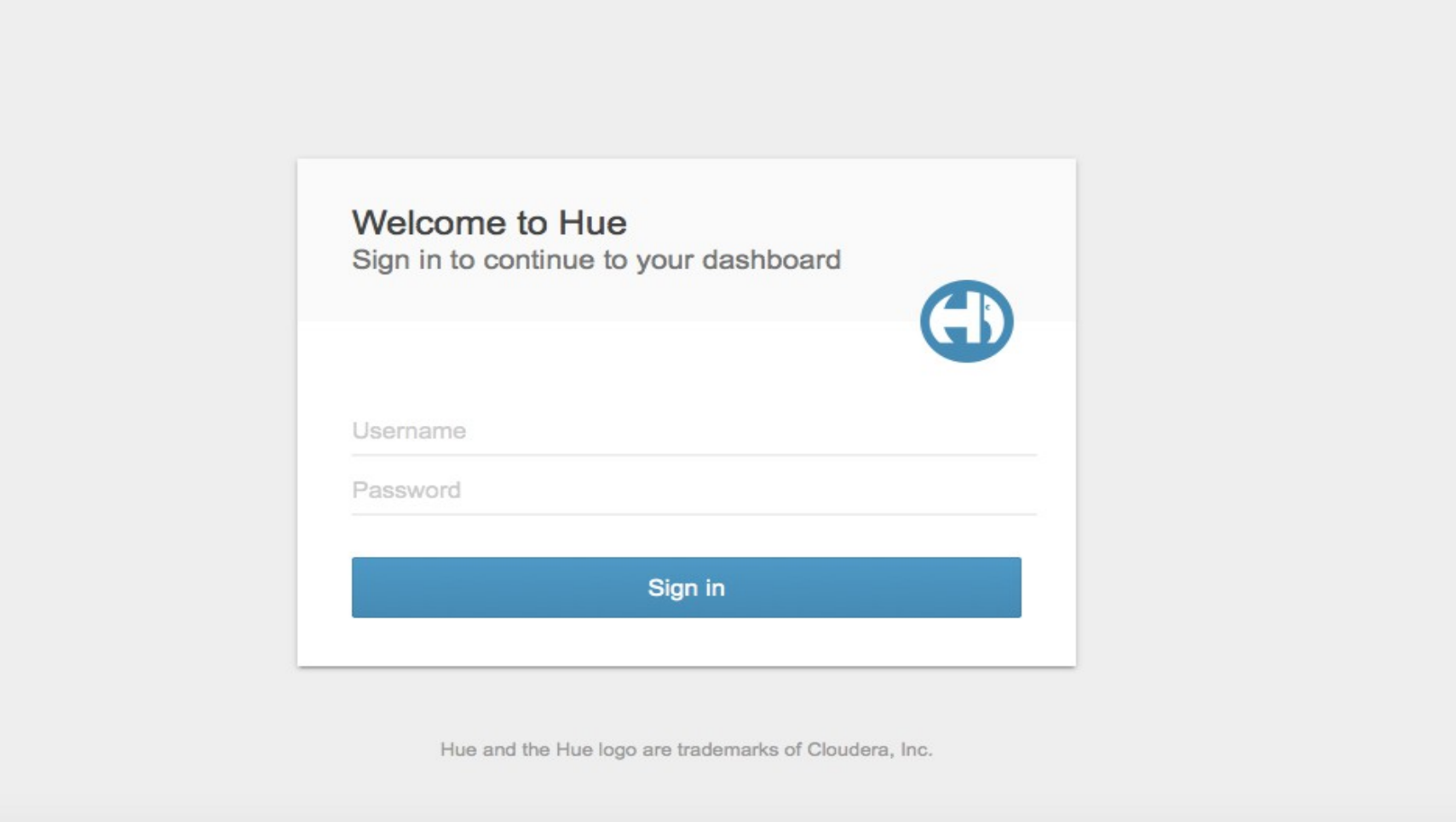
```
ubuntu@ip-172-31-30-238:~$  sudo docker run --hostname=quickstart.cloudera --pri
vileged=true -t -i  -p 8888:8888  -p 7180:7180  cloudera/quickstart  /usr/bin/do
cker-quickstart
Starting mysqld:                                           [  OK  ]

if [ "$1" == "start" ] ; then
    if [ "${EC2}" == 'true' ]; then
        FIRST_BOOT_FLAG=/var/lib/cloudera-quickstart/.ec2-key-installed
        if [ ! -f "${FIRST_BOOT_FLAG}" ]; then
            METADATA_API=http://169.254.169.254/latest/meta-data
            KEY_URL=${METADATA_API}/public-keys/0/openssh-key
```

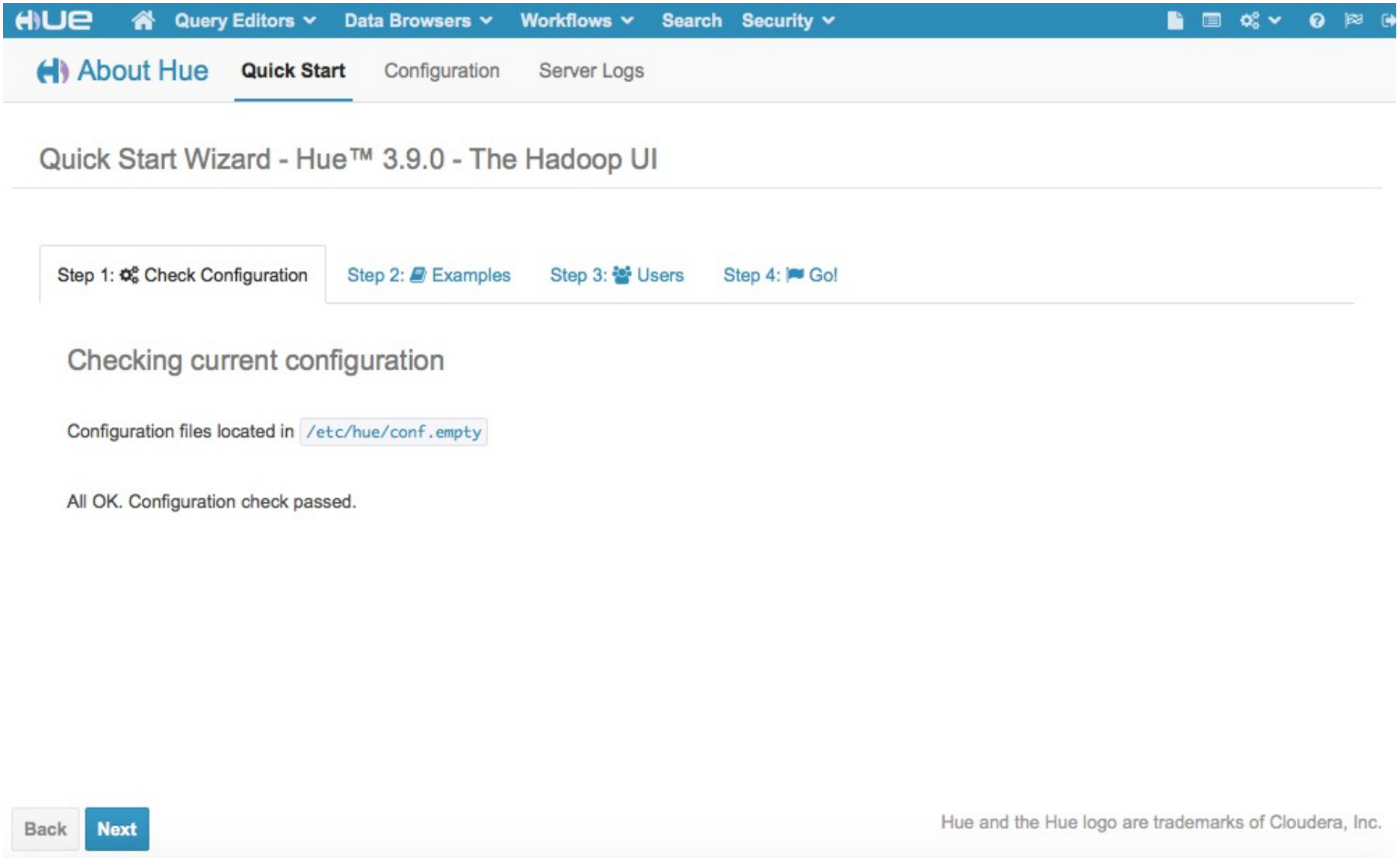
Finding the EC2 instance's DNS



Login to Hue

http://ec2-54-173-154-79.compute-1.amazonaws.com:8888




Query Editors
Data Browsers
Workflows
Search
Security
About Hue
Quick Start
Configuration
Server Logs
Quick Start Wizard - Hue™ 3.9.0 - The Hadoop UI
Step 1: Check Configuration
Step 2: Examples
Step 3: Users
Step 4: Go!
Checking current configuration
Configuration files located in /etc/hue/conf.empty
All OK. Configuration check passed.
Back
Next
Hue and the Hue logo are trademarks of Cloudera, Inc.

Hands-On: Importing/Exporting Data to HDFS

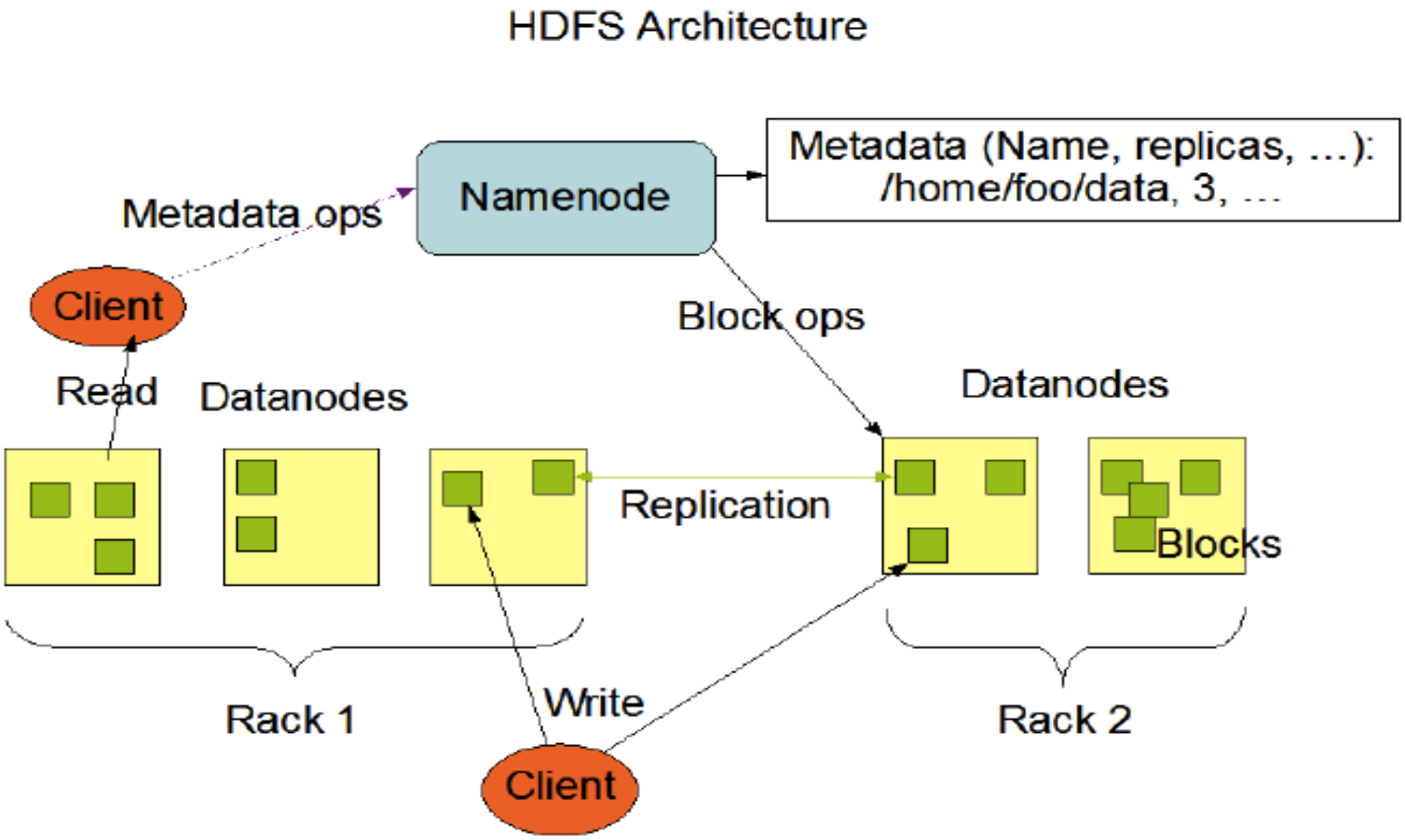
Thanachart Numnonda, thanachart@imcinstitute.com June 2016

HDFS

- Default storage for the Hadoop cluster
- Data is distributed and replicated over multiple machines
- Designed to handle very large files with straming data access patterns.
- NameNode/DataNode
- Master/slave architecture (1 master 'n' slaves)
- Designed for large files (64 MB default, but configurable) across all the nodes

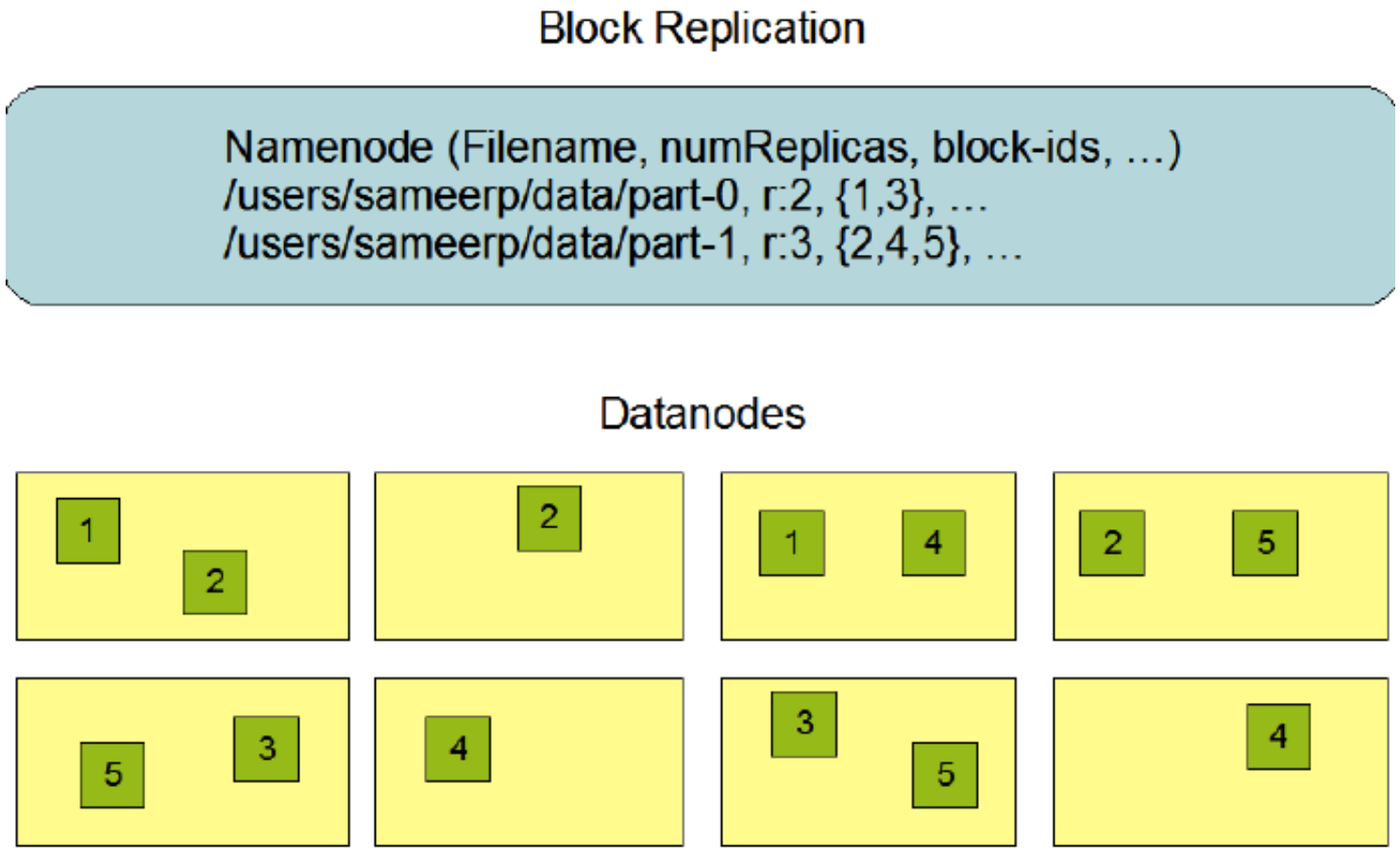
Hive.apache.org

HDFS Architecture



Source Hadoop: Shashwat Shriparv

Data Replication in HDFS

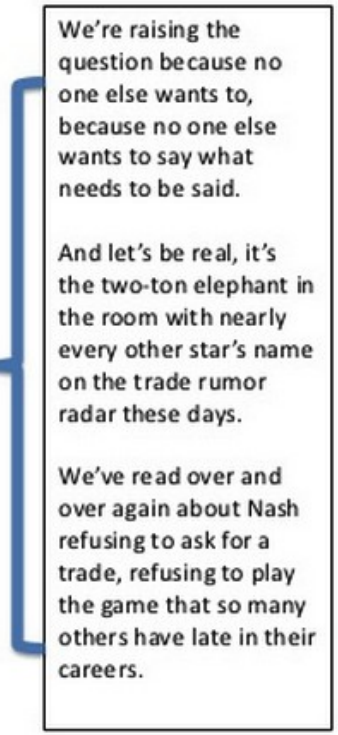


Source Hadoop: Shashwat Shriparv

How does HDFS work?

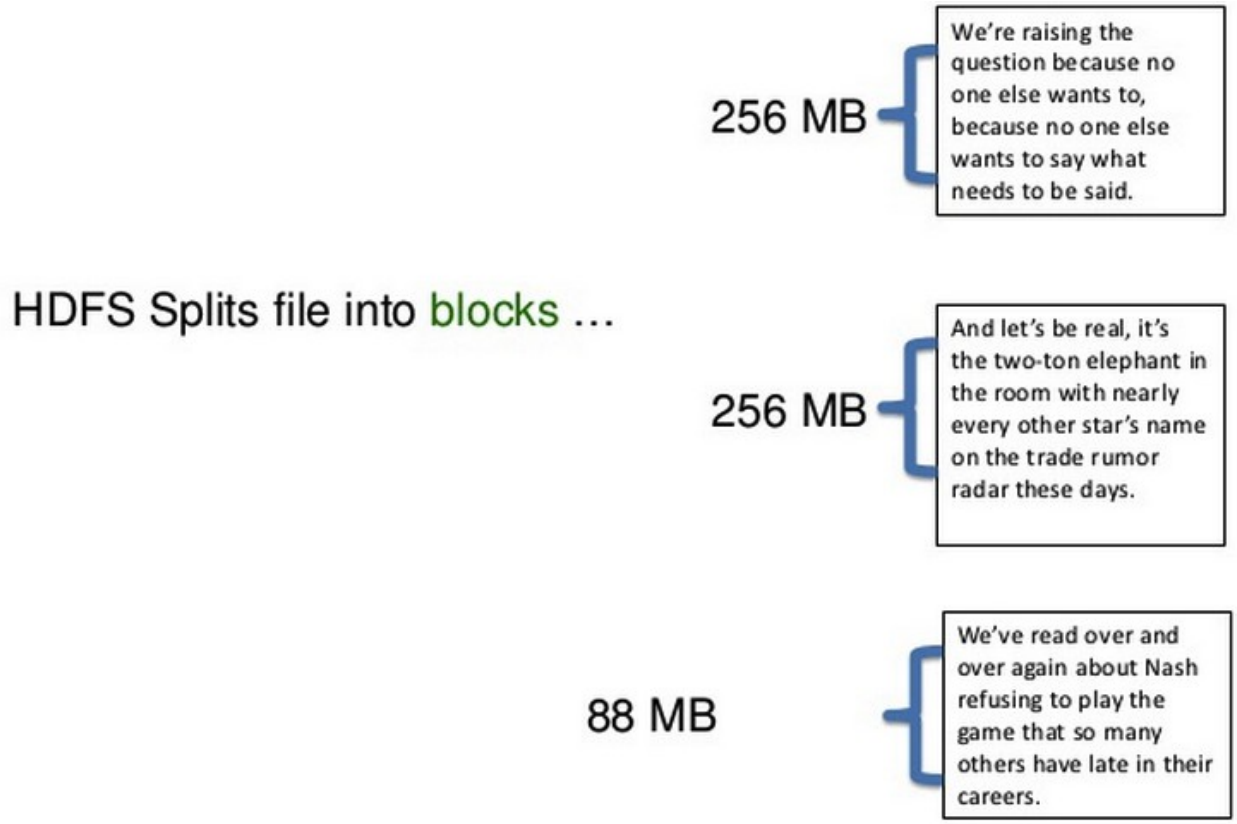
A file we want to store on HDFS …

600 MB



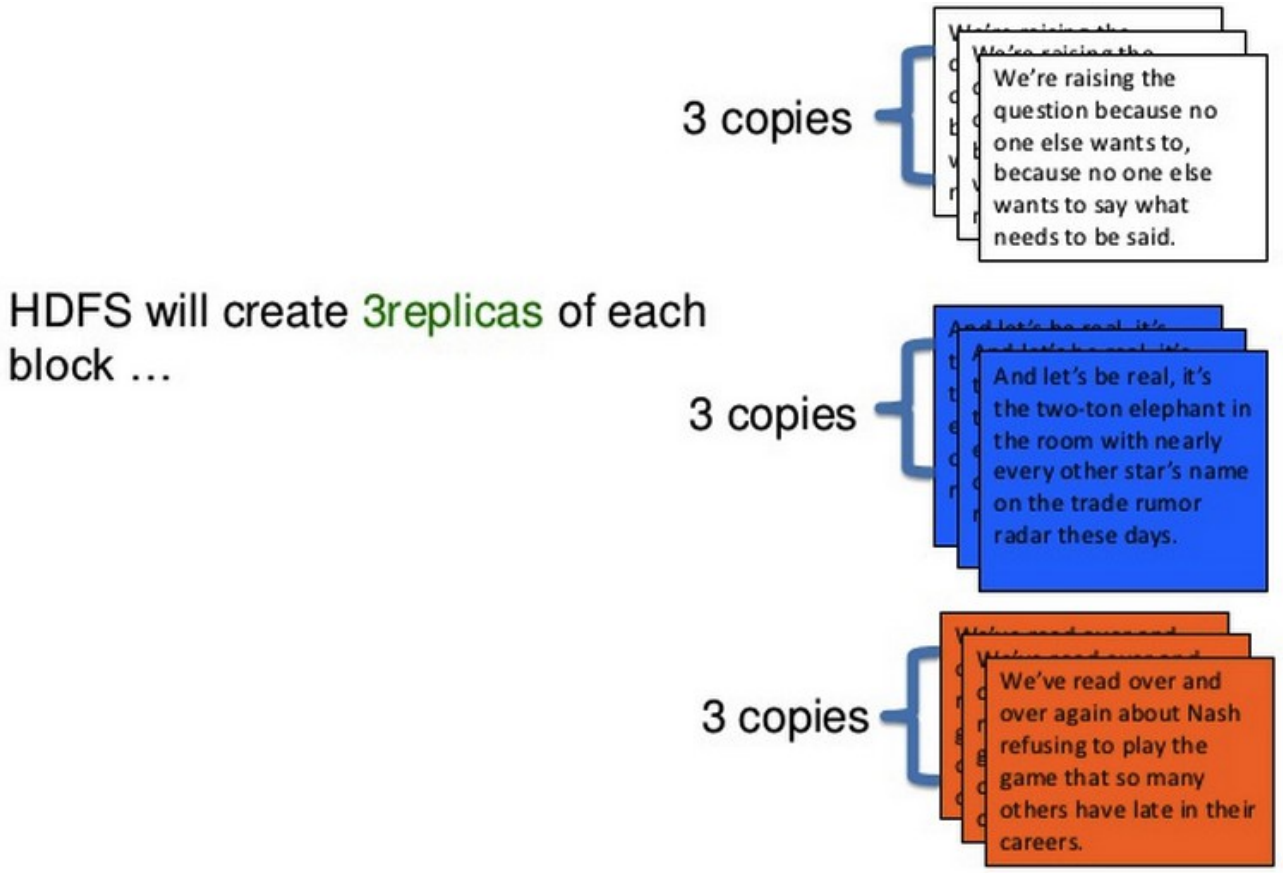
Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

How does HDFS work?



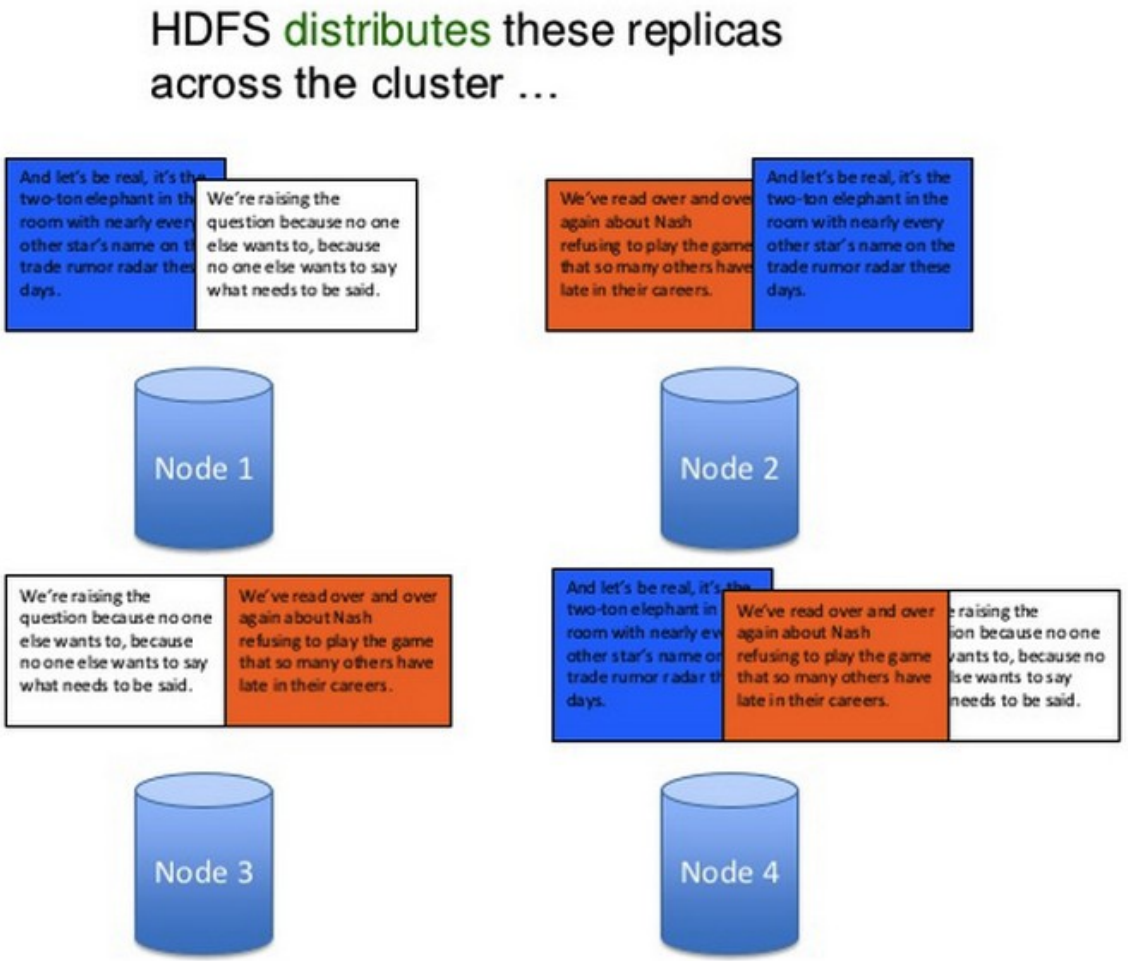
Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

How does HDFS work?



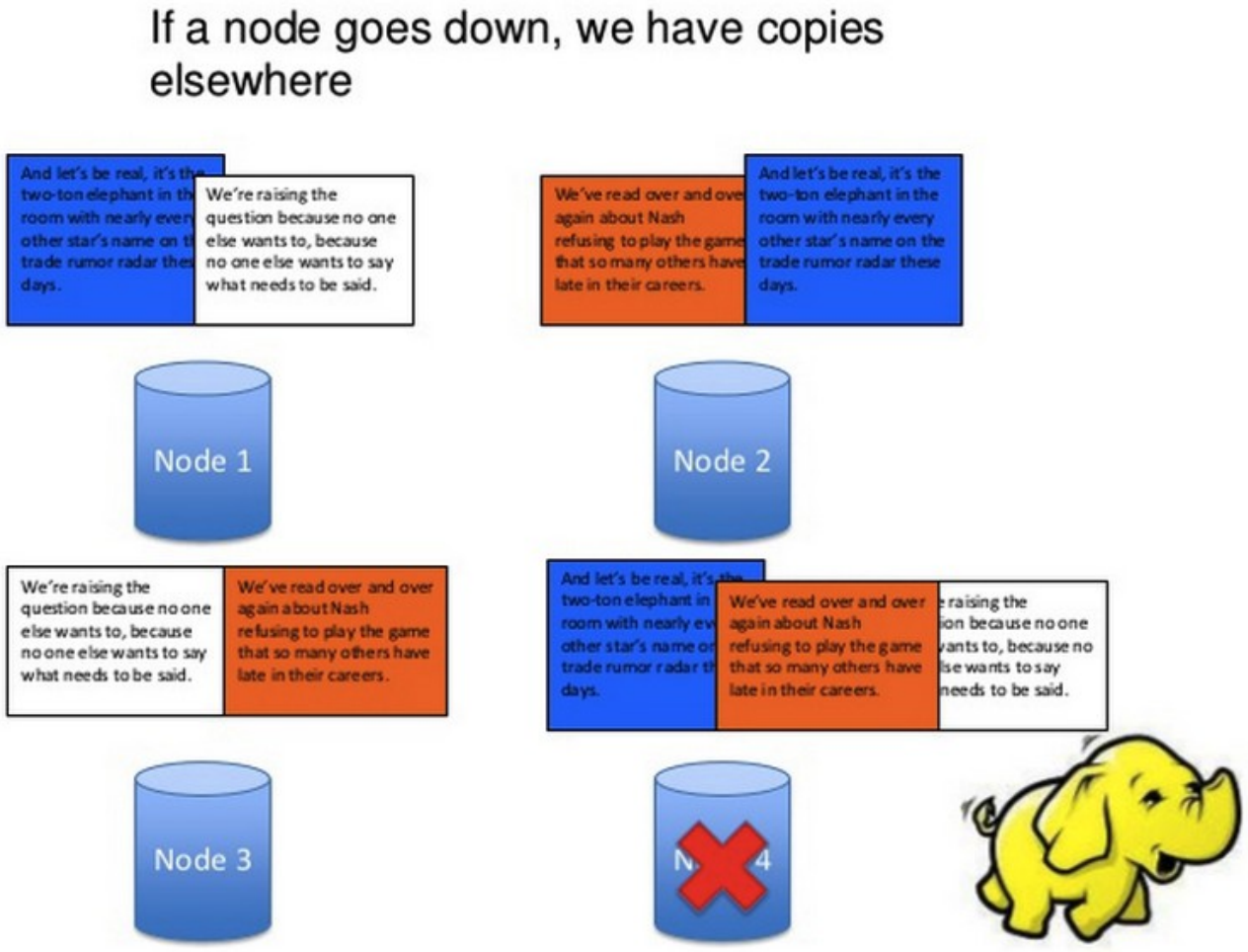
Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

How does HDFS work?



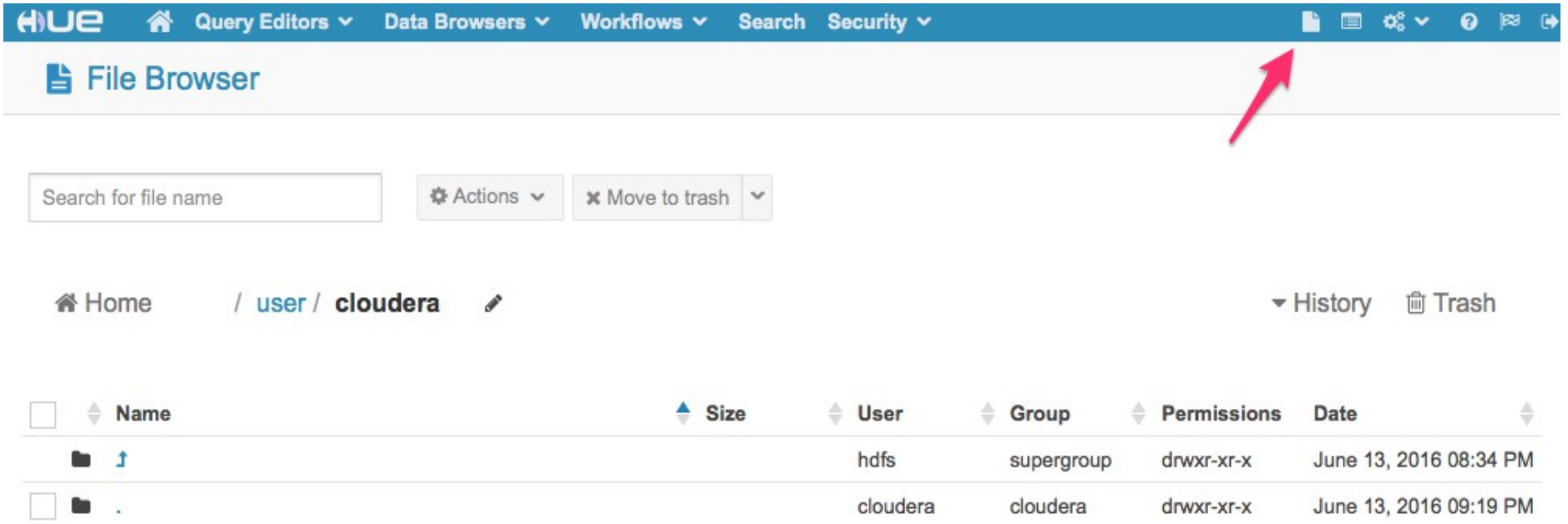
Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

How does HDFS work?

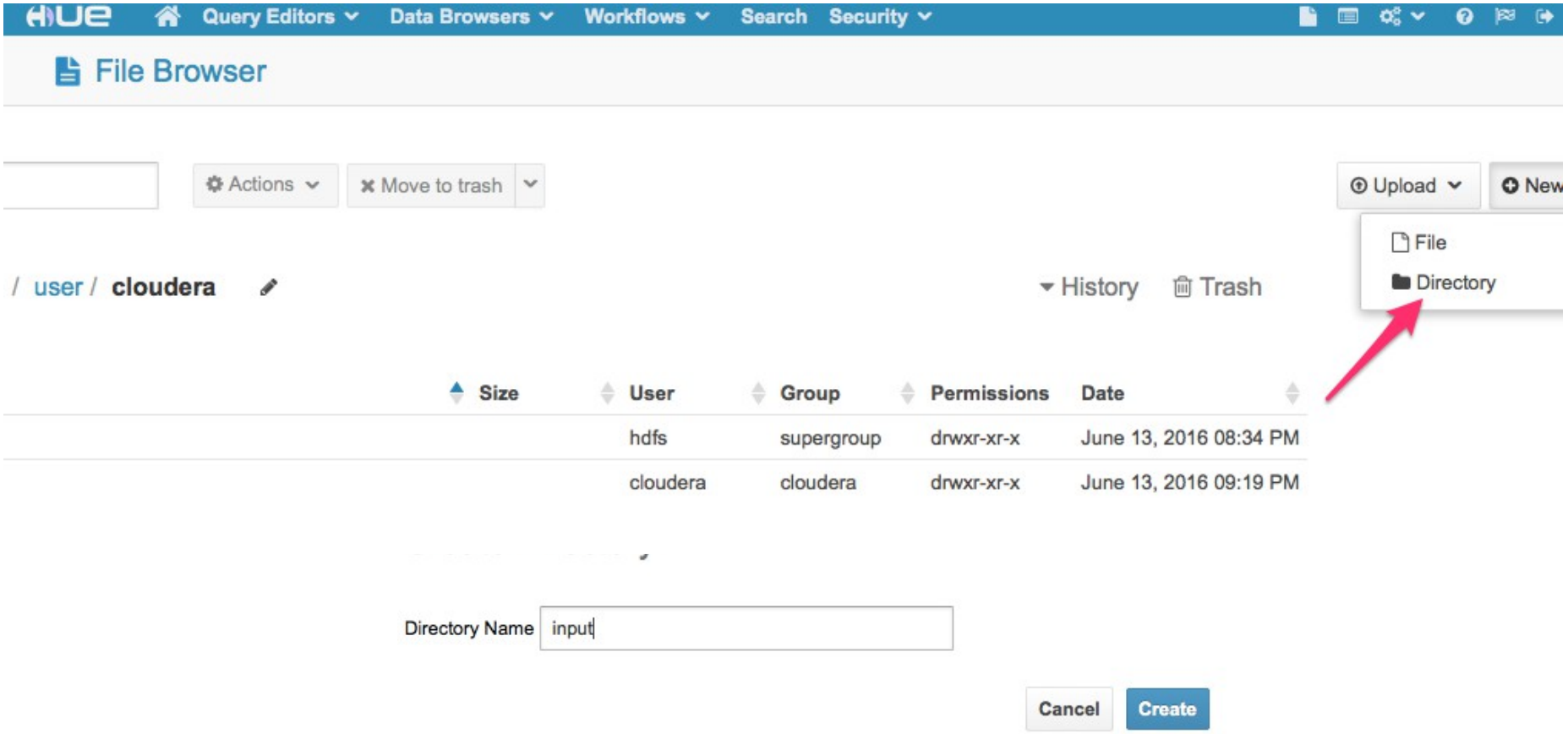


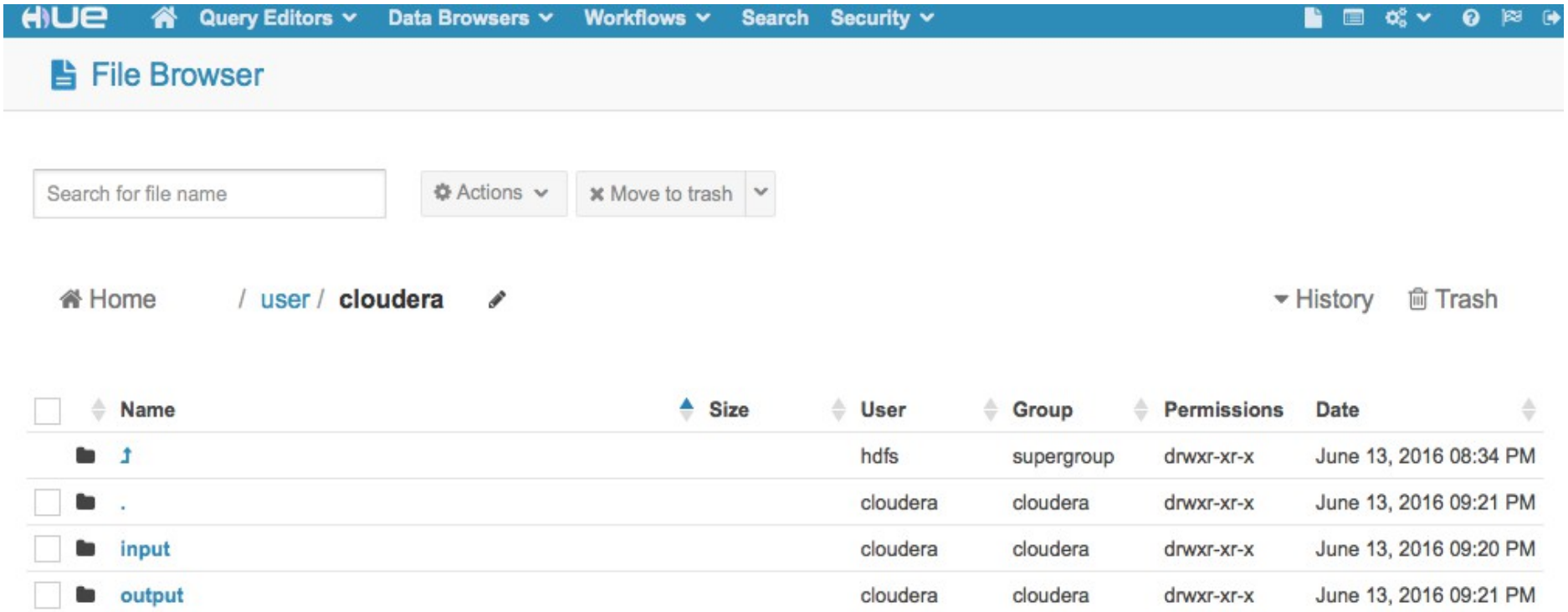
Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

Review file in Hadoop HDFS using File Browse

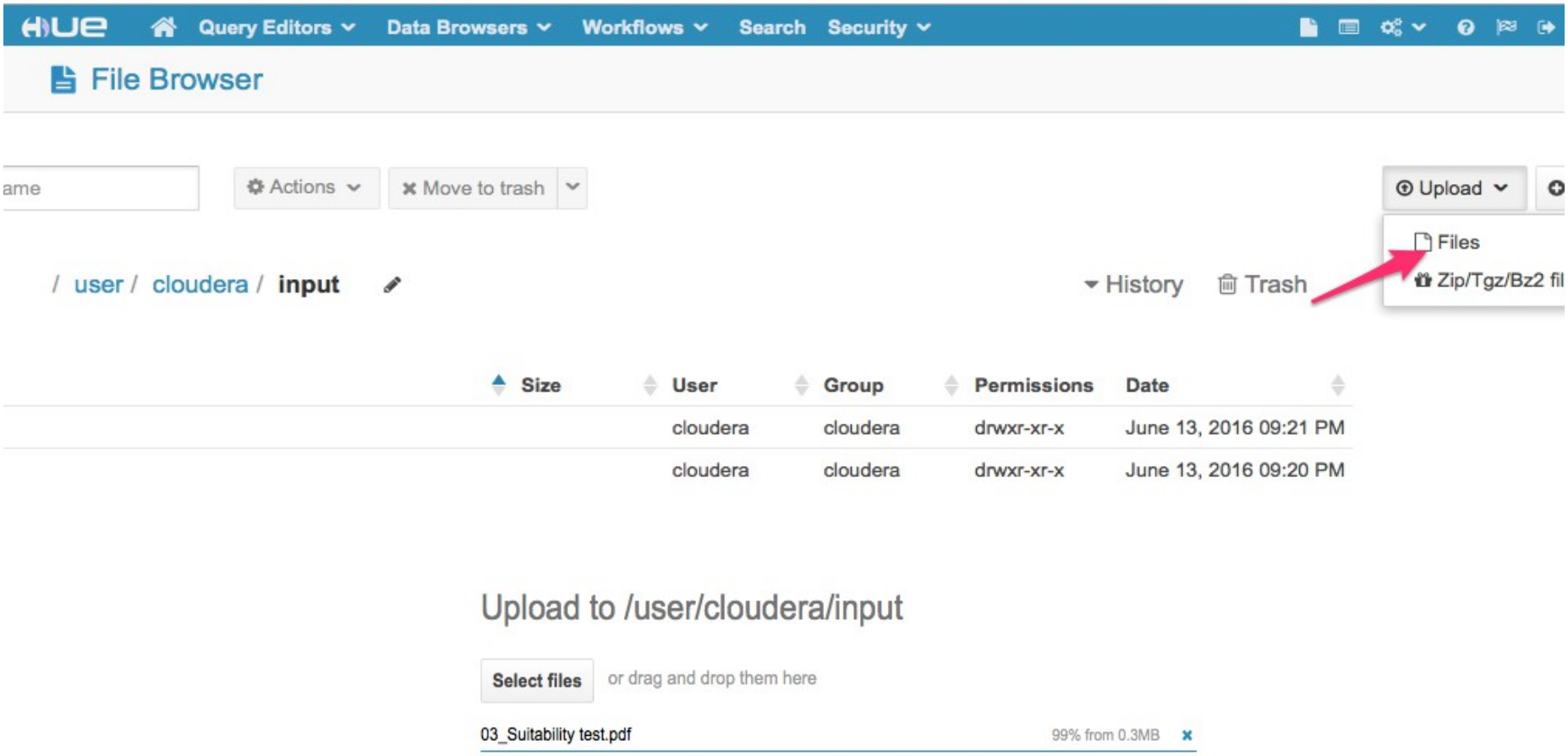


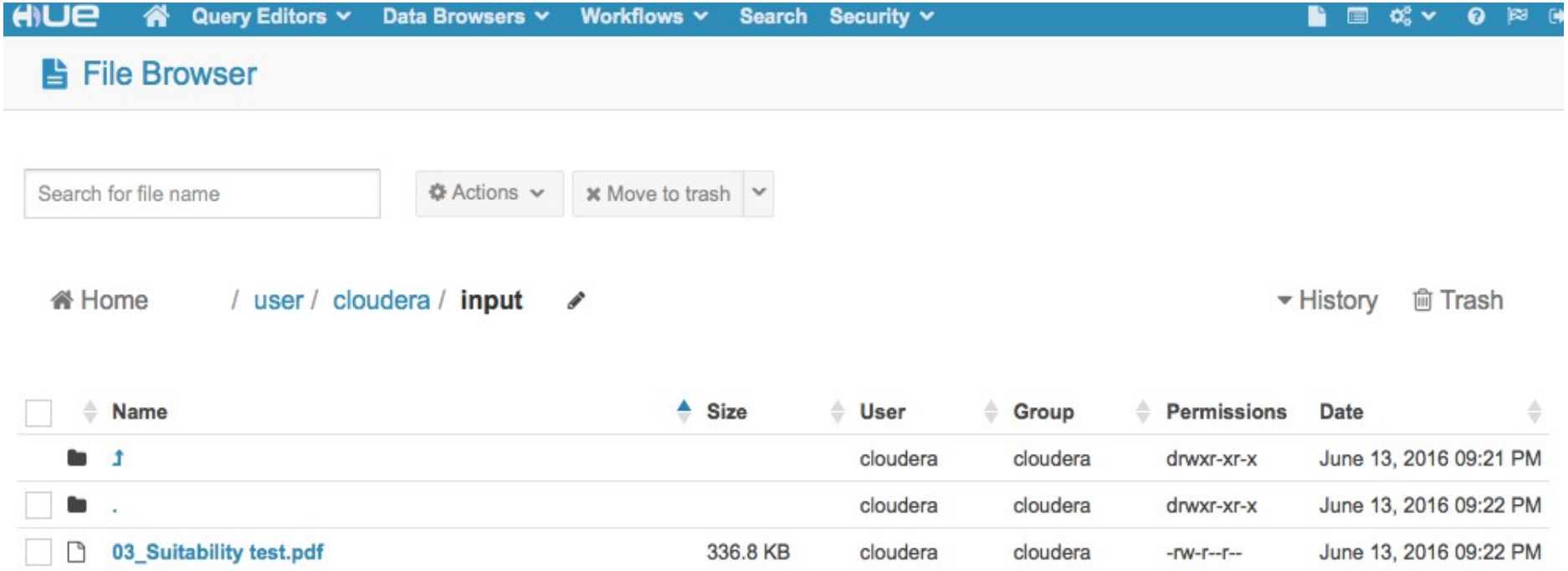
Create a new directory name as: input & output




HUE
Query Editors
Data Browsers
Workflows
Search
Security
File Browser
Search for file name
Actions
Move to trash
Home / user / cloudera
History
Trash
Name
Size
User
Group
Permissions
Date
hdfs supergroup drwxr-xr-x June 13, 2016 08:34 PM
. cloudera cloudera drwxr-xr-x June 13, 2016 09:21 PM
input cloudera cloudera drwxr-xr-x June 13, 2016 09:20 PM
output cloudera cloudera drwxr-xr-x June 13, 2016 09:21 PM

Upload a local file to HDFS




HUE
Query Editors
Data Browsers
Workflows
Search
Security
File Browser
Search for file name
Actions
Move to trash
Home / user / cloudera / input
History
Trash
Name
Size
User
Group
Permissions
Date
cloudera cloudera drwxr-xr-x June 13, 2016 09:21 PM
. cloudera cloudera drwxr-xr-x June 13, 2016 09:22 PM
03_Suitability test.pdf 336.8 KB cloudera cloudera -rw-r--r-- June 13, 2016 09:22 PM

Hands-On: Connect to a master node via SSH

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Docker commands:

- *docker images*
- *docker ps*
- *docker attach id*
- *docker kill id*

- *Exit from container*
  - *exit (exit & kill the running image)*
  - *Ctrl-P, Ctrl-Q (exit without killing the running image)*

Hive.apache.org

SSH Login to a master node

```
THANACHARTs-MacBook-Air:elastic-mapreduce-cli THANACHART$ ssh -i "imchadoop.pem" ub
untu@ec2-54-201-147-59.us-west-2.compute.amazonaws.com
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-74-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sun Mar 27 09:08:01 UTC 2016

  System load:  0.0                Processes:           135
  Usage of /:   27.6% of 29.40GB   Users logged in:     0
  Memory usage: 24%                IP address for eth0: 172.31.10.53
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud


*** System restart required ***
Last login: Sun Mar 27 09:08:01 2016 from node-io5.pool-125-24.dynamic.totbb.net
ubuntu@ip-172-31-10-53:~$
```

Hadoop syntax for HDFS

Command	Syntax
Listing of files in a directory	`hadoop fs -ls /user`
Create a new directory	`hadoop fs -mkdir /user/guest/newdirectory`
Copy a file from a local machine to Hadoop	`hadoop fs -put C:\Users\Administrator\Downloads\localfile.csv /user/rajn/newdirectory/hadoopfile.txt`
Copy a file from Hadoop to a local machine	`hadoop fs –get /user/rajn/newdirectory/hadoopfile.txt C:\Users\Administrator\Desktop\`
Tail last few lines of a large file in Hadoop	`hadoop fs -tail /user/rajn/newdirectory/hadoopfile.txt`
View the complete contents of a file in Hadoop	`hadoop fs -cat /user/rajn/newdirectory/hadoopfile.txt`
Remove a complete directory from Hadoop	`hadoop fs –rm -r /user/rajn/newdirectory`
Check the Hadoop filesystem space utilization	`hadoop fs –du /`

Install wget

- Command: yum install wget

```
[root@quickstart /]# yum install wget
Loaded plugins: fastestmirror
Setting up Install Process
Determining fastest mirrors
epel/metalink                                    |  13 kB     00:00
 * base: mirrors.evowise.com
 * epel: mirror.cogentco.com
 * extras: mirror.us.leaseweb.net
 * updates: mirror.cs.pitt.edu
base                                             | 3.7 kB     00:00
base/primary_db                                  | 4.7 MB     00:06
```

Download an example text file

Make your own durectory at a master node to avoid mixing with others

```
$mkdir guest1
$cd guest1
$wget https://s3.amazonaws.com/imcbucket/input/pg2600.txt
```

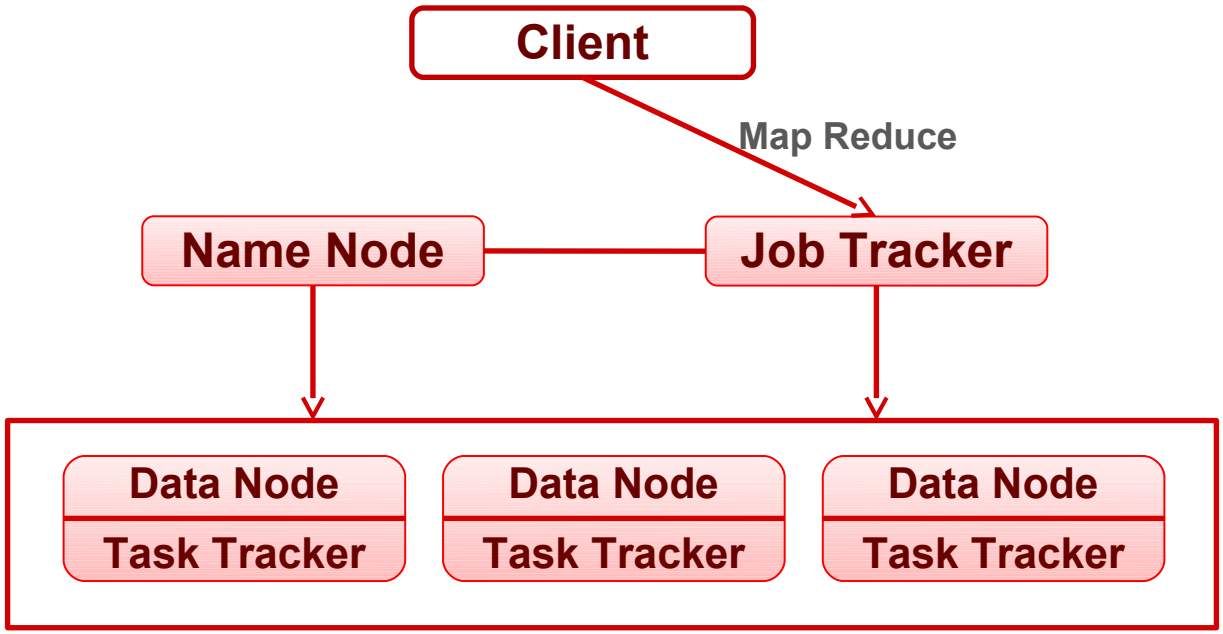
```
--2016-03-27 09:58:48--  https://s3.amazonaws.com/imcbucket/input/pg2600.txt
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.19.187
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.19.187|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3291648 (3.1M) [text/plain]
Saving to: 'pg2600.txt'

100%[======================================>] 3,291,648   3.14MB/s   in 1.0s

2016-03-27 09:58:50 (3.14 MB/s) - 'pg2600.txt' saved [3291648/3291648]
```

Upload Data to Hadoop

Note: you login as ubuntu, so you need to a sudo command to Switch user to hdfs

```
$hadoop fs -ls /user/cloudera/input
$hadoop fs -rm /user/cloudera/input/*
$hadoop fs -put pg2600.txt /user/cloudera/input/
$hadoop fs -ls /user/cloudera/input
```

```
[root@quickstart guest1]# hadoop fs -ls /user/cloudera/input
Found 1 items
-rw-r--r--   1 root cloudera    3291648 2016-06-14 04:29 /user/clou
dera/input/pg2600.txt
[root@quickstart guest1]#
```



Lecture: Understanding Map Reduce Processing

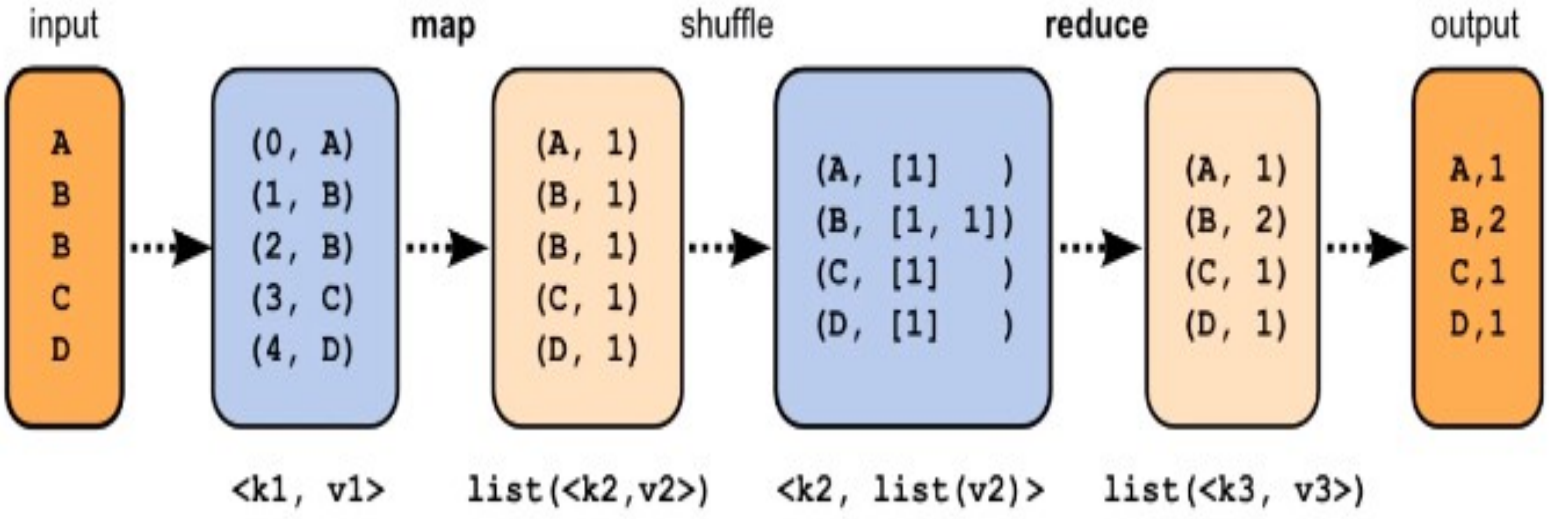
Before MapReduce…

- Large scale data processing was difficult!
  - Managing hundreds or thousands of processors
  - Managing parallelization and distribution
  - I/O Scheduling
  - Status and monitoring
  - Fault/crash tolerance
- MapReduce provides all of these, easily!

Source: http://labs.google.com/papers/mapreduce-osdi04-slides/index-auto-0002.html

thanachart@imcinstitute.com

MapReduce Framework



Source: www.bigdatauniversity.com

How Map and Reduce Work Together

- Map returns information
- Reduces accepts information
- Reduce applies a user defined function to reduce the amount of data

thanachart@imcinstitute.com

Map Abstraction

- Inputs a key/value pair
  - Key is a reference to the input value
  - Value is the data set on which to operate
- Evaluation
  - Function defined by user
  - Applies to every value in value input
    - Might need to parse input
- Produces a new list of key/value pairs
  - Can be different type from input pair

thanachart@imcinstitute.com

Reduce Abstraction

- Starts with intermediate Key / Value pairs
- Ends with finalized Key / Value pairs

- Starting pairs are sorted by key
- Iterator supplies the values for a given key to the Reduce function.

thanachart@imcinstitute.com

Reduce Abstraction

- Typically a function that:
  - Starts with a large number of key/value pairs
    - One key/value for each word in all files being greped (including multiple entries for the same word)
  - Ends with very few key/value pairs
    - One key/value for each unique word across all the files with the number of instances summed into this entry
- Broken up so a given worker works with input of the same key.

thanachart@imcinstitute.com

Why is this approach better?

- Creates an abstraction for dealing with complex overhead
  - The computations are simple, the overhead is messy
- Removing the overhead makes programs much smaller and thus easier to use
  - Less testing is required as well. The MapReduce libraries can be assumed to work properly, so only user code needs to be tested
- Division of labor also handled by the MapReduce libraries, so programmers only need to focus on the actual computation

thanachart@imcinstitute.com

Hands-On: Writing you own Map Reduce Program

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Example MapReduce: WordCount

```
package org.apache.hadoop.examples;

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;
import org.apache.hadoop.util.GenericOptionsParser;

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);
      }
    }
  }
```


thanachart@imcinstitute.com

```
  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);
    }
  }
```

```
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
      System.err.println("Usage: wordcount <in> <out>");
      System.exit(2);
    }
    Job job = new Job(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(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

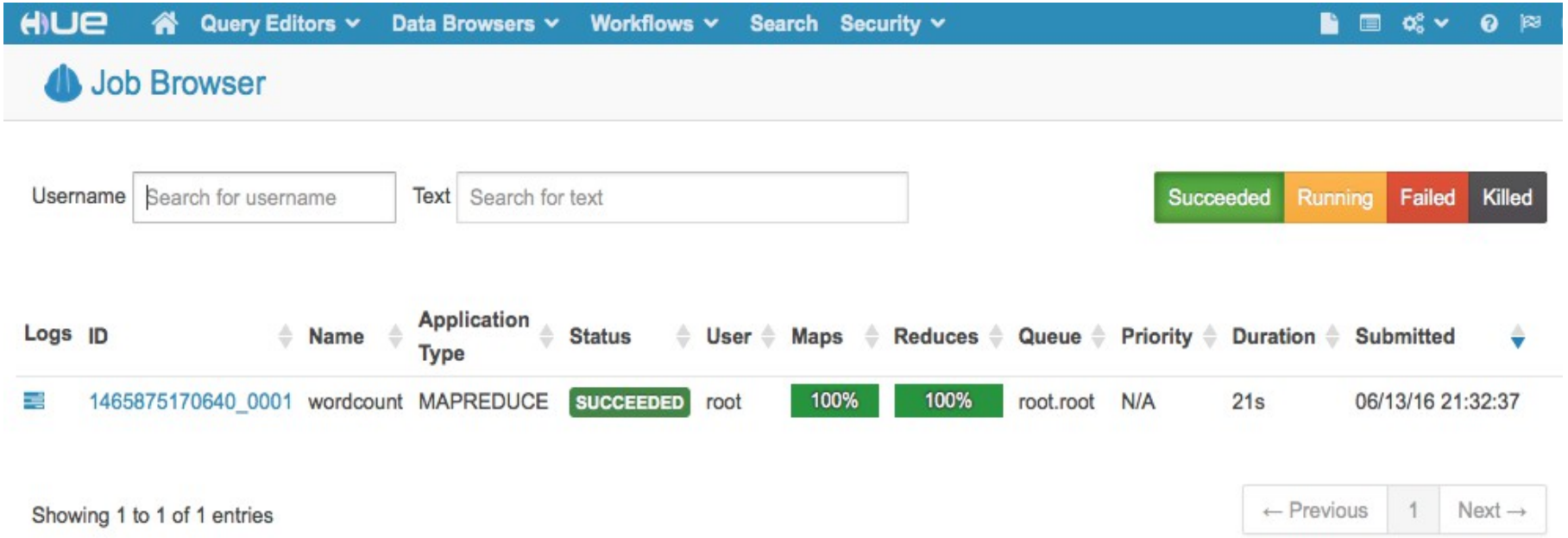

thanachart@imcinstitute.com

Running Map Reduce Program

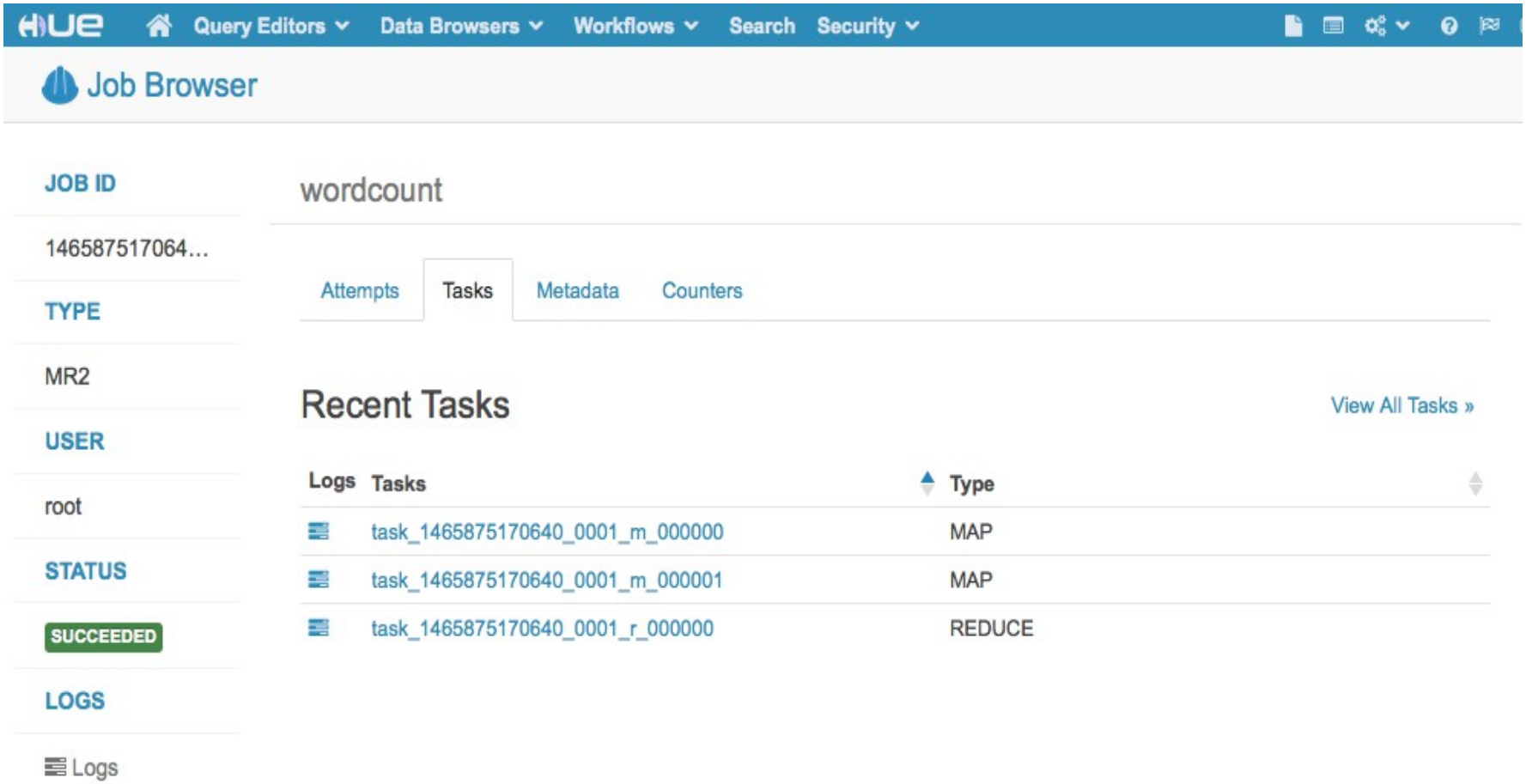
```
[cloudera@quickstart ~]$ cd workspace/
[cloudera@quickstart workspace]$ hadoop jar wordcount.jar org.myorg.WordCount input/*  output/wordcount_output
15/02/08 10:30:31 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/02/08 10:30:32 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/02/08 10:30:33 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool i
ation with ToolRunner to remedy this.
15/02/08 10:30:33 INFO mapred.FileInputFormat: Total input paths to process : 1
15/02/08 10:30:34 INFO mapreduce.JobSubmitter: number of splits:2
15/02/08 10:30:34 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1423408479621_0009
15/02/08 10:30:35 INFO impl.YarnClientImpl: Submitted application application_1423408479621_0009
15/02/08 10:30:35 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_142
15/02/08 10:30:35 INFO mapreduce.Job: Running job: job_1423408479621_0009
15/02/08 10:30:52 INFO mapreduce.Job: Job job_1423408479621_0009 running in uber mode : false
15/02/08 10:30:52 INFO mapreduce.Job:  map 0% reduce 0%
15/02/08 10:31:22 INFO mapreduce.Job:  map 58% reduce 0%
15/02/08 10:31:25 INFO mapreduce.Job:  map 100% reduce 0%
15/02/08 10:31:52 INFO mapreduce.Job:  map 100% reduce 100%
15/02/08 10:31:53 INFO mapreduce.Job: Job job_1423408479621_0009 completed successfully
15/02/08 10:31:53 INFO mapreduce.Job: Counters: 49
```

```
$cd /guest1
$wget https://dl.dropboxusercontent.com/u/12655380/wordcount.jar
$hadoop jar wordcount.jar org.myorg.WordCount
/user/cloudera/input/* /user/cloudera/output/wordcount
```

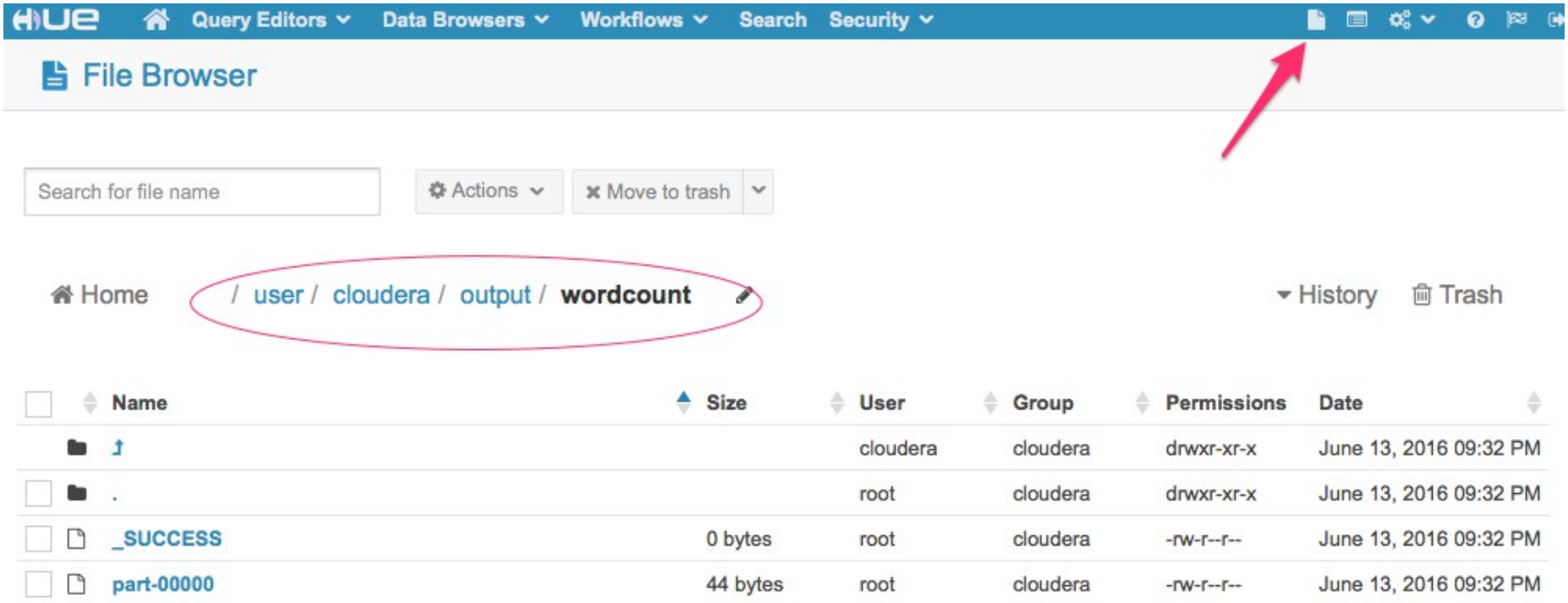
Reviewing MapReduce Job in Hue



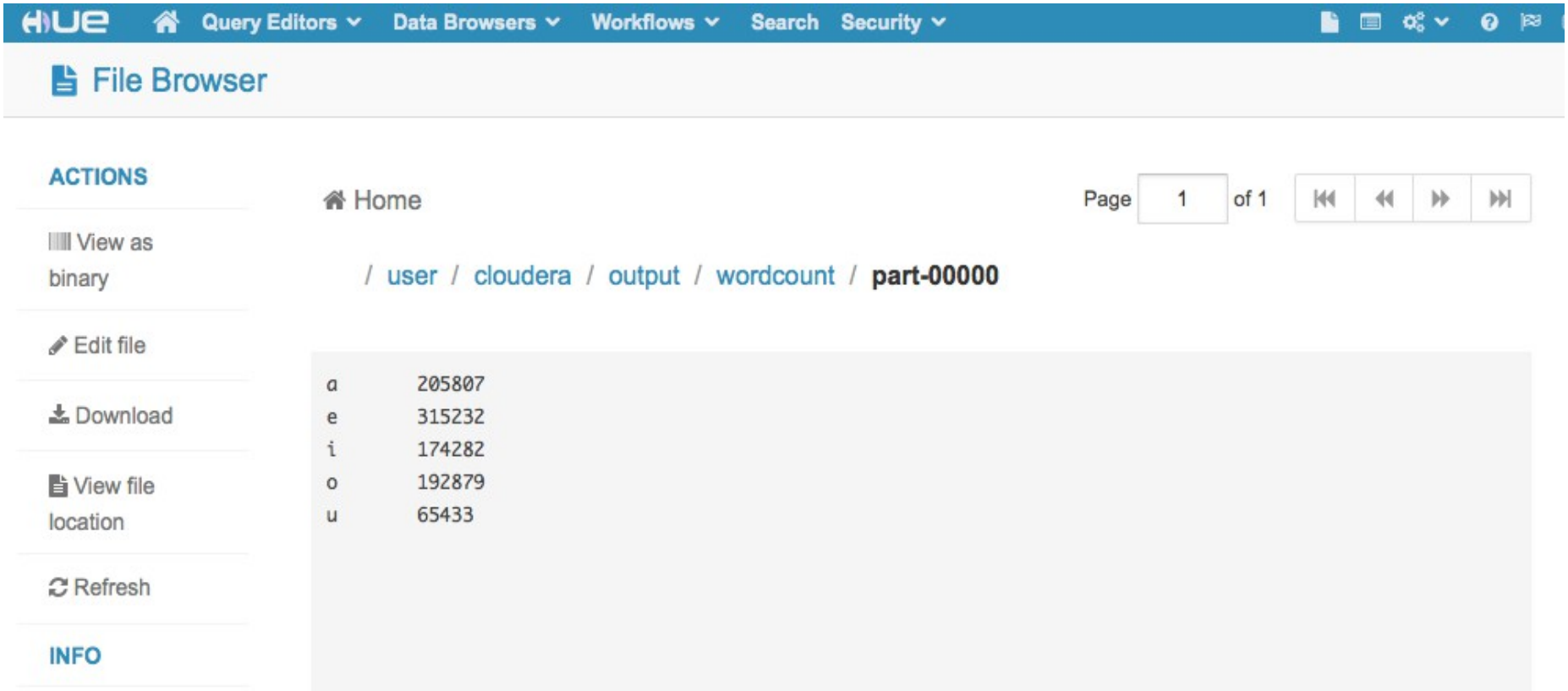
Reviewing MapReduce Job in Hue



Reviewing MapReduce Output Result



Reviewing MapReduce Output Result



Lecture
Understanding Hive

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

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

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

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.

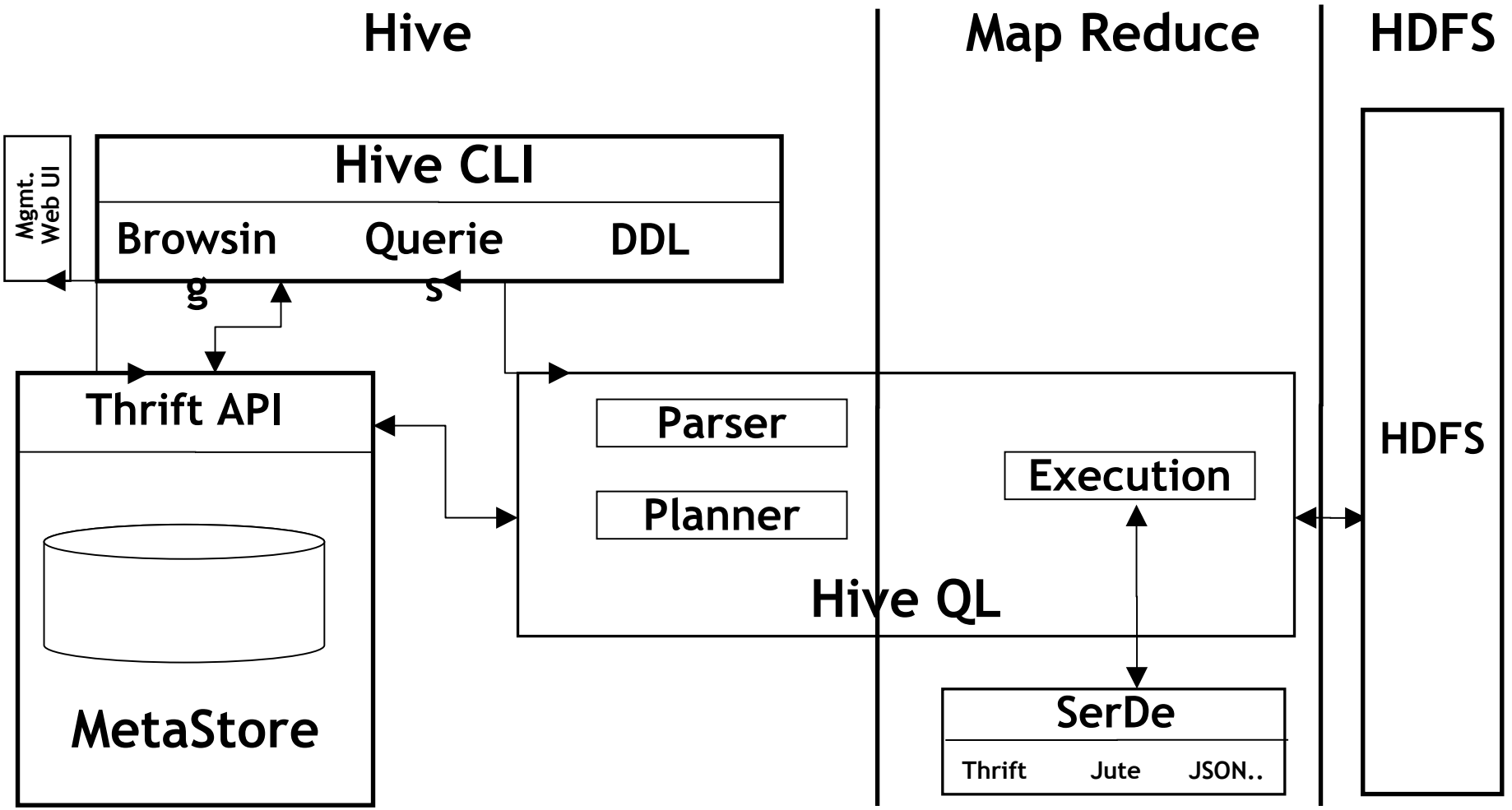
A SerDe is a combination of a Serializer and a Deserializer (hence, Ser-De). The Deserializer interface takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate. The Serializer, however, will take a Java object that Hive has been working with, 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

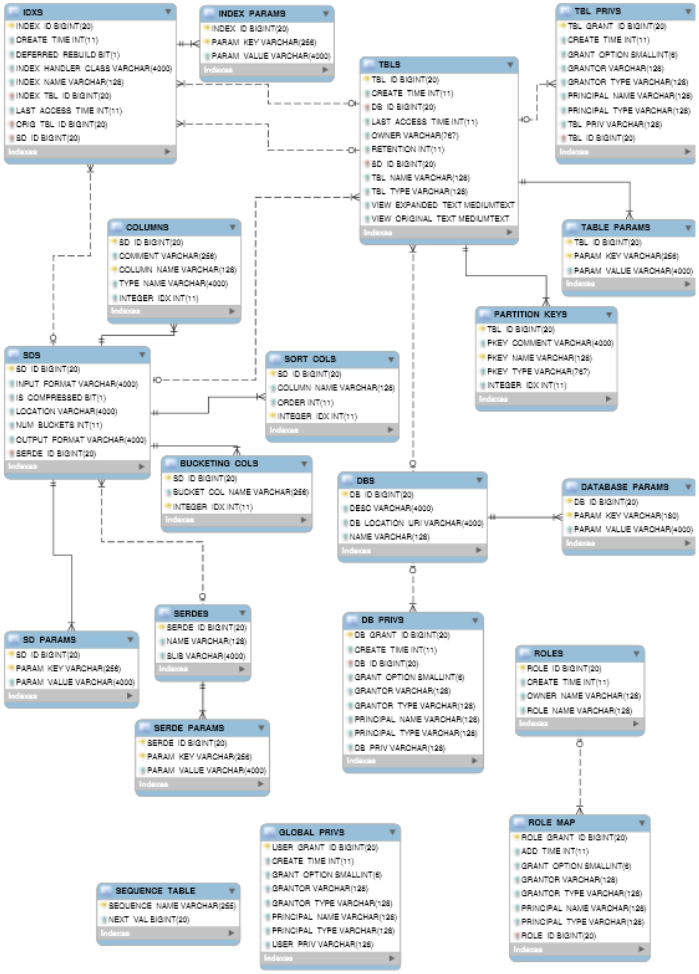
Architecture Overview



Hive.apache.org

Danairat T., danairat@gmail.com: Thanachart N., thanachart@imcinstitute.com April 2015

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`

ortonworks.com

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

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

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

```
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));
    }
```

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);`

ortonworks.com

Hands-On: Loading Data using Hive

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Start Hive

```
[root@quickstart guest1]# hive
2016-06-14 07:48:56,273 WARN  [main] mapreduce.TableMapReduceUtil: The
hbase-prefix-tree module jar containing PrefixTreeCodec is not present.
  Continuing without it.

Logging initialized using configuration in file:/etc/hive/conf.dist/hiv
e-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended
.
hive>
```

Quit from Hive

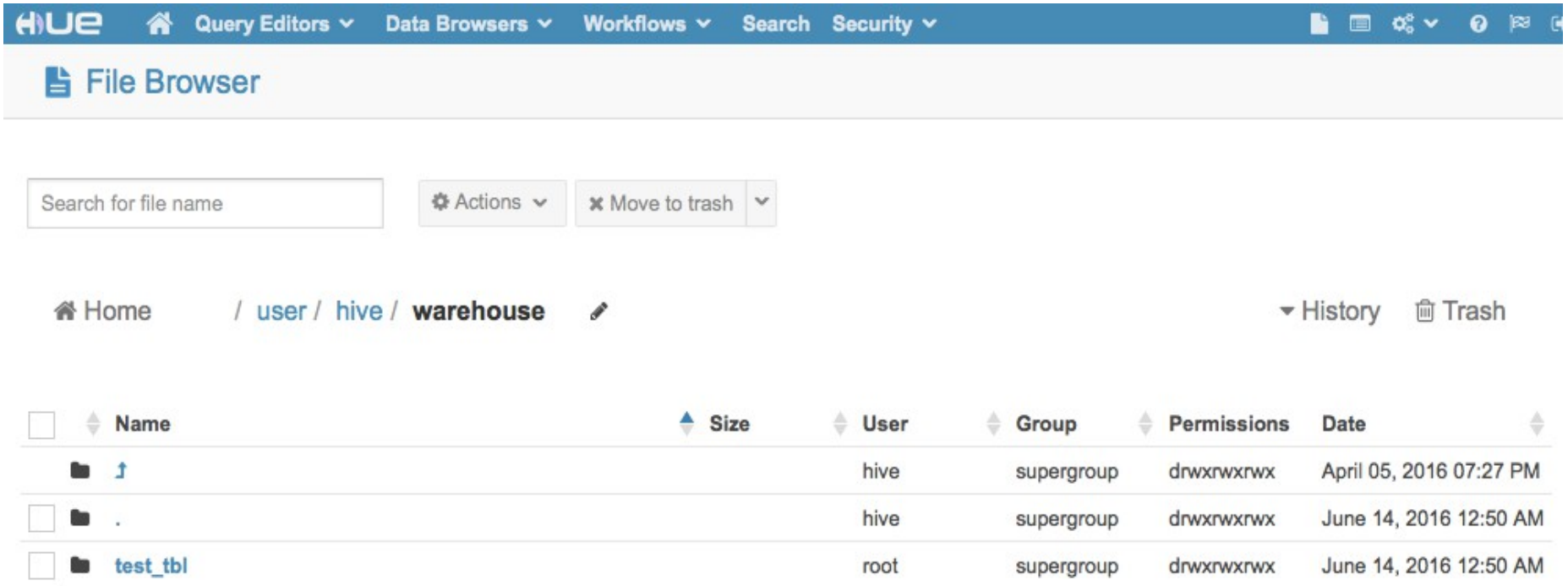
```
hive> quit;
```

Create Hive Table

```
hive> CREATE TABLE test_tbl(id INT, country STRING) ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
OK
Time taken: 0.886 seconds
hive> show tables;
OK
test_tbl
Time taken: 0.125 seconds, Fetched: 1 row(s)
hive> describe test_tbl;
OK
id                      int
country                 string
Time taken: 0.115 seconds, Fetched: 2 row(s)
hive>
```

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

Reviewing Hive Table in HDFS



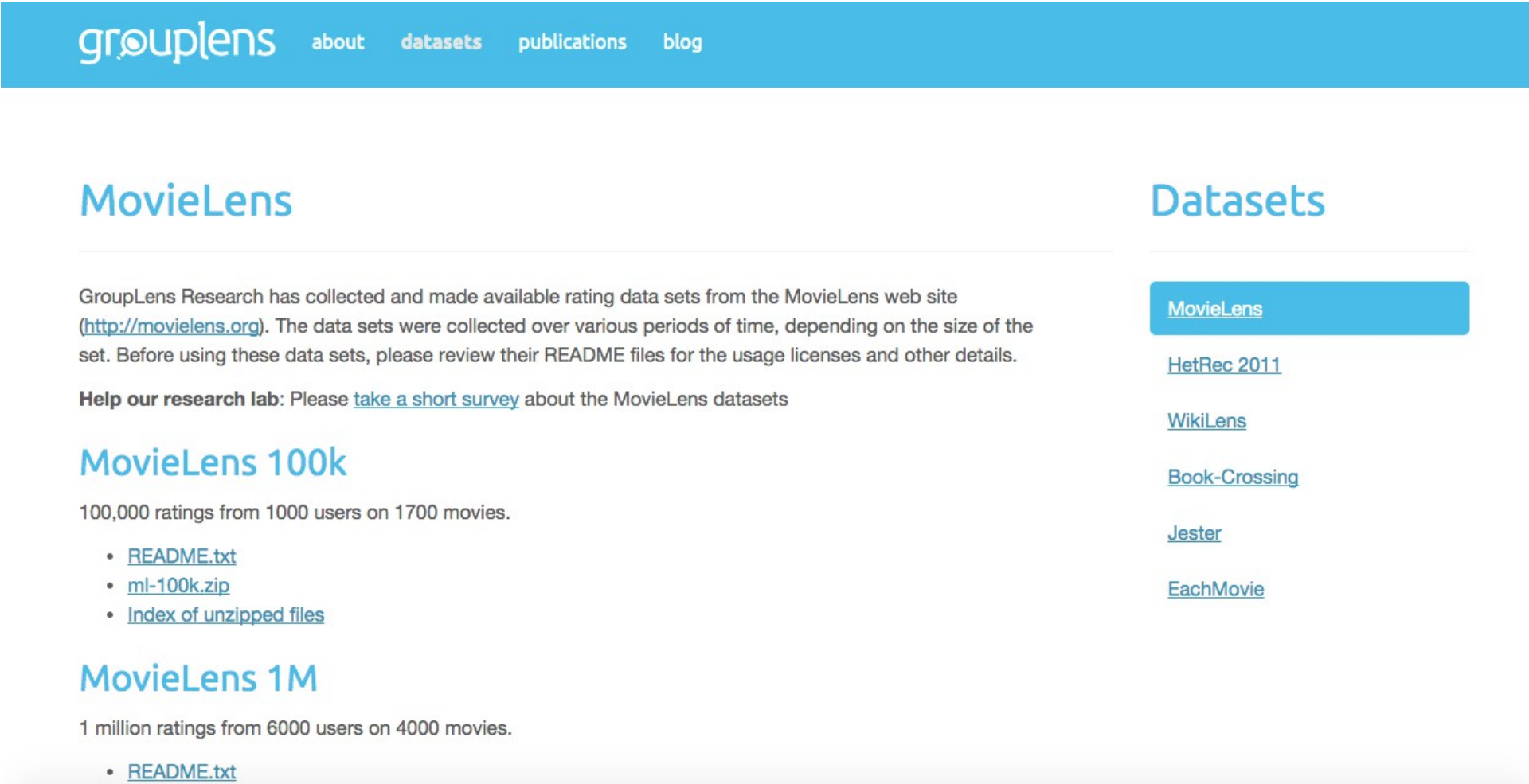
Alter and Drop Hive Table

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

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

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

Preparing Large Dataset

http://grouplens.org/datasets/movielens/



MovieLen Dataset

1) Type command > `wget http://files.grouplens.org/datasets/movielens/ml-100k.zip`

2) Type command > `yum install unzip`

3) Type command > `unzip ml-100k.zip`

4) Type command > `more ml-100k/u.user`

```
[root@quickstart guest1]# more ml-100k/u.user
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
11|39|F|other|30329
```

Moving dataset to HDFS

1) Type command > `cd ml-100k`

2) Type command > `hadoop fs -mkdir /user/cloudera/movielens`

3) Type command > `hadoop fs -put u.user /user/cloudera/movielens`

4) Type command > `hadoop fs -ls /user/cloudera/movielens`

```
[root@quickstart ml-100k]# hadoop fs -ls /user/cloudera/movielens
Found 1 items
-rw-r--r--   1 root cloudera      22628 2016-06-14 08:04 /user/cloudera/
movielens/u.user
[root@quickstart ml-100k]#
```

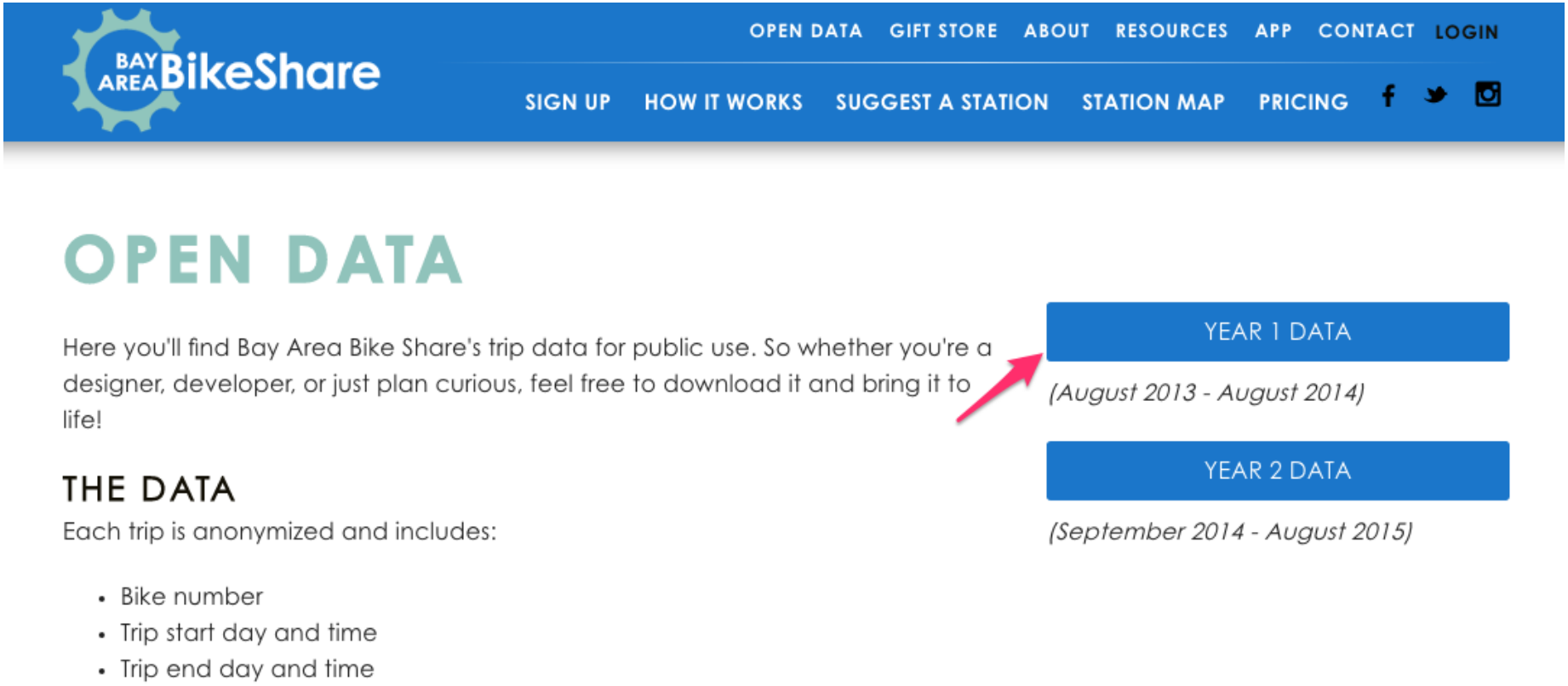
CREATE & SELECT Table

```
hive> CREATE EXTERNAL TABLE users (userid INT,  age INT,
    >   gender STRING, occupation STRING, zipcode STRING) ROW FORMAT
    > DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE
    > LOCATION '/user/cloudera/movielens';
OK
Time taken: 0.646 seconds
hive> SELECT * FROM users;
OK
1       24      M       technician      85711
2       53      F       other   94043
3       23      M       writer  32067
4       24      M       technician      43537
5       33      F       other   15213
6       42      M       executive       98101
7       57      M       administrator   91344
8       36      M       administrator   05201
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Bay Area Bike Share (BABS)

http://www.bayareabikeshare.com/open-data



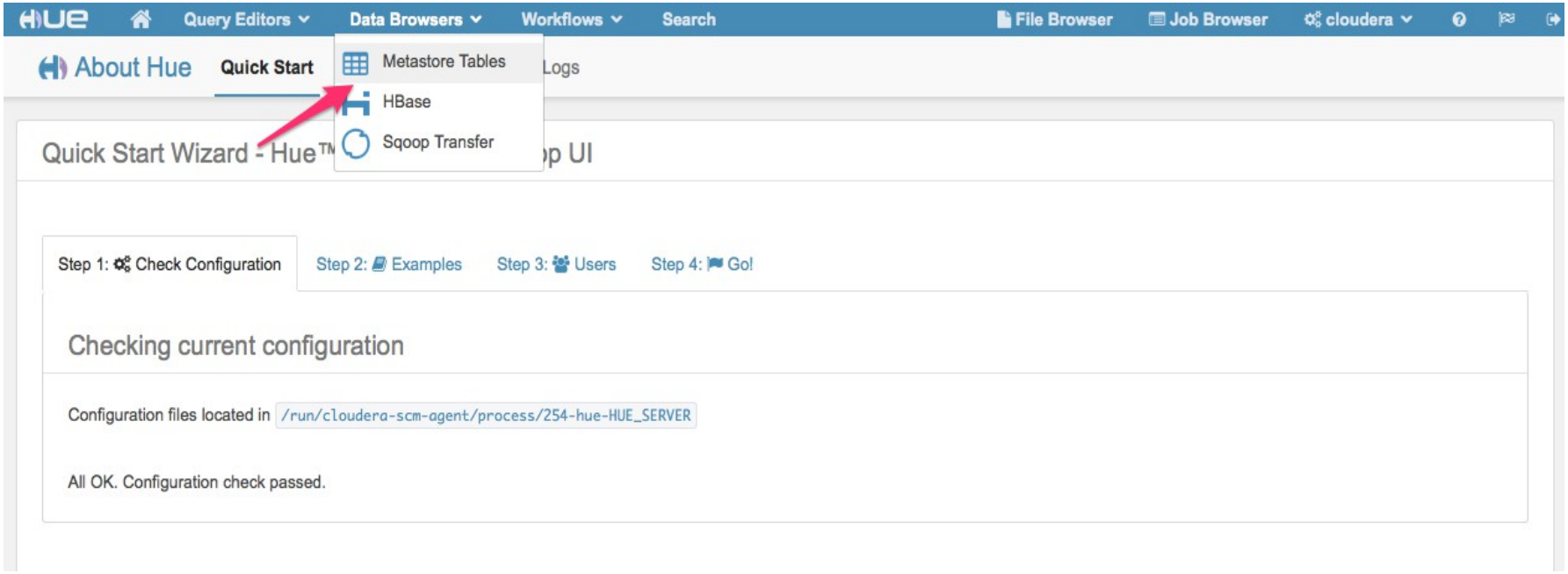
Preparing a bike data

```
$wget https://s3.amazonaws.com/babs-open-data/
babs_open_data_year_1.zip
$unzip babs_open_data_year_1.zip
$cd 201402_babs_open_data/
$hadoop fs -put 201402_trip_data.csv
/user/cloudera
$ hadoop fs -ls /user/cloudera
```

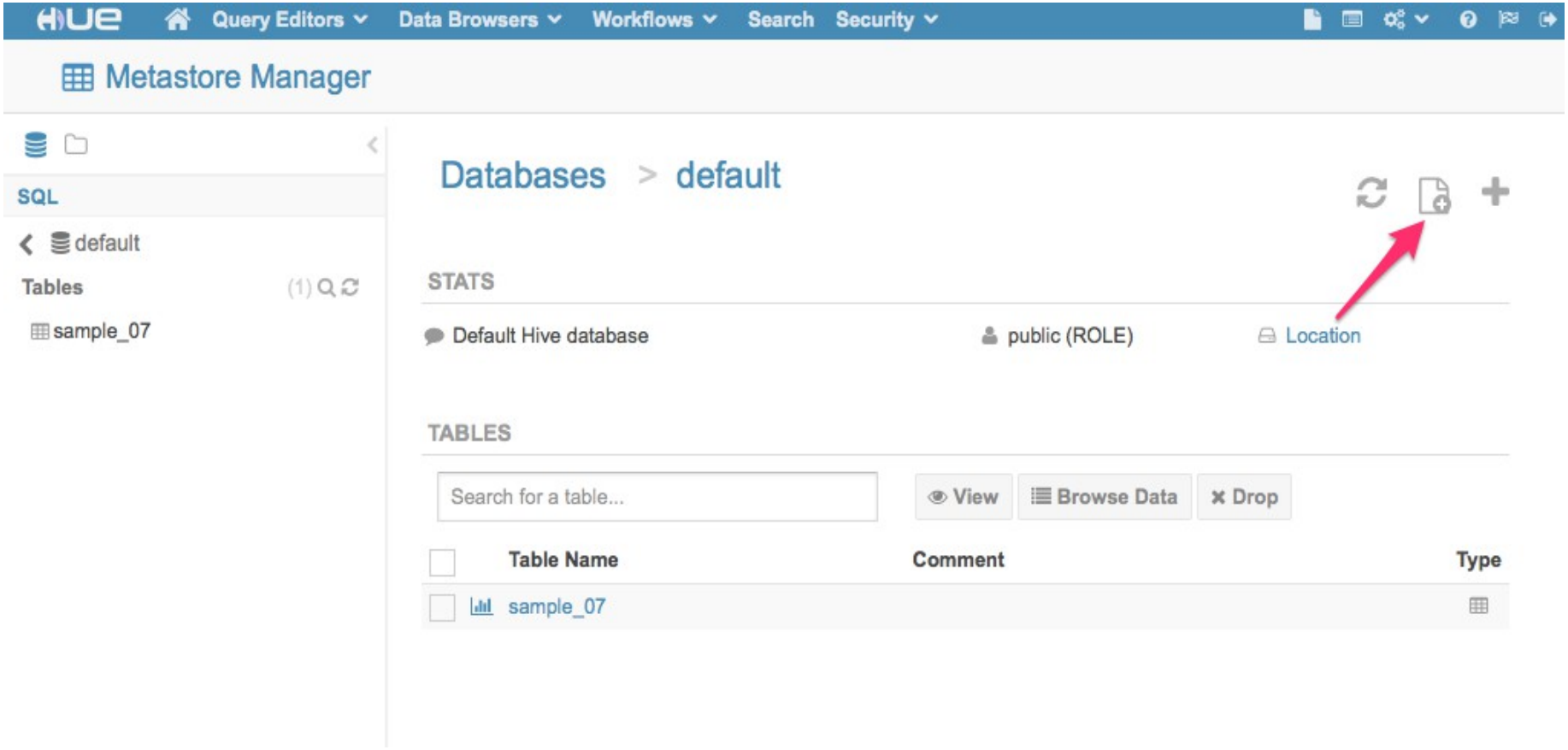
```
[root@quickstart 201402_babs_open_data]# hadoop fs -ls /user/cloudera
Found 4 items
-rw-r--r--   1 root     cloudera   17219022 2016-06-14 08:13 /user/cloudera/201402_t
rip_data.csv
drwxr-xr-x   - cloudera cloudera          0 2016-06-14 04:29 /user/cloudera/input
drwxr-xr-x   - root     cloudera          0 2016-06-14 08:04 /user/cloudera/movielen
s
drwxr-xr-x   - cloudera cloudera          0 2016-06-14 04:32 /user/cloudera/output
[root@quickstart 201402_babs_open_data]#
```

Importing CSV Data with the Metastore App

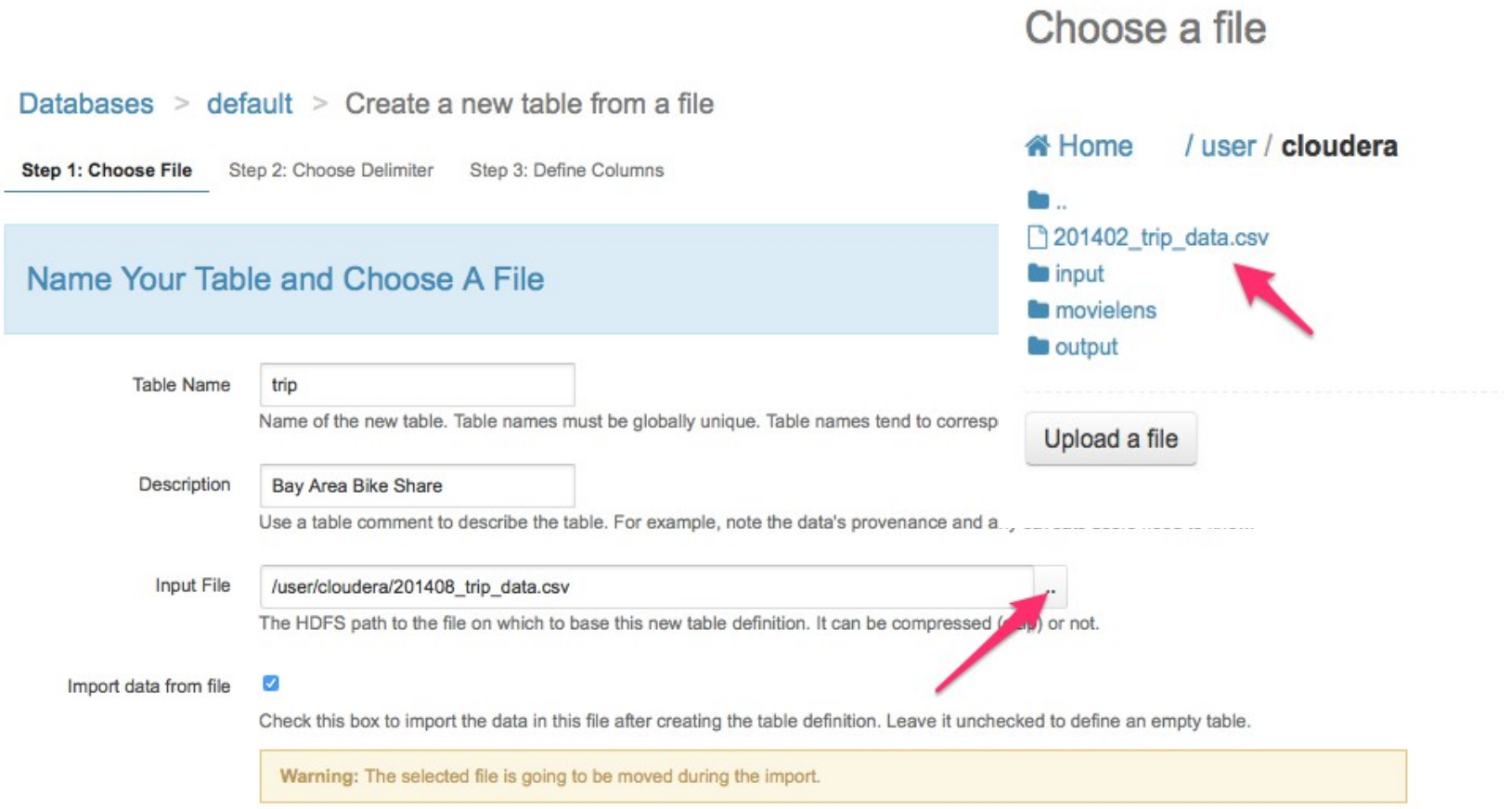
The BABS data set contains 4 CSVs that contain data for stations, trips, rebalancing (availability), and weather. We will import **trips** dataset using Metastore Tables



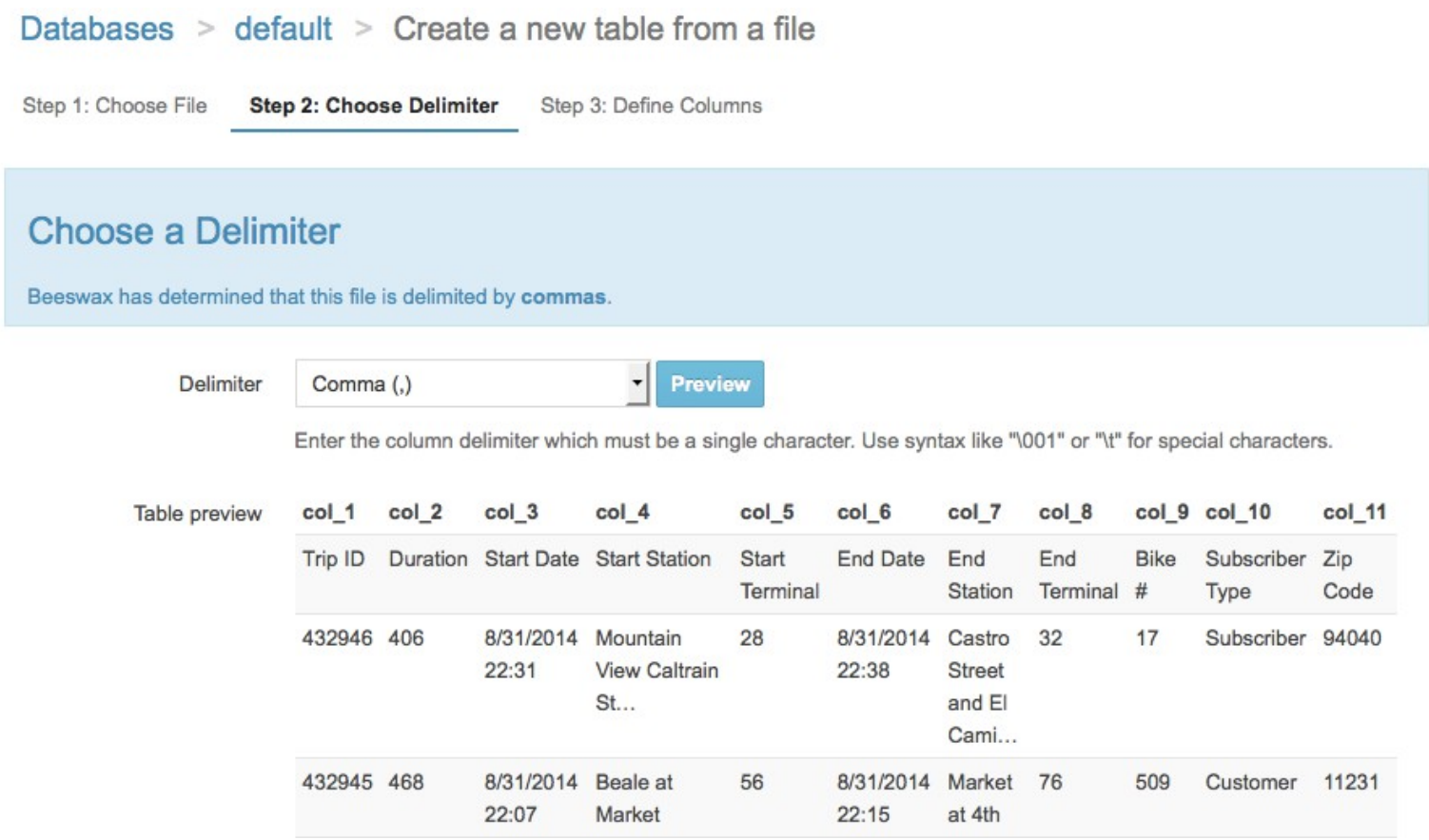
Select: Create a new table from a file



Name a table and select a file

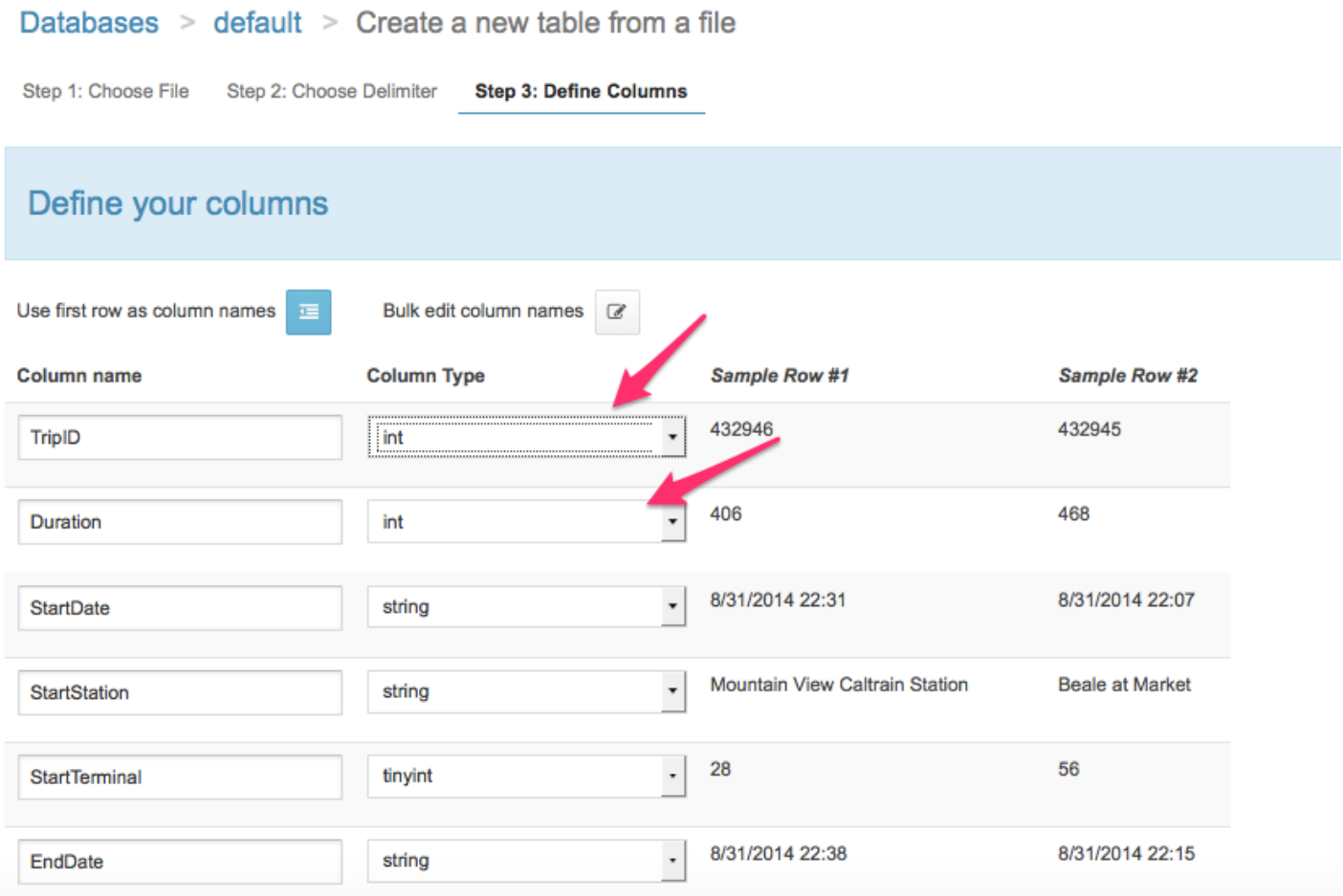


Choose Delimiter



col_1	col_2	col_3	col_4	col_5	col_6	col_7	col_8	col_9	col_10	col_11
Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #	Subscriber Type	Zip Code
432946	406	8/31/2014 22:31	Mountain View Caltrain St…	28	8/31/2014 22:38	Castro Street and El Cami…	32	17	Subscriber	94040
432945	468	8/31/2014 22:07	Beale at Market	56	8/31/2014 22:15	Market at 4th	76	509	Customer	11231

Define Column Types



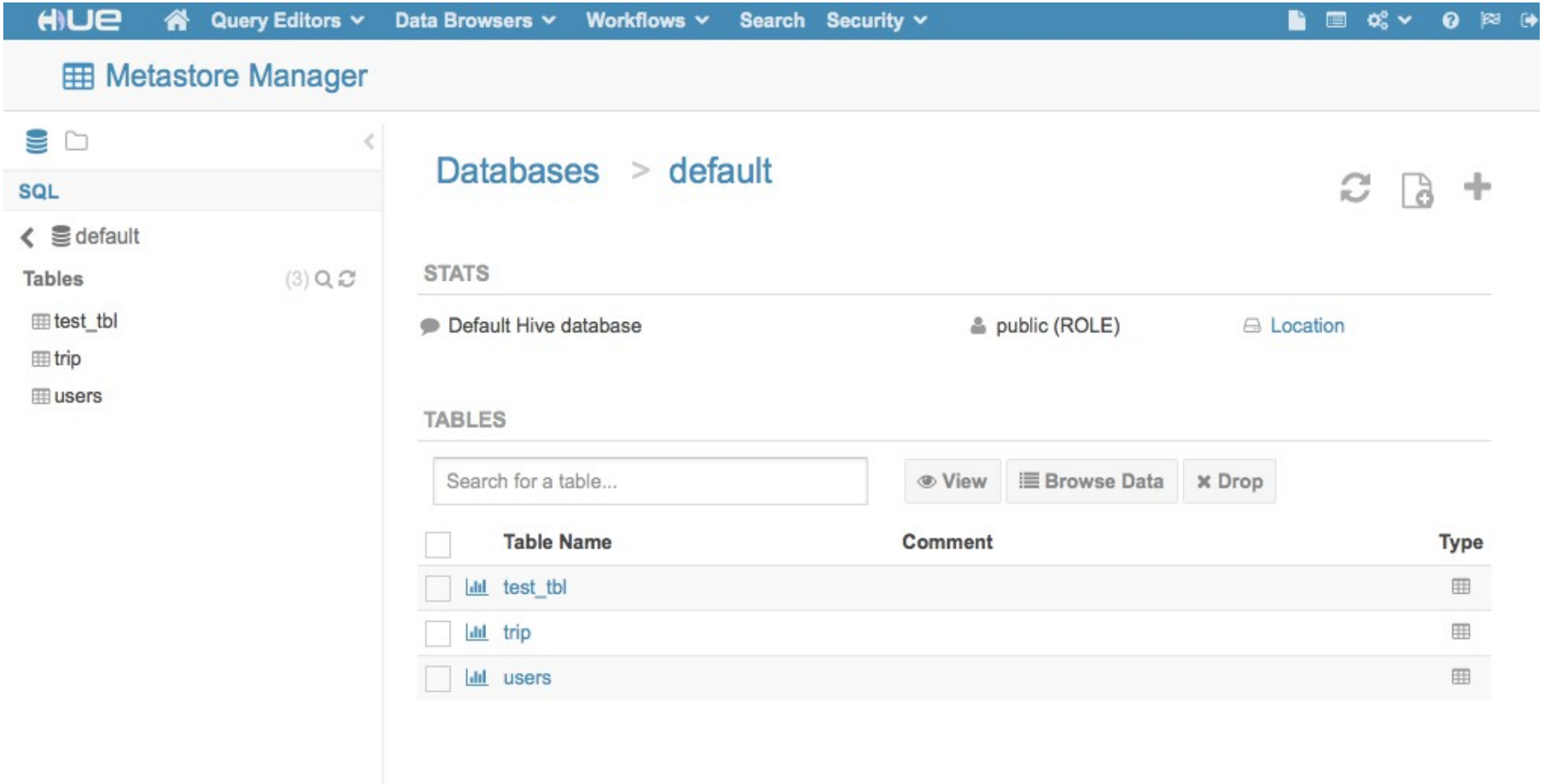
Create Table : Done

Databases > default > trip

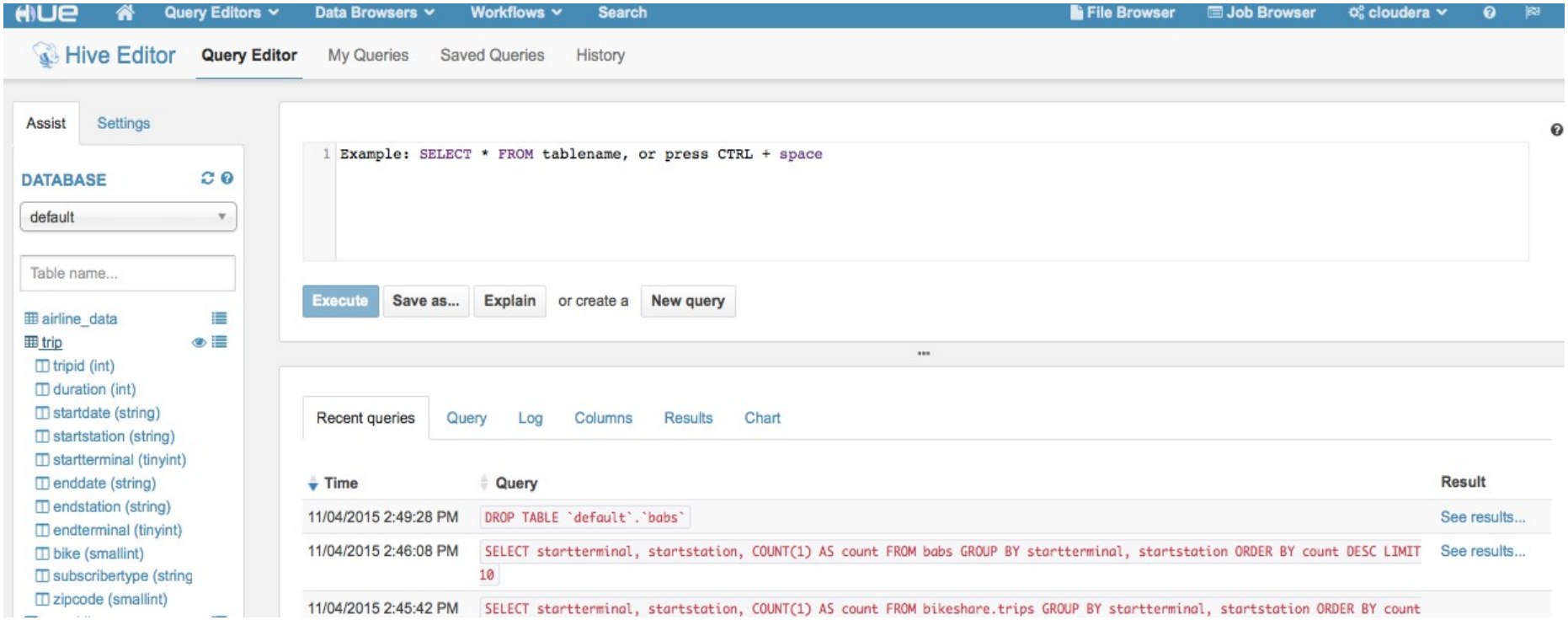
Comment: Bay Area Bike Share

Columns | Sample | Properties

	Name	Type	Comment
0	tripid	int	
1	duration	int	
2	startdate	string	
3	startstation	string	
4	startterminal	tinyint	
5	enddate	string	
6	endstation	string	
7	endterminal	tinyint	
8	bike	smallint	
9	subscribertype	string	
10	zipcode	smallint	

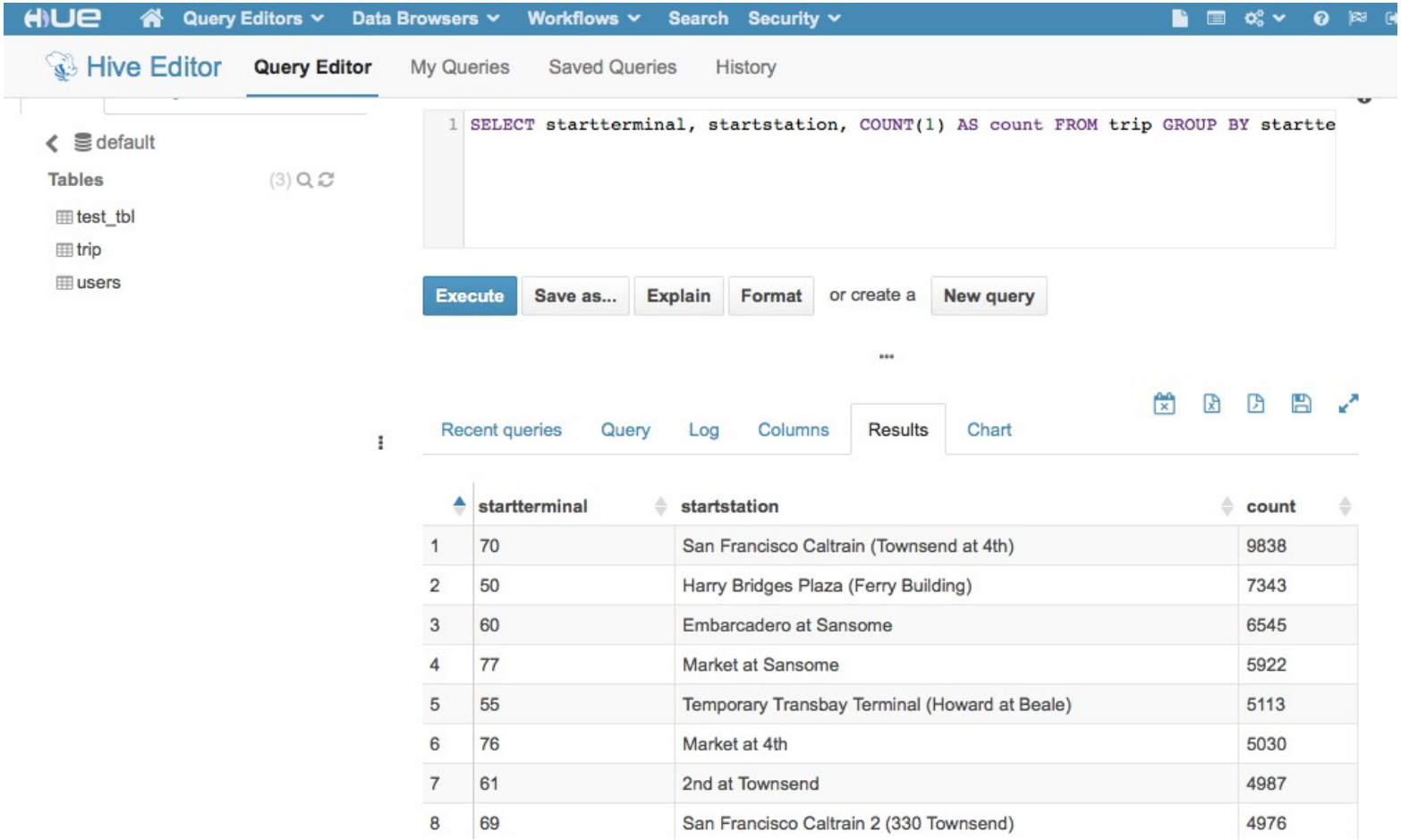

HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
SQL
default
Tables (3)
test_tbl
trip
users
Databases > default
STATS
Default Hive database
public (ROLE)
Location
TABLES
Search for a table...
View
Browse Data
Drop
Table Name
Comment
Type
test_tbl
trip
users

Starting Hive Editor



Find the top 10 most popular start stations based on the trip data

```
SELECT startterminal, startstation, COUNT(1) AS count FROM trip
GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10
```



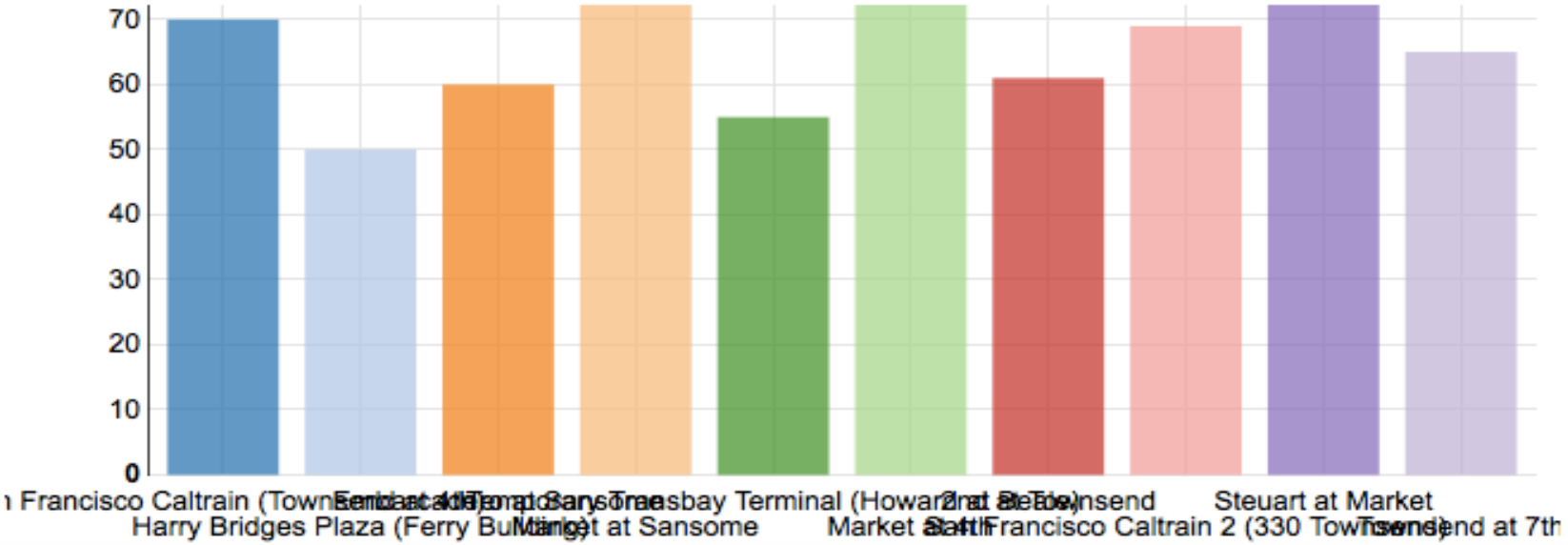
	startterminal	startstation	count
1	70	San Francisco Caltrain (Townsend at 4th)	9838
2	50	Harry Bridges Plaza (Ferry Building)	7343
3	60	Embarcadero at Sansome	6545
4	77	Market at Sansome	5922
5	55	Temporary Transbay Terminal (Howard at Beale)	5113
6	76	Market at 4th	5030
7	61	2nd at Townsend	4987
8	69	San Francisco Caltrain 2 (330 Townsend)	4976

```
1 SELECT startterminal, startstation, COUNT(1) AS count FROM trip GROUP BY startte
```

Execute | Save as... | Explain | Format | or create a | New query

...

Recent queries | Query | Log | Columns | Results | Chart



Find the total number of trips and average duration (in minutes) of those trips, grouped by hour

```
SELECT
    hour,
    COUNT(1) AS trips,
    ROUND(AVG(duration) / 60) AS avg_duration
FROM (
    SELECT
        CAST(SPLIT(SPLIT(t.startdate, ' ')[1], ':')[0] AS INT) AS
hour,
        t.duration AS duration
    FROM `bikeshare`.`trips` t
    WHERE
        t.startterminal = 70
        AND
        t.duration IS NOT NULL
    ) r
GROUP BY hour
ORDER BY hour ASC;
```

Lecture
Understanding Pig

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Introduction

A high-level platform for creating MapReduce programs Using Hadoop

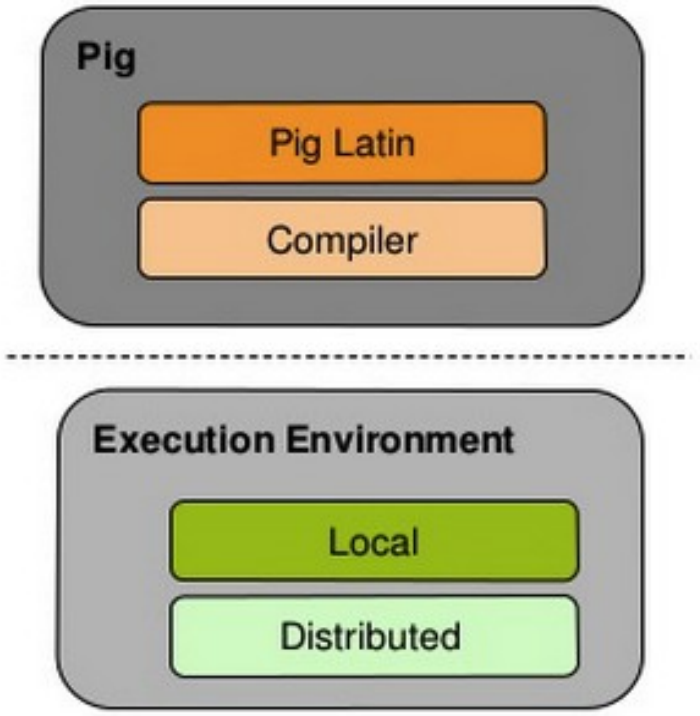
Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns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



Hive.apache.org

Running Pig

- **Script**

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

  pig
- **Embedded**

  Writing a java program

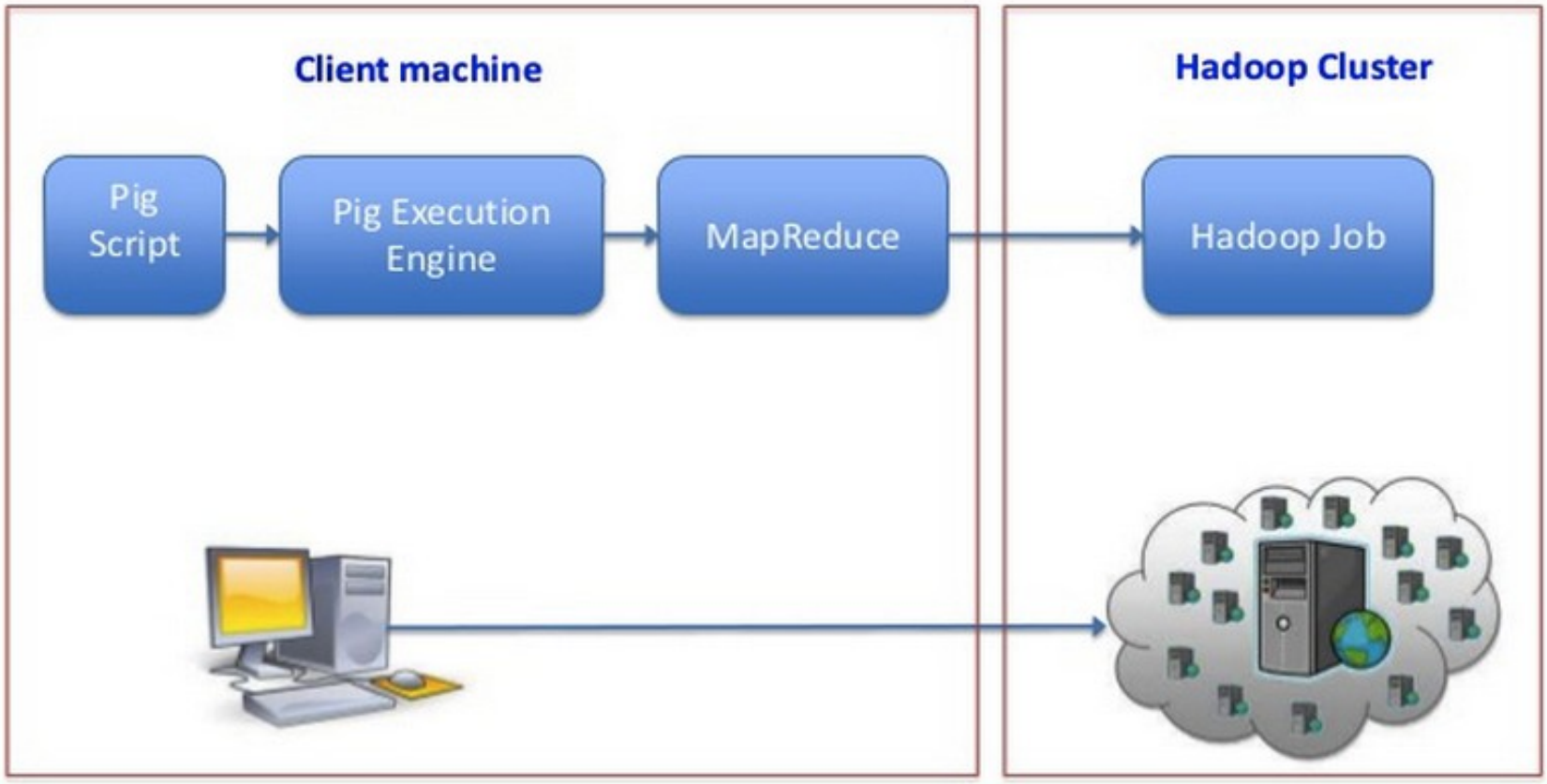
Hive.apache.org

Pig Latin

```
Users = load'users'as (name, age);
Fltrd = filter Users by
        age >= 18 and age <= 25;
Pages = load 'pages' as (user, url);
Jnd = joinFltrdby name, Pages by user;
Grpd = groupJndbyurl;
Smmd = foreachGrpdgenerate group,
COUNT(Jnd) as clicks;
Srtd = orderSmmdby clicks desc;
Top5 = limitSrtd 5;
store Top5 into'top5sites';
```

Hive.apache.org

Pig Execution Stages



Source Introduction to Apache Hadoop-Pig: PrashantKommireddi

Hive.apache.org

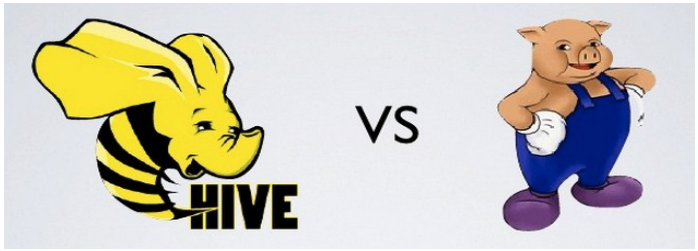
Why Pig?

- **Makes writing Hadoop jobs easier**
  - 5% of the code, 5% of the time
  - 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

Hive.apache.org

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

Hive.apache.org

Hands-On: Running a Pig script

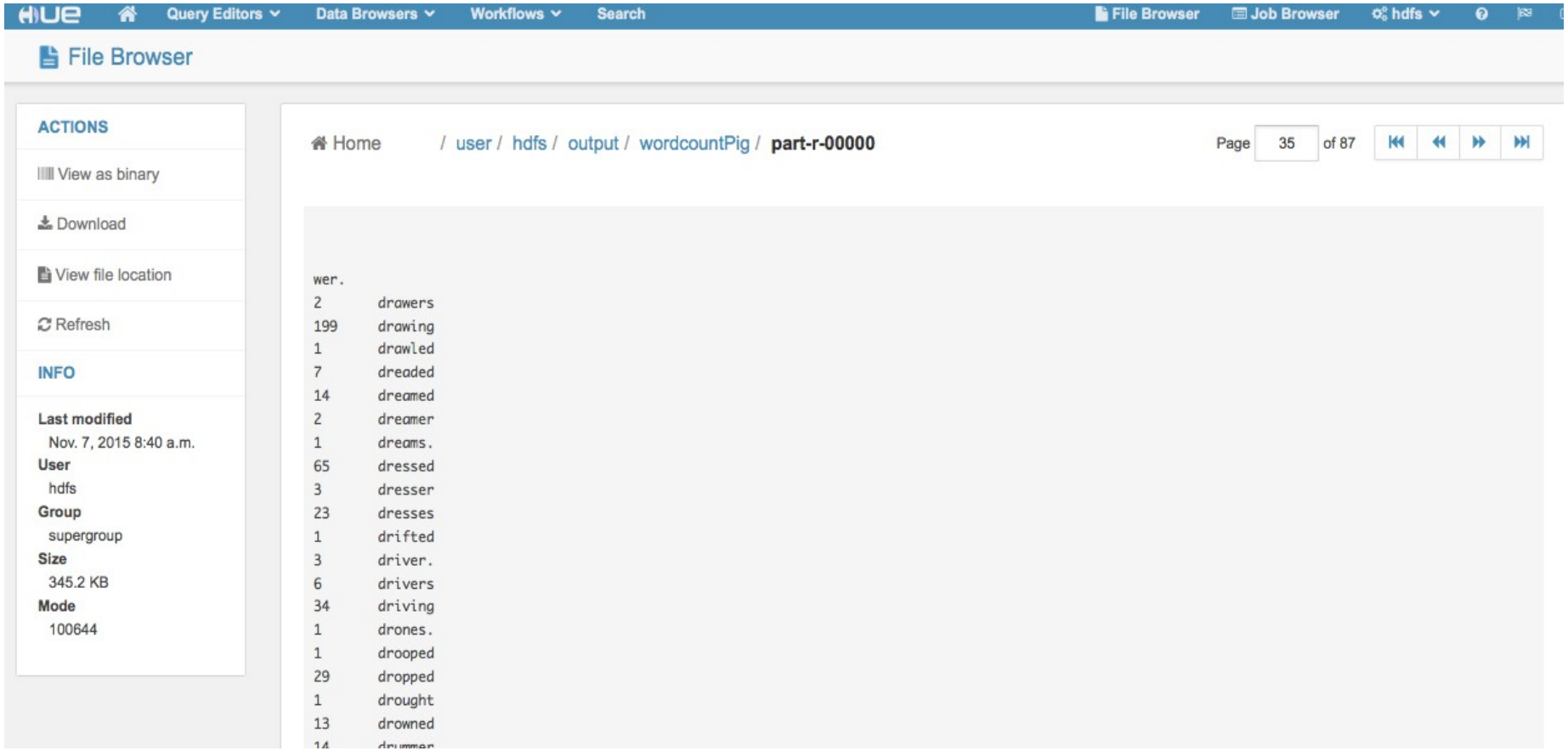
Starting Pig Command Line

```
$  pig -x mapreduce

2013-08-01 10:29:00,027 [main] INFO  org.apache.pig.Main - Apache Pig
version 0.11.1 (r1459641) compiled Mar 22 2013, 02:13:53

2013-08-01 10:29:00,027 [main] INFO  org.apache.pig.Main - Logging error
messages to: /home/hdadmin/pig_1375327740024.log

2013-08-01 10:29:00,066 [main] INFO  org.apache.pig.impl.util.Utils -
Default bootup file /home/hdadmin/.pigbootup not found

2013-08-01 10:29:00,212 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting
to hadoop file system at: file:///

grunt>
```

Writing a Pig Script for wordcount

```
A = load '/user/cloudera/input/*';
B = foreach A generate flatten(TOKENIZE((chararray)$0)) as
word;
C = group B by word;
D = foreach C generate COUNT(B), group;
store D into '.user/cloudera/output/wordcountPig';
```

```
2016-06-14 08:29:25,835 [main] INFO  org.apache.pig.backend.hadoop.executionengine.m
apReduceLayer.MapReduceLauncher - More information at: http://localhost:50030/jobdet
ails.jsp?jobid=job_1465875170640_0004
2016-06-14 08:29:25,871 [main] INFO  org.apache.pig.backend.hadoop.executionengine.m
apReduceLayer.MapReduceLauncher - 0% complete
2016-06-14 08:29:41,625 [main] INFO  org.apache.pig.backend.hadoop.executionengine.m
apReduceLayer.MapReduceLauncher - 50% complete
2016-06-14 08:29:51,359 [main] INFO  org.apache.pig.backend.hadoop.executionengine.m
apReduceLayer.MapReduceLauncher - 100% complete
2016-06-14 08:29:51,362 [main] INFO  org.apache.pig.tools.pigstats.SimplePigStats -
Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.6.0-cdh5.7.0  0.12.0-cdh5.7.0 root    2016-06-14 08:29:18     2016-06-14 08:29:51G
ROUP_BY

Success!
```


Query Editors
Data Browsers
Workflows
Search
File Browser
Job Browser
hdfs
File Browser
ACTIONS
View as binary
Download
View file location
Refresh
INFO
Last modified
Nov. 7, 2015 8:40 a.m.
User
hdfs
Group
supergroup
Size
345.2 KB
Mode
100644
Home / user / hdfs / output / wordcountPig / part-r-00000
Page 35 of 87
wer.
2 drawers
199 drawing
1 drawled
7 dreaded
14 dreamed
2 dreamer
1 dreams.
65 dressed
3 dresser
23 dresses
1 drifted
3 driver.
6 drivers
34 driving
1 drones.
1 drooped
29 dropped
1 drought
13 drowned

Lecture
Understanding Impala

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Introduction

open source massively parallel processing (MPP) SQL query engine

Cloudera Impala is a query engine that runs on Apache Hadoop. Impala brings scalable parallel database technology to Hadoop, enabling users to issue low-latency SQL queries to data stored in HDFS and Apache HBase without requiring data movement or transformation.

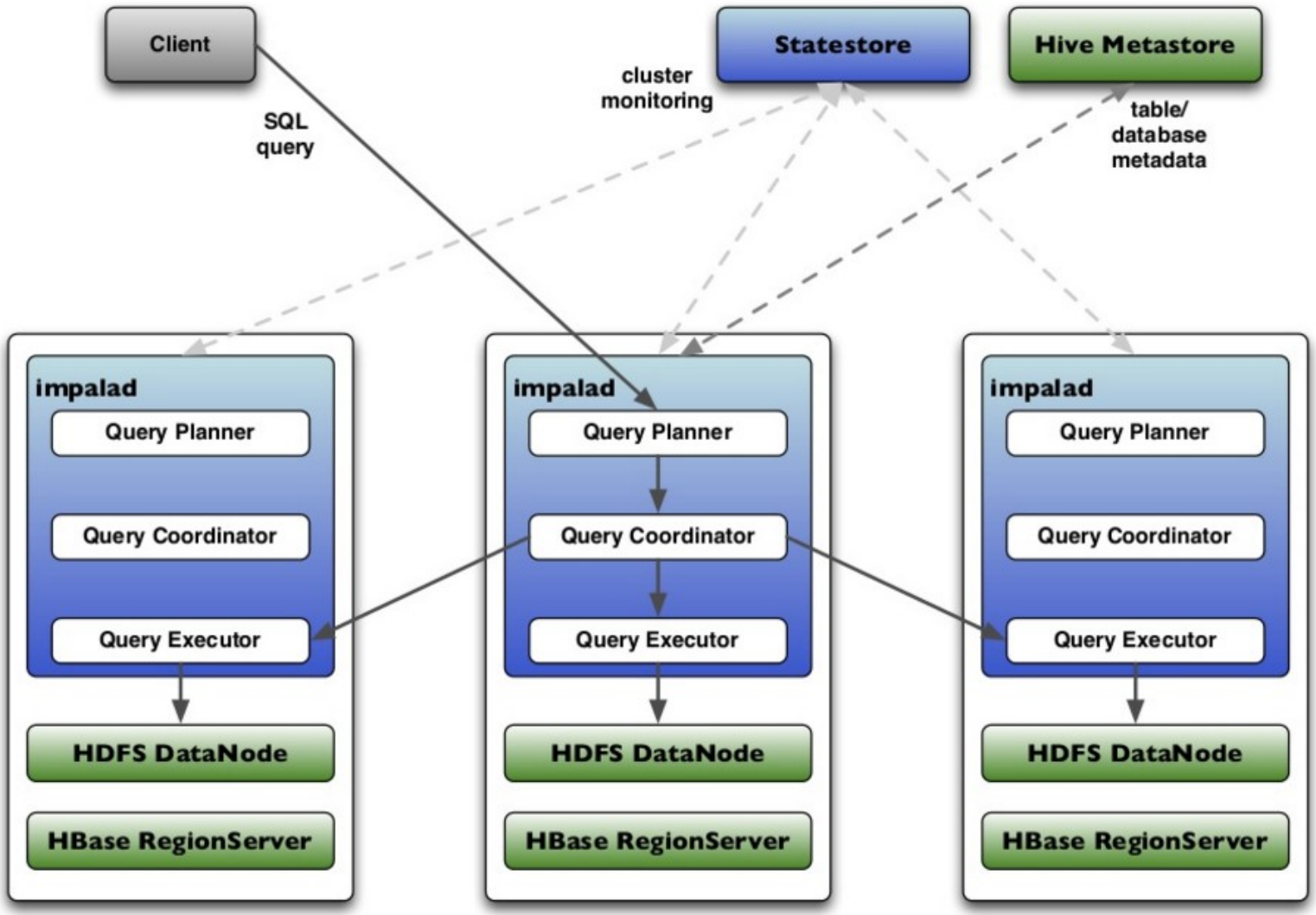
What is Impala?

- General--- purpose SQL engine
- Real--time queries in Apache Hadoop
- Opensource under Apache License
- Runs directly within Hadoop
- High performance
  - C++ instead of Java
  - Runtime code generator
  - Roughly 4-100 x Hive

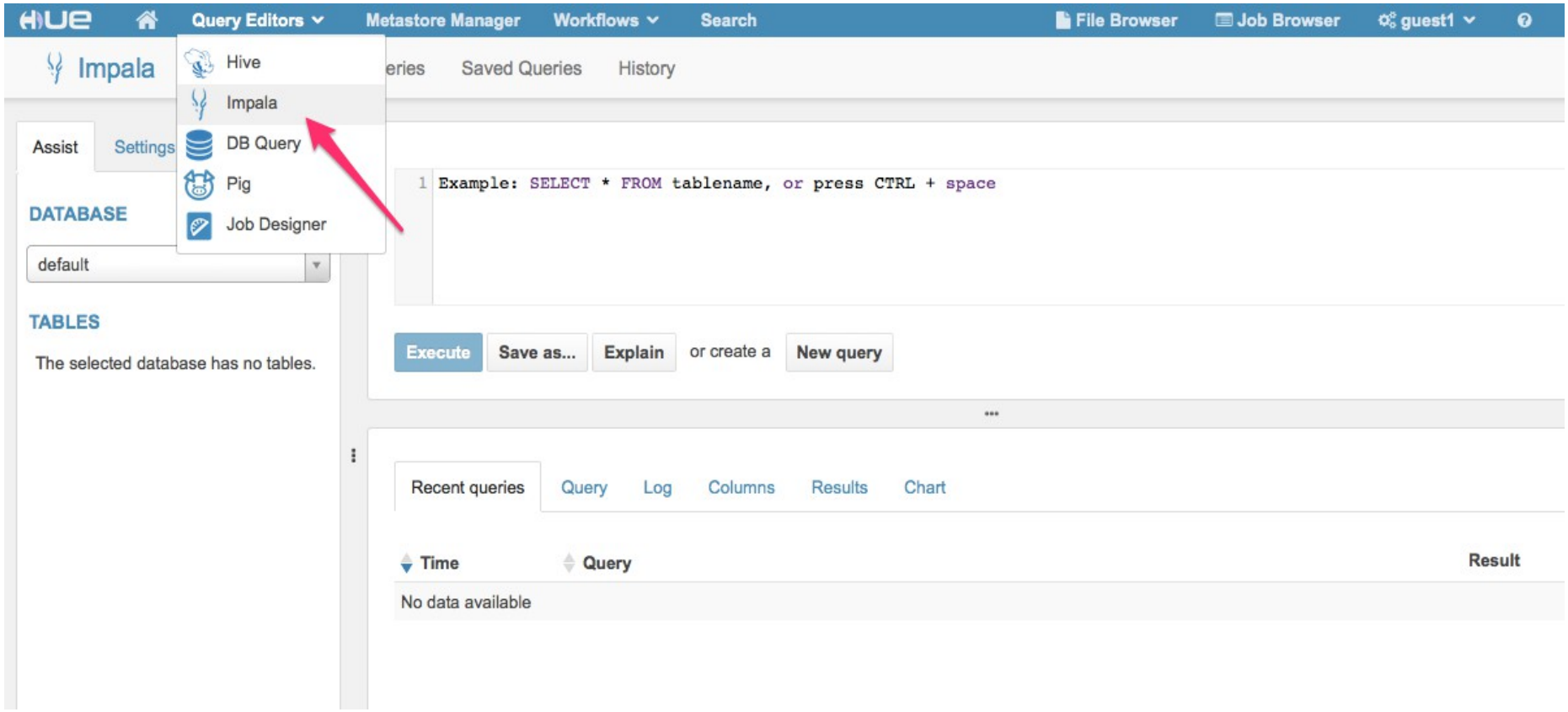
Impala Overview

- Impala daemon run on HDFS nodes
- Statestore (for cluster metadata) v.s. Metastore (for database metastore)
- Queries run on “revelants” nodes
- Support common HDFS file formats
- Submit quries via Hue/Beeswax
- No fault tolerant

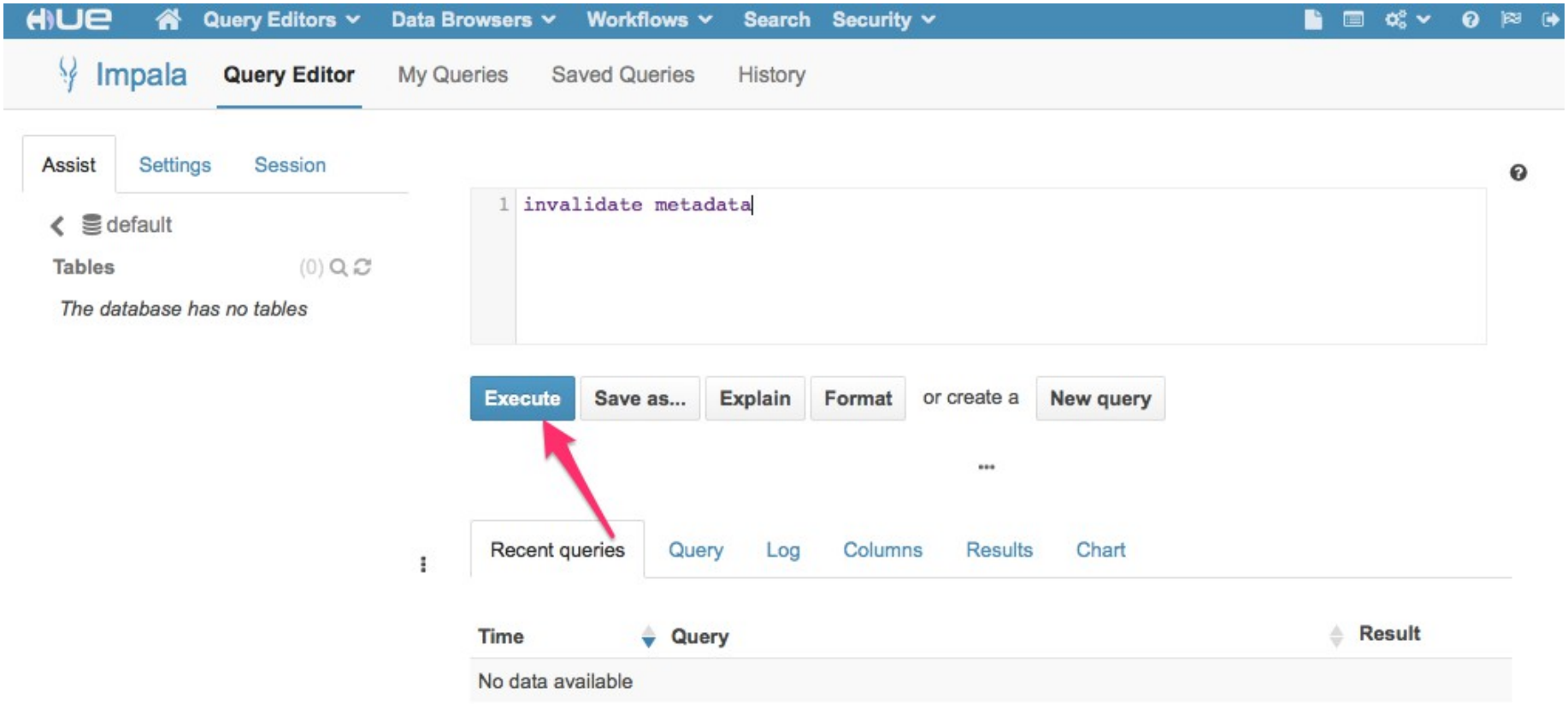
Impala Architecture



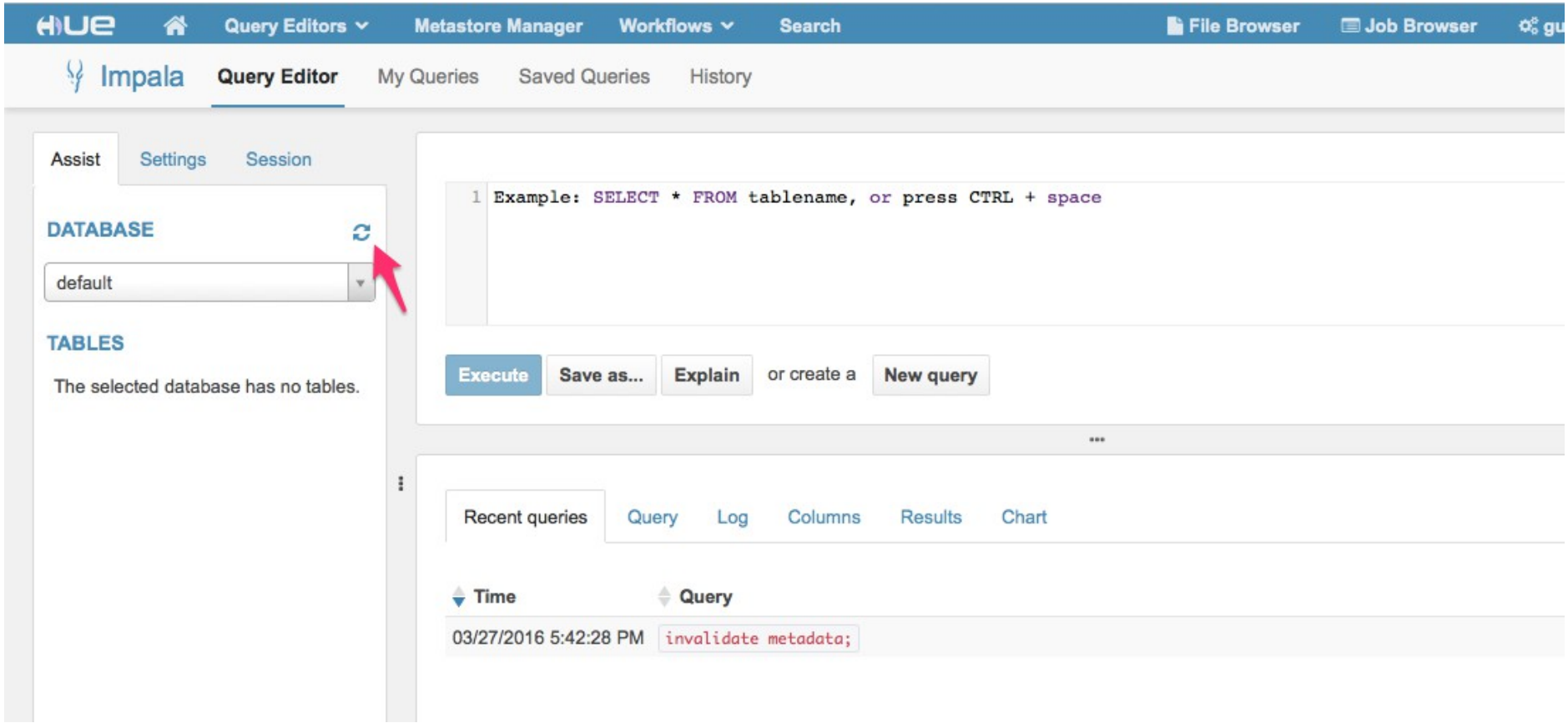
Start Impala Query Editor



Update the list of tables/metadata by excute the command invalidate metadata

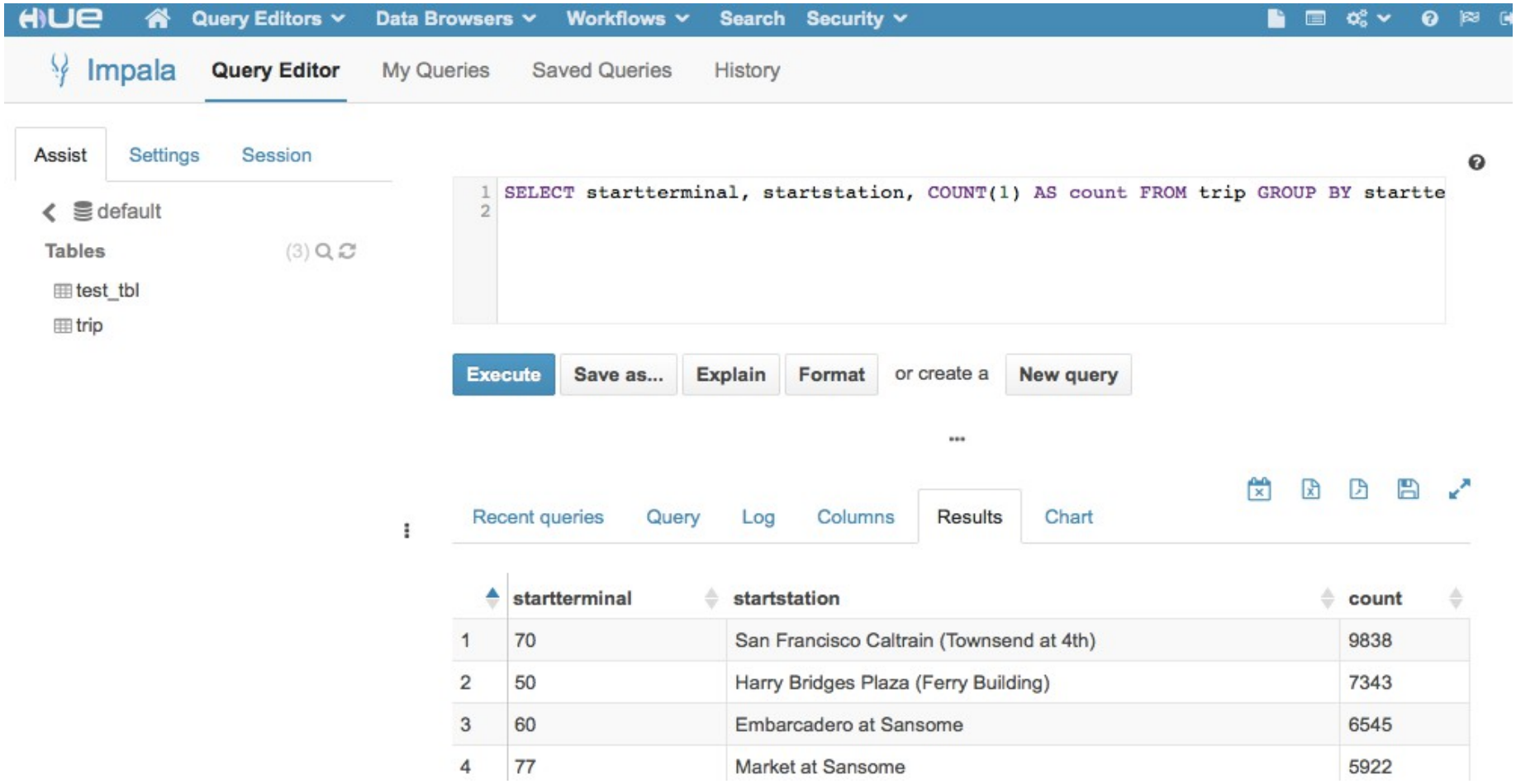


Restart Impala Query Editor and refresh the table list

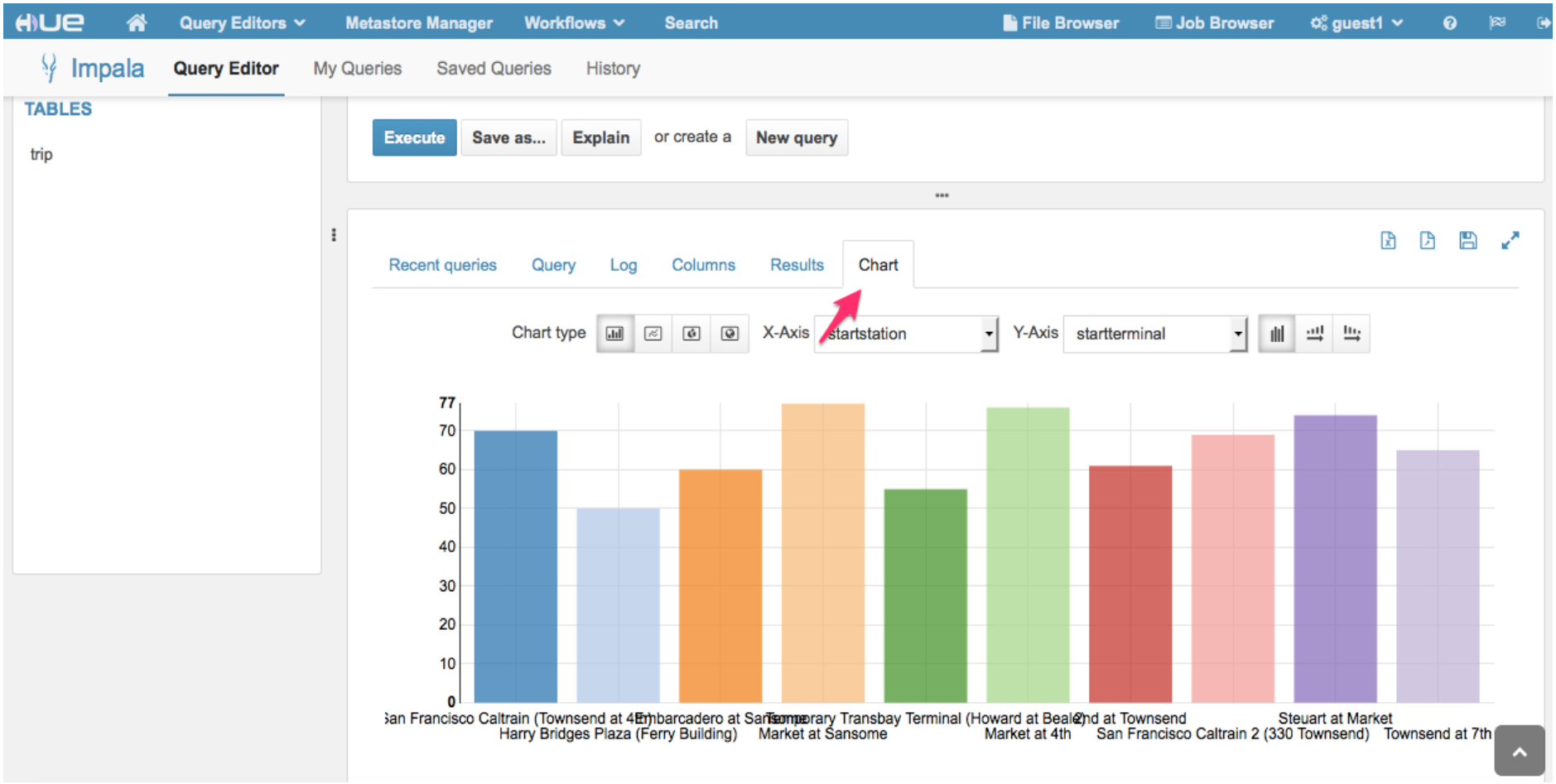


Find the top 10 most popular start stations based on the trip data: Using Impala

```
SELECT startterminal, startstation, COUNT(1) AS count FROM trip
GROUP BY startterminal, startstation ORDER BY count DESC LIMIT 10
```



	startterminal	startstation	count
1	70	San Francisco Caltrain (Townsend at 4th)	9838
2	50	Harry Bridges Plaza (Ferry Building)	7343
3	60	Embarcadero at Sansome	6545
4	77	Market at Sansome	5922


Query Editors
Metastore Manager
Workflows
Search
File Browser
Job Browser
guest1
Impala
Query Editor
My Queries
Saved Queries
History
TABLES
trip
Execute
Save as...
Explain
or create a
New query
Recent queries
Query
Log
Columns
Results
Chart
Chart type
X-Axis
startstation
Y-Axis
startterminal
77
70
60
50
40
30
20
10
0
Harry Bridges Plaza (Ferry Building)
Market at Sansome
Market at 4th
San Francisco Caltrain 2 (330 Townsend)
Steuart at Market
Townsend at 7th

Find the total number of trips and average duration (in minutes) of those trips, grouped by hour

```
SELECT
    hour,
    COUNT(1) AS trips,
    ROUND(AVG(duration) / 60) AS avg_duration
FROM (
    SELECT
        CAST(SPLIT(SPLIT(t.startdate, ' ')[1], ':')[0] AS INT) AS
hour,
        t.duration AS duration
    FROM `bikeshare`.`trips` t
    WHERE
        t.startterminal = 70
        AND
        t.duration IS NOT NULL
    ) r
GROUP BY hour
ORDER BY hour ASC;
```

Lecture
Understanding Spark

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

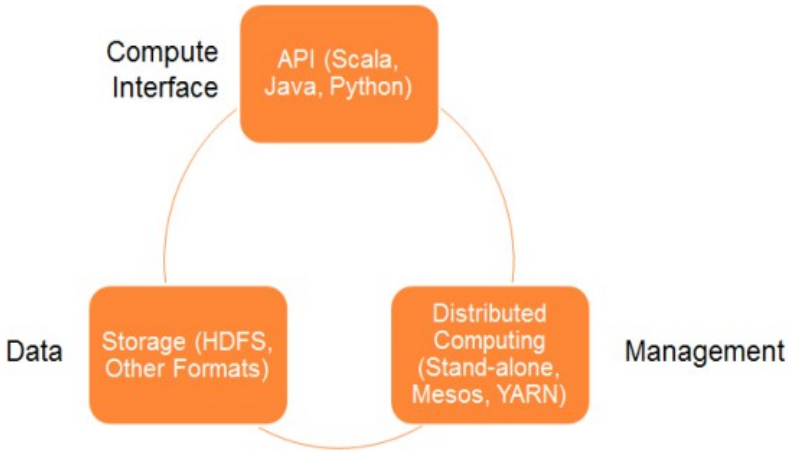
Introduction

A fast and general engine for large scale data processing

An open source big data processing framework built around speed, ease of use, and sophisticated analytics. Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times faster even when running on disk.

What is Spark?

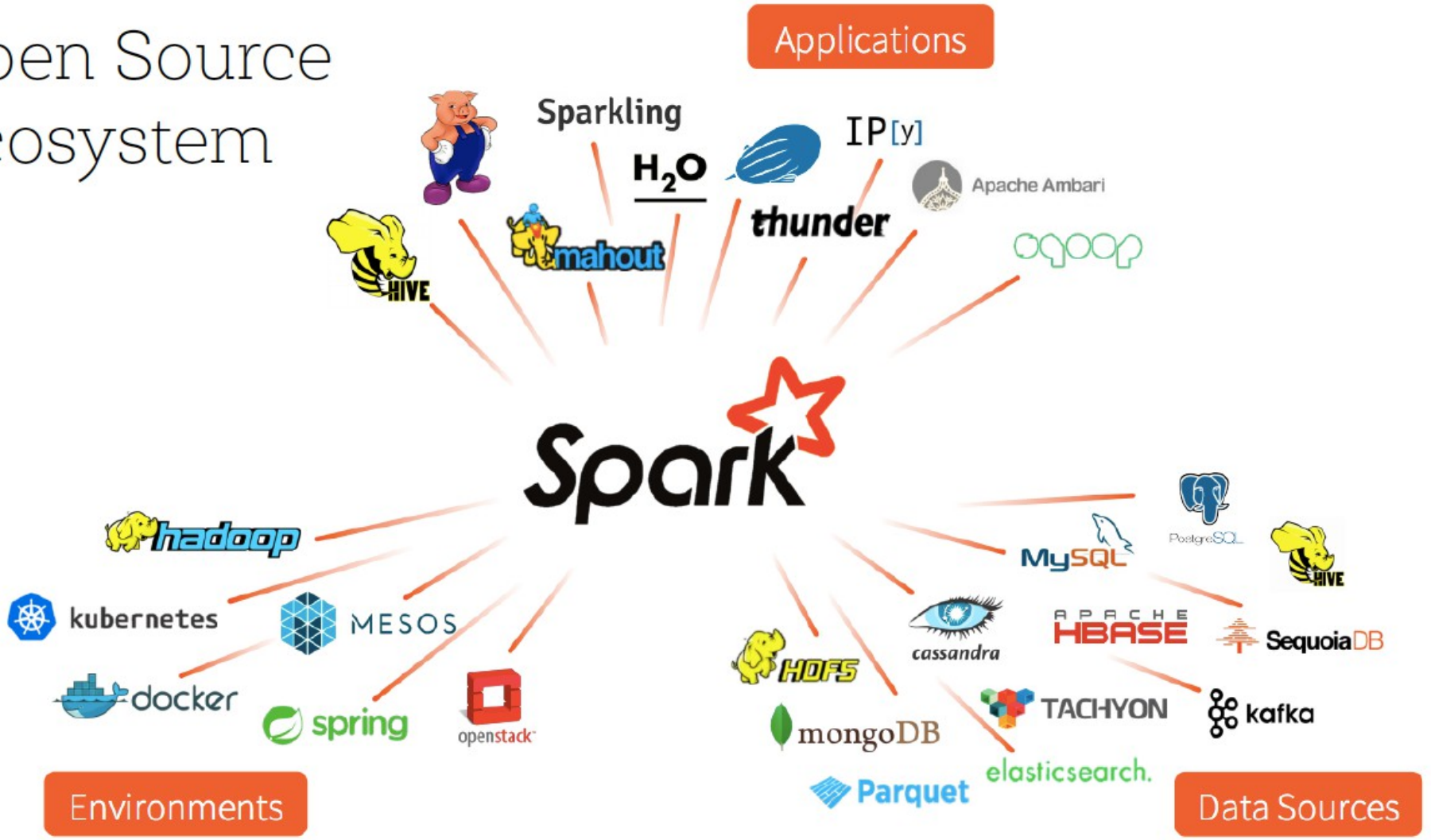
- Framework for distributed processing.
- In-memory, fault tolerant data structures
- Flexible APIs in Scala, Java, Python, SQL, R
- Open source



Why Spark?

- Handle Petabytes of data
- Significant faster than MapReduce
- Simple and intutive APIs
- General framework
  - Runs anywhere
  - Handles (most) any I/O
  - Interoperable libraries for specific use-cases

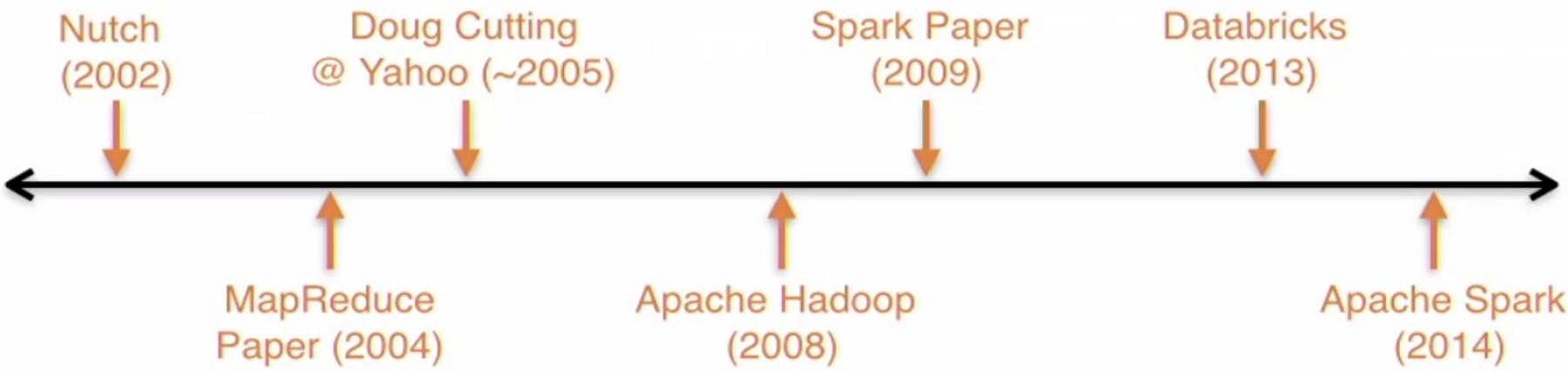
Open Source Ecosystem

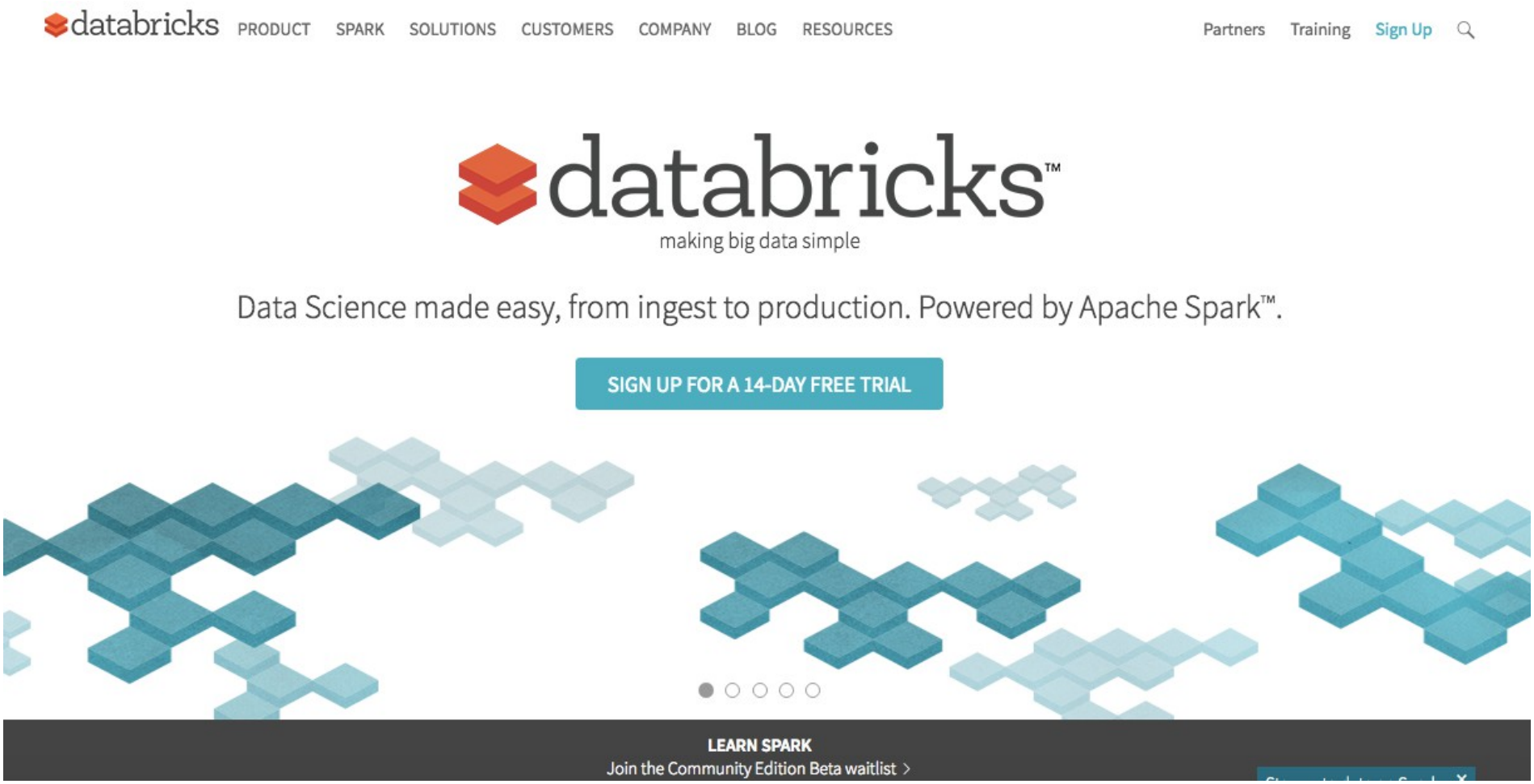


Source: Jump start into Apache Spark and Databricks

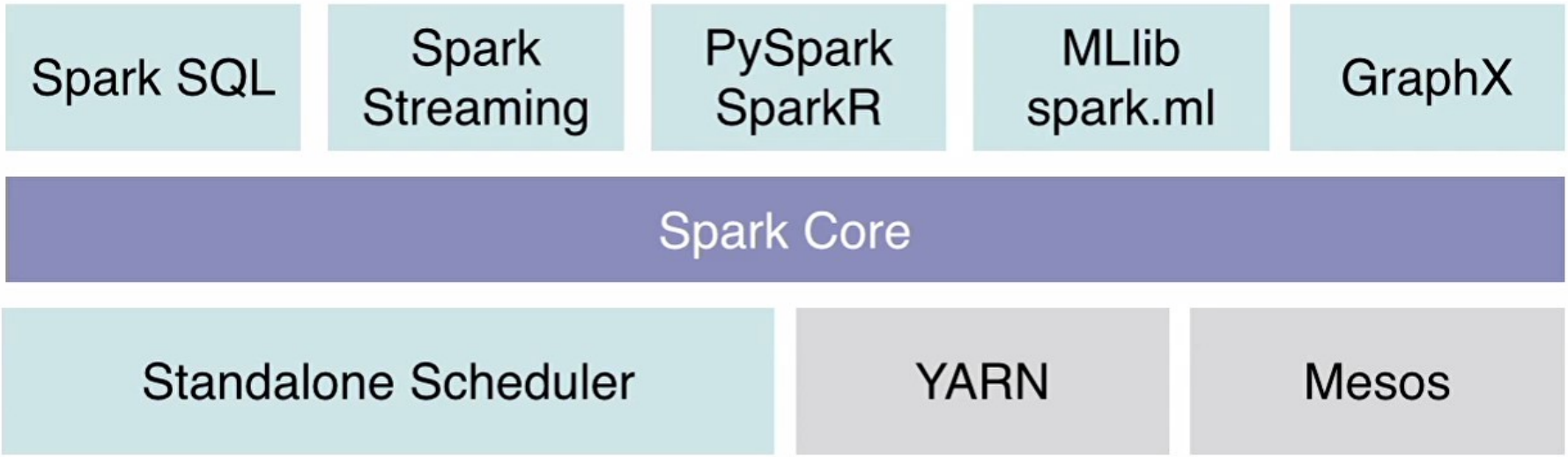
Spark: History

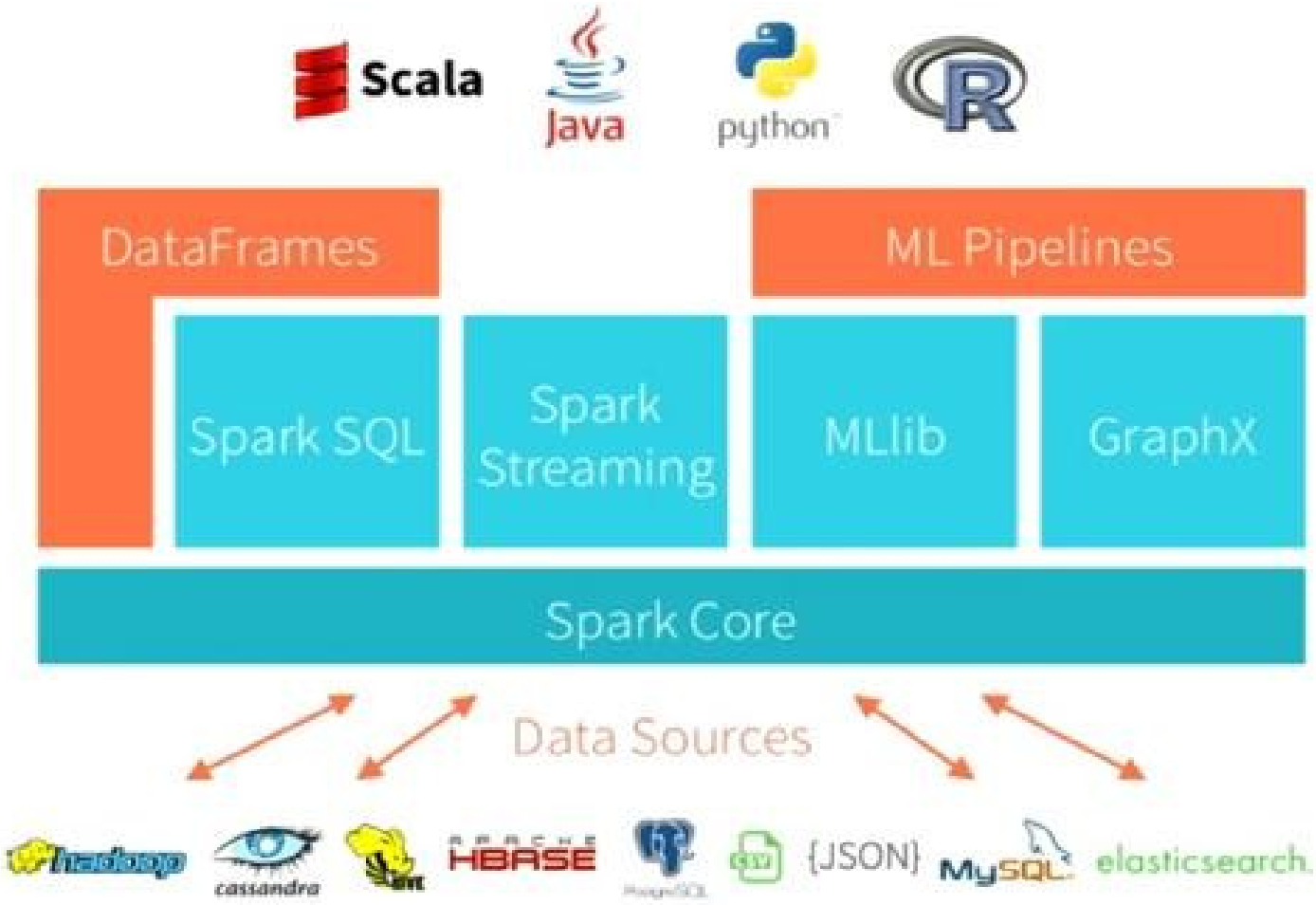
- Founded by AMPlab, UC Berkeley
- Created by Matei Zaharia (PhD Thesis)
- Maintained by Apache Software Foundation
- Commercial support by Databricks



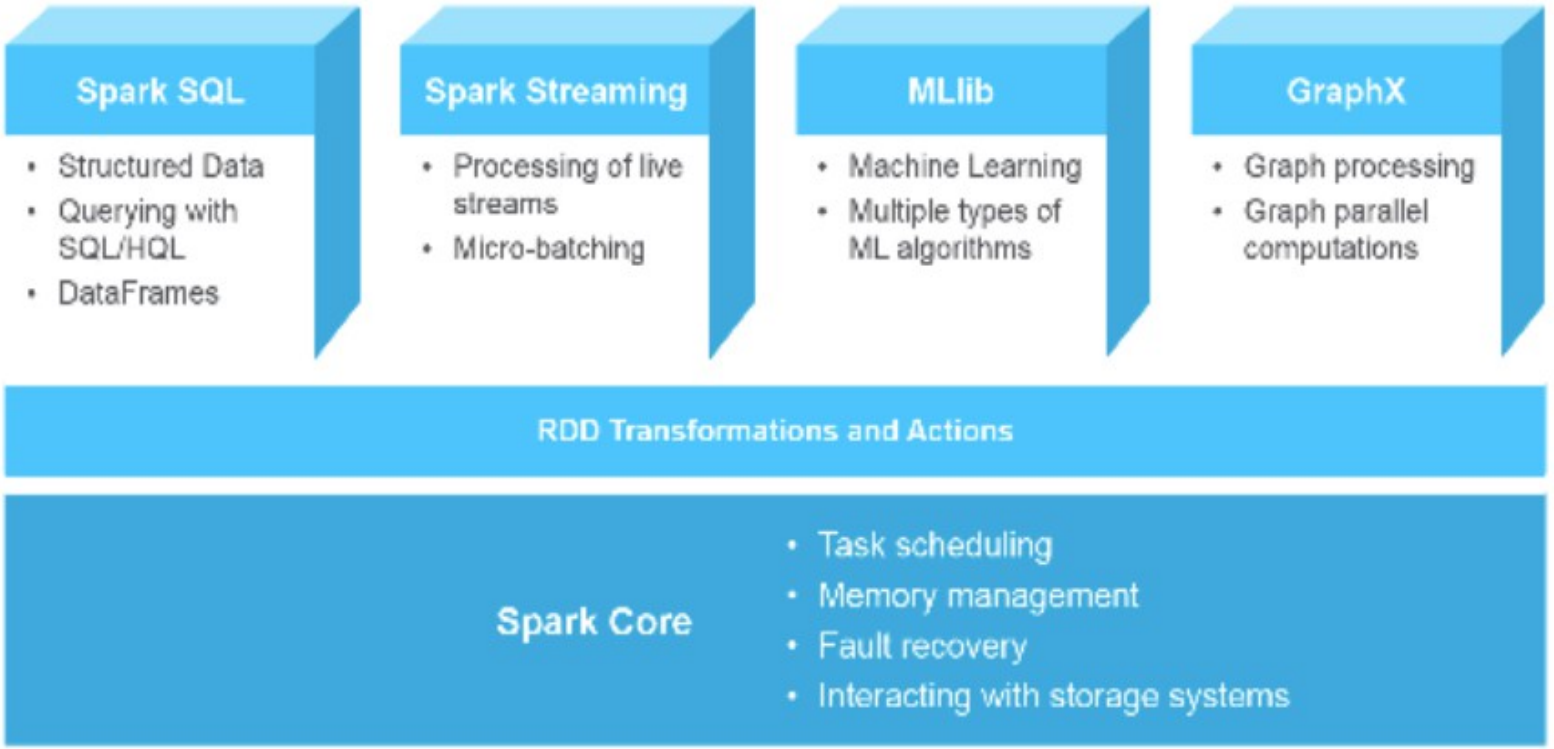

databricks
PRODUCT
SPARK
SOLUTIONS
CUSTOMERS
COMPANY
BLOG
RESOURCES
Partners
Training
Sign Up
databricks™
making big data simple
Data Science made easy, from ingest to production. Powered by Apache Spark™.
SIGN UP FOR A 14-DAY FREE TRIAL
LEARN SPARK
Join the Community Edition Beta waitlist >

Spark Platform



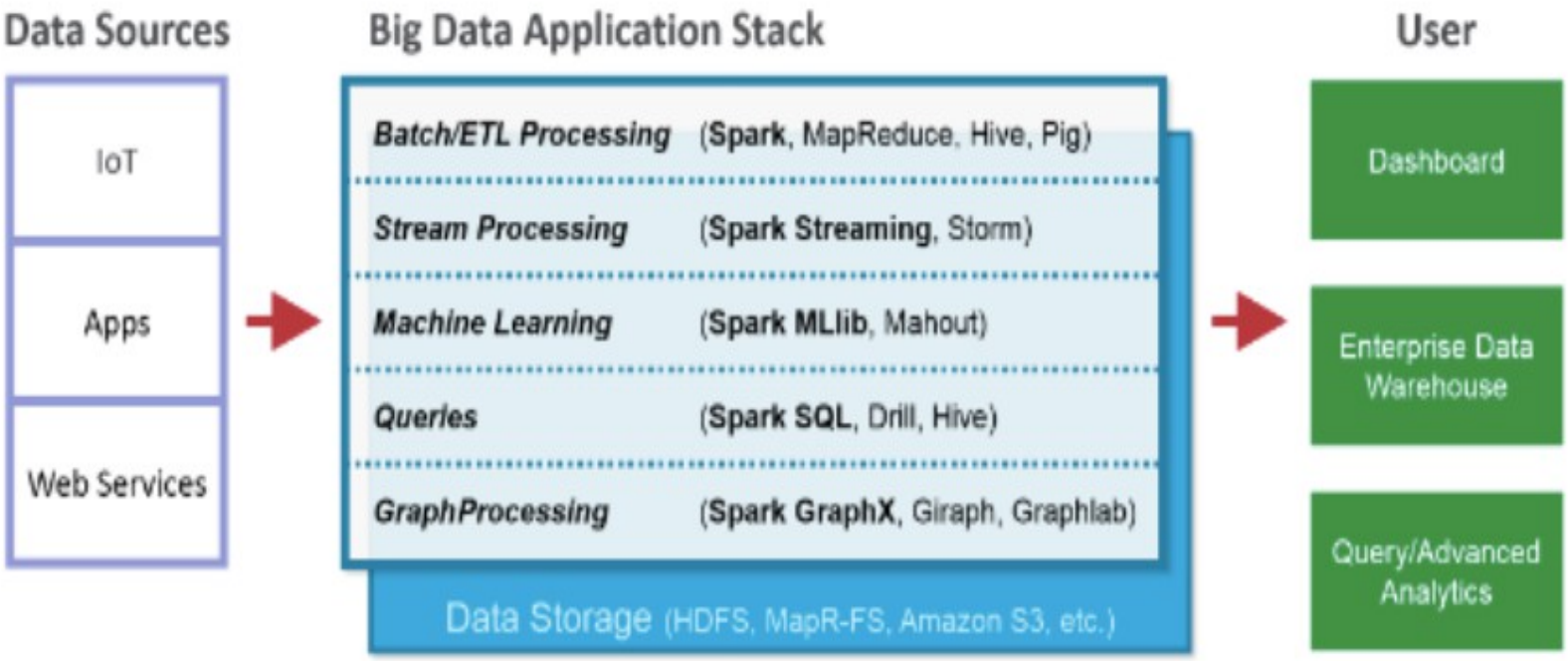

Scala
Java
python
DataFrames
ML Pipelines
Spark SQL
Spark Streaming
MLlib
GraphX
Spark Core
Data Sources
hadoop
cassandra
HBASE
{JSON}
MySQL
elasticsearch

Spark Platform



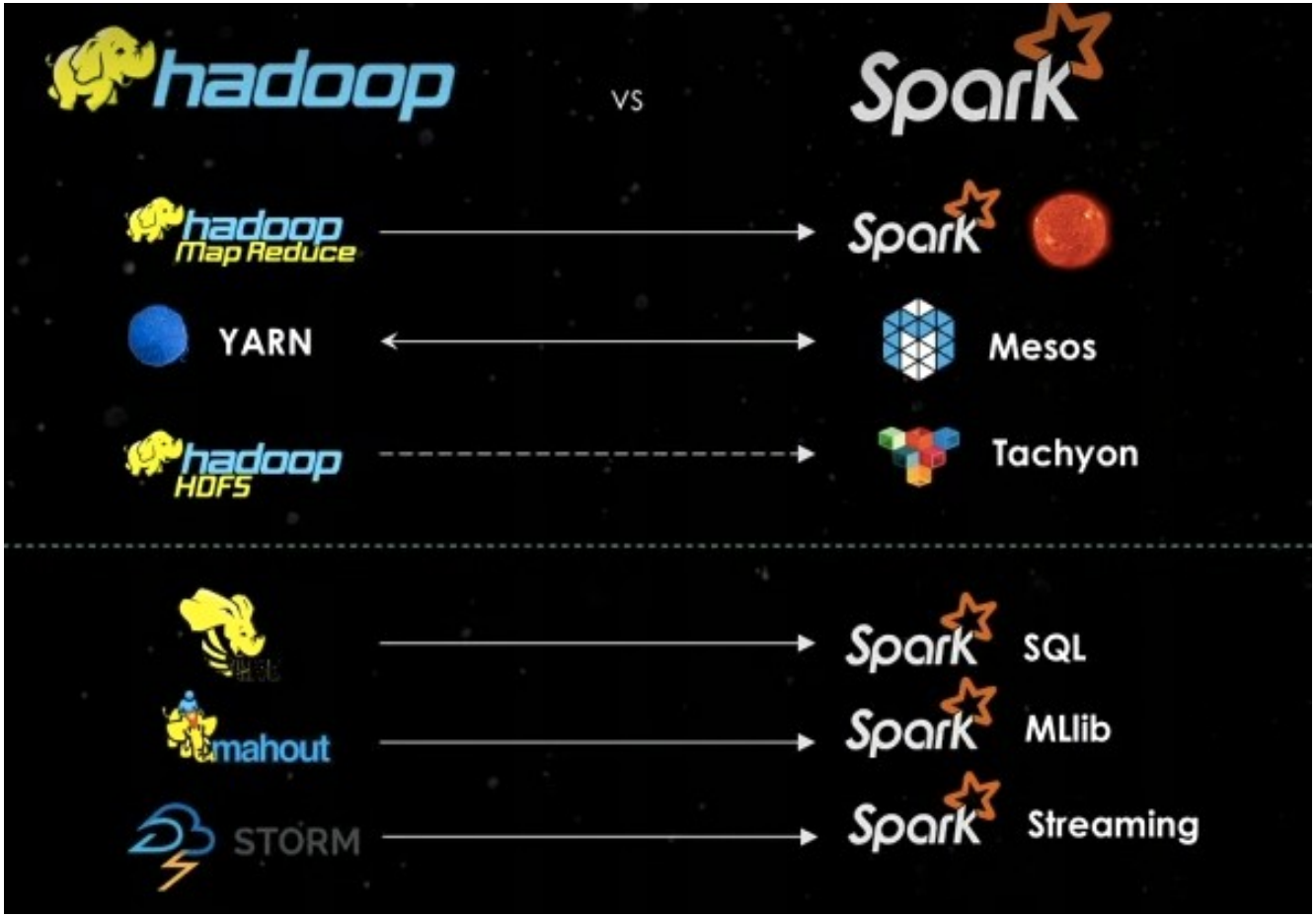
Source: MapR Academy

thanachart@imcinstitute.com



Source: MapR Academy

thanachart@imcinstitute.com



Source: TRAINING Intro to Apache Spark - Brian Clapper

thanachart@imcinstitute.com

Large-Scale Usage

Largest cluster
8000 Nodes (Tencent)

Largest single job
1 PB (Alibaba, Databricks)

Top Streaming Intake
1 TB/hour (HHMI Janelia Farm)

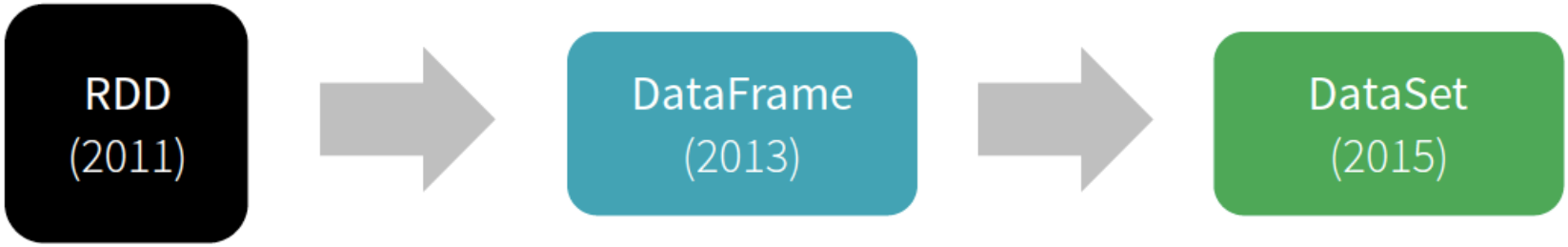
2014 On-Disk Sort Record
Fastest Open Source Engine for sorting a PB

Source: Jump start into Apache Spark and Databricks

Notable Users

Source: Jump start into Apache Spark and Databricks

History of Spark APIs



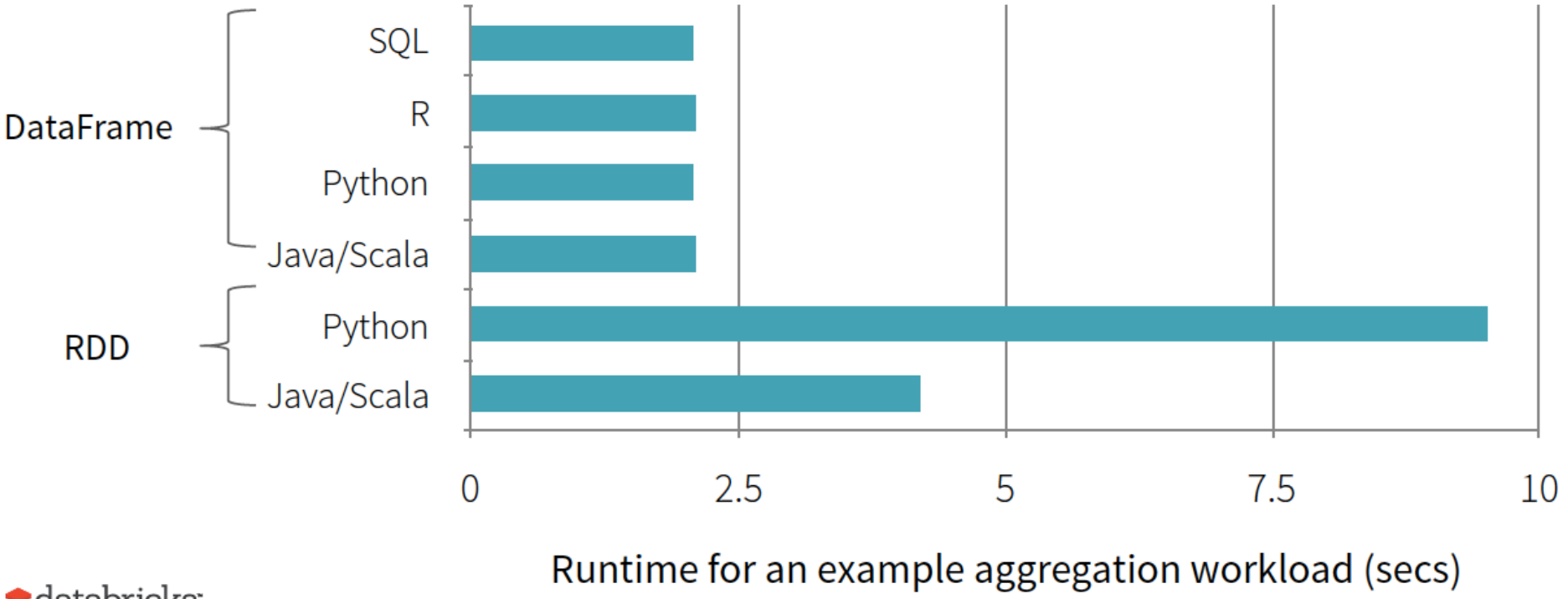
- Distribute collection of JVM objects
- Functional Operators (map, filter, etc.)

- Distribute collection of Row objects
- Expression-based operations and UDFs
- Logical plans and optimizer
- Fast/efficient internal representations

- Internally rows, externally JVM objects
- "Best of both worlds" **type safe + fast**

Source: Jump start into Apache Spark and Databricks

Benefit of Logical Plan: Performance Parity Across Languages



Source: Jump start into Apache Spark and Databricks

What is a RDD?

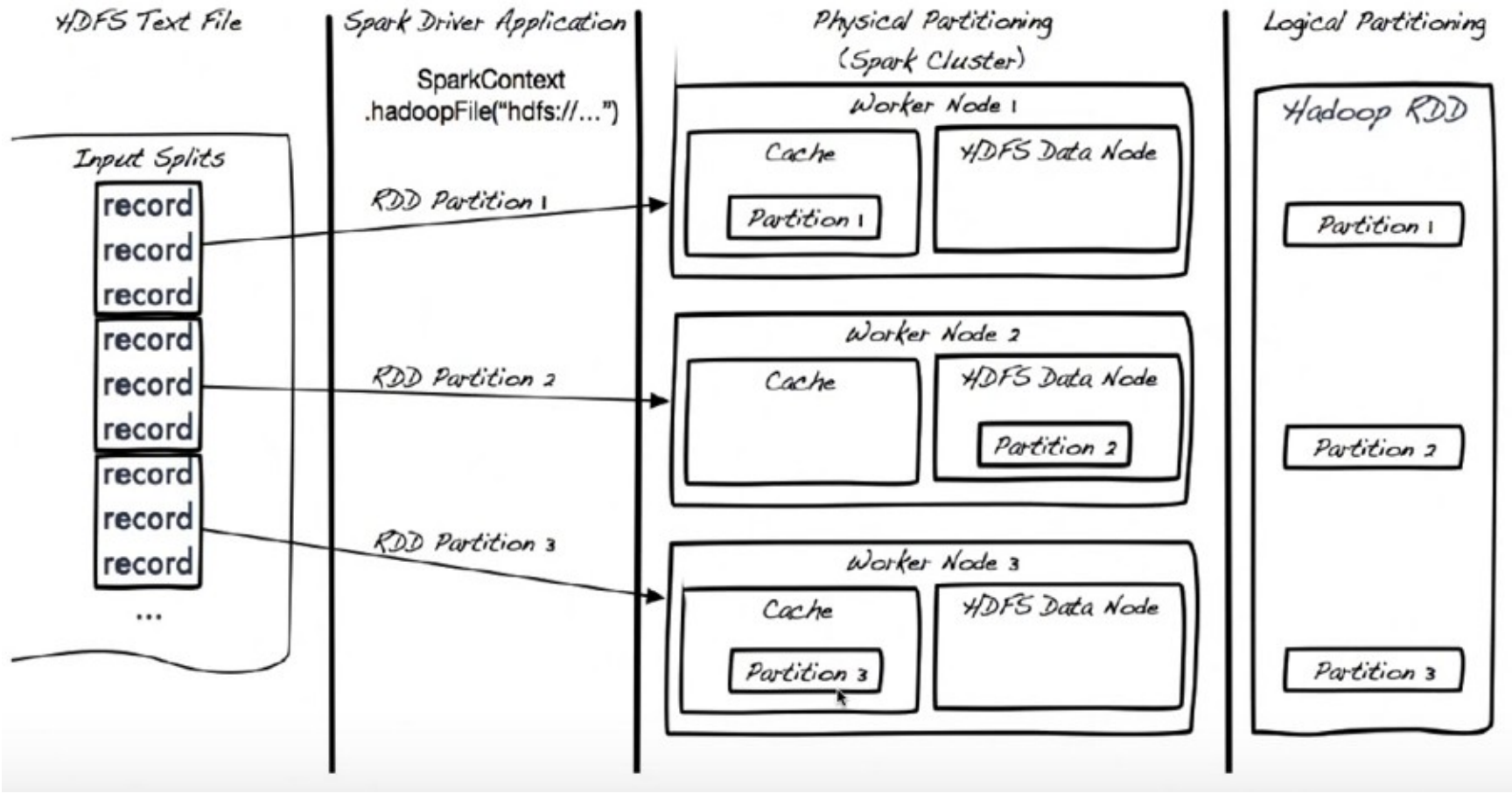
- Resilient: if the data in memory (or on a node) is lost, it can be recreated.
- Distributed: data is chucked into partitions and stored in memory acress the custer.
- Dataset: initial data can come from a table or be created programmatically

RDD:

- Fault tollerant
- Immutable
- Three methods for creating RDD:
  - Parallelizing an existing correction
  - Referencing a dataset
  - Transformation from an existing RDD
- Types of files supported:
  - Text files
  - SequenceFiles
  - Hadoop InputFormat

RDD Creation

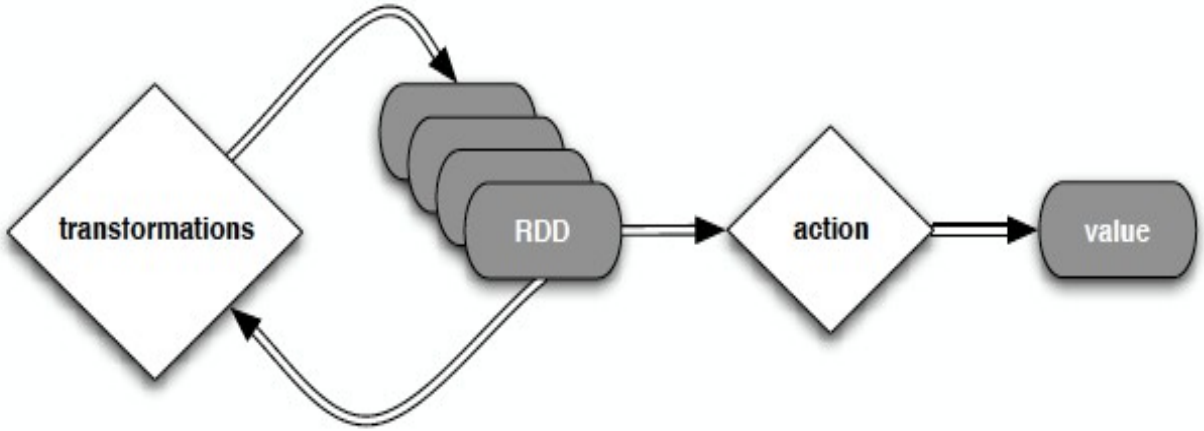
```
hdfsData = sc.textFile("hdfs://data.txt")
```



Source: Pspark: A brain-friendly introduction

RDD: Operations

- Transformations: transformations are lazy (not computed immediately)
- Actions: the transformed RDD gets recomputed when an action is run on it (default)



Direct Acyclic Graph (DAG)

- View the DAG
  linesLength.toDebugString
- Sample DAG

```
res5: String =
MappedRDD[4] at map at <console>:16 (3 partitions)
  MappedRDD[3] at map at <console>:16 (3 partitions)
    FilteredRDD[2] at filter at <console>:14 (3 partitions)
      MappedRDD[1] at textFile at <console>:12 (3 partitions)
        HadoopRDD[0] at textFile at <console>:12 (3 partitions)
```

Functions Deconstructed

```
import random
flips = 1000000

# lazy eval
coins = xrange(flips)

# lazy eval, nothing executed
heads = sc.parallelize(coins) \
          .map(lambda i: random.random()) \
          .filter(lambda r: r < 0.51) \
          .count()
```

Python Generator

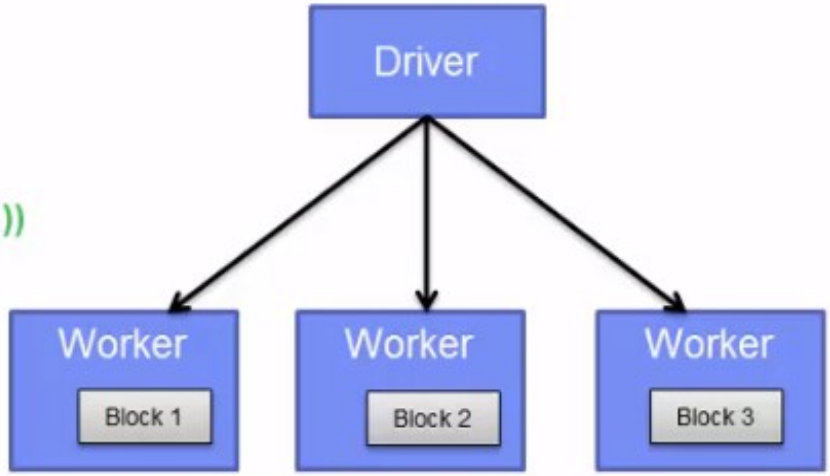
Create RDD

Transformations

Action (materialize result)

What happens when an action is executed

```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Cache
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```

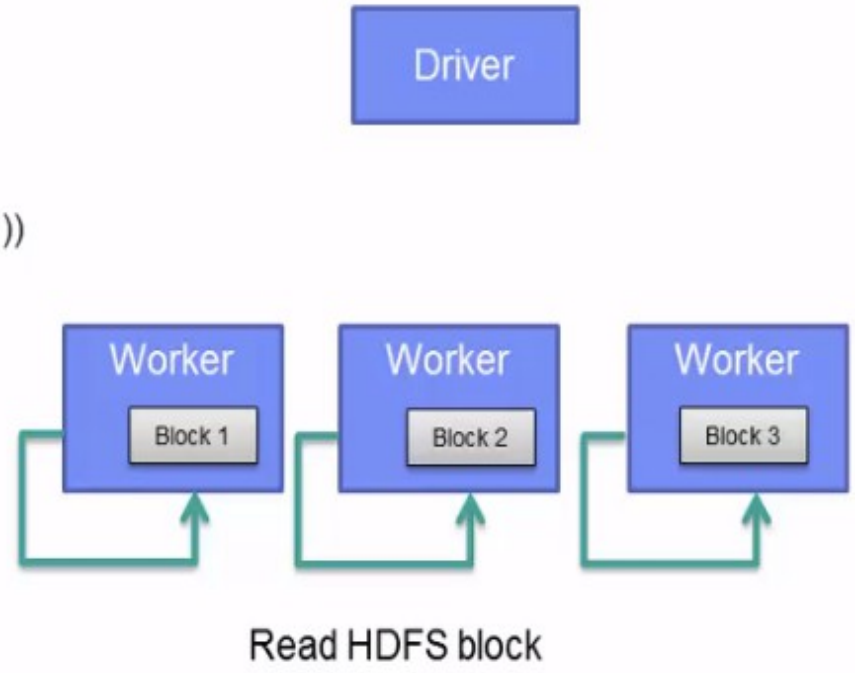


Driver sends the code to be executed on each block

Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

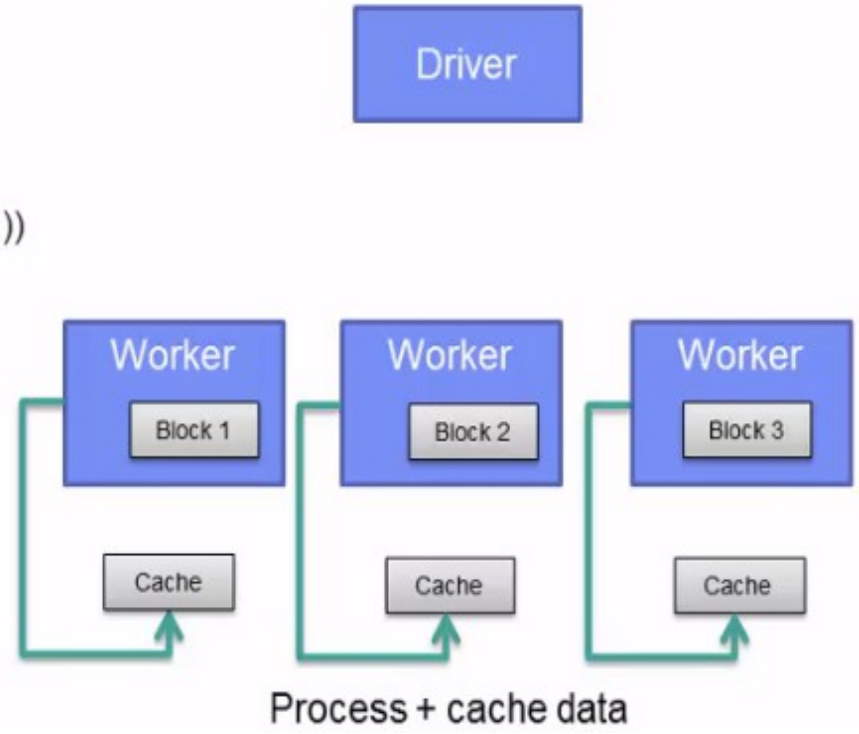
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

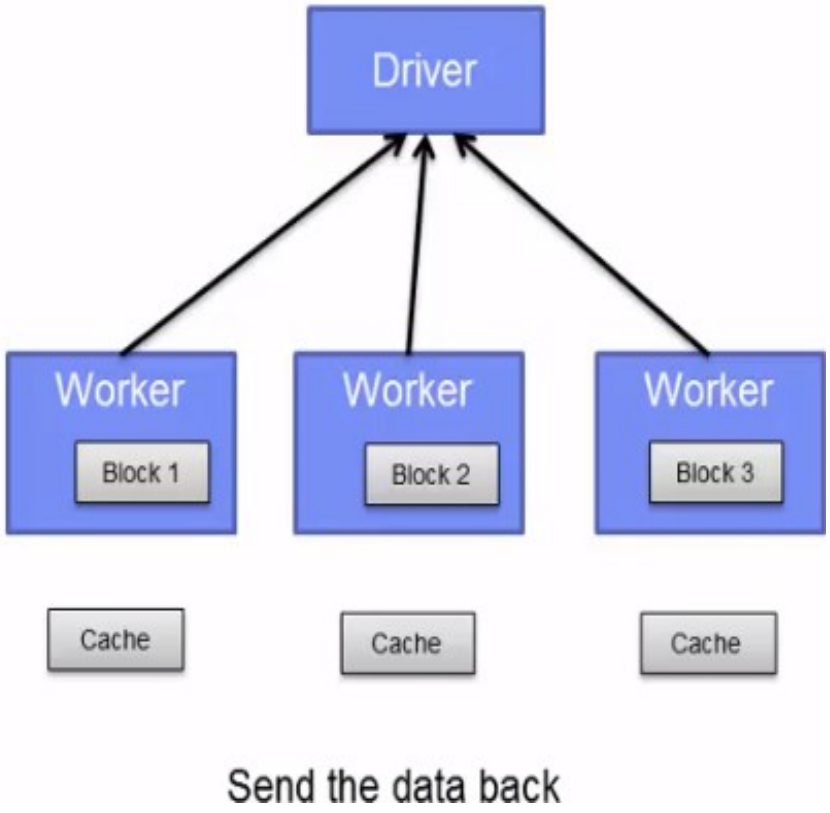
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

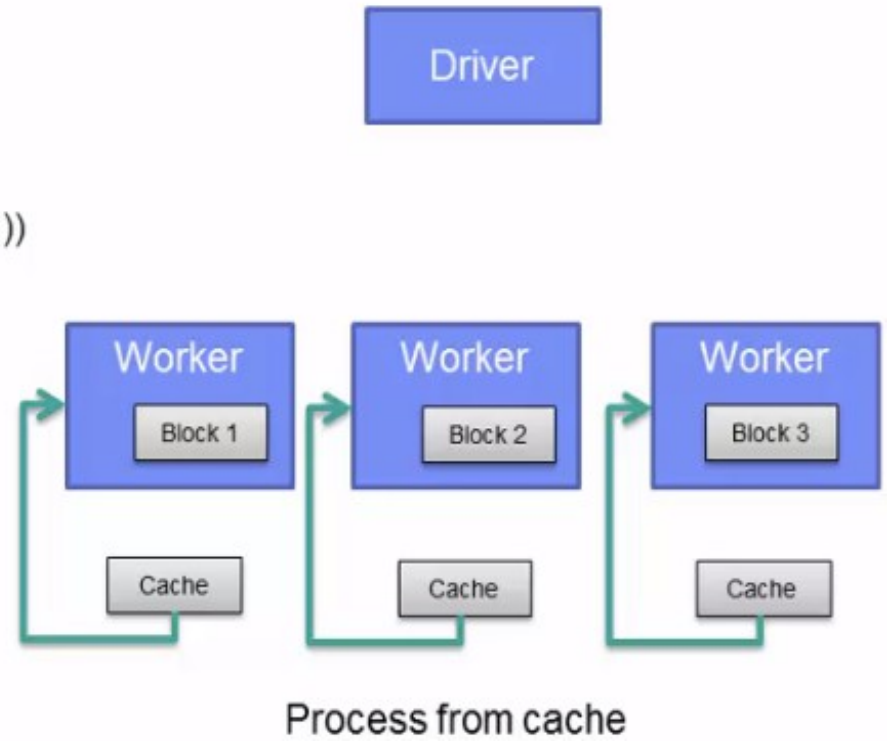
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

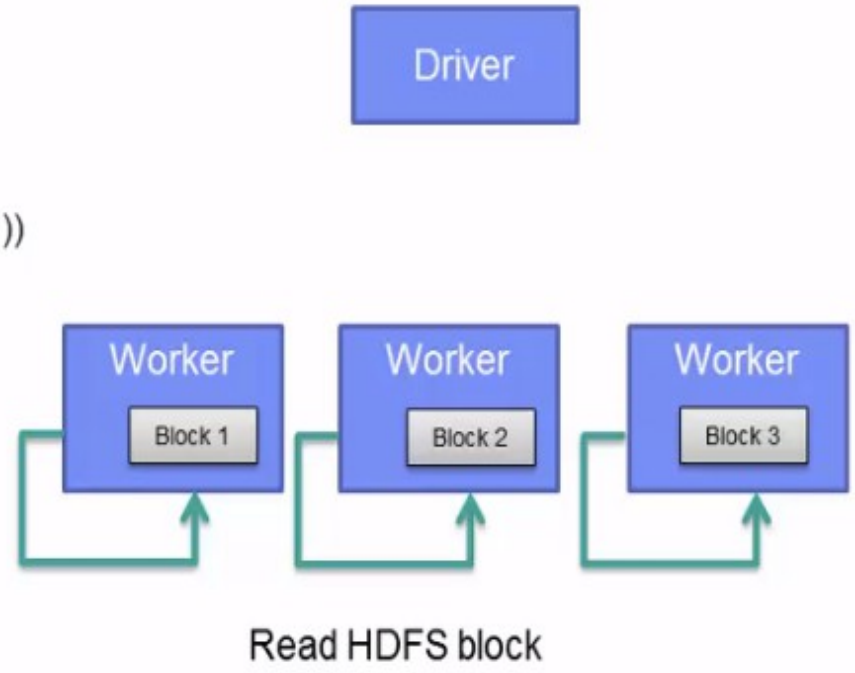
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()
```



Spark:Transformation

transformation	description
map(*func*)	return a new distributed dataset formed by passing each element of the source through a function *func*
filter(*func*)	return a new dataset formed by selecting those elements of the source on which *func* returns true
flatMap(*func*)	similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item)
sample(*withReplacement, fraction, seed*)	sample a fraction *fraction* of the data, with or without replacement, using a given random number generator *seed*
union(*otherDataset*)	return a new dataset that contains the union of the elements in the source dataset and the argument
distinct([*numTasks*]))	return a new dataset that contains the distinct elements of the source dataset

Spark:Transformation

transformation	description
groupByKey([*numTasks*])	when called on a dataset of (K, V) pairs, returns a dataset of (K, Seq[V]) pairs
reduceByKey(*func*, [*numTasks*])	when called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function
sortByKey([*ascending*], [*numTasks*])	when called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument
join(*otherDataset*, [*numTasks*])	when called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key
cogroup(*otherDataset*, [*numTasks*])	when called on datasets of type (K, V) and (K, W), returns a dataset of (K, Seq[V], Seq[W]) tuples – also called groupWith
cartesian(*otherDataset*)	when called on datasets of types T and U, returns a dataset of (T, U) pairs (all pairs of elements)

Single RDD Transformation

filter females to analyze female buying patterns

```
male1, male2, female1 -> female1
```

map squared values

```
2, 5, 6 -> 4, 25, 36
```

flatMap to break up a sentence into words

```
my name is ray -> my, name, is, ray
```

find the distinct values in a dataset

```
apple, apple, banana -> apple, banana
```

sample two values at random

```
apple, banana, guava -> banana, apple
```

Source: Jspark: A brain-friendly introduction

Multiple RDD Transformation

union

```
apple, orange, banana, guava,
banana, pear
```

intersection

```
banana
```

subtract anything shown in Dataset B from Dataset A

```
apple, orange
```

cartesian (every possible pair combo)

```
(apple, guava), (apple, banana), ...
```

Source: Jspark: A brain-friendly introduction

Pair RDD Transformation

- reduceByKey
- groupByKey
- combineByKey
- mapValues
- flatMapValues
- keys
- values
- subtractByKey
- join
- rightOuterJoin
- leftOuterJoin
- cogroup
- sortByKey

Source: Jspark: A brain-friendly introduction

Spark:Actions

action	description
reduce(*func*)	aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one), and should also be commutative and associative so that it can be computed correctly in parallel
collect()	return all the elements of the dataset as an array at the driver program – usually useful after a filter or other operation that returns a sufficiently small subset of the data
count()	return the number of elements in the dataset
first()	return the first element of the dataset – similar to *take*(*1*)
take(*n*)	return an array with the first *n* elements of the dataset – currently not executed in parallel, instead the driver program computes all the elements
takeSample(*withReplacement, fraction, seed*)	return an array with a random sample of *num* elements of the dataset, with or without replacement, using the given random number generator seed

Spark:Actions

action	description
saveAsTextFile(*path*)	write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call `toString` on each element to convert it to a line of text in the file
saveAsSequenceFile(*path*)	write the elements of the dataset as a Hadoop `SequenceFile` in a given path in the local filesystem, HDFS or any other Hadoop-supported file system. Only available on RDDs of key-value pairs that either implement Hadoop's `Writable` interface or are implicitly convertible to `Writable` (Spark includes conversions for basic types like `Int`, `Double`, `String`, etc).
countByKey()	only available on RDDs of type `(K, V)`. Returns a `Map` of `(K, Int)` pairs with the count of each key
foreach(*func*)	run a function *func* on each element of the dataset – usually done for side effects such as updating an accumulator variable or interacting with external storage systems

Spark: Persistence

transformation	*description*
MEMORY_ONLY	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they're needed. This is the default level.
MEMORY_AND_DISK	Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed.
MEMORY_ONLY_SER	Store RDD as serialized Java objects (one byte array per partition). This is generally more space-efficient than deserialized objects, especially when using a fast serializer, but more CPU-intensive to read.
MEMORY_AND_DISK_SER	Similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them on the fly each time they're needed.
DISK_ONLY	Store the RDD partitions only on disk.
MEMORY_ONLY_2, **MEMORY_AND_DISK_2,** etc	Same as the levels above, but replicate each partition on two cluster nodes.

Accumulators

- Similar to a MapReduce “Counter”
- A global variable to track metrics about your Spark program for debugging.
- Reasoning: Excutors do not communicate with each other.
- Sent back to driver

Broadcast Variables

- Similar to a MapReduce “Distributed Cache”
- Sends read-only values to worker nodes.
- Great for lookup tables, dictionaries, etc.

DataFrame

- A distributed collection of rows organied into named columns.
- An abstraction for selecting, filtering, aggregating, and plotting structured data.
- Previously => SchemaRDD

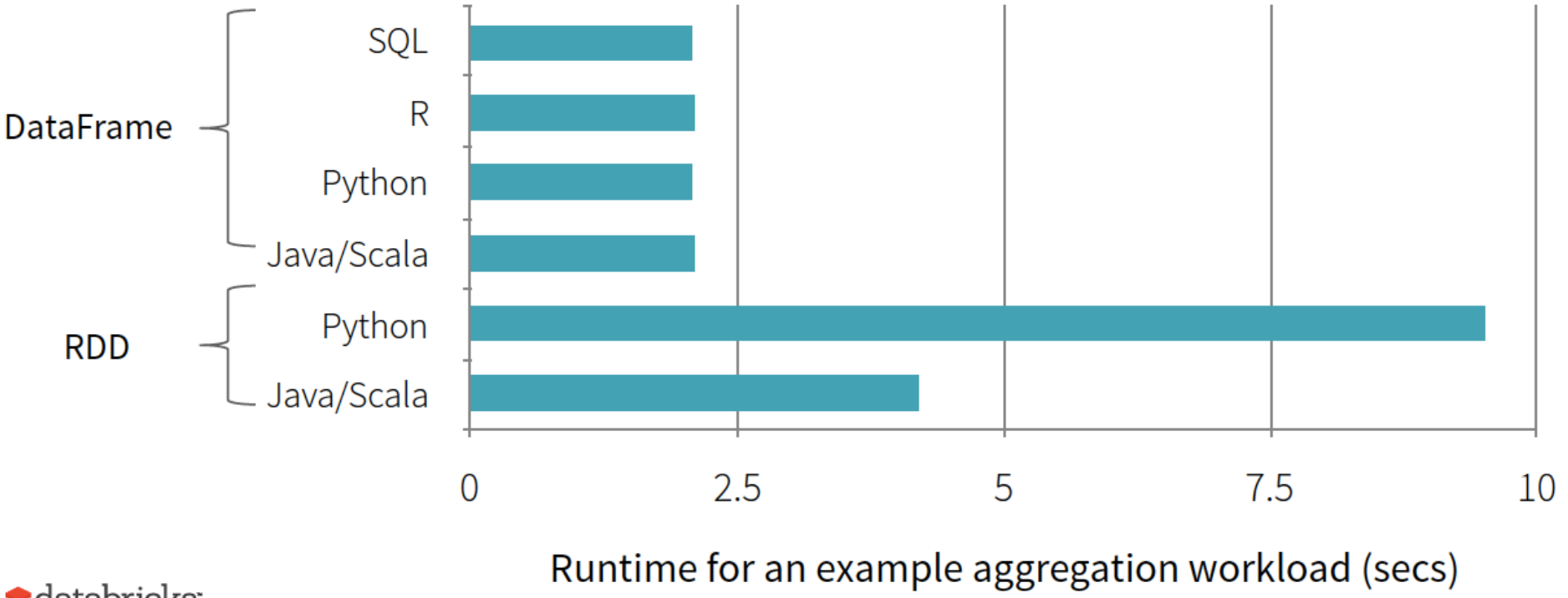
thanachart@imcinstitute.com

SparkSQL

- Creating and running Spark program faster
  - Write less code
  - Read less data
  - Let the optimizer do the hard work

thanachart@imcinstitute.com

Benefit of Logical Plan: Performance Parity Across Languages

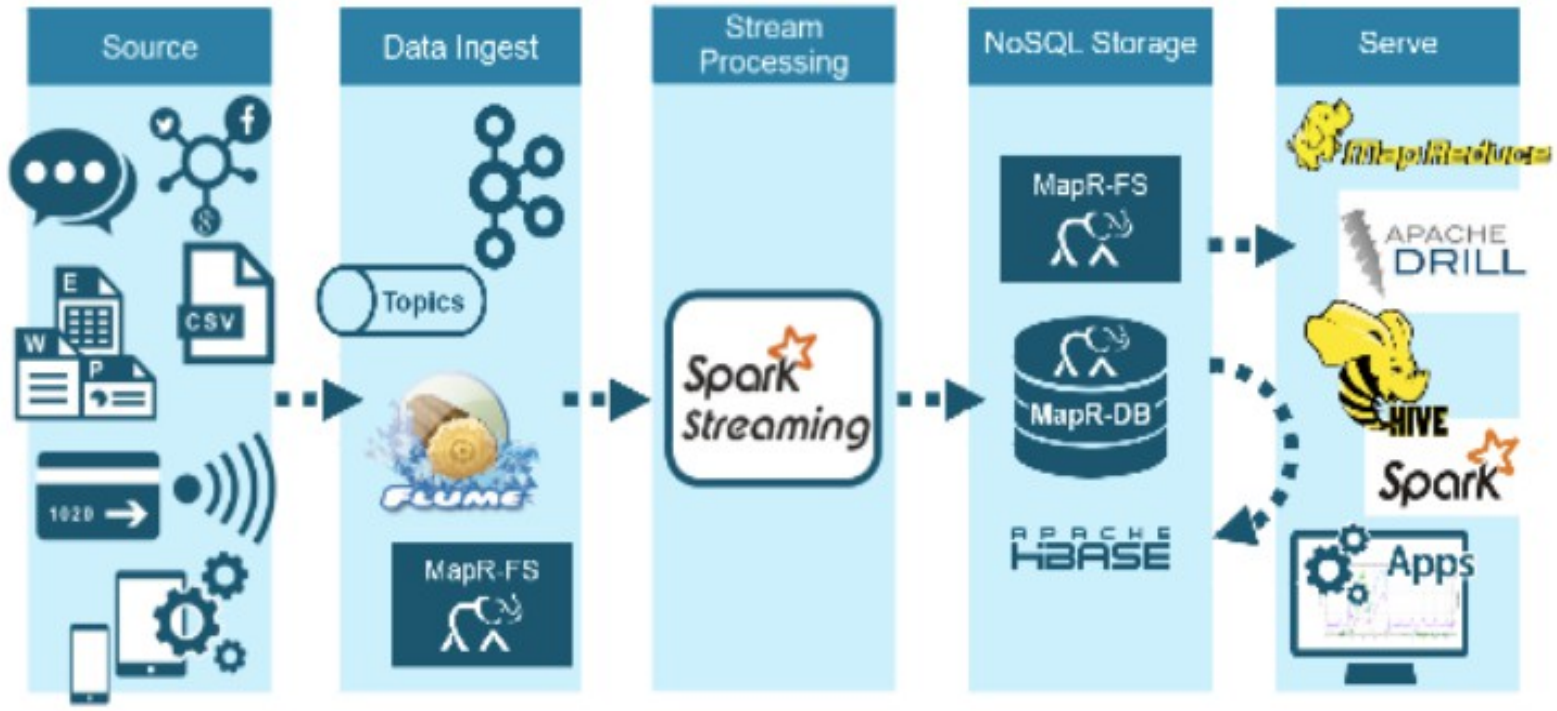


databricks

Source: Jump start into Apache Spark and Databricks

thanachart@imcinstitute.com

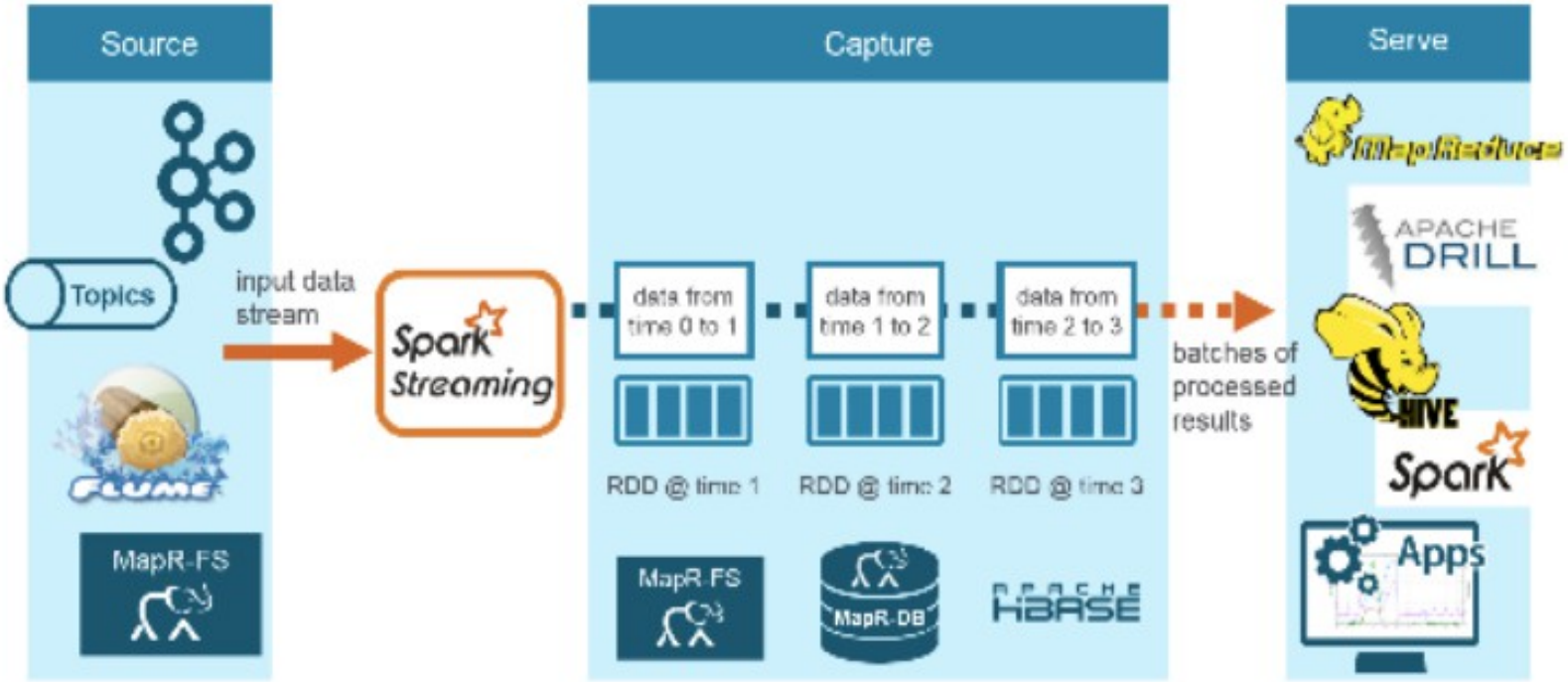
Stream Process Architecture



Source: MapR Academy

thanachart@imcinstitute.com

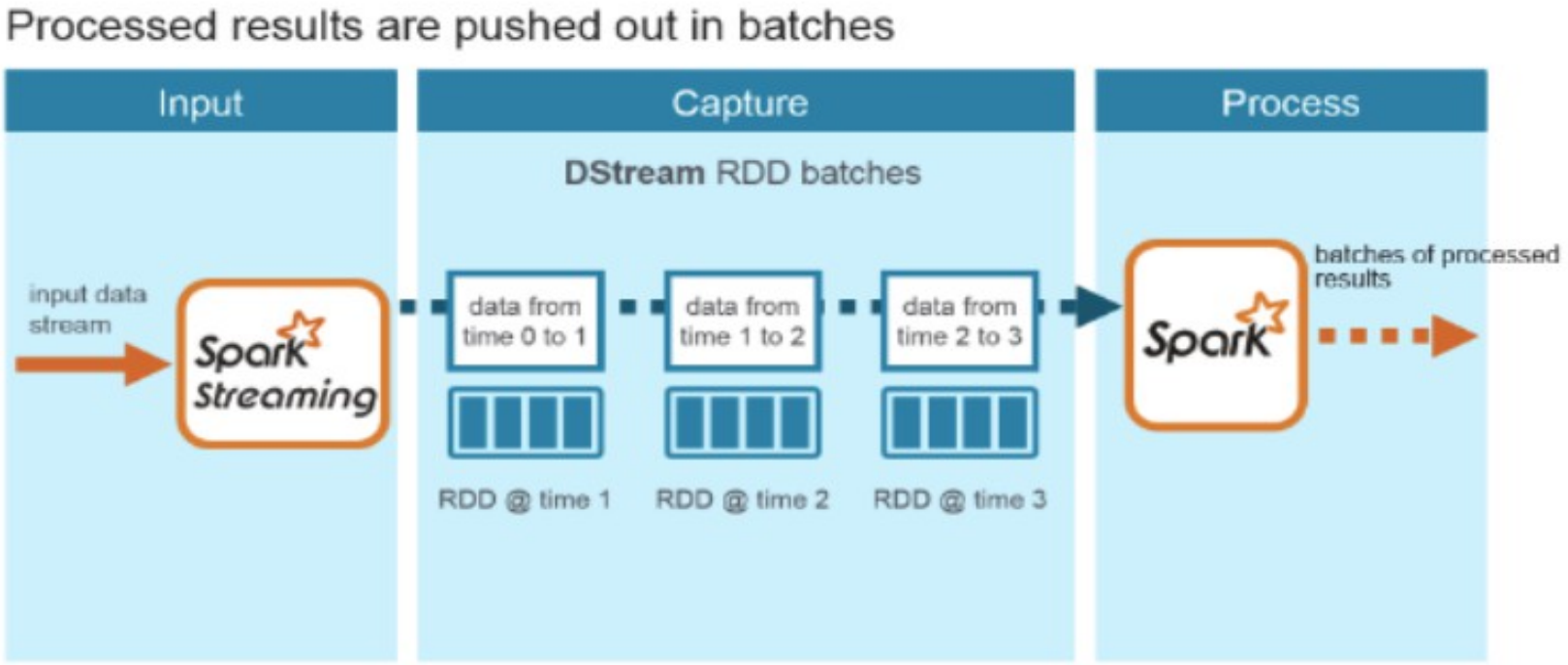
Spark Streaming Architecture



Source: MapR Academy

thanachart@imcinstitute.com

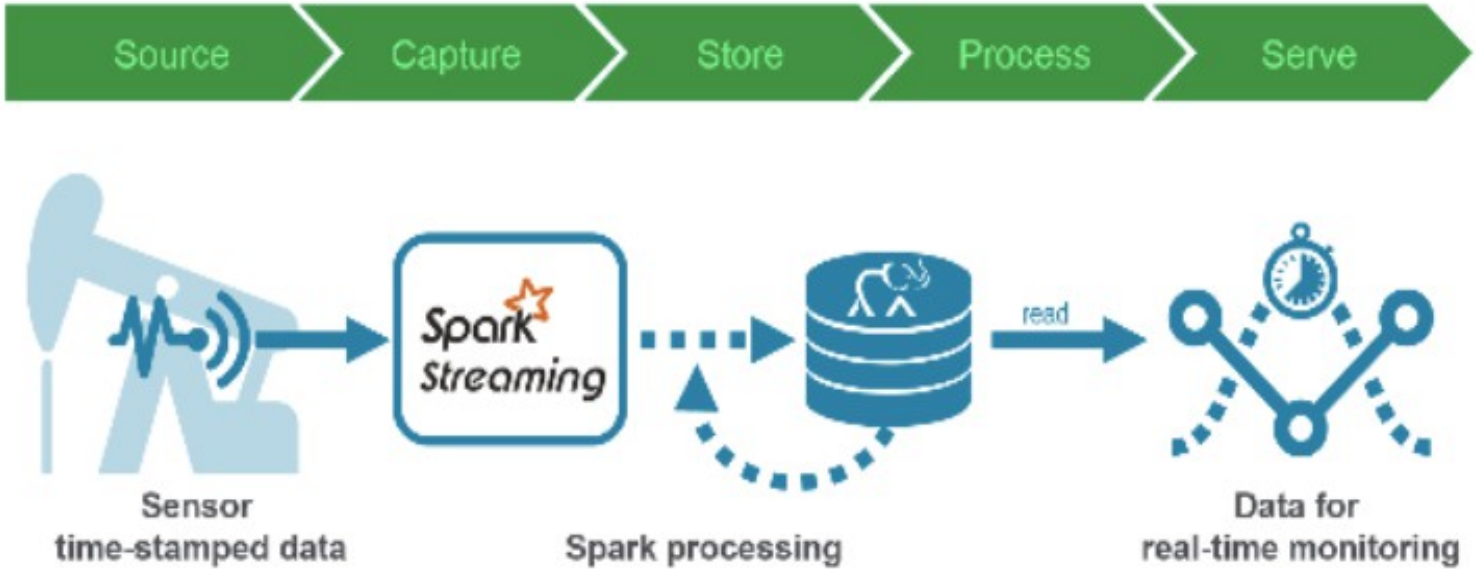
Processing Spark DStreams



Source: MapR Academy

thanachart@imcinstitute.com

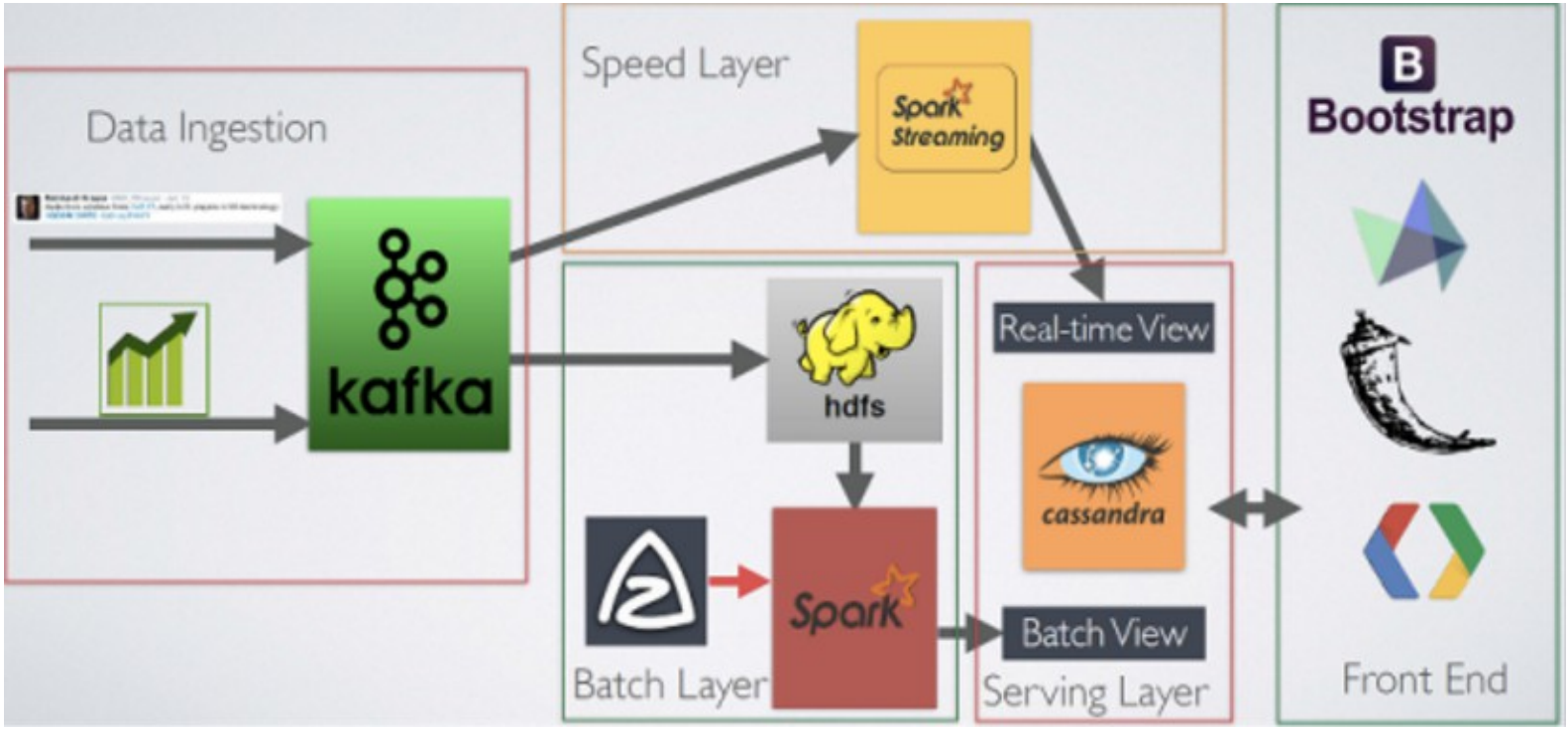
Use Case: Time Series Data



Source: MapR Academy

thanachart@imcinstitute.com

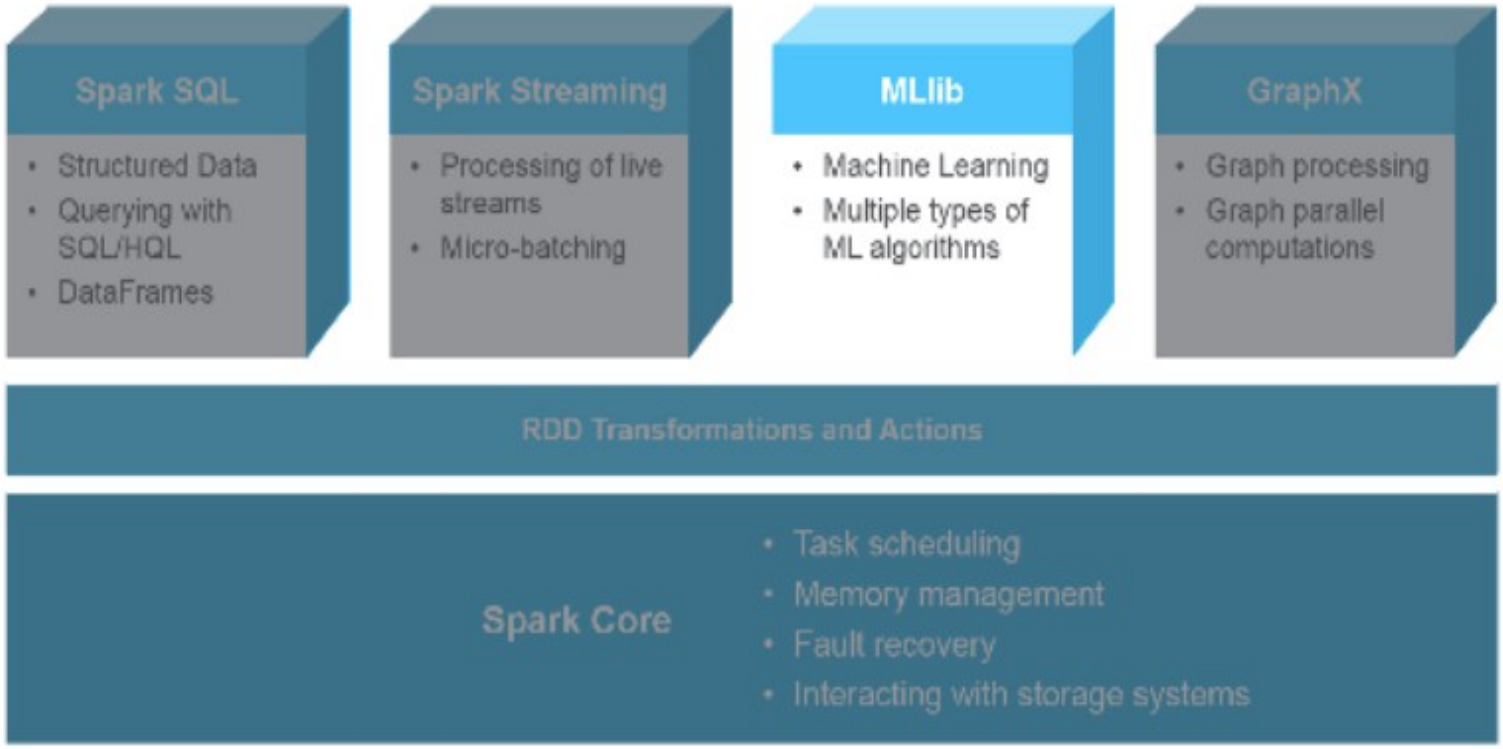
Use Case



Source: http://www.insightdataengineering.com/

thanachart@imcinstitute.com

What is MLlib?



Source: MapR Academy

thanachart@imcinstitute.com

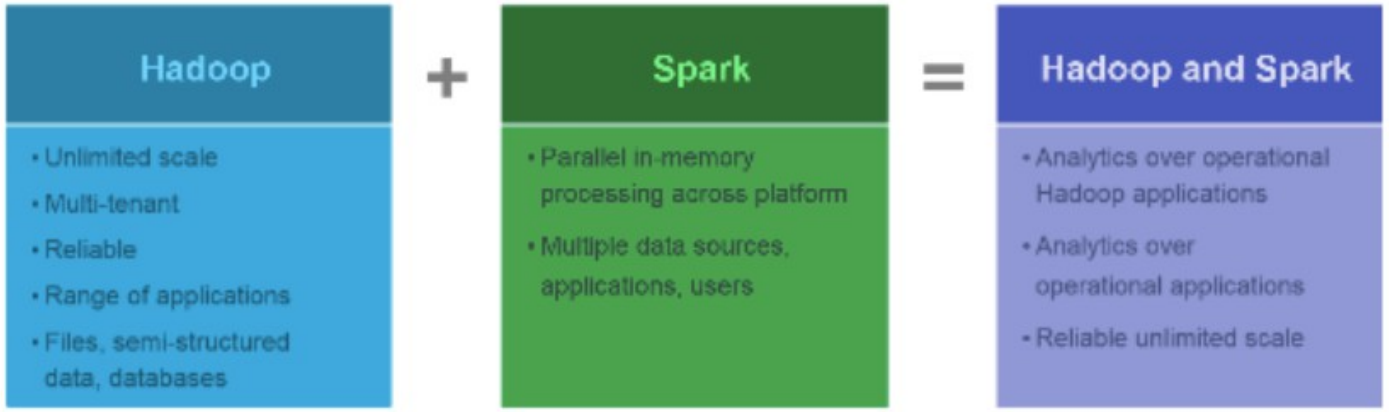
Mllib algorithms and utilities

Algorithms and Utilities	Description
Basic statistics	Includes summary statistics, correlations, hypothesis testing, random data generation
Classification and regression	Includes methods for linear models, decision trees and Naïve Bayes
Collaborative filtering	Supports model-based collaborative filtering using alternating least squares (ALS) algorithm
Clustering	Supports K-means clustering
Dimensionality reduction	Supports dimensionality reduction on the RowMatrix class; singular value decomposition (SVD) and principal component analysis (PCA)
Feature extraction and transformation	Contains several classes for common feature transformations

Source: MapR Academy

thanachart@imcinstitute.com

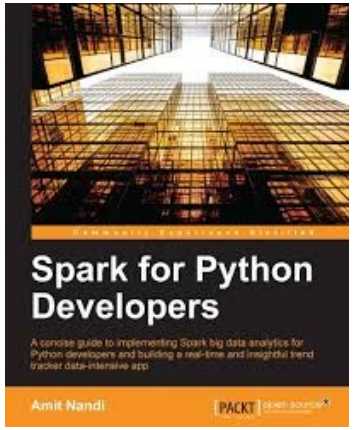
Hadoop + Spark

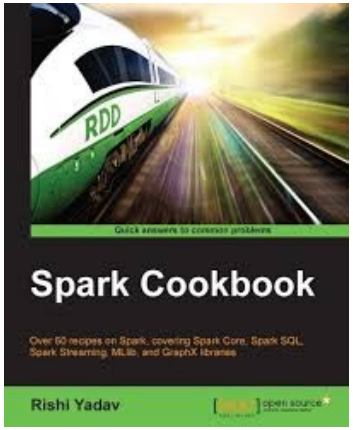


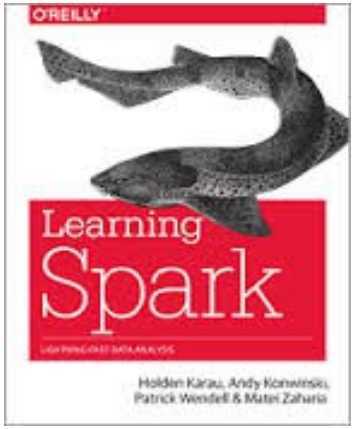
Source: MapR Academy

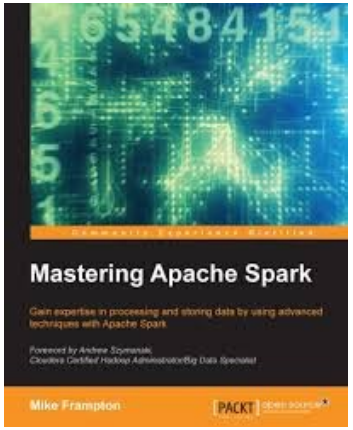
thanachart@imcinstitute.com

Recommended Books









thanachart@imcinstitute.com

Hands-On: Spark Programming

Functional tools in Python

- map
- filter
- reduce
- lambda
- IterTools
    - Chain, flatmap

Map

```
>>> a= [1,2,3]

>>> def add1(x) : return x+1

>>> map(add1, a)

     Result: [2,3,4]
```

Filter

```
>>> a= [1,2,3,4]

>>> def isOdd(x) : return x%2==1

>>> filter(isOdd, a)

    Result: [1,3]
```

Reduce

```
>>> a= [1,2,3,4]

>>> def add(x,y) : return x+y

>>> reduce(add, a)

     Result: 10
```

lambda

```
>>> (lambda x: x + 1)(3)
      Result: 4

>>> map((lambda x: x + 1), [1,2,3])

     Result: [2,3,4]
```

Exercises

- (lambda x: 2*x)(3) => ?
- map(lambda x: 2*x, [1,2,3]) =>
- map(lambda t: t[0], [(1,2), (3,4), (5,6)]) =>
- reduce(lambda x,y: x+y, [1,2,3]) =>
- reduce(lambda x,y: x+y, map(lambda t: t[0], [(1,2), (3,4), (5,6)]))=>

More exercises

- Given
  - a=[(1,2), (3,4), (5,6)]
- Write an expression to get only the second elements of each tuple
- Write an expression to get the sum of the second elements
- Write an expression to get the sum of the odd first elements

Start Spark-shell

```
$spark-shell
```

```
[root@quickstart 201402_babs_open_data]# spark-shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/or
g/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
```

Testing SparkContext

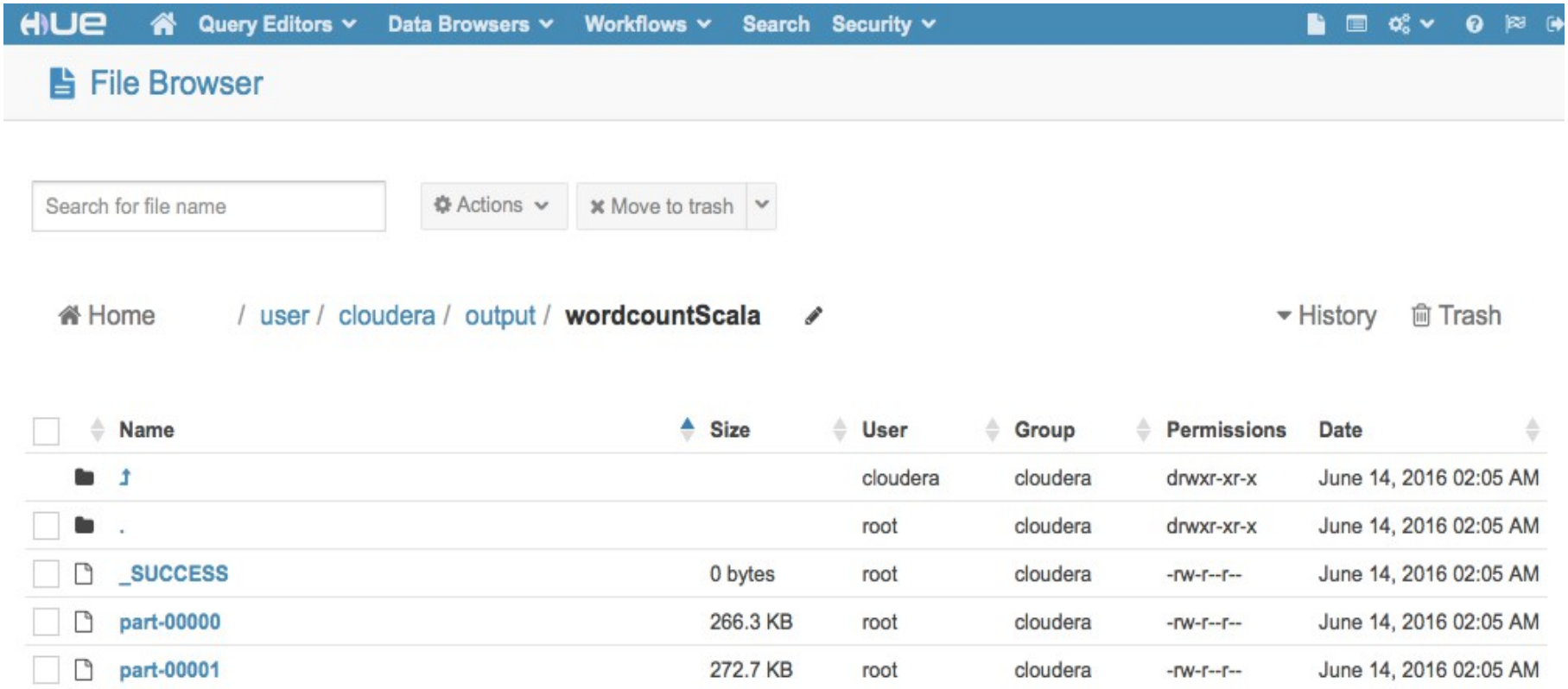
Spark-context

```
scala> sc
```

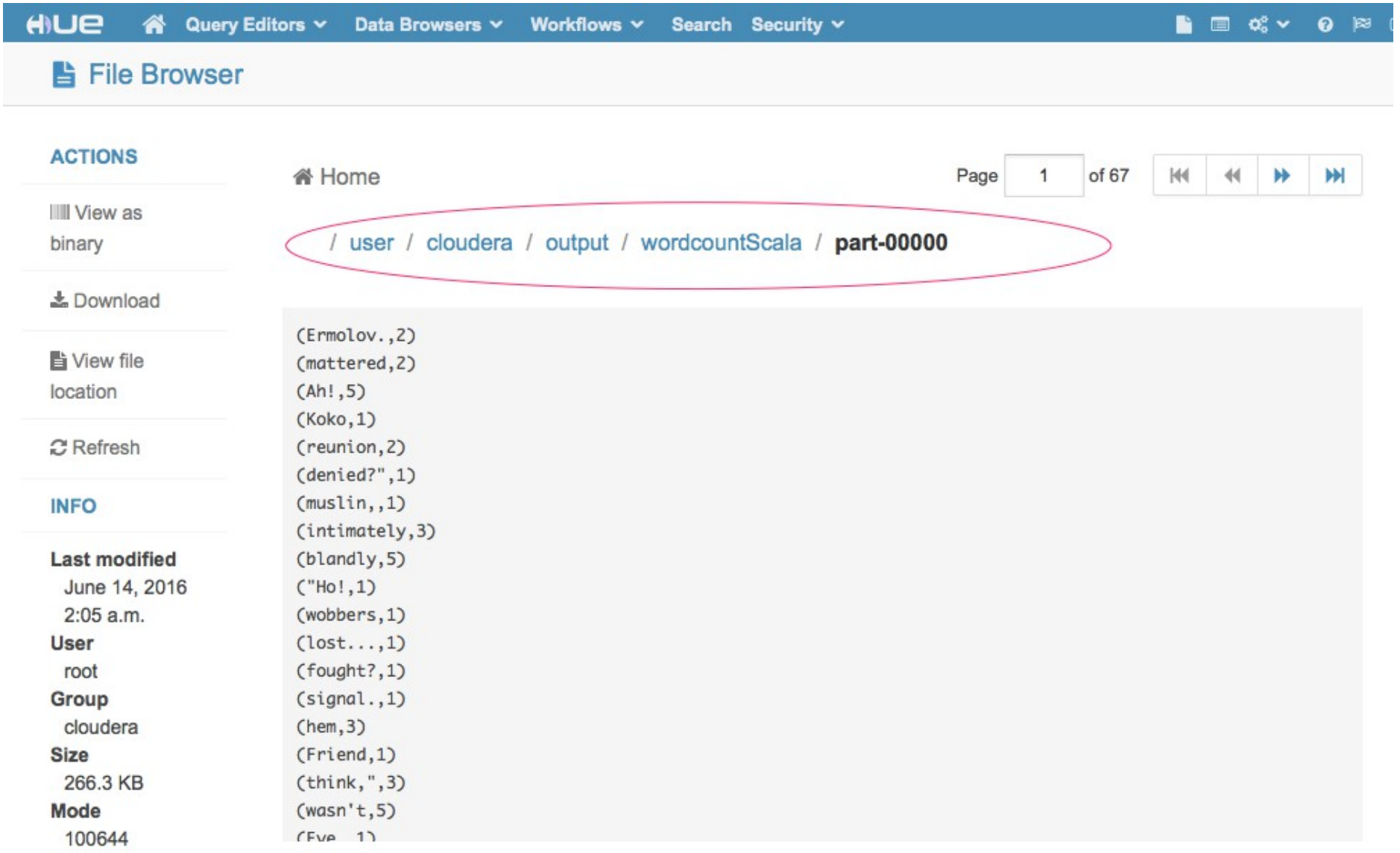
```
scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@18c07e25
```

Spark Program in Scala: WordCount

```
scala> val file =
sc.textFile("hdfs:///user/cloudera/input/pg2600.txt")


scala> val wc = file.flatMap(l => l.split(" ")).map(word =>
(word, 1)).reduceByKey(_ + _)


scala>
wc.saveAsTextFile("hdfs:///user/cloudera/output/wordcountScala
")
```

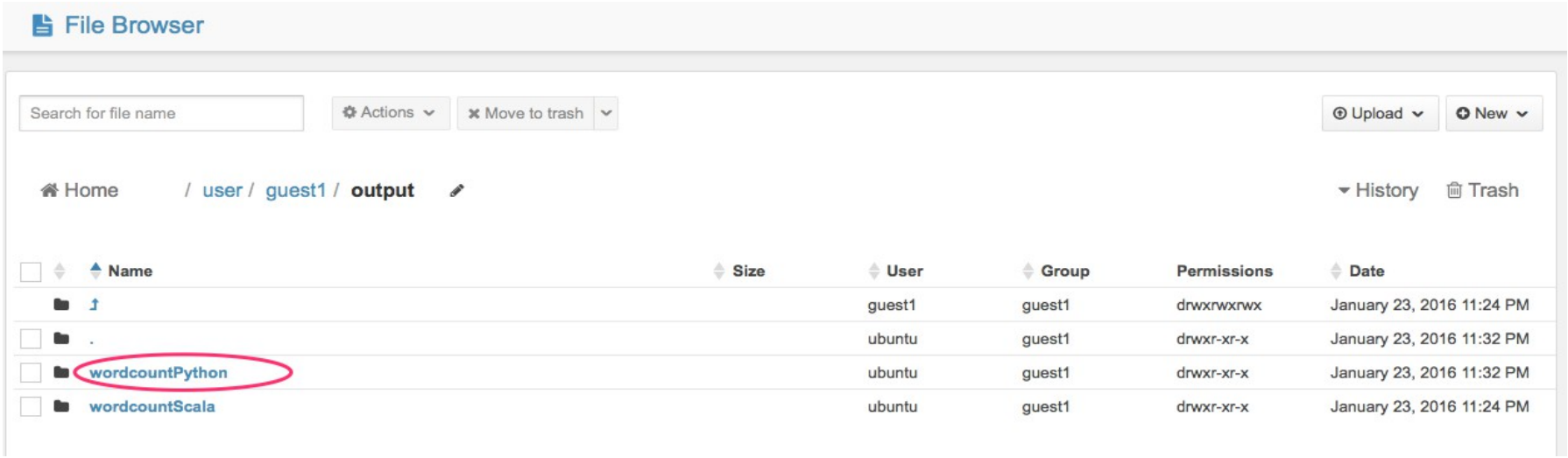

HUE
Query Editors
Data Browsers
Workflows
Search
Security
File Browser
Search for file name
Actions
Move to trash
Home / user / cloudera / output / wordcountScala
History
Trash
Name
Size
User
Group
Permissions
Date
cloudera cloudera drwxr-xr-x June 14, 2016 02:05 AM
. root cloudera drwxr-xr-x June 14, 2016 02:05 AM
_SUCCESS 0 bytes root cloudera -rw-r--r-- June 14, 2016 02:05 AM
part-00000 266.3 KB root cloudera -rw-r--r-- June 14, 2016 02:05 AM
part-00001 272.7 KB root cloudera -rw-r--r-- June 14, 2016 02:05 AM

WordCount output



Spark Program in Python: WordCount

```
$ pyspark
>>> from operator import add
>>> file =
sc.textFile("hdfs:///user/cloudera/input/pg2600.txt")
>>> wc = file.flatMap(lambda x: x.split(' ')).map(lambda x:
(x, 1)).reduceByKey(add)
>>> wc.saveAsTextFile("hdfs:///user/cloudera/output/
wordcountPython")
```



Spark Program in Python: Random

```
>>> import random
>>> flips = 1000000
>>> #lazy eval

>>> coins = xrange(flips)
>>> heads = sc.parallelize(coins).map(lambda i \
random.random()) \
filter(lambda r : r < 0.51).count()
```

Transformations

```
>>> nums = sc.parallelize([1,2,3])
>>> squared = nums.map(lambda x : x*x)
>>> even = squared.filter(lambda x: x%2 == 0)
>>> evens = nums.flatMap(lambda x: range(x))
```

Actions

```
>>> nums = sc.parallelize([1,2,3])
>>> nums.collect()
>>> nums.take(2)
>>> nums.count()
>>> nums.reduce(lambda:x, y:x+y)
>>> nums.saveAsTextFile("hdfs:///user/cloudera/output/test”)
```

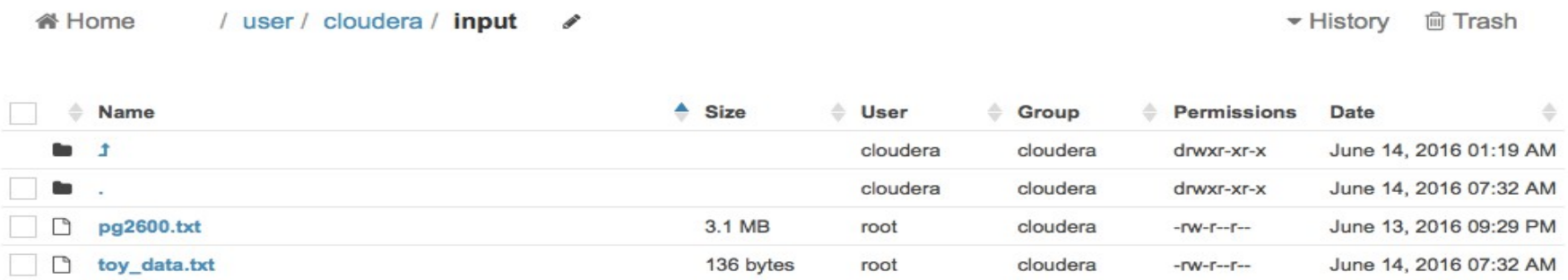
Key-Value Operations

```
>>> pet = sc.parallelize([("cat",1),("dog",1),("cat",2)])
>>> pet2 = pet.reduceByKey(lambda x, y:x+y)
>>> pet3 = pet.groupByKey()
>>> pet4 = pet.sortByKey()
```

Spark Program : Toy_data.txt

Upload a data to HDFS

```
$ wget https://s3.amazonaws.com/imcbucket/data/toy_data.txt
$ hadoop fs -put toy_data.txt /user/cloudera/input
```



Start pyspark

```
$ pyspark
```

Spark Program : Find Big Spenders

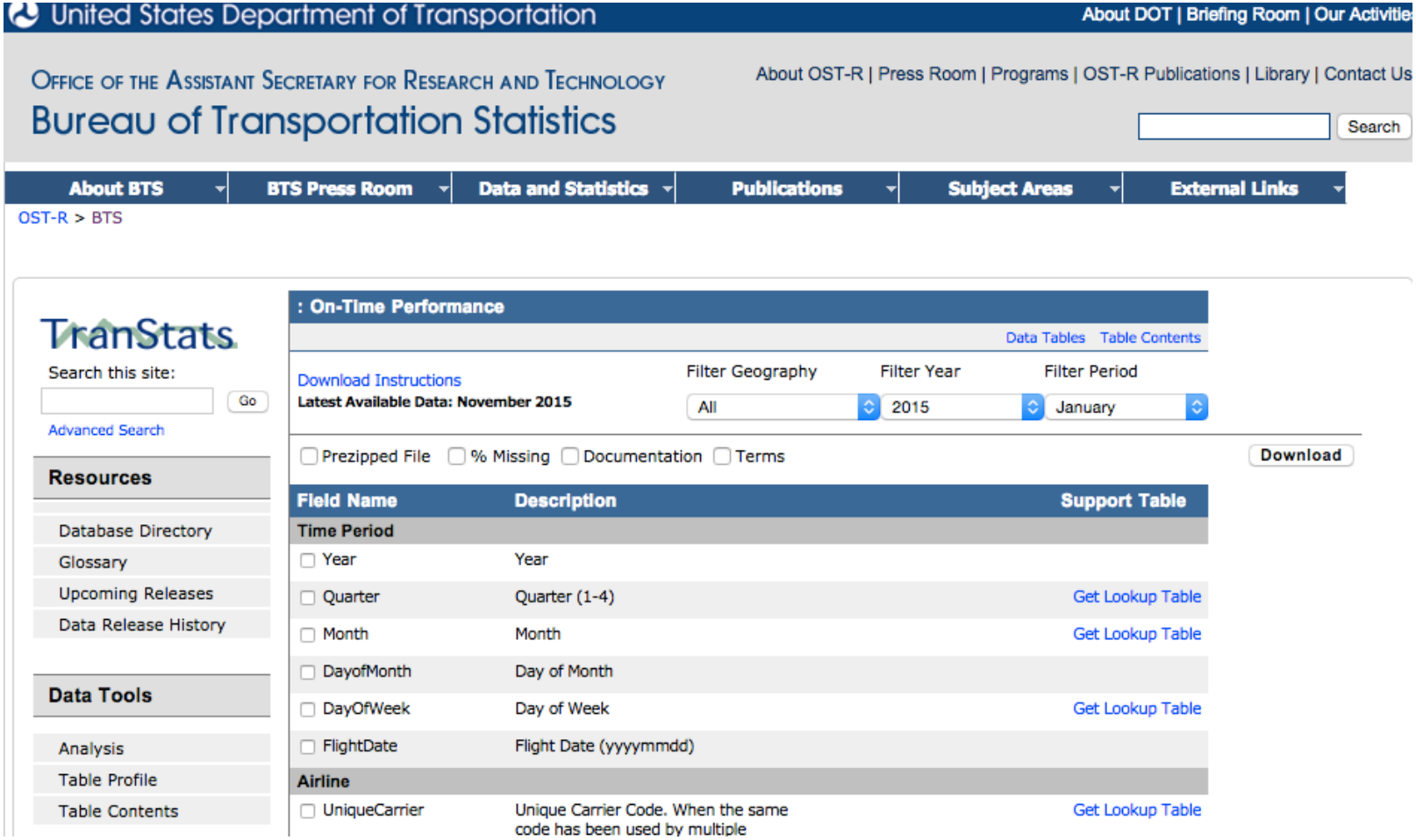
```
>>> file_rdd =
sc.textFile("hdfs:///user/cloudera/input/toy_data.txt")
>>> import json
>>> json_rdd = file_rdd.map(lambda x: json.loads(x))
>>> big_spenders = json_rdd.map(lambda x: tuple((x.keys()
[0],int(x.values()[0]))))
>>> big_spenders.reduceByKey(lambda x,y: x + y).filter(lambda
x: x[1] > 5).collect()
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Project: Flight

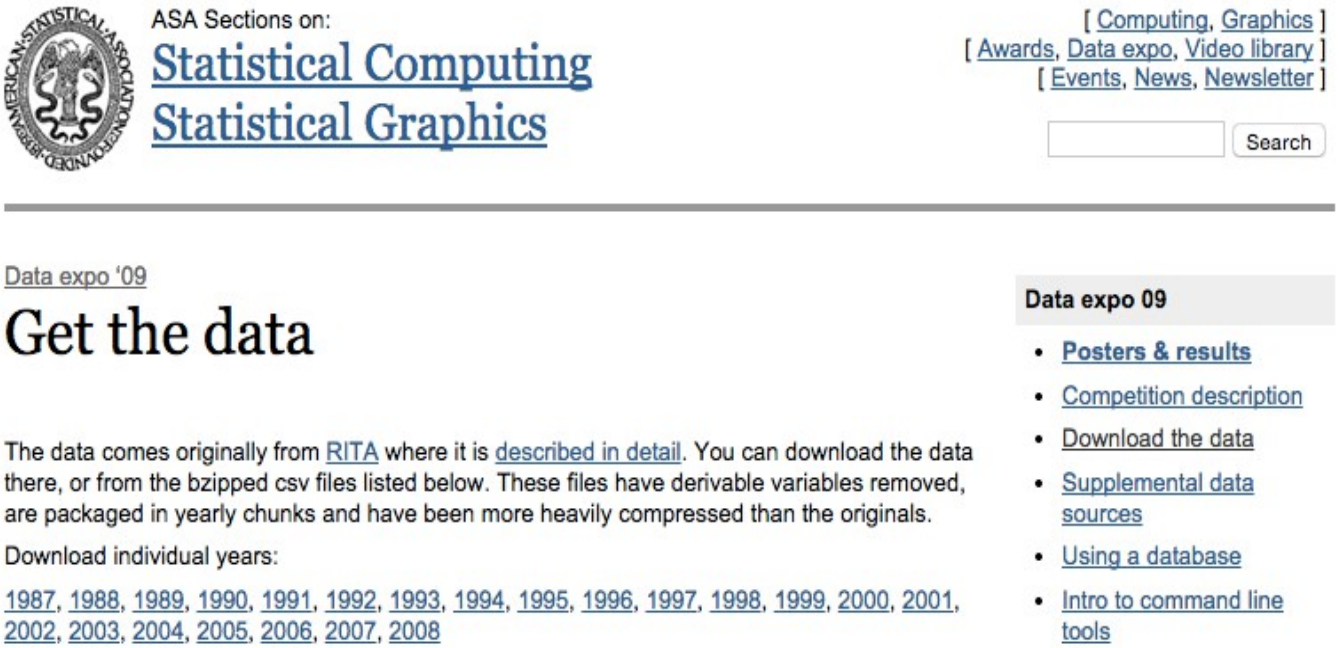
Flight Details Data

http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236



Flight Details Data

http://stat-computing.org/dataexpo/2009/the-data.html



ASA Sections on:
Statistical Computing
Statistical Graphics

[Computing, Graphics]
[Awards, Data expo, Video library]
[Events, News, Newsletter]

Search

Data expo '09

Get the data

The data comes originally from RITA where it is described in detail. You can download the data there, or from the bzipped csv files listed below. These files have derivable variables removed, are packaged in yearly chunks and have been more heavily compressed than the originals.

Download individual years:

1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008

Data expo 09

- **Posters & results**
- Competition description
- Download the data
- Supplemental data sources
- Using a database
- Intro to command line tools

Data Description

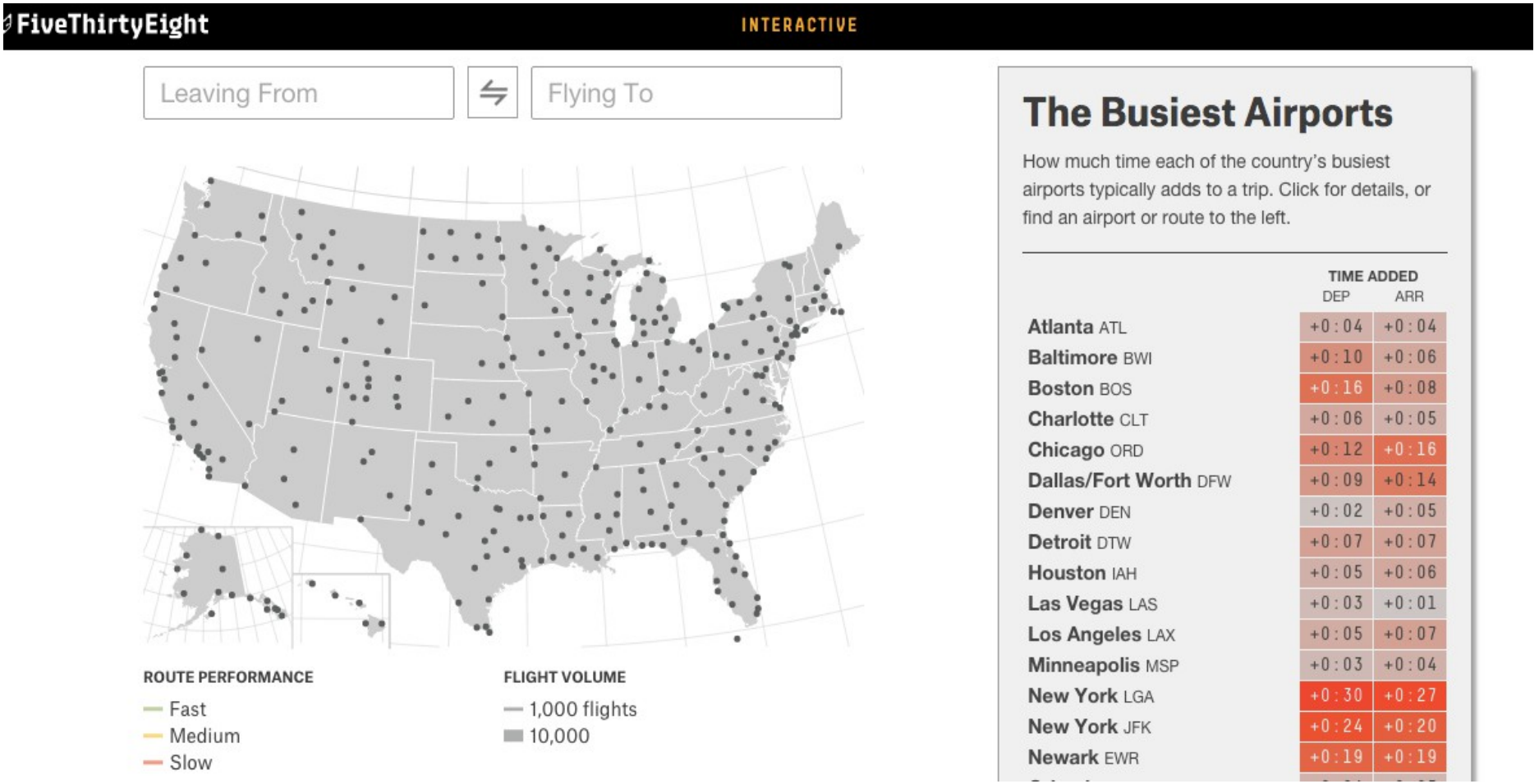
	Name	Description
1	Year	1987-2008
2	Month	1-12
3	DayofMonth	1-31
4	DayOfWeek	1 (Monday) - 7 (Sunday)
5	DepTime	actual departure time (local, hhmm)
6	CRSDepTime	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	UniqueCarrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code
18	Dest	destination IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

Snapshot of Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Year	Month	DayofMor	DayOfWe	DepTime	CRSDepTi	ArrTime	CRSArrTim	UniqueCa	FlightNun	TailNum	ActualEla	CRSElapse	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance	TaxiIn	TaxiO
2	2008	1	5	6	2243	1415	45	1625	WN	1684	N347SW	62	70	41	500	508	SAN	PHX	304	2	
3	2008	1	5	6	1940	1220	2111	1350	WN	1684	N347SW	91	90	64	441	440	SFO	SAN	447	5	
4	2008	1	7	1	111	1845	308	2045	WN	405	N644SW	117	120	103	383	386	MDW	JAN	666	4	
5	2008	1	7	1	2213	1700	2317	1655	WN	1827	N759GS	124	55	75	382	313	IND	MDW	162	10	
6	2008	1	7	1	2143	1720	26	1820	WN	1430	N644SW	163	60	83	366	263	STL	MDW	251	24	
7	2008	1	7	1	117	2020	302	2135	WN	490	N651SW	105	75	87	327	297	STL	TUL	351	5	
8	2008	1	7	1	2358	1855	105	2000	WN	490	N651SW	67	65	50	305	303	MDW	STL	251	4	
9	2008	1	3	4	2245	1730	2354	1850	WN	186	N792SW	69	80	59	304	315	JAN	HOU	359	3	
10	2008	1	7	1	2219	1730	35	1935	WN	2474	N710SW	76	65	67	300	289	MDW	CMH	284	2	
11	2008	1	5	6	2129	1620	2246	1750	WN	1924	N408WN	77	90	56	296	309	SFO	LAS	414	4	
12	2008	1	3	4	1615	1130	1623	1135	WN	10	N617SW	68	65	56	288	285	MAF	ABQ	332	4	
13	2008	1	3	4	1736	1305	2031	1555	WN	1837	N761RR	295	290	268	276	271	MDW	SFO	1855	4	
14	2008	1	5	6	2236	1805	2400	1930	WN	646	N283WN	84	85	71	270	271	LAX	SFO	337	6	
15	2008	1	3	4	2021	1700	2303	1835	WN	2005	N302SW	162	95	73	268	201	LAS	SFO	414	4	
16	2008	1	3	4	2059	1620	2216	1750	WN	1924	N761RR	77	90	60	266	279	SFO	LAS	414	6	
17	2008	1	7	1	2348	2105	307	2250	WN	3137	N358SW	259	165	244	257	163	MCO	MDW	989	1	
18	2008	1	3	4	2255	1820	509	55	WN	1924	N761RR	194	215	176	254	275	LAS	IND	1591	9	
19	2008	1	9	3	1458	1040	1725	1315	WN	2556	N501SW	87	95	76	250	258	BNA	BWI	588	4	
20	2008	1	7	1	2300	1835	113	2105	WN	2804	N420WN	253	270	240	248	265	MDW	PDX	1751	5	
21	2008	1	5	6	47	2040	151	2145	WN	505	N435WN	64	65	51	246	247	BWI	PVD	328	5	
22	2008	1	5	6	1558	1225	14	2010	WN	505	N442WN	316	285	250	244	213	SAN	BWI	2295	5	
23	2008	1	5	6	1931	1540	2104	1705	WN	1179	N718SW	93	85	77	239	231	SAN	OAK	446	7	
24	2008	1	4	5	1822	1425	2003	1605	WN	753	N726SW	101	100	88	238	237	PDX	OAK	543	6	

FiveThirtyEight

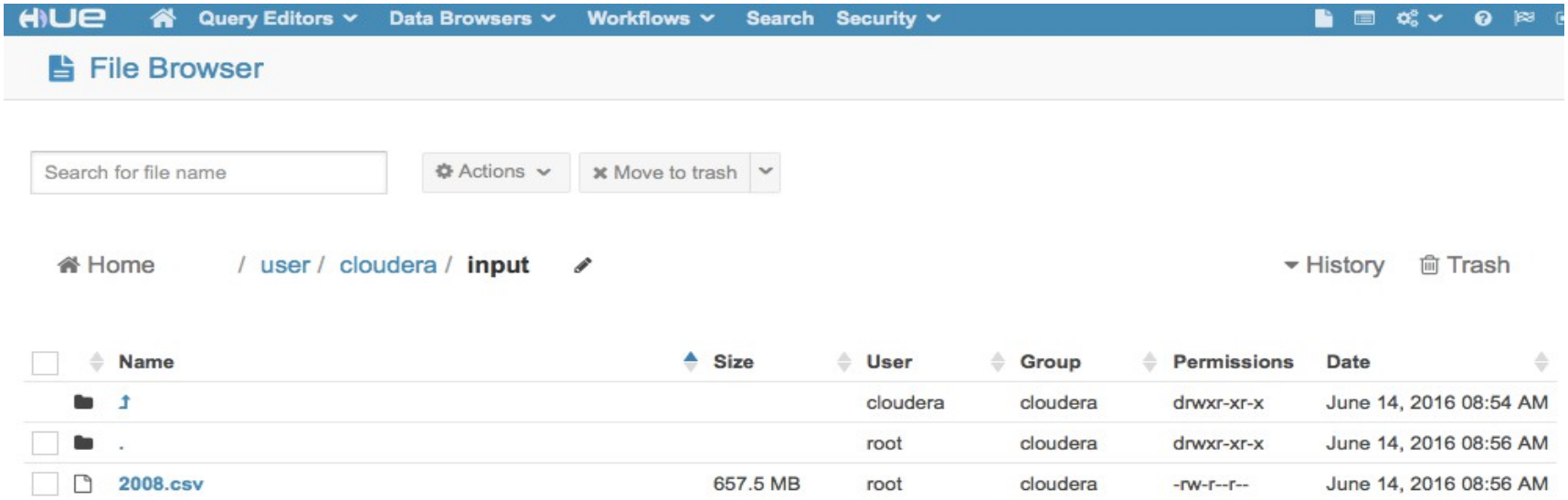
http://projects.fivethirtyeight.com/flights/



Spark Program : Upload Flight Delay Data

Upload a data to HDFS

```
$ wget
https://s3.amazonaws.com/imcbucket/data/flights/2008.csv
$ hadoop fs -put 2008.csv /user/cloudera/input
```



Spark Program : Navigating Flight Delay Data

```
>>> airline =
sc.textFile("hdfs:///user/cloudera/input/2008.csv")
>>> airline.take(2)
```

```
[u'Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueC
arrier,FlightNum,TailNum,ActualElapsedTime,CRSElapsedTime,AirTime,ArrDelay,DepDe
lay,Origin,Dest,Distance,TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,Carr
ierDelay,WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay', u'2008,1,3,4,20
03,1955,2211,2225,WN,335,N712SW,128,150,116,-14,8,IAD,TPA,810,4,8,0,,0,NA,NA,NA,
NA,NA']
```

Spark Program : Preparing Data

```
>>> header_line = airline.first()
>>> header_list = header_line.split(',')
>>> airline_no_header = airline.filter(lambda row: row != header_line)
>>> airline_no_header.first()
>>> def make_row(row):
...     row_list = row.split(',')
...     d = dict(zip(header_list,row_list))
...     return d
...
>>> airline_rows = airline_no_header.map(make_row)
>>> airline_rows.take(5)
```

Spark Program : Define convert_float function

```
>>> def convert_float(value):
...     try:
...             x = float(value)
...             return x
...     except ValueError:
...             return 0
...
>>>
```

Spark Program : Finding best/worst airports

```
>>> destination_rdd = airline_rows.map(lambda row:
(row['Dest'],convert_float(row['ArrDelay'])))
>>> origin_rdd = airline_rows.map(lambda row:
(row['Origin'],convert_float(row['DepDelay'])))
>>> destination_rdd.take(2)
>>> origin_rdd.take(2)
```

```
16/01/24 14:46:15 INFO DAGScheduler: Job 28 finished: runJob at PythonRDD.scala:
361, took 5.763248 s
[(u'TPA', -14.0), (u'TPA', 2.0)]
```

```
16/01/24 15:43:22 INFO DAGScheduler: Job 7 finished: runJob at PythonRDD.scala:3
61, took 0.110790 s
[(u'IAD', 8.0), (u'IAD', 19.0)]
```

Spark Program : Finding best/worst airports

```
>>> import numpy as np
>>> mean_delays_dest =
destination_rdd.groupByKey().mapValues(lambda delays:
np.mean(delays.data))
>>> mean_delays_dest.sortBy(lambda t:t[1],
ascending=True).take(10)
>>> mean_delays_dest.sortBy(lambda t:t[1],
ascending=False).take(10)
```

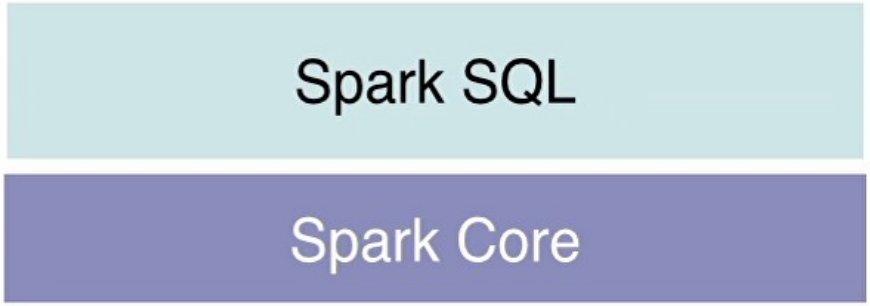
```
16/01/24 15:49:29 INFO DAGScheduler: Job 12 finished: runJob at PythonRDD.scala:
361, took 0.477610 s
[(u'BJI', -7.0810810810810807), (u'TUP', -6.2222222222222223), (u'BLI', -5.43801
65289256199), (u'WYS', -5.0189393939393936), (u'LWS', -2.5858895705521472), (u'B
TM', -1.9021276595744681), (u'IYK', -1.8469601677148848), (u'ITH', -1.6240601503
759398), (u'CPR', -1.0957840236686391), (u'CLD', -0.8474576271186403)]

16/01/24 15:51:14 INFO DAGScheduler: Job 18 finished: runJob at PythonRDD.scala:
361, took 0.419637 s
[(u'MQT', 28.901309164149044), (u'OTH', 25.704061895551256), (u'EWR', 20.0910817
30942804), (u'ACK', 19.771855010660982), (u'CIC', 19.433189655172413), (u'SPI',
18.954434499593166), (u'EYW', 17.974231912784937), (u'CMI', 15.841877256317689),
 (u'LMT', 15.692156862745097), (u'HHH', 15.355263157894736)]
```

Hands-On: Spark SQL

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

SparkSQL

```
context = ps.HiveContext(sc)

# query with SQL
results = context.sql(
  "SELECT * FROM people")

# apply Python transformation
names = results.map(lambda p: p.name)
```



Spark Program : Upload Data

Upload a data to HDFS

```
$ wget https://s3.amazonaws.com/imcbucket/data/events.txt
$ wget https://s3.amazonaws.com/imcbucket/data/meals.txt
$ hadoop fs -put events.txt /user/cloudera/input
$ hadoop fs -put meals.txt /user/cloudera/input
```

Spark SQL : Preparing data

```
>>> meals_rdd =
sc.textFile("hdfs:///user/cloudera/input/meals.txt")
>>> events_rdd =
sc.textFile("hdfs:///user/cloudera/input/events.txt")
>>> header_meals = meals_rdd.first()
>>> header_events = events_rdd.first()
>>> meals_no_header = meals_rdd.filter(lambda row:row !=
header_meals)
>>> events_no_header =events_rdd.filter(lambda row:row !=
header_events)
>>> meals_json = meals_no_header.map(lambda
row:row.split(';')).map(lambda row_list:
dict(zip(header_meals.split(';'), row_list)))
>>> events_json = events_no_header.map(lambda
row:row.split(';')).map(lambda row_list:
dict(zip(header_events.split(';'), row_list)))
```

Spark SQL : Preparing data

```
>>> import json
>>> def type_conversion(d, columns) :
...     for c in columns:
...             d[c] = int(d[c])
...     return d
...
>>> meal_typed = meals_json.map(lambda
j:json.dumps(type_conversion(j, ['meal_id','price'])))
'{"type": "french", "dt": "2013-01-01", "meal_id": 1, "price": 10}'

>>> event_typed = events_json.map(lambda
j:json.dumps(type_conversion(j, ['meal_id','userid'])))
'{"meal_id": 18, "dt": "2013-01-01", "userid": 3, "event": "bought"}'
```

Spark SQL : Create DataFrame

```
>>> meals_dataframe = sqlContext.jsonRDD(meal_typed)
>>> events_dataframe = sqlContext.jsonRDD(event_typed)
>>> meals_dataframe.head()

Row(dt=u'2013-07-10', meal_id=1018, price=13, type=u'italian')

>>> meals_dataframe.printSchema()

root
 |-- dt: string (nullable = true)
 |-- meal_id: long (nullable = true)
 |-- price: long (nullable = true)
 |-- type: string (nullable = true)
```

Spark SQL : Running SQL Query

```
>>> meals_dataframe.registerTempTable('meals')
>>> events_dataframe.registerTempTable('events')
>>> sqlContext.sql("SELECT * FROM meals LIMIT 5").collect()
```

```
[Row(dt=u'2013-07-10', meal_id=1018, price=13, type=u'italian'), Row(dt=u'2013-0
7-11', meal_id=1019, price=11, type=u'japanese'), Row(dt=u'2013-07-11', meal_id=
1020, price=14, type=u'italian'), Row(dt=u'2013-07-11', meal_id=1021, price=14,
type=u'italian'), Row(dt=u'2013-07-11', meal_id=1022, price=13, type=u'mexican')
]
```

```
>>> meals_dataframe.take(5)
```

Spark SQL : More complex query

```
>>> sqlContext.sql("""
...     SELECT type, COUNT(type) AS cnt FROM
...             meals
...     INNER JOIN
...             events on meals.meal_id = events.meal_id
...     WHERE
...             event = 'bought'
...     GROUP BY
...             type
...     ORDER BY cnt DESC
... """).collect()
```

```
[Row(type=u'italian', cnt=22575), Row(type=u'french', cnt=16179), Row(type=u'me
xican', cnt=8792), Row(type=u'japanese', cnt=6921), Row(type=u'chinese', cnt=62
67), Row(type=u'vietnamese', cnt=3535)]
```

Hands-On: WordCount using Spark Streaming

Start Spark-shell with extra memory

```
[root@quickstart ~]# spark-shell --driver-memory 1G
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/or
g/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/jars/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/
```

WordCount using Spark Streaming

```
$ scala> :paste
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.storage.StorageLevel
import StorageLevel._
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
val ssc = new StreamingContext(sc, Seconds(2))
val lines = ssc.socketTextStream("localhost",8585,MEMORY_ONLY)
val wordsFlatMap = lines.flatMap(_.split(" "))
val wordsMap = wordsFlatMap.map( w => (w,1))
val wordCount = wordsMap.reduceByKey( (a,b) => (a+b))
wordCount.print
ssc.start
```

Running the netcat server on another window

```
ubuntu@ip-172-31-30-238:~$ sudo su
root@ip-172-31-30-238:/home/ubuntu# docker ps
CONTAINER ID        IMAGE                              COMMAND                  CREATED
        STATUS              PORTS                      NAMES
581c45e85683        cloudera/quickstart:latest   "/usr/bin/docker-qui   About an hou
r ago   Up About an hour    0.0.0.0:8888->8888/tcp   backstabbing_lumiere
root@ip-172-31-30-238:/home/ubuntu# docker exec -i -t 581c45e85683 nc -lk 8585

test this
It is another test on Spark streaming. It is great
```

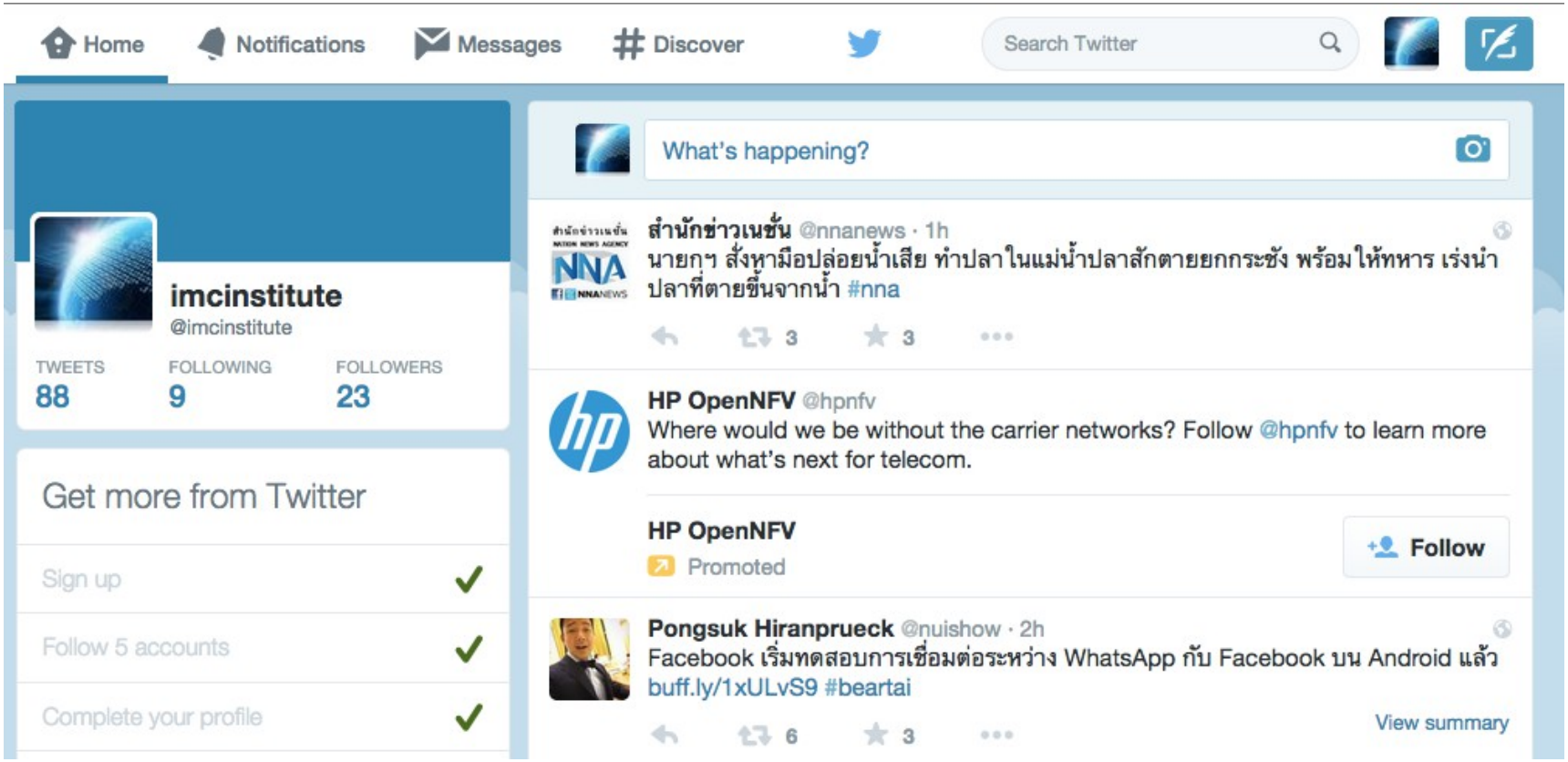
```
-------------------------------------------
Time: 1465924608000 ms
-------------------------------------------
(streaming.,1)
(Spark,1)
(great,1)
(is,2)
(test,1)
(another,1)
(on,1)
(It,2)
```

Hands-On: Streaming Twitter data

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

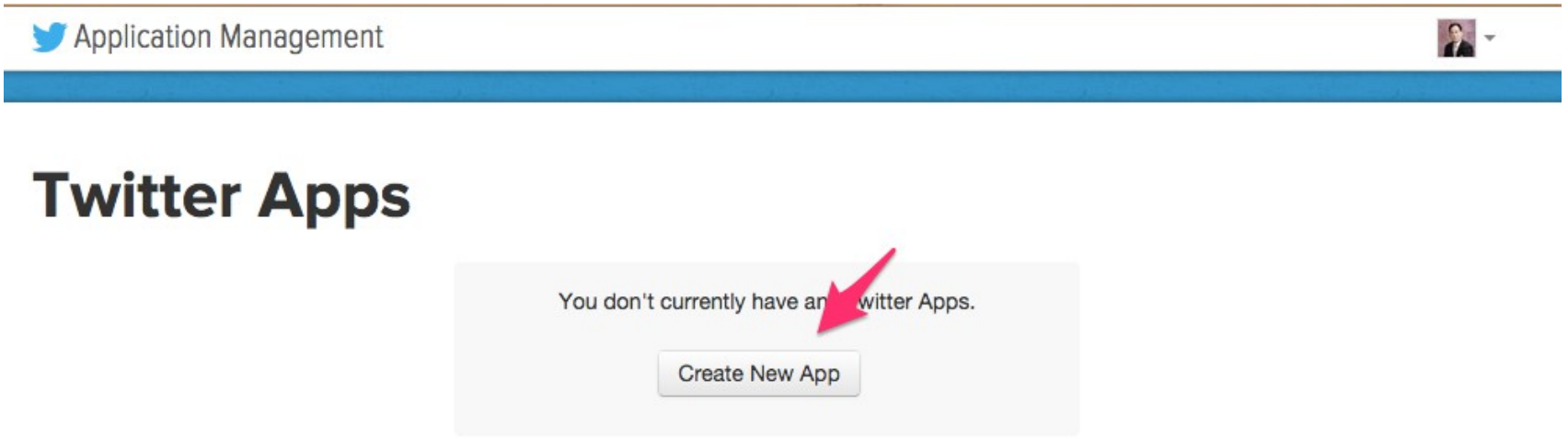
Create a new Twitter App

Login to your Twitter @ twitter.com



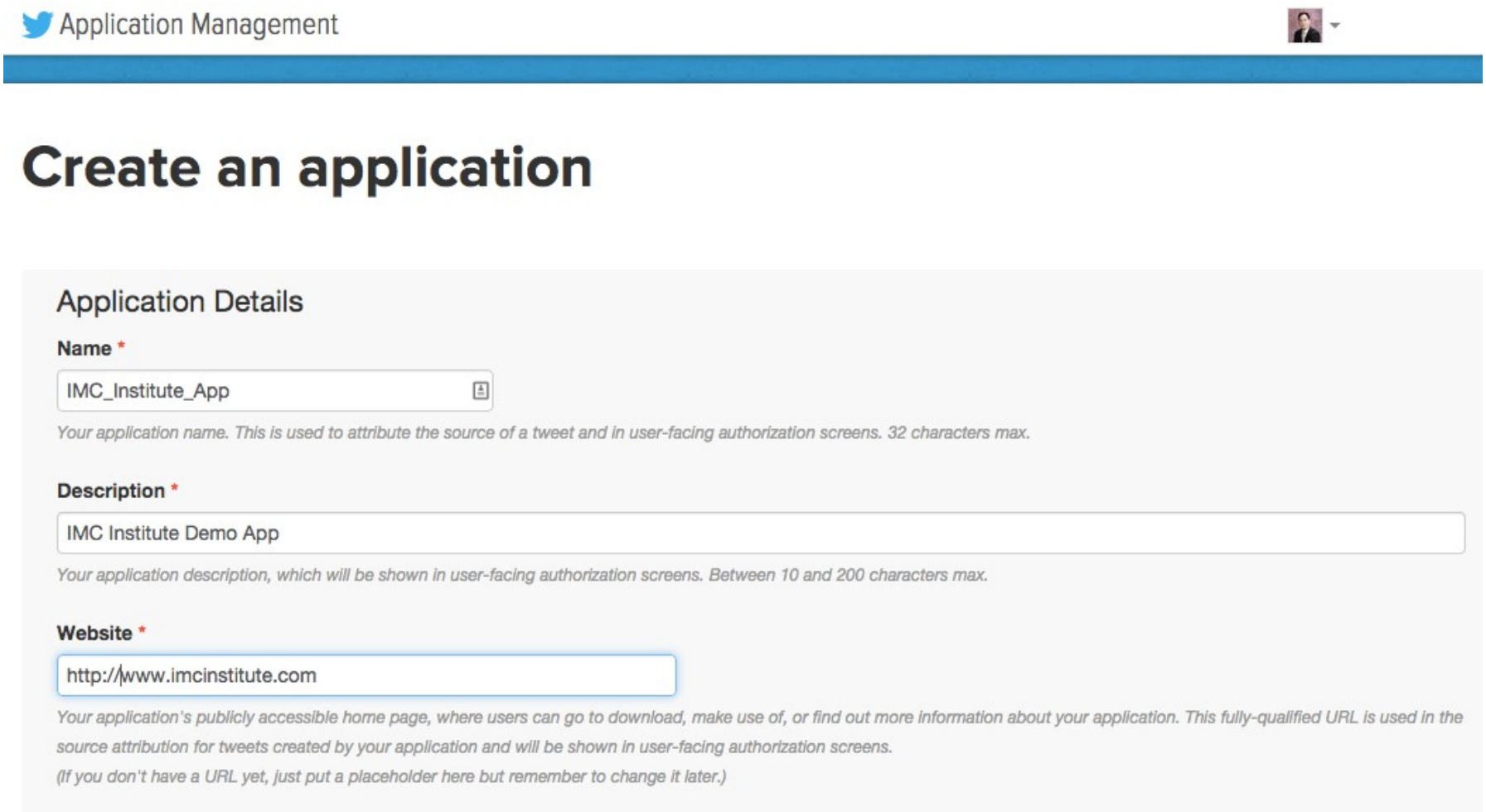
Create a new Twitter App (cont.)

Create a new Twitter App @ apps.twitter.com



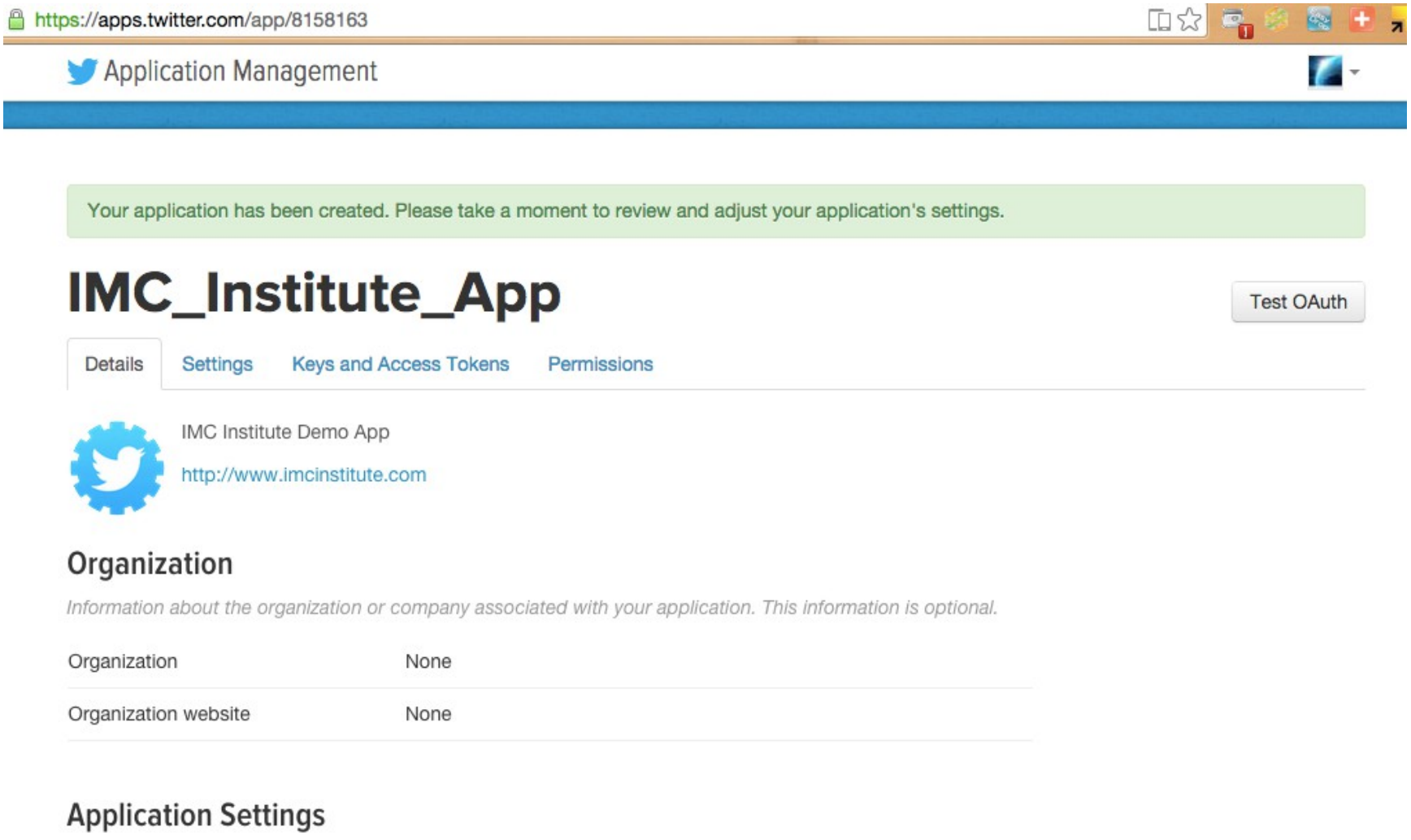
Create a new Twitter App (cont.)

Enter all the details in the application:



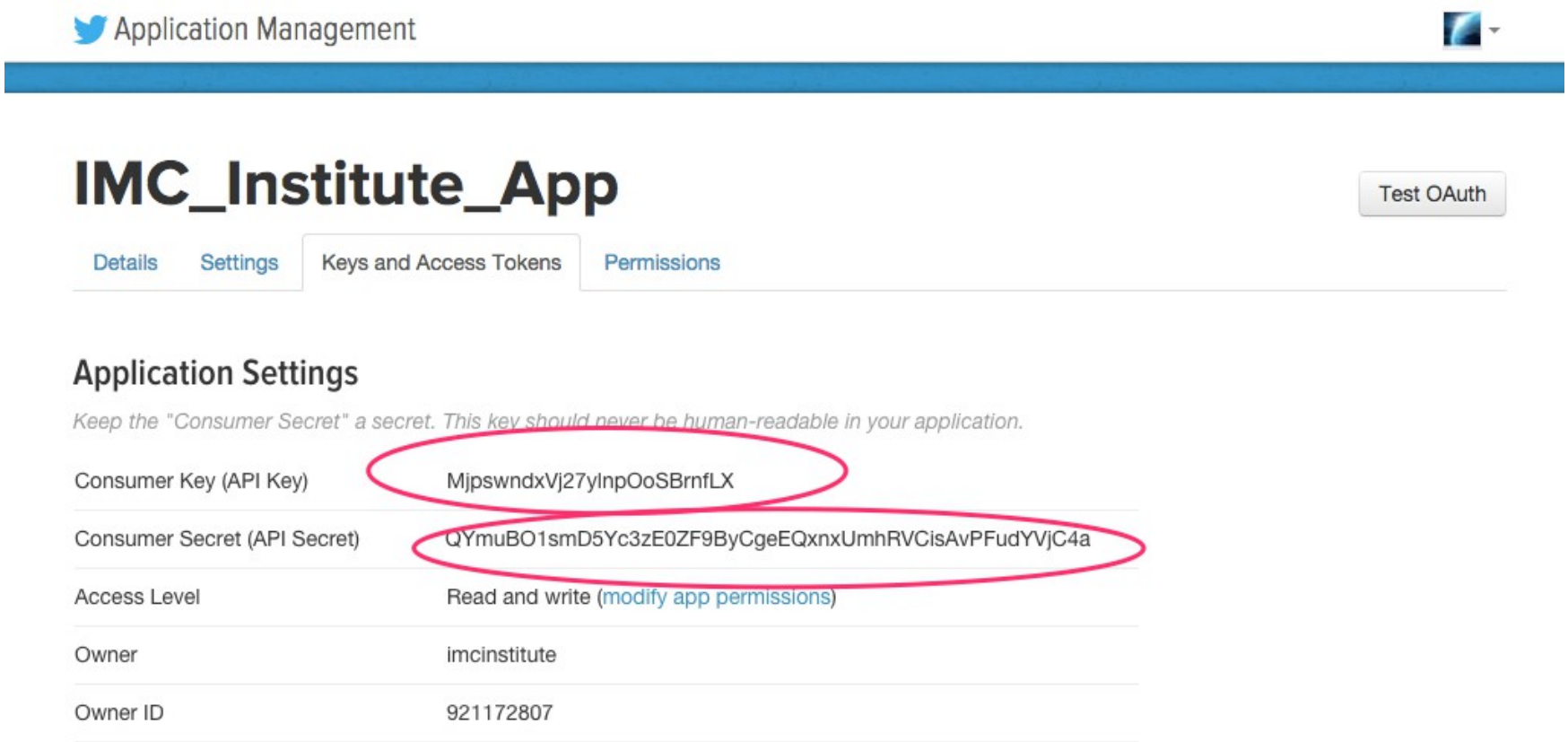
Create a new Twitter App (cont.)

Your application will be created:



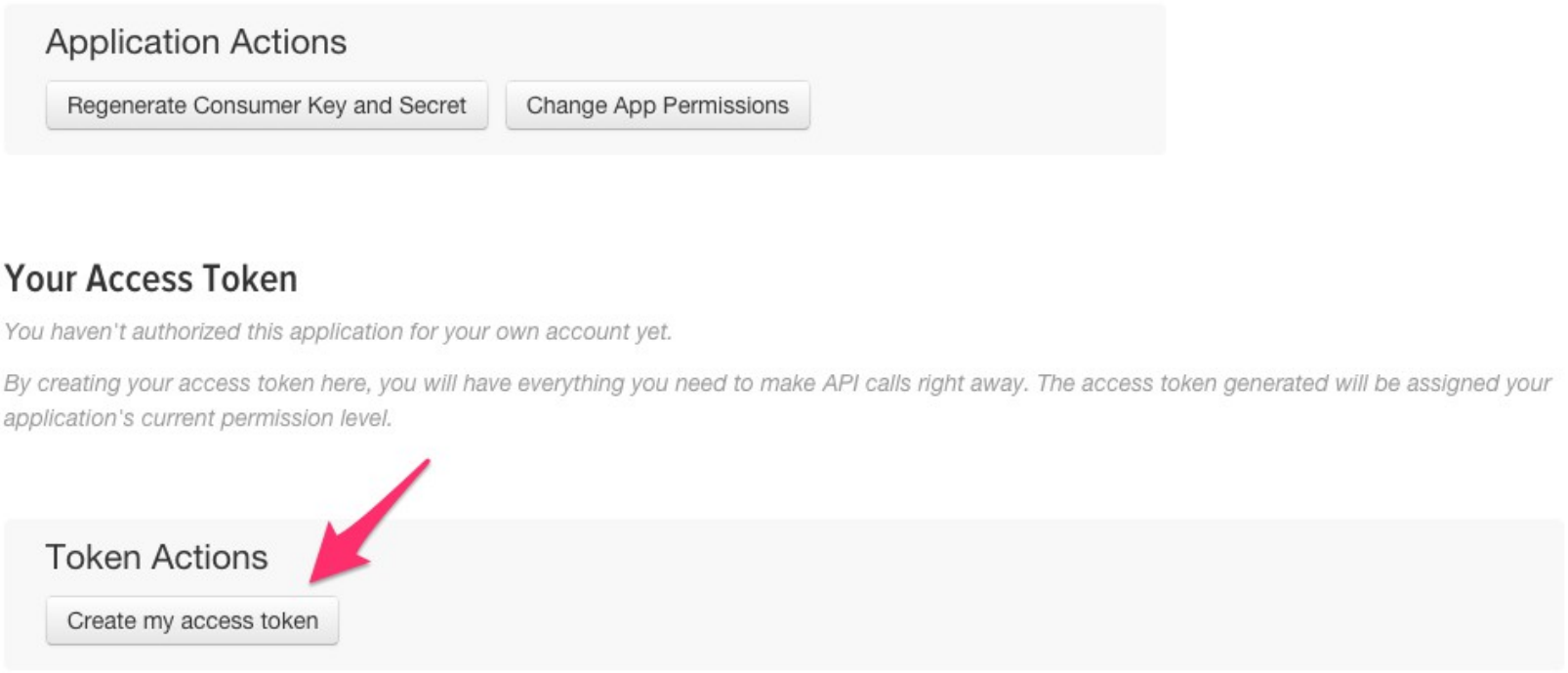
Create a new Twitter App (cont.)

Click on Keys and Access Tokens:



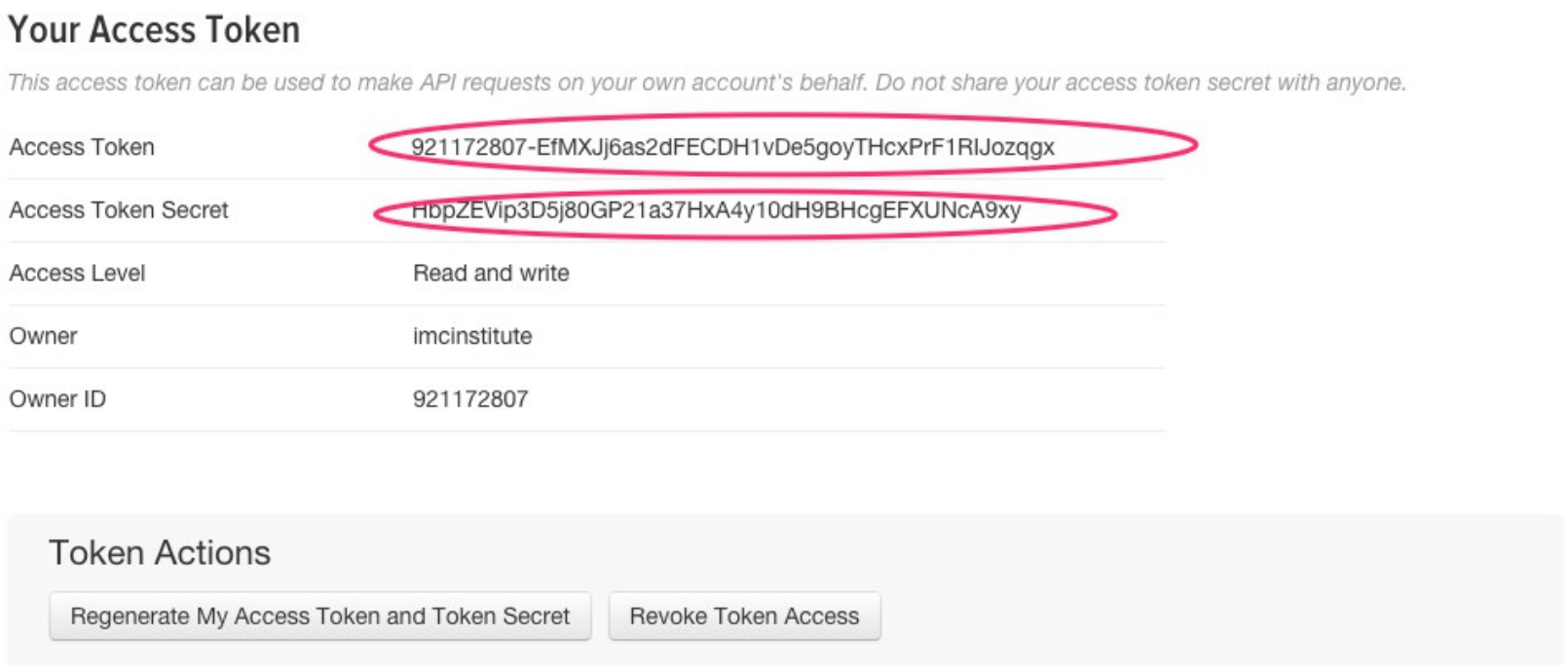
Create a new Twitter App (cont.)

Click on Keys and Access Tokens:



Create a new Twitter App (cont.)

Your Access token got created:



Download the third-party libraries

```
$ wget http://central.maven.org/maven2/org/apache/spark/spark-streaming-twitter_2.10/1.2.0/spark-streaming-twitter_2.10-1.2.0.jar

$ wget http://central.maven.org/maven2/org/twitter4j/twitter4j-stream/4.0.2/twitter4j-stream-4.0.2.jar

$ wget http://central.maven.org/maven2/org/twitter4j/twitter4j-core/4.0.2/twitter4j-core-4.0.2.jar
```

Run Spark-shell

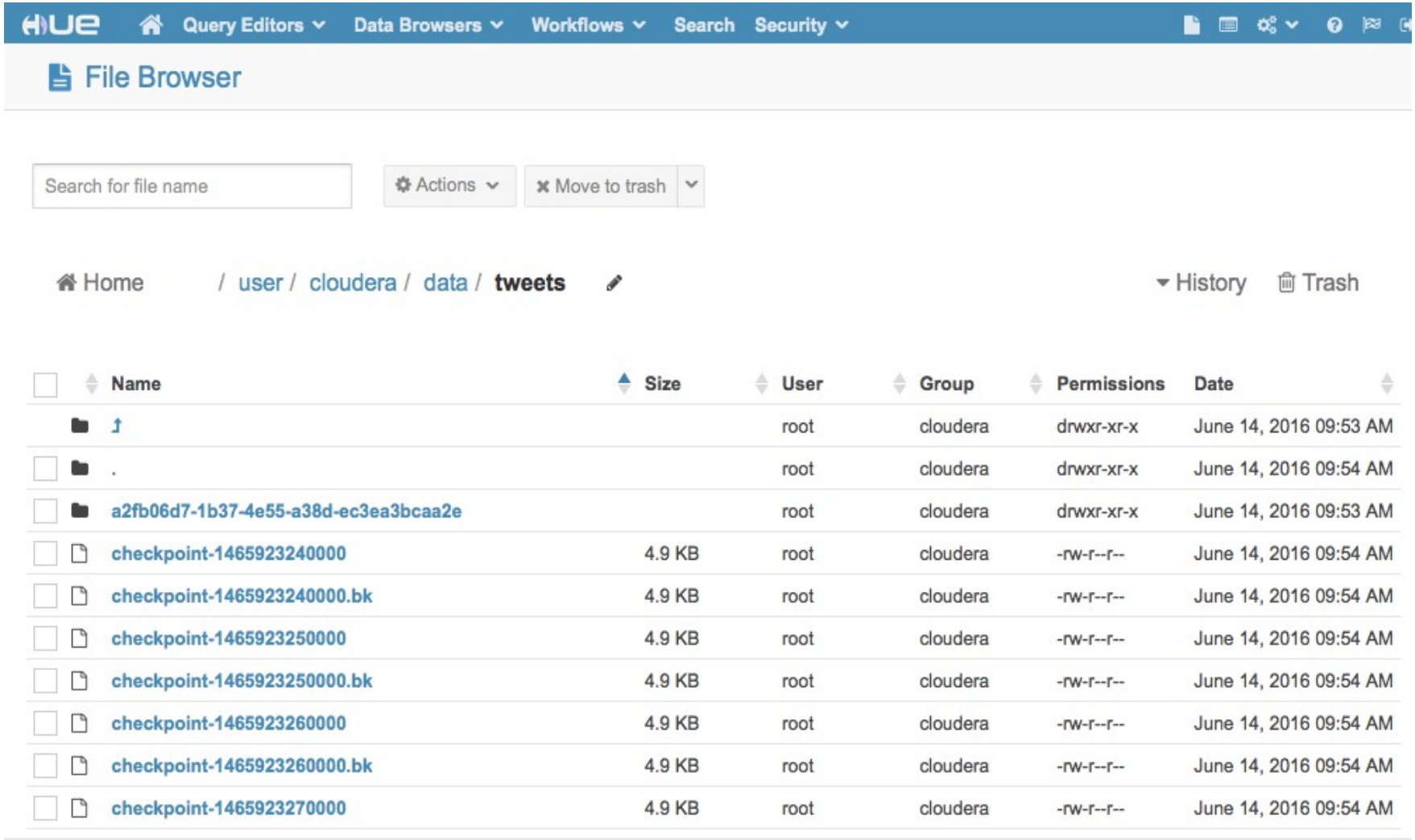
```
$ spark-shell --jars spark-streaming-twitter_2.10-1.2.0.jar,twitter4j-stream-4.0.2.jar,twitter4j-core-4.0.2.jar
```

Running Spark commands

```
$ scala> :paste
// Entering paste mode (ctrl-D to finish)

import org.apache.spark.streaming.twitter._
import twitter4j.auth._
import twitter4j.conf._
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
val ssc = new StreamingContext(sc, Seconds(10))
val cb = new ConfigurationBuilder
```

Running Spark commands

```
cb.setDebugEnabled(true).setOAuthConsumerKey("MjpswndxVj27ylnp
OoSBrnfLX").setOAuthConsumerSecret("QYmuBO1smD5Yc3zE0ZF9ByCgeE
QxnxUmhRVCisAvPFudYVjC4a").setOAuthAccessToken("921172807-
EfMXJj6as2dFECDH1vDe5goyTHcxPrF1RIJozqgx").setOAuthAccessToken
Secret("HbpZEVip3D5j80GP21a37HxA4y10dH9BHcgEFXUNcA9xy")

val auth = new OAuthAuthorization(cb.build)

val tweets = TwitterUtils.createStream(ssc,Some(auth))

val status = tweets.map(status => status.getText)

status.print

ssc.checkpoint("hdfs:///user/cloudera/data/tweets")

ssc.start

ssc.awaitTermination
```


HUE
Query Editors
Data Browsers
Workflows
Search
Security
File Browser
Search for file name
Actions
Move to trash
Home / user / cloudera / data / tweets
History
Trash
Name
Size
User
Group
Permissions
Date
. root cloudera drwxr-xr-x June 14, 2016 09:53 AM
. root cloudera drwxr-xr-x June 14, 2016 09:54 AM
a2fb06d7-1b37-4e55-a38d-ec3ea3bcaa2e root cloudera drwxr-xr-x June 14, 2016 09:53 AM
checkpoint-1465923240000 4.9 KB root cloudera -rw-r--r-- June 14, 2016 09:54 AM
checkpoint-1465923240000.bk 4.9 KB root cloudera -rw-r--r-- June 14, 2016 09:54 AM
checkpoint-1465923250000 4.9 KB root cloudera -rw-r--r-- June 14, 2016 09:54 AM
checkpoint-1465923250000.bk 4.9 KB root cloudera -rw-r--r-- June 14, 2016 09:54 AM
checkpoint-1465923260000 4.9 KB root cloudera -rw-r--r-- June 14, 2016 09:54 AM
checkpoint-1465923260000.bk 4.9 KB root cloudera -rw-r--r-- June 14, 2016 09:54 AM
checkpoint-1465923270000 4.9 KB root cloudera -rw-r--r-- June 14, 2016 09:54 AM

Thank you

www.imcinstitute.com
www.facebook.com/imcinstitute