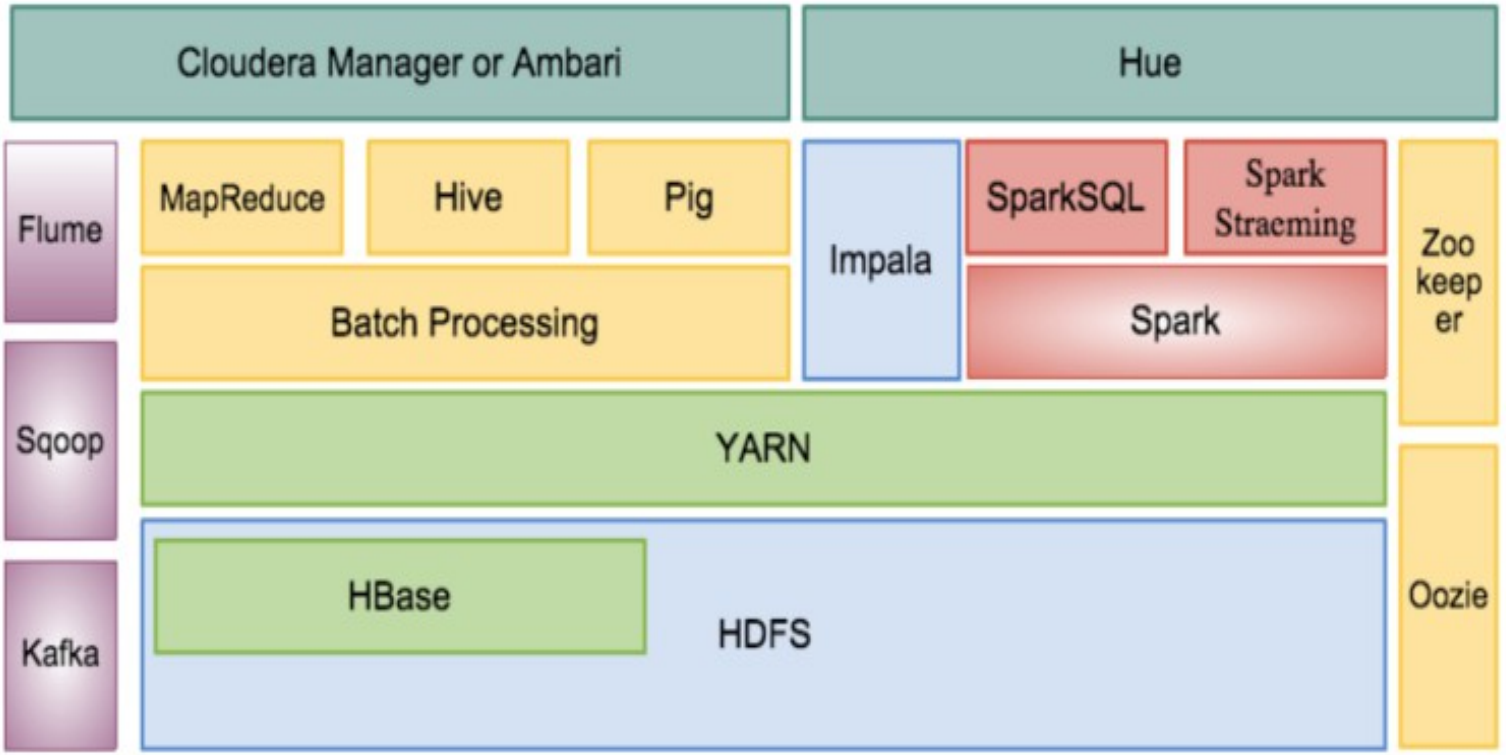
Apache Spark & Hadoop

August 2016

Dr.Thanachart Numnonda
IMC Institute
thanachart@imcinstitute.com

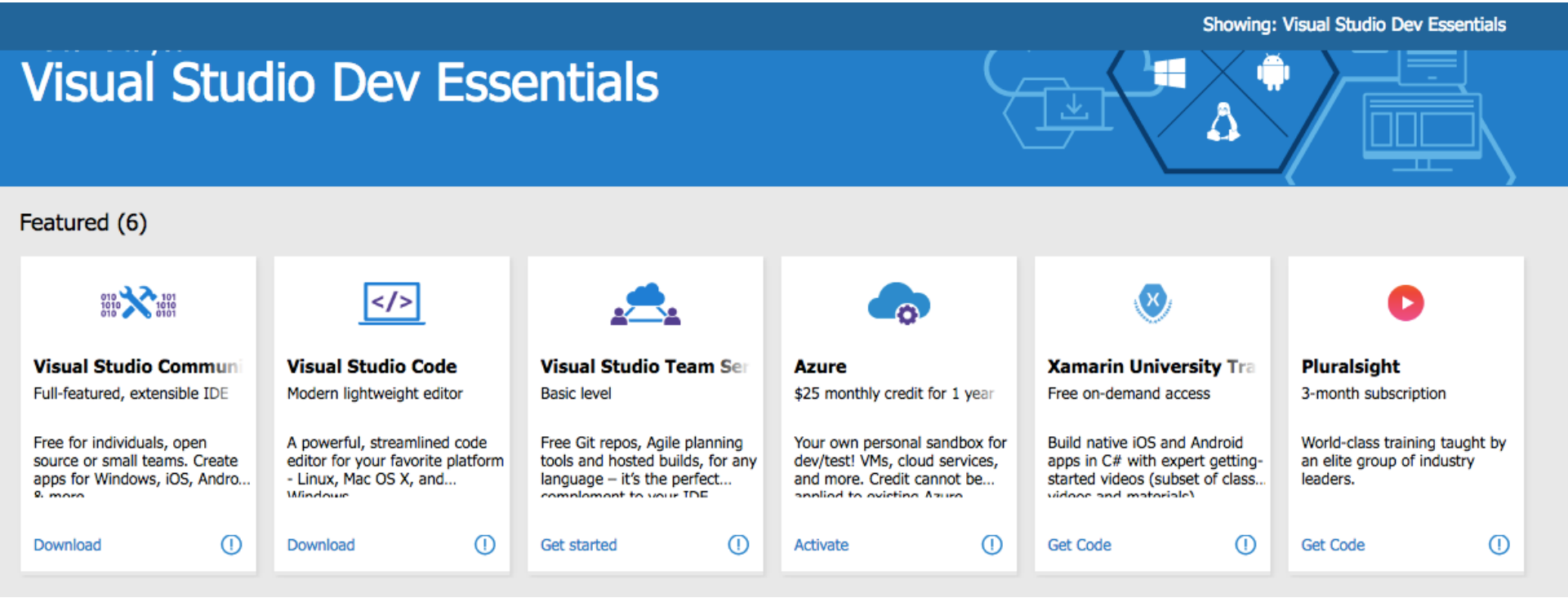
Hadoop Ecosystem



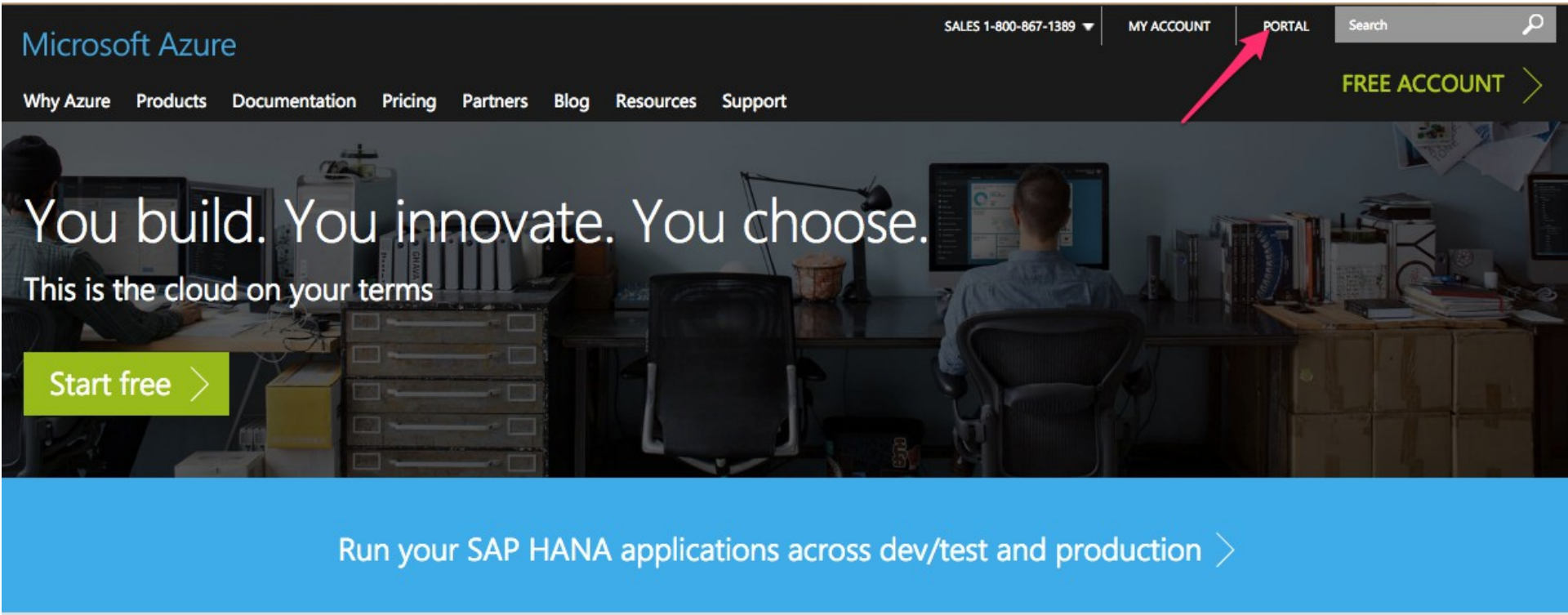
Hands-On: Launch a virtual server on Microsoft Azure

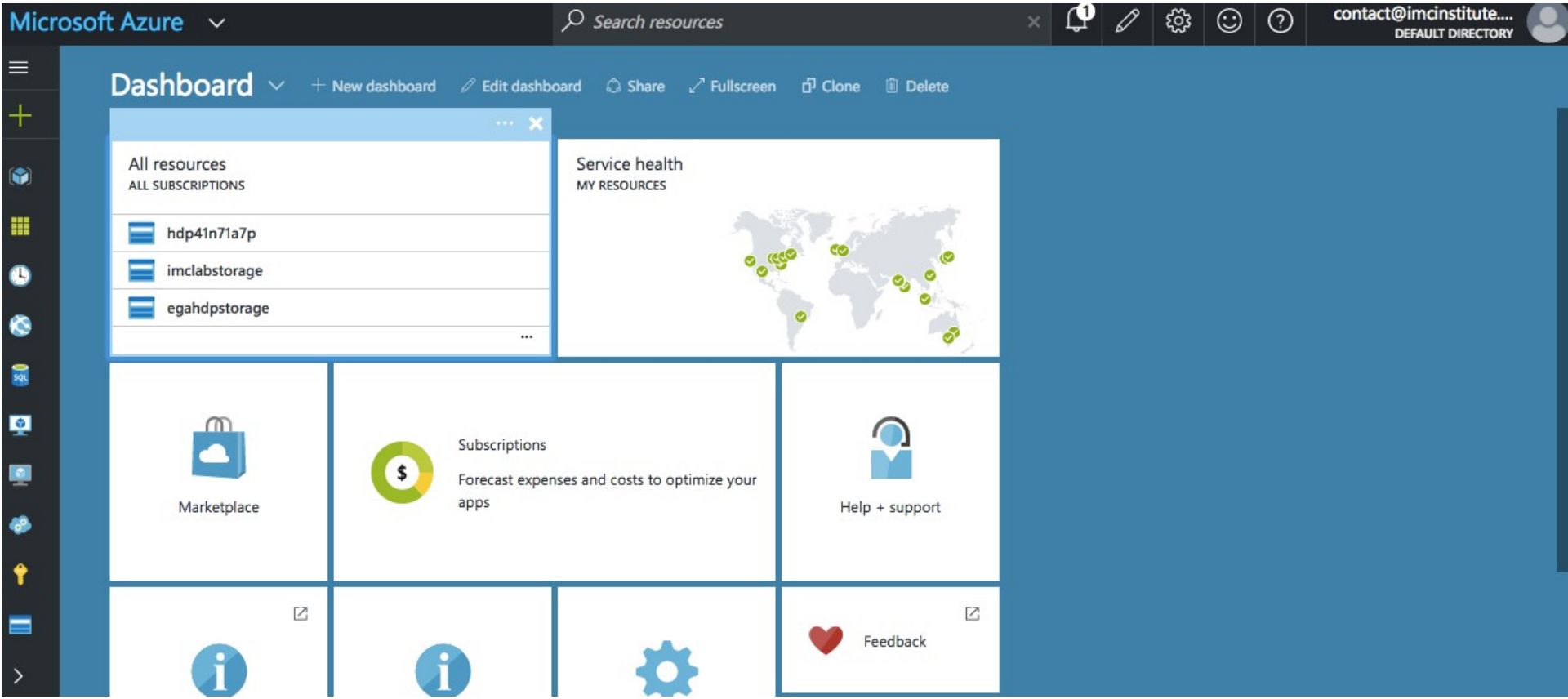
(Note: You can skip this session if you use your own computer or another cloud service)

Sign up for Visual Studio Dev Essential to get free Azure credit.



Sign in to Azure Portal




Microsoft Azure
Search resources
contact@imcinstitute....
DEFAULT DIRECTORY
Dashboard
New dashboard
Edit dashboard
Share
Fullscreen
Clone
Delete
All resources
ALL SUBSCRIPTIONS
hdp41n71a7p
imclabstorage
egahdpstorage
Service health
MY RESOURCES
Marketplace
Subscriptions
Forecast expenses and costs to optimize your apps
Help + support
Feedback

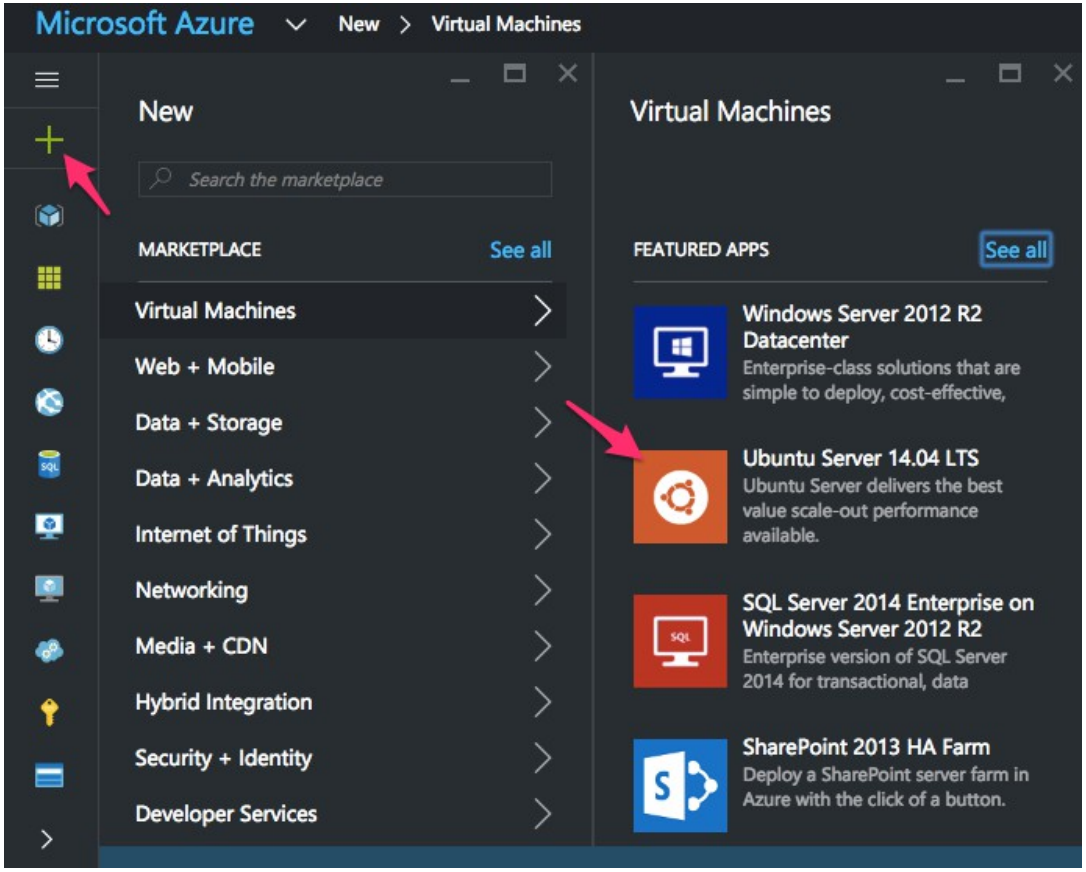
Virtual Server

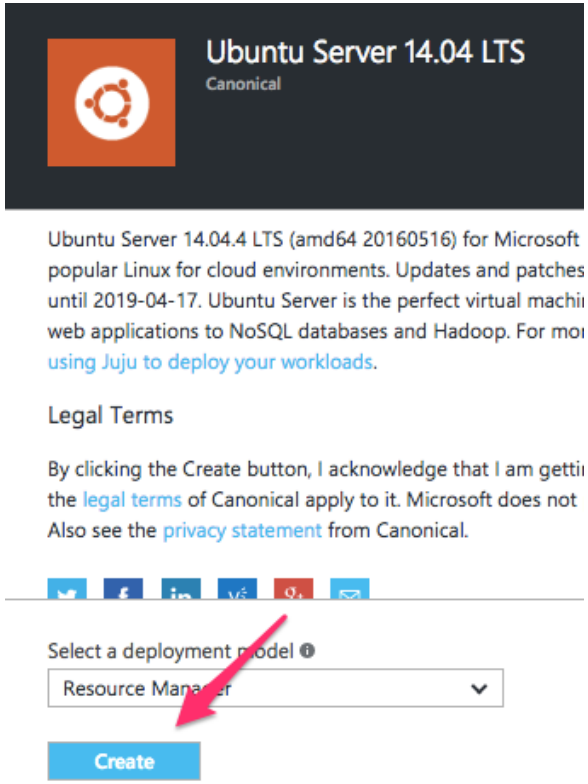
This lab will use an Azure virtual server to install a Cloudera Quickstart Docker using the following features:

Ubuntu Server 14.04 LTS

DS3_V2 Standard 4 Core, 14 GB memory,28 GB SSD

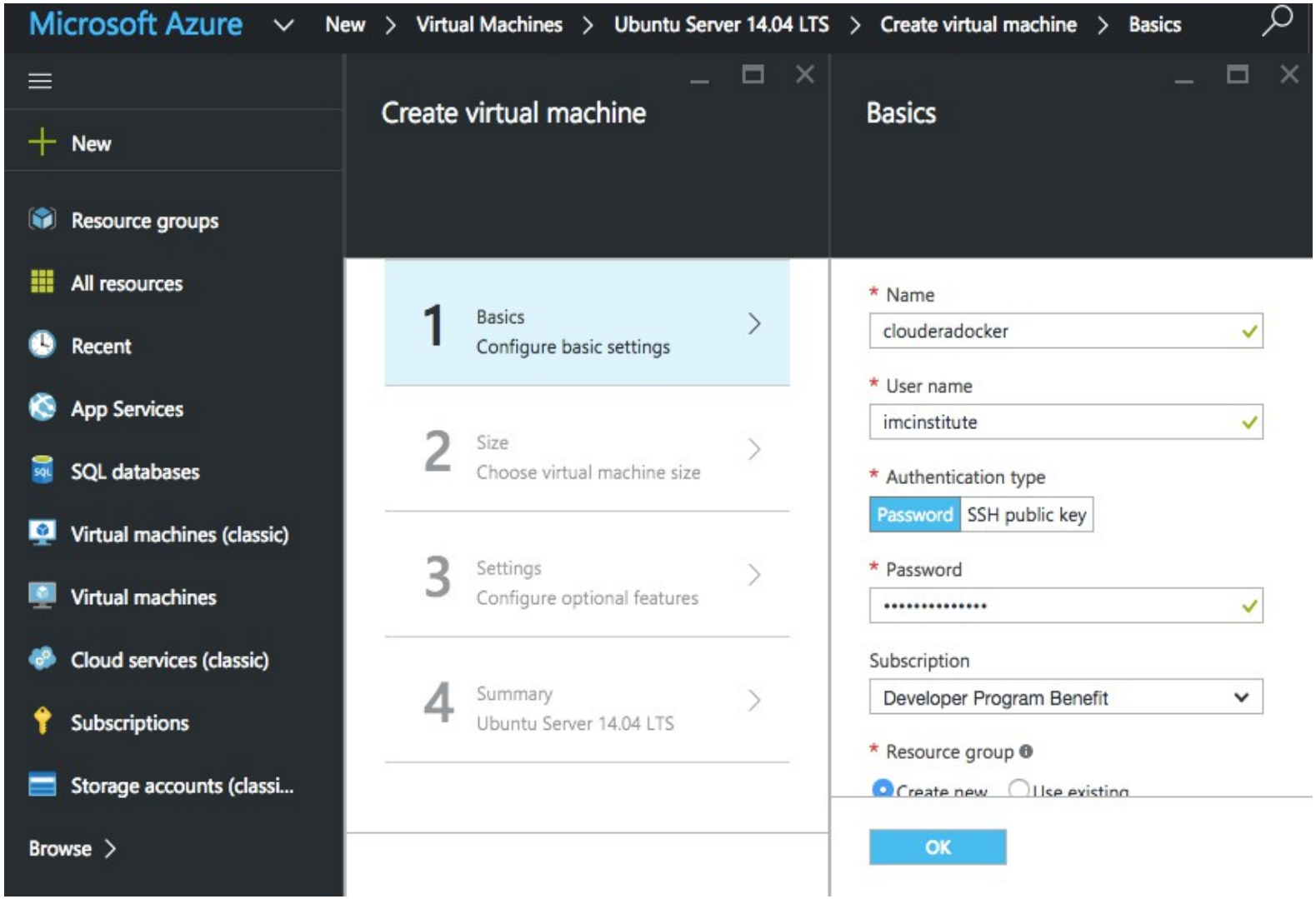
Select New => Virtual Machines => Virtual Machines



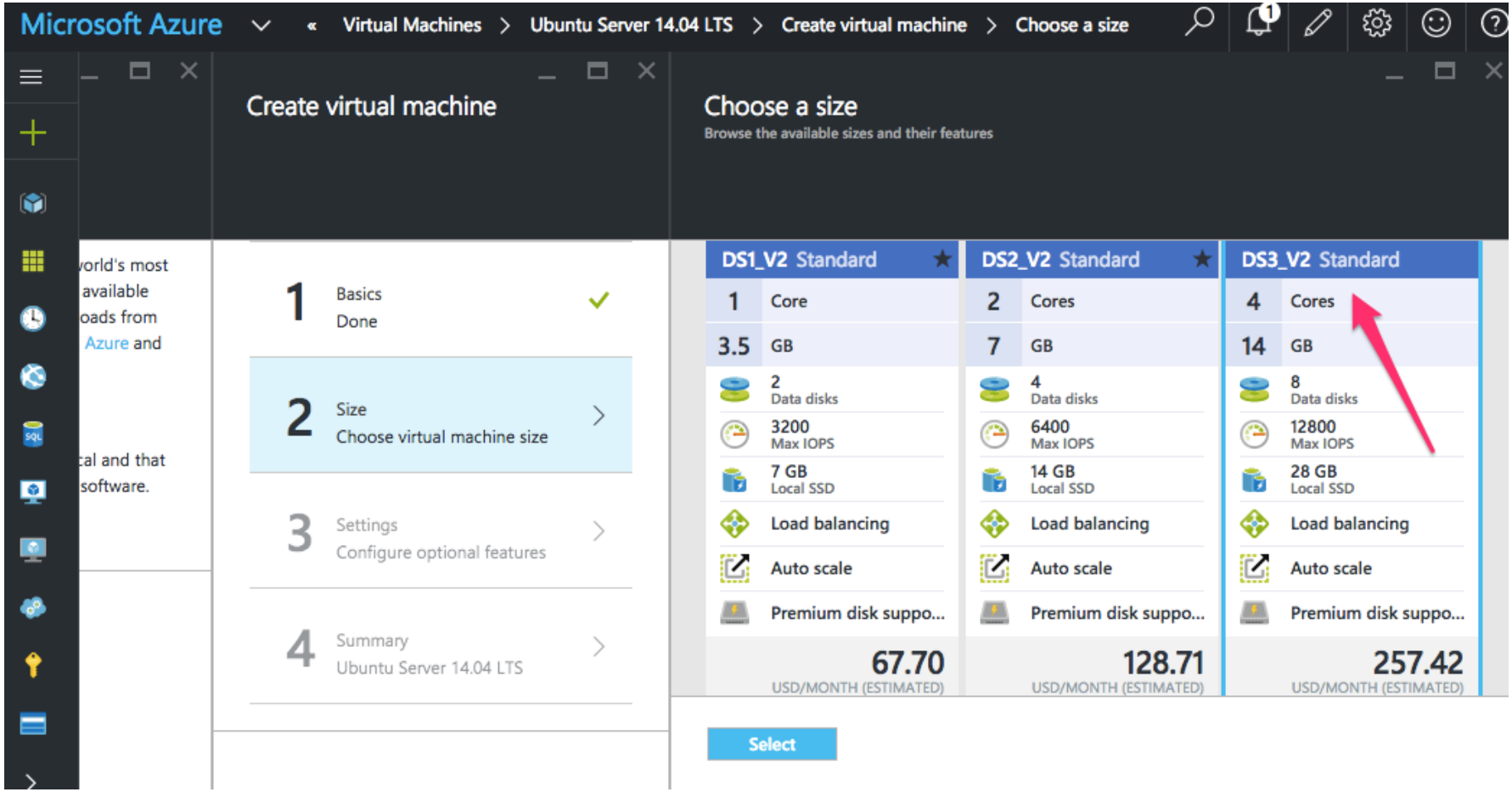


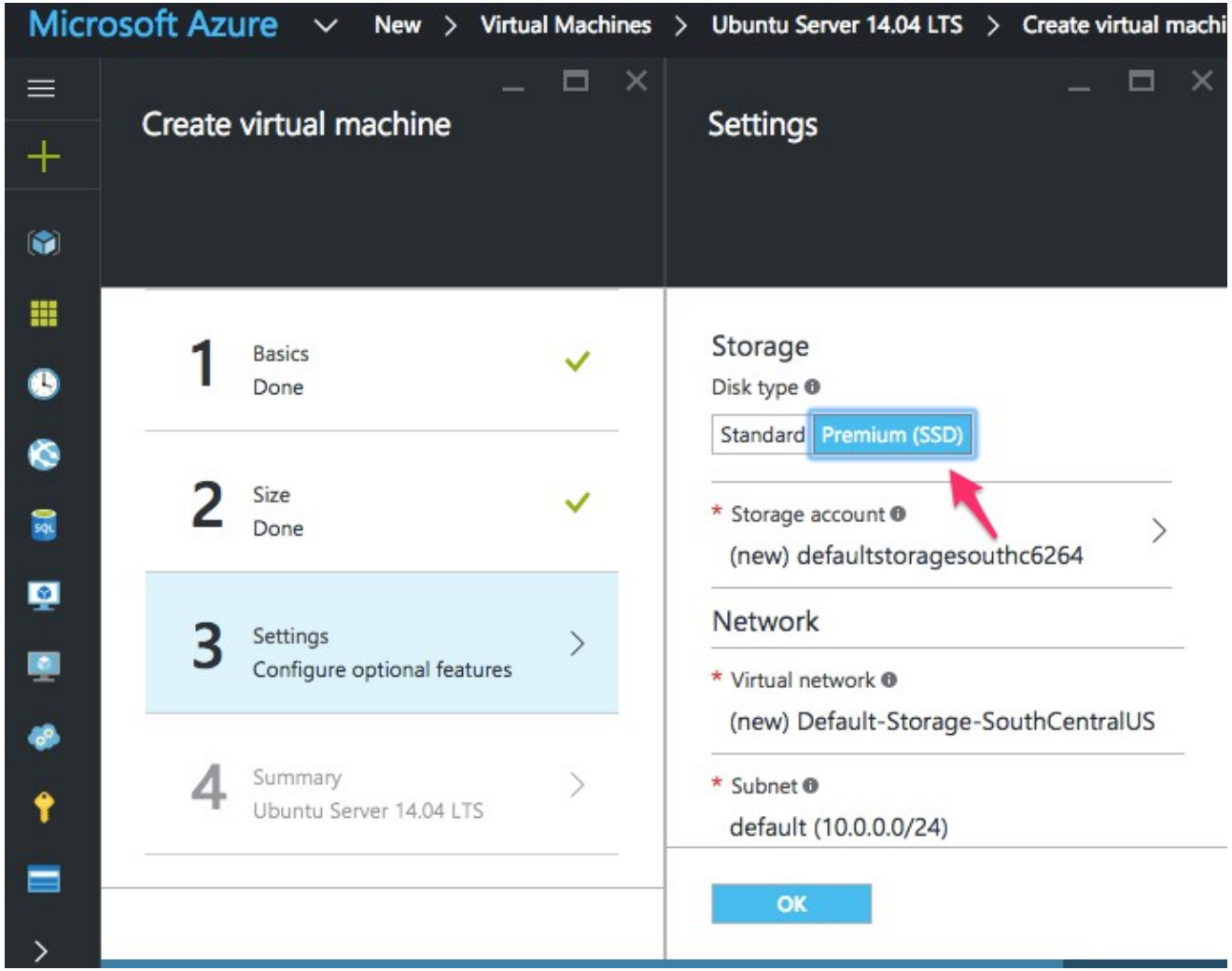
On the Basics page, enter:

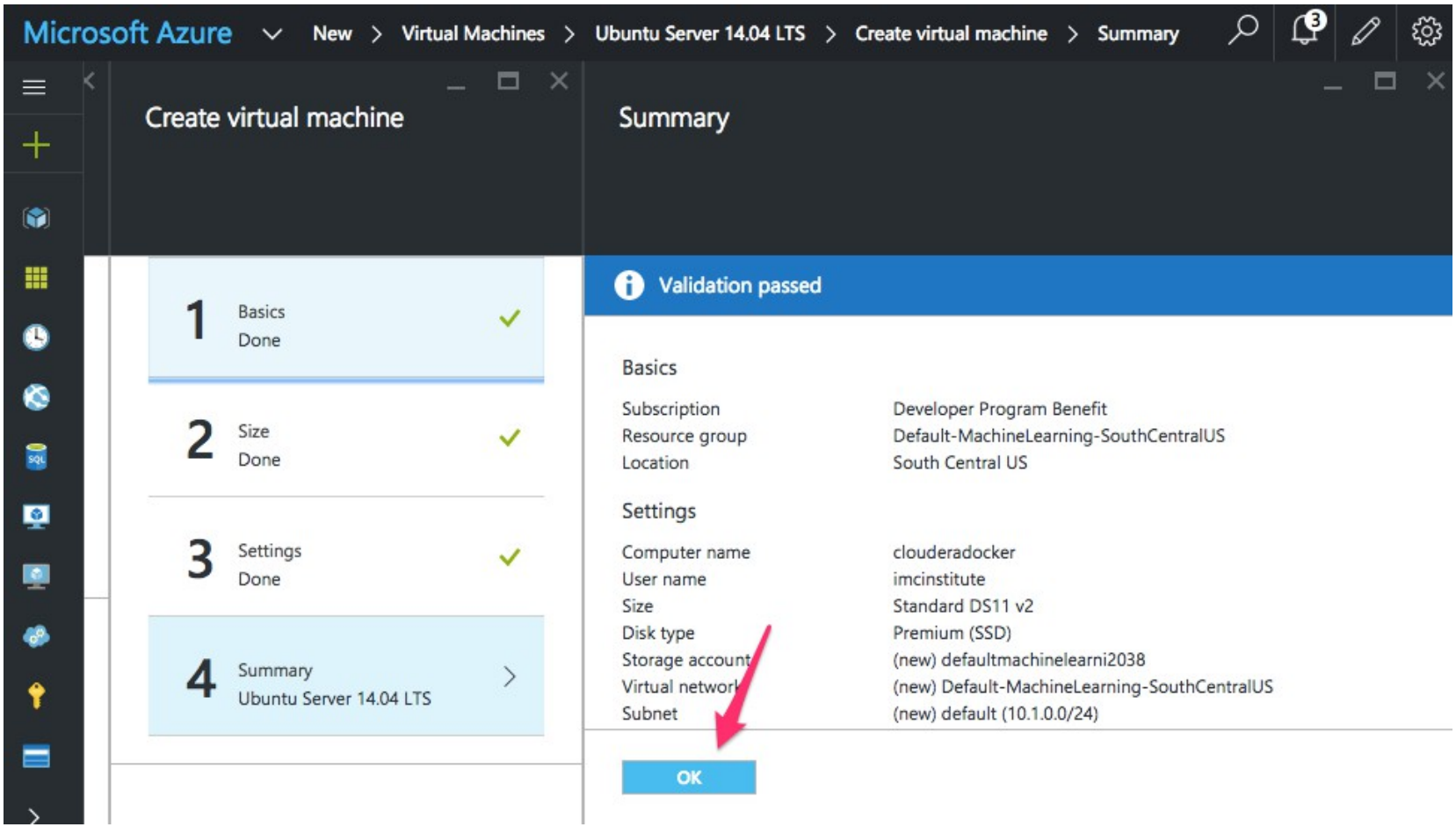
- a name for the VM
- a username for the Admin User
- the Authentication Type set to password
- a password
- a resource group name

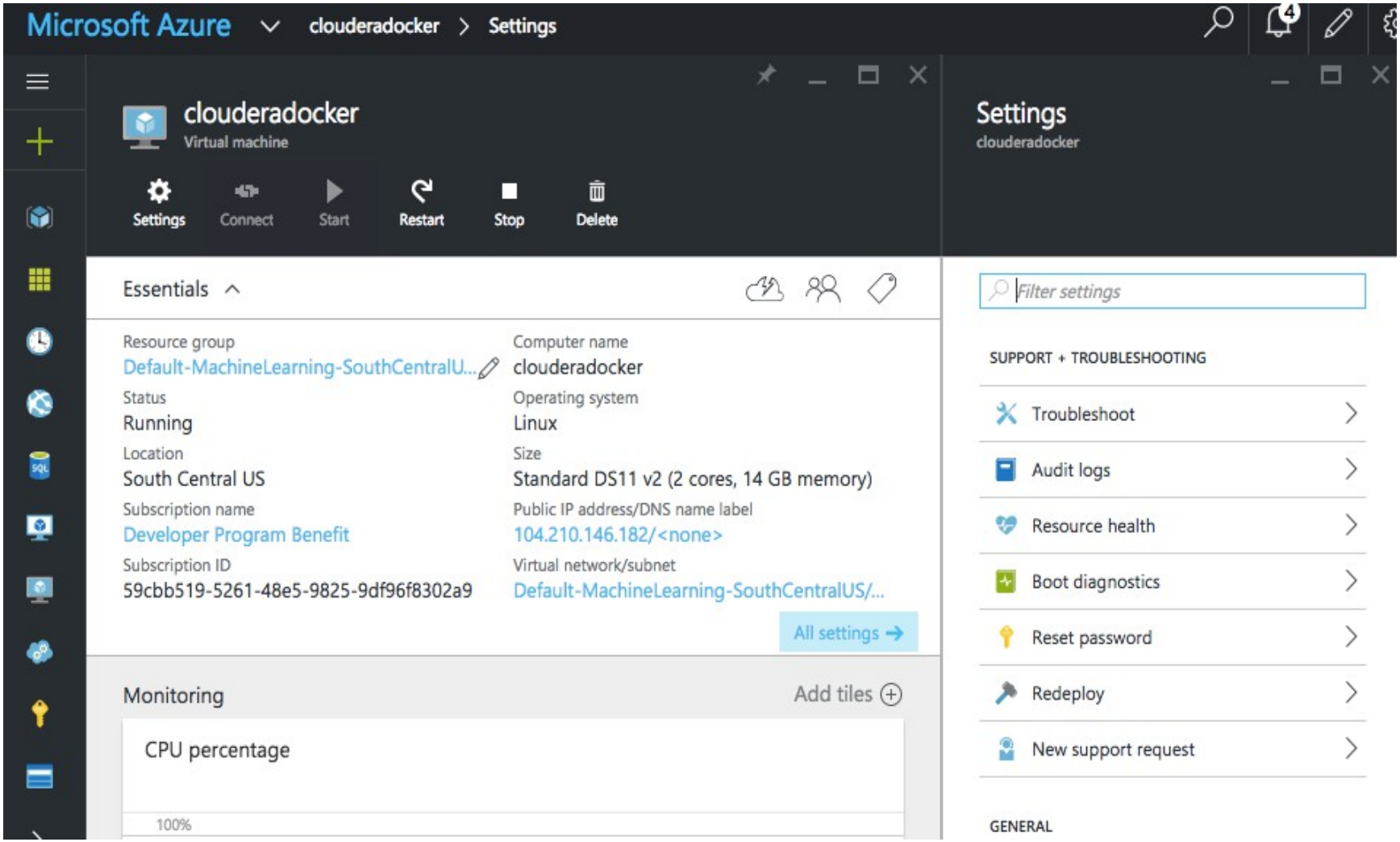

Microsoft Azure
New
Virtual Machines
Ubuntu Server 14.04 LTS
Create virtual machine
Basics
New
Resource groups
All resources
Recent
App Services
SQL databases
Virtual machines (classic)
Virtual machines
Cloud services (classic)
Subscriptions
Storage accounts (classi...
Browse
Create virtual machine
1 Basics
Configure basic settings
2 Size
Choose virtual machine size
3 Settings
Configure optional features
4 Summary
Ubuntu Server 14.04 LTS
Basics
* Name
clouderadocker
* User name
imcinstitute
* Authentication type
Password
SSH public key
* Password
Subscription
Developer Program Benefit
* Resource group
Create new
Use existing
OK

Choose DS3_v2 Standard

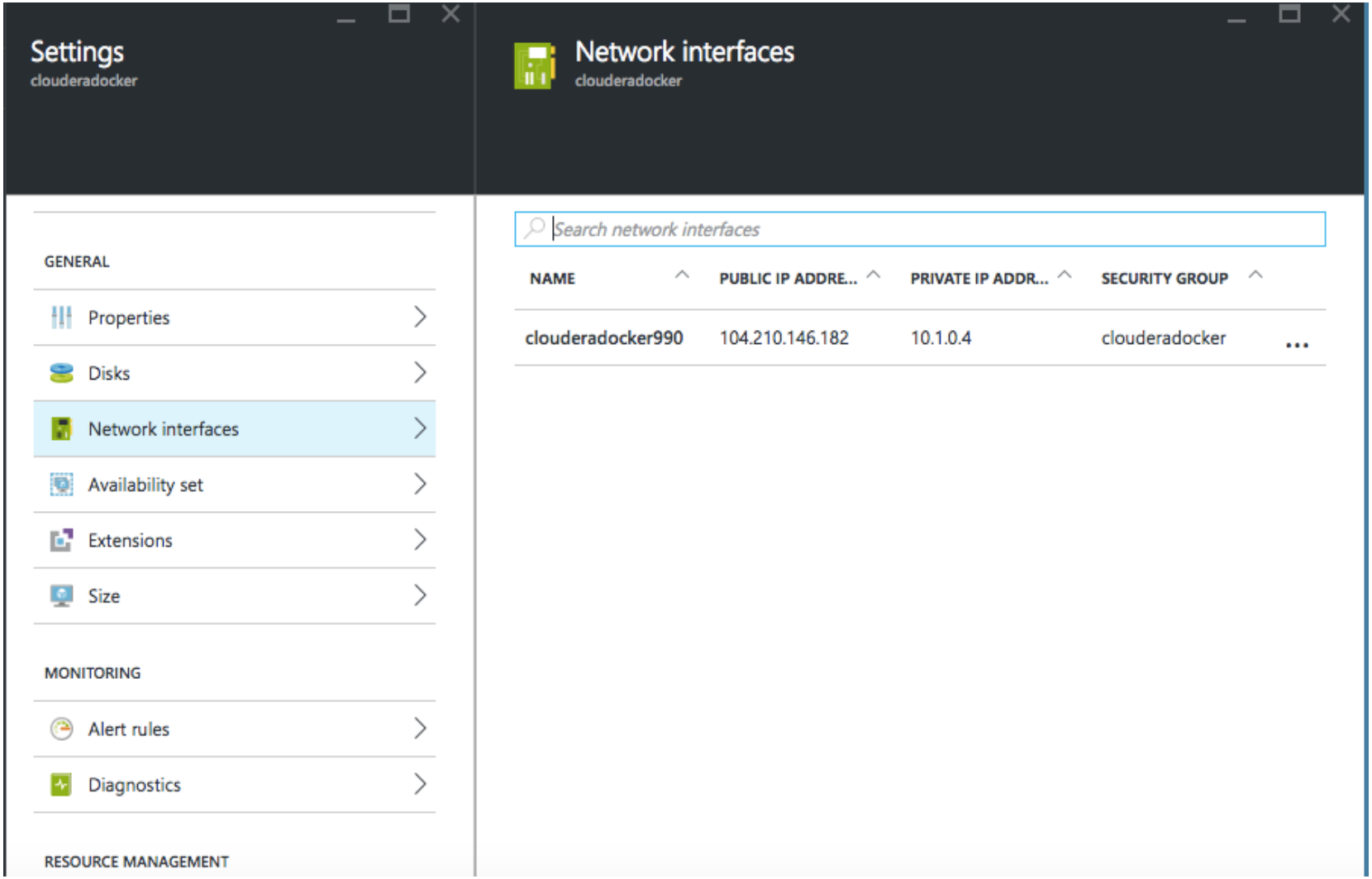


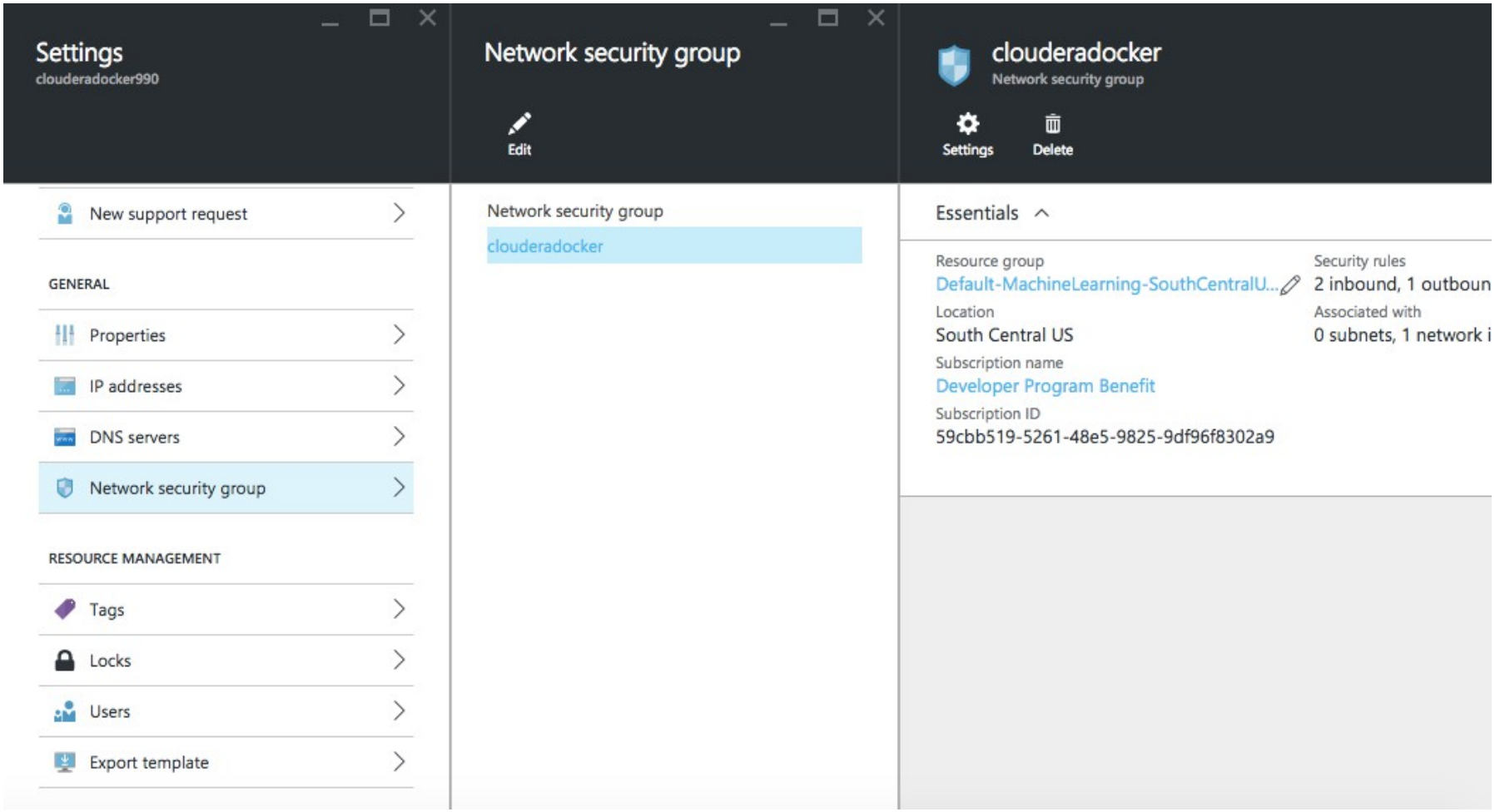

Microsoft Azure
New
Virtual Machines
Ubuntu Server 14.04 LTS
Create virtual machi
Create virtual machine
Settings
1
Basics
Done
2
Size
Done
3
Settings
Configure optional features
4
Summary
Ubuntu Server 14.04 LTS
Storage
Disk type
Standard
Premium (SSD)
* Storage account
(new) defaultstoragesouthc6264
Network
* Virtual network
(new) Default-Storage-SouthCentralUS
* Subnet
default (10.0.0.0/24)
OK

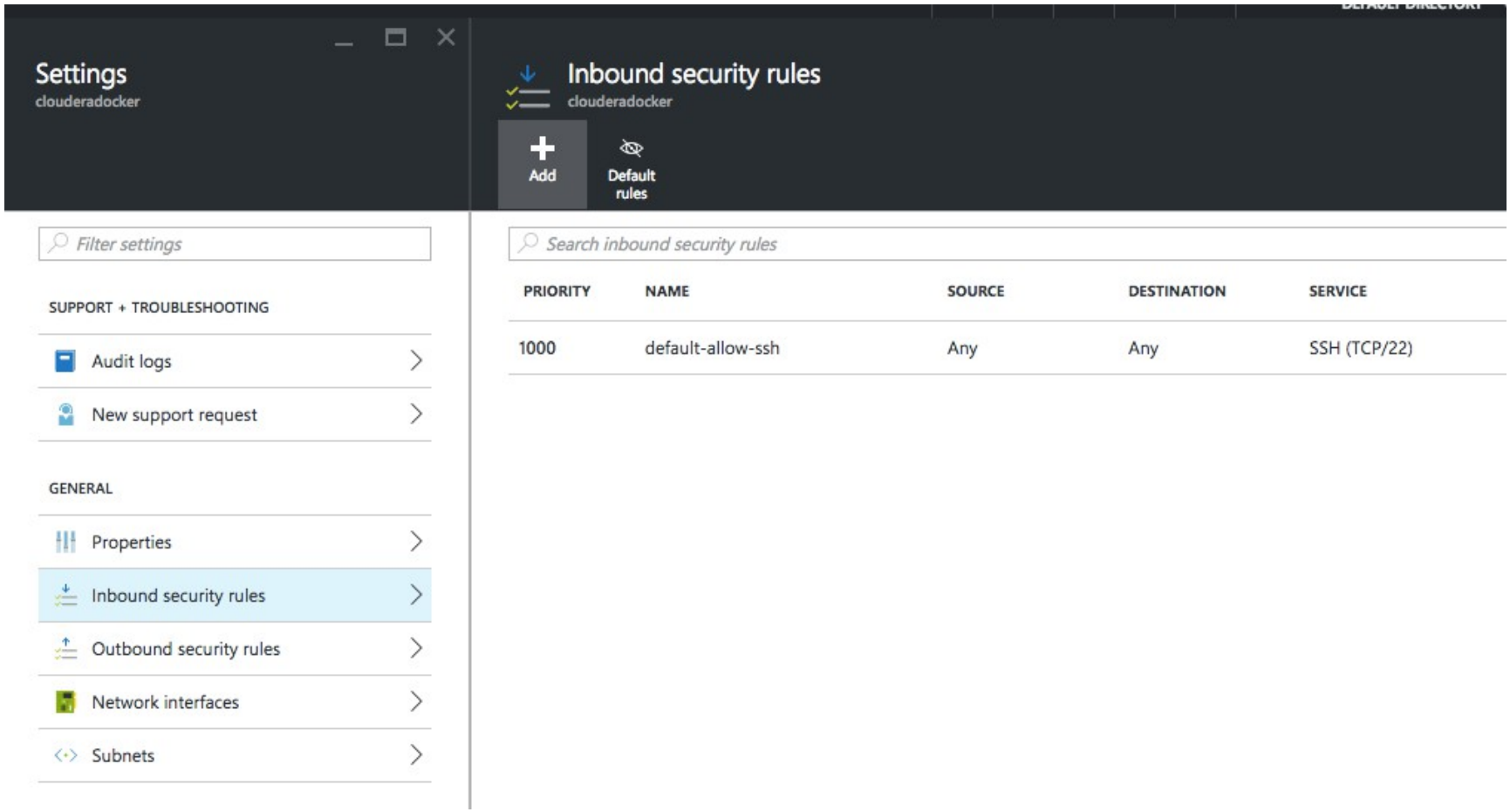

Microsoft Azure
New
Virtual Machines
Ubuntu Server 14.04 LTS
Create virtual machine
Summary
Create virtual machine
1 Basics Done
2 Size Done
3 Settings Done
4 Summary Ubuntu Server 14.04 LTS
Summary
Validation passed
Basics
Subscription Developer Program Benefit
Resource group Default-MachineLearning-SouthCentralUS
Location South Central US
Settings
Computer name clouderadocker
User name imcinstitute
Size Standard DS11 v2
Disk type Premium (SSD)
Storage account (new) defaultmachinelearni2038
Virtual network (new) Default-MachineLearning-SouthCentralUS
Subnet (new) default (10.1.0.0/24)
OK

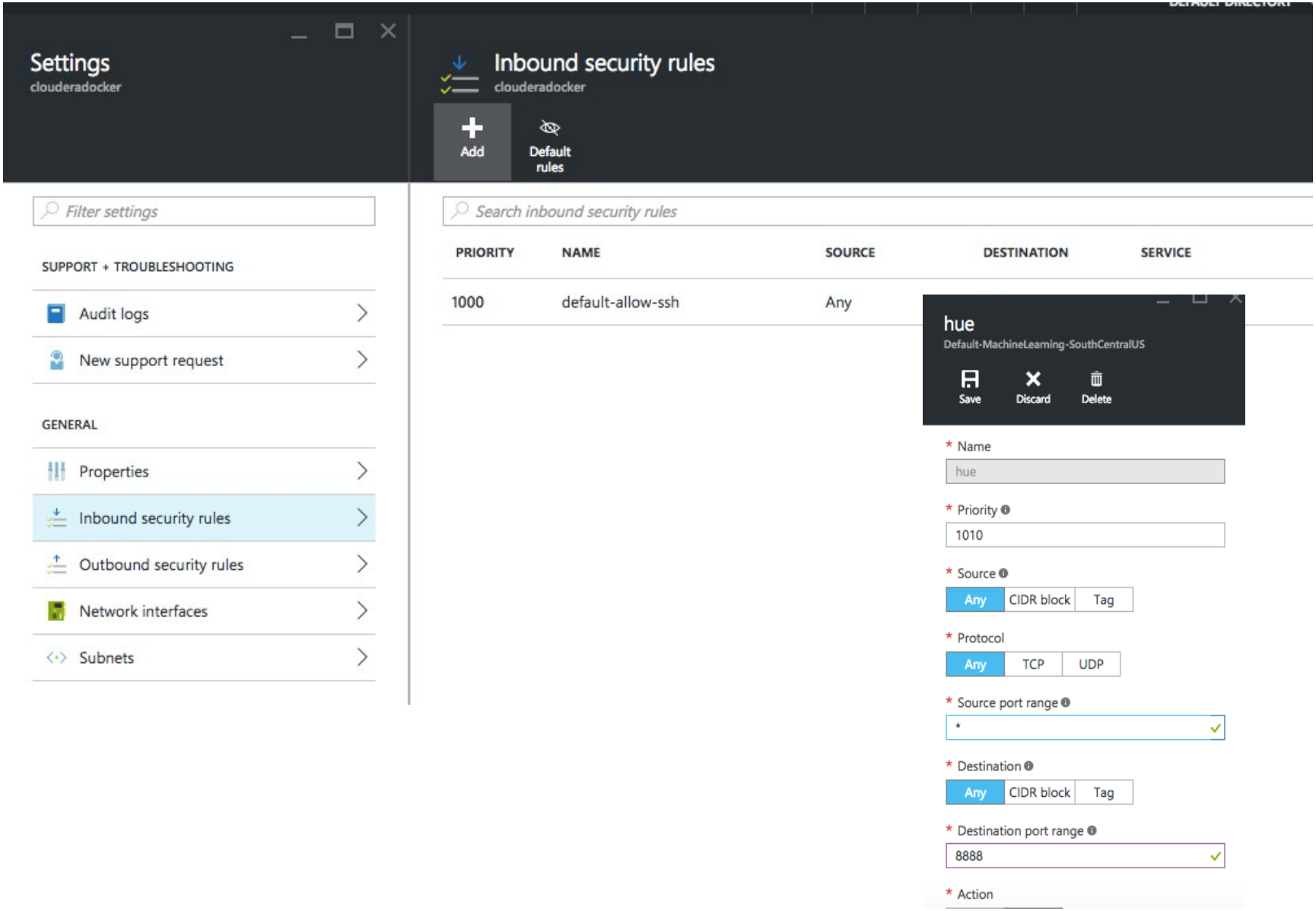

Microsoft Azure
cloderadocker
Settings
cloderadocker
Virtual machine
Settings
Connect
Start
Restart
Stop
Delete
Essentials
Resource group
Default-MachineLearning-SouthCentralU...
Computer name
cloderadocker
Status
Running
Operating system
Linux
Location
South Central US
Size
Standard DS11 v2 (2 cores, 14 GB memory)
Subscription name
Developer Program Benefit
Public IP address/DNS name label
104.210.146.182/<none>
Subscription ID
59cbb519-5261-48e5-9825-9df96f8302a9
Virtual network/subnet
Default-MachineLearning-SouthCentralUS/...
All settings
Monitoring
Add tiles
CPU percentage
100%
Settings
cloderadocker
Filter settings
SUPPORT + TROUBLESHOOTING
Troubleshoot
Audit logs
Resource health
Boot diagnostics
Reset password
Redeploy
New support request
GENERAL

Setting the inbound port for Hue (8888)

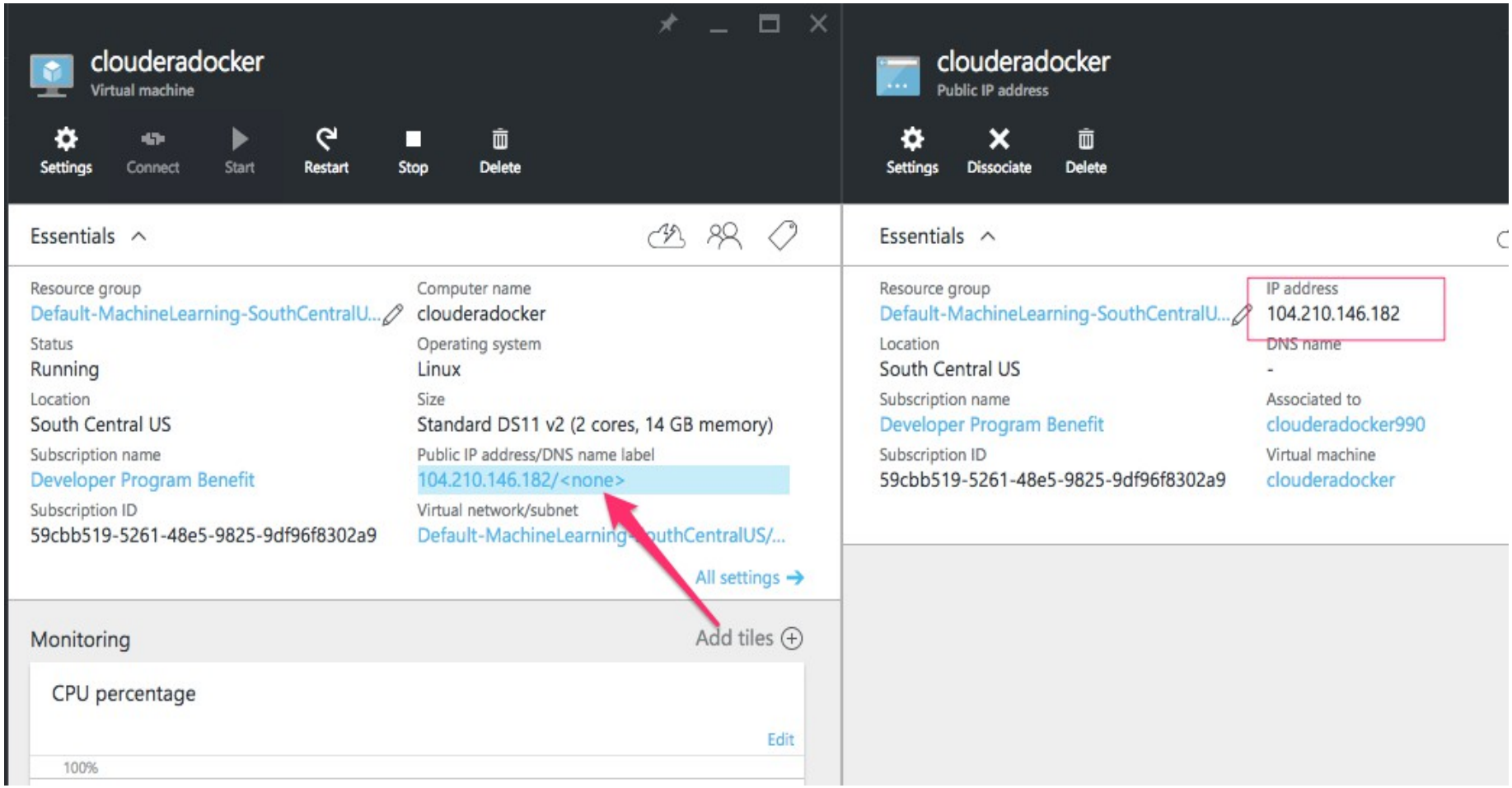



Settings
clouderadocker990
New support request
GENERAL
Properties
IP addresses
DNS servers
Network security group
RESOURCE MANAGEMENT
Tags
Locks
Users
Export template
Network security group
Edit
Network security group
clouderadocker
clouderadocker
Network security group
Settings
Delete
Essentials
Resource group
Default-MachineLearning-SouthCentralU...
Location
South Central US
Subscription name
Developer Program Benefit
Subscription ID
59cbb519-5261-48e5-9825-9df96f8302a9
Security rules
2 inbound, 1 outboun
Associated with
0 subnets, 1 network i


Settings
clouderadocker
Inbound security rules
clouderadocker
Add
Default rules
Filter settings
Search inbound security rules
SUPPORT + TROUBLESHOOTING
Audit logs
New support request
GENERAL
Properties
Inbound security rules
Outbound security rules
Network interfaces
Subnets
PRIORITY
NAME
SOURCE
DESTINATION
SERVICE
1000
default-allow-ssh
Any
Any
SSH (TCP/22)


Settings
clouderadocker
Inbound security rules
clouderadocker
Add
Default rules
Filter settings
SUPPORT + TROUBLESHOOTING
Audit logs
New support request
GENERAL
Properties
Inbound security rules
Outbound security rules
Network interfaces
Subnets
Search inbound security rules
PRIORITY
NAME
SOURCE
DESTINATION
SERVICE
1000
default-allow-ssh
Any
hue
Default-MachineLearning-SouthCentralUS
Save
Discard
Delete
* Name
hue
* Priority
1010
* Source
Any
CIDR block
Tag
* Protocol
Any
TCP
UDP
* Source port range
*
* Destination
Any
CIDR block
Tag
* Destination port range
8888
* Action

Get the IP address

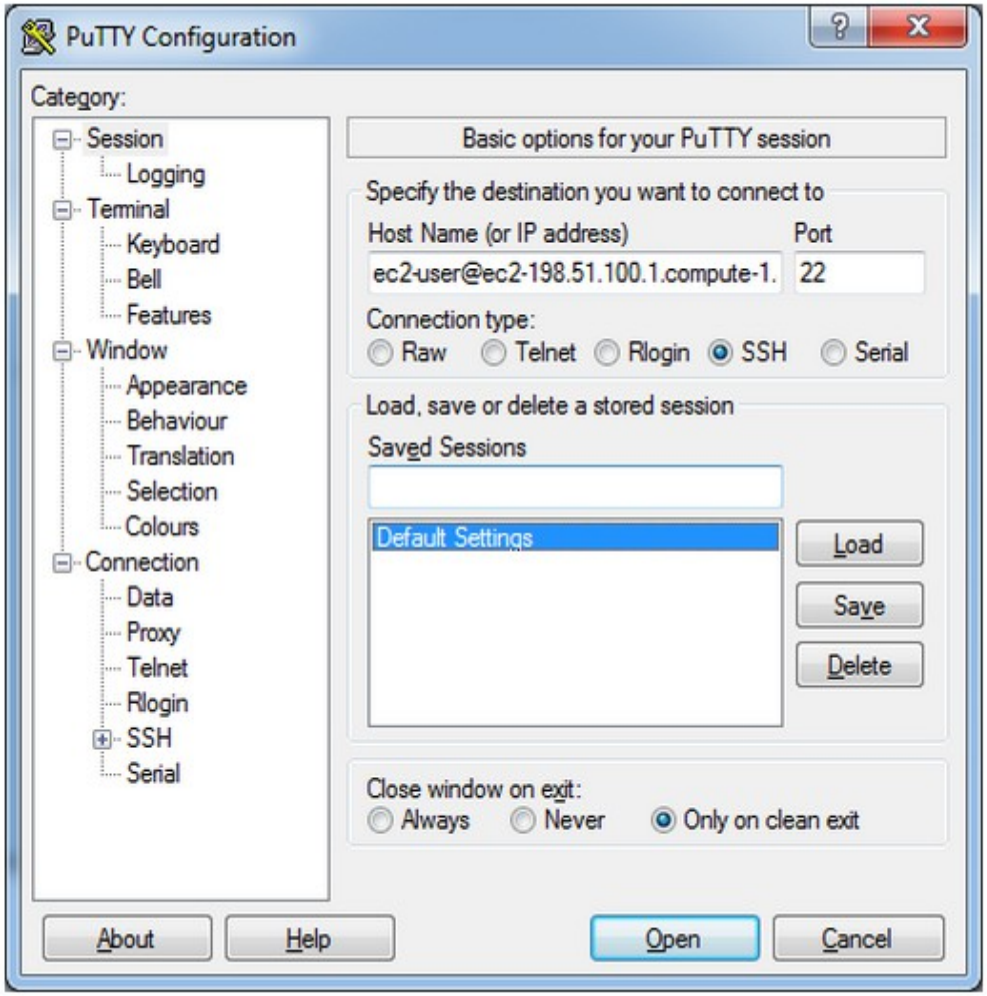


Connect to an instance from Mac/Linux
ssh -i ~/.ssh/id_rsa imcinstitute@104.210.146.182

```
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
```

```
imcinstitute@clouderadocker:~$
```

Connect to an instance from Windows using Putty



Hands-On: Installing Cloudera Quickstart on Docker Container

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
```

Docker commands:

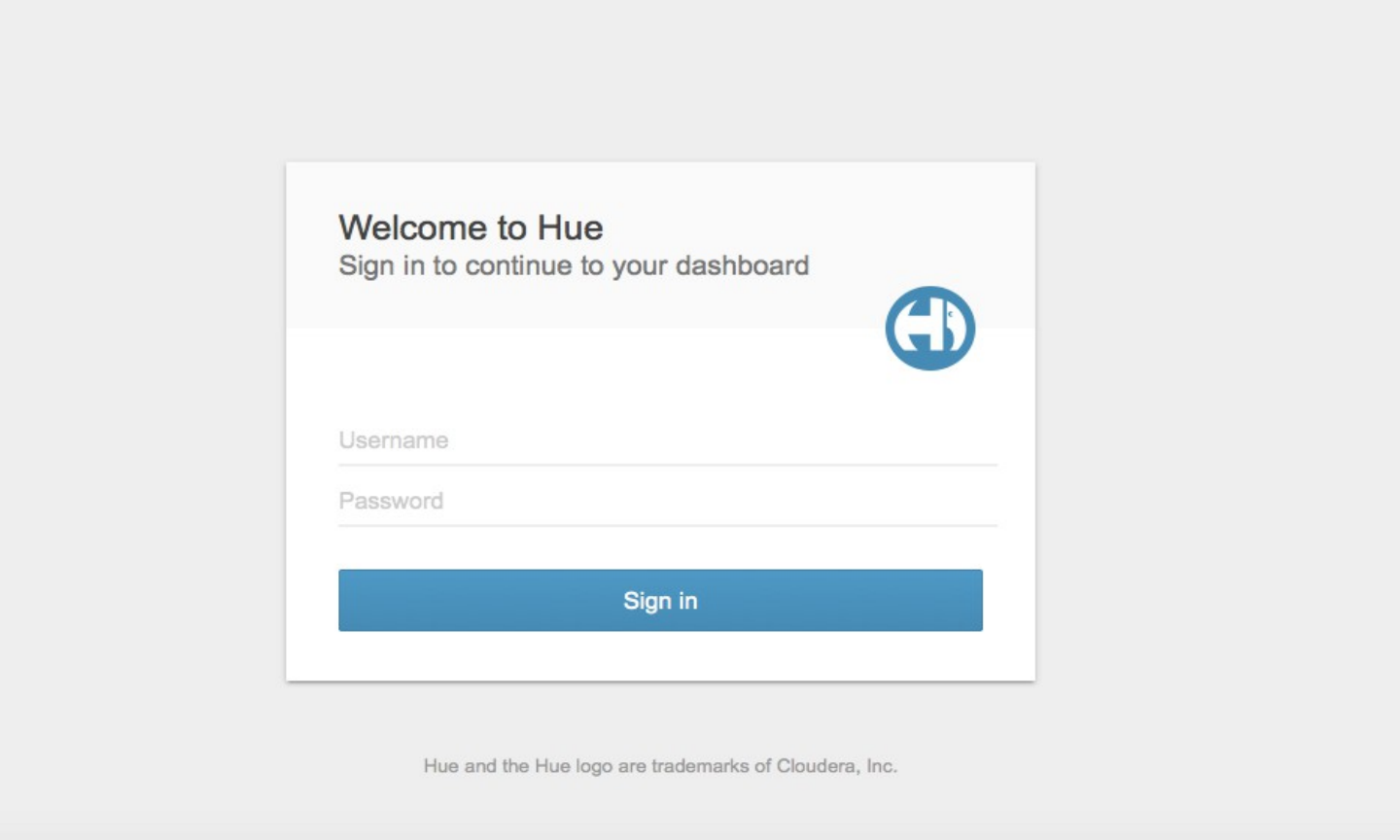
- *docker images*
- *docker ps*
- *docker attach id*
- *docker kill id*

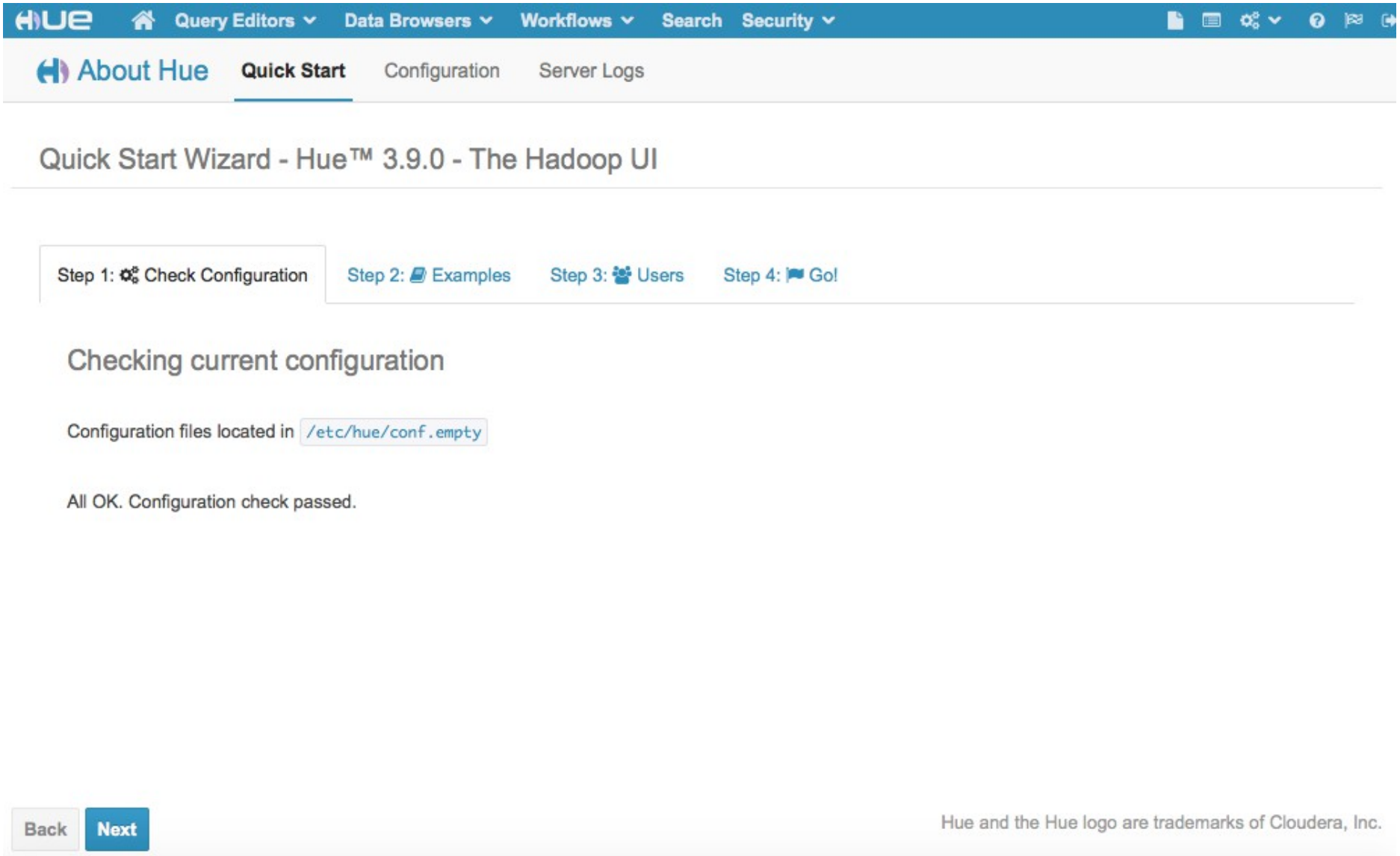
- *Exit from container*
  - *exit (exit & kill the running image)*
  - *Ctrl-P, Ctrl-Q (exit without killing the running image)*

Hive.apache.org

Login to Hue

http://104.210.146.182:8888
(username = cloudera; password = cloudera)




HUE
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.

Lecture
Understanding Spark

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

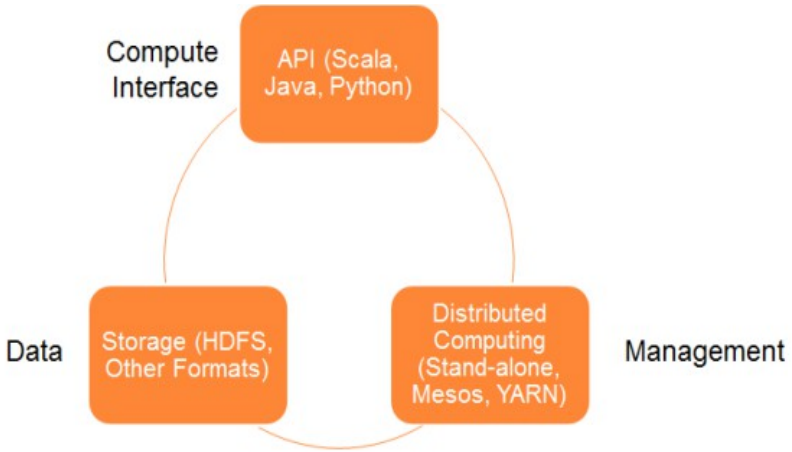
Introduction

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?

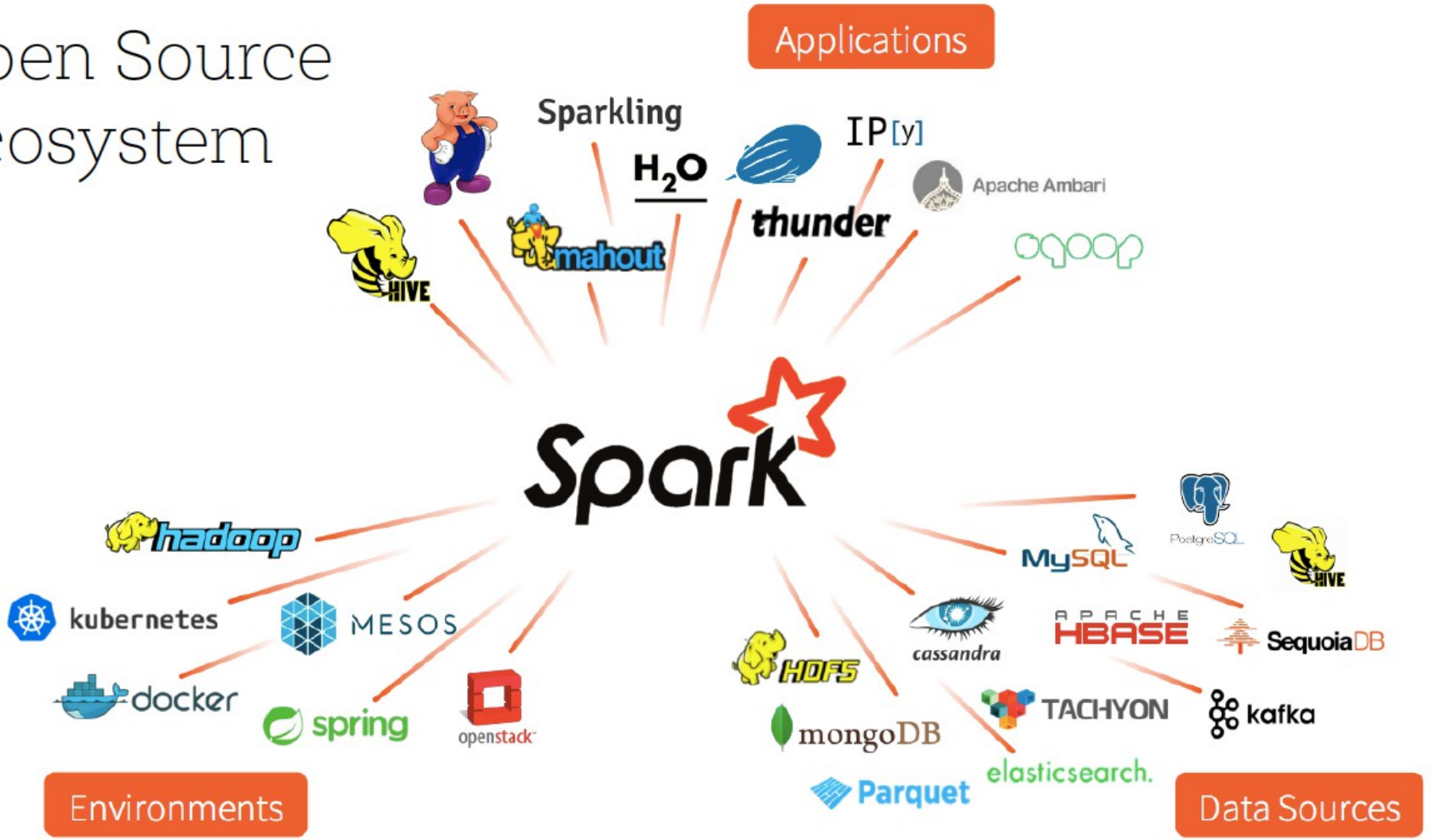
- Framework for distributed processing.
- In-memory, fault tolerant data structures
- Flexible APIs in Scala, Java, Python, SQL, R
- Open source



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

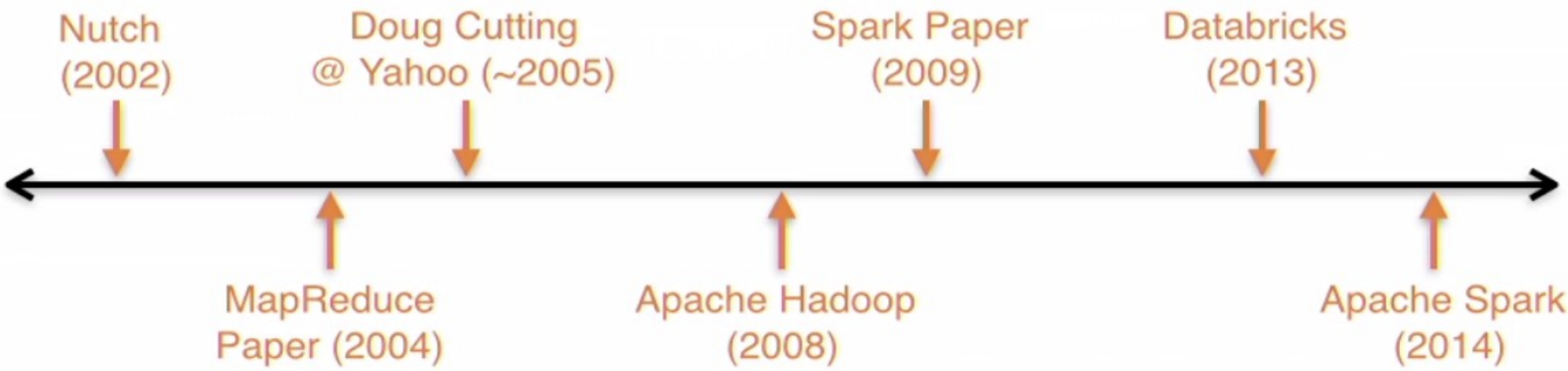
Open Source Ecosystem

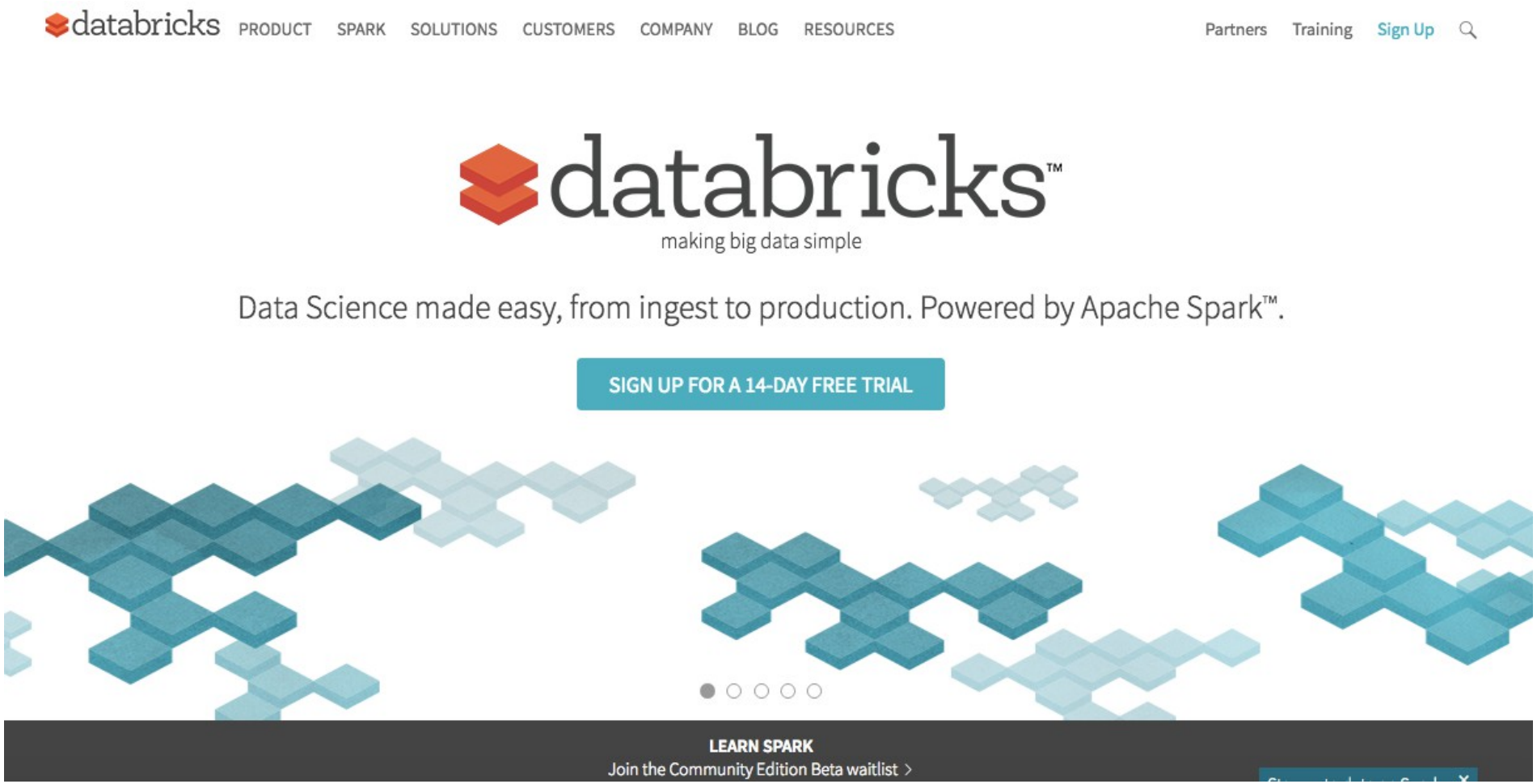


Source: Jump start into Apache Spark and Databricks

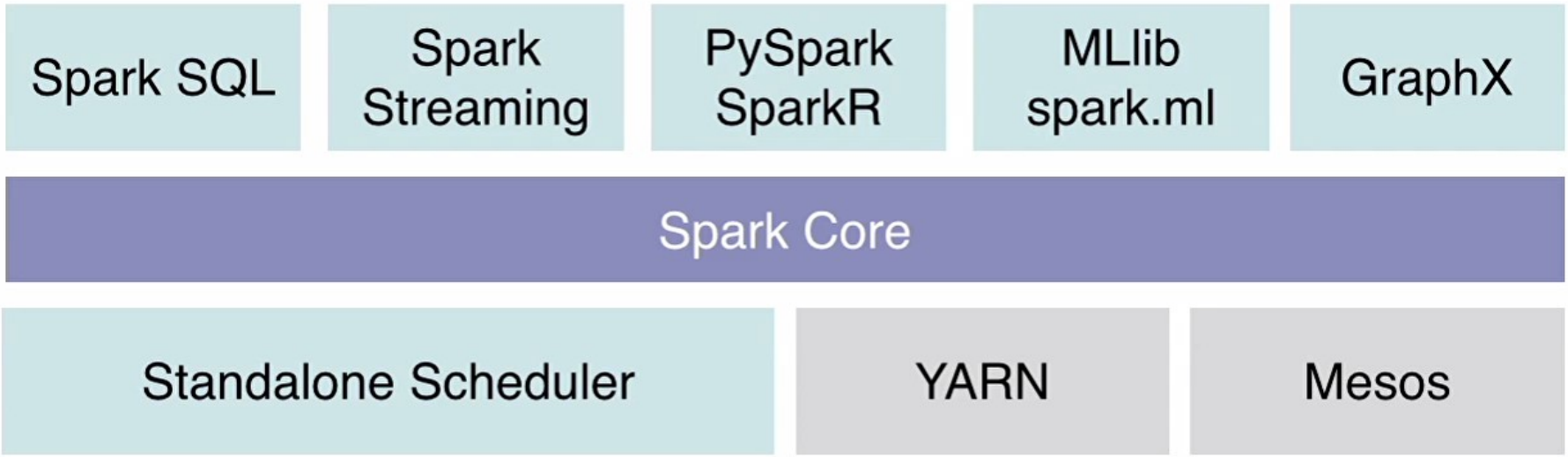
Spark: History

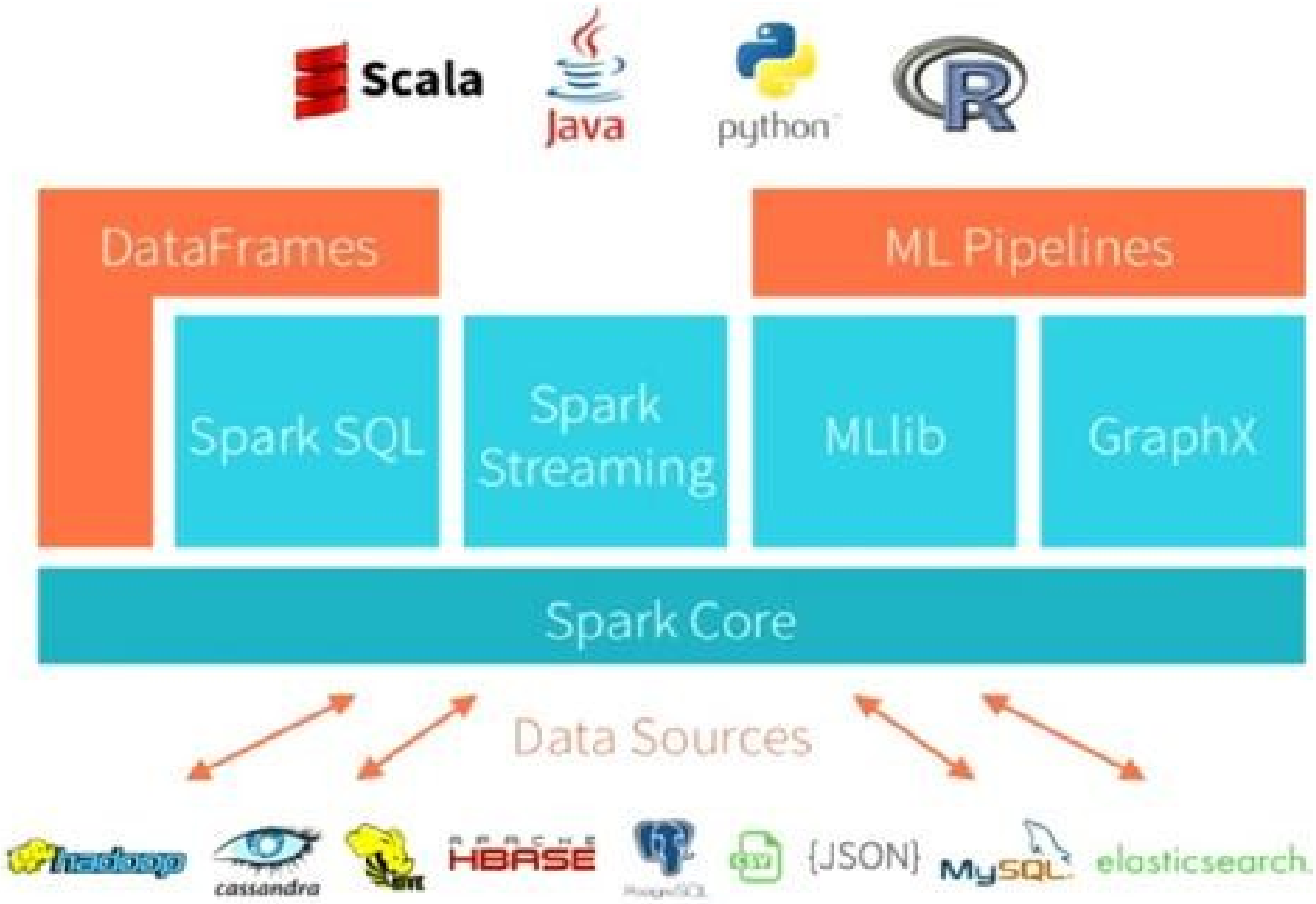
- Founded by AMPlab, UC Berkeley
- Created by Matei Zaharia (PhD Thesis)
- Maintained by Apache Software Foundation
- Commercial support by Databricks



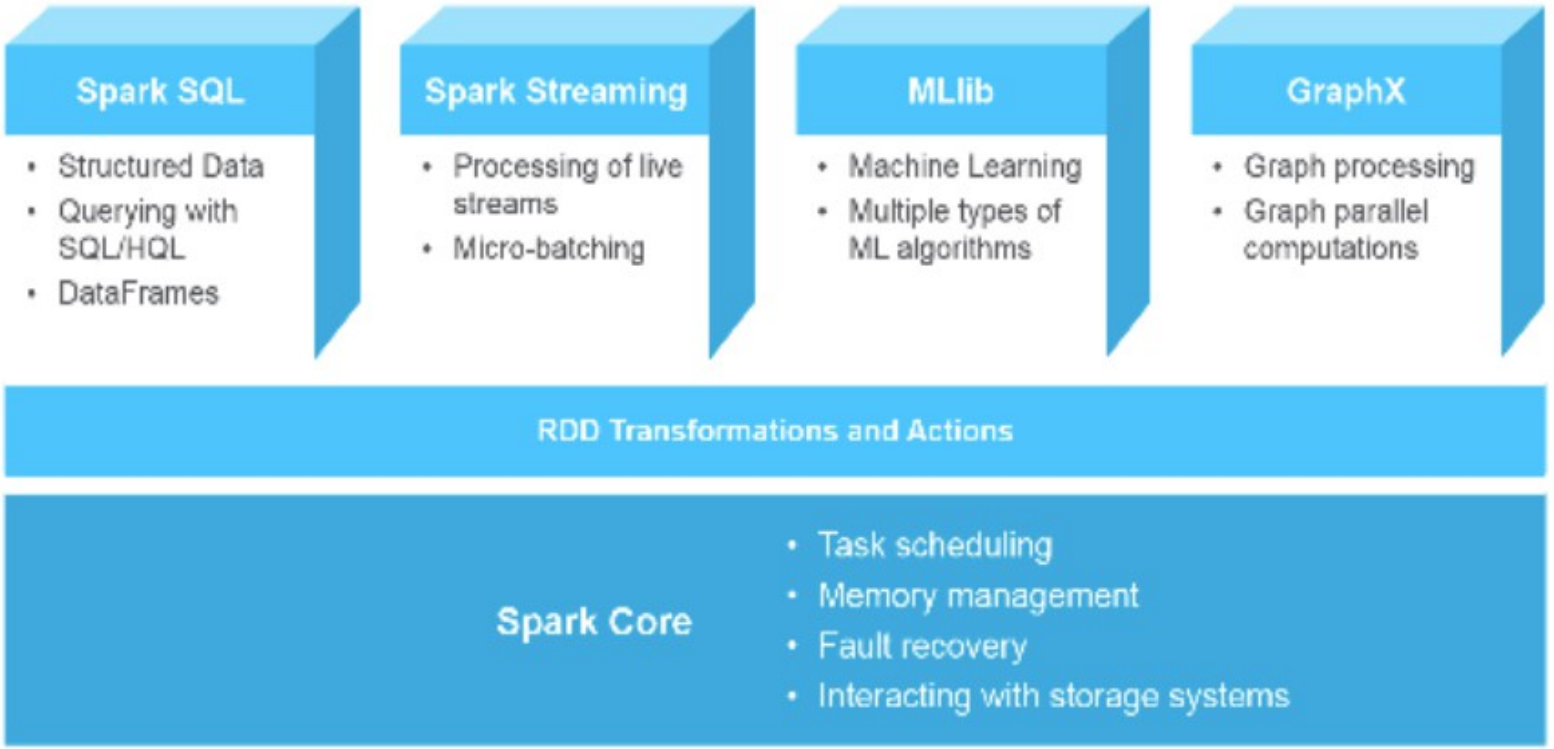

databricks
PRODUCT
SPARK
SOLUTIONS
CUSTOMERS
COMPANY
BLOG
RESOURCES
Partners
Training
Sign Up
databricks™
making big data simple
Data Science made easy, from ingest to production. Powered by Apache Spark™.
SIGN UP FOR A 14-DAY FREE TRIAL
LEARN SPARK
Join the Community Edition Beta waitlist >

Spark Platform



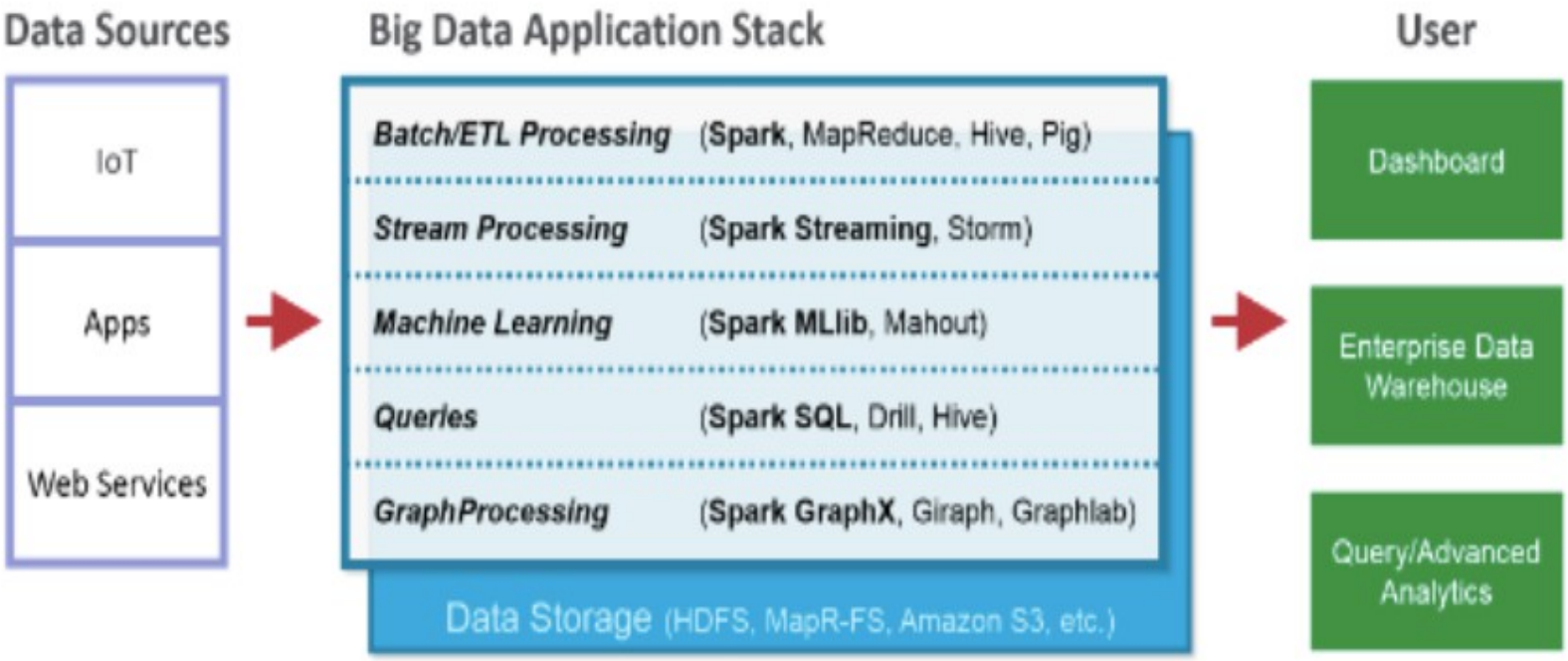

Scala
Java
python
DataFrames
ML Pipelines
Spark SQL
Spark Streaming
MLlib
GraphX
Spark Core
Data Sources
hadoop
cassandra
HBASE
{JSON}
MySQL
elasticsearch

Spark Platform



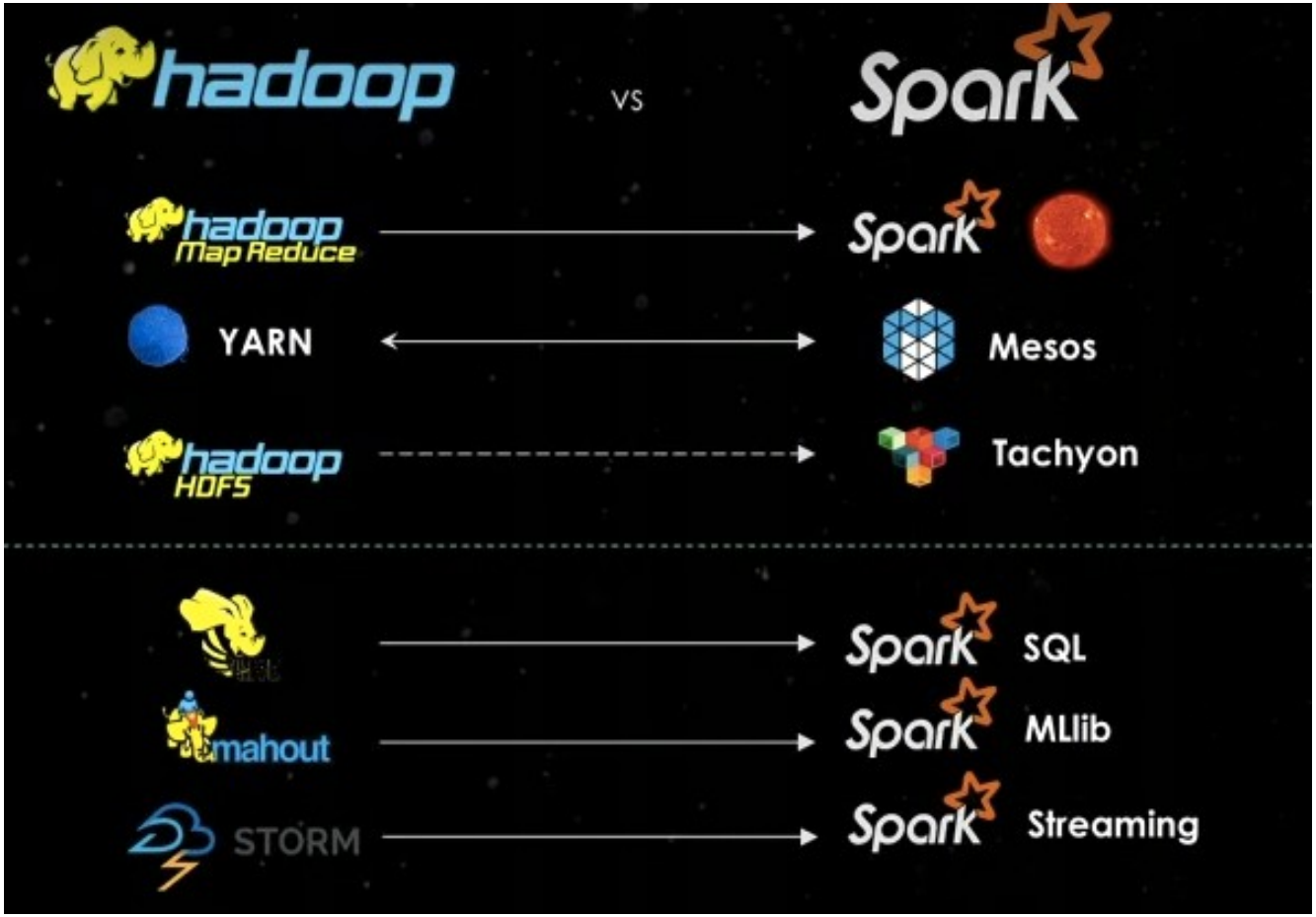
Source: MapR Academy

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

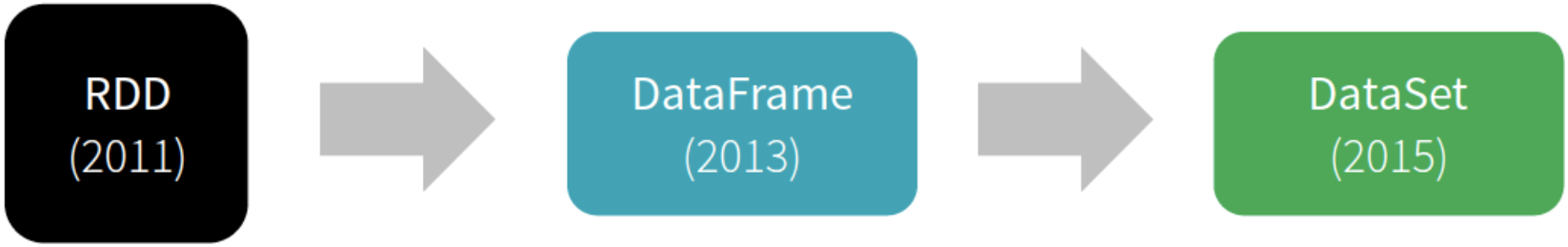
Companies That Presented at Spark Summit 2015 in San Francisco

Source: Jump start into Apache Spark and Databricks

Do we still need Hadoop?

- Yes, why Hadoop?
  - HDFS
  - YARN
  - MapReduce is mature and still be appropriate for certain workloads
  - Other services: Sqoop, Flume, etc.
- But you can still use other resource management, storages
  - Spark Standalone
  - Amazon S3
  - Mesos

History of Spark APIs



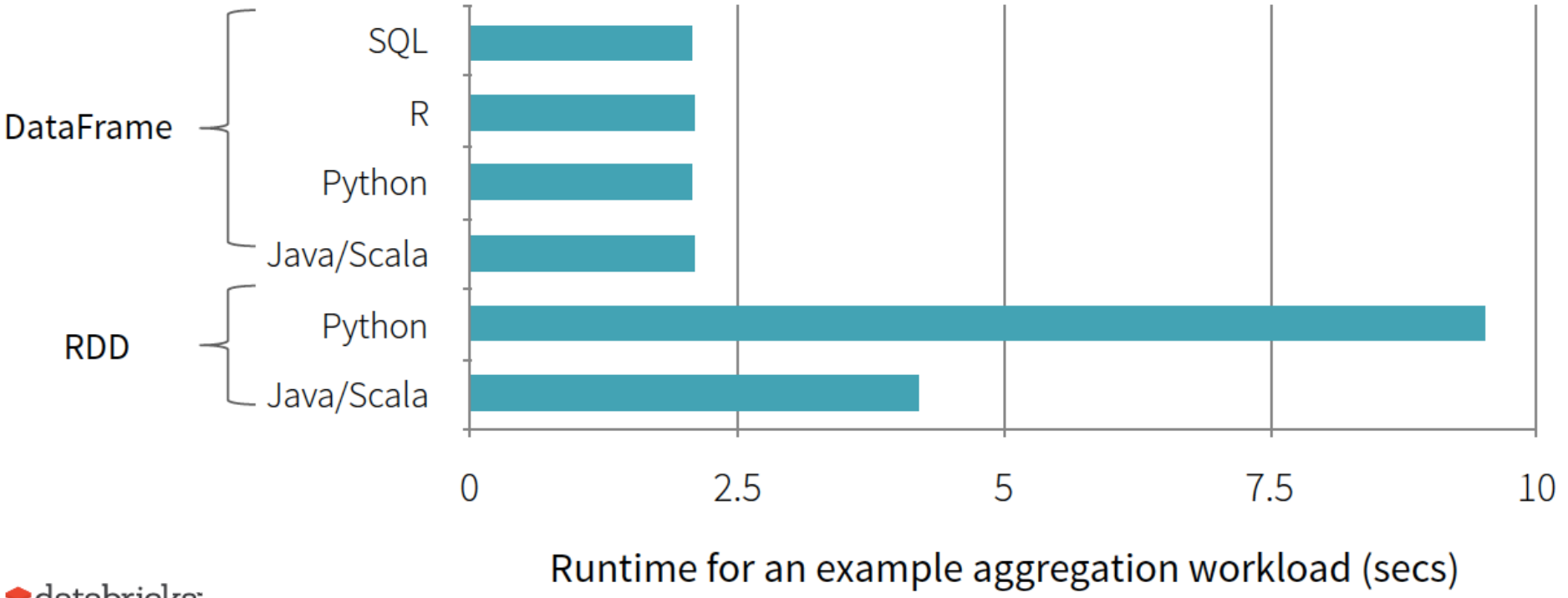
- Distribute collection of JVM objects
- Functional Operators (map, filter, etc.)

- 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



databricks

Source: Jump start into Apache Spark and Databricks

What is a RDD?

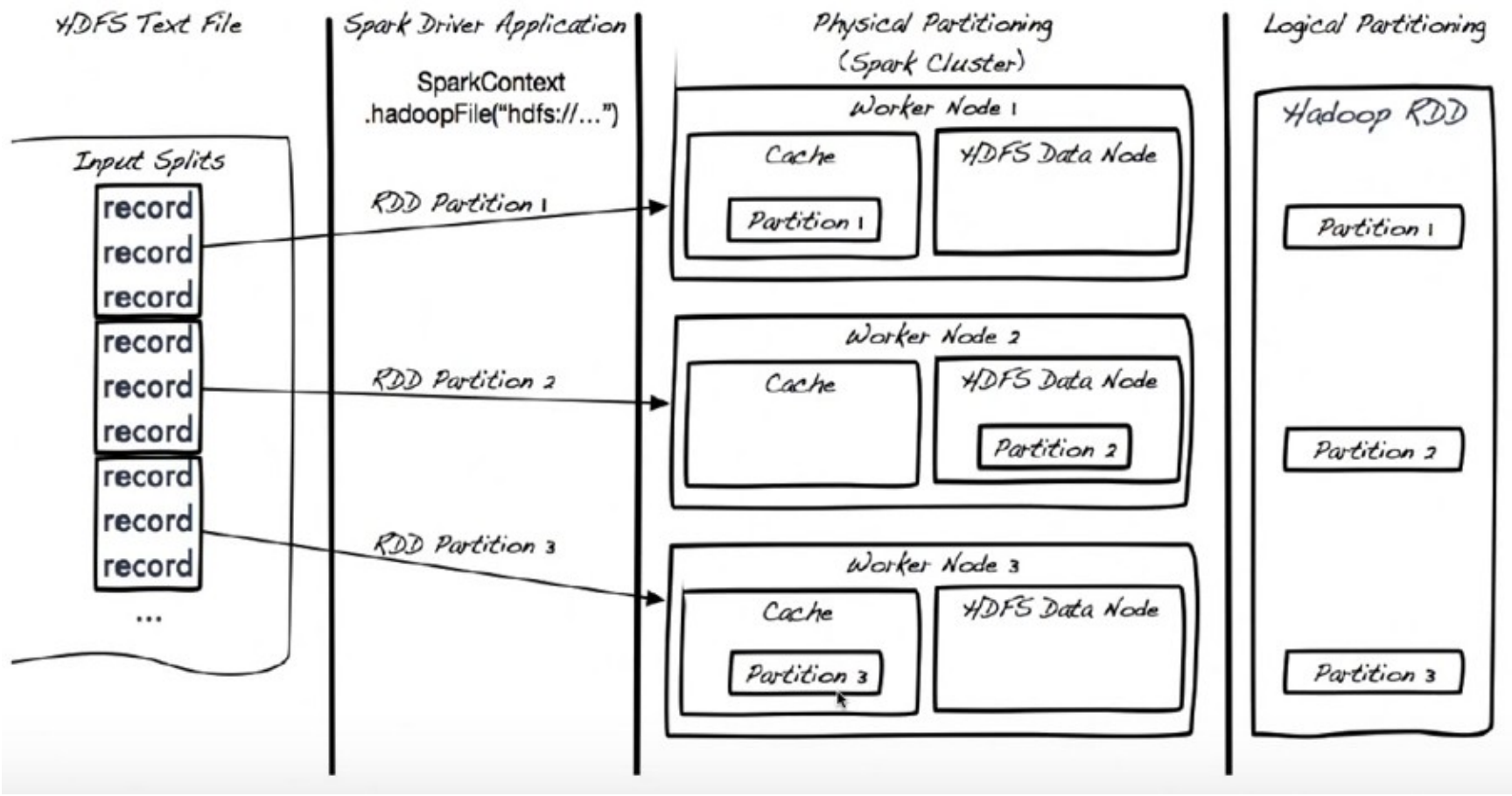
- Resilient: if the data in memory (or on a node) is lost, it can be recreated.
- Distributed: data is chucked into partitions and stored in memory acress the custer.
- Dataset: initial data can come from a table or be created programmatically

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")
```

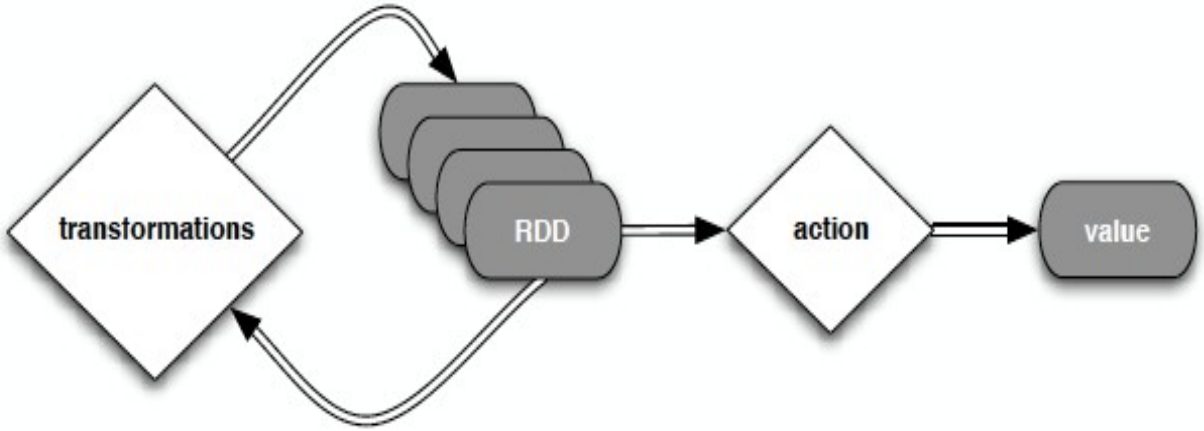


Source: Pspark: A brain-friendly introduction

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

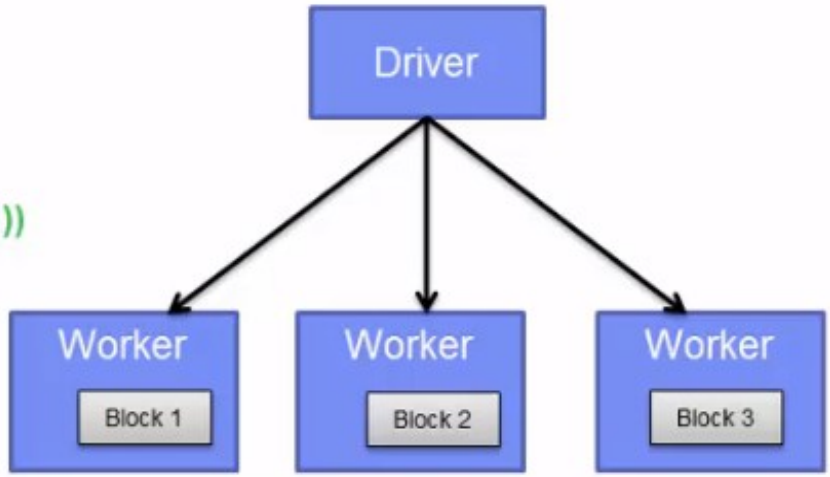
Create RDD

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()
```

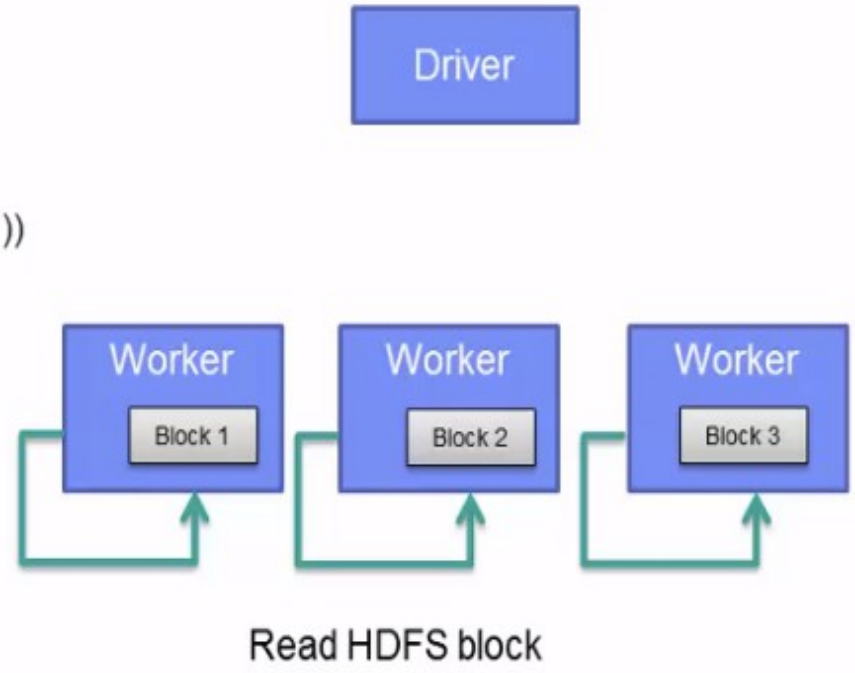


Driver sends the code to be executed on each block

Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

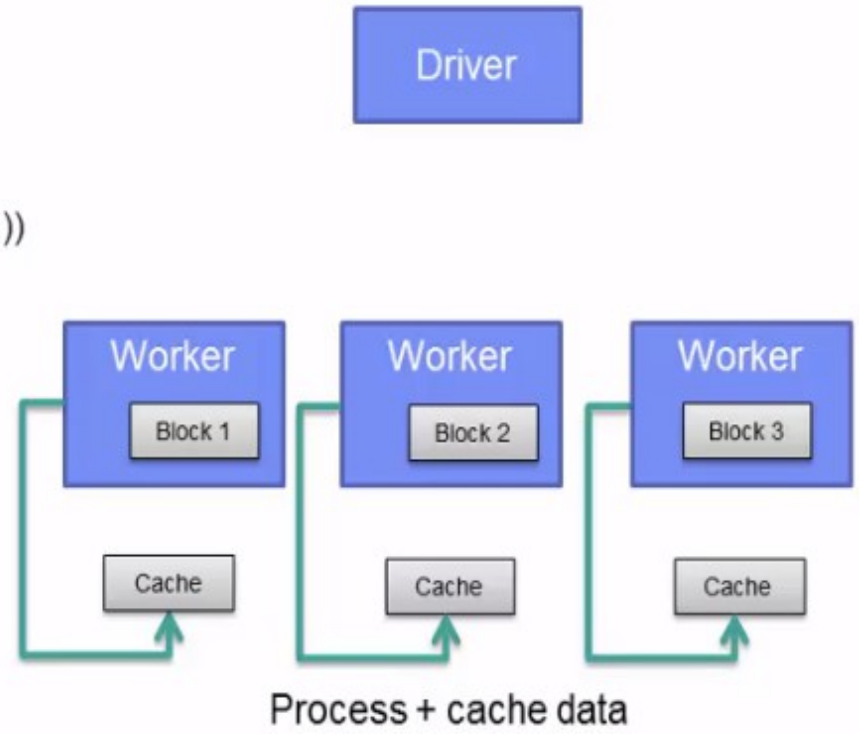
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()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



Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

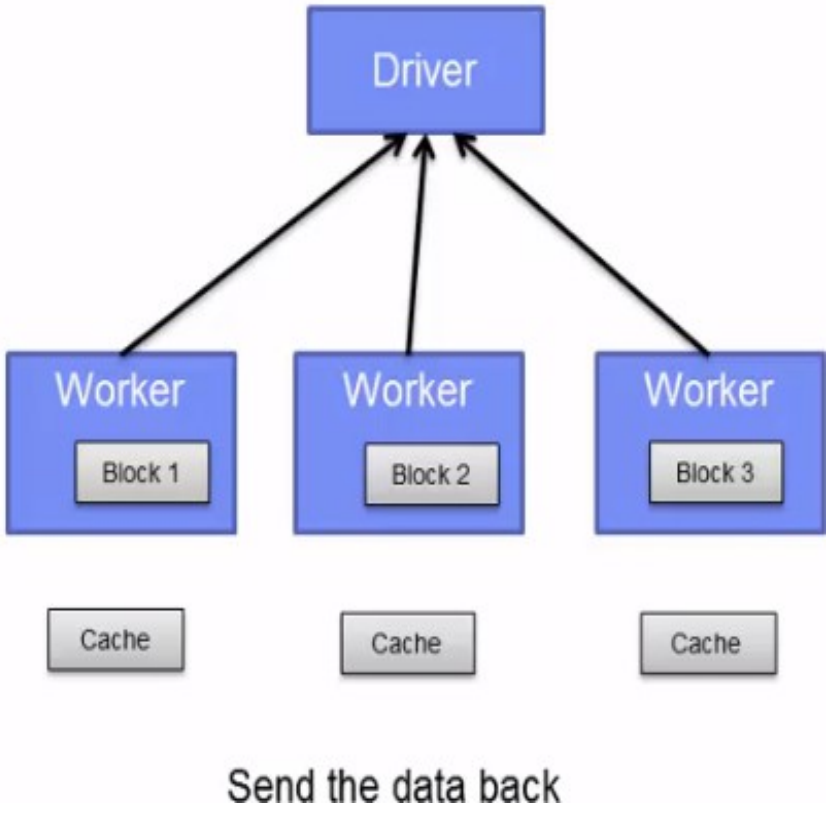
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()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



Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

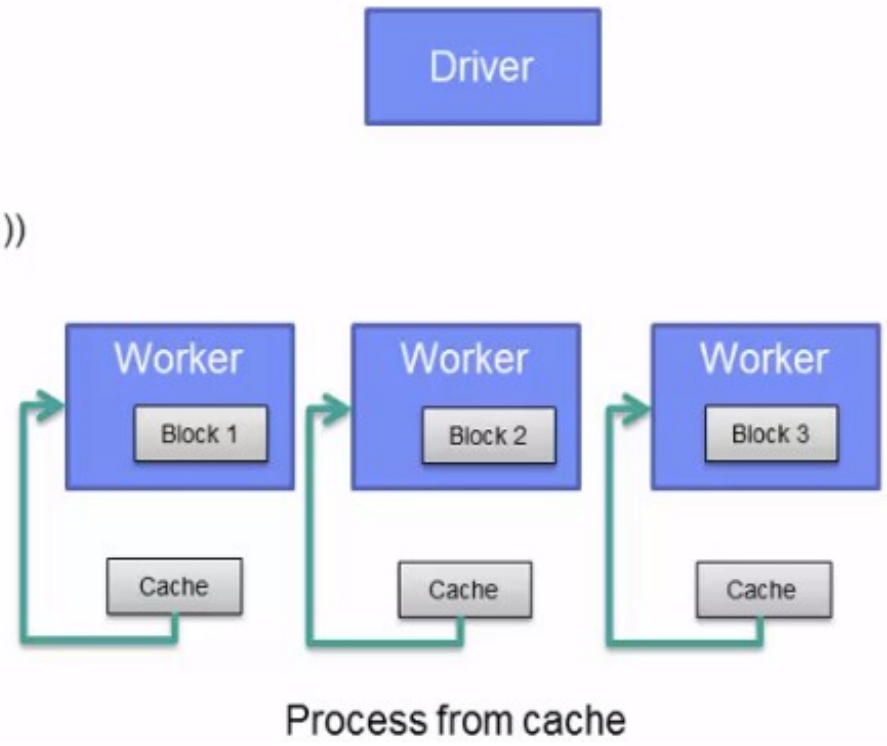
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()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



Source: Spark Fundamentals I Big Data Usibersity

What happens when an action is executed

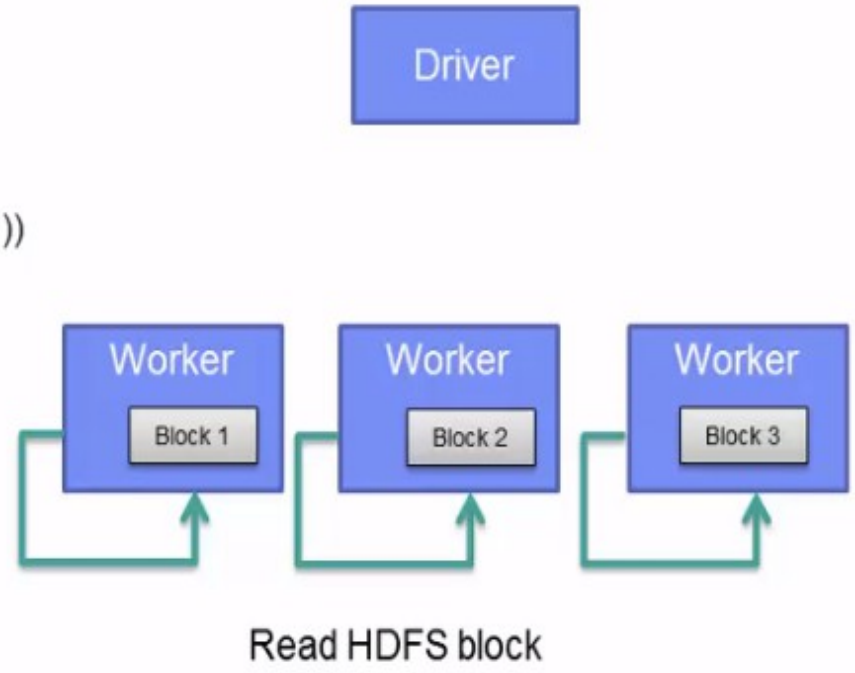
```
// Creating the RDD
val logFile = sc.textFile("hdfs://...")
// Transformations
val errors = logFile.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
//Caching
messages.cache()
// Actions
messages.filter(_.contains("mysql")).count()
messages.filter(_.contains("php")).count()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



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.

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]
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

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)]))=>

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

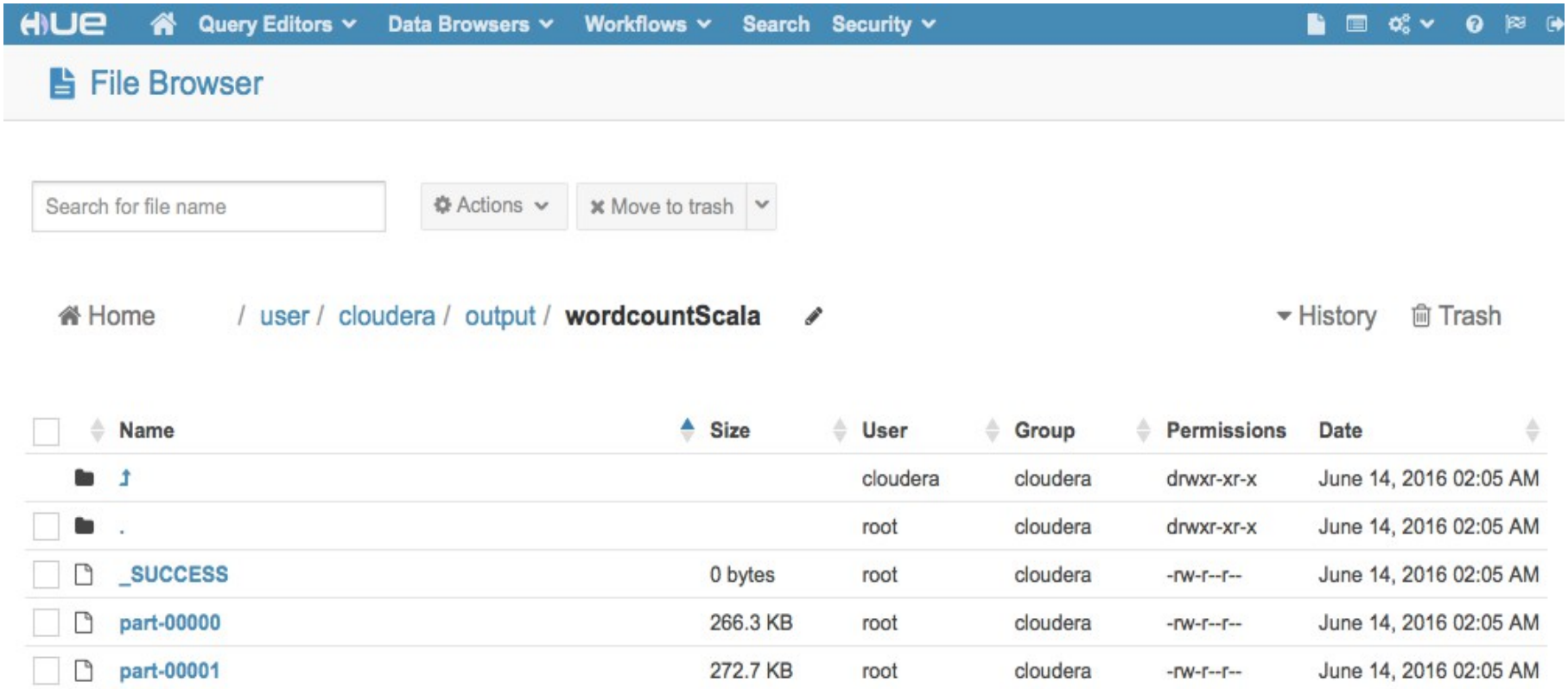
Spark-context

```
scala> sc
```

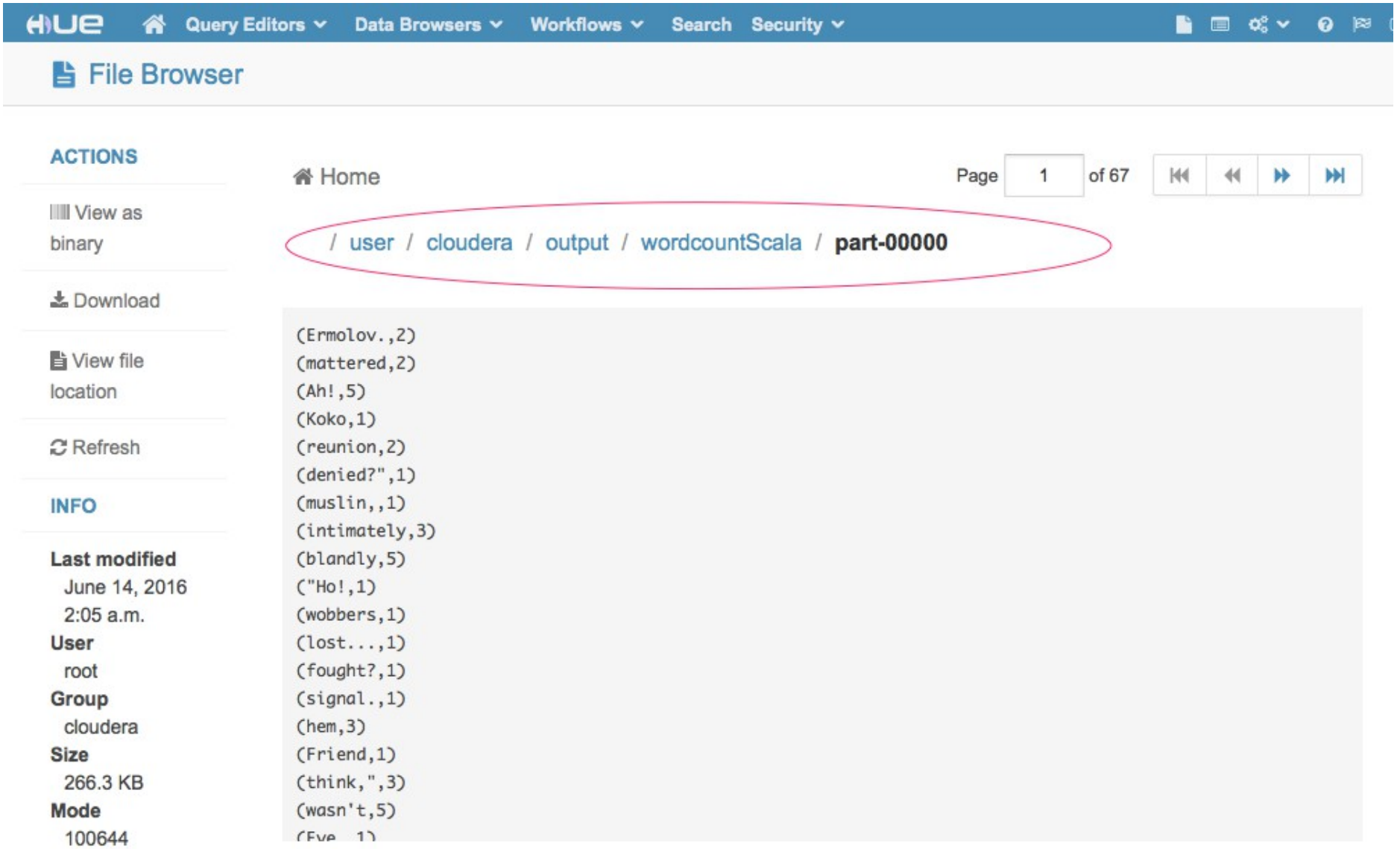
```
scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@18c07e25
```

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
")
```

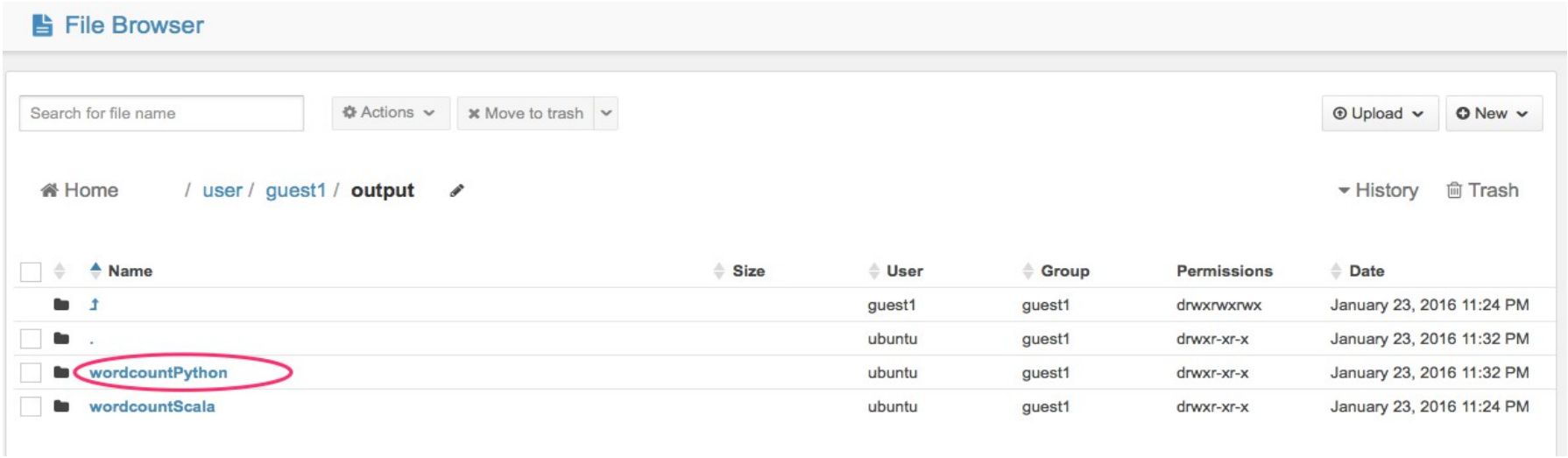

HUE
Query Editors
Data Browsers
Workflows
Search
Security
File Browser
Search for file name
Actions
Move to trash
Home / user / cloudera / output / wordcountScala
History
Trash
Name
Size
User
Group
Permissions
Date
cloudera cloudera drwxr-xr-x June 14, 2016 02:05 AM
. root cloudera drwxr-xr-x June 14, 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")
```



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”)
```

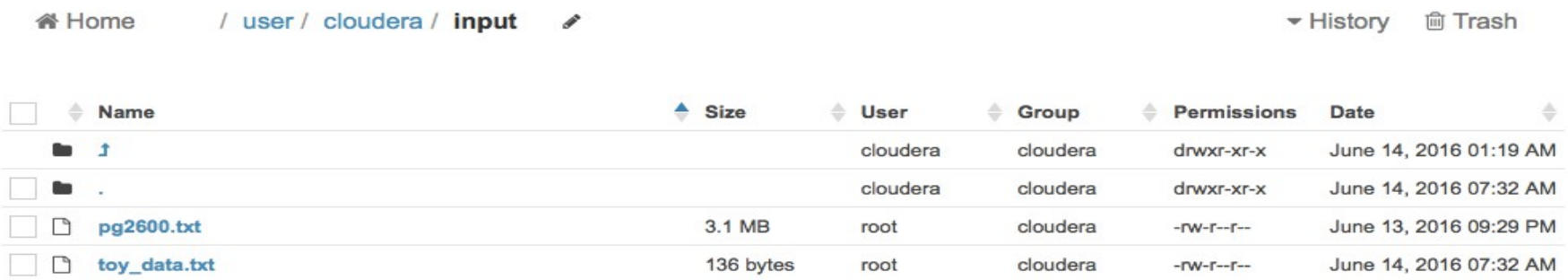
Key-Value Operations

```
>>> 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()
```

Hands-on:
Loading data from MySQL

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

MySQL RDS Server on AWS

A RDS Server is running on AWS with the following configuration

> database: imc_db

> username: admin

> password: imcinstitute

>addr: imcdb.cmw65obdqfnx.us-west-2.rds.amazonaws.com

[This address may change]

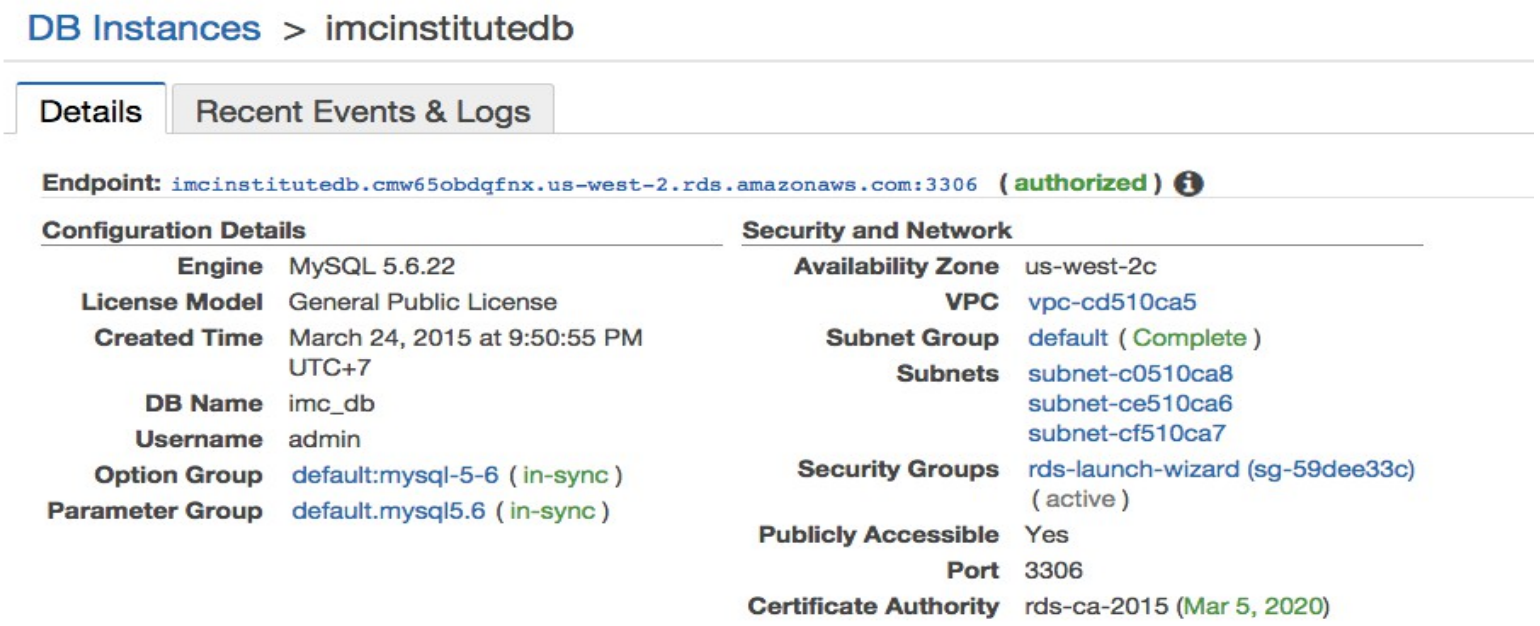


Table in MySQL RDS

Table 1 > country_tbl : Columns: id, country

Result Set Filter: Search

id	country
1	USA
2	Canada
61	Japan
66	Thailand
NULL	NULL

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

JdbcRDD

```
JdbcRDD( SparkContext, getConnection: () => Connection,
sql: String, lowerBound: Long, upperBound: Long,
numPartitions: Int,  mapRow: (ResultSet) => T =
 JdbcRDD.resultSetToObjectArray)
```

Download MySQL driver & Start Spark-shell

```
# wget http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.23/mysql-connector-java-5.1.23.jar
```

Running Spark-shell

```
#spark-shell --jars mysql-connector-java-5.1.23.jar
```

```
16/06/28 15:23:35 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
SQL context available as sqlContext.

scala>
```

Reading JDBC

```
$ scala> :paste
val url="jdbc:mysql://imcdb.cmw65obdqfnx.us-west-
2.rds.amazonaws.com:3306/imc_db"
val username = "admin"
val password = "imcinstitute"
import org.apache.spark.rdd.JdbcRDD
import java.sql.{Connection, DriverManager, ResultSet}
Class.forName("com.mysql.jdbc.Driver").newInstance
val myRDD = new JdbcRDD( sc, () =>
DriverManager.getConnection(url,username,password) ,
"SELECT * FROM country_tbl LIMIT ?, ?", 0, 5, 2, r =>
r.getString("id") + ", " + r.getString("country"))
myRDD.count
myRDD.foreach(println)
```

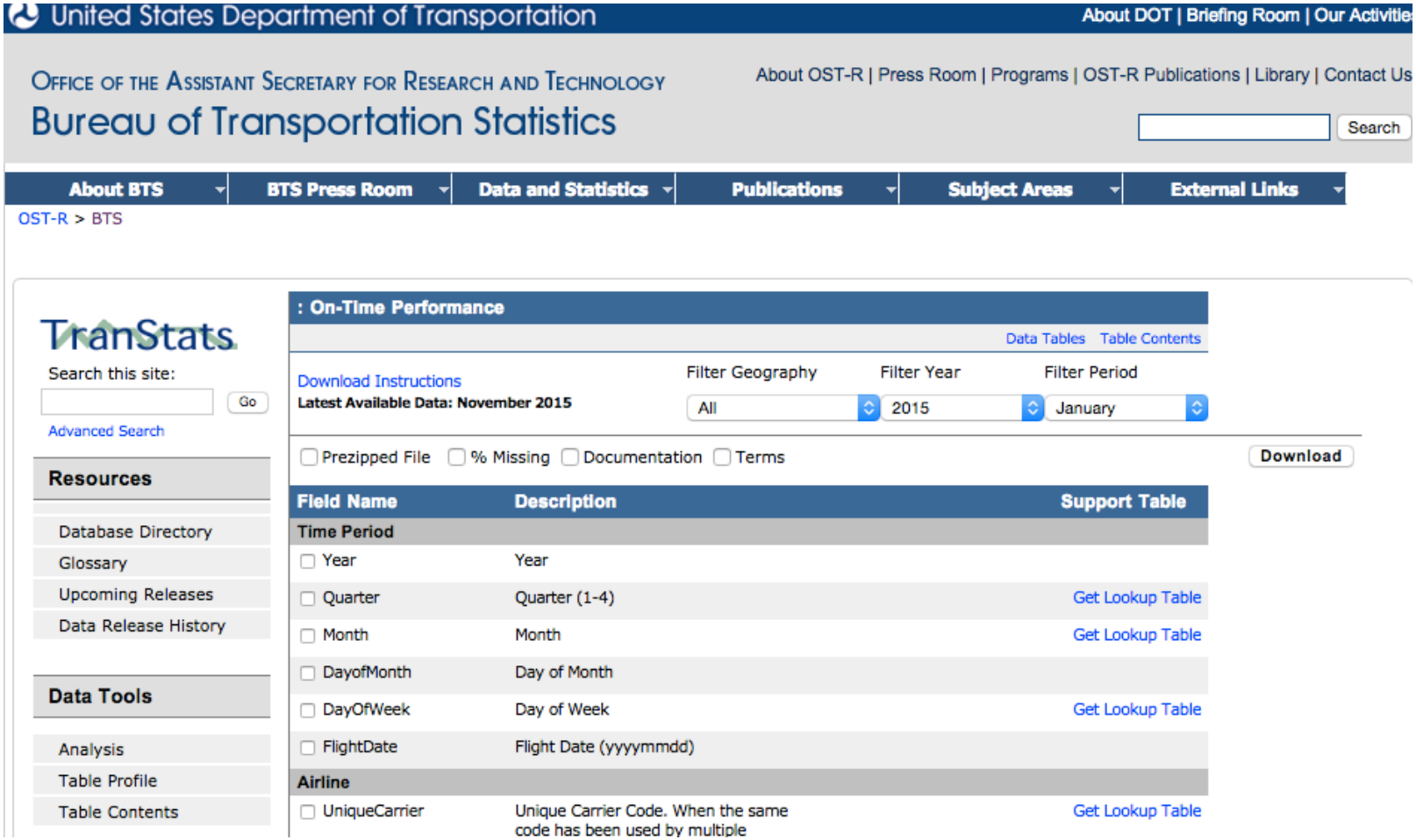
Output

```
// Exiting paste mode, now interpreting.

[Stage 5:>                                                          (0 + 2) / 2]66,
Thailand
1, USA
2, Canada
url: String = jdbc:mysql://imcdb.cmw65obdqfnx.us-west-2.rds.amazonaws.com:3306/imc_d
b
username: String = admin
password: String = imcinstitute
import org.apache.spark.rdd.JdbcRDD
import java.sql.{Connection, DriverManager, ResultSet}
myRDD: org.apache.spark.rdd.JdbcRDD[String] = JdbcRDD[3] at JdbcRDD at <console>:43
```

Project: Flight

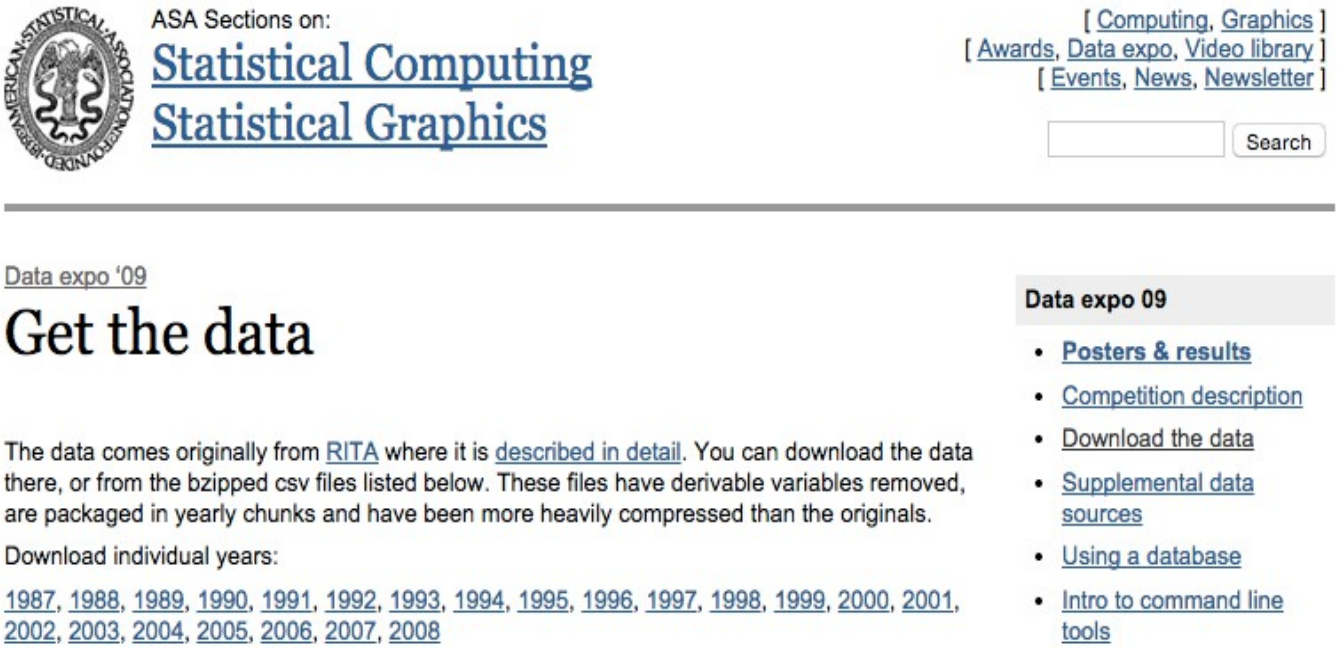
Flight Details Data

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

	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/

FiveThirtyEight INTERACTIVE

Leaving From ⇆ Flying To

ROUTE PERFORMANCE
- Fast
- Medium
- Slow

FLIGHT VOLUME
- 1,000 flights
- 10,000

The Busiest Airports

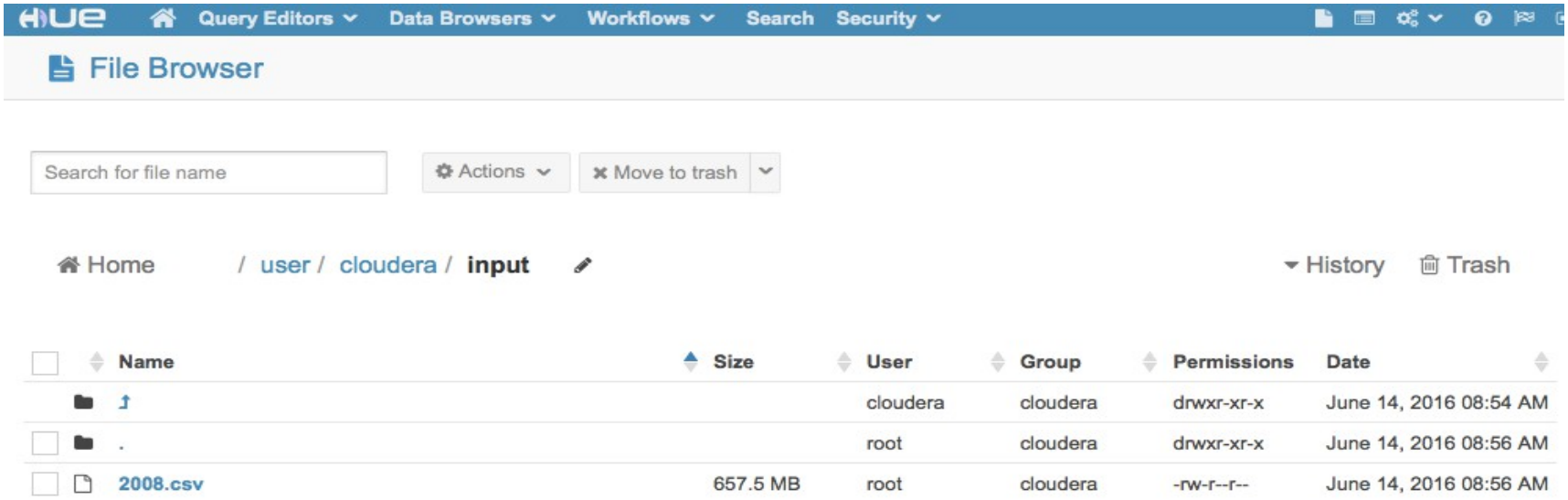
How much time each of the country's busiest airports typically adds to a trip. Click for details, or find an airport or route to the left.

	TIME ADDED DEP	ARR
Atlanta ATL	+0:04	+0:04
Baltimore BWI	+0:10	+0:06
Boston BOS	+0:16	+0:08
Charlotte CLT	+0:06	+0:05
Chicago ORD	+0:12	+0:16
Dallas/Fort Worth DFW	+0:09	+0:14
Denver DEN	+0:02	+0:05
Detroit DTW	+0:07	+0:07
Houston IAH	+0:05	+0:06
Las Vegas LAS	+0:03	+0:01
Los Angeles LAX	+0:05	+0:07
Minneapolis MSP	+0:03	+0:04
New York LGA	+0:30	+0:27
New York JFK	+0:24	+0:20
Newark EWR	+0:19	+0:19

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)
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Spark Program : Define convert_float function

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


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Spark Program : Finding best/worst airline

```
>>> carrier_rdd = airline_rows.map(lambda row:
(row['UniqueCarrier'],convert_float(row['ArrDelay'])))
>>> carrier_rdd.take(2)
```

```
16/06/17 17:48:06 INFO scheduler.DAGScheduler: Job 5 finished: runJob at PythonRDD.s
cala:393, took 0.062345 s
[(u'WN', -14.0), (u'WN', 2.0)]
```

Spark Program : Finding best/worst airlines

```
>>> mean_delays_dest =
carrier_rdd.groupByKey().mapValues(lambda delays:
sum(delays.data)/len(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)
```

```
[(u'AQ', -2.8708974358974357), (u'HA', 1.2518519716624075), (u'US', 2.800998260539828), (u'9E', 3.9874908469611912), (u'AS', 4.721360405553864), (u'WN', 5.1157033802259031), (u'F9', 6.0841356696810847), (u'OO', 6.4389386397817896), (u'NW', 7.2934658796727758), (u'DL', 7.7161646357519178)]
```

```
[(u'AA', 12.202853434950445), (u'OH', 11.404110178283158), (u'YV', 11.322566979170753), (u'UA', 11.001550560048052), (u'B6', 10.859381613638567), (u'CO', 10.809820575966226), (u'XE', 10.320298523403915), (u'EV', 10.00033146217589), (u'MQ', 9.4969706109522658), (u'FL', 8.98815747237125 61)]
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Spark SQL

DataFrame

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

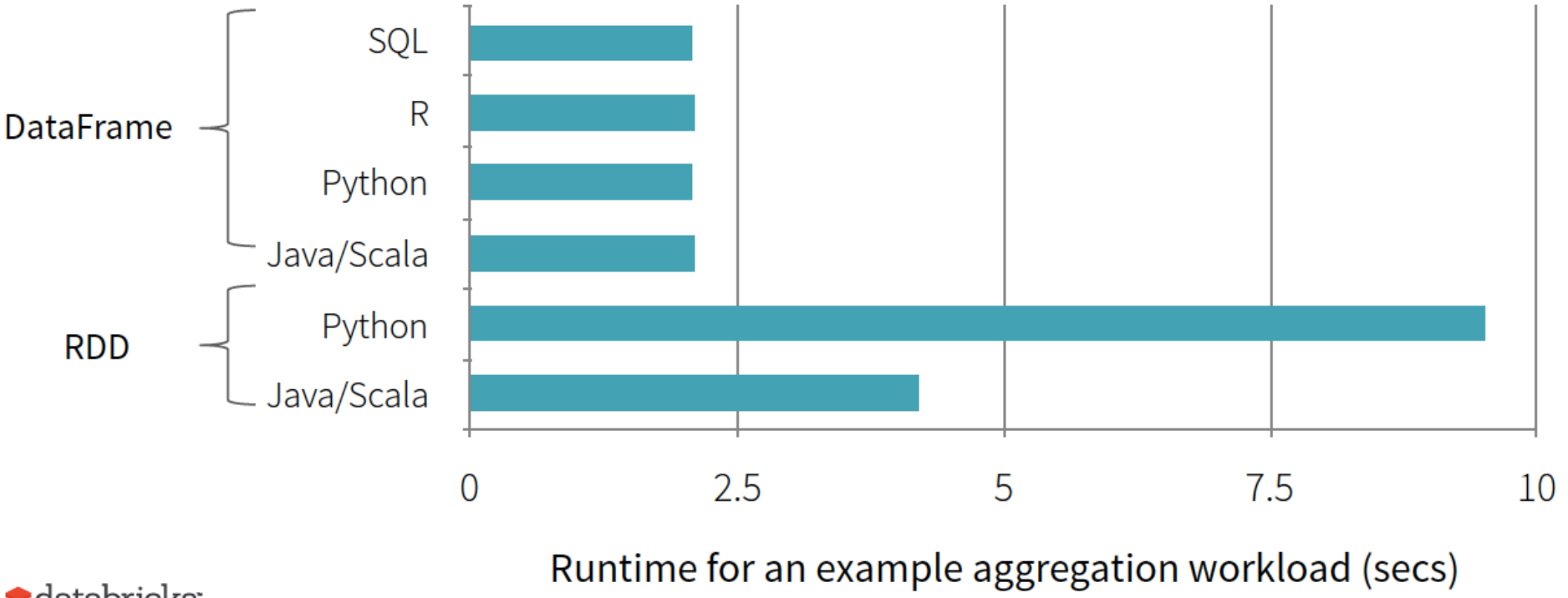
thanachart@imcinstitute.com

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

SparkSQL can leverage the Hive metastore

- Hive Metastore can also be leveraged by a wide array of applications
  - Spark
  - Hive
  - Impala
- Available from HiveContext

thanachart@imcinstitute.com

SparkSQL

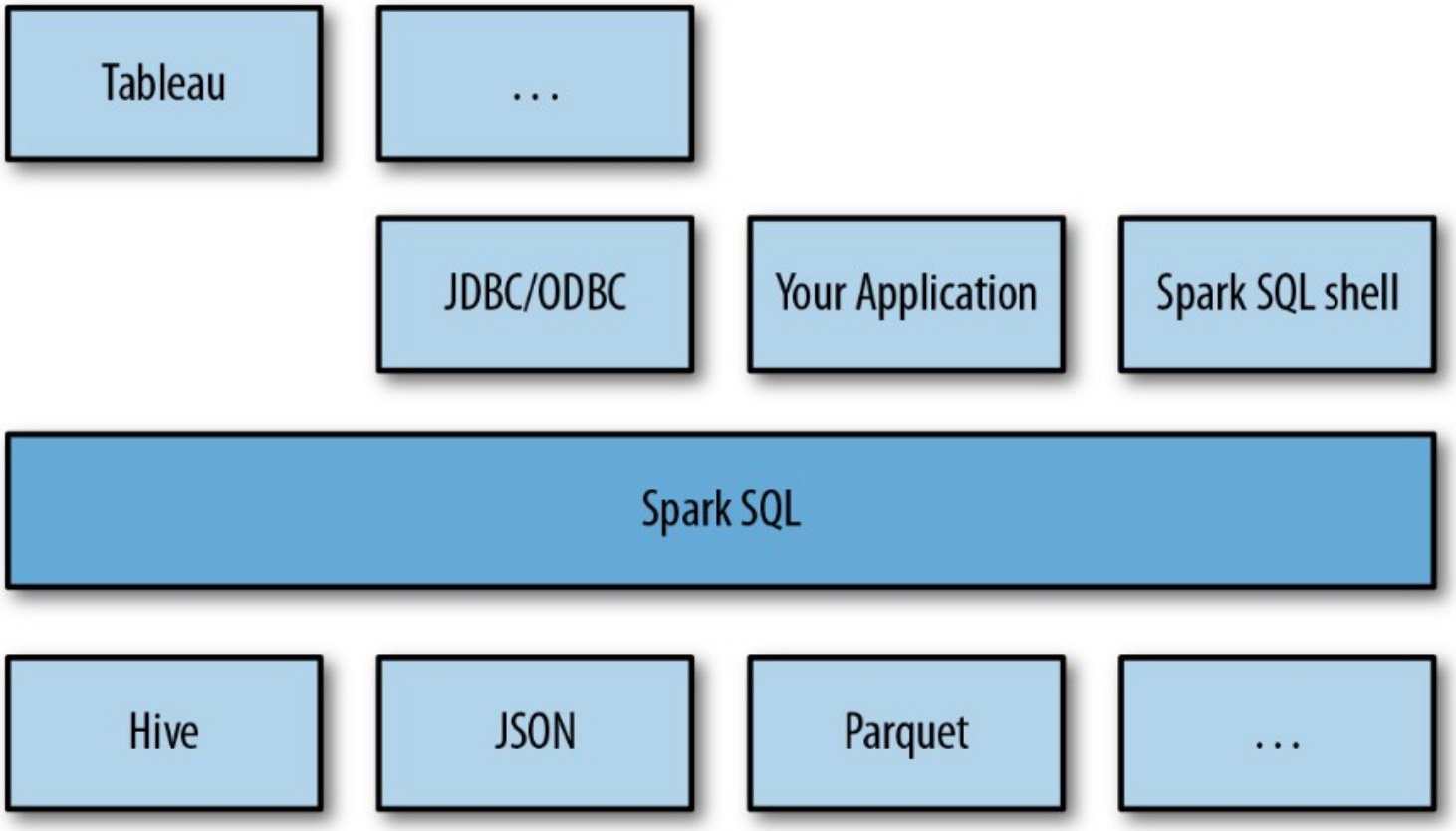
```
context = ps.HiveContext(sc)

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

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

Unified interface for structured data

amplab UC BERKELEY databricks

Image credit: http://barrymieny.deviantart.com/

Spark SQL usage



Source: Learning Spark, O'Reilly Media, Inc.

Hands-on: Spark SQL

Link Hive Metastore with Spark-Shell

```
scala > val sqlContext = new
org.apache.spark.sql.hive.HiveContext(sc)

scala> sqlContext.sql("CREATE TABLE IF NOT EXISTS movie(userid
STRING, movieid STRING, rating INT, timestamp STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY
'\n'")

scala> sqlContext.sql("LOAD DATA LOCAL INPATH '/root/ml-
100k/u.data' INTO TABLE movie")

scala> val result = sqlContext.sql("SELECT * FROM movie")

scala> result.show()
```

Link Hive Metastore with Spark-Shell

Copy the configuration file

```
$cp   /usr/lib/hive/conf/hive-site.xml    /usr/lib/spark/conf/
```

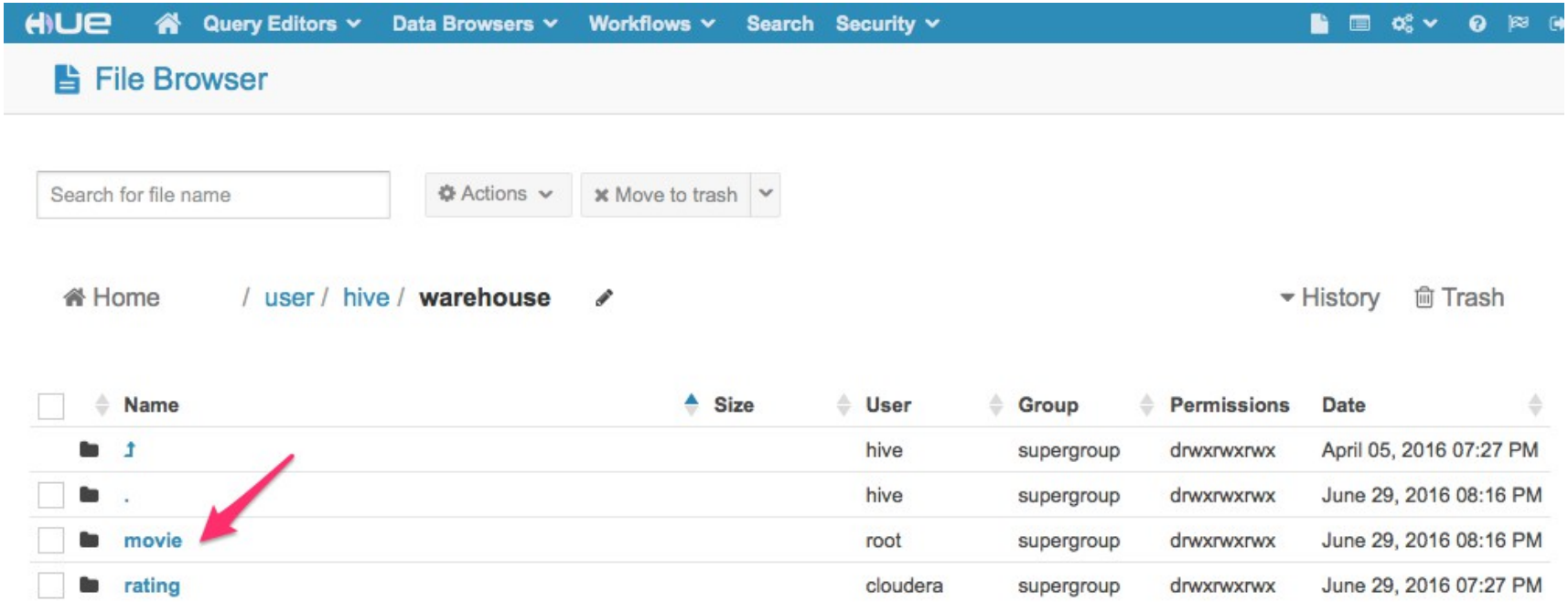
Running Spark

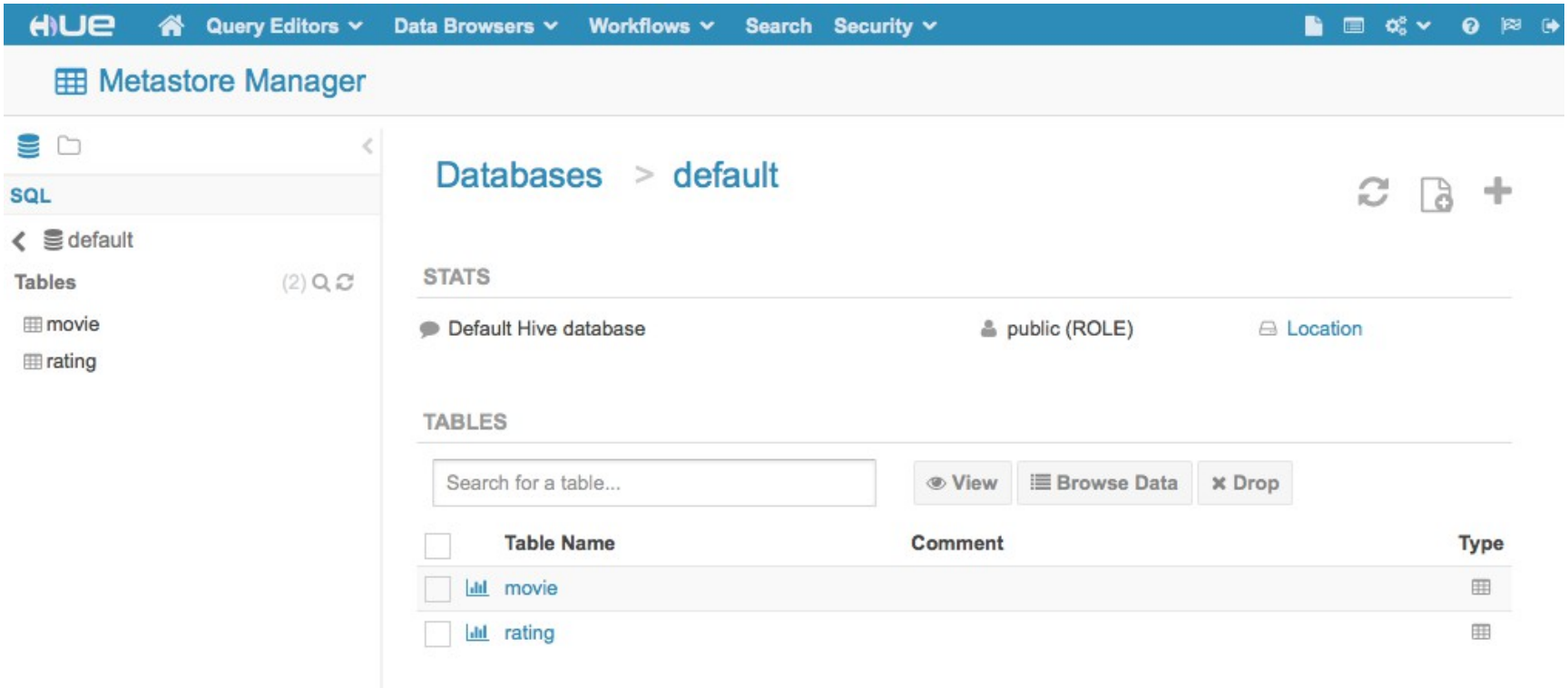
```
$spark-shell
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Exercise 1: HiveContext

```
scala > val sqlContext = new
org.apache.spark.sql.hive.HiveContext(sc)

scala> sqlContext.sql("CREATE TABLE IF NOT EXISTS movie(userid
STRING, movieid STRING, rating INT, timestamp STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY
'\n'")

scala> sqlContext.sql("LOAD DATA LOCAL INPATH '/root/ml-
100k/u.data' INTO TABLE movie")

scala> val result = sqlContext.sql("SELECT * FROM movie")

scala> result.show()
```


HUE
Query Editors
Data Browsers
Workflows
Search
Security
File Browser
Search for file name
Actions
Move to trash
Home / user / hive / warehouse
History
Trash
Name
Size
User
Group
Permissions
Date
hive supergroup drwxrwxrwx April 05, 2016 07:27 PM
. hive supergroup drwxrwxrwx June 29, 2016 08:16 PM
movie root supergroup drwxrwxrwx June 29, 2016 08:16 PM
rating cloudera supergroup drwxrwxrwx June 29, 2016 07:27 PM


HUE
Query Editors
Data Browsers
Workflows
Search
Security
Metastore Manager
SQL
default
Tables (2)
movie
rating
Databases > default
STATS
Default Hive database
public (ROLE)
Location
TABLES
Search for a table...
View
Browse Data
Drop
Table Name
Comment
Type
movie
rating

Exercise 2: JSON Data

Upload a data to HDFS

```
$ wget https://s3.amazonaws.com/imcbucket/data/hue.json
$ hadoop fs -put hue.json /user/cloudera
```

Spark SQL : Preparing data

```
>>> from pyspark.sql import HiveContext
>>> hiveCtx = HiveContext(sc)
>>> input =
hiveCtx.read.json("hdfs:///user/cloudera/hue.json")
>>> input.registerTempTable("testTable")
>>> input.printSchema()
```

```
root
 |-- fields: struct (nullable = true)
 |     |-- action: string (nullable = true)
 |     |-- app: string (nullable = true)
 |     |-- app_label: string (nullable = true)
 |     |-- app_name: string (nullable = true)
 |     |-- applied: string (nullable = true)
 |     |-- codename: string (nullable = true)
 |     |-- content_type: long (nullable = true)
 |     |-- description: string (nullable = true)
 |     |-- domain: string (nullable = true)
 |     |-- group: long (nullable = true)
 |     |-- hue_permission: long (nullable = true)
 |     |-- migration: string (nullable = true)
 |     |-- model: string (nullable = true)
 |     |-- name: string (nullable = true)
 |     |-- permissions: array (nullable = true)
 |     |     |-- element: string (containsNull = true)
 |-- model: string (nullable = true)
 |-- pk: long (nullable = true)
```

Spark SQL : Query Data

```
>>> hiveCtx.sql("SELECT * FROM testTable").show()
```

```
+--------------------+--------------------+---+
|              fields|               model| pk|
+--------------------+--------------------+---+
|[null,null,auth,n...|contenttypes.cont...|  1|
|[null,null,auth,n...|contenttypes.cont...|  2|
|[null,null,auth,n...|contenttypes.cont...|  3|
|[null,null,django...|contenttypes.cont...|  4|
|[null,null,django...|contenttypes.cont...|  5|
|[null,null,django...|contenttypes.cont...|  6|
|[null,null,conten...|contenttypes.cont...|  7|
|[null,null,sessio...|contenttypes.cont...|  8|
|[null,null,sites,...|contenttypes.cont...|  9|
|[null,null,admin,...|contenttypes.cont...| 10|
|[null,null,south,...|contenttypes.cont...| 11|
```

Spark SQL : Query Data

```
>>> hiveCtx.sql("SELECT * FROM testTable").collect()
```

```
auth.permission', pk=127), Row(fields=Row(action=None, app=None, app_label=None, app
_name=None, applied=None, codename=u'change_dataoutput', content_type=44, descriptio
n=None, domain=None, group=None, hue_permission=None, migration=None, model=None, na
me=u'Can change data output', permissions=None), model=u'auth.permission', pk=128),
Row(fields=Row(action=None, app=None, app_label=None, app_name=None, applied=None, c
odename=u'delete_dataoutput', content_type=44, description=None, domain=None, group=
None, hue_permission=None, migration=None, model=None, name=u'Can delete data output
', permissions=None), model=u'auth.permission', pk=129), Row(fields=Row(action=None,
 app=None, app_label=None, app_name=None, applied=None, codename=u'add_bundledcoordi
nator', content_type=45, description=None, domain=None, group=None, hue_permission=N
one, migration=None, model=None, name=u'Can add bundled coordinator', permissions=No
ne), model=u'auth.permission', pk=130), Row(fields=Row(action=None, app=None, app_la
bel=None, app_name=None, applied=None, codename=u'change_bundledcoordinator', conten
t_type=45, description=None, domain=None, group=None, hue_permission=None, migration
=None, model=None, name=u'Can change bundled coordinator', permissions=None), model=
```

Exercise 3 SQL Spark MovieLens (csv data)

Upload a data to HDFS then

```
$ pyspark --packages com.databricks:spark-csv_2.10:1.2.0
```

```
>>> df =
sqlContext.read.format('com.databricks.spark.csv').options(hea
der='false').load('hdfs:///user/cloudera/u.user')
```

```
>>> df.registerTempTable('user')
```

```
>>> sqlContext.sql("SELECT * FROM user").show()
```

```
+--------------------+
|                  C0|
+--------------------+
|1|24|M|technician...|
|  2|53|F|other|94043|
| 3|23|M|writer|32067|
|4|24|M|technician...|
|  5|33|F|other|15213|
|6|42|M|executive|...|
|7|57|M|administra...|
|8|36|M|administra...|
|9|29|M|student|01002|
|10|53|M|lawyer|90703|
| 11|39|F|other|30329|
| 12|28|F|other|06405|
|13|47|M|educator|...|
|14|45|M|scientist...|
|15|49|F|educator|...|
|16|21|M|entertain...|
|17|30|M|programme...|
| 18|35|F|other|37212|
|19|40|M|librarian...|
|20|42|F|homemaker...|
+--------------------+
only showing top 20 rows
```

Exercise 4: Spark SQL Meals 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)
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

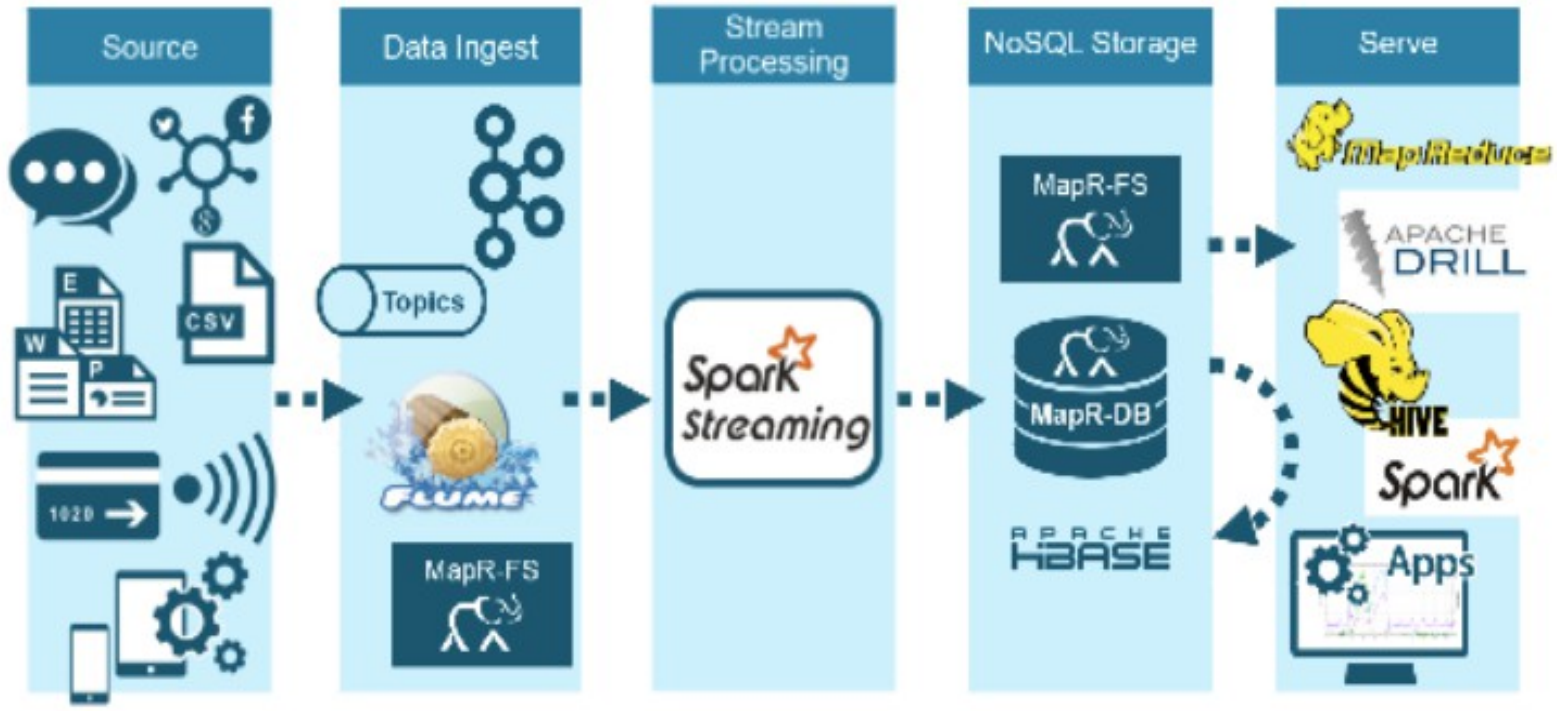
Spark SQL : More complex query

```
>>> 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)]
```

Spark Streaming

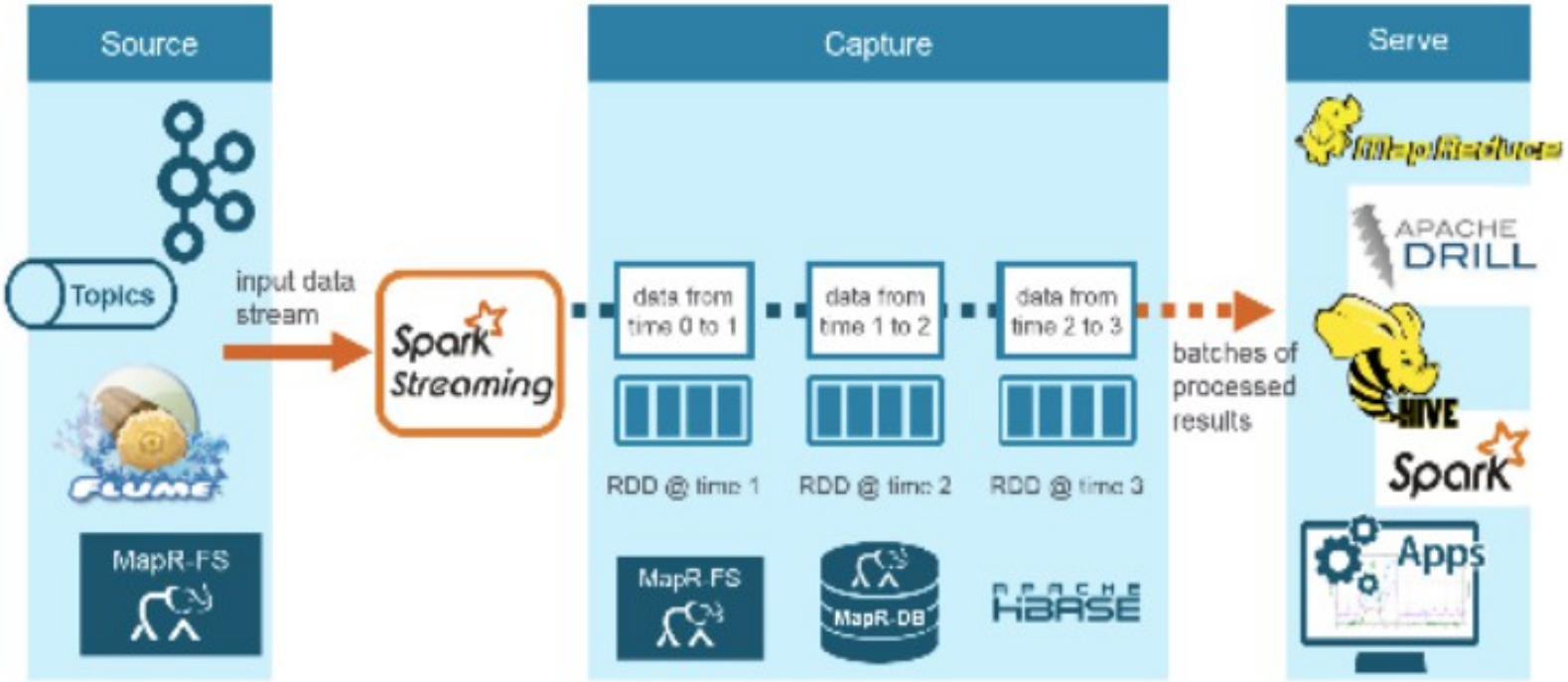
Stream Process Architecture



Source: MapR Academy

thanachart@imcinstitute.com

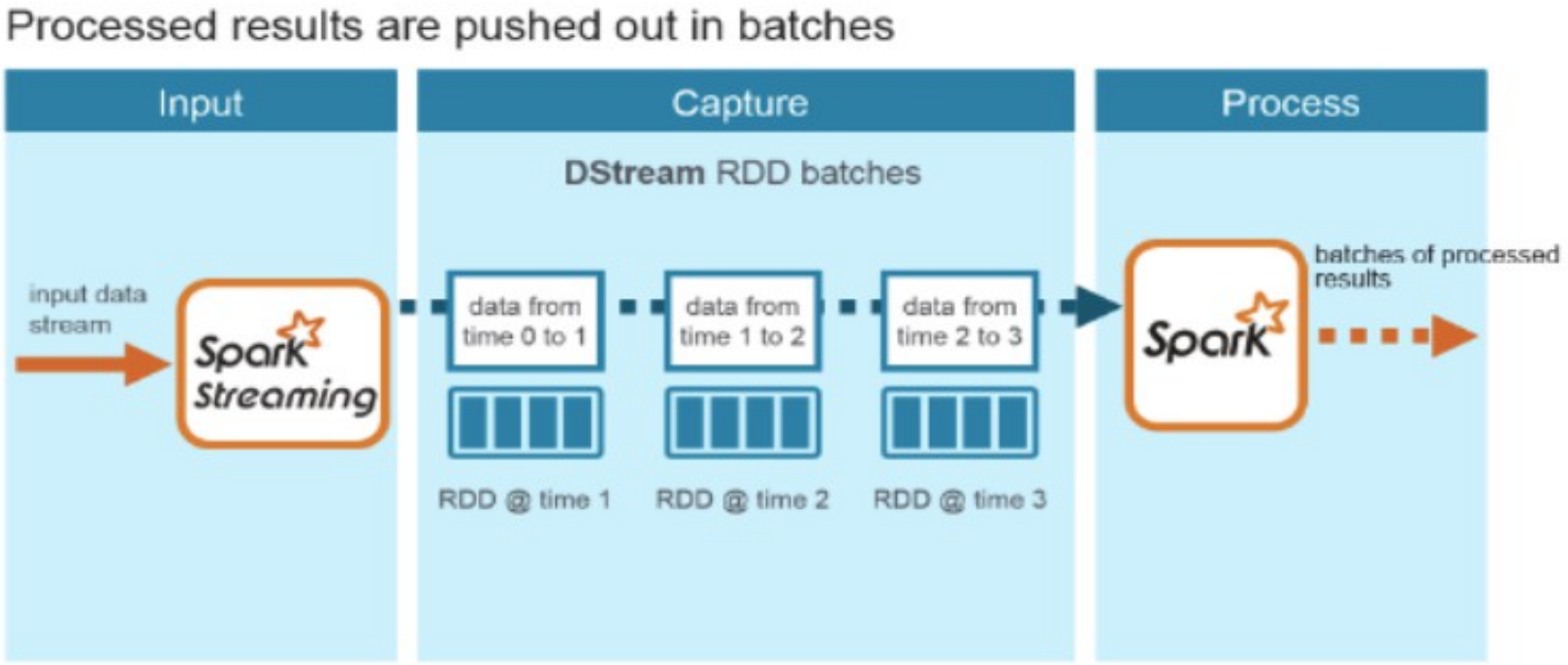
Spark Streaming Architecture



Source: MapR Academy

thanachart@imcinstitute.com

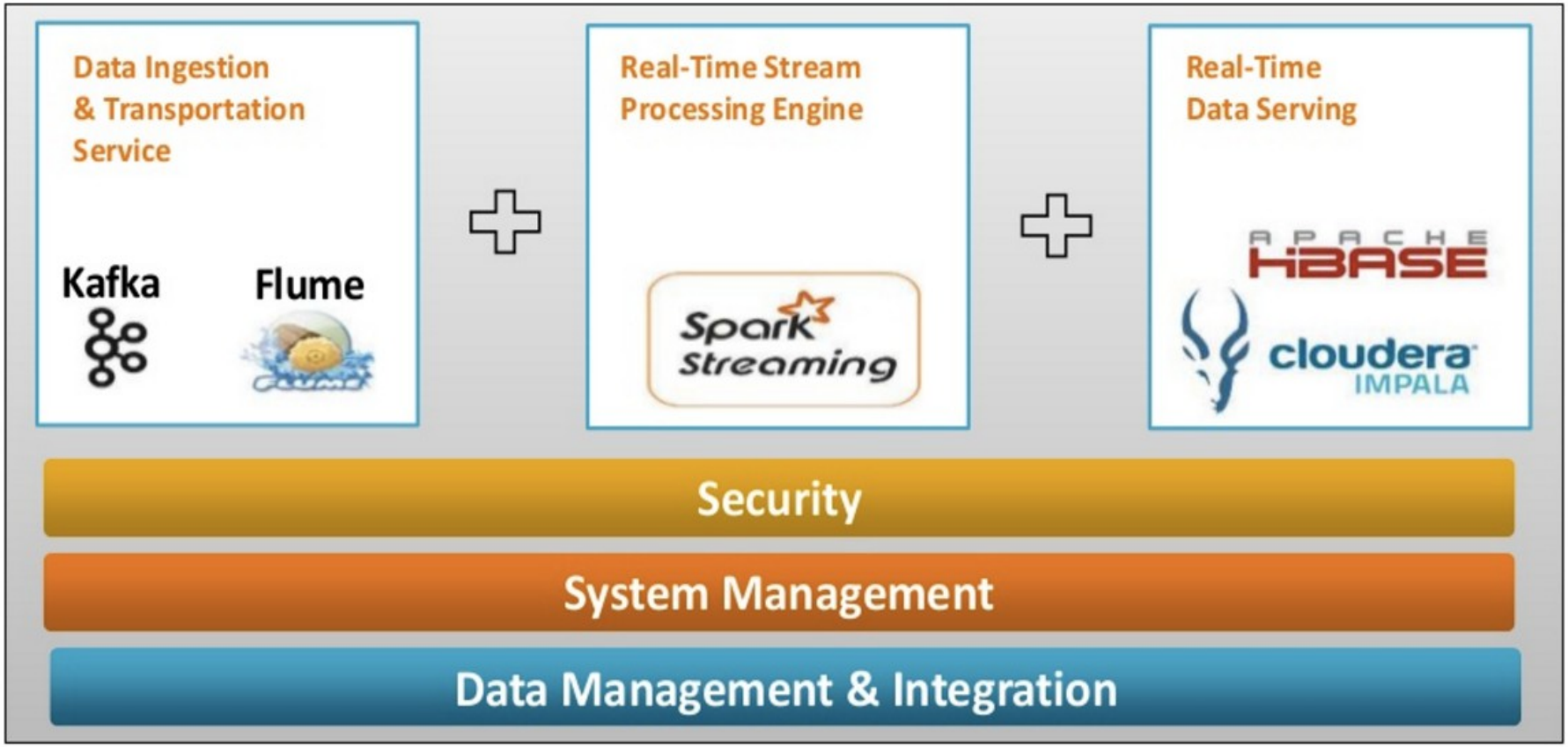
Processing Spark DStreams



Source: MapR Academy

thanachart@imcinstitute.com

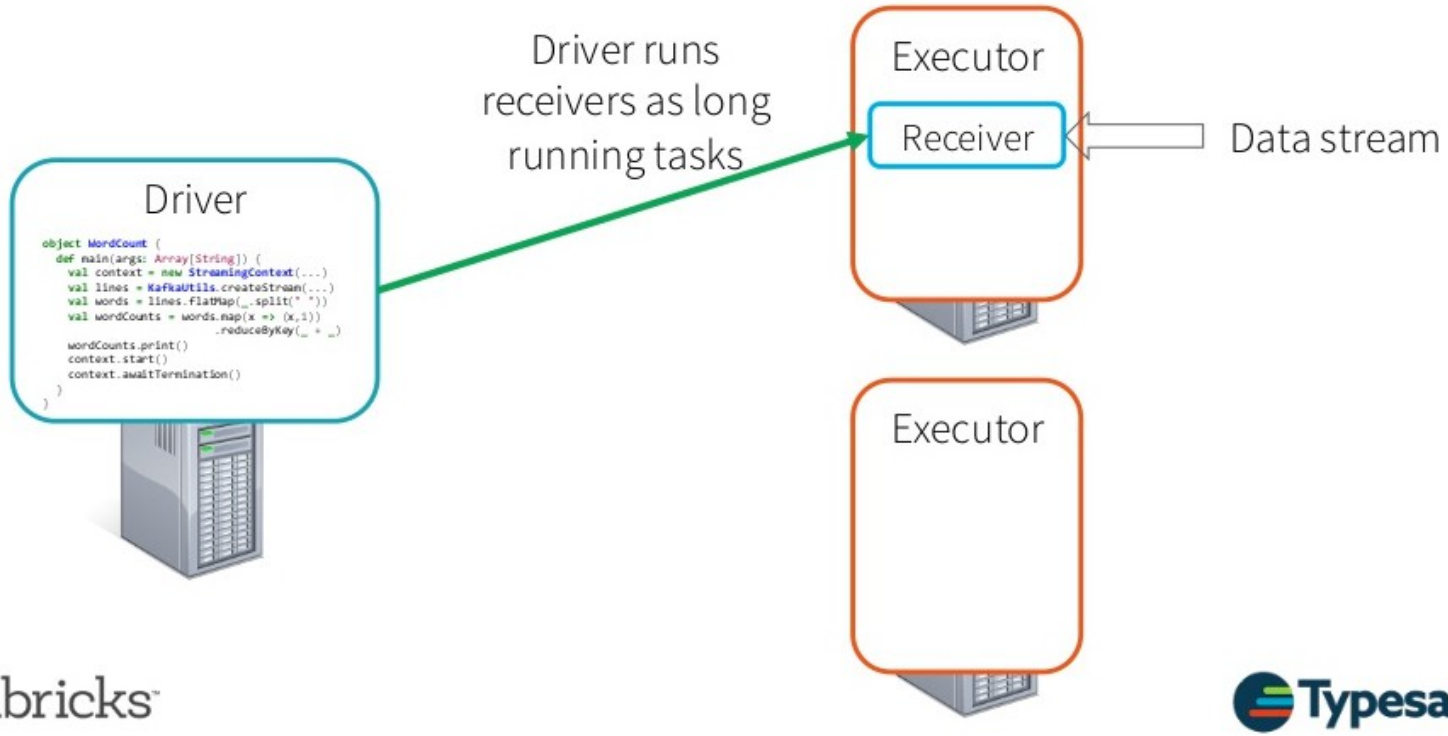
Streaming Architecture



Source: Cloudera

thanachart@imcinstitute.com

Spark Streaming Application: Receive data

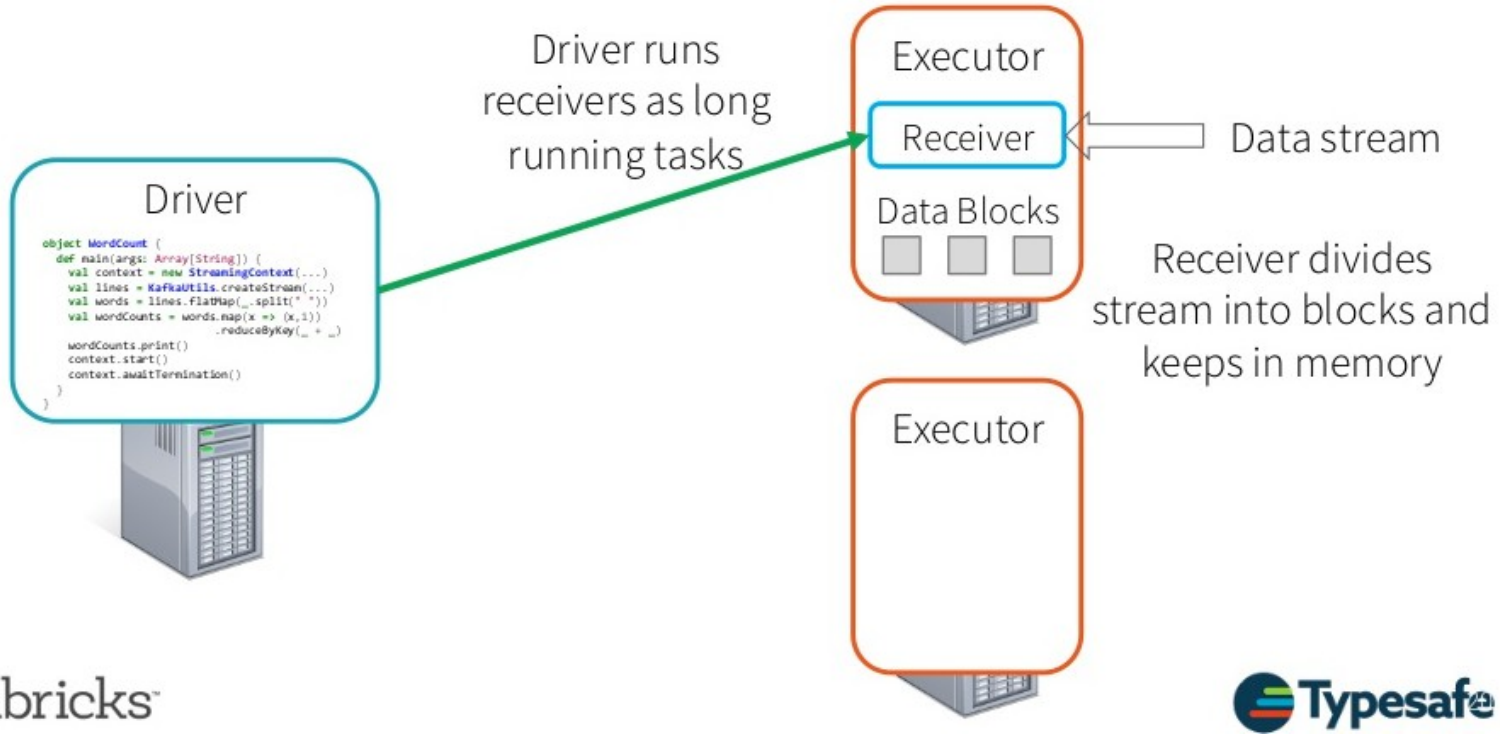


databricks

Typesafe

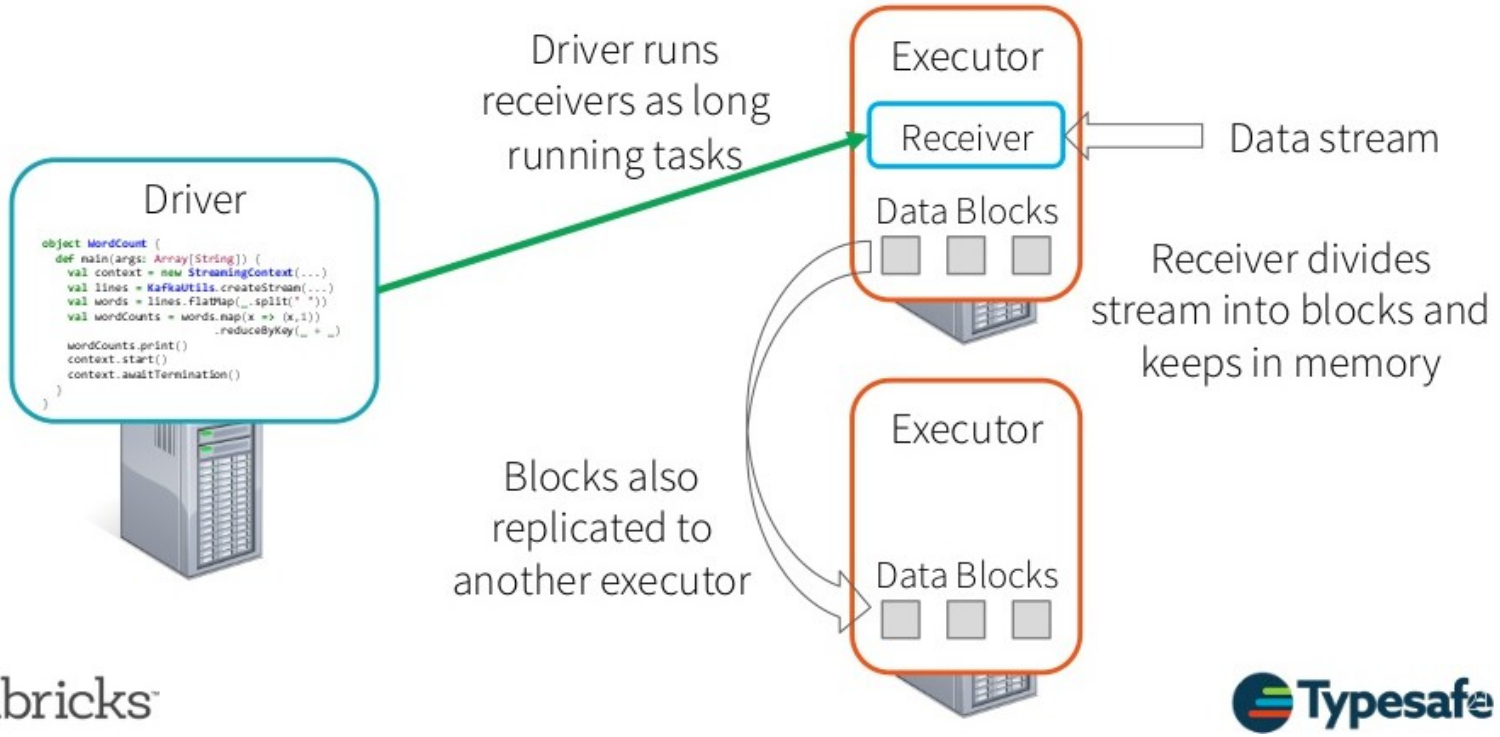
thanachart@imcinstitute.com

Spark Streaming Application: Receive data



thanachart@imcinstitute.com

Spark Streaming Application: Receive data

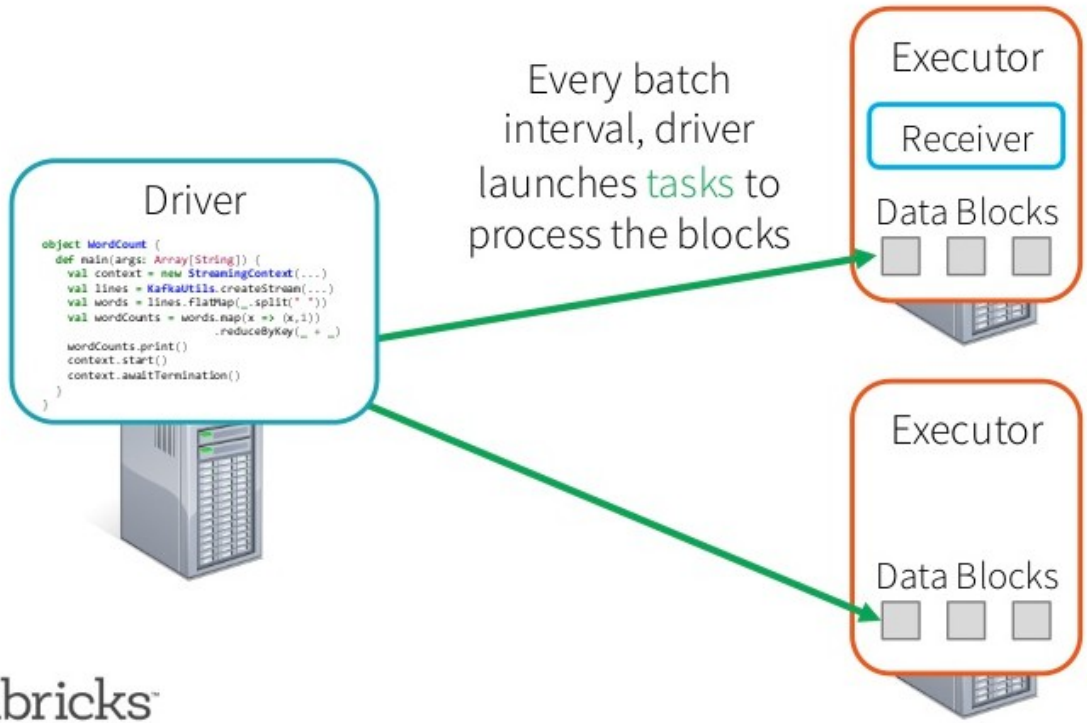


databricks

Typesafe

thanachart@imcinstitute.com

Spark Streaming Application: Process data

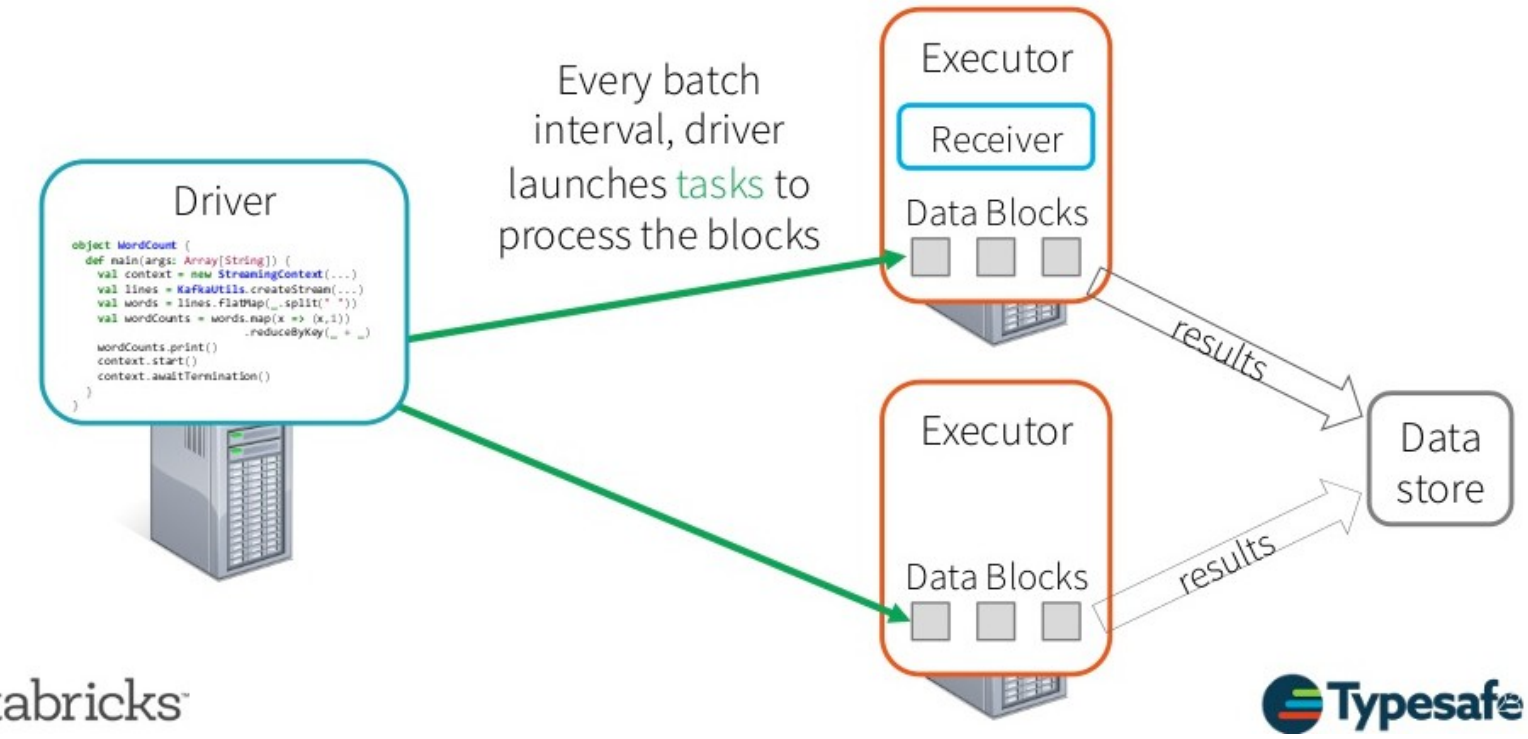


databricks

Typesafe

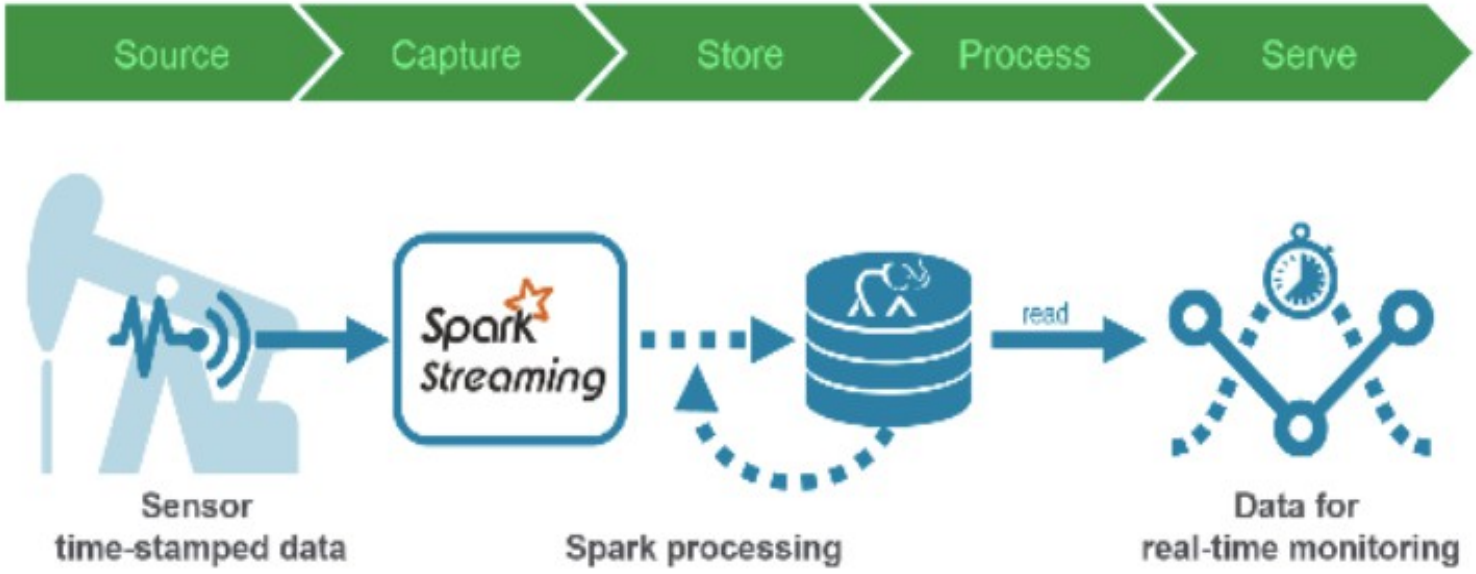
thanachart@imcinstitute.com

Spark Streaming Application: Process data



thanachart@imcinstitute.com

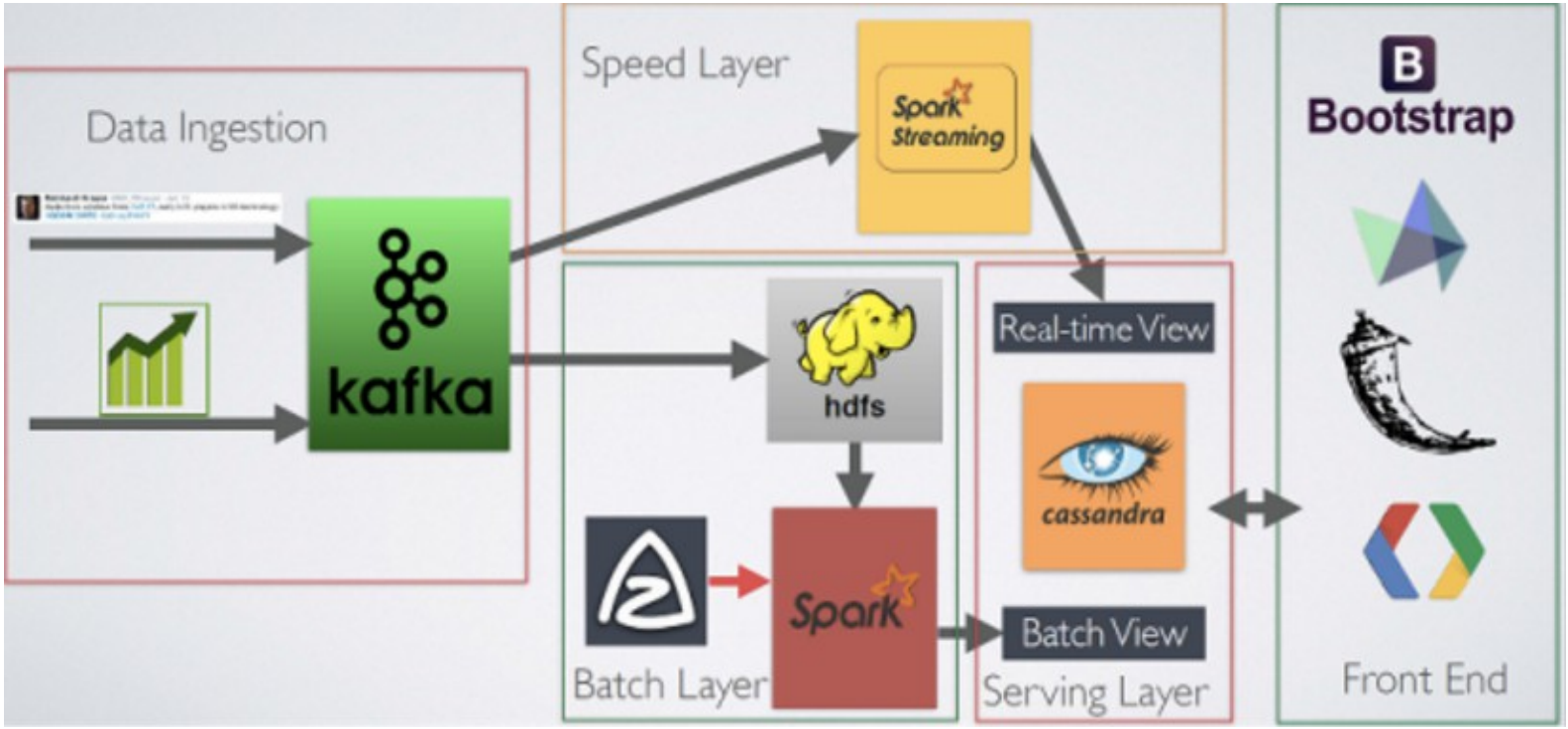
Use Case: Time Series Data



Source: MapR Academy

thanachart@imcinstitute.com

Use Case



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

thanachart@imcinstitute.com

DStream Functional operations

- flatMap(flatMapFunc)
- filter(filterFunc)
- map(mapFunc)
- mapPartitions(mapPartFunc, preservePartitioning)
- forEachRDD(forEachFunc)

thanachart@imcinstitute.com

DStream Output operations

- print()
- saveAsHadoopFiles(…)
- saveAsTextFiles(…)
- saveAsObjectFiles(…)
- saveAsNewAPIHadoopFiles(…)
- forEachRDD(..)

thanachart@imcinstitute.com

Hands-on: 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)
```

Lecture
Understanding Kafka

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Introduction

Open-source message broker project

An open-source message broker project developed by the Apache Software Foundation written in Scala. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. It is, in its essence, a "massively scalable pub/sub message queue architected as a distributed transaction log", making it highly valuable for enterprise infrastructures.

What is Kafka?

- An apache project initially developed at LinkedIn
- Distributed publish-subscribe messaging system
- Designed for processing of real time activity stream data e.g. logs, metrics collections
- Written in Scala
- Does not follow JMS Standards, neither uses JMS APIs

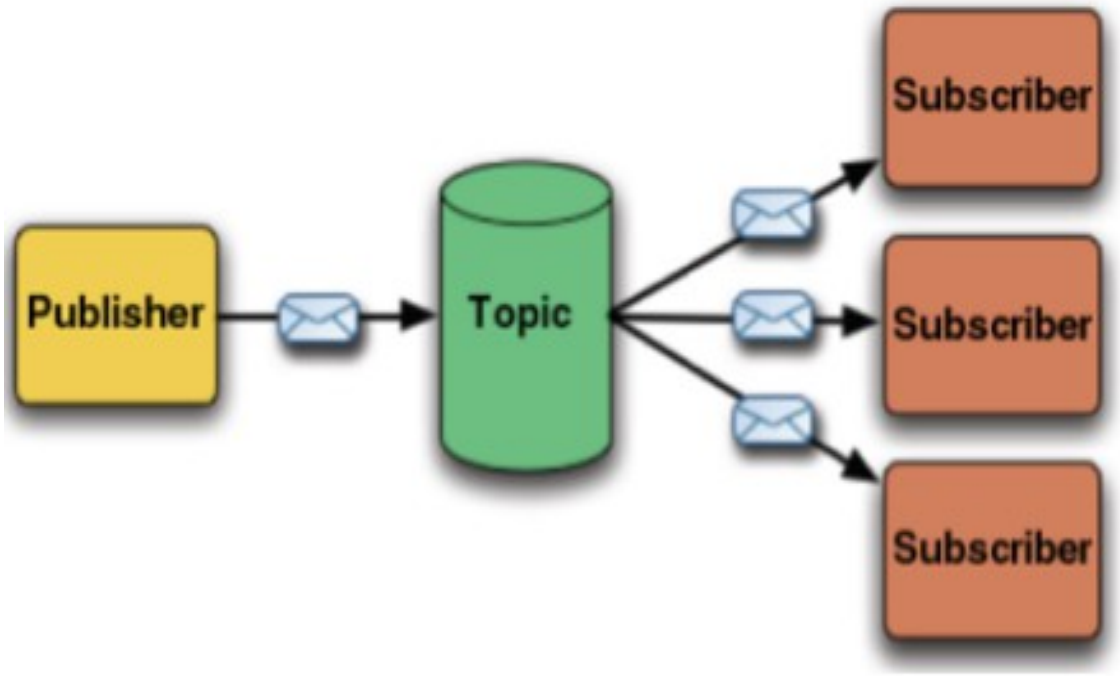
Kafka: Features

- Persistent messaging
- High-throughput
- Supports both queue and topic semantics
- Uses Zookeeper for forming a cluster of nodes (producer/consumer/broker)
- and many more…

Why Kafka?

- Built with speed and scalability in mind.
- Enabled near real-time access to any data source
- Empowered hadoop jobs
- Allowed us to build real-time analytics
- Vastly improved our site monitoring and alerting capability
- Enabled us to visualize and track our call graphs.

Messaging System Concept: Topic



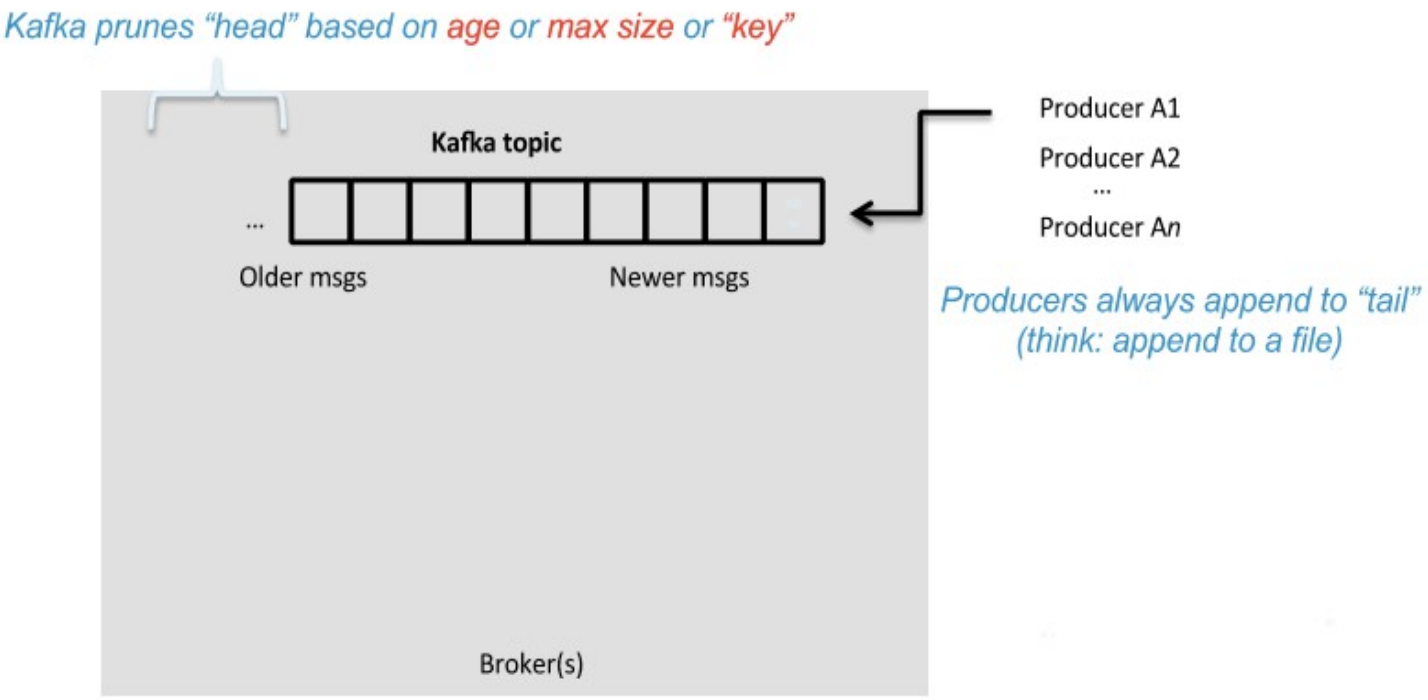
Source: Real time Analytics with Apache Kafka and Spark, Rahul Jain

Terminology

- Kafka maintains feeds of messages in categories called topics.
- Processes that publish messages to a Kafka topic are called producers.
- Processes that subscribe to topics and process the feed of published messages are called consumers.
- Kafka is run as a cluster comprised of one or more servers each of which is called a broker.

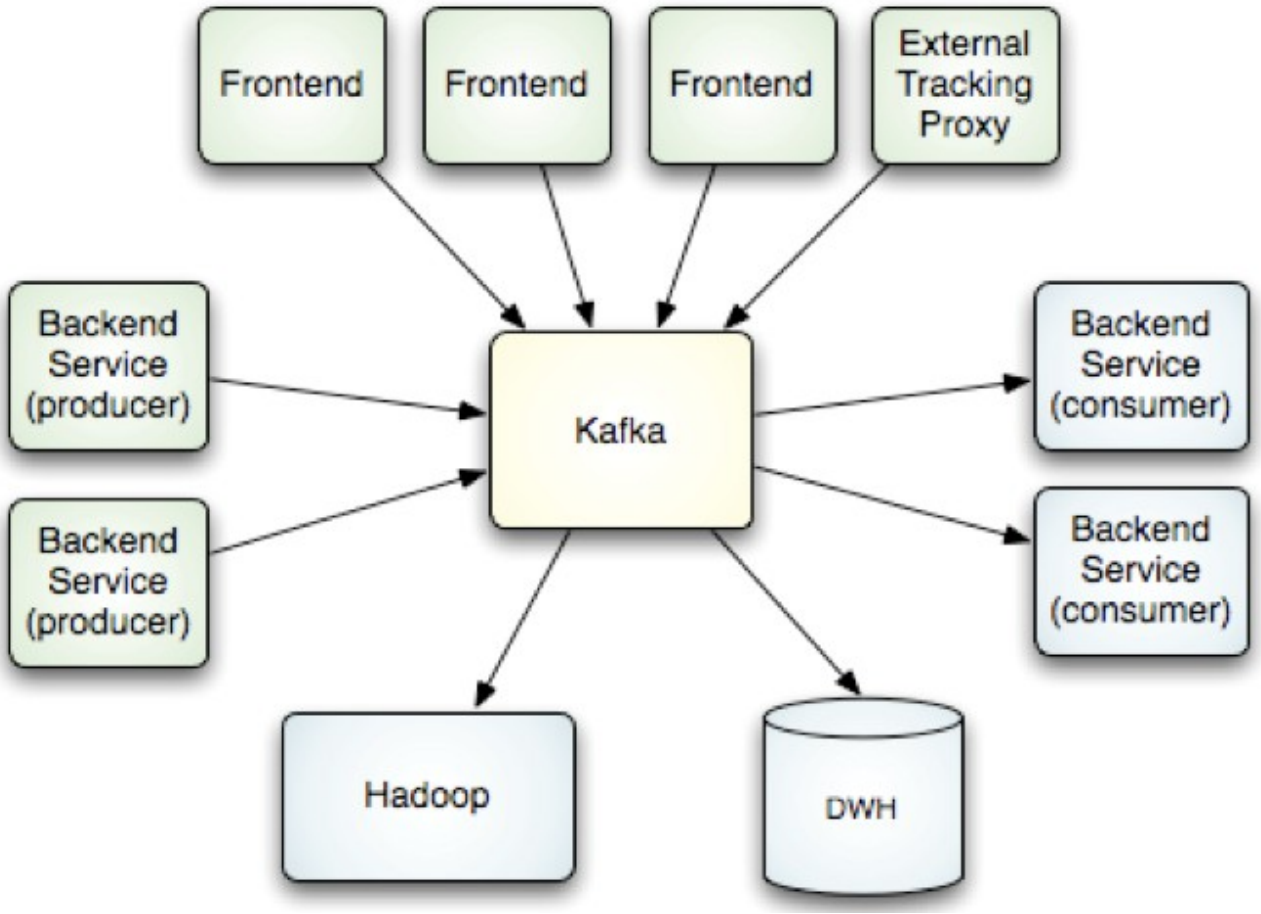
Topics

- Topic: feed name to which messages are published



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

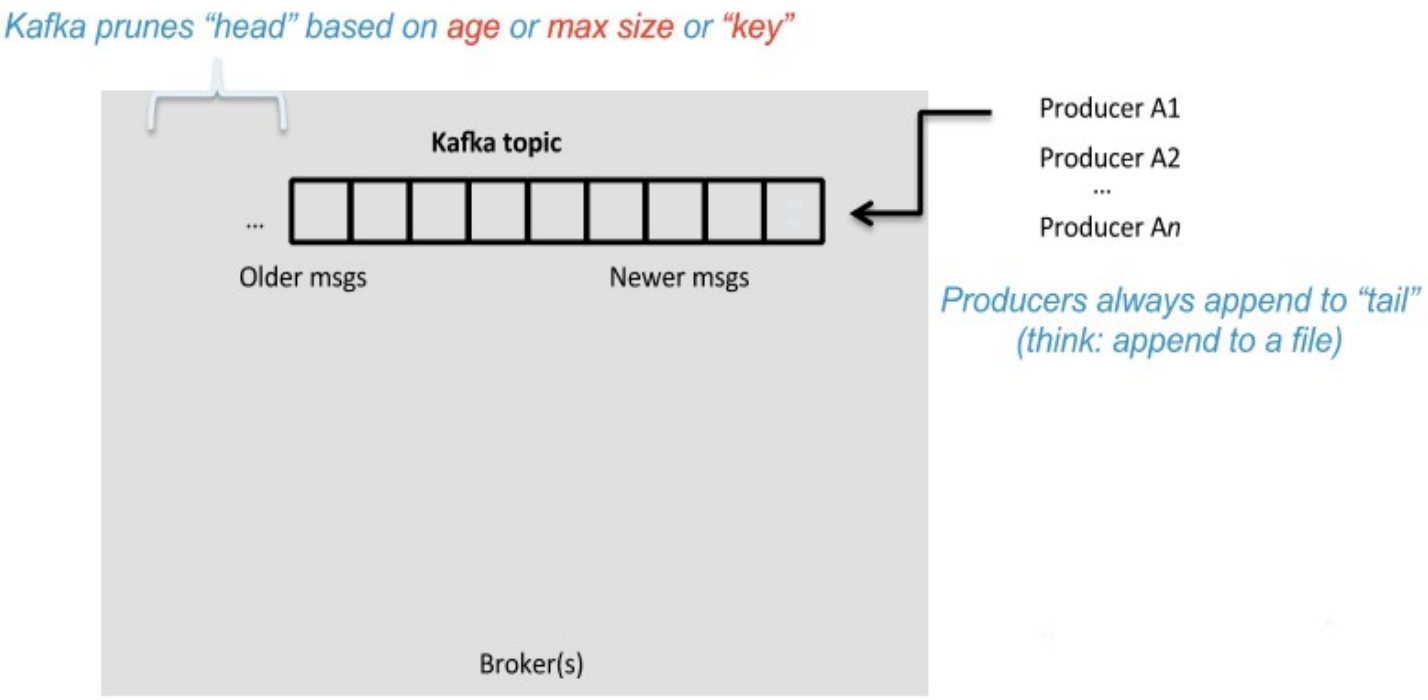
Kafka



Source: Real time Analytics with Apache Kafka and Spark, Rahul Jain

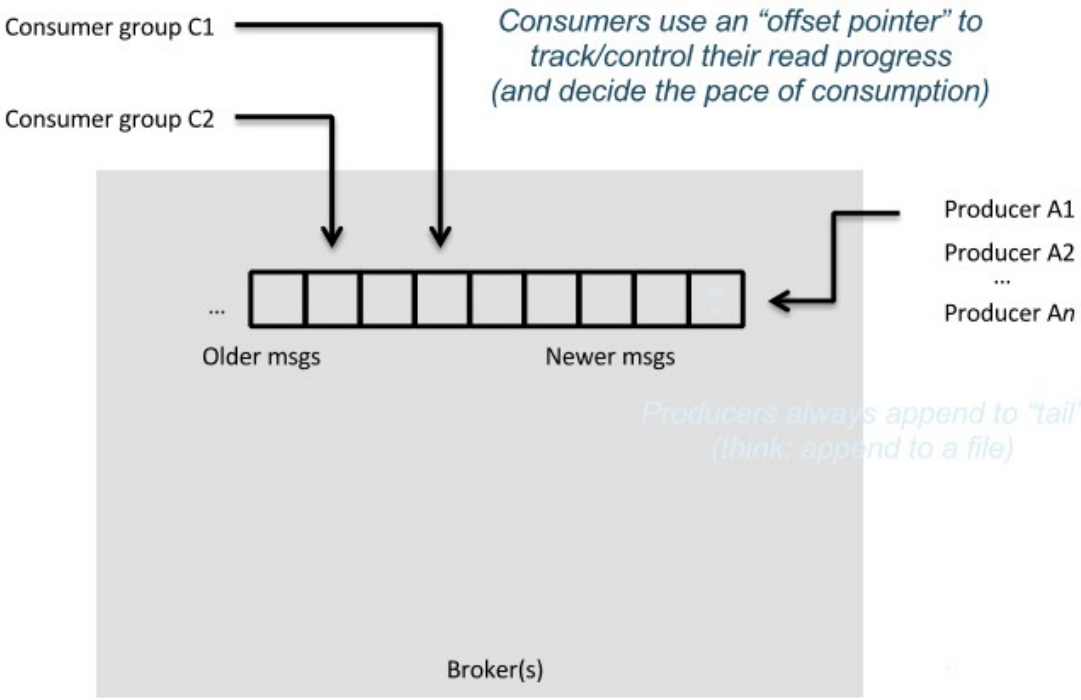
Topics

- Topic: feed name to which messages are published



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

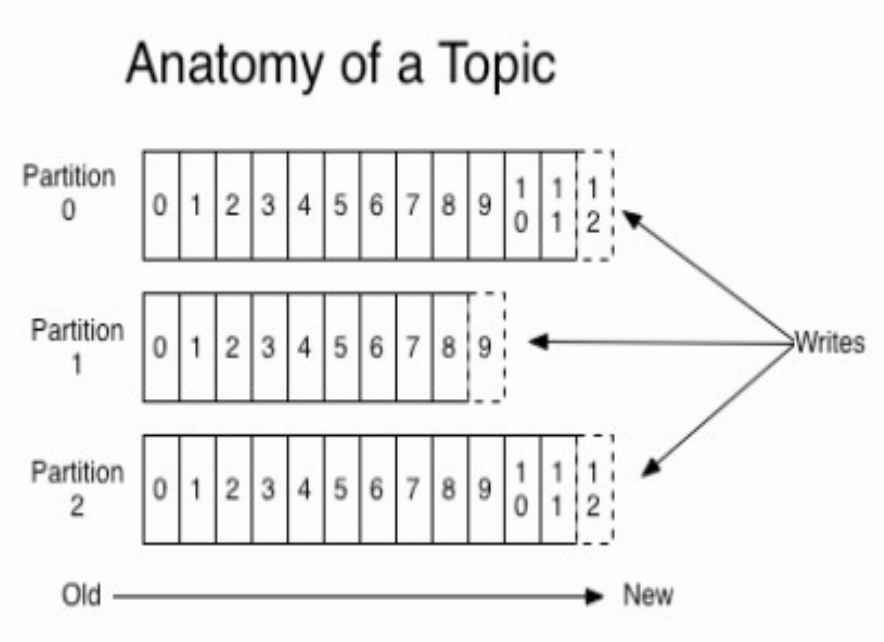
Topics



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

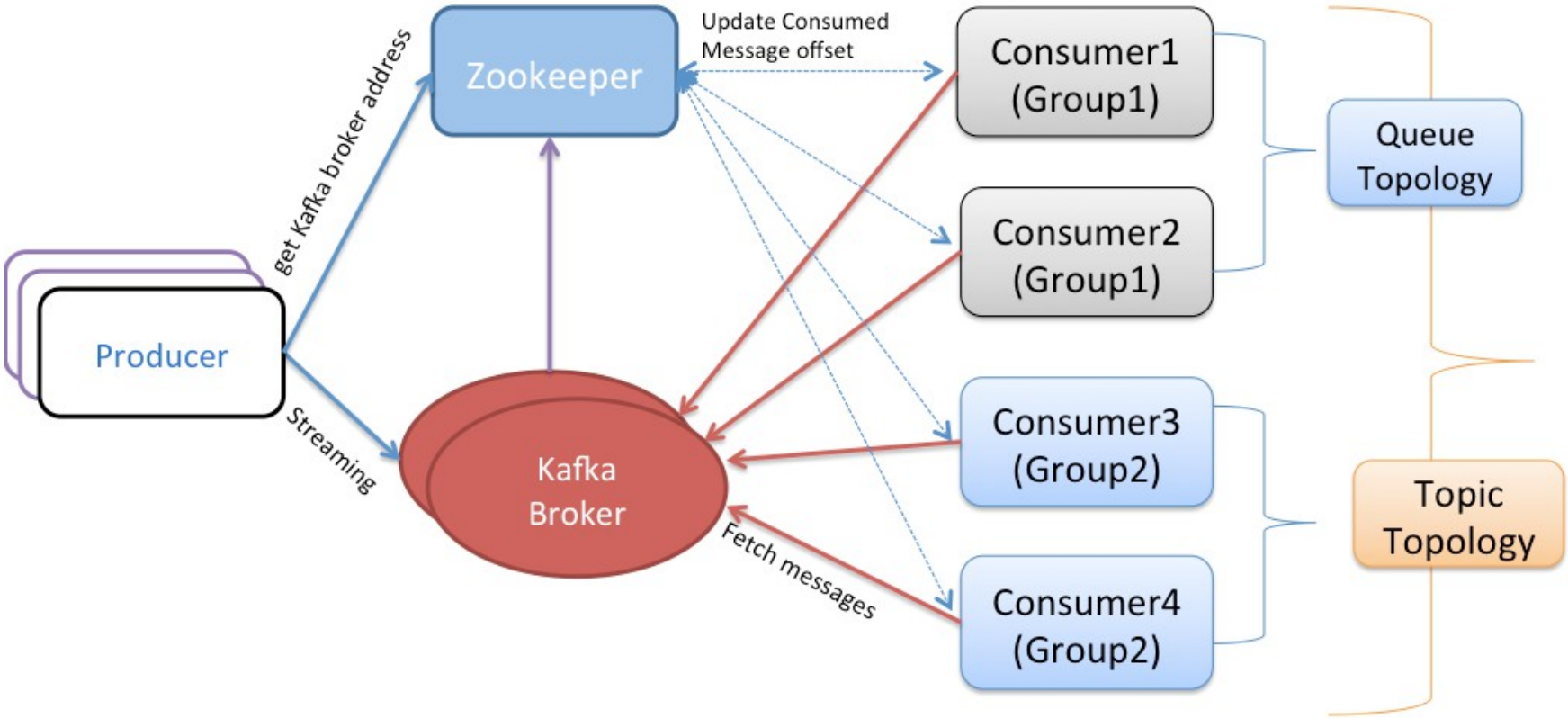
Topics

- A topic consists of partitions.
- Partition: ordered + immutable sequence of messages that is continually appended



Source: Apache Kafka with Spark Streaming - Real Time Analytics Redefined

Kafka Architecture



Source: Real time Analytics with Apache Kafka and Spark, Rahul Jain

Hands-on
SparkStreaming from Kafka

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Install & Start Kafka Server

```
# wget http://www-us.apache.org/dist/kafka/0.9.0.1/kafka_2.10-0.9.0.1.tgz
# tar xzf kafka_2.10-0.9.0.1.tgz
# cd kafka_2.10-0.9.0.1
# bin/kafka-server-start.sh config/server.properties&
```

```
[2016-06-23 04:37:21,426] INFO Kafka commitId : 23c69d62a0cabf06 (o
rg.apache.kafka.common.utils.AppInfoParser)
[2016-06-23 04:37:21,430] INFO [Kafka Server 0], started (kafka.ser
ver.KafkaServer)
[2016-06-23 04:37:21,446] INFO New leader is 0 (kafka.server.Zookee
perLeaderElector$LeaderChangeListener)
```

Running Kafka Producer

```
# bin/kafka-console-producer.sh --topic test --broker-list
localhost:9092
```

type some random messages followed by Ctrl-D to finish

```
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-producer.sh
 --topic test --broker-list localhost:9092
This is a test message from IMC Institute

Big Data School
Test
[root@quickstart kafka_2.10-0.9.0.1]#
```

Running Kafka Consumer

```
# bin/kafka-console-consumer.sh --topic test --zookeeper
localhost:2181 --from-beginning
```

```
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-consumer.sh
 --topic test --zookeeper localhost:2181 --from-beginning
This is a test message from IMC Institute
Big Data School
Test
```

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
      /_/
```

Spark Streaming with Kafka

```
$ 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._
import org.apache.spark.streaming.kafka.KafkaUtils
val ssc = new StreamingContext(sc, Seconds(2))
val kafkaStream = KafkaUtils.createStream(ssc,
"localhost:2181","spark-streaming-consumer-group", Map("spark-
topic" -> 5))
kafkaStream.print()
ssc.start
```

Running Kafka Producer on another terminal

```
# docker ps
```

```
CONTAINER ID        IMAGE                          COMMAND
     CREATED             STATUS              PORTS
   NAMES
c77e4dc1ed9b        cloudera/quickstart:latest   "/usr/bin/docker-q
ui   22 minutes ago      Up 22 minutes       0.0.0.0:8888->8888/tcp
   trusting_newton
```

```
# docker exec -i -t c77e4dc1ed9b /bin/bash
```

```
[root@quickstart ~]# cd /root/kafka_2.10-0.9.0.1
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-
producer.sh --broker-list localhost:9092 --topic spark-topic
```

Test & View the result

```
[root@quickstart kafka_2.10-0.9.0.1]# bin/kafka-console-producer.sh
 --broker-list localhost:9092 --topic spark-topic
Hello from IMC Institute
```

Result from another terminal

```
85200 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1466658086000 ms
-------------------------------------------
(null,Hello from IMC Institute)
```

Hands-on
Realtime analytics with Kafka

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Java Code: Kafka Producer

```
import java.util.Properties;
import kafka.producer.KeyedMessage;
import kafka.producer.ProducerConfig;
import java.io.*;

public class HelloKafkaProducer {
    final static String TOPIC = "java-topic";
    public static void main(String[] argv){
        Properties properties = new Properties();

properties.put("metadata.broker.list","localhost:9092");

properties.put("serializer.class","kafka.serializer.StringEncoder");
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Java Code: Kafka Producer (cont.)

```
        try(BufferedReader br = new BufferedReader(new
FileReader(argv[0]))) {
                StringBuilder sb = new StringBuilder();
                ProducerConfig producerConfig = new
ProducerConfig(properties);
                kafka.javaapi.producer.Producer<String,String>
producer = new kafka.javaapi.producer.Producer<String,
String>(producerConfig);
                String line = br.readLine();

                while (line != null) {
                        KeyedMessage<String, String> message
=new KeyedMessage<String, String>(TOPIC,line);
                        producer.send(message);
                        line = br.readLine();
                }
```

Java Code: Kafka Producer (cont.)

```
                producer.close();
            } catch (IOException ex) {
                    ex.printStackTrace();
            }
        }
}
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Compile & Run the program

```
// Using a vi Editor to edit the sourcecode
# vi HelloKafkaProducer.java
// Alternatively
# wget
https://s3.amazonaws.com/imcbucket/apps/HelloKafkaProducer.java

// Compile progeram
# export CLASSPATH=".:/root/kafka_2.10-0.9.0.1/libs/*"
# javac HelloKafkaProducer.java

//prepare the data
# cd
# wget https://s3.amazonaws.com/imcbucket/input/pg2600.txt
# cd kafka_2.10-0.9.0.1
// Run the program
# java HelloKafkaProducer  /root/pg2600.txt
```

Spark Streaming with Kafka : Wordcount

```
$ 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._
import org.apache.spark.streaming.kafka.KafkaUtils
val ssc = new StreamingContext(sc, Seconds(2))
val kafkaStream = KafkaUtils.createStream(ssc,
"localhost:2181","spark-streaming-consumer-group", Map("java-
topic" -> 5))
```

Spark Streaming with Kafka : WordCount

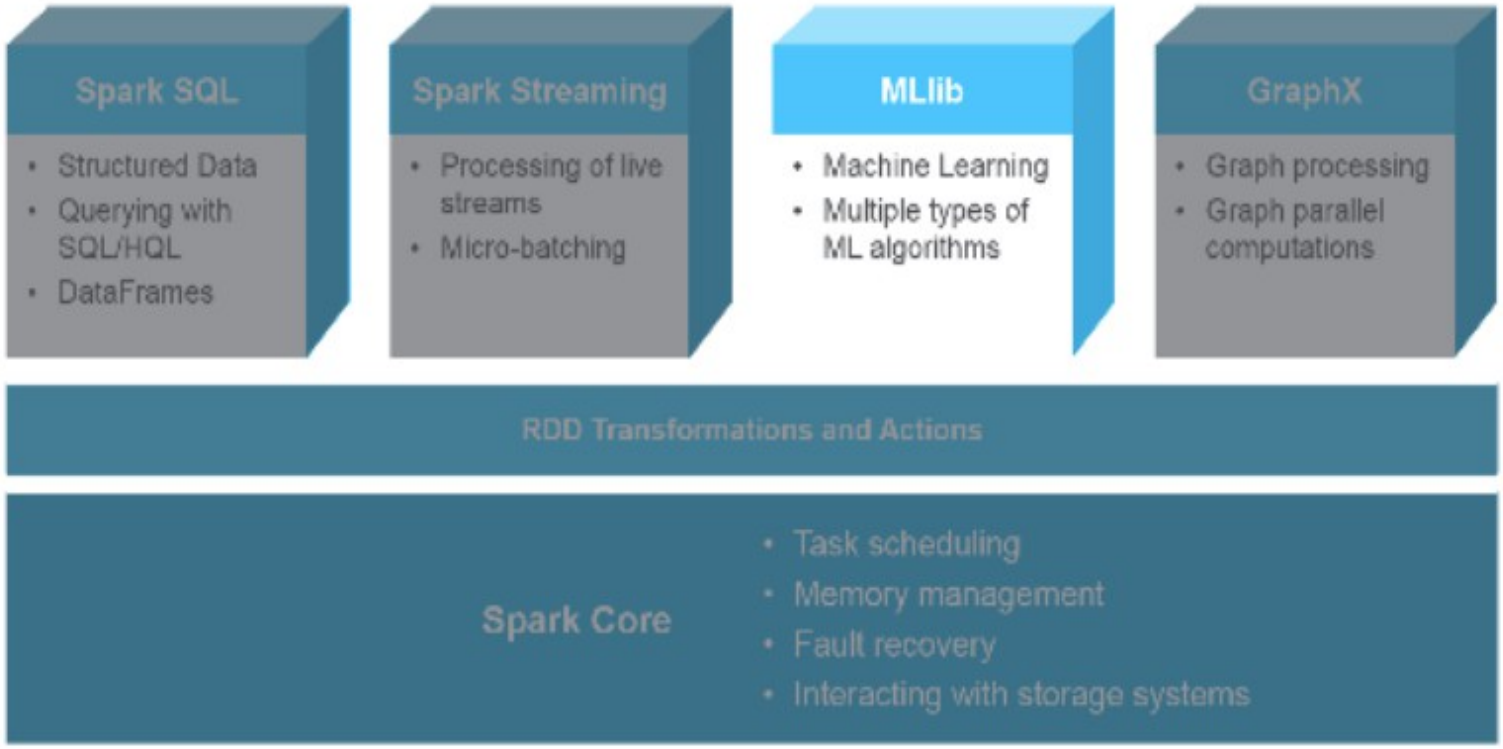
```
val lines = kafkaStream.map(_._2)
val words = lines.flatMap(_.split(" "))
val wordCounts = words.map(x => (x, 1L)).reduceByKey(_ + _)
wordCounts.print()


ssc.start()
ssc.awaitTermination()
```

Output

```
-------------------------------------------
Time: 1467251428000 ms
-------------------------------------------
(Ermolov.,4)
(mattered,4)
(Ah!,10)
(intimately,6)
(Koko,2)
(denied?",2)
(muslin,,2)
(reunion,4)
(blandly,10)
("Ho!,2)
...
```

Spark MLlib

What is MLlib?



Source: MapR Academy

thanachart@imcinstitute.com

What is MLlib?

- MLlib is a Spark subproject providing machine learning primitives:
  - initial contribution from AMPLab, UC Berkeley
  - shipped with Spark since version 0.8
  - 33 contributors

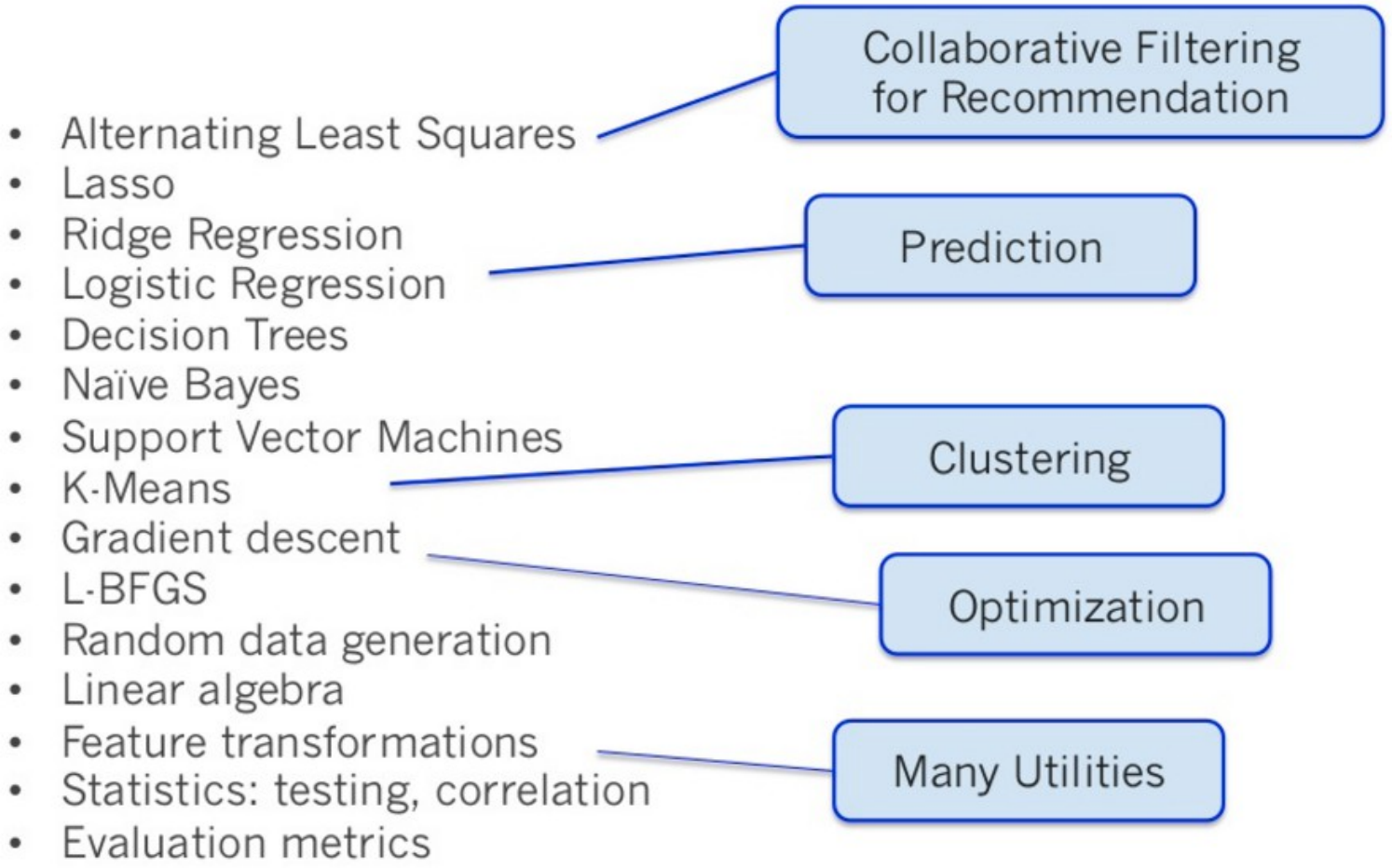
thanachart@imcinstitute.com

Mllib Algorithms

- **Classification**: logistic regression, linear support vector machine(SVM), naive Bayes
- **Regression**: generalized linear regression (GLM)
- **Collaborative filtering**: alternating least squares (ALS)
- **Clustering**: k-means
- **Decomposition**: singular value decomposition (SVD), principal component analysis (PCA)

thanachart@imcinstitute.com

What is in MLlib?



Source: Mllib:Spark's Machine Learning Library, A. Talwalkar

thanachart@imcinstitute.com

MLlib: Benefits

- Part of Spark
- Scalable
- Support: Python, Scala, Java
- Broad coverage of applications & algorithms
- Rapid developments in speed & robustness

thanachart@imcinstitute.com

Machine Learning

Machine learning is a scientific discipline that explores the construction and study of algorithms that can learn from data.

[Wikipedia]

Vectors

- A point is just a set of numbers. This set of numbers or coordinates defines the point's position in space.
- Points and vectors are same thing.
- Dimensions in vectors are called features
- Hyperspace is a space with more than three dimensions.
- Example: A person has the following dimensions:
  - Weight
  - Height
  - Age
- Thus, the interpretation of point (160,69,24) would be 160 lb weight, 69 inches height, and 24 years age.

Source:Spark Cookbook

thanachart@imcinstitute.com

Vectors in MLlib

- Spark has local vectors and matrices and also distributed matrices.
  - Distributed matrix is backed by one or more RDDs.
  - A local vector has numeric indices and double values, and is stored on a single machine.
- Two types of local vectors in MLlib:
  - **Dense vector** is backed by an array of its values.
  - **Sparse vector** is backed by two parallel arrays, one for indices and another for values.
- Example
  - Dense vector: [160.0,69.0,24.0]
  - Sparse vector: (3,[0,1,2],[160.0,69.0,24.0])

Source:Spark Cookbook

thanachart@imcinstitute.com

Vectors in Mllib (cont.)

- Library
  - import org.apache.spark.mllib.linalg.{Vectors,Vector}
- Signature of **Vectors.dense:**
  - def dense(values: Array[Double]): Vector
- Signature of **Vectors.sparse**:
  - def sparse(size: Int, indices: Array[Int], values: Array[Double]): Vector

thanachart@imcinstitute.com

Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Vector}
import org.apache.spark.mllib.linalg.{Vectors, Vector}

scala> val dvPerson = Vectors.dense(160.0,69.0,24.0)
dvPerson: org.apache.spark.mllib.linalg.Vector = [160.0,69.0,24.0]

scala> val svPerson = Vectors.sparse(3,Array(0,1,2),Array(160.0,69.
0,24.0))
svPerson: org.apache.spark.mllib.linalg.Vector = (3,[0,1,2],[160.0,
69.0,24.0])
```


thanachart@imcinstitute.com

Labeled point

- Labeled point is a local vector (sparse/dense),), which has an associated label with it.
- Labeled data is used in supervised learning to help train algorithms.
- Label is stored as a double value in **LabeledPoint**.

Type	Label values
Binary classification	0 or 1
Multiclass classification	0, 1, 2…
Regression	Decimal values

Source:Spark Cookbook

thanachart@imcinstitute.com

Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Vector}
scala> import org.apache.spark.mllib.regression.LabeledPoint
scala>  val willBuySUV =
LabeledPoint(1.0,Vectors.dense(300.0,80,40))

scala> val willNotBuySUV =
LabeledPoint(0.0,Vectors.dense(150.0,60,25))

scala> val willBuySUV =
LabeledPoint(1.0,Vectors.sparse(3,Array(0,1,2),Array(300.0,80,
40)))

scala> val willNotBuySUV =
LabeledPoint(0.0,Vectors.sparse(3,Array(0,1,2),Array(150.0,60,
25)))
```

Example (cont)

```
# vi person_libsvm.txt
```

```
0  1:150 2:60 3:25
1  1:300 2:80 3:40
```

```
scala> import org.apache.spark.mllib.util.MLUtils

scala> import org.apache.spark.rdd.RDD

scala> val persons =
MLUtils.loadLibSVMFile(sc,"hdfs:///user/cloudera/person_libsvm
.txt")

scala> persons.first()
```

```
res0: org.apache.spark.mllib.regression.LabeledPoint = (0.0,(3,[0,1
,2],[150.0,60.0,25.0]))
```

Matrices in MLlib

- Spark has local matrices and also distributed matrices.
  - Distributed matrix is backed by one or more RDDs.
  - A local matrix stored on a single machine.
- There are three types of distributed matrices in MLlib:
  - **RowMatrix**: This has each row as a feature vector.
  - **IndexedRowMatrix**: This also has row indices.
  - **CoordinateMatrix**: This is simply a matrix of MatrixEntry. A MatrixEntry represents an entry in the matrix represented by its row and column index

Source:Spark Cookbook

thanachart@imcinstitute.com

Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Matrix,
Matrices}
scala> val people = Matrices.dense(3,2,Array(150d,60d,25d,
300d,80d,40d))
```

```
people: org.apache.spark.mllib.linalg.Matrix =
150.0  300.0
60.0   80.0
25.0   40.0
```

```
scala>  val personRDD =
sc.parallelize(List(Vectors.dense(150,60,25),
Vectors.dense(300,80,40)))
scala> import org.apache.spark.mllib.linalg.distributed.
{IndexedRow, IndexedRowMatrix,RowMatrix, CoordinateMatrix,
MatrixEntry}
scala> val personMat = new RowMatrix(personRDD)
```

Example

```
scala> print(personMat.numRows)
scala> val personRDD = sc.parallelize(List(IndexedRow(0L,
Vectors.dense(150,60,25)), IndexedRow(1L,
Vectors.dense(300,80,40))))
scala>  val pirmat = new IndexedRowMatrix(personRDD)
scala> val personMat = pirmat.toRowMatrix
scala> val meRDD = sc.parallelize(List(
  MatrixEntry(0,0,150), MatrixEntry(1,0,60),
MatrixEntry(2,0,25), MatrixEntry(0,1,300),
MatrixEntry(1,1,80),MatrixEntry(2,1,40) ))
scala> val pcmat = new CoordinateMatrix(meRDD)
```

```
scala> print(pcmat.numRows)
3
scala> print(pcmat.numCols)
2
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Statistic functions

- Central tendency of data—mean, mode, median
- Spread of data—variance, standard deviation
- Boundary conditions—min, max

thanachart@imcinstitute.com

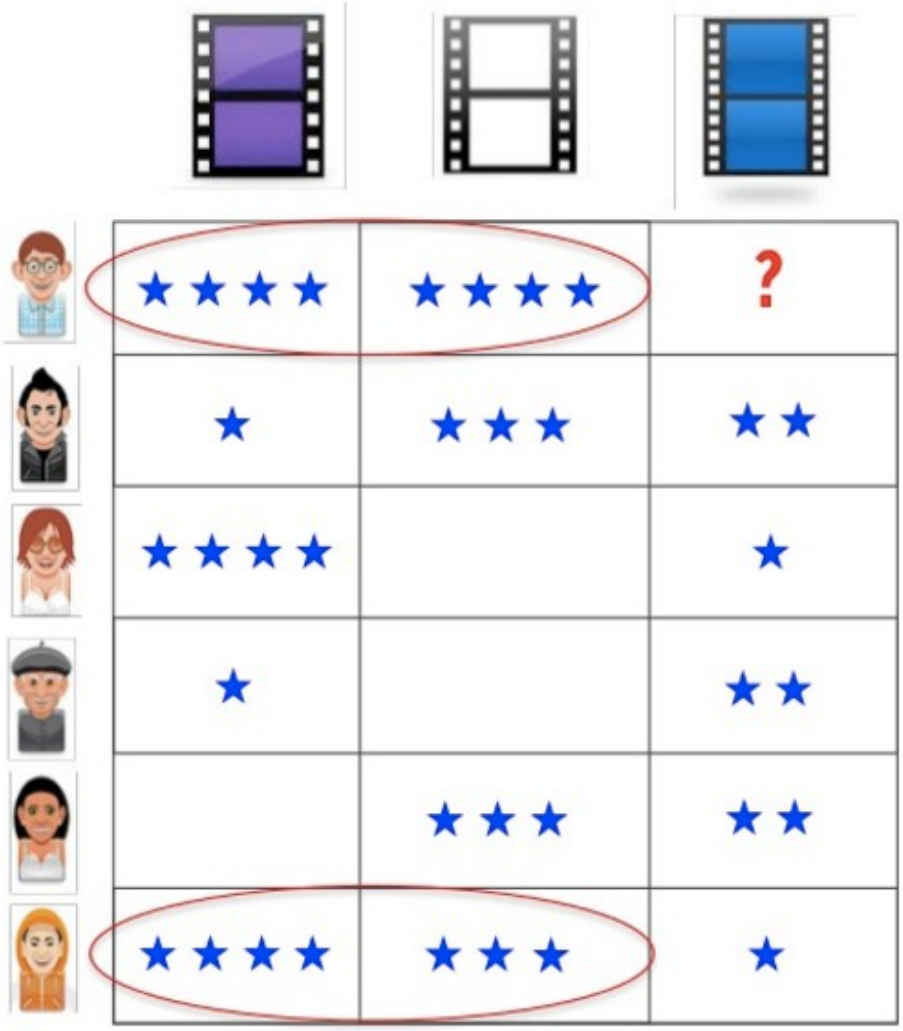
Example

```
scala> import org.apache.spark.mllib.linalg.{Vectors,Vector}
scala> import org.apache.spark.mllib.stat.Statistics
scala> val personRDD =
sc.parallelize(List(Vectors.dense(150,60,25),
Vectors.dense(300,80,40)))
scala> val summary = Statistics.colStats(personRDD)
```

```
scala> print(summary.mean)
[225.0,70.0,32.5]
scala> print(summary.variance)
[11250.0,200.0,112.5]
scala> print(summary.numNonzeros)
[2.0,2.0,2.0]
scala> print(summary.count)
2
scala> print(summary.max)
[300.0,80.0,40.0]
```

Hands-on
Movie Recommendation

Recommendation

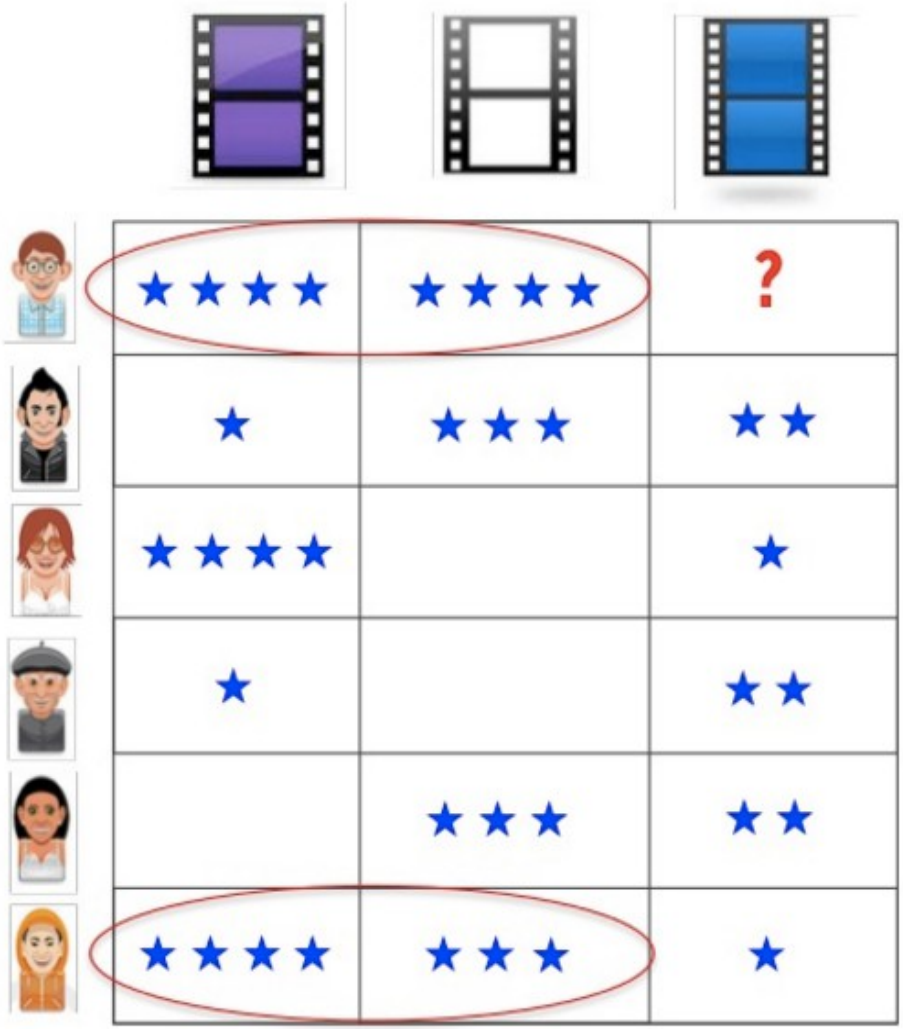


Goal: Recommend movies to users

Source: Mllib:Spark's Machine Learning Library, A. Talwalkar

thanachart@imcinstitute.com

Recommendation: Collaborative Filtering



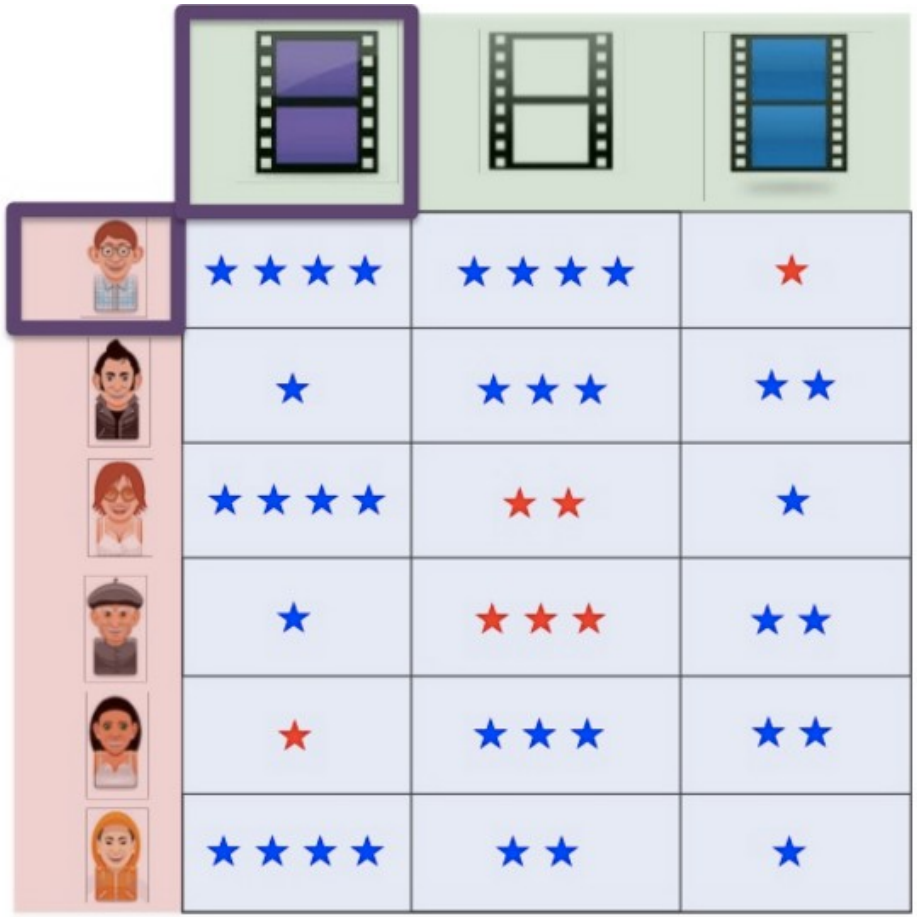
Goal: Recommend movies to users

NETFLIX

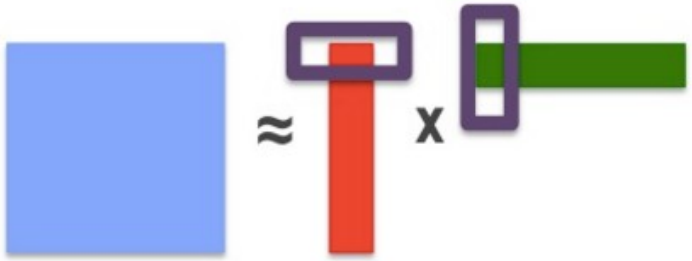
Source: Mllib:Spark's Machine Learning Library, A. Talwalkar

thanachart@imcinstitute.com

Recommendation



Solution: Assume ratings are determined by a small number of factors.



25M Users, 100K Movies
→ 2.5 trillion ratings
With 10 factors/user
→ 250M parameters

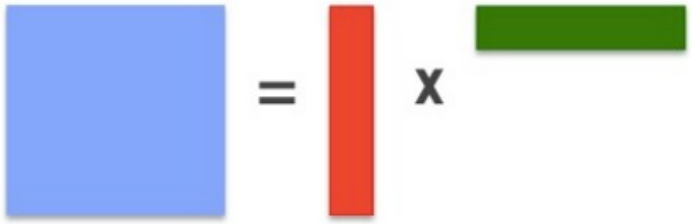
Source: Mllib:Spark's Machine Learning Library, A. Talwalkar

thanachart@imcinstitute.com

Recommendation: ALS

Algorithm

Alternating update of user/movie factors



Can update factors in parallel

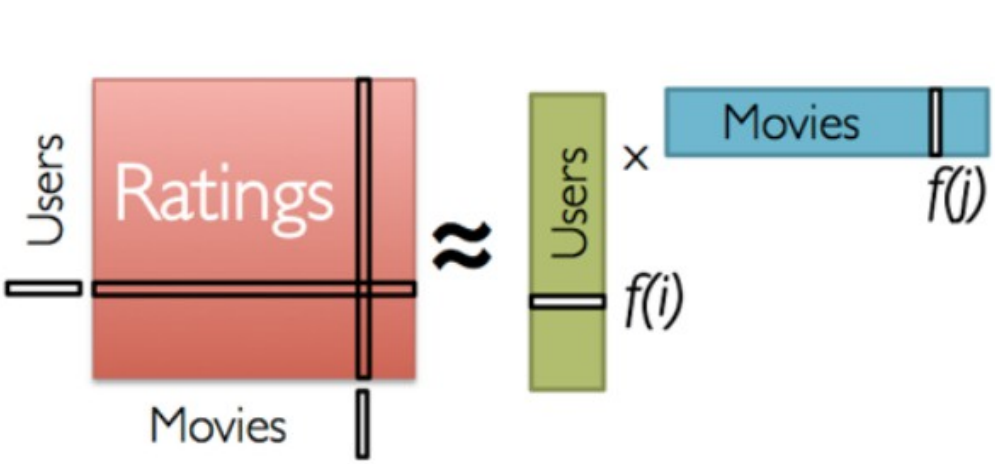
Must be careful about communication

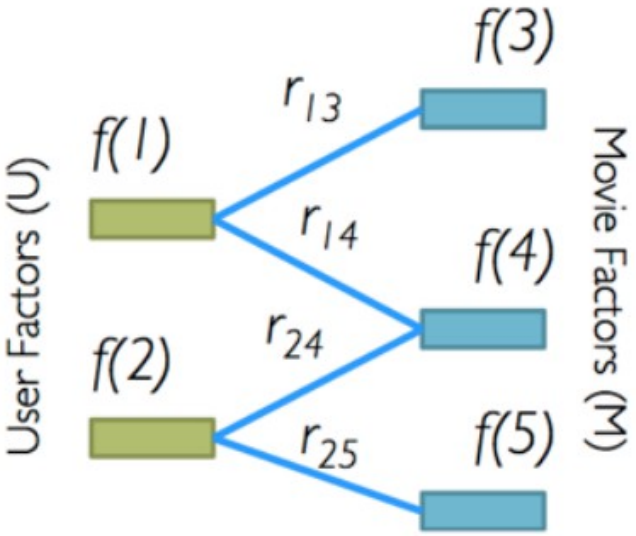
NETFLIX last·fm amazon.com PANDORA LinkedIn

Source: Mllib:Spark's Machine Learning Library, A. Talwalkar

thanachart@imcinstitute.com

Alternating least squares (ALS)





Iterate:

$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

Source: MLlib: Scalable Machine Learning on Spark, X. Meng

thanachart@imcinstitute.com

MLlib: ALS Algorithm

- **numBlocks** is the number of blocks used to parallelize computation (set to -1 to autoconfigure)
- **rank** is the number of latent factors in the model
- **iterations** is the number of iterations to run
- **lambda** specifies the regularization parameter in ALS
- **implicitPrefs** specifies whether to use the explicit feedback ALS variant or one adapted for an implicit feedback data
- **alpha** is a parameter applicable to the implicit feedback variant of ALS that governs the baseline confidence in preference observations

thanachart@imcinstitute.com

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
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

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 -put u.data /user/cloudera/movielens`

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

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

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

```
[root@quickstart ml-100k]# hadoop fs -ls /user/cloudera/movielens
Found 3 items
-rw-r--r--   1 root cloudera        202 2016-07-01 06:34 /user/clou
dera/movielens/u.genre
-rw-r--r--   1 root cloudera     236344 2016-07-01 06:35 /user/clou
dera/movielens/u.item
-rw-r--r--   1 root cloudera      22628 2016-07-01 06:34 /user/clou
dera/movielens/u.user
[root@quickstart ml-100k]#
```

Start Spark-shell with extra memory

```
[root@quickstart ml-100k]# spark-shell --driver-memory 4g
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
      /_/
```

Extracting features from the MovieLens dataset

```
scala> val rawData =
sc.textFile("hdfs:///user/cloudera/movielens/u.data")
scala> rawData.first()
```

```
res0: String = 196      242     3       881250949
```

```
scala> val rawRatings = rawData.map(_.split("\t").take(3))
scala> rawRatings.first()
```

```
res2: Array[String] = Array(196, 242, 3)
```

```
scala> import org.apache.spark.mllib.recommendation.Rating
scala> val ratings = rawRatings.map { case Array(user, movie,
rating) =>Rating(user.toInt, movie.toInt, rating.toDouble) }
scala> ratings.first()
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Training the recommendation model

```
scala> import org.apache.spark.mllib.recommendation.ALS
scala> val model = ALS.train(ratings, 50, 10, 0.01)
```

Note: We'll use rank of 50, 10 iterations, and a lambda parameter of 0.01

```
scala> model.userFeatures.count
res5: Long = 943

scala> model.productFeatures.count
res6: Long = 1682

scala> val predictedRating = model.predict(789, 123)
predictedRating: Double = 3.2037183608258197
```

Inspecting the recommendations

```
scala> val movies =
sc.textFile("hdfs:///user/cloudera/movielens/u.item")

scala> val titles = movies.map(line =>
line.split("\\|").take(2)).map(array
=>(array(0).toInt,array(1))).collectAsMap()
```

```
titles: scala.collection.Map[Int,String] = Map(137 -> Big Night (1996), 891 -> Bent
(1997), 550 -> Die Hard: With a Vengeance (1995), 1205 -> Secret Agent, The (1996),
146 -> Unhook the Stars (1996), 864 -> My Fellow Americans (1996), 559 -> Interview
with the Vampire (1994), 218 -> Cape Fear (1991), 568 -> Speed (1994), 227 -> Star T
rek VI: The Undiscovered Country (1991), 765 -> Boomerang (1992), 1115 -> Twelfth Ni
ght (1996), 774 -> Prophecy, The (1995), 433 -> Heathers (1989), 92 -> True Romance
(1993), 1528 -> Nowhere (1997), 846 -> To Gillian on Her 37th Birthday (1996), 1187
-> Switchblade Sisters (1975), 1501 -> Prisoner of the Mountains (Kavkazsky Plennik)
 (1996), 442 -> Amityville Curse, The (1990), 1160 -> Love! Valour! Compassion! (199
7), 101 -> Heavy Metal (1981), 1196 -> Sa...
```

Inspecting the recommendations (cont.)

```
scala> val moviesForUser = ratings.keyBy(_.user).lookup(789)
```

```
moviesForUser: Seq[org.apache.spark.mllib.recommendation.Rating] = WrappedArray(Rati
ng(789,1012,4.0), Rating(789,127,5.0), Rating(789,475,5.0), Rating(789,93,4.0), Rati
ng(789,1161,3.0), Rating(789,286,1.0), Rating(789,293,4.0), Rating(789,9,5.0), Ratin
g(789,50,5.0), Rating(789,294,3.0), Rating(789,181,4.0), Rating(789,1,3.0), Rating(7
89,1008,4.0), Rating(789,508,4.0), Rating(789,284,3.0), Rating(789,1017,3.0), Rating
(789,137,2.0), Rating(789,111,3.0), Rating(789,742,3.0), Rating(789,248,3.0), Rating
(789,249,3.0), Rating(789,1007,4.0), Rating(789,591,3.0), Rating(789,150,5.0), Ratin
g(789,276,5.0), Rating(789,151,2.0), Rating(789,129,5.0), Rating(789,100,5.0), Ratin
g(789,741,5.0), Rating(789,288,3.0), Rating(789,762,3.0), Rating(789,628,3.0), Ratin
g(789,124,4.0))
```

```
scala> moviesForUser.sortBy(-_.rating).take(10).map(rating =>
(titles(rating.product), rating.rating)).foreach(println)
```

```
(Godfather, The (1972),5.0)
(Trainspotting (1996),5.0)
(Dead Man Walking (1995),5.0)
(Star Wars (1977),5.0)
(Swingers (1996),5.0)
(Leaving Las Vegas (1995),5.0)
(Bound (1996),5.0)
(Fargo (1996),5.0)
(Last Supper, The (1995),5.0)
(Private Parts (1997),4.0)
```

Top 10 Recommendation for userid 789

```
scala> val topKRecs = model.recommendProducts(789,10)
scala> topKRecs.map(rating => (titles(rating.product),
rating.rating)).foreach(println)
```

```
(GoodFellas (1990),5.561893309975536)
(Apocalypse Now (1979),5.359509740087787)
(Being There (1979),5.253109995320087)
(Carrie (1976),5.214960672591296)
(Aliens (1986),5.18467232737804)
(Psycho (1960),5.184123552034558)
(One Flew Over the Cuckoo's Nest (1975),5.174956083257432)
(Full Monty, The (1997),5.145369582639113)
(Flirting With Disaster (1996),5.128468420256269)
(Heavy Metal (1981),5.112027118820185)
```

Evaluating Performance: Mean Squared Error

```
scala> val actualRating = moviesForUser.take(1)(0)

scala> val predictedRating = model.predict(789,
actualRating.product)

scala> val squaredError = math.pow(predictedRating -
actualRating.rating, 2.0)
```

```
scala> val actualRating = moviesForUser.take(1)(0)
actualRating: org.apache.spark.mllib.recommendation.Rating = Rating(789,1012,4.0)

scala> val predictedRating = model.predict(789, actualRating.product)
predictedRating: Double = 3.9903742702273326

scala> val squaredError = math.pow(predictedRating - actualRating.rating, 2.0)
squaredError: Double = 9.265467365641563E-5
```

Overall Mean Squared Error

```
scala> val usersProducts = ratings.map{ case Rating(user,
product, rating)  => (user, product)}

scala> val predictions = model.predict(usersProducts).map{

 case Rating(user, product, rating) => ((user, product),
rating)}

scala> val ratingsAndPredictions = ratings.map{

  case Rating(user, product, rating) => ((user, product),
rating)

}.join(predictions)

scala> val MSE = ratingsAndPredictions.map{

    case ((user, product), (actual, predicted)) =>
math.pow((actual - predicted), 2)

}.reduce(_ + _) / ratingsAndPredictions.count
```

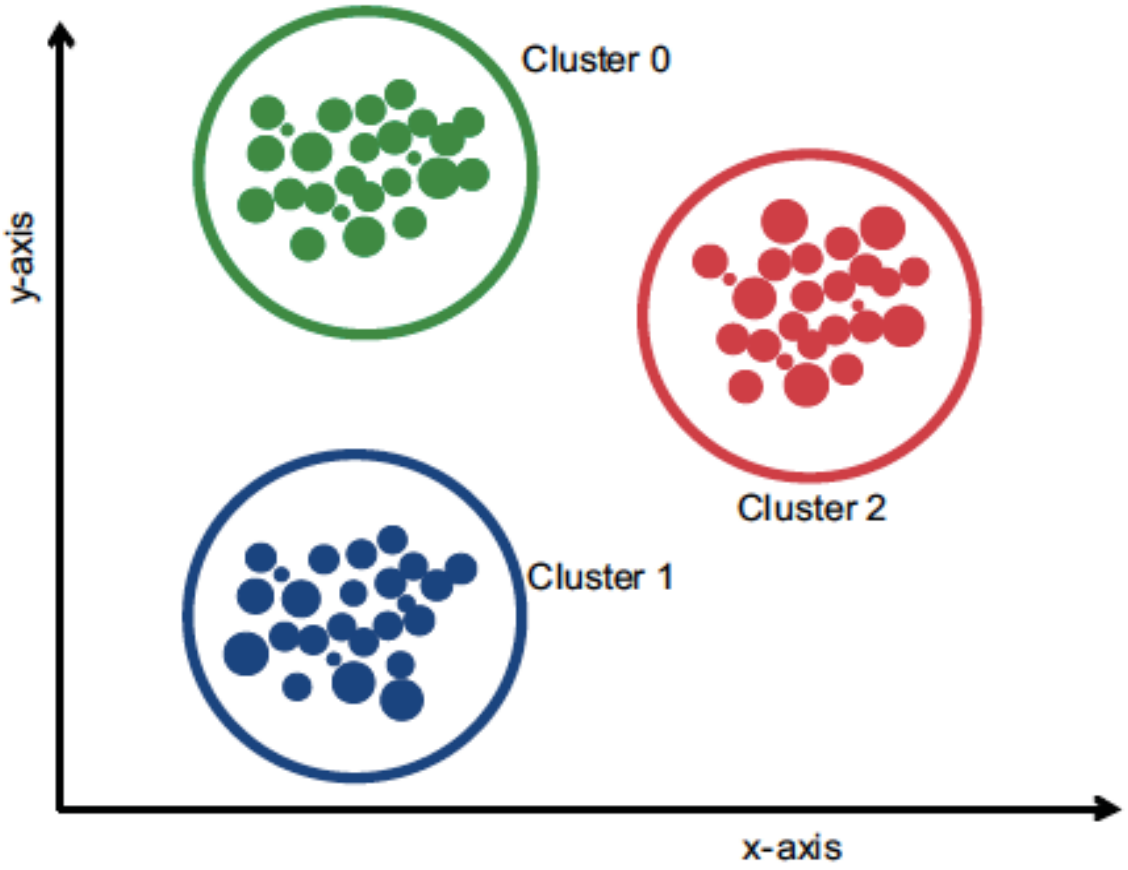
```
scala> println("Mean Squared Error = " + MSE)
Mean Squared Error = 0.097528985120825
```

Clustering using K-Means

Clustering use cases

- Market segmentation
- Social network analysis: Finding a coherent group of people in the social network for ad targeting
- Data center computing clusters
- Real estate: Identifying neighborhoods based on similar features
- Text analysis: Dividing text documents, such as novels or essays, into genres

thanachart@imcinstitute.com

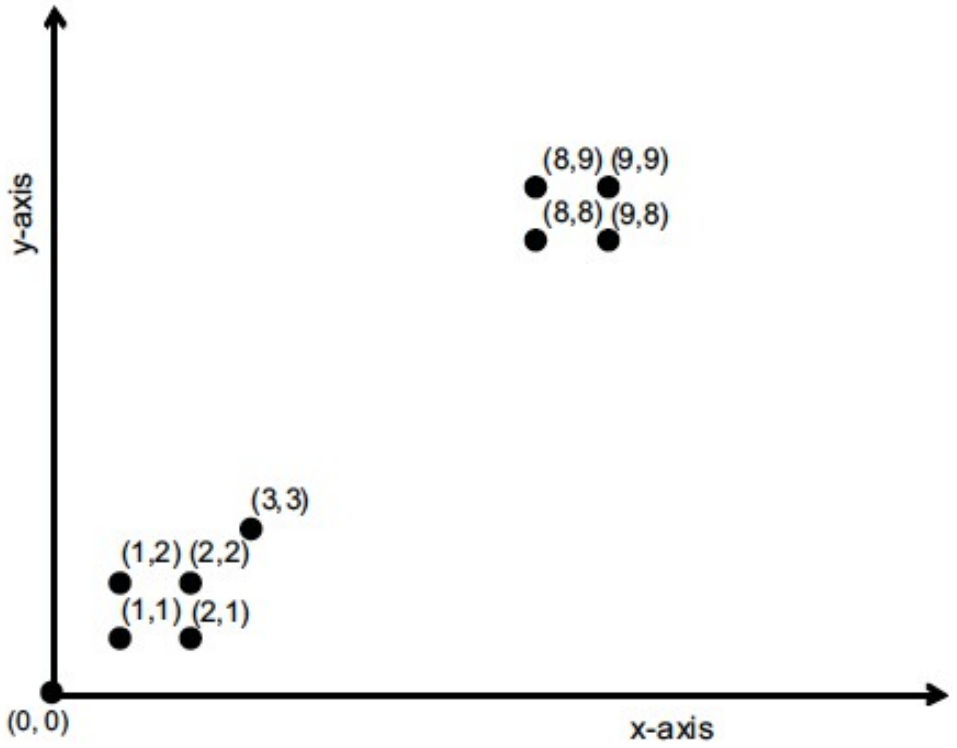


Source: Mahout in Action

thanachart@imcinstitute.com

Sample Data

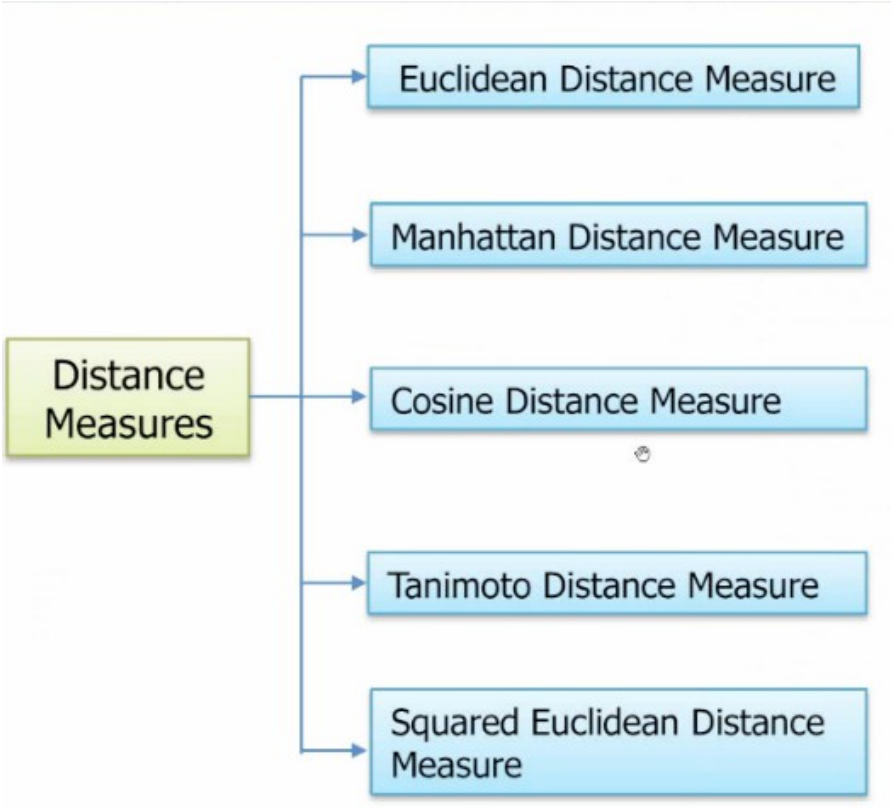
```
(1,1)
(2,1)
(1,2)
(2,2)
(3,3)
(8,8)
(8,9)
(9,8)
(9,9)
```



Source: Mahout in Action

thanachart@imcinstitute.com

Distance Measures



Source www.edureka.in/data-science

thanachart@imcinstitute.com

Distance Measures

- Euclidean distance measure

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

- Squared Euclidean distance measure

$$d = (a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2$$

- Manhattan distance measure

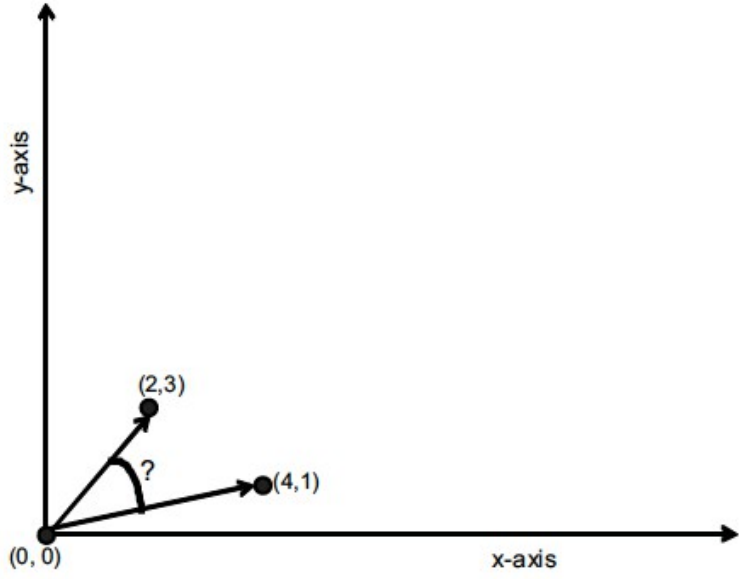
$$d = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n|$$

- Cosine distance measure

$$d = 1 - \frac{(a_1 b_1 + a_2 b_2 + \ldots + a_n b_n)}{(\sqrt{(a_1^2 + a_2^2 + \ldots + a_n^2)}\sqrt{(b_1^2 + b_2^2 + \ldots + b_n^2)})}$$

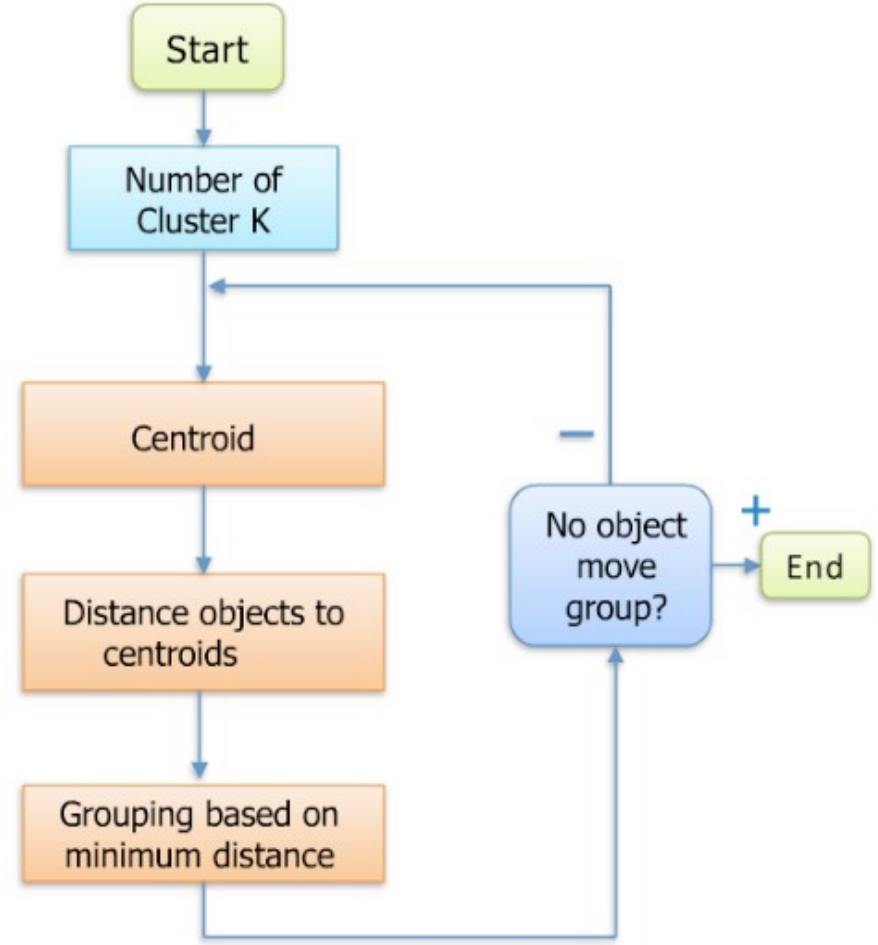
thanachart@imcinstitute.com

Distance Measures



thanachart@imcinstitute.com

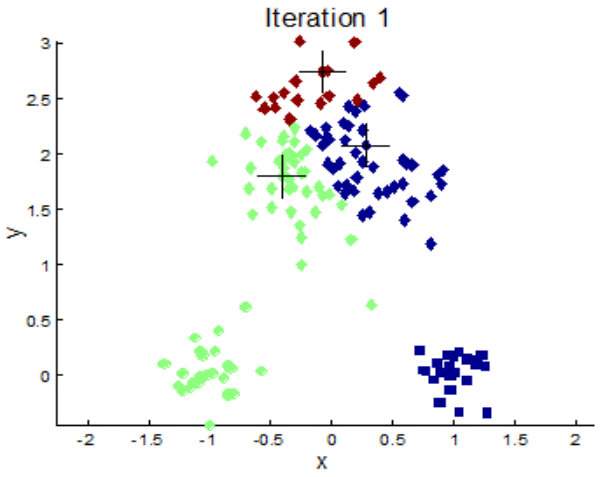
K-Means Clustering

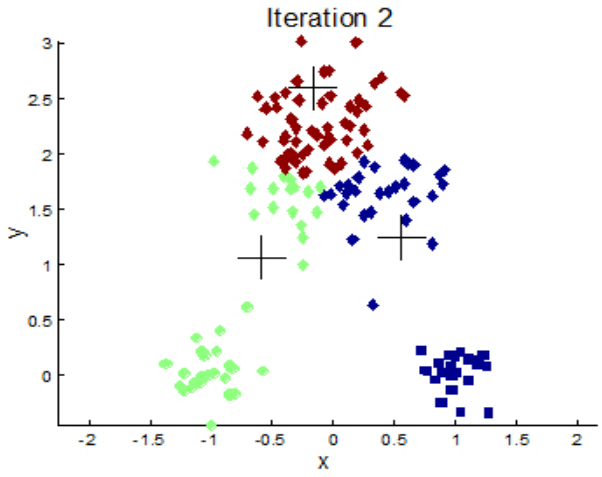


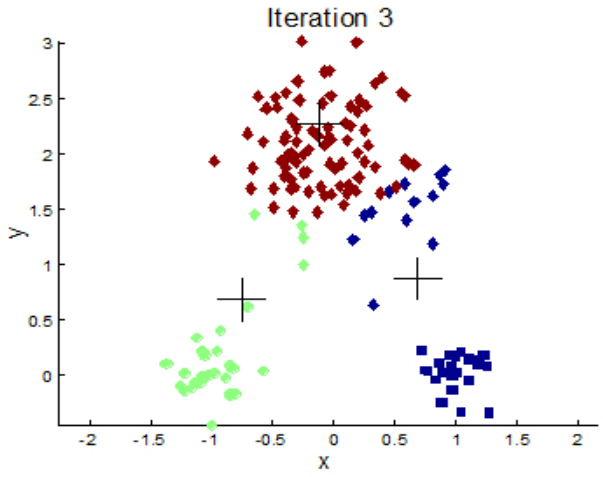
Source: www.edureka.in/data-science

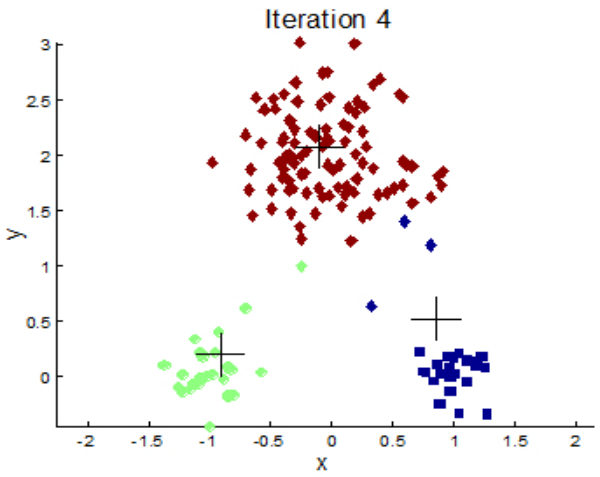
thanachart@imcinstitute.com

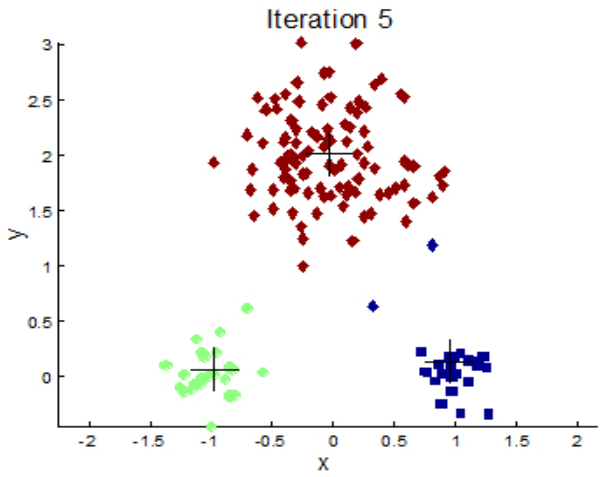
Example of K-Means Clustering

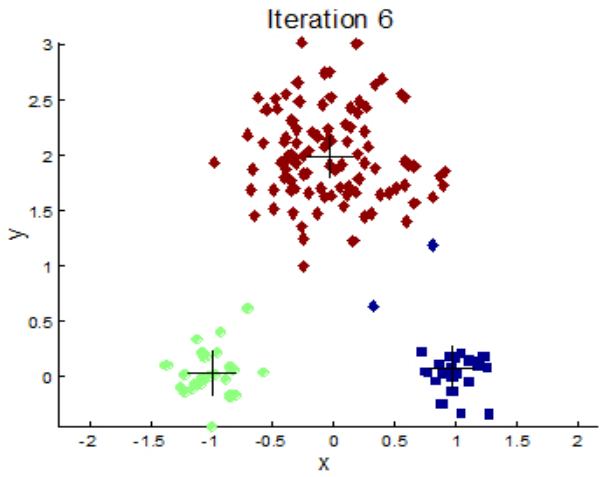








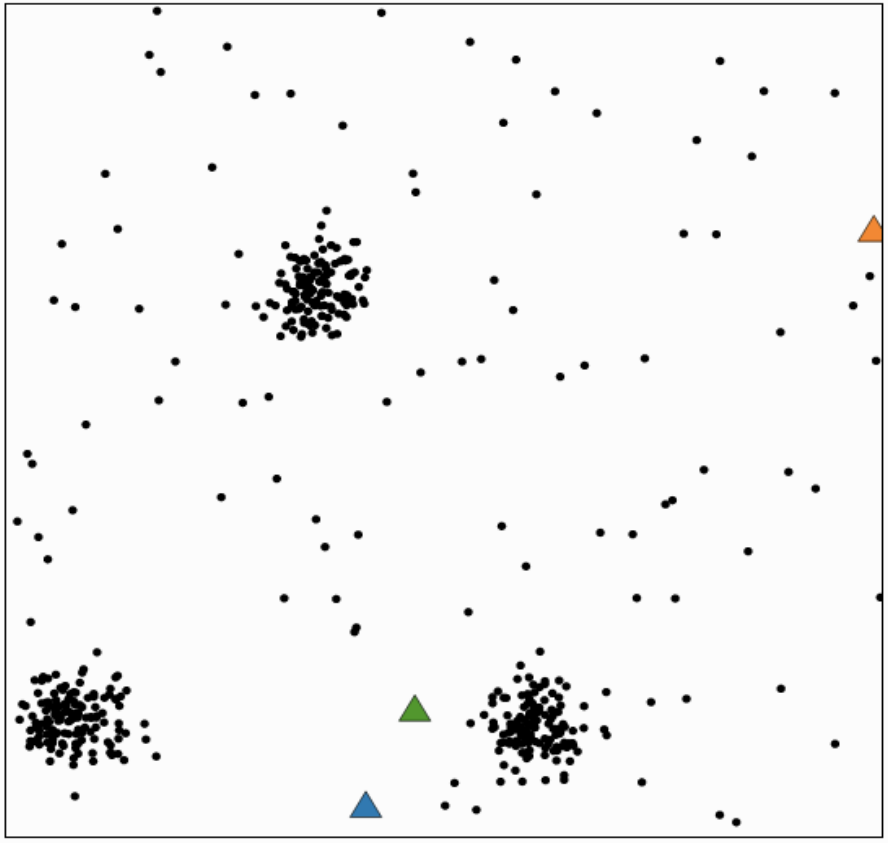




thanachart@imcinstitute.com

http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

Visualizing K-Means Clustering



Mean square point-centroid distance: not yet calculated

The k-means algorithm is an iterative method for clustering a set of N points (vectors) into k groups or clusters of points.

Algorithm

Repeat until convergence:

Find closest centroid

Find the closest centroid to each point, and group points that share the same closest centroid.

Update centroid

Update each centroid to be the mean of the points in its group.

Find closest centroid

Data

Clustered points — Random

Number of clusters : 3

Number of centroids: 3

New points | New centroids

Made by Karanveer Mohan for EE103. Source code on Github.

thanachart@imcinstitute.com

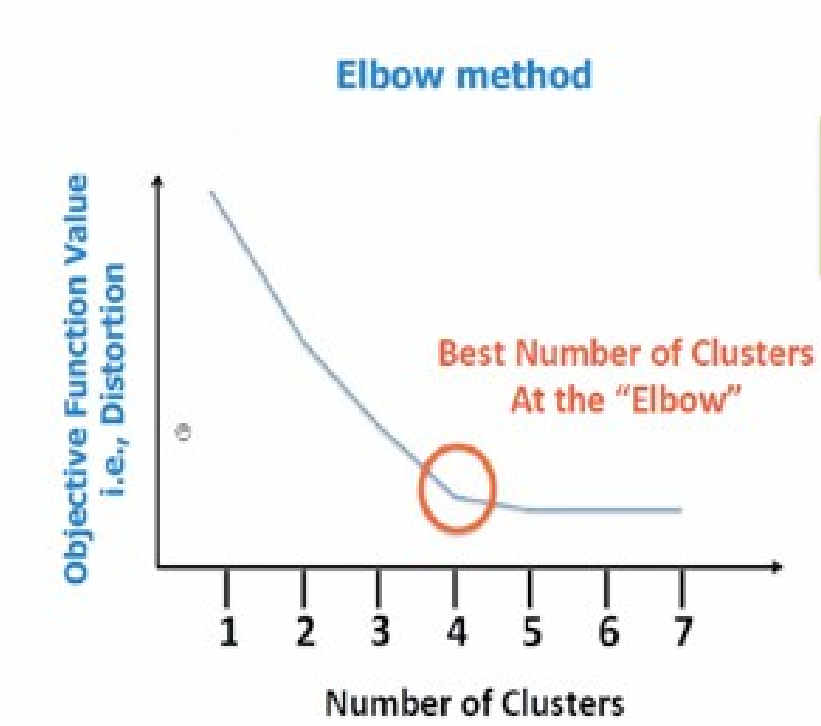
K-Means with different distance measures

Distance measure	Number of iterations	Vectors[a] in cluster 0	Vectors in cluster 1
EuclideanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
SquaredEuclideanDistanceMeasure	5	0, 1, 2, 3, 4	5, 6, 7, 8
ManhattanDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8
CosineDistanceMeasure	1	1	0, 2, 3, 4, 5, 6, 7, 8
TanimotoDistanceMeasure	3	0, 1, 2, 3, 4	5, 6, 7, 8

Source: Mahout in Action

thanachart@imcinstitute.com

Choosing number of clusters



$$Distortion = \sum_{i=1}^{m}(x_i - c_i)^2 = \sum_{j=1}^{k} \sum_{i \in OwnedBy(\mu_j)} (x_i - \mu_j)^2$$

(within cluster sum of squares)

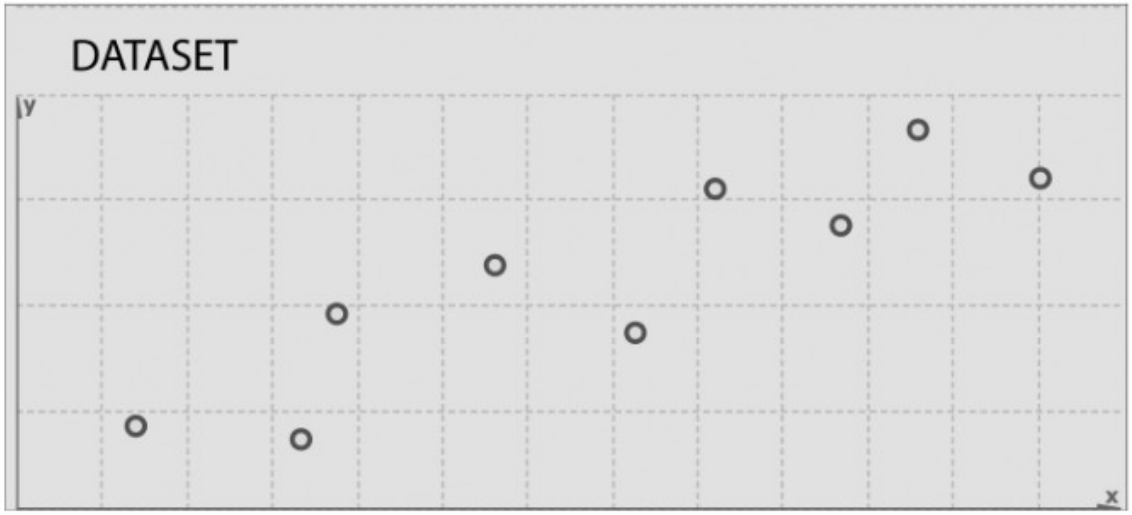
thanachart@imcinstitute.com

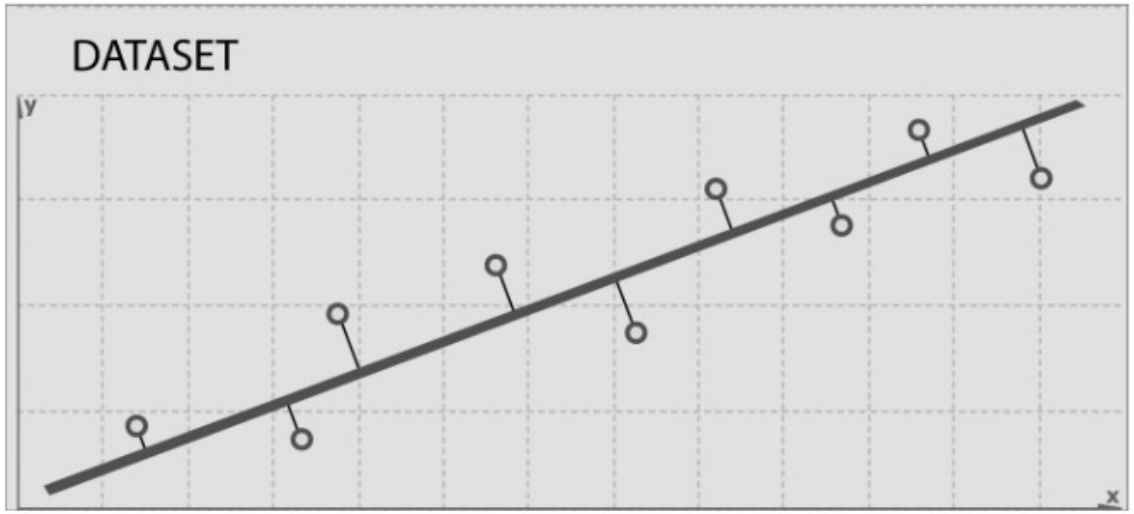
Dimensionality reduction

- Process of reducing the number of dimensions or features.
- Dimensionality reduction serves several purposes
  - Data compression
  - Visualization
- The most popular algorithm: Principal component analysis (PCA).

thanachart@imcinstitute.com

Dimensionality reduction





Source: Spark Cookbook

thanachart@imcinstitute.com

Dimensionality reduction with SVD

- Singular Value Decomposition (SVD): is based on a theorem from linear algebra that a rectangular matrix A can be broken down into a product of three matrices

$$A = USV^T$$

$$U^T U = 1$$

$$V^T V = 1$$

thanachart@imcinstitute.com

Dimensionality reduction with SVD

- The basic idea behind SVD
  - Take a high dimension, a highly variable set of data points
  - Reduce it to a lower dimensional space that exposes the structure of the original data more clearly and orders it from the most variation to the least.
- So we can simply ignore variation below a certain threshold to massively reduce the original data, making sure that the original relationship interests are retained.

thanachart@imcinstitute.com

Hands-on
Clustering on MovieLens Dataset

Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Extracting features from the MovieLens dataset

```
scala> val rawData =
sc.textFile("hdfs:///user/cloudera/movielens/u.item")

scala> println(movies.first)
```

```
1|Toy Story (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995
)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0
```

```
scala> val genres =
sc.textFile("hdfs:///user/cloudera/movielens/u.genre")

scala> genres.take(5).foreach(println)
```

```
unknown|0
Action|1
Adventure|2
Animation|3
Children's|4
```


Thanachart Numnonda, thanachart@imcinstitute.com June 2016

Extracting features from the MovieLens dataset (cont.)

```
scala> val genreMap = genres.filter(!_.isEmpty).map(line =>
line.split("\\|")).map(array=> (array(1),
array(0))).collectAsMap
```

```
genreMap: scala.collection.Map[String,String] = Map(2 -> Adventure, 5 -> Comedy, 12
-> Musical, 15 -> Sci-Fi, 8 -> Drama, 18 -> Western, 7 -> Documentary, 17 -> War, 1
-> Action, 4 -> Children's, 11 -> Horror, 14 -> Romance, 6 -> Crime, 0 -> unknown, 9
 -> Fantasy, 16 -> Thriller, 3 -> Animation, 10 -> Film-Noir, 13 -> Mystery)
```

Extracting features from the MovieLens dataset (cont.)

```
scala> val titlesAndGenres = movies.map(_.split("\\|")).map
{ array =>
  val genres = array.toSeq.slice(5, array.size)
  val genresAssigned = genres.zipWithIndex.filter { case (g,
idx) =>
    g == "1"
  }.map { case (g, idx) =>
    genreMap(idx.toString)
  }
  (array(0).toInt, (array(1), genresAssigned))
}
```

```
scala> println(titlesAndGenres.first)
(1,(Toy Story (1995),ArrayBuffer(Animation, Children's, Comedy)))
```

Training the recommendation model

```
scala> :paste
import org.apache.spark.mllib.recommendation.ALS
import org.apache.spark.mllib.recommendation.Rating
val rawData =
sc.textFile("hdfs:///user/cloudera/movielens/u.data")
val rawRatings = rawData.map(_.split("\t").take(3))
val ratings = rawRatings.map{ case Array(user, movie,
rating) => Rating(user.toInt, movie.toInt,
rating.toDouble) }
ratings.cache
val alsModel = ALS.train(ratings, 50, 10, 0.1)
import org.apache.spark.mllib.linalg.Vectors
val movieFactors = alsModel.productFeatures.map { case (id,
factor) => (id, Vectors.dense(factor)) }
val movieVectors = movieFactors.map(_._2)
val userFactors = alsModel.userFeatures.map { case (id,
factor) => (id, Vectors.dense(factor)) }
val userVectors = userFactors.map(_._2)
```

Normalization

```
scala> :paste

import org.apache.spark.mllib.linalg.distributed.RowMatrix

val movieMatrix = new RowMatrix(movieVectors)

val movieMatrixSummary =
movieMatrix.computeColumnSummaryStatistics()

val userMatrix = new RowMatrix(userVectors)

val userMatrixSummary =

userMatrix.computeColumnSummaryStatistics()

println("Movie factors mean: " + movieMatrixSummary.mean)

println("Movie factors variance: " +
movieMatrixSummary.variance)

println("User factors mean: " + userMatrixSummary.mean)

println("User factors variance: " +
userMatrixSummary.variance)
```

Output from Normalization

```
Movie factors mean:
[0.28047737659519767,0.26886479057520024,0.2935579964446398,0.27821738264113755,
...
Movie factors variance:
[0.038242041794064895,0.03742229118854288,0.044116961097355877,0.057116244055791986
, ...
User factors mean:
[0.2043520841572601,0.22135773814655782,0.2149706318418221,0.23647602029329481, ...
User factors variance:
[0.037749421148850396,0.02831191551960241,0.032831876953314174,0.036775110657850954
, ...
```

Training a clustering model

```
val userClusterModel = KMeans.train(userVectors, numClusters, numIterations,
numRuns)
```

```
scala> import org.apache.spark.mllib.clustering.KMeans
scala> val numClusters = 5
scala> val numIterations = 10
scala> val numRuns = 3
scala> val movieClusterModel = KMeans.train(movieVectors,
numClusters, numIterations, numRuns)
```

Making predictions using a clustering model

```
scala> val movie1 = movieVectors.first

scala> val movieCluster = movieClusterModel.predict(movie1)

scala> val predictions =
movieClusterModel.predict(movieVectors)
```

```
scala> println(predictions.take(10).mkString(","))
3,0,0,0,0,0,0,2,0,0
```

Interpreting cluster predictions

```
scala> :paste

import breeze.linalg._

import breeze.numerics.pow

def computeDistance(v1: DenseVector[Double], v2:
DenseVector[Double]) = pow(v1 - v2, 2).sum

val titlesWithFactors = titlesAndGenres.join(movieFactors)

val moviesAssigned = titlesWithFactors.map { case (id,
((title, genres), vector)) =>

  val pred = movieClusterModel.predict(vector)

  val clusterCentre = movieClusterModel.clusterCenters(pred)

  val dist =
computeDistance(DenseVector(clusterCentre.toArray),
DenseVector(vector.toArray))

  (id, title, genres.mkString(" "), pred, dist)

}
```

Interpreting cluster predictions (cont.)

```
val clusterAssignments = moviesAssigned.groupBy { case (id,
title, genres, cluster, dist) => cluster }.collectAsMap

for ( (k, v) <- clusterAssignments.toSeq.sortBy(_._1)) {

  println(s"Cluster $k:")

  val m = v.toSeq.sortBy(_._5)

  println(m.take(20).map { case (_, title, genres, _, d) =>
(title, genres, d) }.mkString("\n"))

  println("=====\n")

}
```

```
Cluster 0:
(Last Time I Saw Paris, The (1954),Drama,0.15088816303186323)
(Witness (1985),Drama Romance Thriller,0.2018937474956098)
(Substance of Fire, The (1996),Drama,0.26580331444304967)
(King of the Hill (1993),Drama,0.27090751738692787)
(Mamma Roma (1962),Drama,0.30508676553769926)
(Beans of Egypt, Maine, The (1994),Drama,0.31880331503649484)
(Scream of Stone (Schrei aus Stein) (1991),Drama,0.33904627647373703)
(All Things Fair (1996),Drama,0.3449680047501059)
(Angel and the Badman (1947),Western,0.3519092167012976)
(Nelly & Monsieur Arnaud (1995),Drama,0.36300591397764 54)
(Cosi (1996),Comedy,0.3781303586431162)
(Object of My Affection, The (1998),Comedy Romance,0.39398318062694987)
(Wife, The (1995),Comedy Drama,0.399375163806288)
(They Made Me a Criminal (1939),Crime Drama,0.42158316491602227)
(Spellbound (1945),Mystery Romance Thriller,0.42881078192699107)
(Spirits of the Dead (Tre passi nel delirio) (1968),Horror,0.43991392186284806)
(Farewell to Arms, A (1932),Romance War,0.44324604591789385)
(Sleepover (1995),Comedy Drama,0.4473239416648149)
(Love Is All There Is (1996),Comedy Drama,0.4473239416648149)
(Century (1993),Drama,0.4473239416648149)
```

Thank you

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