Module 10

Introduction to Apache Spark

Thanachart Numnonda, Executive Director, IMC Institute

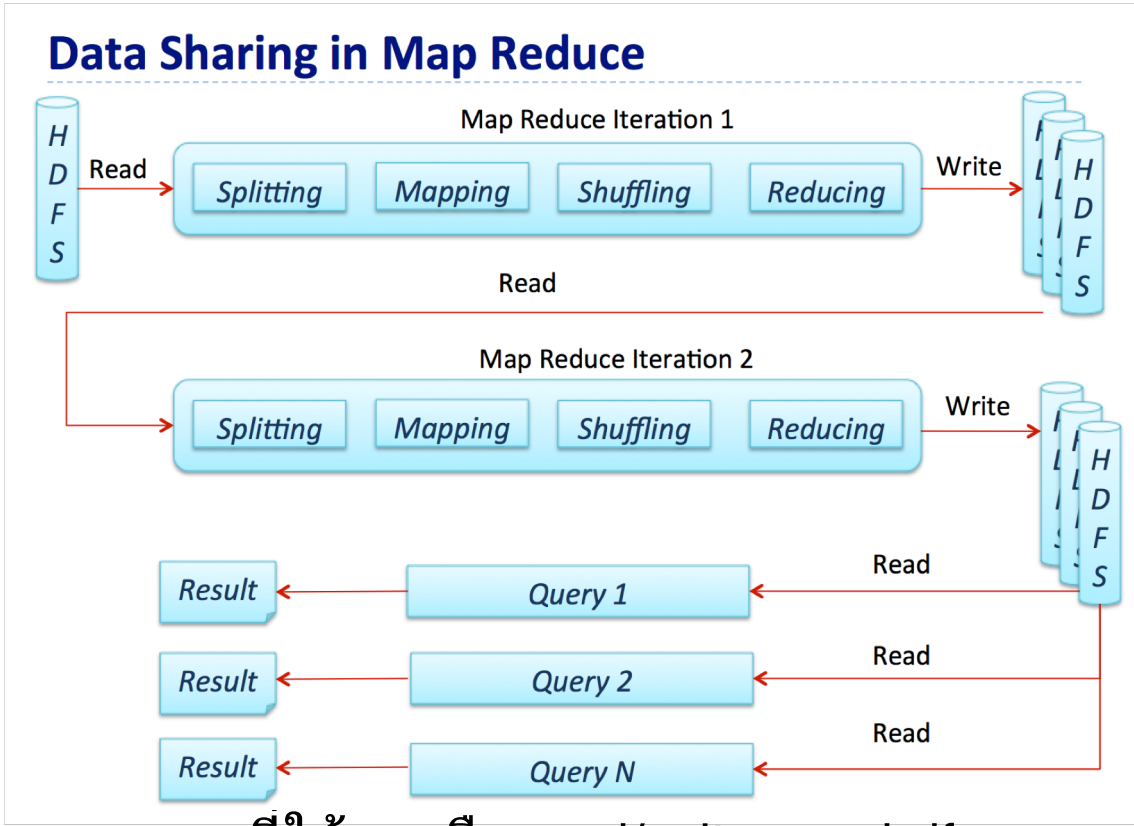
Mr.Aekanun Thongtae, Big Data Consultant, IMC Institute

Thanisa Numnonda, Faculty of Information Technology,

King Mongkut's Institute of Technology Ladkrabang

Typical Hadoop Job: MapReduce

map มันช้า เพราะอาจจะ ทำหลายเครื่อง



เวลาที่ใช้เยอะคือ read/write ตอน hdfs

Source: dzone.com

Issues

1. **Streaming data processing to perform near real-time analysis.**

เป็นไปไม่ได้เลยที่จะได้เวลา real time

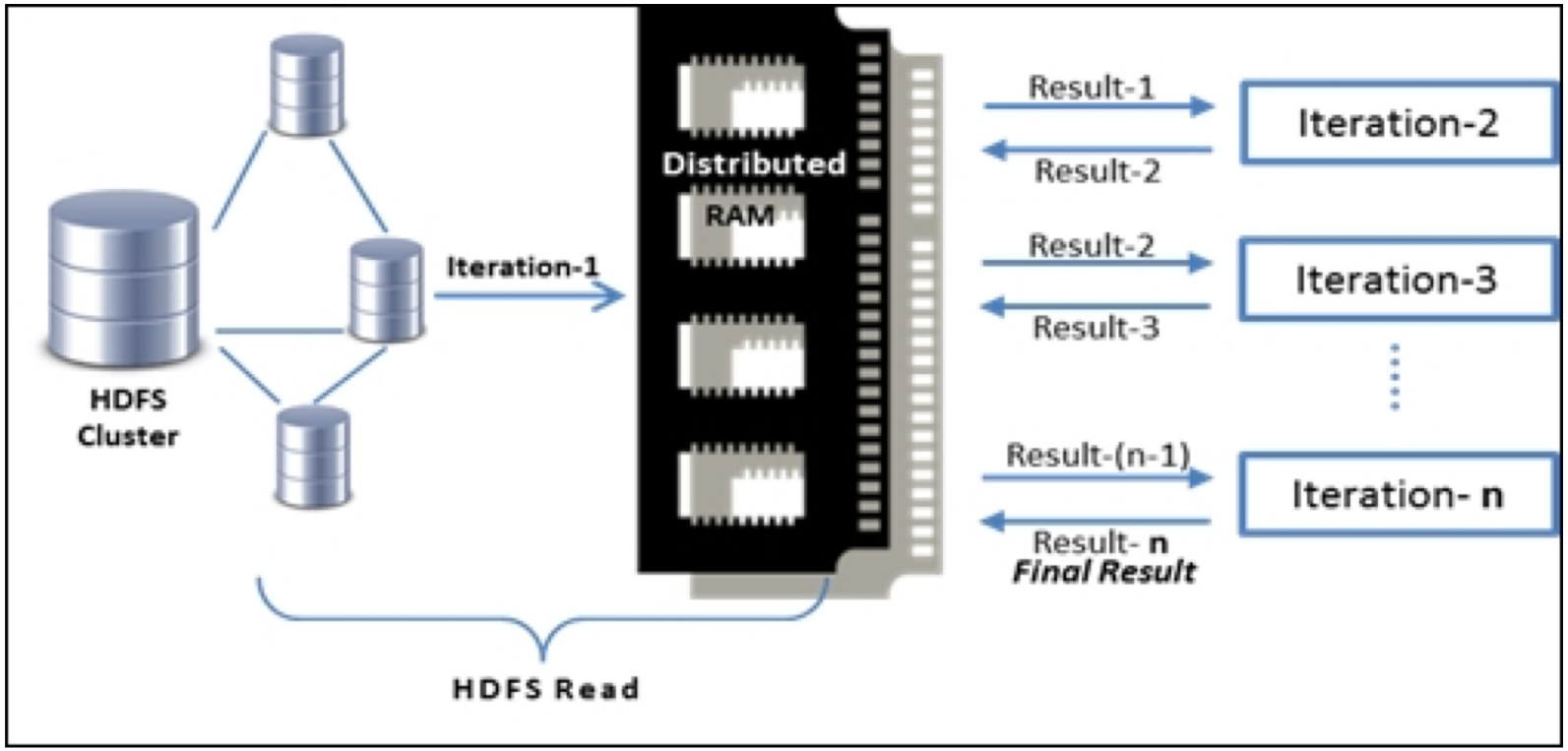
2. **Interactive querying of large datasets so a data scientist may run ad-hoc queries on data set.**

Evolution of big data analytics

Spark: Instead of redesigning all the algorithms, a general-purpose engine was needed that could be leveraged by most of the algorithms for in-memory computation on a distributed computing platform.

cache

ไม่ต้อง read write จาก hdfs แต่จะทำบน cache เป็นหลัก ประหยัดเวลา



Source: Srinivas Duvvuri; Bikramaditya, “SinghalSpark for Data Science,” Packt Publishing, 2016

Ideal Solution for Big Data Analytics

100 เท่า จาก map

hive
Batch

spark แบบ all in one
Single
Framework

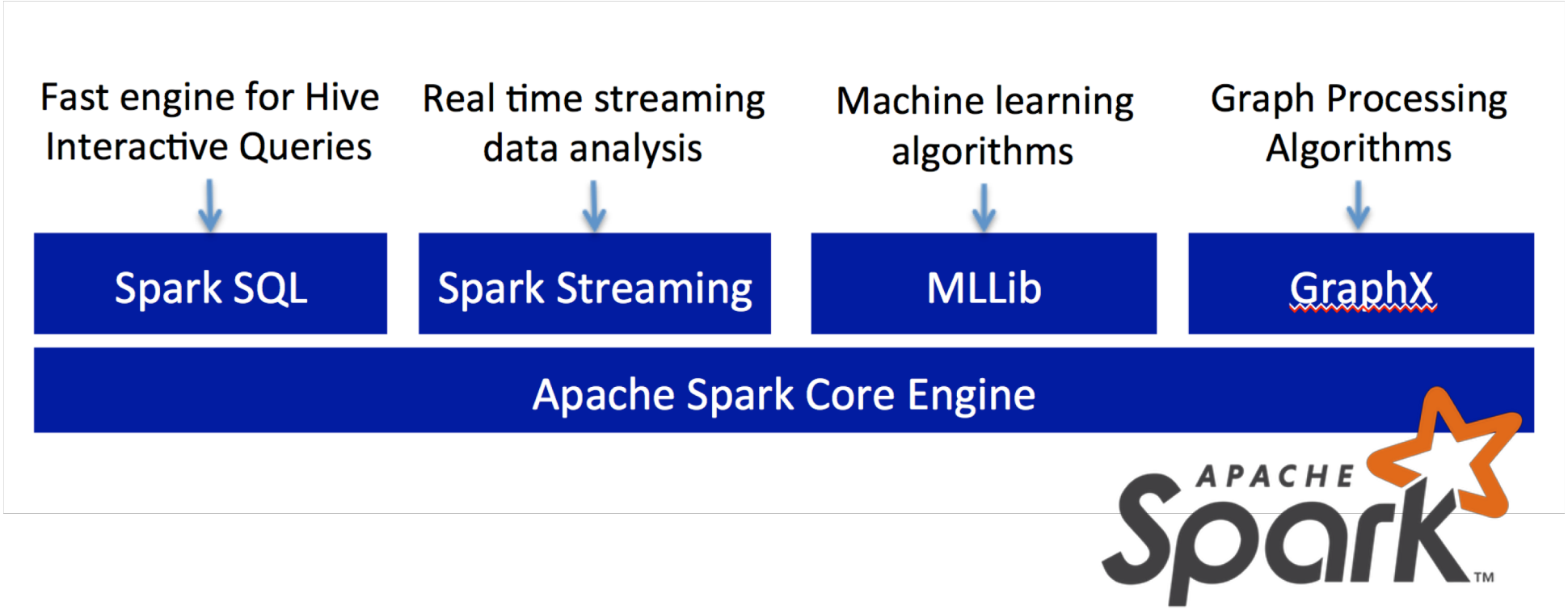
storm(ยังไม่ได้เรียน)
Streaming

impala faster
Interactive

Source: dzone.com

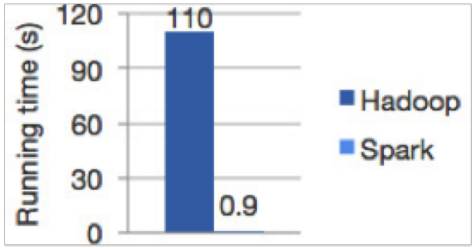
Various Components of Apache Spark

ทำได้หลายอย่างเพราะ มีหลาย module



Source: dzone.com

A fast and general engine for large scale data processing



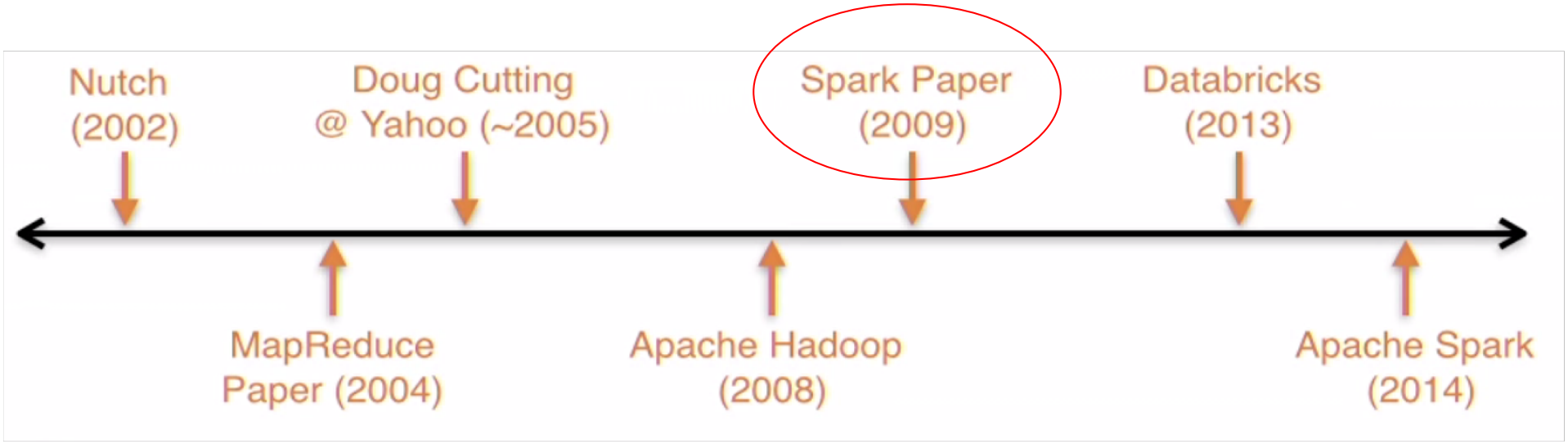
An open source big data processing framework built around speed, ease of use, and sophisticated analytics.

Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times faster even when running on disk.

speed,
simple(write many lang ex. python, java),
support ติดต่อกับ data soure ได้หลาย มี api รองรับ

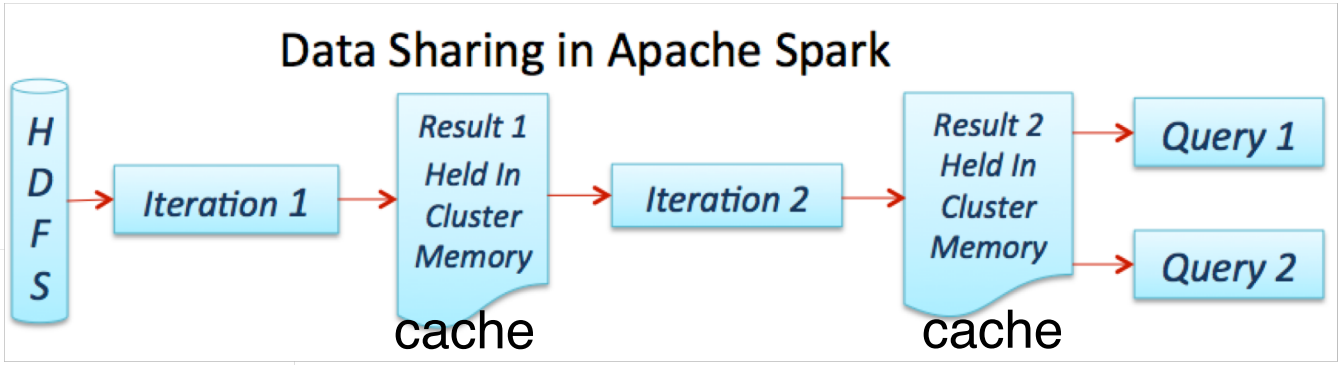
Spark: History

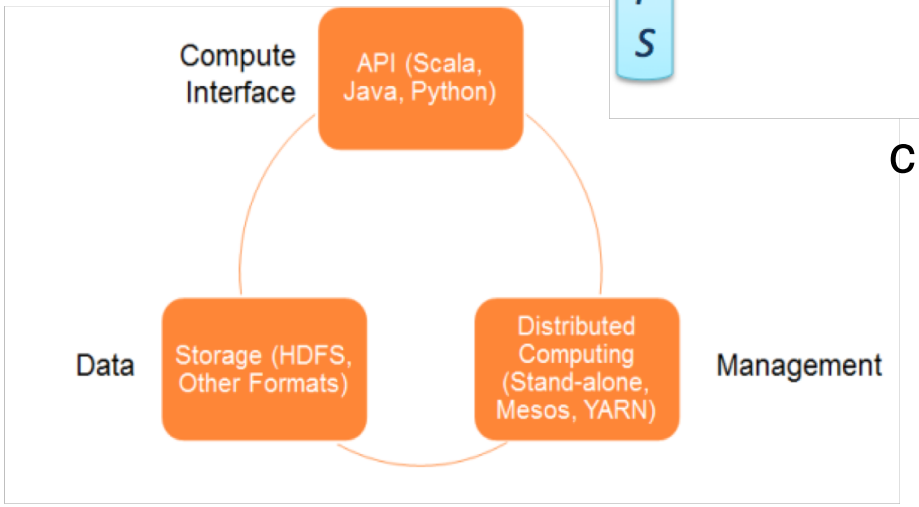
- Founded by AMPlab, UC Berkeley
- Created by Matei Zaharia (PhD Thesis)
- Maintained by Apache Software Foundation
- Commercial support by Databricks



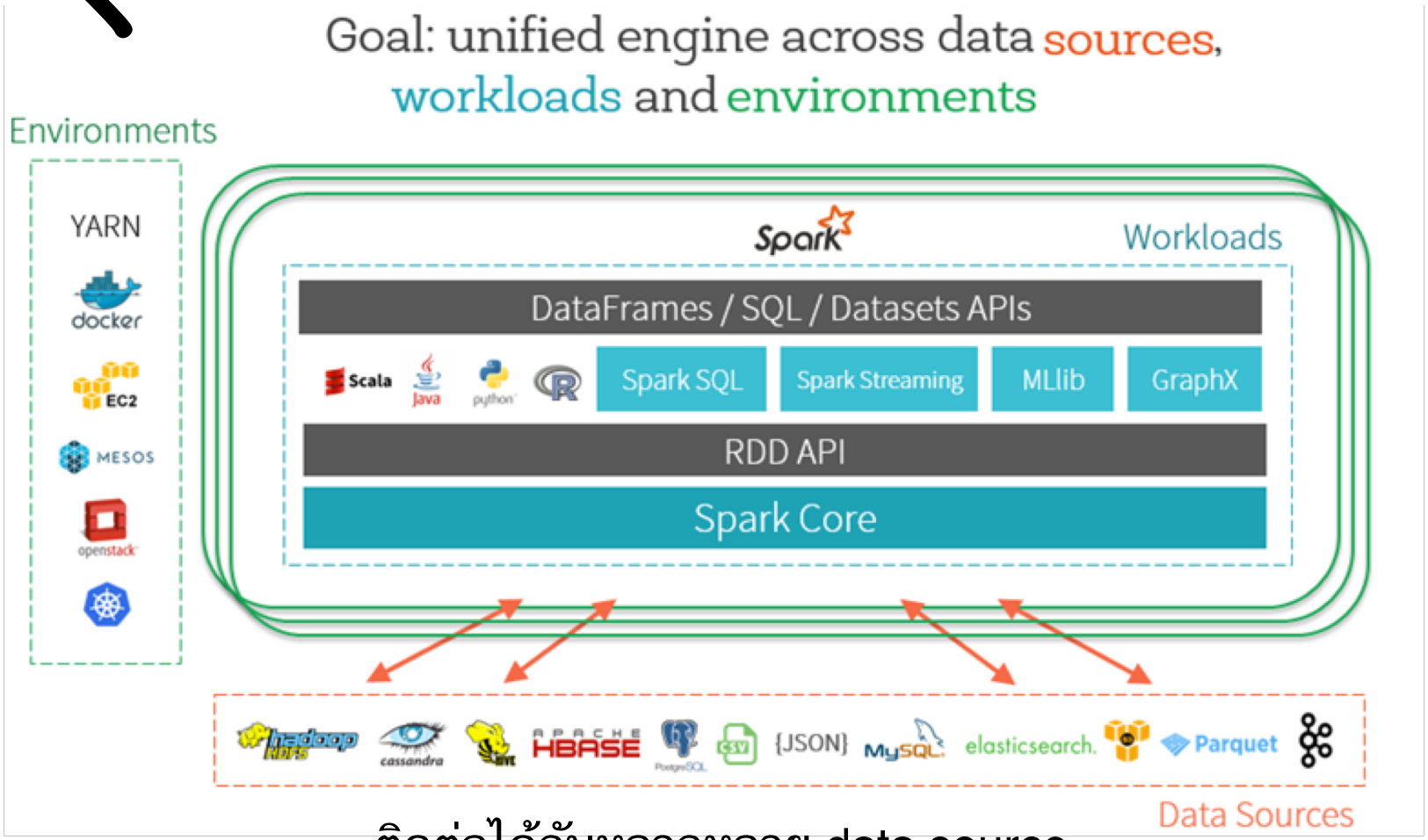
What is Spark?

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



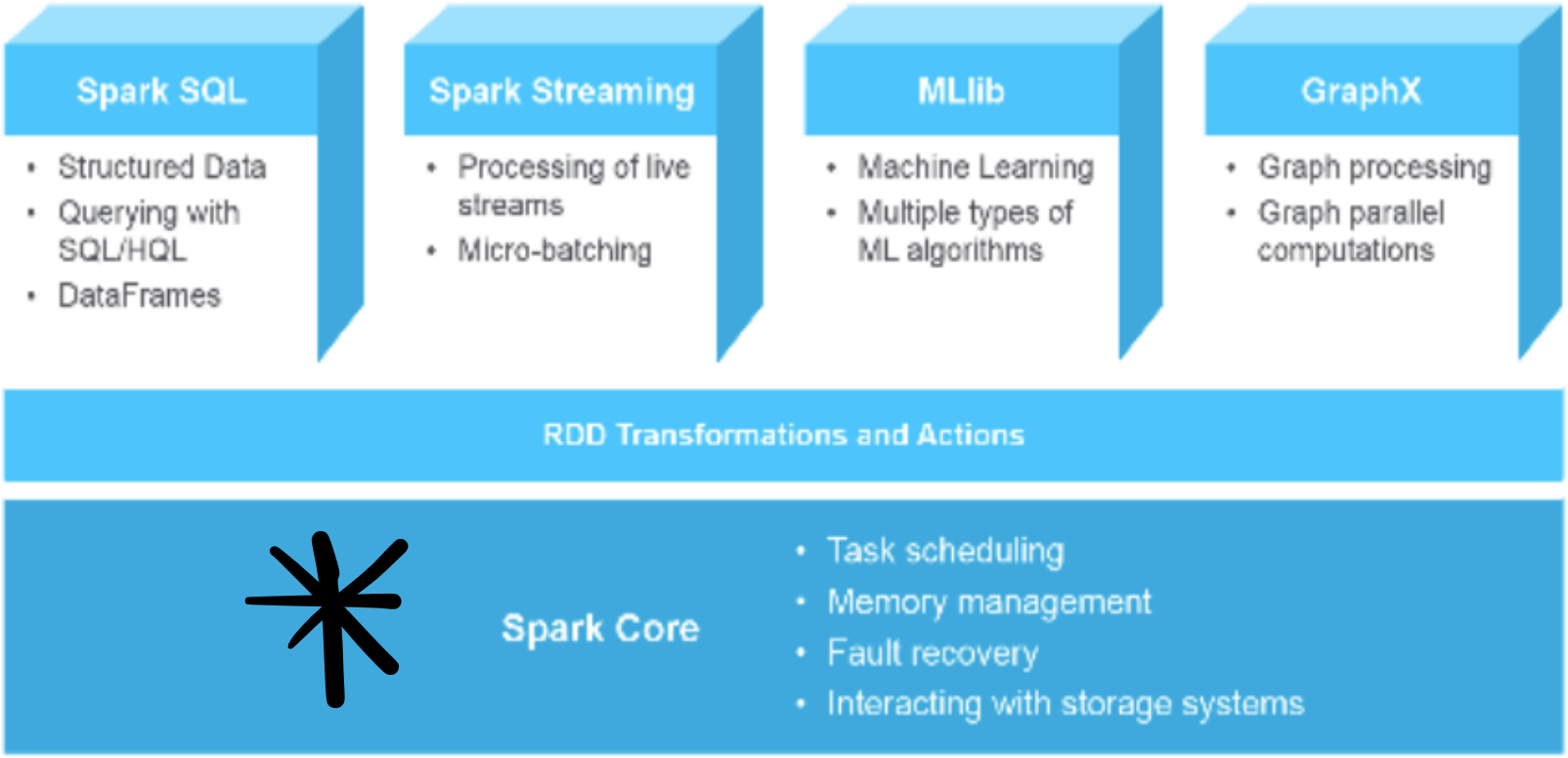


Source: dzone.com



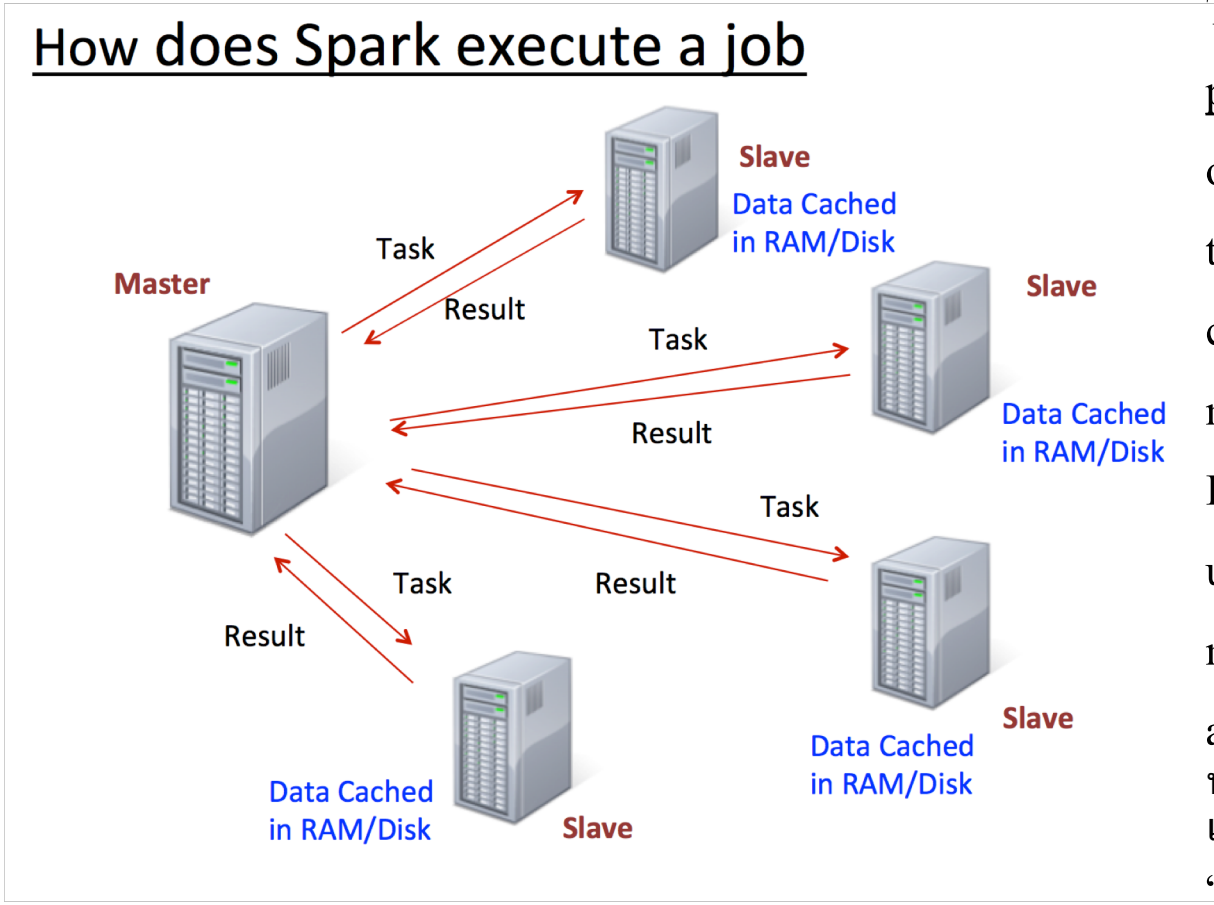
ติดต่อได้กับหลากหลาย data source

Spark Platform



เป็นตัวกลางในการ manage memory จะทำไรก้ต้องใช้

Source: MapR Academy



1. ส่งข้อมูลไปเก็บในแต่ละเครื่อง
2. ส่งงานแล้วก้ประม้วนผลงานในแต่ละเครื่องกับข้อมูลที่มีอยู่ในแต่ละเครื่อง

Source: dzone.com

The Master controls how data is partitioned, and it takes advantage of data locality while keeping track of all the distributed data computation on the Slave machines.

แบ่งข้อมูล

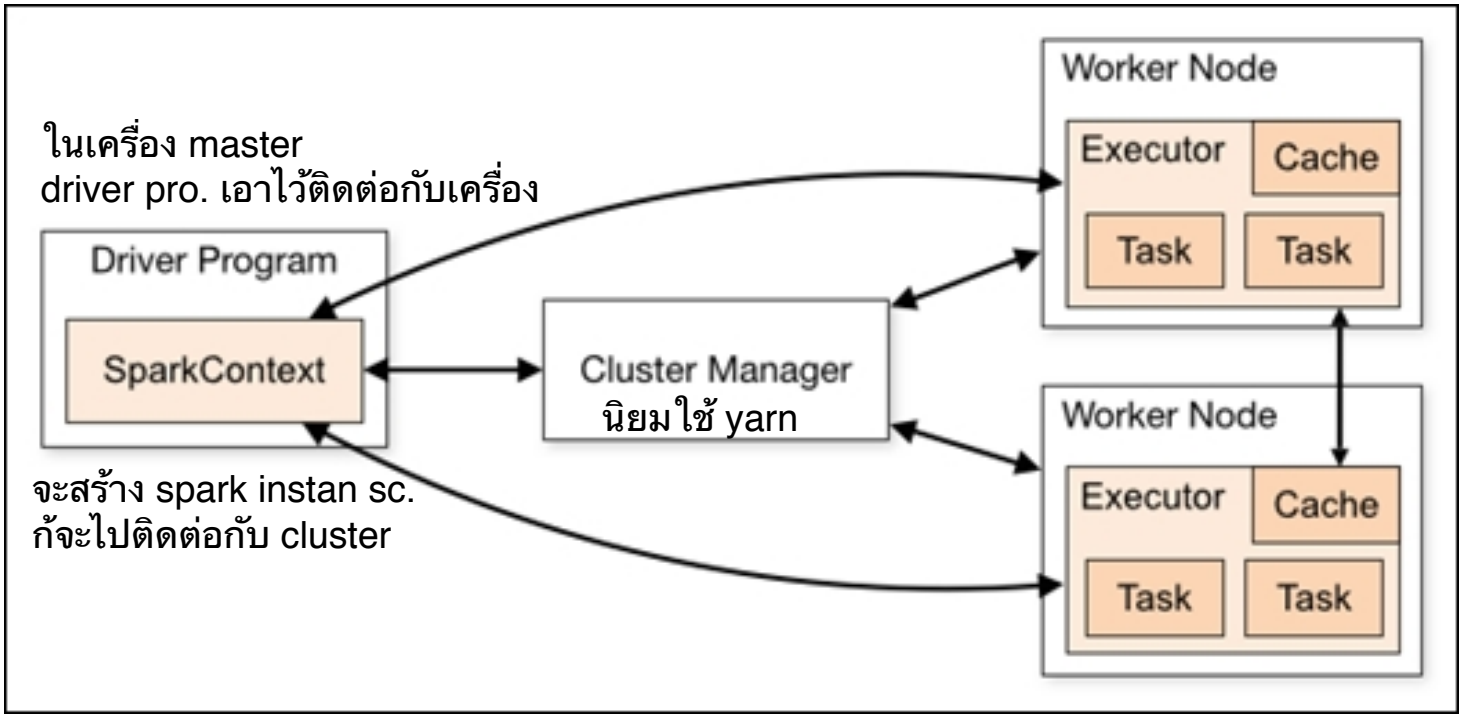
ข้อมูลอยู่ในแต่ละเครื่อง

If a certain Slave machine is unavailable, the data on that machine is reconstructed on other available machine(s).

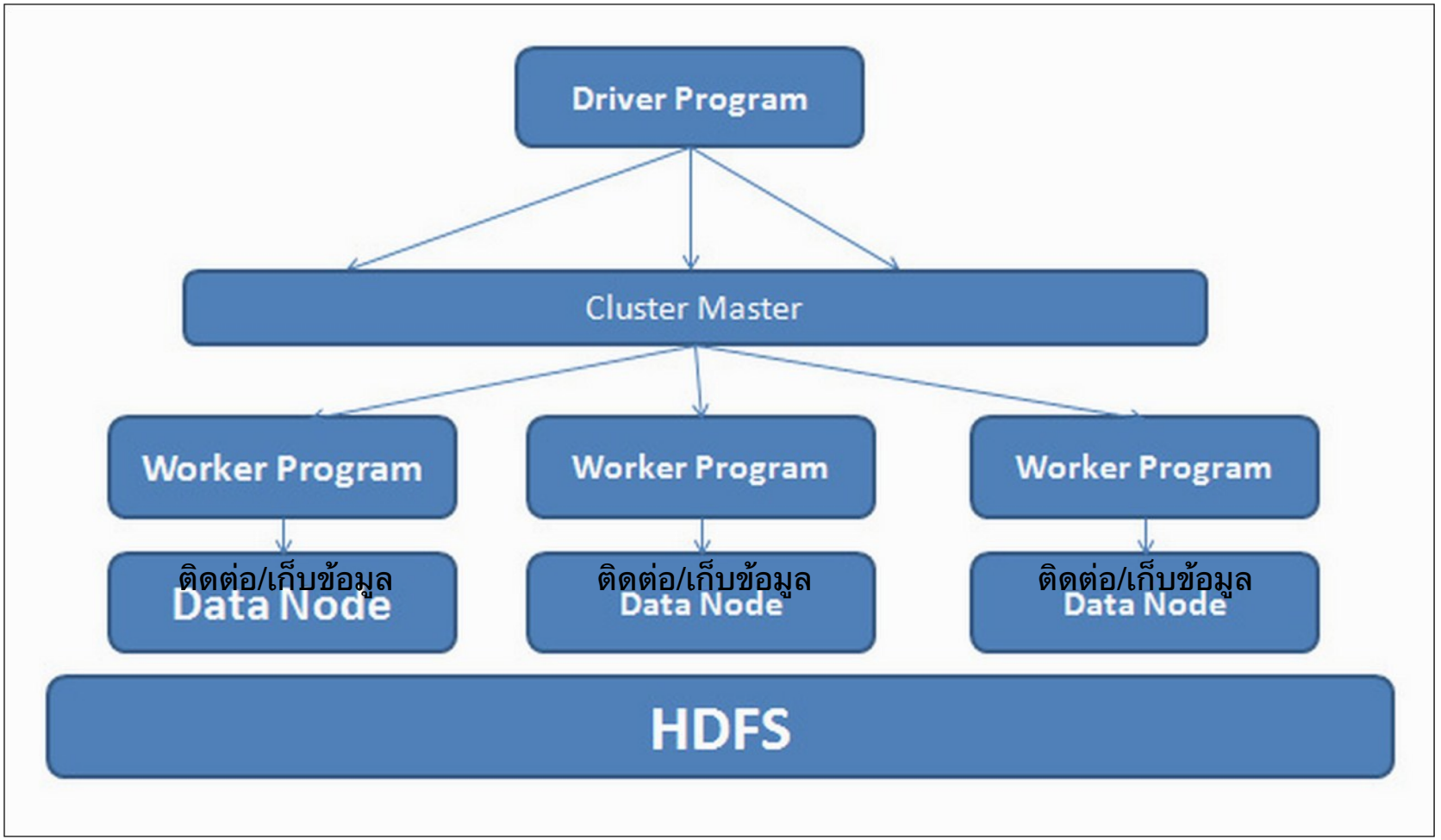
ทดทานต่อความผิดพลาด หากเครื่องนึงพังก้ย้ายไปที่เครื่องอื่นที่ว่าง

"Master" is currently a single point of failure, but it will be fixed in upcoming releases.

The Spark engine

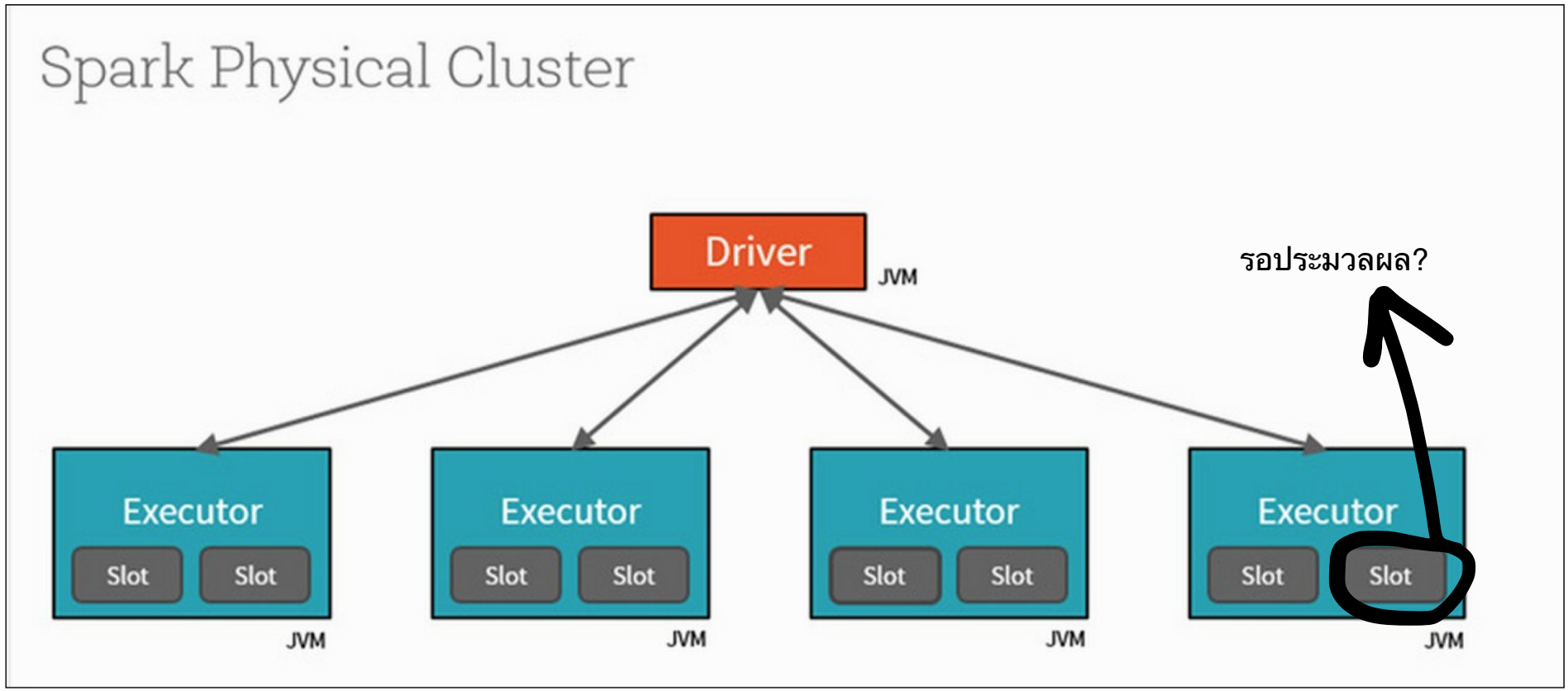


Apache Spark



Source: Suresh Kumar Gorakala, “Building Recommendation,” Packt Publishing, 2016

Apache Spark



Source: Suresh Kumar Gorakala, “Building Recommendation,” Packt Publishing, 2016

What is a RDD? ชนิดข้อมูลที่ใช้ใน spark

java, python, scala

- 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 across the cluster.
- Dataset: initial data can come from a table or be created programmatically รูปแบบ unstructure

ไม่สามารถเขียน sql และ r ได้ เพราะ มันต้องใช้ข้อมูลเป็นโครงสร้าง แล้ว d ตัวสุดท้าย มันมีหลักแบบ un

RDD:

- Fault tolerance
- Immutable
- Three methods for creating RDD:
  - Parallelizing an existing correction sc. parallelize
  - Referencing a dataset sc.
  - Transformation from an existing RDD filter.
- Types of files supported:
  - Text files
  - Sequence Files
  - Hadoop InputFormat

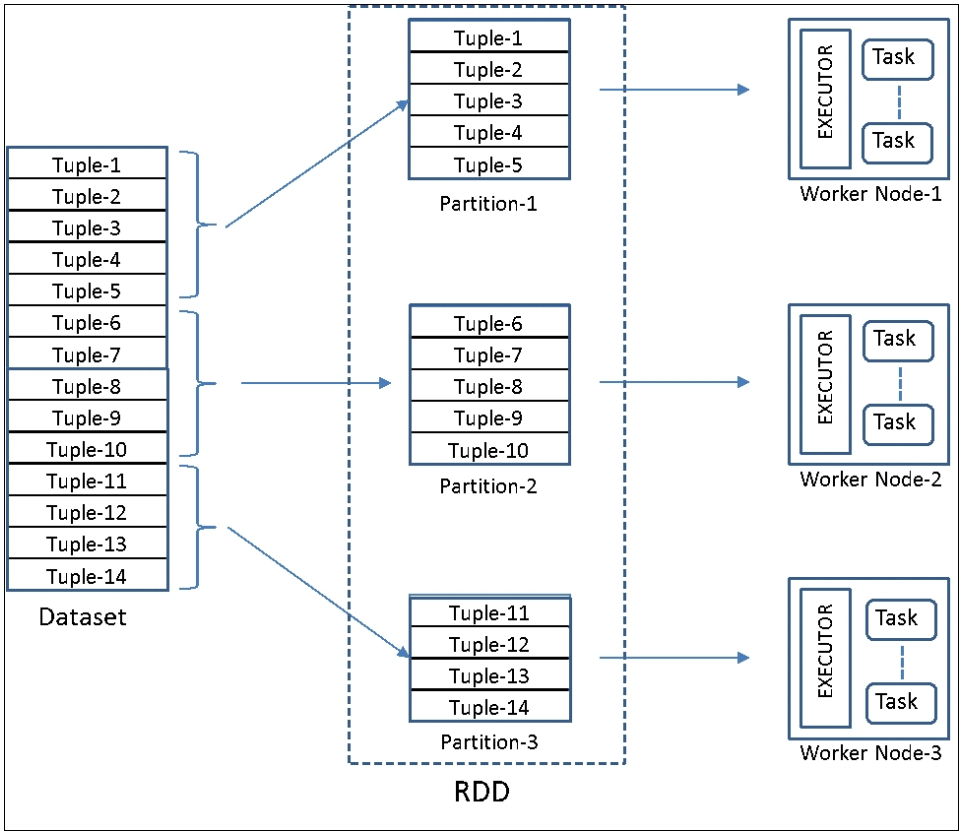
RDD & Partition

มันจะสั่งแบบ auto

- **Though Spark sets the number of partitions automatically based on the cluster, we have the liberty to set it manually by passing it as a second argument to the parallelize function (for example, sc.parallelize(data, 3)).**

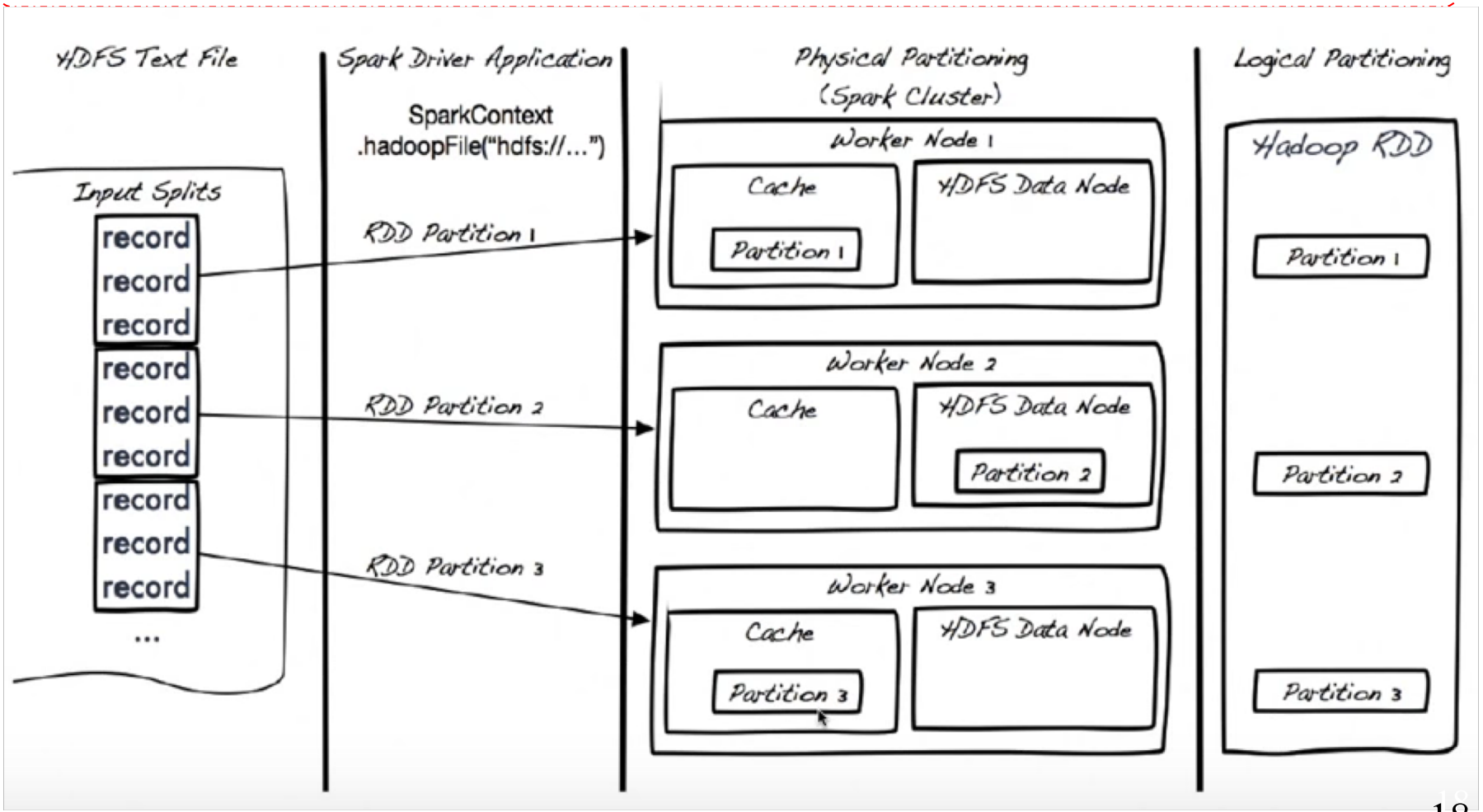
  สร้างเองได้ด้วยคำสั่งนี้

- **A diagrammatic representation of an RDD which is created with a dataset with, say, 14 records (or tuples) and is partitioned into 3, distributed across 3 nodes:**



RDD Creation

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

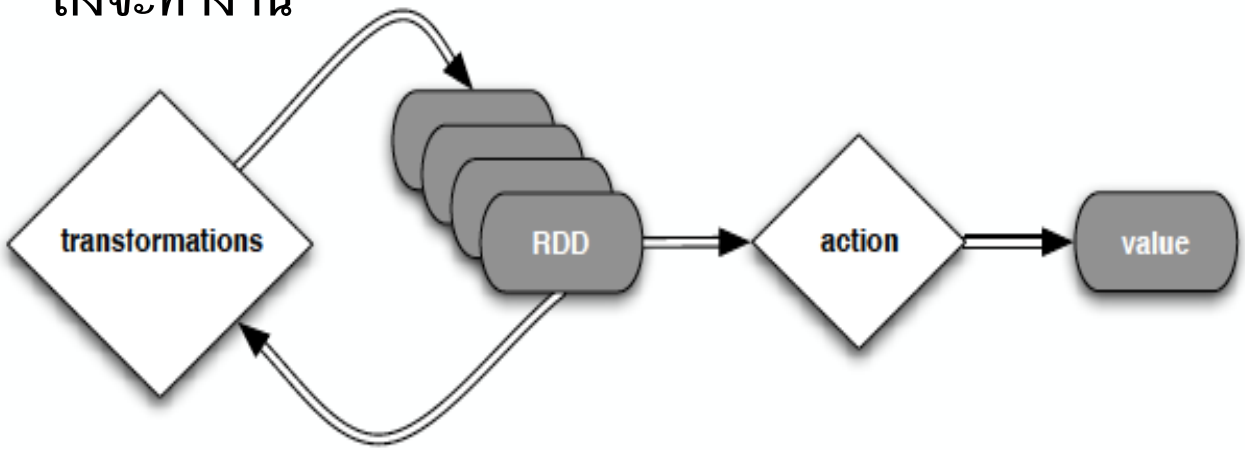


Source: Pyspark: A brain-friendly introduction

RDD: Operations

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

มันจะไม่ทำทันที รอก่อน จนกว่าจะเจอ action(collect , take) ถึงจะทำงาน



Direct Acyclic Graph (DAG)

เก็บประวัติคำสั่งเอาไว้ มีปัญหาไรจะได้ roll back กลับมาได้

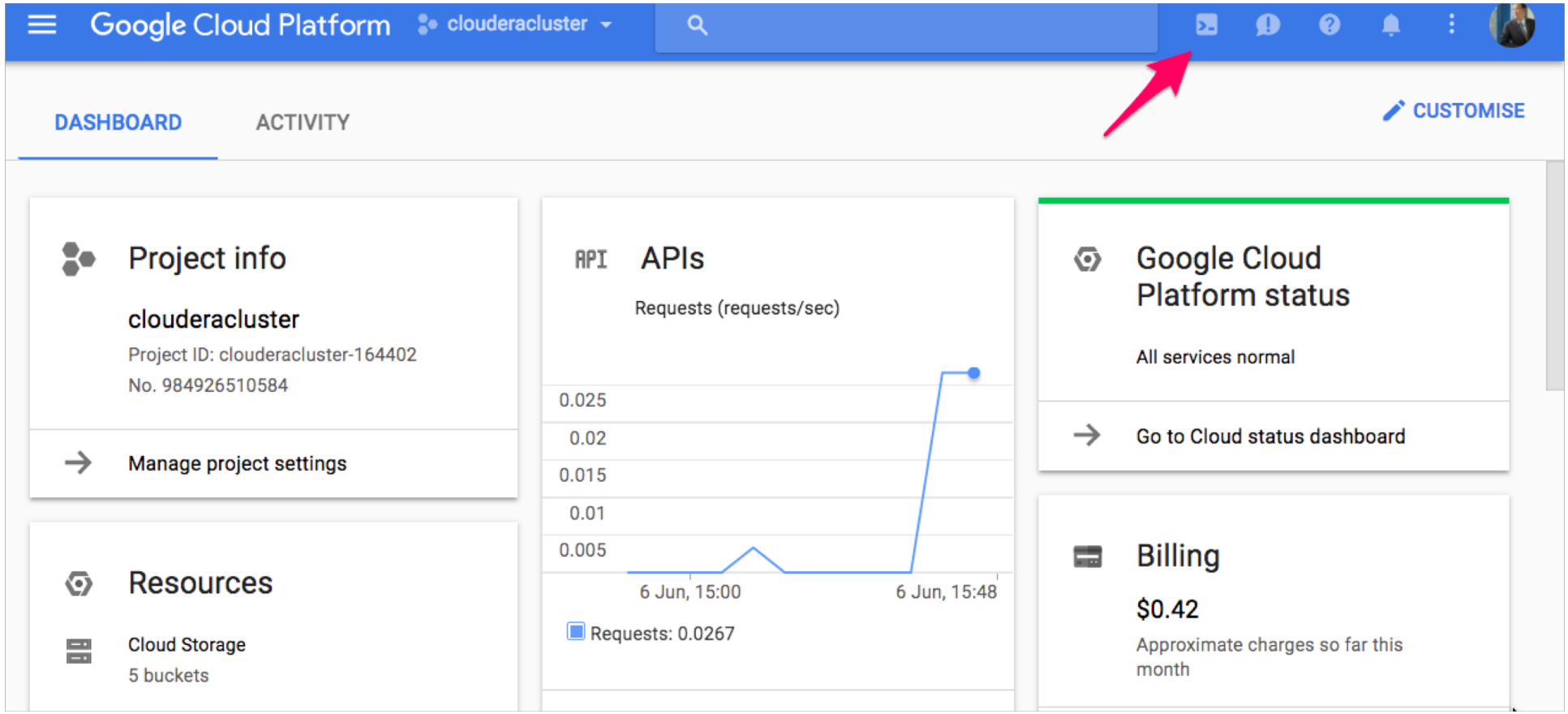
- View the DAG
  linesLength.toDebugString
- Sample DAG

```
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)
```

Hands-On: Prepare a Dataproc Cluster

Launch Dataproc using gcloud command:

with Jupyter Notebook I) Launch Cloud Shell



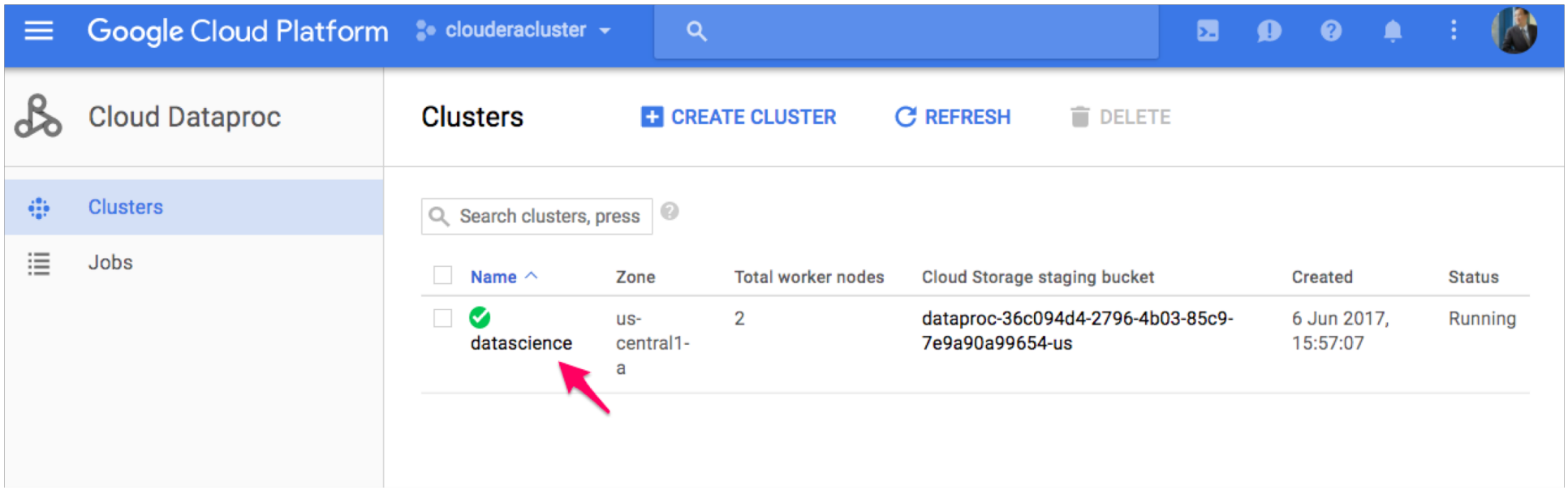
Launch Dataproc using gcloud command

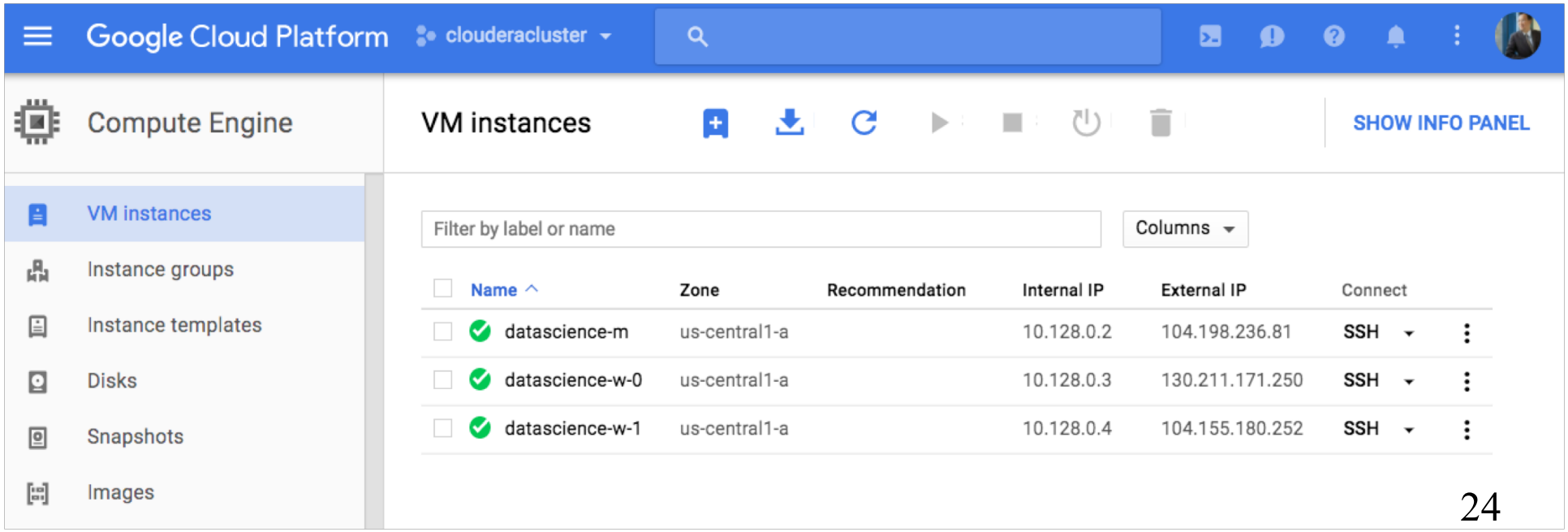
with Jupyter Notebook

II) Type the following command

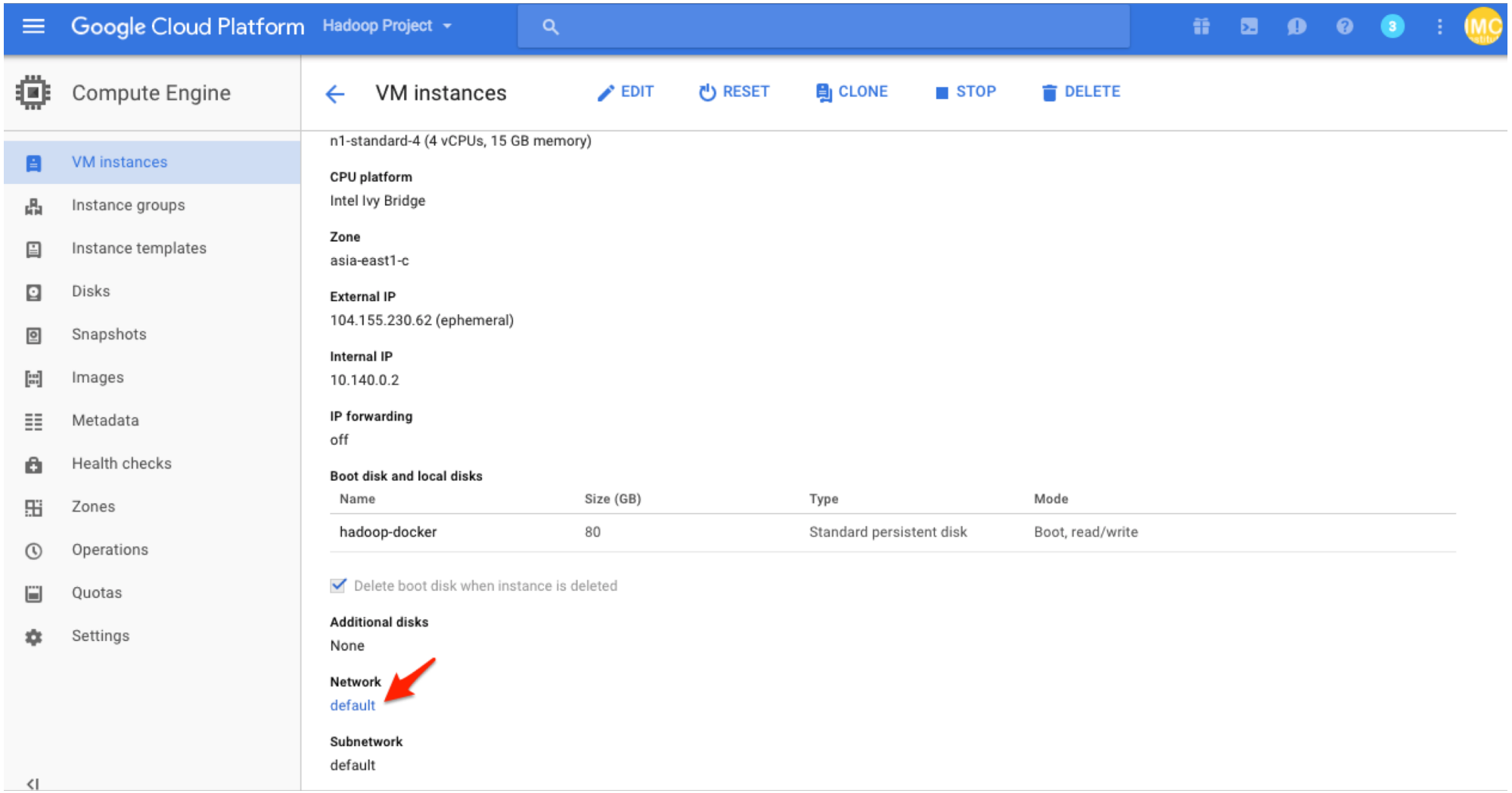
- gcloud dataproc clusters create **datascience** \

```
  --zone us-central1-a \
  --master-machine-type=n1-standard-2 \
  --worker-machine-type=n1-standard-2 \
   --initialization-actions \
      gs://dataproc-initialization-actions/jupyter/jupyter.sh
```

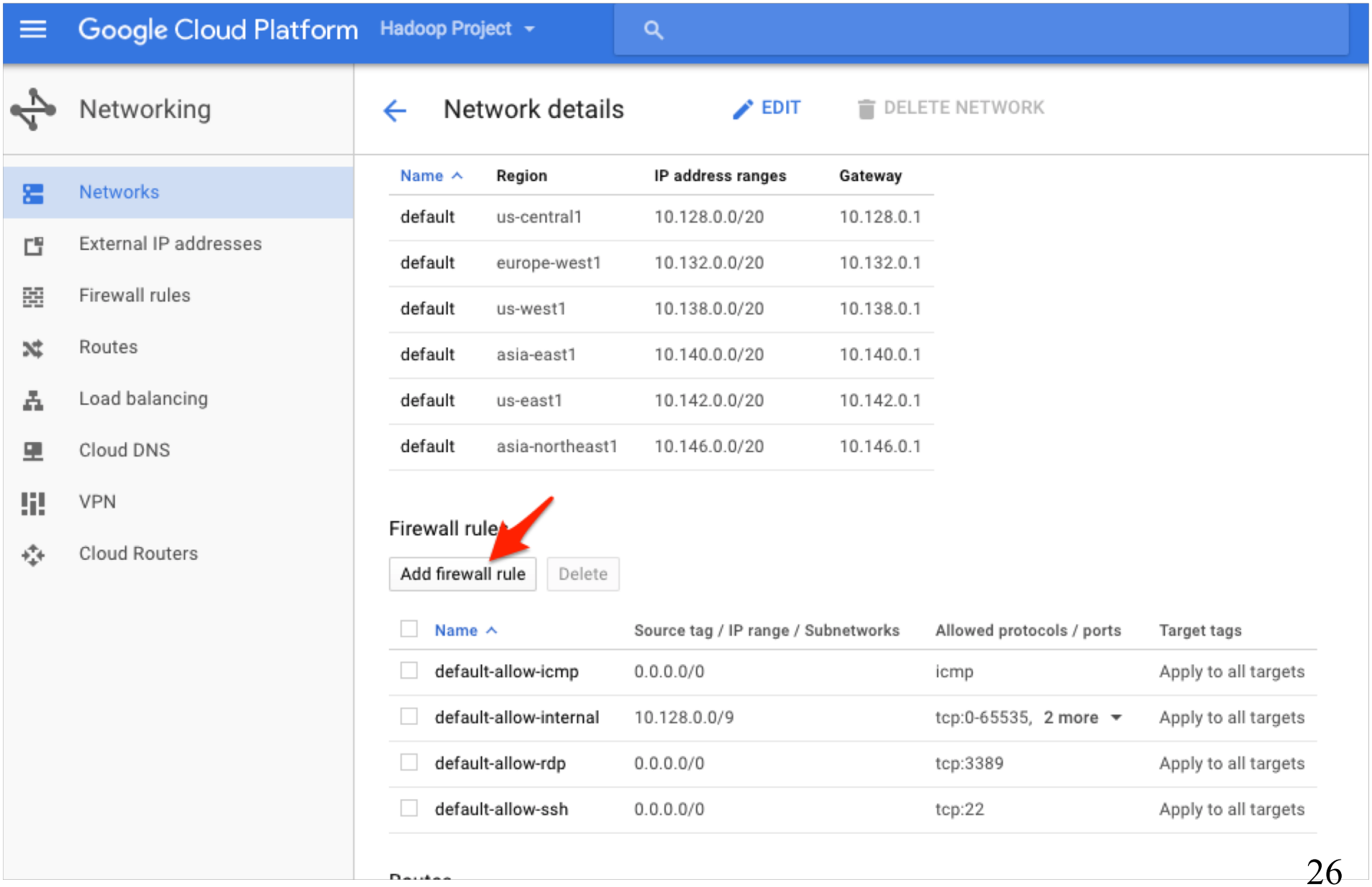

Google Cloud Platform
clouderacluster
Cloud Dataproc
Clusters
CREATE CLUSTER
REFRESH
DELETE
Clusters
Jobs
Search clusters, press
Name
Zone
Total worker nodes
Cloud Storage staging bucket
Created
Status
datascience
us-central1-a
2
dataproc-36c094d4-2796-4b03-85c9-7e9a90a99654-us
6 Jun 2017, 15:57:07
Running

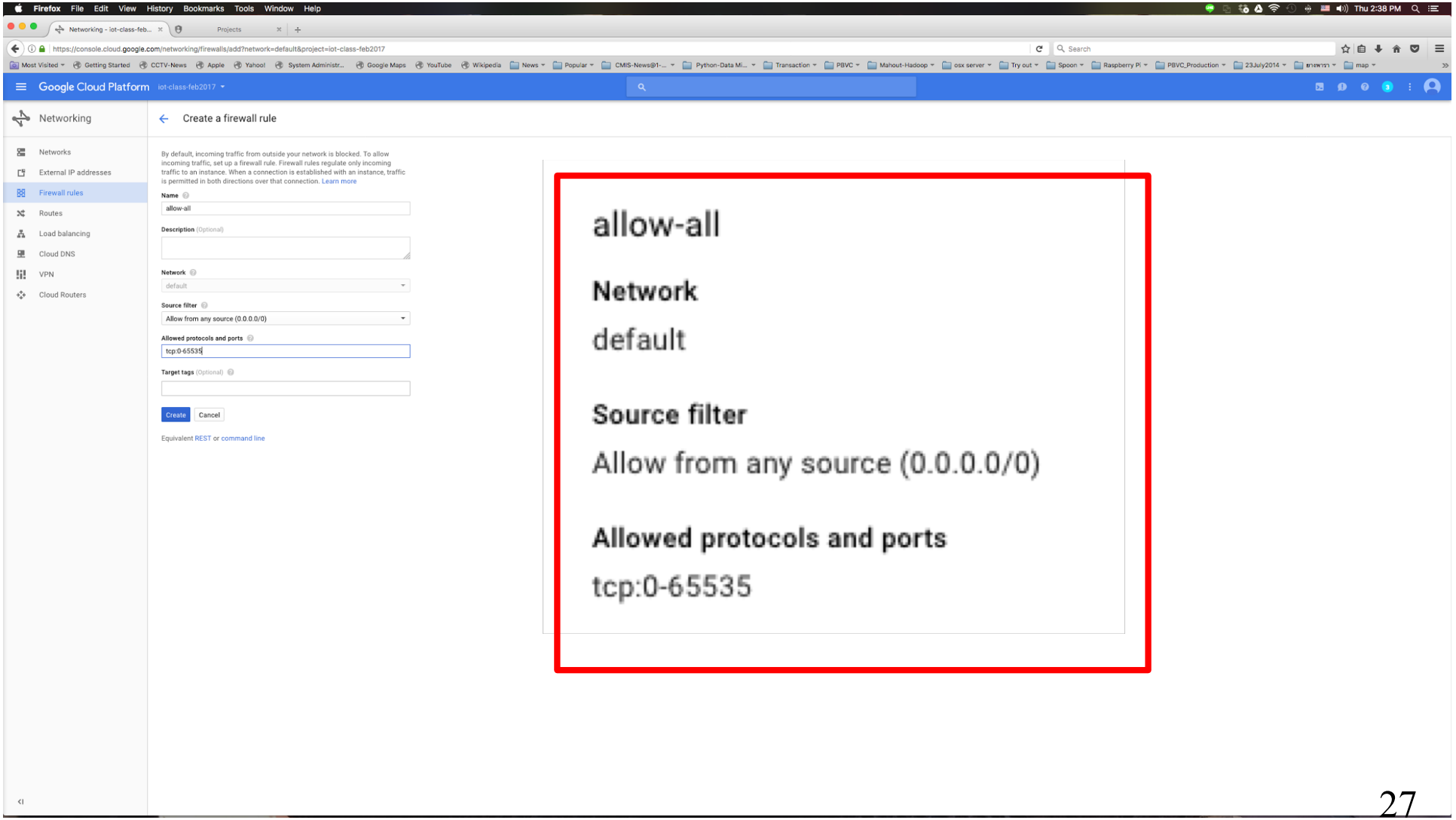

Google Cloud Platform
clouderacluster
Compute Engine
VM instances
SHOW INFO PANEL
VM instances
Instance groups
Instance templates
Disks
Snapshots
Images
Filter by label or name
Columns
Name
Zone
Recommendation
Internal IP
External IP
Connect
datascience-m
us-central1-a
10.128.0.2
104.198.236.81
SSH
datascience-w-0
us-central1-a
10.128.0.3
130.211.171.250
SSH
datascience-w-1
us-central1-a
10.128.0.4
104.155.180.252
SSH

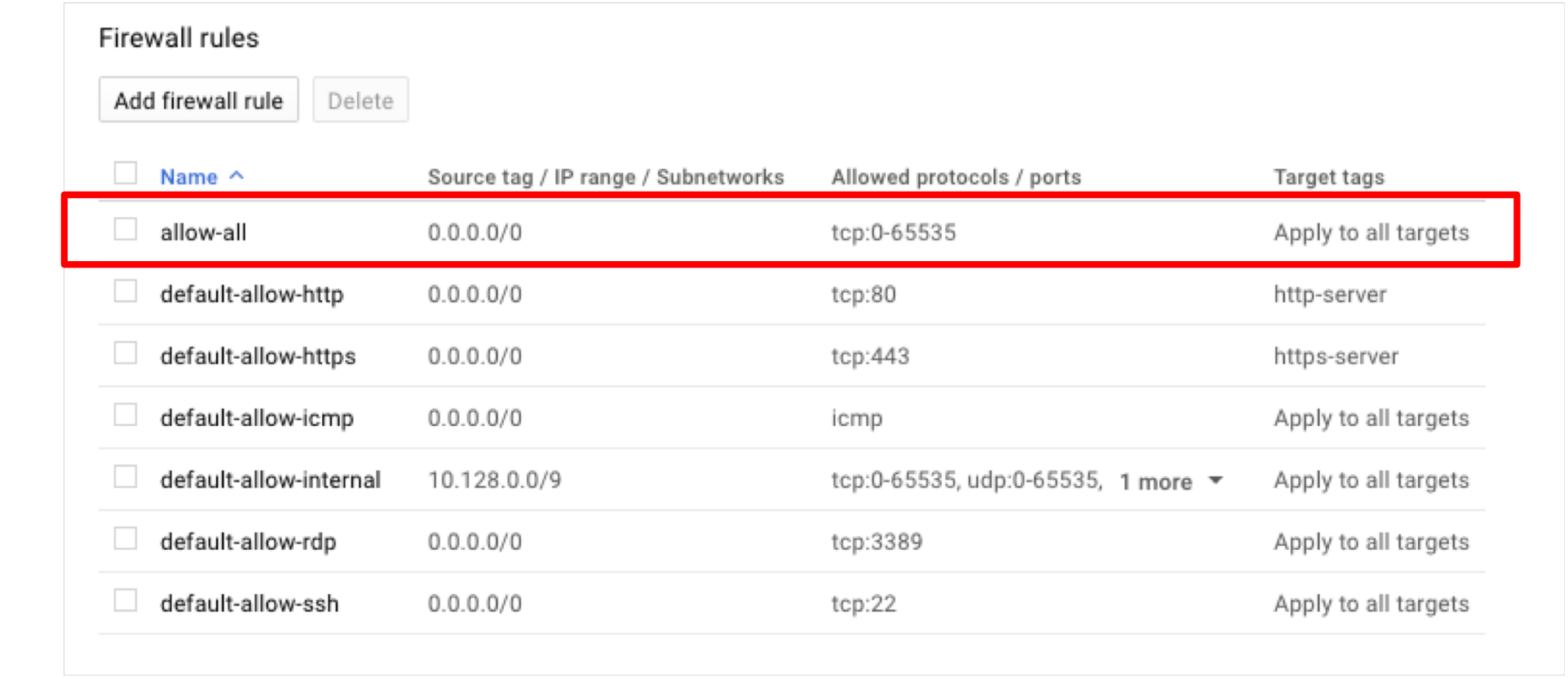
Config Firewall: Select the Network



Config Firewall:

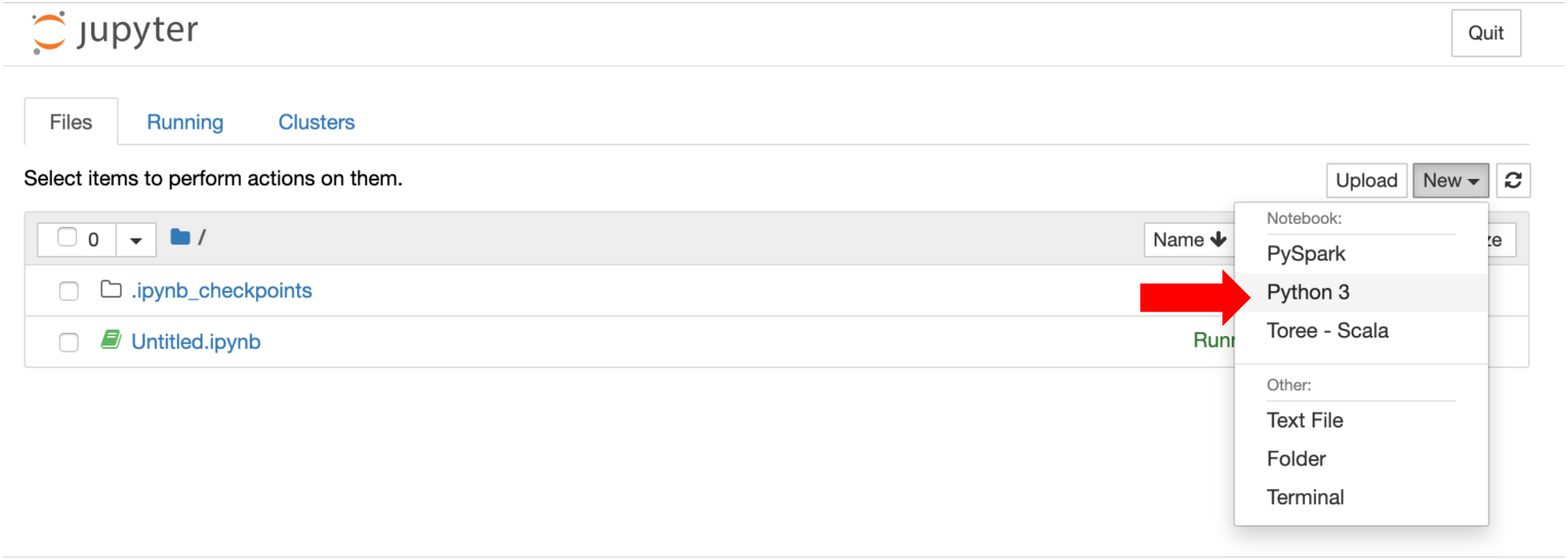



Google Cloud Platform
Networking
Create a firewall rule
Networks
External IP addresses
Firewall rules
Routes
Load balancing
Cloud DNS
VPN
Cloud Routers
Name
allow-all
Description (Optional)
Network
default
Source filter
Allow from any source (0.0.0.0/0)
Allowed protocols and ports
tcp:0-65535
Target tags (Optional)
Create
Cancel
Equivalent REST or command line
allow-all
Network
default
Source filter
Allow from any source (0.0.0.0/0)
Allowed protocols and ports
tcp:0-65535


Firewall rules
Add firewall rule
Delete
Name
Source tag / IP range / Subnetworks
Allowed protocols / ports
Target tags
allow-all
0.0.0.0/0
tcp:0-65535
Apply to all targets
default-allow-http
0.0.0.0/0
tcp:80
http-server
default-allow-https
0.0.0.0/0
tcp:443
https-server
default-allow-icmp
0.0.0.0/0
icmp
Apply to all targets
default-allow-internal
10.128.0.0/9
tcp:0-65535, udp:0-65535, 1 more
Apply to all targets
default-allow-rdp
0.0.0.0/0
tcp:3389
Apply to all targets
default-allow-ssh
0.0.0.0/0
tcp:22
Apply to all targets

Lunch the Jupyter notebook

<<public ip>> :8123



Functional tools in Python

- map

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

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

```
map(add1, a)
```

```
map(lambda x: x+1,a)
```

- filter

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

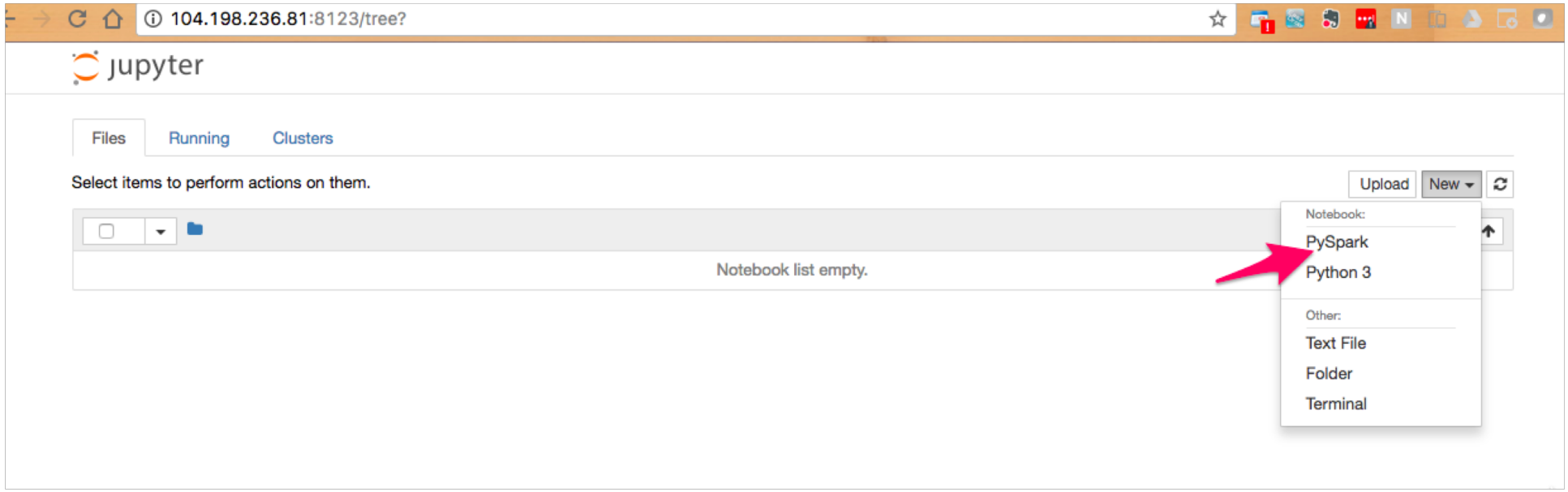
```
def isOdd(x) : return x%2==1
```

```
filter(isOdd,a)
```

```
filter(lambda x: x%2==1, a)
```

Lunch the Jupyter notebook

<<public ip>> :8123



Spark:Transformation

transformation	description
map(*func*) ใส่เท่าไร ออกเท่านั้น =	return a new distributed dataset formed by passing each element of the source through a function *func*
filter(*func*) น้อยหรือเท่าเดิม <=	return a new dataset formed by selecting those elements of the source on which *func* returns true
flatMap(*func*) แตกตัว >=	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:Actions

action	description
`collect()` แสดงหมด	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)` ระบุได้ว่าจะเอากี่ตัว	return an array with the first *n* elements of the dataset – currently not executed in parallel, instead the driver program computes all the elements

map & filter & collect

```
sc
```

SparkContext

Spark UI

Version
`v2.3.2`

Master
`yarn`

AppName
`pyspark-shell`

```
data = [1,2,3,4]
```

```
data1_rdd = sc.parallelize(data)
```

```
data2_rdd = data1_rdd.map(lambda x: x+1)
```

```
data2_rdd.collect()
```

```
[2, 3, 4, 5]
```

```
data3_rdd = data2_rdd.filter(lambda x: x%2==1)
```

```
data3_rdd.collect()
```

```
[3, 5]
```

flatMap & count & first & take

```
word = ["This","is","an","example"]
```

```
word1_rdd = sc.parallelize(word)
```

```
word2_rdd = word1_rdd.flatMap(lambda x: x.split(","))
```

```
word2_rdd.count()
```

```
4
```

```
word2_rdd.first()
```

```
'This'
```

```
word2_rdd.take(2)
```

```
['This', 'is']
```

Exercises

filter females to analyze female buying patterns

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

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

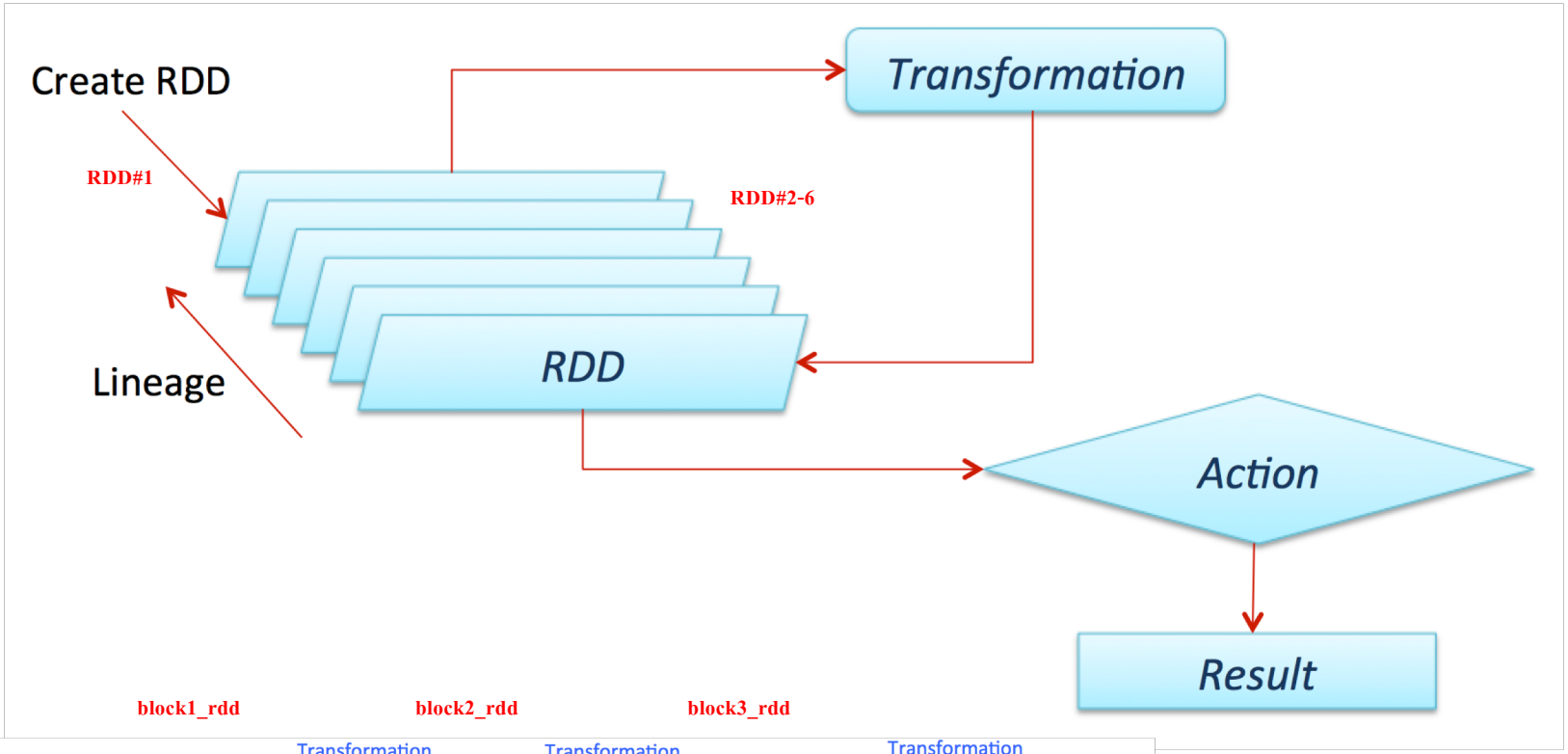
Spark:Transformation

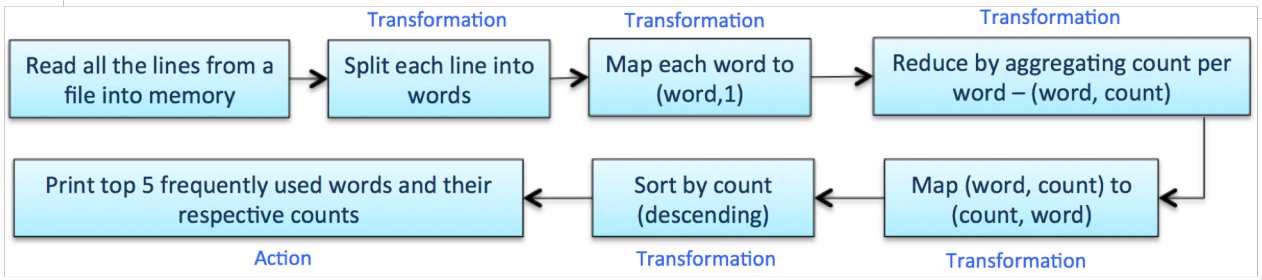
transformation	description
groupByKey([*numTasks*])	when called on a dataset of (K, V) pairs, returns a dataset of (K, Seq[V]) pairs
reduceByKey(*func*, [*numTasks*])	when called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given 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

Hands-On: Word Count (LAB I)

Focus: Spark's Transformation & Action

Workflow of Word Count





Platform: Cloudera/Dataproc

Tools: Jupyter

```
! wget https://storage.googleapis.com/aekanun/Basic/file-lines.txt
```

```
! ls -l file*
```

```
-rw-r--r-- 1 root root 1038 Jun 27  2017 file-lines.txt
```

```
! hdfs dfs -put file-lines.txt /
```

```
! hdfs dfs -ls /
```

```
Found 4 items
-rw-r--r--   2 root   hadoop       1038 2019-04-25 08:07 /file-lines.txt
drwx------   - mapred hadoop          0 2019-04-25 03:46 /hadoop
drwxrwxrwt   - hdfs   hadoop          0 2019-04-25 03:46 /tmp
drwxrwxrwt   - hdfs   hadoop          0 2019-04-25 06:02 /user
```


```
# Read all the lines from a file into the memory
block1_rdd = sc.textFile('/file-lines.txt').cache()
```

```
# Split each line into words
block2_rdd = block1_rdd.flatMap(lambda x: x.split(' '))
```

```
block2_rdd.collect()
```

```
['Apache',
 'Hadoop',
 'is',
 'an',
 'open-source',
 'software',
 'framework',
 'written',
 'in',
 'Java',
```

```
# Map each word to (word,1)
block3_rdd = block2_rdd.map(lambda x: (x,1))
```

```
# Reduce by aggregating count per word - (word,count)
block4_rdd = block3_rdd.reduceByKey(add)
```

```
# Map (word, count) to (count, word)
block5_rdd = block4_rdd.map(lambda x:(x[1],x[0]))
```

```
# Sort by count (descending)
block6_rdd = block5_rdd.sortByKey(ascending=False)
```

```
# Print top 5 frequently used words and their respective counts
block6_rdd.take(5)
```

```
[(6, 'Hadoop'), (6, 'and'), (6, 'a'), (5, 'in'), (5, 'data')]
```

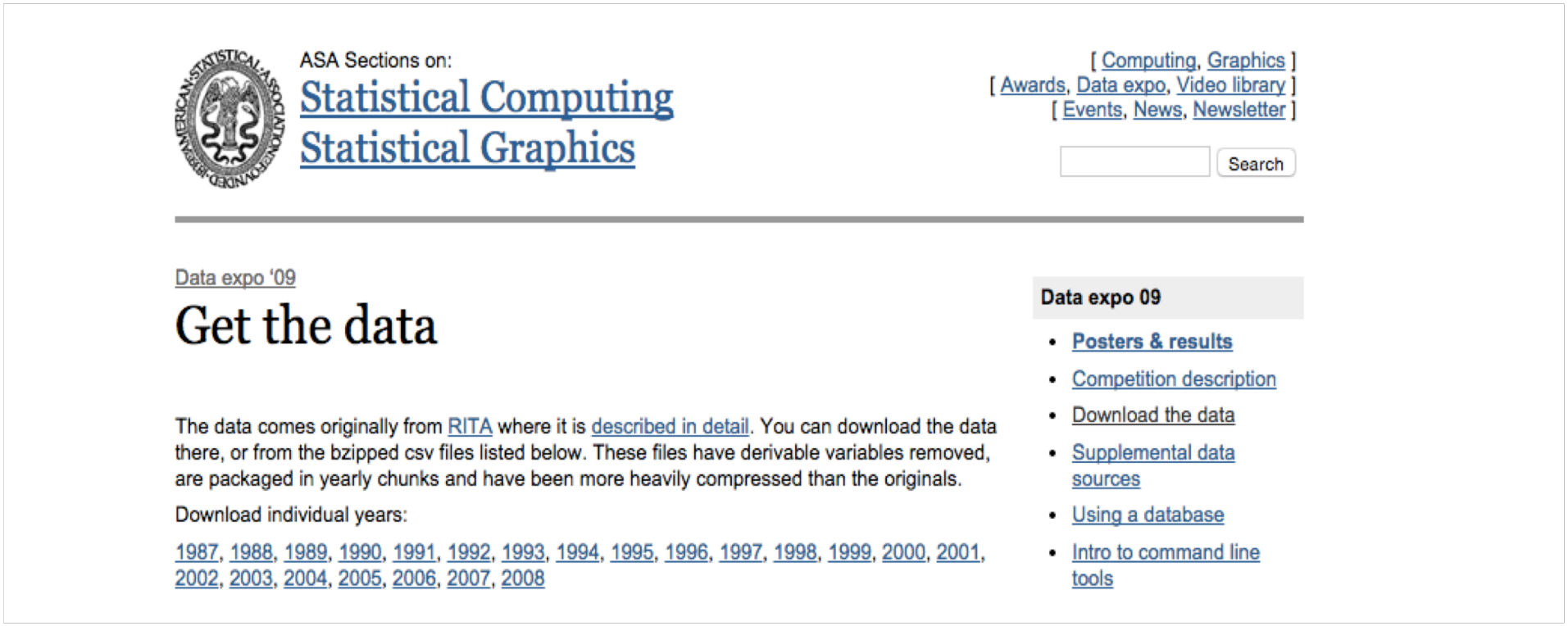
Hands-On: Make a format of “key,value”

(LAB II)

Focus: Data Preparation

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

Platform: Cloudera/Dataproc

Tools: Jupyter

```
! hdfs dfs -mkdir -p /user/cloudera/input/
```

```
! wget https://s3.amazonaws.com/imcbucket/data/flights/2008.csv
```

```
! hdfs dfs -rm -f /user/cloudera/input/2008.csv
```

```
! hdfs dfs -put 2008.csv /user/cloudera/input/
```

cloudera®

Platform: Cloudera/Dataproc

Tools: Jupyter

Header: Attributes

```
! head -3 ./2008.csv
```

```
Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarrier,FlightNum,TailNum,ActualElapsedTi
me,CRSElapsedTime,AirTime,ArrDelay,DepDelay,Origin,Dest,Distance,TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,C
arrierDelay,WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay
2008,1,3,4,2003,1955,2211,2225,WN,335,N712SW,128,150,116,-14,8,IAD,TPA,810,4,8,0,,0,NA,NA,NA,NA,NA
2008,1,3,4,754,735,1002,1000,WN,3231,N772SW,128,145,113,2,19,IAD,TPA,810,5,10,0,,0,NA,NA,NA,NA,NA
```

Data/Fact

cloudera®

Platform: Cloudera/Dataproc

Tools: Jupyter

```
# Create RDD with referencing a file.
airline_rdd =  sc.textFile("hdfs:///user/cloudera/input/2008.csv")
```

```
airline_rdd.take(2)
```

```
[u'Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarrier,FlightNum,TailNum,ActualElapse
dTime,CRSElapsedTime,AirTime,ArrDelay,DepDelay,Origin,Dest,Distance,TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverte
d,CarrierDelay,WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay',
 u'2008,1,3,4,2003,1955,2211,2225,WN,335,N712SW,128,150,116,-14,8,IAD,TPA,810,4,8,0,,0,NA,NA,NA,NA,NA']
```

Each element of RDD is one line of the file

Result from "take,collect" operation is data structure of "List"

Platform: Cloudera/Dataproc

Tools: Jupyter

text ***

```
#remove the header
header_text = airline_rdd.first()
airline_noheader_rdd = airline_rdd.filter(lambda row: row != header_text )
```

text ***

cloudera®

Platform: Cloudera/Dataproc

Tools: Jupyter

```
airline_noheader_rdd.take(3)
```

```
[u'2008,1,3,4,2003,1955,2211,2225,WN,335,N712SW,128,150,116,-14,8,IAD,TPA,810,4,8,0,,0,NA,NA,NA,NA,NA',
 u'2008,1,3,4,754,735,1002,1000,WN,3231,N772SW,128,145,113,2,19,IAD,TPA,810,5,10,0,,0,NA,NA,NA,NA,NA',
 u'2008,1,3,4,628,620,804,750,WN,448,N428WN,96,90,76,14,8,IND,BWI,515,3,17,0,,0,NA,NA,NA,NA,NA']
```

Platform: Cloudera/Dataproc

Tools: Jupyter

text

Data Structure: List

```
#Create Key,Value in Dictionary format.
fieldname_list = header_text.split(',')
def make_row(row):
    fieldvalues_list = row.split(',')
    # zip - convert key,value to tuple
    # dict - convert tuple to dictionary
    d = dict(zip(fieldname_list,fieldvalues_list))
    return d
```

```
airline_rows_rdd = airline_noheader_rdd.map(make_row)
```

Data Structure: List

Platform: Cloudera/Dataproc

Tools: Jupyter

```
airline_rows_rdd.take(3)
```


```
[{u'ActualElapsedTime': u'128',
  u'AirTime': u'116',
  u'ArrDelay': u'-14',
  u'ArrTime': u'2211',
  u'CRSArrTime': u'2225',
  u'CRSDepTime': u'1955',
  u'CRSElapsedTime': u'150',
  u'CancellationCode': u'',
  u'Cancelled': u'0',
  u'CarrierDelay': u'NA',
  u'DayOfWeek': u'4',
  u'DayofMonth': u'3',
  u'DepDelay': u'8',
  u'DepTime': u'2003',
  u'Dest': u'TPA',
  u'Distance': u'810',
  u'Diverted': u'0',
  u'FlightNum': u'335',
  u'LateAircraftDelay': u'NA',
  u'Month': u'1',
  u'NASDelay': u'NA',
  u'Origin': u'IAD',
  u'SecurityDelay': u'NA',
  u'TailNum': u'N712SW',
  u'TaxiIn': u'4',
  u'TaxiOut': u'8',
  u'UniqueCarrier': u'WN',
  u'WeatherDelay': u'NA',
  u'Year': u'2008'},
```