MSAN 694 : Distributed Computing

Diane Woodbridge, Ph.D.
MSAN, University of San Francisco

Reviews

Pair RDD Operations

Tuning Spark – Persist in Memory/Disk

Spark Interview Questions

~~What is Apache Spark?~~

~~Explain the key features of Spark.~~

~~What is RDD?~~

~~How to create RDD.~~

~~What is "partitions"?~~

~~Types or RDD operations?~~

~~What is "transformation"?~~

~~What is "action"?~~

~~Functions of "spark core"?~~

~~What is "spark context"?~~

~~What is an "RDD lineage"?~~

Which file systems does Spark support?

~~List the various types of "Cluster Managers" in Spark.~~

What is "YARN"?

What is "Mesos"?

What is a "worker node"?

What is an "accumulator"?

~~What is "Spark SQL" (Shark)?~~

~~What is "SparkStreaming"?~~

~~What is "GraphX"?~~

~~What is "MLlib"?~~

https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Spark Interview Questions

~~What are the advantages of using Apache Spark over Hadoop MapReduce for big data processing?~~

~~What are the languages supported by Apache Spark for developing big data applications?~~

~~Can you use Spark to access and analyze data stored in Cassandra databases?~~

Is it possible to run Apache Spark on Apache Mesos?

~~How can you minimize data transfers when working with Spark?~~

Why is there a need for broadcast variables?

~~Name a few companies that use Apache Spark in production.~~

What are the various data sources available in SparkSQL?

What is the advantage of a Parquet file?

~~What do you understand by Pair RDD?~~

Is Apache Spark a good fit for Reinforcement learning?

https://www.dezyre.com/article/top-50-spark-interview-questions-and-answers-for-2016/208

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Final Exams

- December 11th, 10 am
  (101 Howard Room 154, 155 and 156)
- Scope : Week 1 – Week 7
- Grading
  - Multiple Choices - 15 pt
    (Canvas-based Closed-book)
  - Coding – 10 pt

Contents

Contents

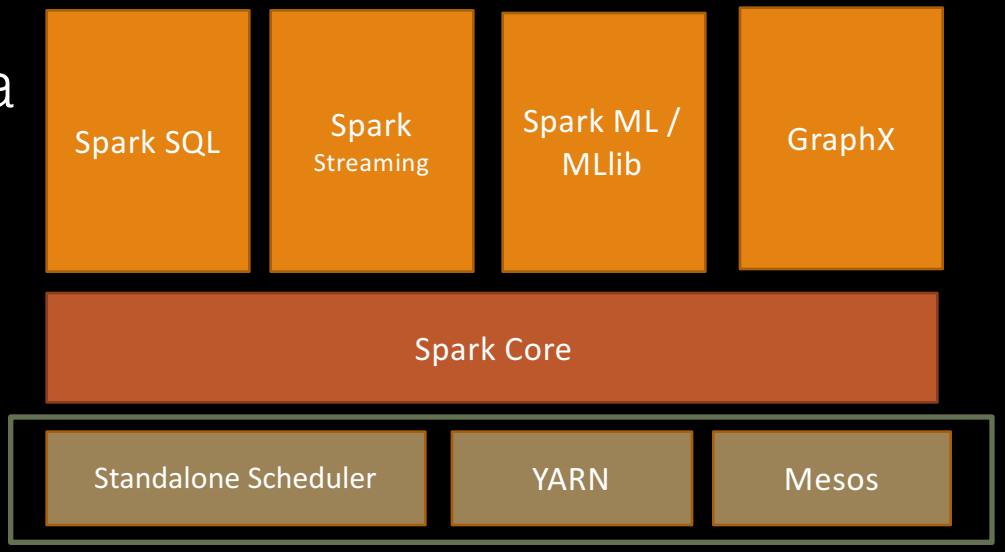
Running on a Spark Cluster

Running on a Spark Standalone Cluster

Running on a Spark Cluster

So far, we've programmed in a local mode. (not a cluster mode!)
Spark cluster

- A set of interconnected processes, running in a distributed manner on different machines.
- Types (3)
  - Spark standalone
  - Hadoop YARN
  - Mesos



*The local mode and local cluster mode, although the easiest and quickest methods of setting up Spark, are used mainly for **testing purposes.**

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Spark Architecture

Spark Runtime Components (3)

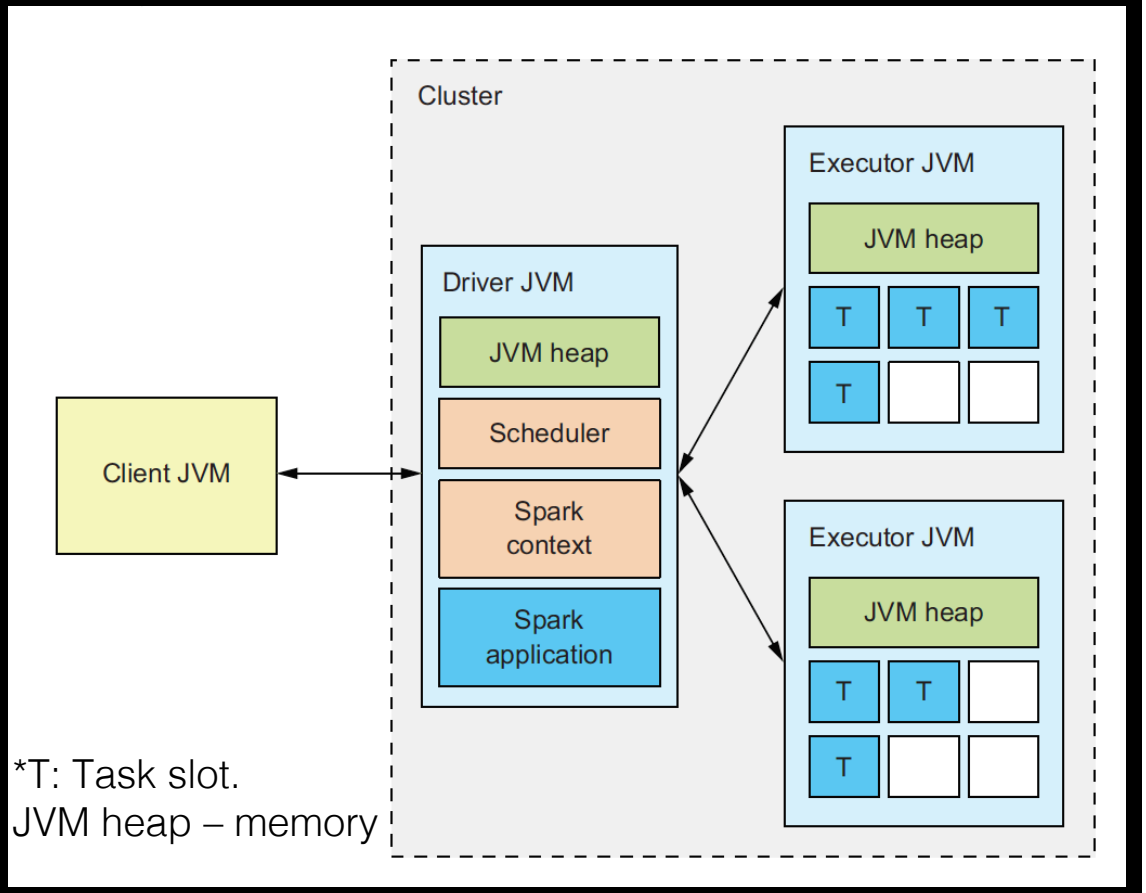
1. Client
   - Starts the driver program.
   - Prepares the classpath and all configuration options.
   - Client process : spark-submit, pyspark, spark-shell scripts
2. Driver
   - Orchestrates and monitors executor of an Spark application.
     - Once it gets information from the Spark master of all the workers in the cluster and where they are, the driver program distributes Spark tasks to each worker's executor.
   - The driver also receives computed results from each executor's tasks.
   - Always one driver per application.
3. Executor
   - Executes Spark tasks with a configurable number of cores.
   - Stores and caches all data partitions in its memory.

Spark Architecture

Deploy-mode (2)

- Physical placement of executor and driver processes depends on the cluster types and its configuration.

1. Cluster-deploy mode
2. Client-deploy mode



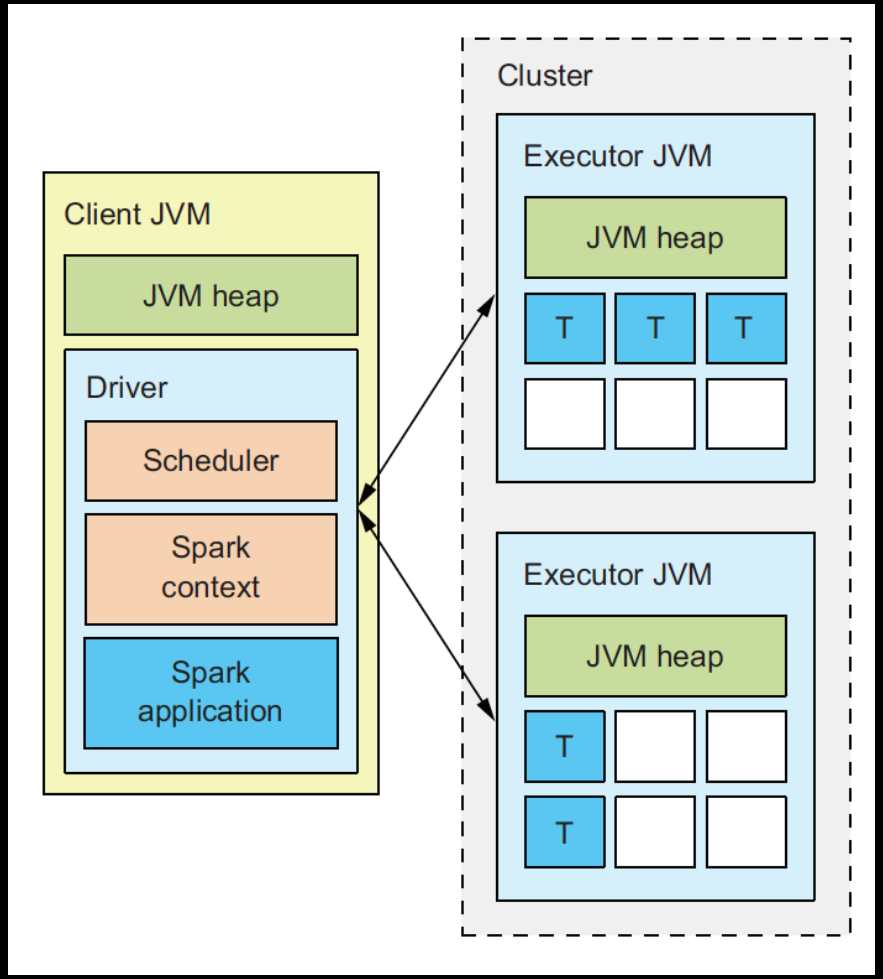
Cluster-deploy mode

Spark Architecture

Deploy-mode (2)

- Physical placement of executor and driver processes depends on the cluster types and its configuration.

1. Cluster-deploy mode
2. Client-deploy mode



Client-deploy mode

Spark Architecture

Spark Cluster Components

1. Cluster Manager
    - Monitors the worker nodes and reserves resources upon request by the master.
2. Master
    - Accepts applications to be run.
    - Requests resources in the cluster and makes them available to the driver.
    - Depending on the mode, it acts as a resource manager and decides where and how many executors to launch, and on what Spark workers in the cluster.
3. Worker
    - Upon receiving instructions from Spark Master, the Spark worker JVM launches executors on the worker on behalf of the Spark Driver.
    - Spark has to be installed on all nodes.

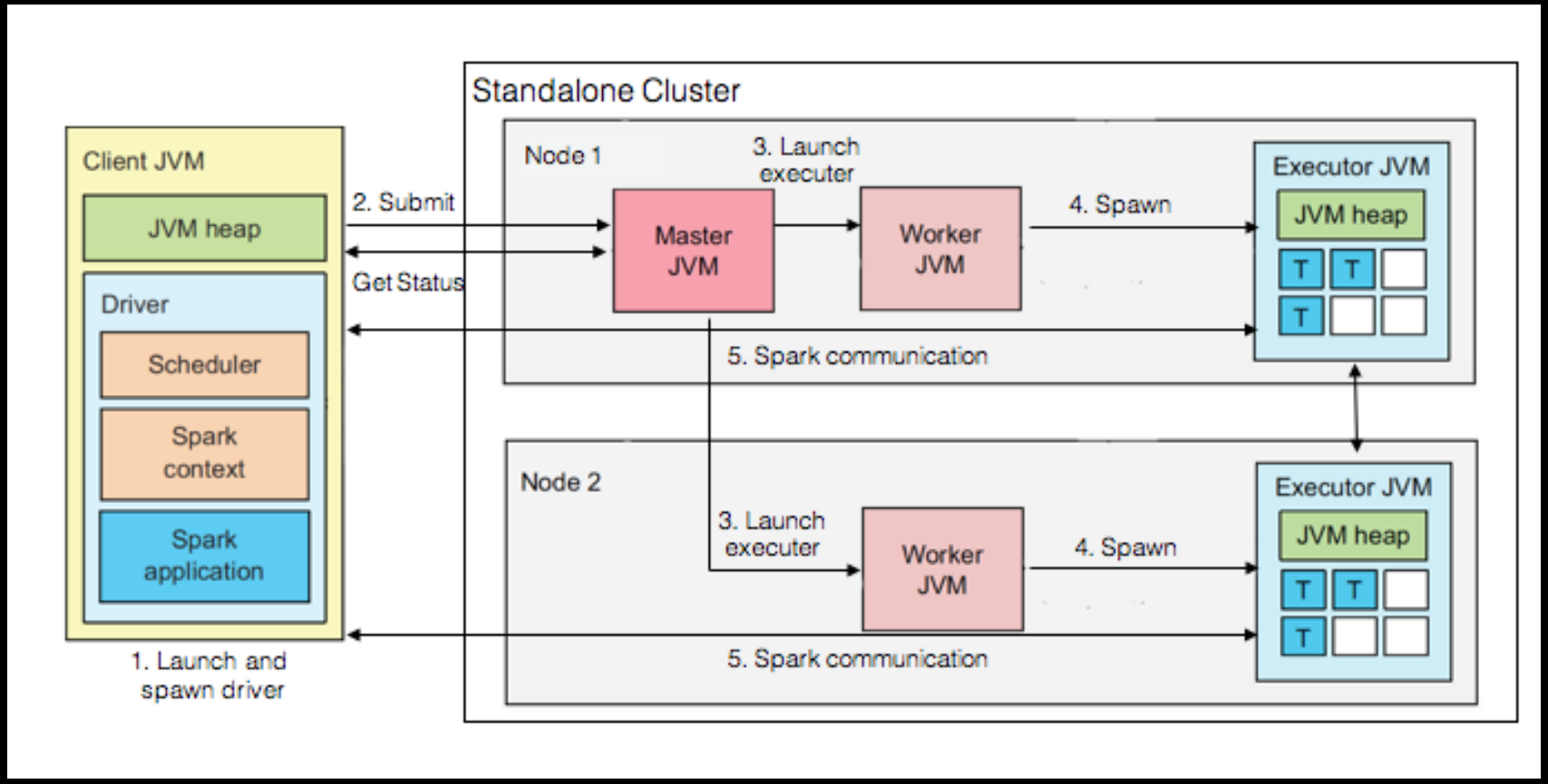
Contents

Running on a Spark Standalone Cluster

Spark Standalone Cluster

- Built and optimized specifically for Spark.
- A master process acts as the cluster manager.
- Doesn’t support communication with an HDFS secured with the Kerberos authentication protocol.
- Provide simple and faster job startup.

Running on a Spark Standalone Cluster



Client-deploy mode

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Spark Standalone Cluster

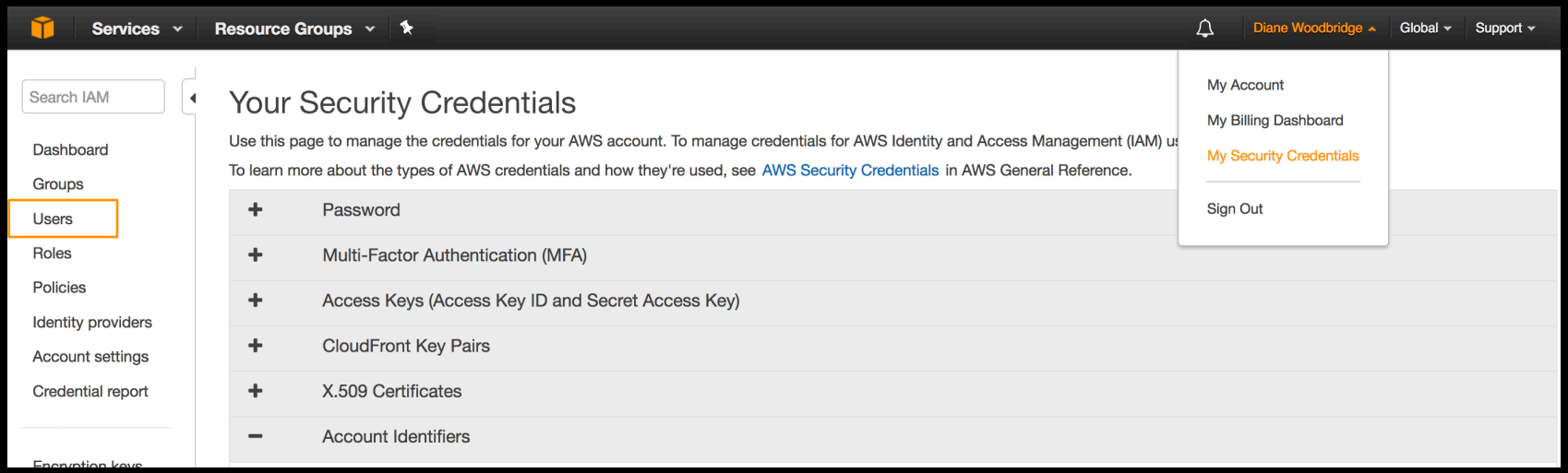
Running on Amazon EC2.

- AWS EC2 : Amazon's cloud service that lets you rent virtual servers to run your applications.
- spark-ec2 script
  - You can launch a new cluster (by telling the script its size and giving it a name), shutdown an existing cluster, or log into a cluster.
  - It automatically sets up Apache Spark and HDFS on the cluster for you.

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- On your AWS console, go to the user information ➔ My Security Credentials. ➔ Choose Users.

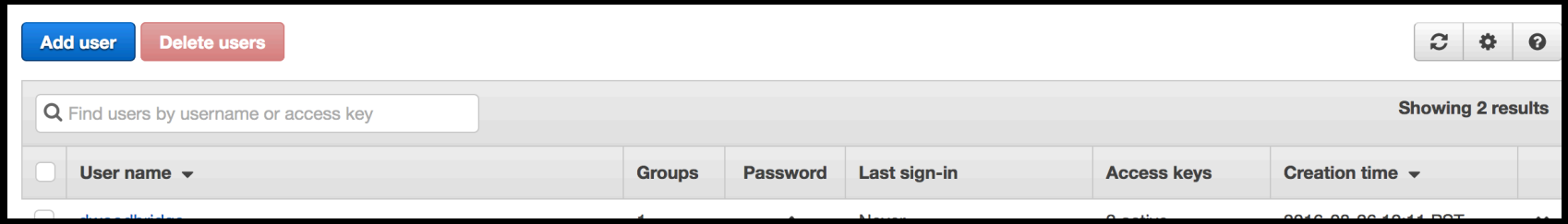


DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Choose Add User.

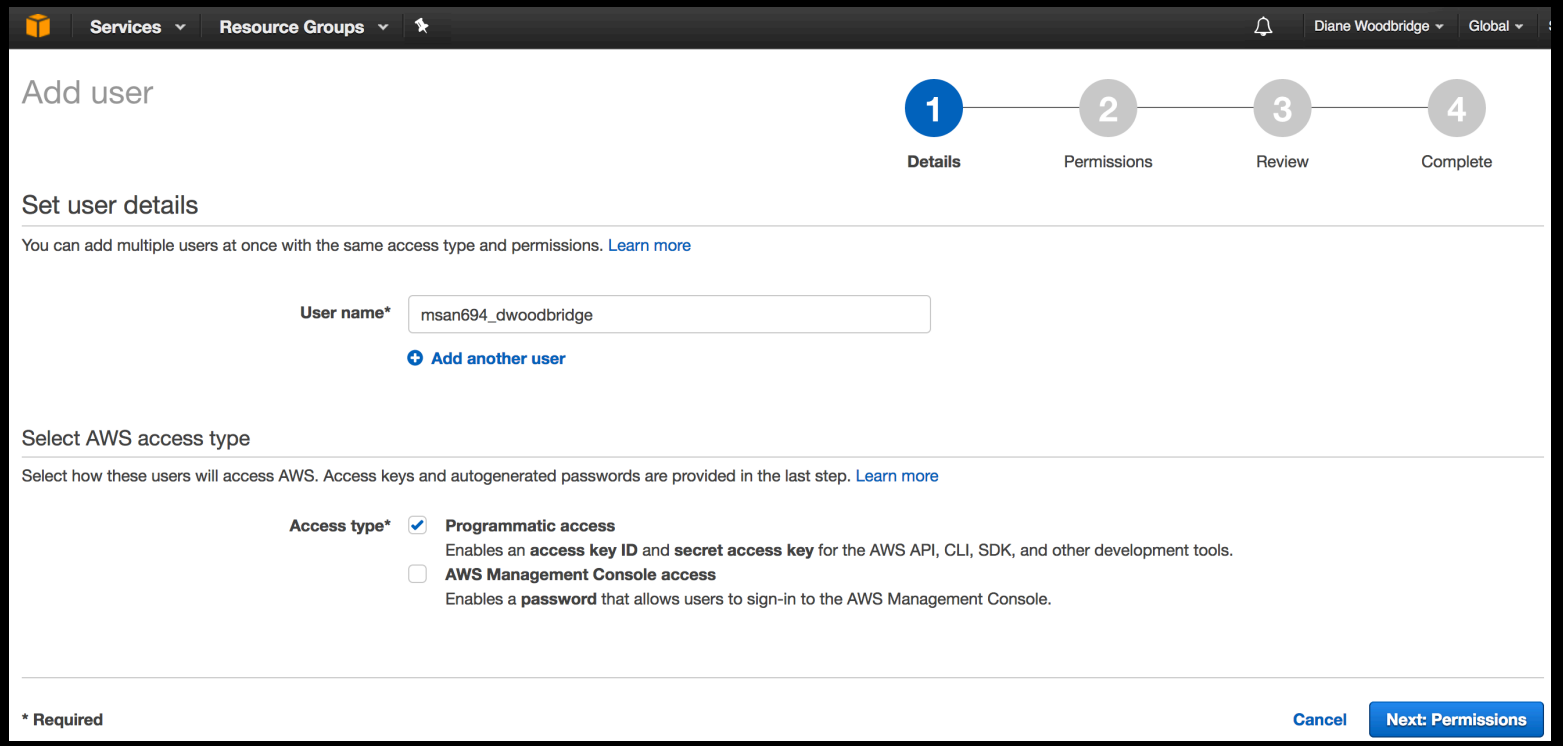


DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Spark Standalone Cluster

Running on Amazon EC2.

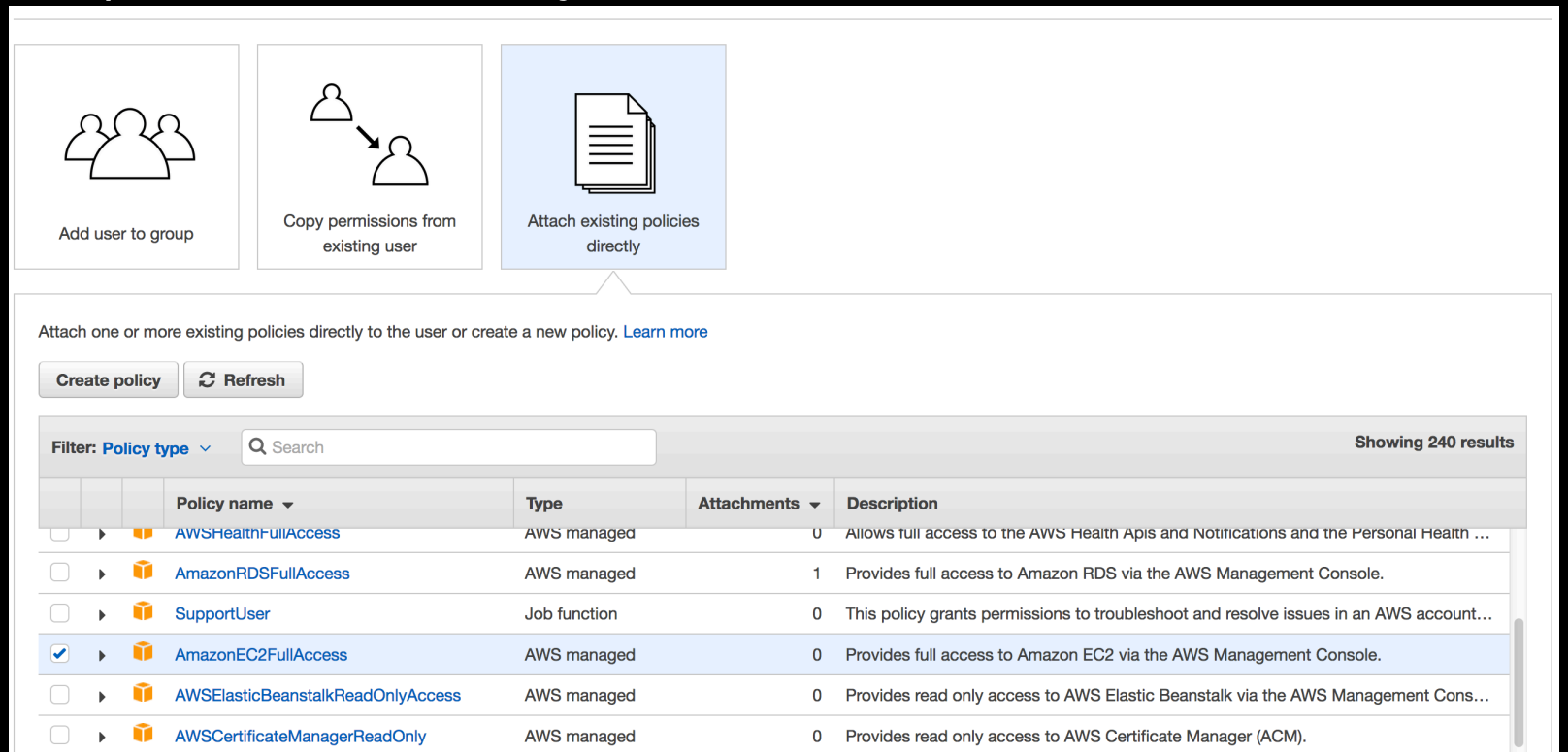
- Choose a user name and access type.



Running on a Spark Standalone Cluster

Running on Amazon EC2.

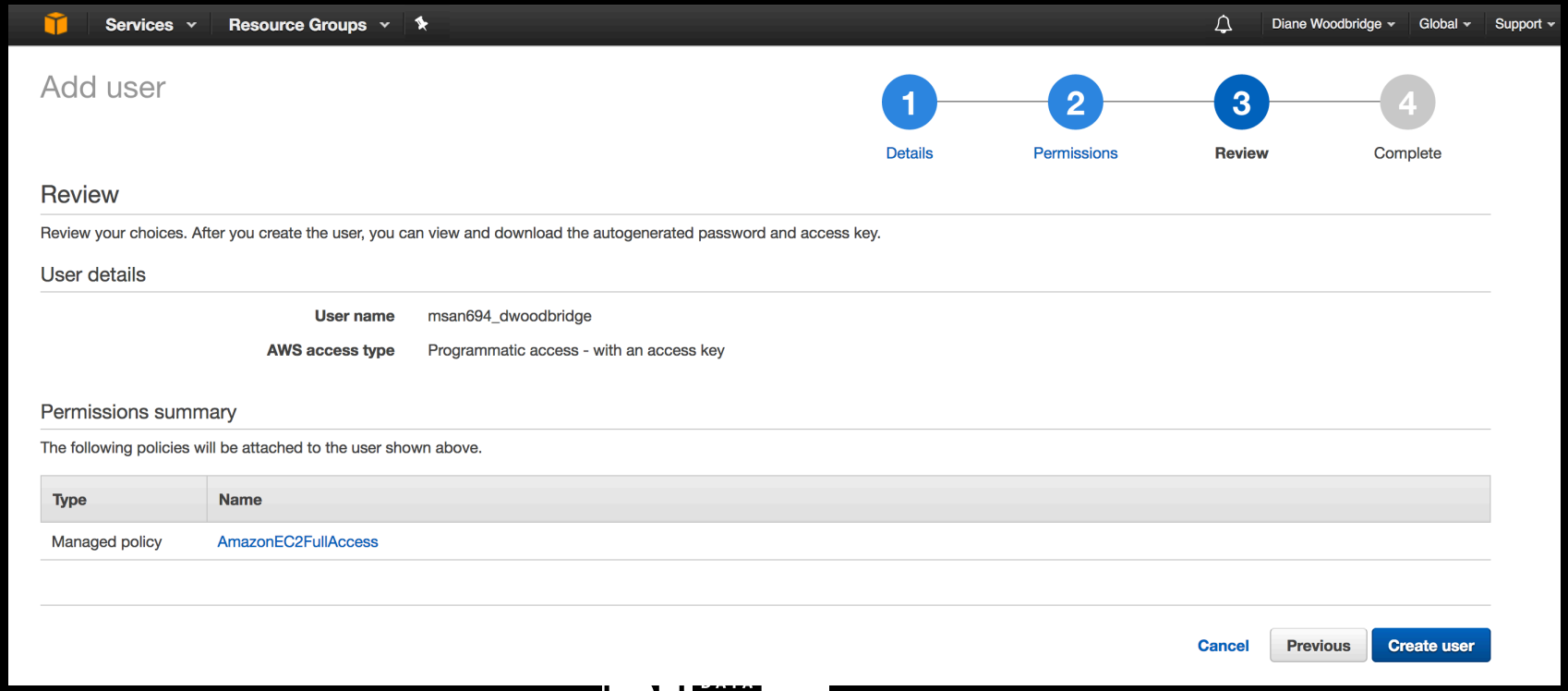
- Set permissions for the user. : Attach existing policies directly. ➔ Amazon EC2FullAccess.



Running on a Spark Standalone Cluster

Running on Amazon EC2.

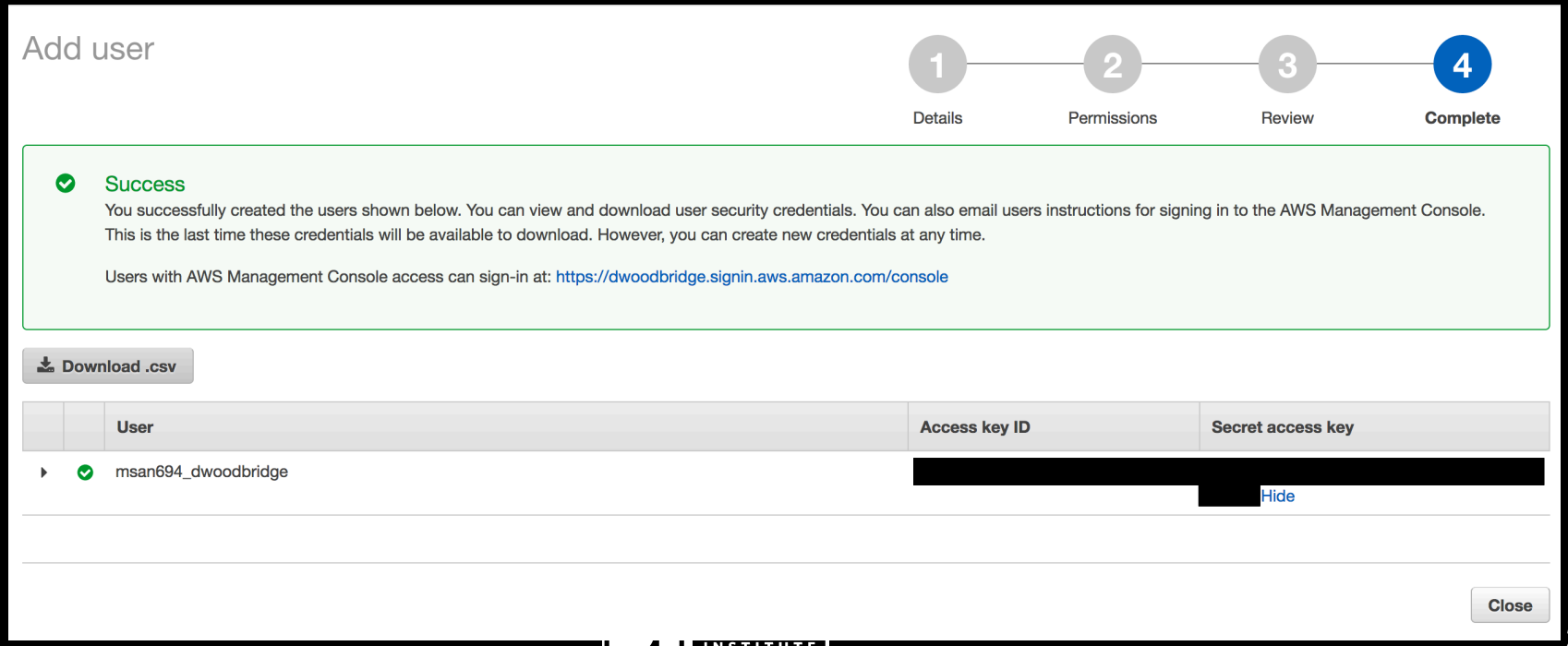
- Create User.



Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Copy the Access Key ID and Secret Access Key (Or download .csv).



Running on a Spark Standalone Cluster

Using Amazon EMR.

- On your .profile or .bash_profile store AWS_ACESS_KEY_IDA and AWS_SECRET_ACCESS_KEY.

```
export AWS_ACCESS_KEY_ID=AKIAI
export AWS_SECRET_ACCESS_KEY=b
```

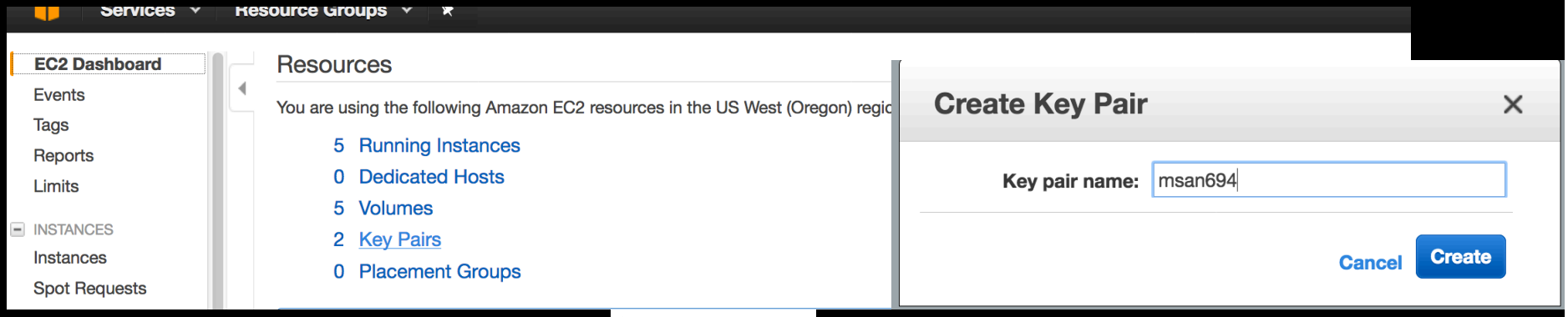
AWS **Access Key ID** and **Secret Access Key**.
This can be used to access and control basic AWS services through the API including EC2, S3, SimpleDB, CloudFront, SQS, EMR, RDS, etc.

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- On EC2 Console (https://console.aws.amazon.com/ec2/), Choose Key Pairs ➔ Create Key Pairs , download the .pem and store somewhere on your machine*.
- Make sure that you set the permissions for the private key file to 600 (i.e. only you can read and write it) so that ssh will work.
  - `chmod 600 filename`



Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Clone AMPLab's code
  https://github.com/amplab/spark-ec2.git

- Switch the branch to 2.0 (Default 1.6)
  - **`git checkout branch-2.0`**

```
ML-ITS-603436:spark-ec2 dwoodbridge$ git branch
  branch-1.6
* branch-2.0
```

- Go to spark-ec2 directory
  - **`./spark-ec2 --help`** for more info

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Available commands and arguments

Command	Description
launch	Launches EC2 instances, installs the required software packages, and starts the master and slaves.
login	Logs in to the instance running the Spark master.
stop	Stops all the cluster instances.
start	Starts all the cluster instances and reconfigures the cluster.
get-master	Return the address of the instance where the master is running
destroy	An unrecoverable action that terminate EC2 instances and destroys the cluster.

Running on a Spark Standalone Cluster

Running on Amazon EC2.

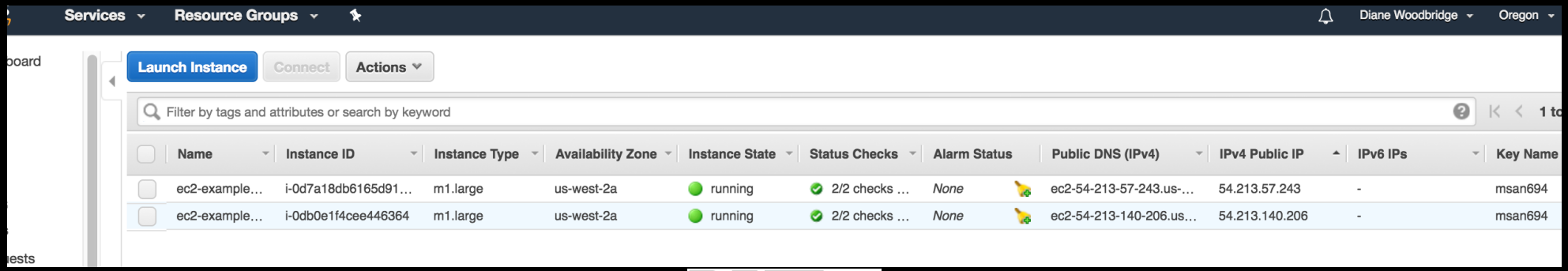
- Available commands and arguments
  - --key-pair (-k) : The name of your EC2 key pair.
  - --identity-file (-i) : the private key file for your key pair.
  - --region (-r) : specifies an EC2 region in which to launch instances.
  - --zone (-z) : used to specify an EC2 availability zone to launch instances in.
  - --slave (-s) : the number of slaves.

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Launch : **`./spark-ec2 -k msan694 -i msan694.pem -s 1 -r us-west-2 -z us-west-2a launch ec2-example`**

```
ML-ITS-603436:spark-ec2 dwoodbridge$ ./spark-ec2 -k msan694 -i msan694.pem -s 1 -r us-west-2 -z us-west-2a launch ec2-example
Setting up security groups...
Searching for existing cluster ec2-example in region us-west-2...
Spark AMI: ami-9a6e0daa
Launching instances...
Launched 1 slave in us-west-2a, regid = r-04b0ea2f297c3a270
Launched master in us-west-2a, regid = r-05256fa60b23ed6d6
Waiting for AWS to propagate instance metadata...
Applying tags to master nodes
Applying tags to slave nodes
Waiting for cluster to enter 'ssh-ready' state...
```



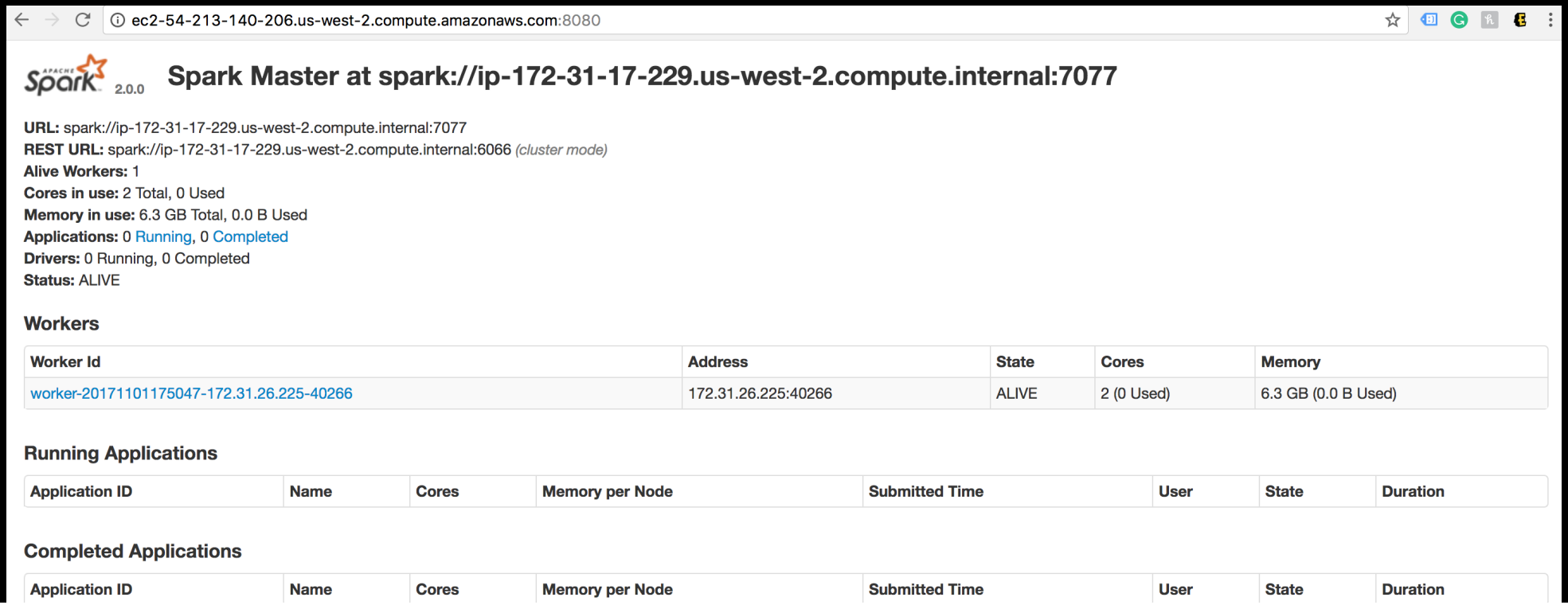
DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Once launched, web UI should be available.
- Master

```
Spark standalone cluster started at http://ec2-54-213-140-206.us-west-2.compute.amazonaws.com:8080
Ganglia started at http://ec2-54-213-140-206.us-west-2.compute.amazonaws.com:5080/ganglia
Done!
ML-ITS-603436:spark-ec2 dwoodbridge$
```



Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Workers

Spark 2.0.0

Spark Worker at 172.31.26.225:43485

ID: worker-20171101222446-172.31.26.225-43485
Master URL: spark://ip-172-31-17-229.us-west-2.compute.internal:7077
Cores: 2 (0 Used)
Memory: 6.3 GB (0.0 B Used)

Back to Master

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Login to the cluster

`./spark-ec2 -k msan694 -i msan694.pem -r us-west-2 -z us-west-2a login ec2-example`

```
ML-ITS-603436:spark-ec2 dwoodbridge$ ./spark-ec2 -k msan694 -i msan694.pem -r us-west-2 login ec2-example
Searching for existing cluster ec2-example in region us-west-2...
Found 1 master, 3 slaves.
Logging into master ec2-54-202-247-225.us-west-2.compute.amazonaws.com...
Warning: Permanently added 'ec2-54-202-247-225.us-west-2.compute.amazonaws.com,54.202.247.225' (ECDSA) to the list of known hosts.
Last login: Tue Oct 24 07:19:27 2017 from 73.231.58.161

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2013.03-release-notes/
Amazon Linux version 2017.09 is available.
root@ip-172-31-37-124 ~]$
```

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Running pyspark

~/spark/bin/pyspark

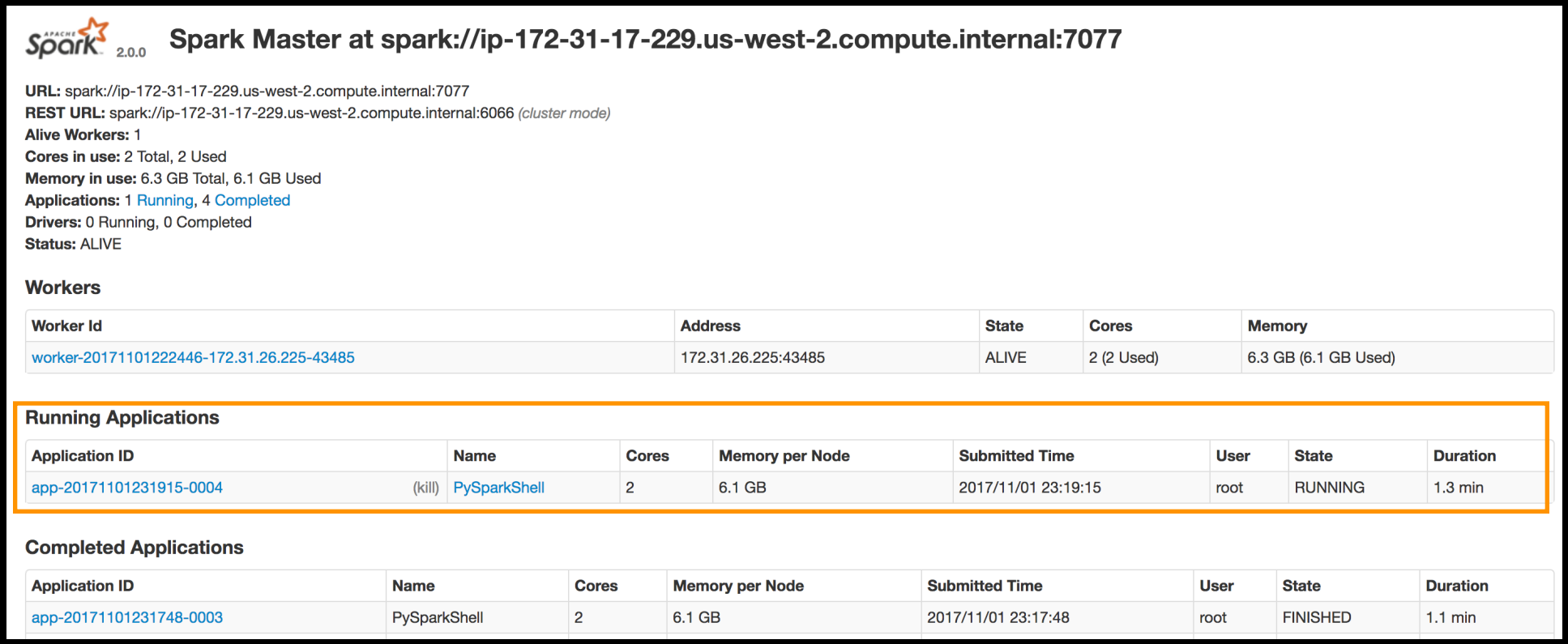
```
[root@ip-172-31-17-229 bin]$ ~/spark/bin/pyspark
Python 2.7.12 (default, Sep  1 2016, 22:14:00)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/11/01 23:19:14 WARN NativeCodeLoader: Unable to load native-hadoop library for your p
ble
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Python version 2.7.12 (default, Sep  1 2016 22:14:00)
SparkSession available as 'spark'.
>>> sc
<pyspark.context.SparkContext object at 0x7fbfb1e66ed0>
>>>
```

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Running pyspark



Spark 2.0.0 **Spark Master at spark://ip-172-31-17-229.us-west-2.compute.internal:7077**

URL: spark://ip-172-31-17-229.us-west-2.compute.internal:7077
REST URL: spark://ip-172-31-17-229.us-west-2.compute.internal:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 2 Total, 2 Used
Memory in use: 6.3 GB Total, 6.1 GB Used
Applications: 1 Running, 4 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20171101222446-172.31.26.225-43485	172.31.26.225:43485	ALIVE	2 (2 Used)	6.3 GB (6.1 GB Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20171101231915-0004 (kill)	PySparkShell	2	6.1 GB	2017/11/01 23:19:15	root	RUNNING	1.3 min

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20171101231748-0003	PySparkShell	2	6.1 GB	2017/11/01 23:17:48	root	FINISHED	1.1 min

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Running spark-submit
  - If your running in a cluster, you need to copy files across all the nodes.

```
Traceback (most recent call last):
  File "/root/example.py", line 19, in <module>
    print sc.textFile("file:///root/input.txt").count()
  File "/root/spark/python/lib/pyspark.zip/pyspark/rdd.py", line 1008, in count
  File "/root/spark/python/lib/pyspark.zip/pyspark/rdd.py", line 999, in sum
  File "/root/spark/python/lib/pyspark.zip/pyspark/rdd.py", line 873, in fold
  File "/root/spark/python/lib/pyspark.zip/pyspark/rdd.py", line 776, in collect
  File "/root/spark/python/lib/py4j-0.10.1-src.zip/py4j/java_gateway.py", line 933, in __call__
  File "/root/spark/python/lib/py4j-0.10.1-src.zip/py4j/protocol.py", line 312, in get_return_value
py4j.protocol.Py4JJavaError: An error occurred while calling z:org.apache.spark.api.python.PythonRDD.collectAndServe.
: org.apache.spark.SparkException: Job aborted due to stage failure: Task 1 in stage 1.0 failed 4 times, most recent failure: Los
t task 1.3 in stage 1.0 (TID 11, 172.31.26.225): java.io.FileNotFoundException: File file:/root/input.txt does not exist
  at org.apache.hadoop.fs.RawLocalFileSystem.deprecatedGetFileStatus(RawLocalFileSystem.java:511)
  at org.apache.hadoop.fs.RawLocalFileSystem.getFileLinkStatusInternal(RawLocalFileSystem.java:724)
  at org.apache.hadoop.fs.RawLocalFileSystem.getFileStatus(RawLocalFileSystem.java:501)
  at org.apache.hadoop.fs.FilterFileSystem.getFileStatus(FilterFileSystem.java:397)
  at org.apache.hadoop.fs.ChecksumFileSystem$ChecksumFSInputChecker.<init>(ChecksumFileSystem.java:137)
  at org.apache.hadoop.fs.ChecksumFileSystem.open(ChecksumFileSystem.java:339)
  at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:764)
  at org.apache.hadoop.mapred.LineRecordReader.<init>(LineRecordReader.java:108)
  at org.apache.hadoop.mapred.TextInputFormat.getRecordReader(TextInputFormat.java:67)
  at org.apache.spark.rdd.HadoopRDD$$anon$1.<init>(HadoopRDD.scala:246)
  at org.apache.spark.rdd.HadoopRDD.compute(HadoopRDD.scala:209)
  at org.apache.spark.rdd.HadoopRDD.compute(HadoopRDD.scala:102)
  at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:319)
  at org.apache.spark.rdd.RDD.iterator(RDD.scala:283)
  at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:38)
```

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Running spark-submit
  - Update your code
    - Master – master node address and its port (Given in the Spark Web UI)
    - If you are reading from a local file (This must be copied to other worker nodes), update the path.
      - file:///ABSOLUTE_PATH

spark-ec2 — ssh ‹ spark-ec2 -i msan694.pem -k msan694 -r us-west-2 login ec2-example — 160×33

```
from pyspark import SparkContext, SparkConf

conf = SparkConf().setMaster("spark://ip-172-31-17-229.us-west-2.compute.internal:7077").setAppName("Woodbridge_Diane")
sc = SparkContext(conf = conf)

print  sc.parallelize([1,2,3,4]).mean()

print sc.textFile("file:///root/example/input_2.txt").count()

sc.stop()
~
```

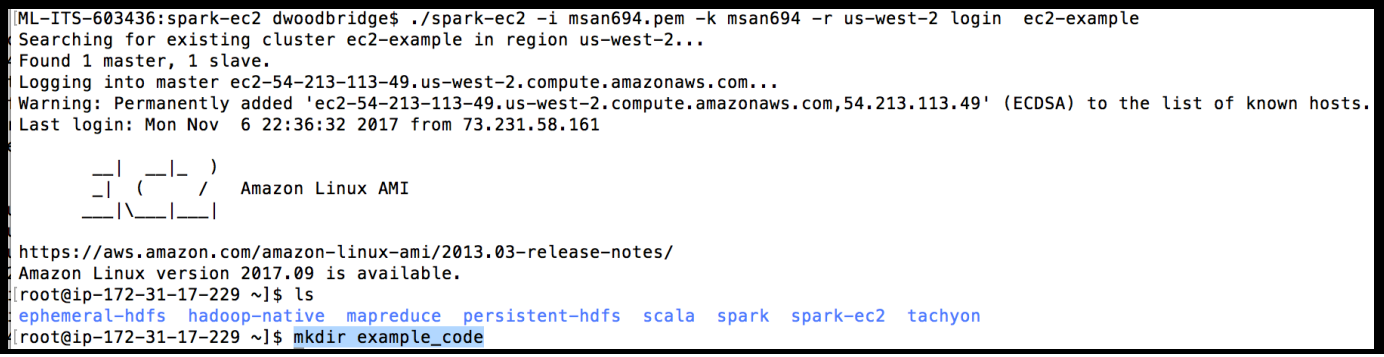
Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Running spark-submit
  - Copy Files (If you want to first write code on your local machine)
    - Create a folder on the master node:

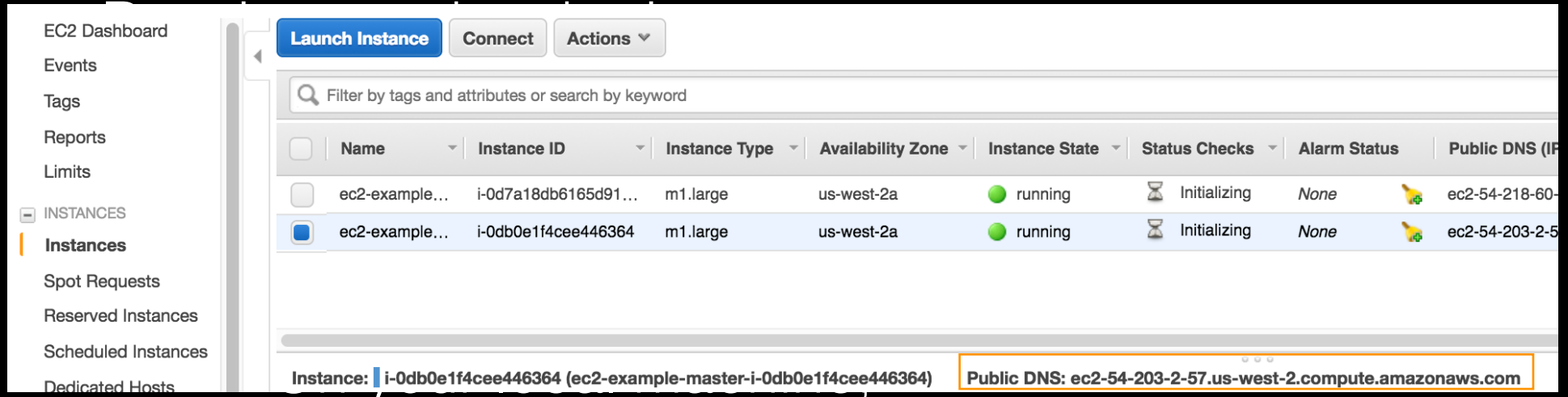
      `mkdir DIR`
    - On your local machine,

`scp -i msan694.pem -r YourDir root@ec2-54-203-2-57.us-west-2.compute.amazonaws.com:~/DIR`



Running on a Spark Standalone Cluster

Running on Amazon EC2.




```
scp -i Dir/msan694.pem -r YourDir/* root@ec2-54-203-2-57.us-west-2.compute.amazonaws.com:~/DIR
```

```
ML-ITS-603436:spark-ec2 dwoodbridge$ ./spark-ec2 -i msan694.pem -k msan694 -r us-west-2 login  ec2-example
Searching for existing cluster ec2-example in region us-west-2...
Found 1 master, 1 slave.
Logging into master ec2-54-213-113-49.us-west-2.compute.amazonaws.com...
Warning: Permanently added 'ec2-54-213-113-49.us-west-2.compute.amazonaws.com,54.213.113.49' (ECDSA) to the list of known hosts.
Last login: Mon Nov  6 22:36:32 2017 from 73.231.58.161

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2013.03-release-notes/
Amazon Linux version 2017.09 is available.
[root@ip-172-31-17-229 ~]$ ls
ephemeral-hdfs  hadoop-native  mapreduce  persistent-hdfs  scala  spark  spark-ec2  tachyon
[root@ip-172-31-17-229 ~]$ mkdir example_code
```

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Running spark-submit
  - RSYNC (Remote copy to workers)
    - On the master, copy the folder to worker nodes.

`~/spark-ec2/copy-dir ~/DIR`

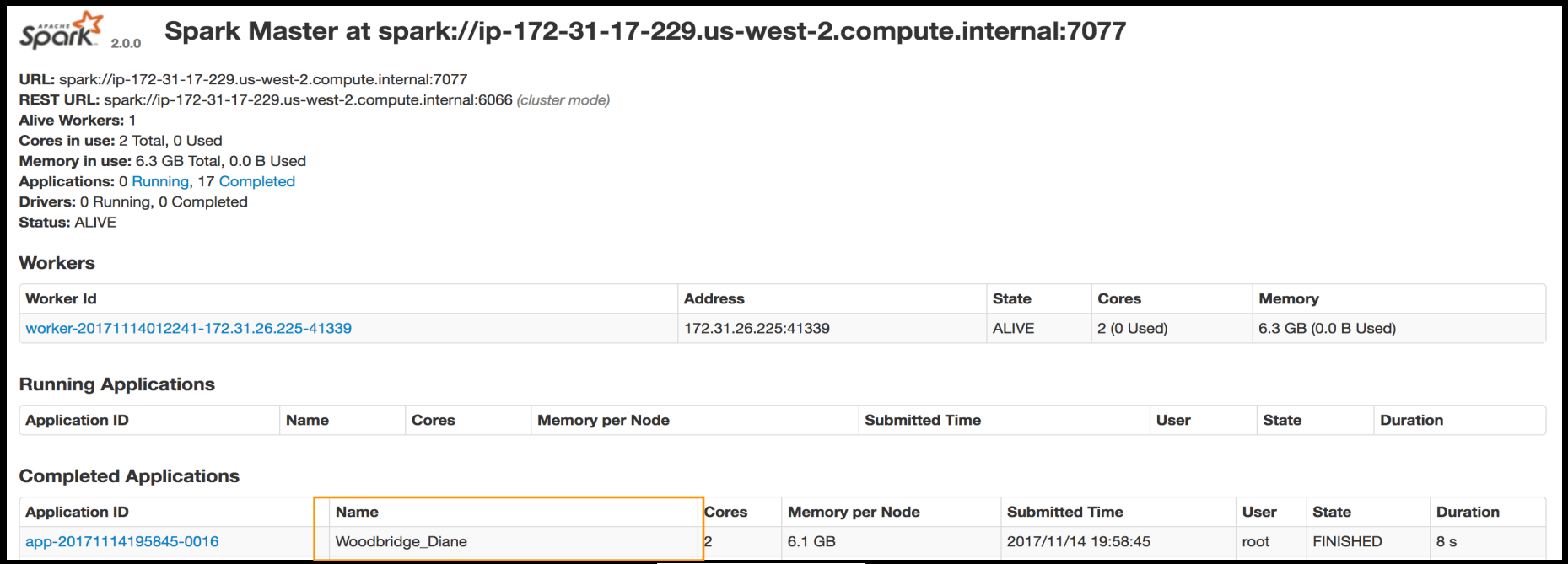
```
root@ip-172-31-17-229 ~]$ ls
ephemeral-hdfs  example_code   mapreduce          scala  spark-ec2
example         hadoop-native  persistent-hdfs    spark  tachyon
root@ip-172-31-17-229 ~]$ spark-ec2/copy-dir ~/example_code
RSYNC'ing /root/example_code to slaves...
ec2-54-218-60-36.us-west-2.compute.amazonaws.com
```

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Submit a job

```
~/spark/bin/spark-submit --deploy-mode
client example.py > output.txt
```



Spark 2.0.0 **Spark Master at spark://ip-172-31-17-229.us-west-2.compute.internal:7077**

URL: spark://ip-172-31-17-229.us-west-2.compute.internal:7077
REST URL: spark://ip-172-31-17-229.us-west-2.compute.internal:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 6.3 GB Total, 0.0 B Used
Applications: 0 Running, 17 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20171114012241-172.31.26.225-41339	172.31.26.225:41339	ALIVE	2 (0 Used)	6.3 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

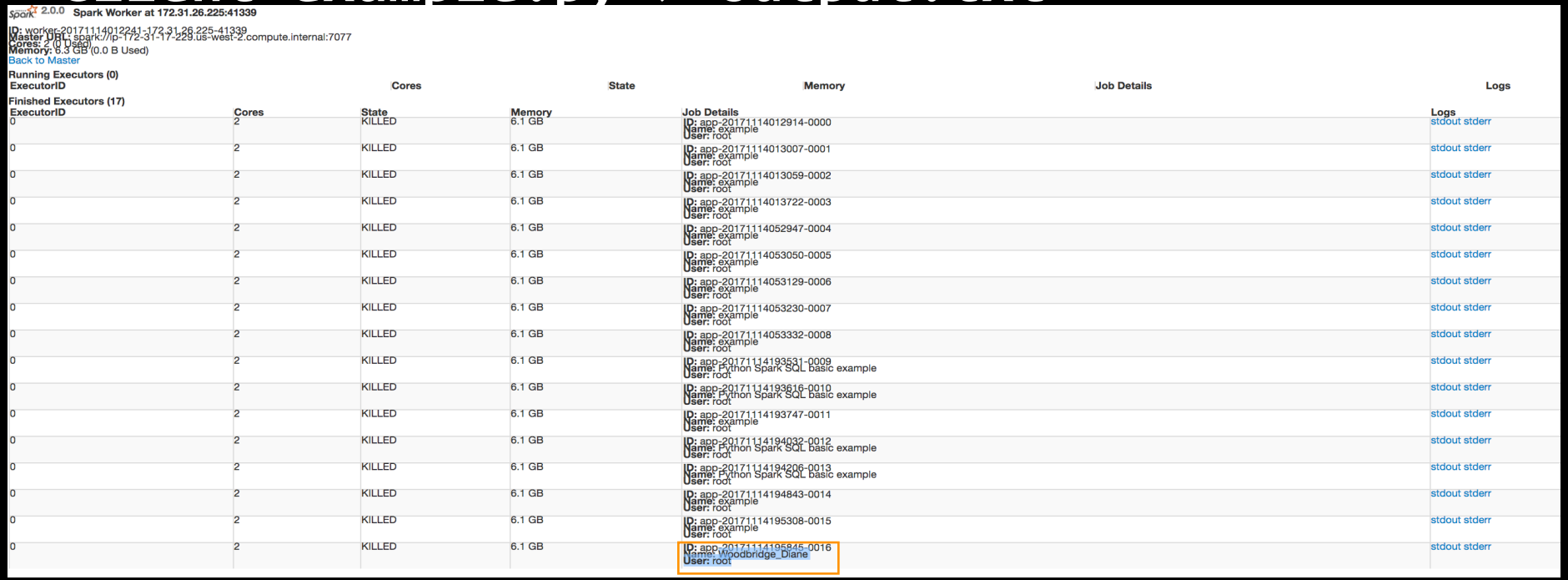
Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20171114195845-0016	Woodbridge_Diane	2	6.1 GB	2017/11/14 19:58:45	root	FINISHED	8 s

Running on a Spark Standalone Cluster

Running on Amazon EC2

- Submit a job

```
~/spark/bin/spark-submit --deploy-mode
client example.py > output.txt
```



Spark 2.0.0 Spark Worker at 172.31.26.225:41339

ID: worker-20171114012241-172.31.26.225-41339
Master URL: spark://ip-172-31-17-229.us-west-2.compute.internal:7077
Cores: 2 (0 Used)
Memory: 6.3 GB (0.0 B Used)
Back to Master

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs

Finished Executors (17)

ExecutorID	Cores	State	Memory	Job Details	Logs
0	2	KILLED	6.1 GB	ID: app-20171114012914-0000 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114013007-0001 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114013059-0002 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114013722-0003 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114052947-0004 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114053050-0005 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114053129-0006 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114053230-0007 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114053332-0008 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114193531-0009 Name: Python Spark SQL basic example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114193616-0010 Name: Python Spark SQL basic example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114193747-0011 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114194032-0012 Name: Python Spark SQL basic example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114194206-0013 Name: Python Spark SQL basic example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114194843-0014 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114195308-0015 Name: example User: root	stdout stderr
0	2	KILLED	6.1 GB	ID: app-20171114195845-0016 Name: Woodbridge_Diane User: root	stdout stderr

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Spark Standalone Cluster

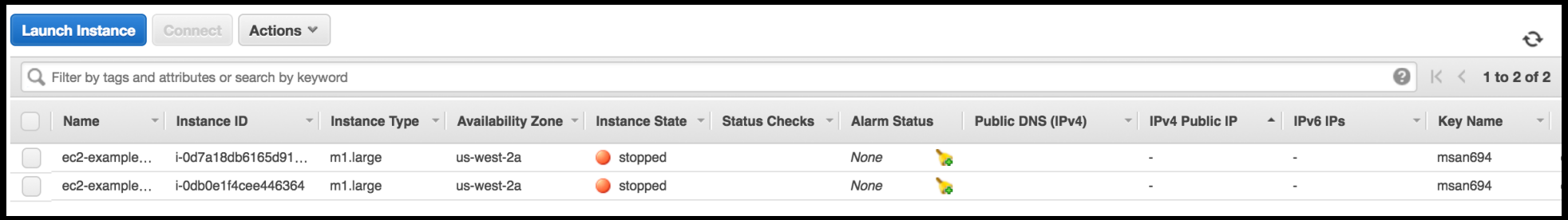
Running on Amazon EC2.

- Stop the cluster

```
./spark-ec2 -r us-west-2  -z us-west-2a stop ec2-example
```

If you don't want to be charged, don't forget to stop or terminate the cluster.

```
ML-ITS-603436:spark-ec2 dwoodbridge$  ./spark-ec2 -r us-west-2 stop ec2-example
Are you sure you want to stop the cluster ec2-example?
DATA ON EPHEMERAL DISKS WILL BE LOST, BUT THE CLUSTER WILL KEEP USING SPACE ON
AMAZON EBS IF IT IS EBS-BACKED!!
All data on spot-instance slaves will be lost.
Stop cluster ec2-example (y/N): y
Searching for existing cluster ec2-example in region us-west-2...
Found 1 master, 1 slave.
Stopping master...
Stopping slaves...
```



Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Start the cluster

```
./spark-ec2 -k msan694 -i msan694.pem -r us-west-2 -z us-west-2a start ec2-example
```

```
ML-ITS-603436:spark-ec2 dwoodbridge$ ./spark-ec2 -k msan694 -i msan694.pem -s 1 -r us-west-2 -z us-west-2a start ec2-example
Searching for existing cluster ec2-example in region us-west-2...
Found 1 master, 1 slave.
Starting slaves...
Starting master...
Waiting for cluster to enter 'ssh-ready' state.
Cluster is now in 'ssh-ready' state. Waited 2 seconds.
Cloning spark-ec2 scripts from https://github.com/amplab/spark-ec2/tree/branch-2.0 on master...
Warning: Permanently added 'ec2-54-213-140-206.us-west-2.compute.amazonaws.com,54.213.140.206' (ECDSA) to the list of known hosts.
Cloning into 'spark-ec2'...
remote: Counting objects: 2159, done.
```

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Spark Standalone Cluster

Running on Amazon EC2.

- Terminate the cluster

If you don't want to be charged, don't forget to stop or terminate the cluster.

```
./spark-ec2 -k msan694 -i msan694.pem -r us-west-2 -z us-west-2a destroy ec2-example
```

```
ML-ITS-603436:spark-ec2 dwoodbridge$ ./spark-ec2  -i msan694.pem  -k msan694 -r us-west-2 -z us-west-2a destroy
 ec2-example
Searching for existing cluster ec2-example in region us-west-2...
Found 1 master, 1 slave.
The following instances will be terminated:
> ec2-54-203-2-57.us-west-2.compute.amazonaws.com
> ec2-54-218-60-36.us-west-2.compute.amazonaws.com
ALL DATA ON ALL NODES WILL BE LOST!!
Are you sure you want to destroy the cluster ec2-example? (y/N)
```

References

Distributed Computing with Spark, Reza Zadeh, http://stanford.edu/~rezab/slides/bayacm_spark.pdf

Spark Online Documentation : http://spark.apache.org/docs/latest/

Karau, Holden, et al. Learning spark: lightning-fast big data analysis. O'Reilly Media, Inc., 2015.

Zecevic, Petar, et al. Spark in Action, Manning, 2016.

HW5

Spark Master at spark://ip-172-31-17-229.us-west-2.compute.internal:7077

URL: spark://ip-172-31-17-229.us-west-2.compute.internal:7077
REST URL: spark://ip-172-31-17-229.us-west-2.compute.internal:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 6.3 GB Total, 0.0 B Used
Applications: 0 Running, 17 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20171114012241-172.31.26.225-41339	172.31.26.225:41339	ALIVE	2 (0 Used)	6.3 GB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20171114195845-0016	Woodbridge_Diane	2	6.1 GB	2017/11/14 19:58:45	root	FINISHED	8 s

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

HW5

Spark 2.0.0 **Spark Worker at 172.31.26.225:41339**

ID: worker-20171114012241-172.31.26.225-41339
Master URL: spark://ip-172-31-17-229.us-west-2.compute.internal:7077
Cores: 2 (0 Used)
Memory: 6.3 GB (0.0 B Used)
Back to Master

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs

Finished Executors (17)

ExecutorID	Cores	State	Memory	Job Details	Logs
0	2	KILLED	6.1 GB	**ID:** app-20171114012914-0000 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114013007-0001 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114013059-0002 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114013722-0003 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114052947-0004 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114053050-0005 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114053129-0006 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114053230-0007 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114053332-0008 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114193531-0009 **Name:** Python Spark SQL basic example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114193616-0010 **Name:** Python Spark SQL basic example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114193747-0011 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114194032-0012 **Name:** Python Spark SQL basic example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114194206-0013 **Name:** Python Spark SQL basic example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114194843-0014 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114195308-0015 **Name:** example **User:** root	stdout stderr
0	2	KILLED	6.1 GB	**ID:** app-20171114195845-0016 **Name:** Woodbridge_Diane **User:** root	stdout stderr

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO