MSAN 694 : Distributed Computing

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

Reviews

Running on a Spark Cluster

Running on a Spark Standalone Cluster

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 YARN Cluster

YARN (Yet Another Resource Negotiator)

- Hadoop's resource manger and execution system.
  - Most Hadoop installations have YARN configured alongside HDFS.
  - Many organizations already have YARN clusters of a significant size, along with the technical know-how, tools, and procedures for managing and monitoring them.
- You don't have to install Spark on all nodes in the cluster.
- YARN lets you run different types of Java applications, not just Spark, so you can mix legacy Hadoop and Spark applications with ease.
- YARN provides methods for isolating and prioritizing applications among users and organizations, functionality the standalone cluster doesn't have.
- It's the only cluster type that supports Kerberos-secured HDFS (Secure HDFS mode) and authorizes access to resources to certain users using Kerberos-provided identity and access control lists in the Hadoop configuration.

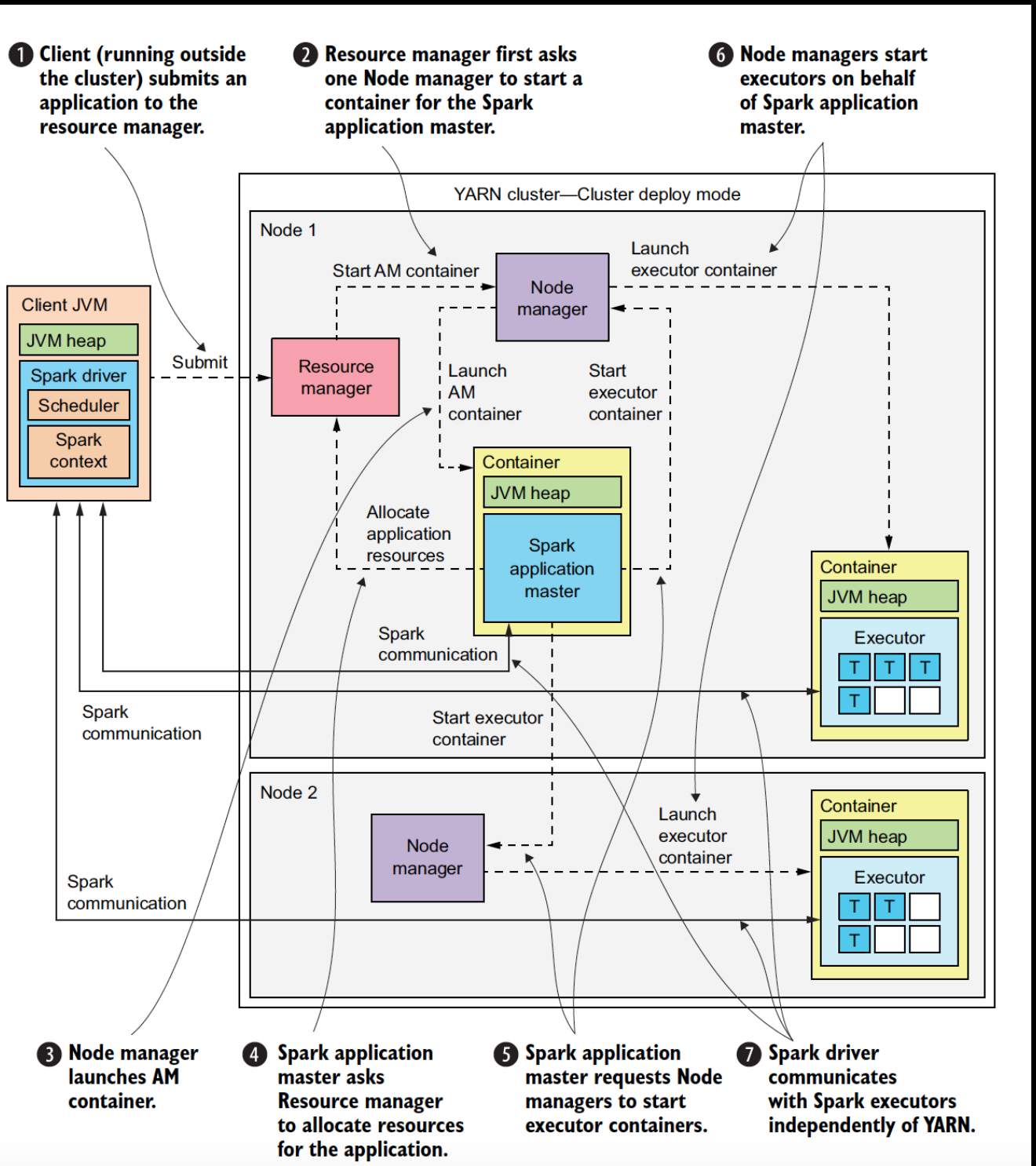
Running on a YARN Cluster

YARN (Yet Another Resource Negotiator)

- Architecture
  - Unlike running on Spark's standalone cluster, applications on YARN run in containers.
    - Container - JVM process to which CPU and memory resource are granted.
  - Cluster Components
    1. Resource Manger : Similar to Spark's master process.
       - Allocate an application master via a node manager when an application is submitted.
    2. Node Manager : Similar to Spark's worker process.
       - Launch executor containers.
    3. Application Master : Similar to cluster manager.
       - Responsible for negotiating resources with resource manager and launches and monitors containers with executors.

YARN

- Architecture
  - Example cluster of two nodes in client-deploy mode.



Running on a YARN Cluster

Amazon EMR.

- Apache Spark on Hadoop YARN is natively supported in Amazon EMR, and you can quickly and easily create managed Apache Spark clusters from the AWS. Management Console, AWS CLI, or the Amazon EMR API.
- You can leverage additional Amazon EMR features, including fast Amazon S3 connectivity using the Amazon EMR File System (EMRFS), integration with the Amazon EC2 Spot market, and Auto Scaling to add or remove instances from your cluster.

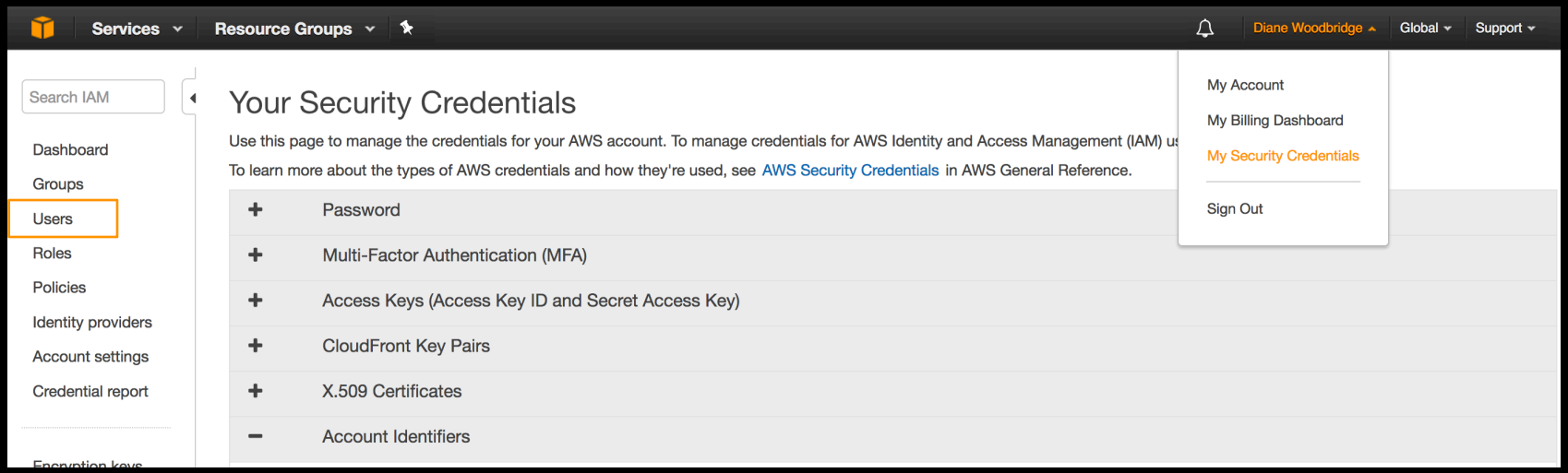
http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-spark.html
http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-spark-shell.html

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a YARN Cluster

Running on Amazon EMR.

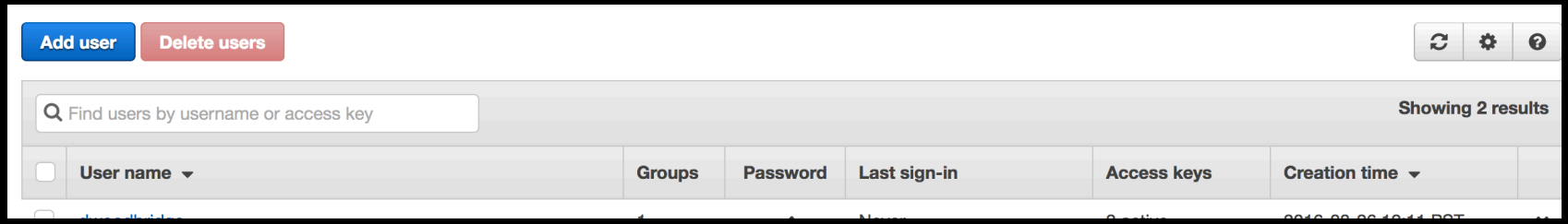
- On your AWS console, go to the user information ➔ My Security Credentials. ➔ Choose Users.



Running on a YARN Cluster

Running on Amazon EMR.

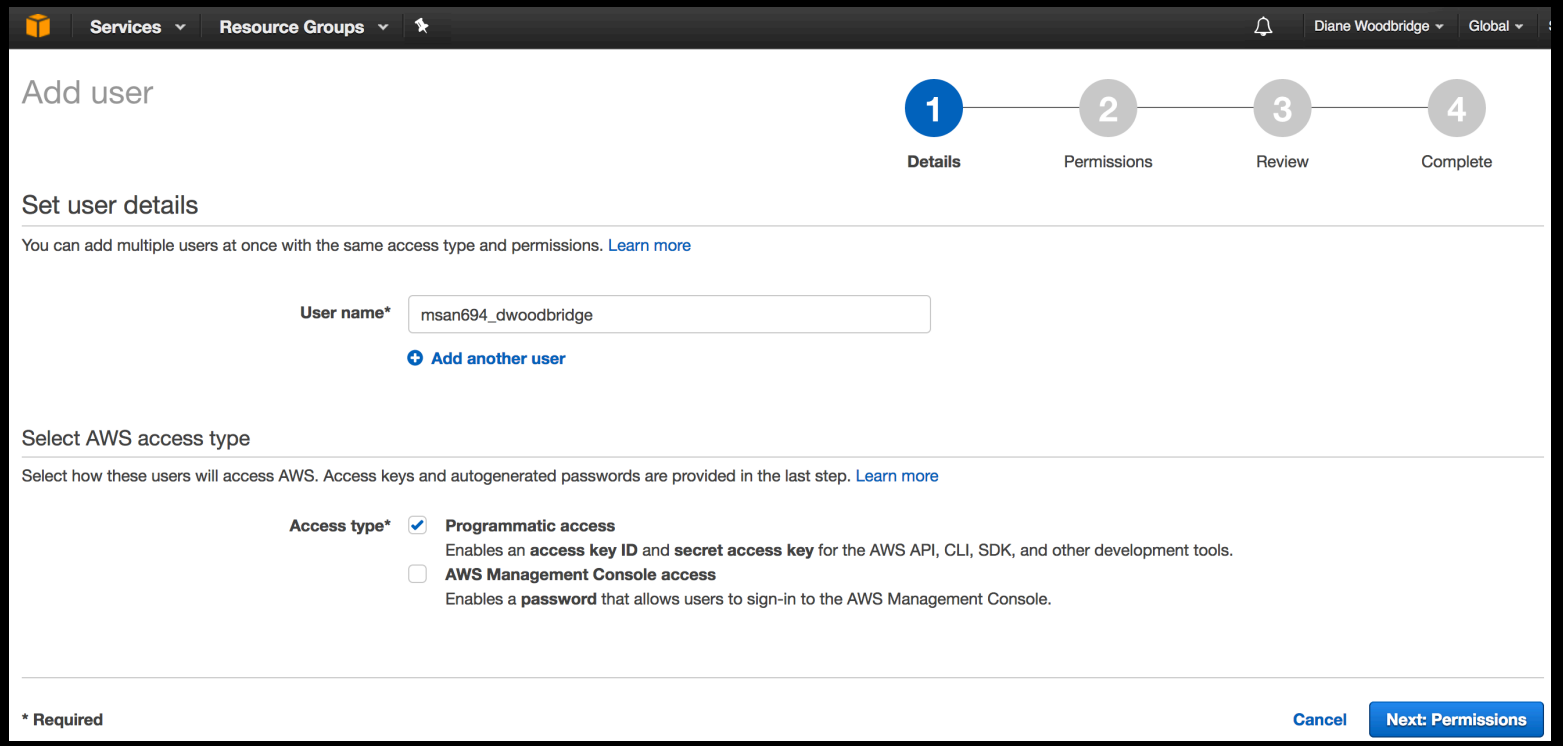
- Choose Add User.



Running on a YARN Cluster

Running on Amazon EMR.

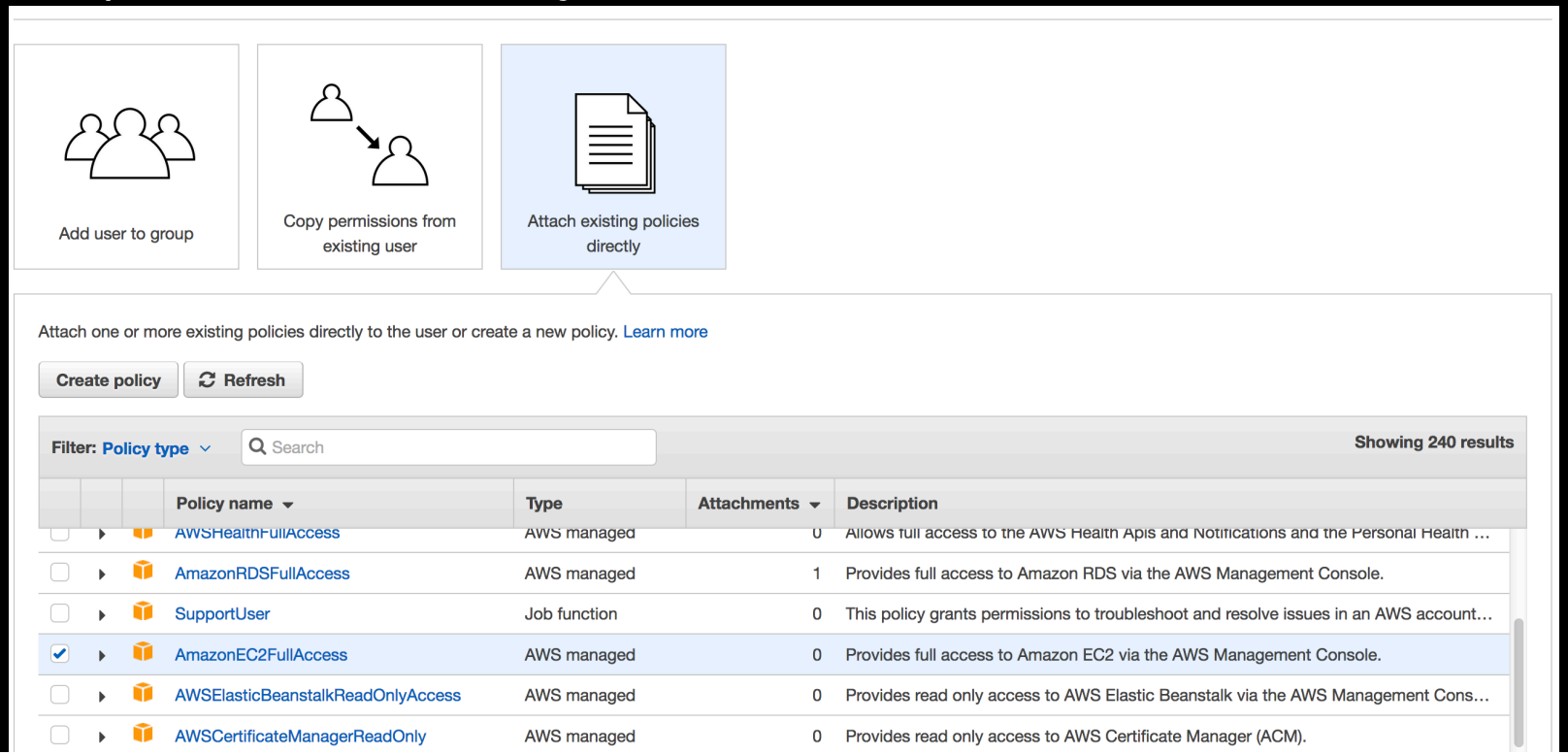
- Choose a user name and access type.



Running on a YARN Cluster

Running on Amazon EMR.

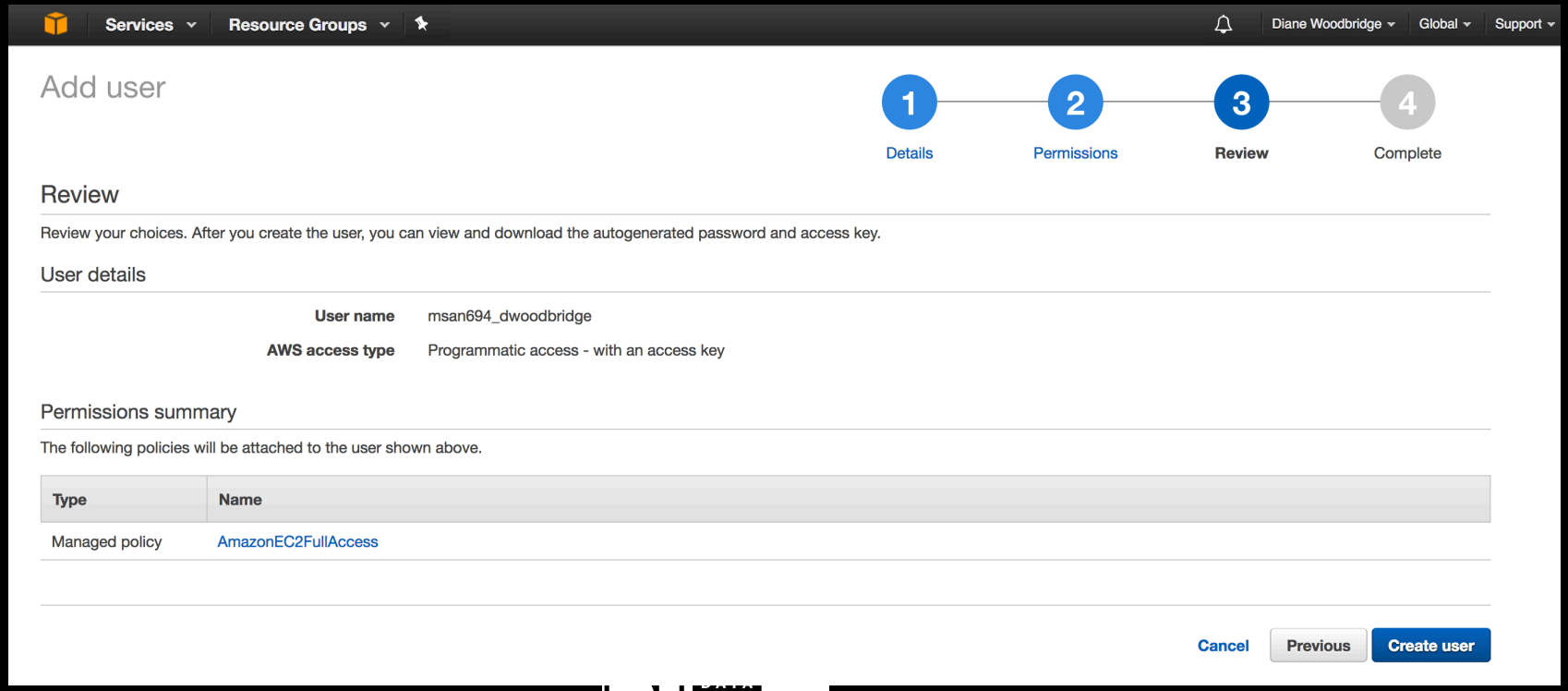
- Set permissions for the user. : Attach existing policies directly. ➔ Amazon EC2FullAccess.



Running on a YARN Cluster

Running on Amazon EMR.

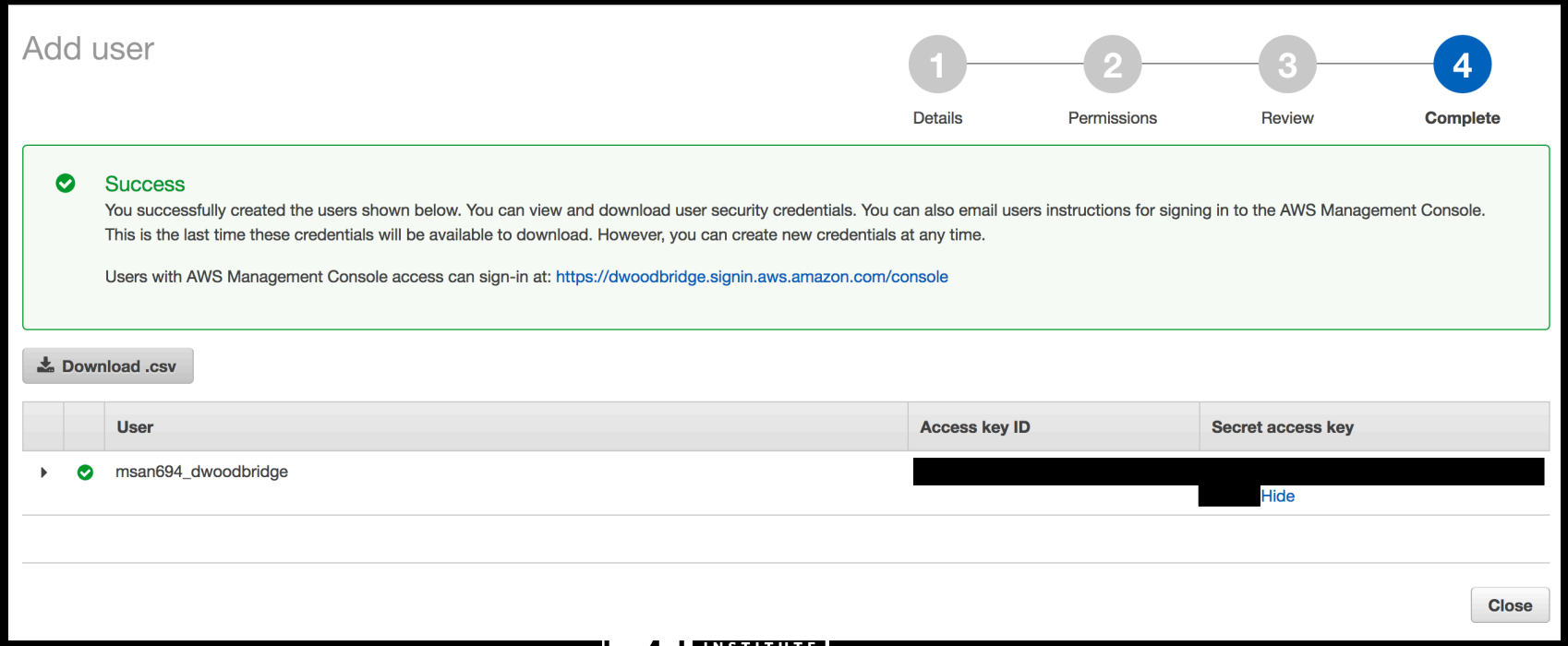
- Create User.



Running on a YARN Cluster

Running on Amazon EMR.

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



Running on a YARN Cluster

Running on Amazon EMR.

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

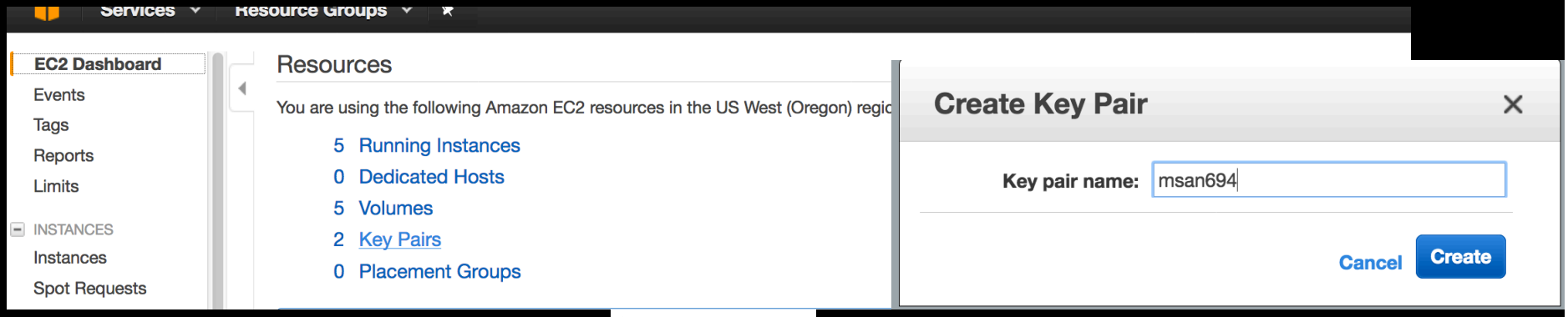
```
export AWS_ACCESS_KEY_ID=AKIAI
export AWS_SECRET_ACCESS_KEY=b
```

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.

Running on a YARN Cluster

Running on Amazon EMR.

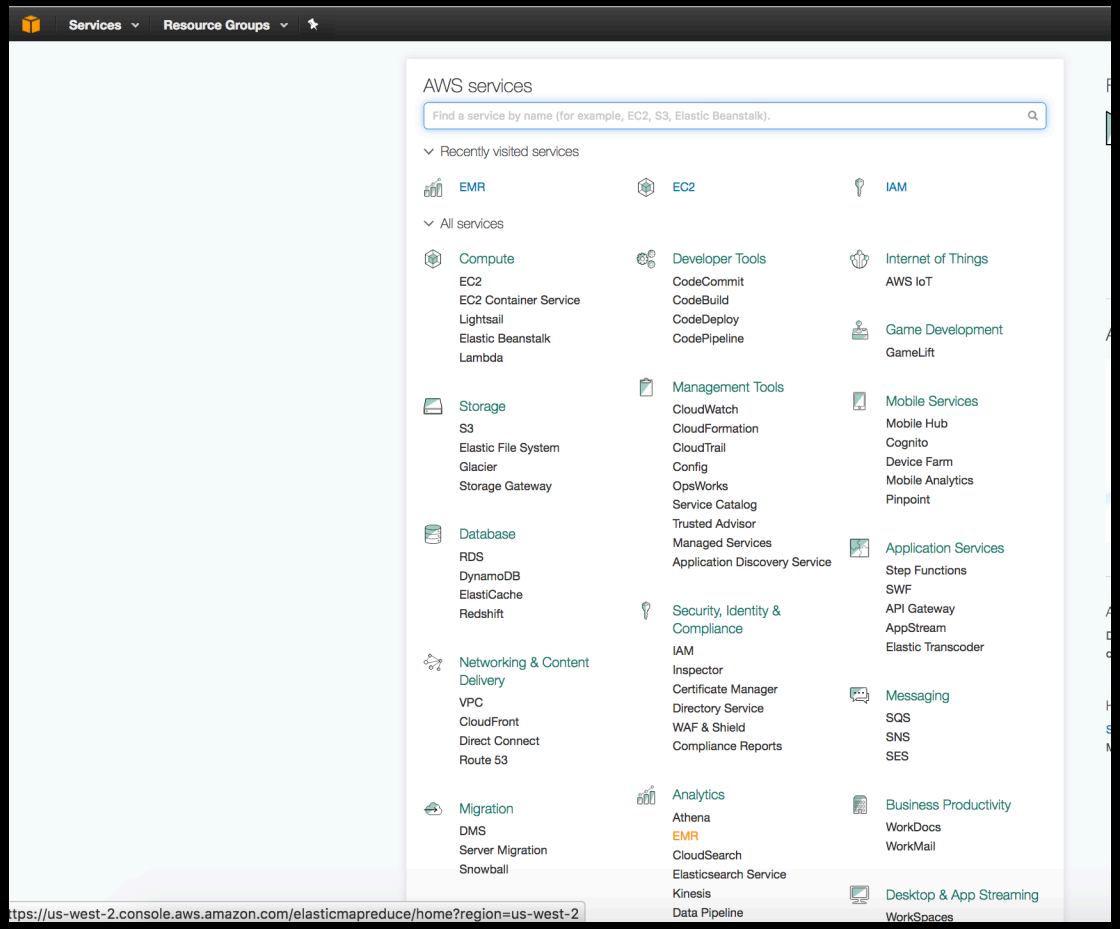
- On EC2 Console (https://console.aws.amazon.com/ec2/), Choose Key Pairs ➔ Create Key Pairs , download the .pem and store somewhere on your machine*.
- Make sure that you set the permissions for the private key file to 600 (i.e. only you can read and write it) so that ssh will work.
  - `chmod 600 filename`



Running on a YARN Cluster

Running on Amazon EMR.

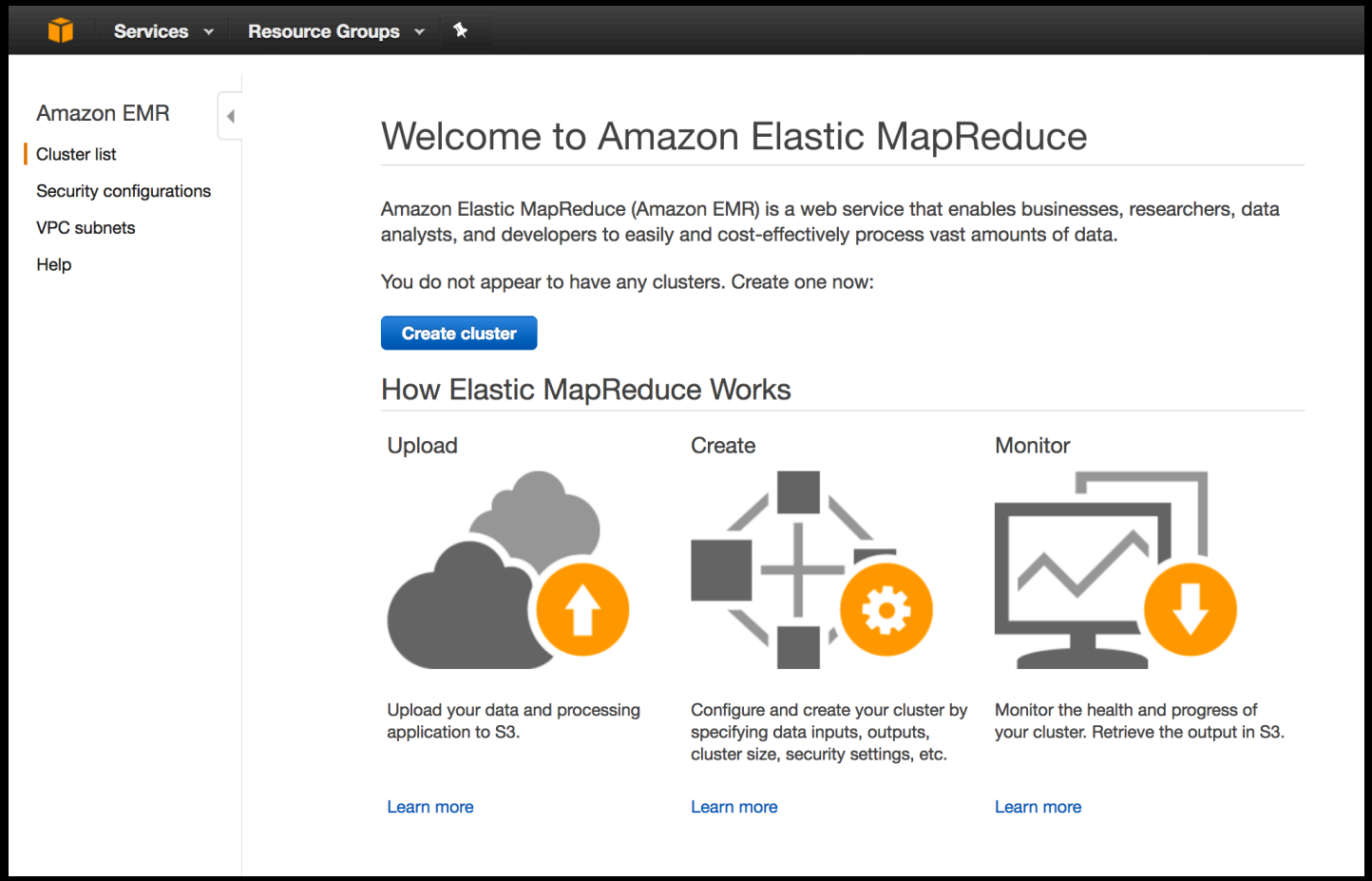
- On your AWS console, choose EMR under Analysis.



Running on a YARN Cluster

Running on Amazon EMR.

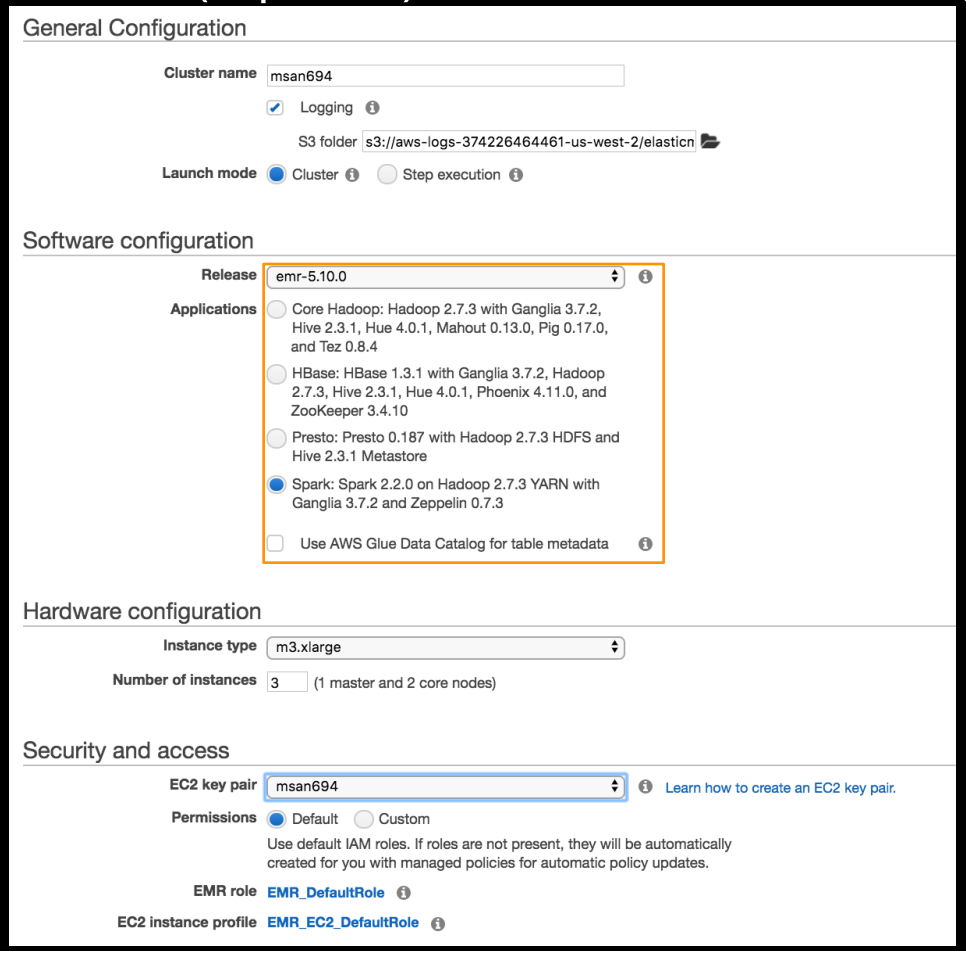
- Choose ”Create cluster”.



Running on a YARN Cluster

Running on Amazon EMR.

- Insert all the information including Cluster name, Applications (Spark), Hardware configuration.

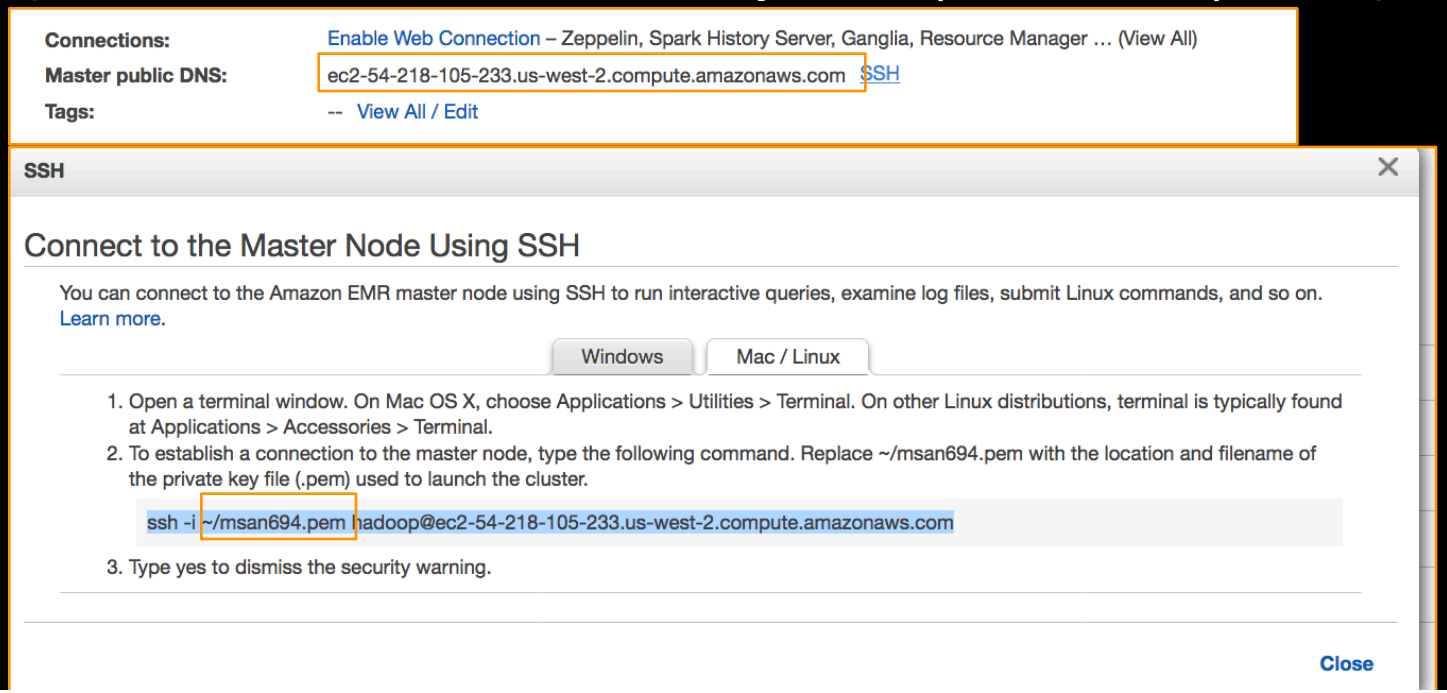


DONE!

Running on a YARN Cluster

Running on Amazon EMR.

- Once your cluster is ready, click SSH on your Master public DNS and copy the command. (Make sure to indicate your .pem file path.)



Running on YARN

Running on Amazon EMR.

- Run the command and say “yes” when it asks ”Are you sure you want to continue connecting?”

```
[ML-ITS-603436:spark-2.0.0-bin-hadoop2.7 dwoodbridge$ ssh -i msan694.pem  hadoop@ec2-54-218-105-233.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-54-218-105-233.us-west-2.compute.amazonaws.com (54.218.105.233)' can't be established.
ECDSA key fingerprint is SHA256:lvuUviA9QfdsQoyEngU4aEQ8eqyNOa21uv1g2BWK16I.
Are you sure you want to continue connecting (yes/no)? yes
```

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on YARN

Running on Amazon EMR.

```
ML-ITS-603436:spark-2.0.0-bin-hadoop2.7 dwoodbridge$ ssh -i msan694.pem  hadoop@ec2-54-218-105-233.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-54-218-105-233.us-west-2.compute.amazonaws.com (54.218.105.233)' can't be established.
ECDSA key fingerprint is SHA256:lvuUviA9QfdsQoyEngU4aEQ8eqyNOa21uv1g2BWK16I.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-218-105-233.us-west-2.compute.amazonaws.com,54.218.105.233' (ECDSA) to the list of known hosts.
Last login: Sun Jan  1 02:12:37 2017

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

https://aws.amazon.com/amazon-linux-ami/2016.09-release-notes/

EEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-11-131 ~]$
```

Running on YARN

Running on Amazon EMR.

- Try pyspark.

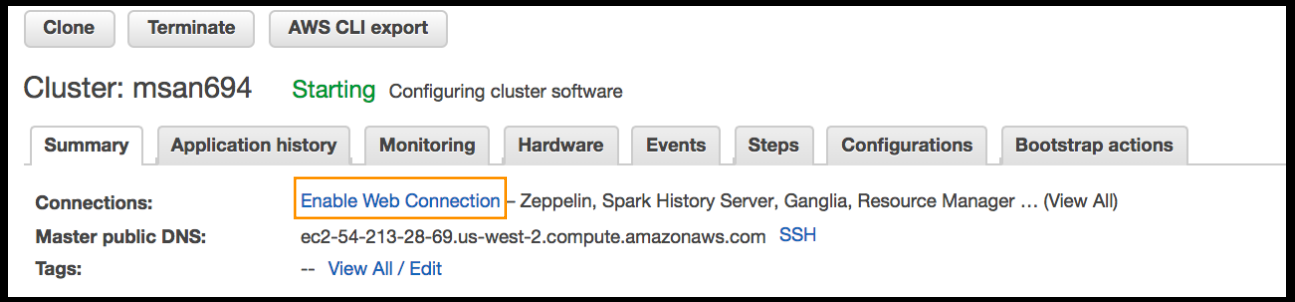
```
[hadoop@ip-172-31-11-131 ~]$ pyspark
Python 2.7.12 (default, Sep  1 2016, 22:14:00)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/01/01 02:21:37 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.2
      /_/

Using Python version 2.7.12 (default, Sep  1 2016 22:14:00)
SparkSession available as 'spark'.
>>>
```

Running on a YARN Cluster

Running on Amazon EMR.

- If you want to enable Zeppelin notebook, Resource Manager, etc., follow the direction in “Enable Web Connection” to create an ssh tunnel for web access.



- You can also install Jupyter notebook (The script is in the Example repository.)

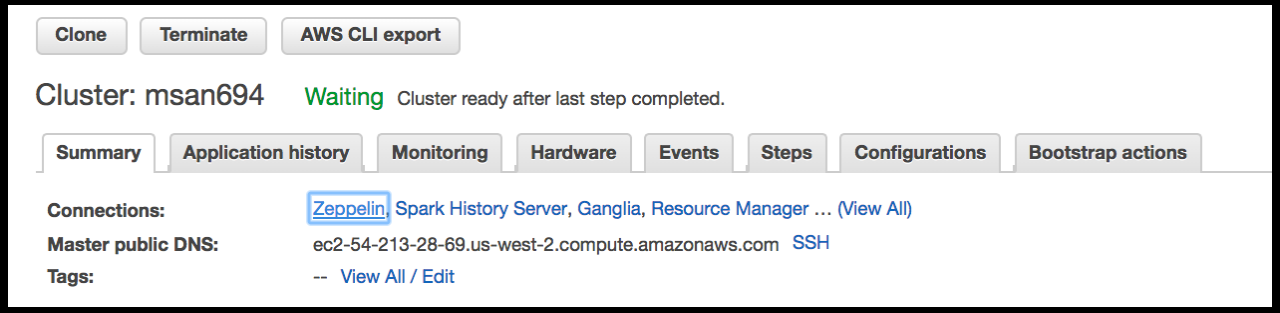
http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-connect-master-node-proxy.html

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a YARN Cluster

Running on Amazon EMR.

- If you want to enable Zeppelin notebook , Resource Manager, etc., follow the direction in "Enable Web Connection" to create an ssh tunnel for web access.



- You can also install Jupyter notebook (The script is in the Example repository.)

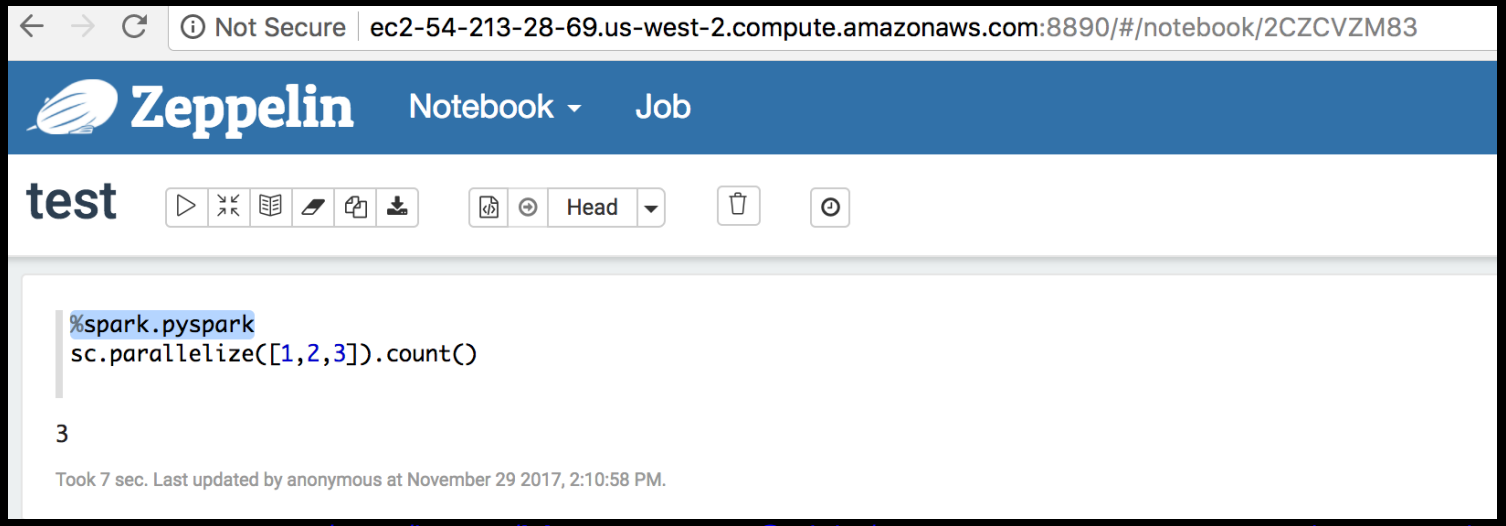
http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-connect-master-node-proxy.html

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a YARN Cluster

Running on Amazon EMR.

- If you want to enable Zeppelin notebook , Resource Manager, etc., follow the direction in "Enable Web Connection" to create an ssh tunnel for web access.

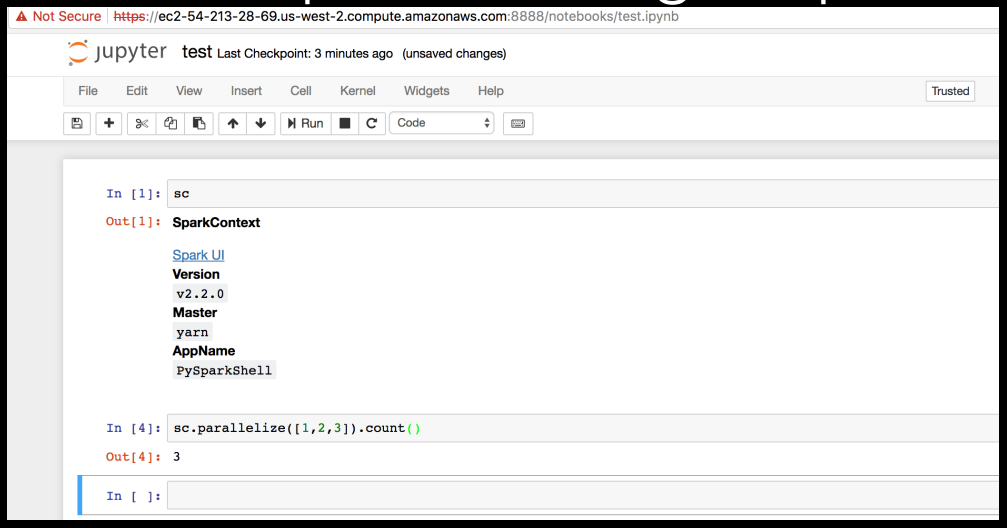


http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-connect-master-node-proxy.html

Running on a YARN Cluster

Running on Amazon EMR.

- If you want to enable Zeppelin notebook , Resource Manager, etc., follow the direction in “Enable Web Connection” to create an ssh tunnel for web access.
- You can also install Jupyter notebook (See Appendix. The script is in the git repository.)



Running on a YARN Cluster

Running on Amazon EMR.

- Run an application to cluster.
  - spark-submit code_name.py

2017_MSAN694 — hadoop@ip-172-31-37-158:~ — ssh -

```
from pyspark import SparkConf, SparkContext

#Create SparkContext
conf = SparkConf().setAppName("example")
sc = SparkContext(conf = conf)

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

sc.stop()
~
```

```
[hadoop@ip-172-31-37-158 ~]$ spark-submit --master yarn test.py > output.txt
17/11/30 22:58:17 INFO SparkContext: Running Spark version 2.2.0
17/11/30 22:58:18 INFO SparkContext: Submitted application: example
17/11/30 22:58:18 INFO SecurityManager: Changing view acls to: hadoop
17/11/30 22:58:18 INFO SecurityManager: Changing modify acls to: hadoop
17/11/30 22:58:18 INFO SecurityManager: Changing view acls groups to:
17/11/30 22:58:18 INFO SecurityManager: Changing modify acls groups to:
17/11/30 22:58:18 INFO SecurityManager: SecurityManager: authentication disable
```

https://spark.apache.org/docs/latest/submitting-applications.html
https://spark.apache.org/docs/latest/configuration.html
https://spark.apache.org/docs/latest/running-on-yarn.html

Running on a YARN Cluster

```
–master yarn
–deploy-mode client
–driver-memory 1024M
–executor-memory 2048M
–num-executors 2
–conf spark.yarn.executor.memoryOverhead=512
```

Running on Amazon EMR.

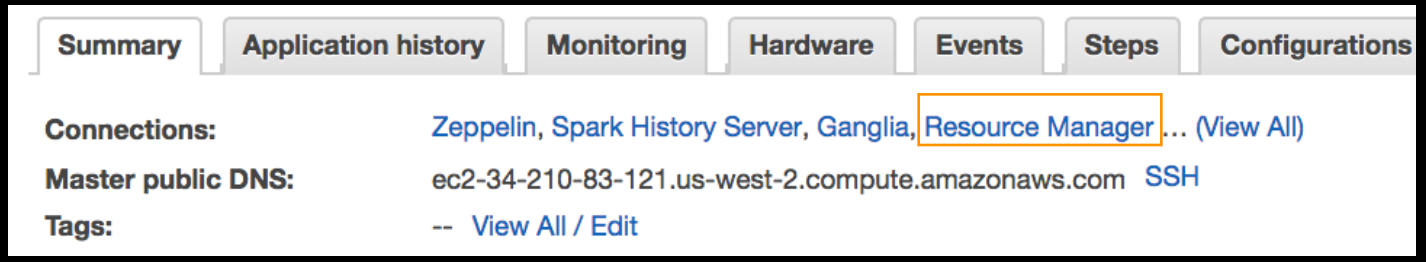
- Run an application to cluster.
  - Parameters to be configured (do this for the other schedulers.)
    - --master : The master URL for the cluster.
    - --deploy-mode : client (default) or cluster.
    - --driver-memory : Amount of memory to use for the driver process.
    - --executor-memory : Amount of memory to use per executor process.
    - --conf spark.yarn.executor.memoryOverhead : The amount of off-heap memory (in megabytes) to be allocated per executor.
    - And many more…

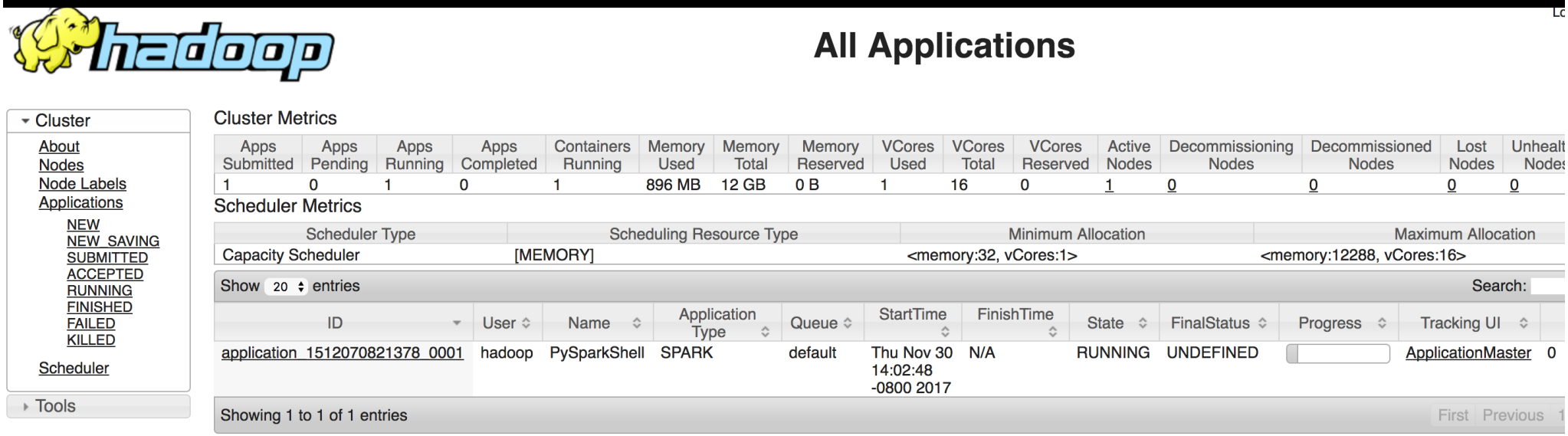
https://spark.apache.org/docs/latest/submitting-applications.html
https://spark.apache.org/docs/latest/configuration.html
https://spark.apache.org/docs/latest/running-on-yarn.html

Running on a YARN Cluster

YARN UI (Resource Manager, Port 8088)

- Monitors the cluster state (capacity usage, log files, and the status of applications)





Contents

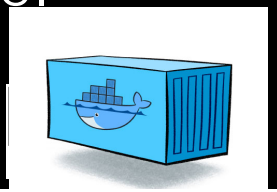
DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Running on a Mesos Cluster

Mesos

- The spark project was originally started in order to demonstrate the usefulness of Mesos.
- Provides fine-grained/coarse-grained resource sharing.
- Support applications written in Java, C, C++, and Python.
- Run Docker containers
  - Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries – anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.
  - With Docker support, you can run Mesos on any applications that can run in a Docker Container.
- Example applications running on Mesos – Apple Siri, eBay, Netflix, Twitter, Uber, etc.

http://spark.apache.org/docs/latest/running-on-mesos.html
https://www.docker.com/



Contents

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Loading and Saving Data

Spark supports unstructured/semi-structured/structured types of data.
File Formats include the followings (and many more).

Format	Structured	Comments
Text File	No	One line is one element for a single file.
JSON	Semi	Most libraries require one record per line.
CSV/TSV	Yes	Contains fixed number of fields per line.
Sequence File	Yes	Hadoop file format used for key-value data.
Object File	Yes	Used for saving data from a Spark job to be consumed by shared code.

Loading and Saving Data

File Formats

- Text Format
  - Load
    - One line is one element in a single file.
      - Use textFile(file_name) method.
    - Can load multiple text files as **pair RDDs**, with the key being the file name and the value being the file content.
      - Use wholeTextFiles(dir _name).
      - Useful when each file represent a certain time period's data and need to use it for statistics. Ex. sales stat.
  - Save
    - Use saveAsTextFile(new_subdir_name) method.
    - Return multiple output files underneath the new_subdir_name, as Spark writes the output from multiple partitions.

Example 1

Load all the .csv files under a directory as Pair RDDs with key being a file name and values being the content of the file.

Example 2

From "supervisor_sf.csv”, filter data including “94103” and save under ”supervisor_94103”.

Todo : Change the number of partitions, and see the output.

Loading and Saving Data

File Formats

- JSON
  - Use python's json package.
  - Load
    - Use json.loads() method.

```
import json

data = input.map(lamdbda x: json.loads(x))
```

    - Load one JSON record per row.
  - Use SparkSQL (Intersession).
- Save
  - Use json.dumps() and then saveAsTextFile().

```
json_output = data.map(lamdbda x: json.dumps(x))
```

Example 3

Load ”example.json”.
Filter item that has 3 in “array”.
Convert the filtered output in JSON format and save in “json_data_with_3” folder.

Loading and Saving Data

File Formats

- Comma Separated Value (CSV) and Tab separated Value (TSV)
  - Use python's csv package.
  - Load
    - Use textFile()/wholeTextFiles() and parse the csv to load the data.
    - And use StringIO.StringIO() to read a string buffer and csv.DictReader(input, *fieldnames*) to read information into a dict whose keys are given by the optional *fieldnames* parameter.
  - Save
    - Write a function to write each row using stringIO(),DictWriter(), WriteRow(), etc.
    - Save the file using saveAsTextFile().

Example 4

Read data from supervisor_sf.csv as a dictionary structure of (Zip, Supervisor).
Choose Zip starting with “94”.

Example 4

```
import csv
import StringIO
```

```
input = sc.textFile("../Data/supervisor_sf.csv")
```

```
def csvLoader(line):
    input = StringIO.StringIO(line)
    reader = csv.DictReader(input, fieldnames=["Zip","Supervisor"])
    return reader.next()
```

```
csv_data = input.map(csvLoader)
```

```
csv_data.collect()
```

```
sf_supervisor = csv_data.filter(lambda x :  (x['Zip']).startswith('94'))
```

```
sf_supervisor.collect()
```

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

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.

Appendix

Running on YARN

Running on Amazon EMR.

- Set Jupyter Notebook.
  - First install anaconda.

    `wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86_64.sh`

    `sh Anaconda3-4.1.1-Linux-x86_64.sh`
    - **Make sure to install on place with enough disk space. (df)**
    - Add "export PATH=/mnt/anaconda3/bin:$PATH" in your .bashrc.

```
[hadoop@ip-172-31-15-224 ~]$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
devtmpfs        15382916      28  15382888   1% /dev
tmpfs           15413684       0  15413684   0% /dev/shm
/dev/xvda1      10190136 6901256   3188612  69% /
/dev/xvdb1       5232640   36772   5195868   1% /emr
/dev/xvdb2      73352692  309812  73042880   1% /mnt
/dev/xvdc       78594056   34052  78560004   1% /mnt1
[hadoop@ip-172-31-15-224 ~]$ cd /mnt
```

https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297#.csw3m7v6b

Running on YARN

Running on Amazon EMR.

- Set Jupyter Notebook.
  - First install anaconda.

    `wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86_64.sh`

    `sh Anaconda3-4.1.1-Linux-x86_64.sh`
    - **Make sure to install on place with enough disk space. (df)**
    - Add "export PATH=/mnt/anaconda3/bin:$PATH" in your .bashrc.

```
Anaconda3 will now be installed into this location:
/home/hadoop/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/hadoop/anaconda3] >>> /mnt/anaconda3
```

https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297#.csw3m7v6b

Running on YARN

Running on Amazon EMR.

- Set Jupyter Notebook.
  - First install anaconda.

    `wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86_64.sh`

    `sh Anaconda3-4.1.1-Linux-x86_64.sh`
    - Add “export PATH=/mnt/anaconda3/bin:$PATH” in your .bashrc.
  - Change the python version to 2.7.

```
[hadoop@ip-172-31-32-214 ~]$ which python
/mnt/anaconda3/bin/python
```

```
[hadoop@ip-172-31-15-224 mnt]$ conda install python=2.7
Fetching package metadata .......
Solving package specifications: .......
```

https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297#.csw3m7v6b

Running on YARN

Running on Amazon EMR.

- Set Jupyter Notebook.
  - Change the notebook version to 5.2.1

```
[hadoop@ip-172-31-5-211 mnt]$ conda update jupyter notebook
Fetching package metadata ...........
Solving package specifications: 5.2.1
```

https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297#.csw3m7v6b

Running on YARN

Running on Amazon EMR.

- Set Jupyter Notebook.
  - Configure Jupyter
    - Jupyter comes with Anaconda, but we will need to configure it in order to use it through EC2 and connect with SSH.

      `jupyter notebook --generate-config`

```
[hadoop@ip-172-31-15-224 ~]$ source ~/.bashrc
[hadoop@ip-172-31-15-224 ~]$ jupyter notebook --generate-config
Writing default config to: /home/hadoop/.jupyter/jupyter_notebook_config.py
```

Running on YARN

Running on Amazon EMR.

- Set Jupyter Notebook.
  - Create certifications.
    - mkdir certs
    - cd certs
    - sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout msan694.pem -out msan694.pem

```
Generating a 1024 bit RSA private key
.........................++++++
.....................................................................................++++++
writing new private key to 'msan694.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:California
Locality Name (eg, city) [Default City]:San Francisco
Organization Name (eg, company) [Default Company Ltd]:USF
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:msan694
Email Address []:dwoodbridge@usfca.edu
```

Running on YARN

```
c = get_config()
c.NotebookApp.certfile = u'/home/hadoop/certs/diane_msan694.pem'
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.port = 8888
```

Running on Amazon EMR.

- Edit configuration file.
  - Change the configuration we generated.
    - cd ~/.jupyter
    - vi jupyter_notebook_config.py

```
# Configuration file for jupyter-notebook.

c= get_config()
c.NotebookApp.certfile = u'/home/hadoop/certs/msan694.pem'
c.NotebookApp.ip = '*' # Run on all IP addresses of your instance
c.NotebookApp.open_browser = False   # Don't open browser by default
c.NotebookApp.port = 8888 # Use port 8888 for notebook
```

 # Notebook config this is where you saved your pem cert

Running on YARN

Running on Amazon EMR.

- Add PYSPARK_DRIVER_PYTHON and PYSPARK_DRIVER_PYTHON_OPTS.

```
export PYSPARK_DRIVER_PYTHON=jupyter
export PYSPARK_DRIVER_PYTHON_OPTS="notebook"
```

Running on YARN

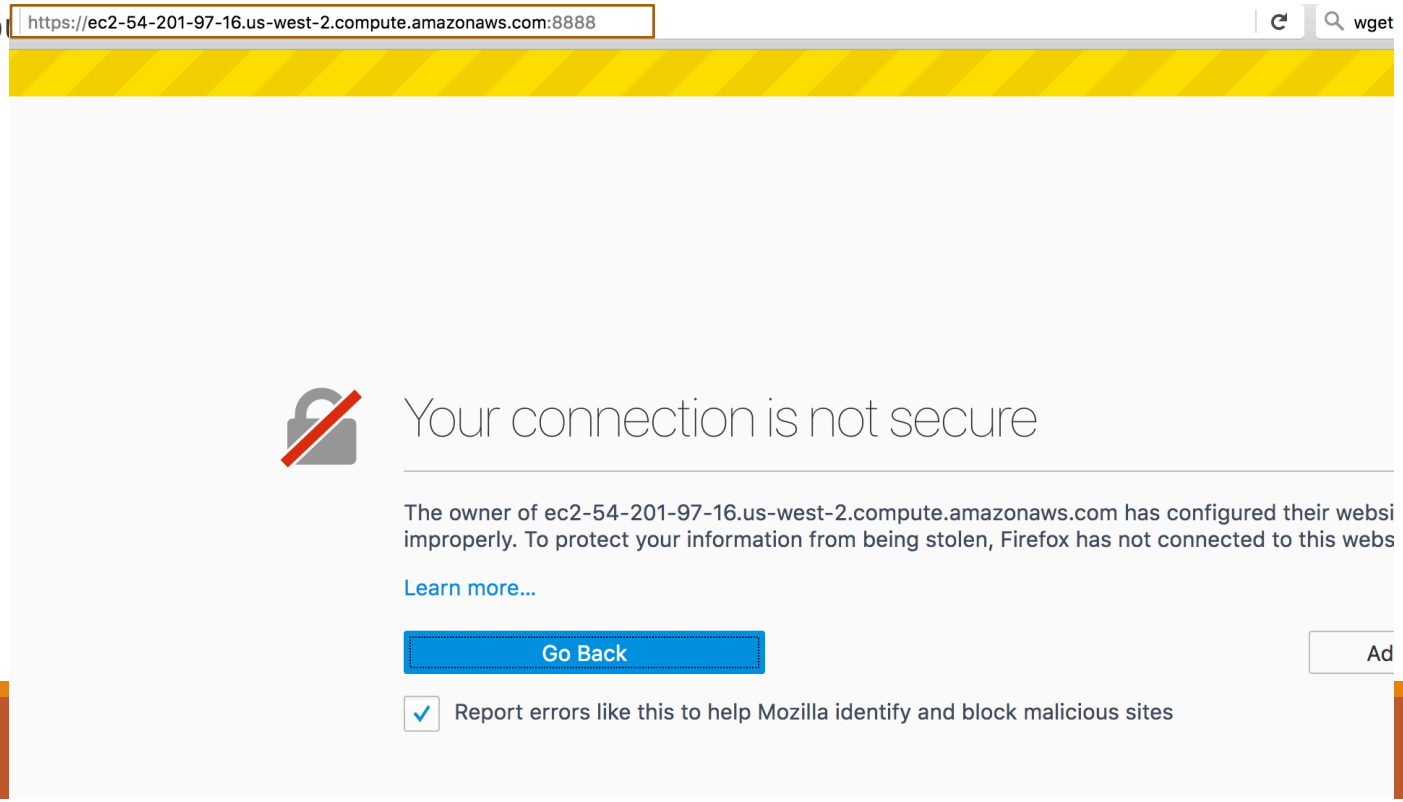
Running on Amazon EMR.

- Check security setting and enable notebook for pyspark.
  - Make sure port 8888 is available.

Running on YARN

Running on Amazon EMR.

- pyspark
- Check jupyter notebook and enable notebook for pyspark.
  - type jupyter notebook (or **pyspark**) and access it by Master public DNS:8888 on your compu



Running on YARN

Run an application to cluster.

- Copy an example code to EMR using scp.

  scp –i *.pem file_name ec2_address:dir_location

```
ML-ITS-603436:~ dwoodbridge$ cd spark_example/
ML-ITS-603436:spark_example dwoodbridge$ scp -i diane_msan694.pem examples/src/main/python/pi.py hadoop@ec2-54-202-143-9
7.us-west-2.compute.amazonaws.com:/mnt/
pi.py                                         100% 1467     1.4KB/s   00:00
ML-ITS-603436:spark_example dwoodbridge$ x
```

```
[hadoop@ip-172-31-32-214 mnt]$ ls
anaconda3       day4_ex4.ipynb  emr_test.py   metastore_db  pi.py  tmp              Untitled2.ipynb  var
day4_ex1.ipynb  derby.log       emr_test.py~  namenode      s3     Untitled1.ipynb  Untitled.ipynb
```

Running on YARN

Run an application to cluster.

- spark-submit code.py --driver-memory 1G

```
[hadoop@ip-172-31-32-214 mnt]$ spark-submit pi.py
17/01/13 20:58:13 INFO SparkContext: Running Spark version 2.0.2
17/01/13 20:58:14 INFO SecurityManager: Changing view acls to: hadoop
17/01/13 20:58:14 INFO SecurityManager: Changing modify acls to: hadoop
17/01/13 20:58:14 INFO SecurityManager: Changing view acls groups to:
17/01/13 20:58:14 INFO SecurityManager: Changing modify acls groups to:
17/01/13 20:58:14 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permi
th modify permissions: Set(hadoop); groups with modify permissions: Set()
17/01/13 20:58:14 INFO Utils: Successfully started service 'sparkDriver' on port 36298.
17/01/13 20:58:14 INFO SparkEnv: Registering MapOutputTracker
17/01/13 20:58:14 INFO SparkEnv: Registering BlockManagerMaster
17/01/13 20:58:14 INFO DiskBlockManager: Created local directory at /mnt/tmp/blockmgr-5c6846cc-3991-40ef-a2ad-0945486ae408
17/01/13 20:58:14 INFO MemoryStore: MemoryStore started with capacity 5.9 GB
17/01/13 20:58:14 INFO SparkEnv: Registering OutputCommitCoordinator
17/01/13 20:58:15 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
17/01/13 20:58:15 INFO Utils: Successfully started service 'SparkUI' on port 4041.
17/01/13 20:58:15 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://172.31.32.214:4041
17/01/13 20:58:15 INFO Utils: Using initial executors = 2, max of spark.dynamicAllocation.initialExecutors, spark.dynamicA
17/01/13 20:58:16 INFO RMProxy: Connecting to ResourceManager at ip-172-31-32-214.us-west-2.compute.internal/172.31.32.214
```