

# 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"?~~

# 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?

# 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

Running on a YARN Cluster

Running on a Mesos Cluster

Loading and Saving Data

# Contents

## **Running on a YARN Cluster**

Running on a Mesos Cluster

Loading and Saving Data

# Running on a YARN Cluster

YARN (Yet Another Resource Negotiator)

- Hadoop's resource manager 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.

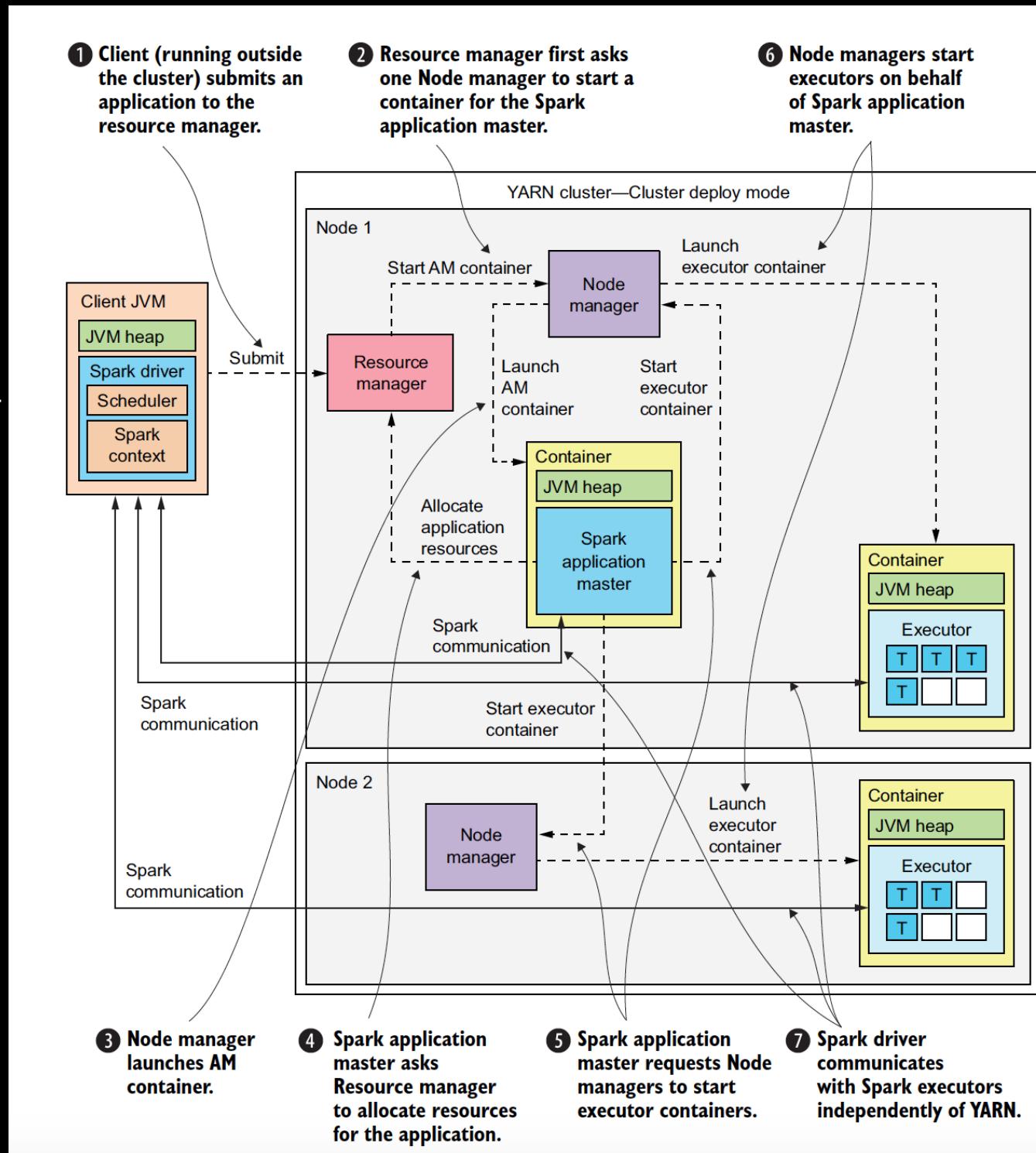
# Running on a YARN Cluster

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 Manager : 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.

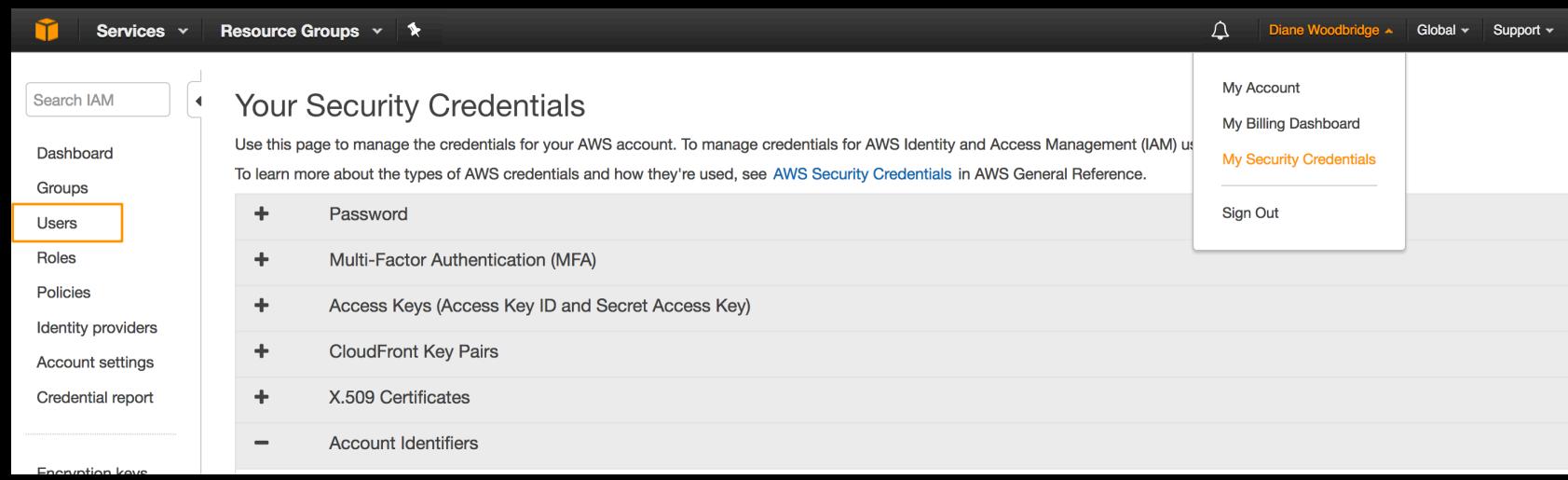
<http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-spark.html>

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

# Running on a YARN Cluster

Running on Amazon EMR.

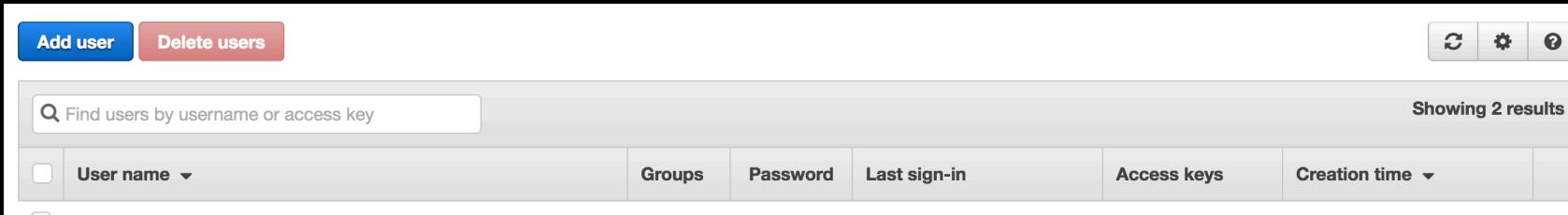
- On your AWS console, go to the user information → My Security Credentials. → Choose Users.



# Running on a YARN Cluster

Running on Amazon EMR.

- Choose Add User.



The screenshot shows the AWS IAM User Management console. At the top, there are two buttons: 'Add user' (blue) and 'Delete users' (red). To the right are three small icons: a refresh symbol, a gear, and a question mark. Below these are two larger icons: a user icon and a key icon. A search bar with the placeholder 'Find users by username or access key' is positioned above a table. The table has a header row with columns: 'User name' (with a dropdown arrow), 'Groups', 'Password', 'Last sign-in', 'Access keys', and 'Creation time' (with a dropdown arrow). The table body contains one row of data. The 'Showing 2 results' message is visible at the top right of the table area.

User name	Groups	Password	Last sign-in	Access keys	Creation time
datainst1	None	None	Never	None	2016-09-20 10:11 PDT

# Running on a YARN Cluster

Running on Amazon EMR.

- Choose a user name and access type.

The screenshot shows the 'Add user' wizard in the AWS IAM console. The title bar says 'Add user'. A progress bar at the top right shows four steps: 1 (Details) is highlighted in blue, while 2 (Permissions), 3 (Review), and 4 (Complete) are in grey. The main section is titled 'Set user details' and contains a note: 'You can add multiple users at once with the same access type and permissions.' Below this is a 'User name\*' field containing 'msan694\_dwoodbridge'. There is also a '+ Add another user' button. The next section, 'Select AWS access type', includes a note: 'Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step.' It shows two options: 'Programmatic access' (checked) and 'AWS Management Console access'. The checked option is described as enabling an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools. The unselected option is described as enabling a password for the AWS Management Console. At the bottom left is a note: '\* Required'. At the bottom right are 'Cancel' and 'Next: Permissions' buttons.

Add user

1 2 3 4

Details Permissions Review Complete

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\* msan694\_dwoodbridge

+ Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type\*  **Programmatic access**  
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

**AWS Management Console access**  
Enables a **password** that allows users to sign-in to the AWS Management Console.

\* Required

Cancel Next: Permissions

# Running on a YARN Cluster

Running on Amazon EMR.

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

The screenshot shows the 'Permissions' section of the AWS IAM console. At the top, there are three options: 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly'. The third option is highlighted with a blue background. Below this, a large table lists available policies. The table has columns for 'Policy name', 'Type', 'Attachments', and 'Description'. A search bar at the top of the table allows filtering by policy type. The table shows 240 results. One policy, 'AmazonEC2FullAccess', is selected, indicated by a checked checkbox in the first column. The 'Description' column for this policy states: 'Provides full access to Amazon EC2 via the AWS Management Console.' Other policies listed include 'AWSHealthFullAccess', 'AmazonRDSFullAccess', 'SupportUser', 'AWSelasticBeanstalkReadOnlyAccess', and 'AWSCertificateManagerReadOnly'.

	Policy name	Type	Attachments	Description
<input type="checkbox"/>	AWSHealthFullAccess	AWS managed	0	Allows full access to the AWS Health APIs and Notifications and the Personal Health ...
<input type="checkbox"/>	AmazonRDSFullAccess	AWS managed	1	Provides full access to Amazon RDS via the AWS Management Console.
<input type="checkbox"/>	SupportUser	Job function	0	This policy grants permissions to troubleshoot and resolve issues in an AWS account...
<input checked="" type="checkbox"/>	AmazonEC2FullAccess	AWS managed	0	Provides full access to Amazon EC2 via the AWS Management Console.
<input type="checkbox"/>	AWSelasticBeanstalkReadOnlyAccess	AWS managed	0	Provides read only access to AWS Elastic Beanstalk via the AWS Management Cons...
<input type="checkbox"/>	AWSCertificateManagerReadOnly	AWS managed	0	Provides read only access to AWS Certificate Manager (ACM).

# Running on a YARN Cluster

Running on Amazon EMR.

- Create User.

The screenshot shows the 'Add user' review step in the AWS IAM console. The top navigation bar includes 'Services', 'Resource Groups', a search bar, and user information for Diane Woodbridge. Below the header, the title 'Add user' is displayed, followed by a four-step progress bar with steps 1 (Details), 2 (Permissions), 3 (Review), and 4 (Complete). Step 3 is highlighted with a blue background. The 'Review' section contains a summary of the user's details and attached policies. The 'User details' section shows a user name of 'msan694\_dwoodbridge' and 'Programmatic access - with an access key' as the AWS access type. The 'Permissions summary' section lists the 'AmazonEC2FullAccess' managed policy. At the bottom right are 'Cancel', 'Previous', and 'Create user' buttons.

Add user

Review

User details

Type	Name
Managed policy	AmazonEC2FullAccess

Permissions

Cancel Previous Create user

# Running on a YARN Cluster

Running on Amazon EMR.

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

Add user

1 2 3 4

Details Permissions Review Complete

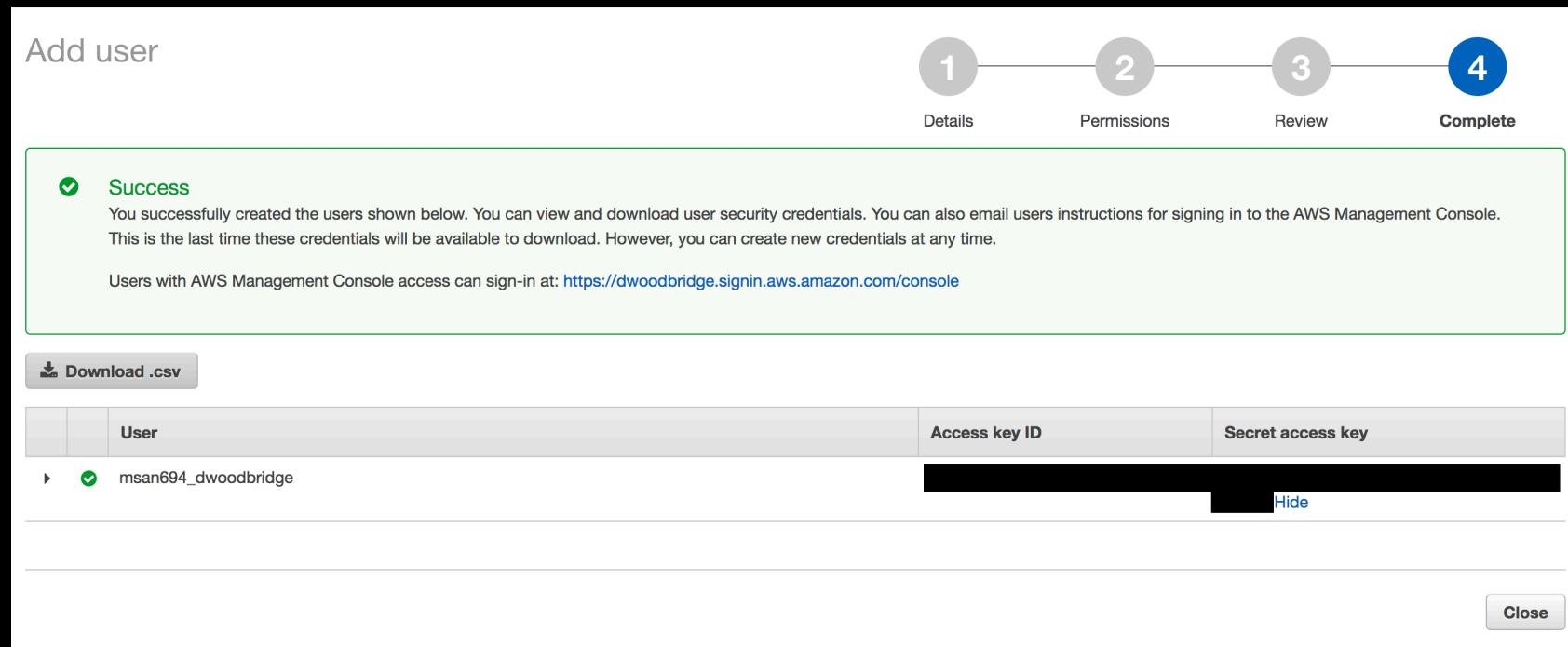
**Success**  
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://dwoodbridge.signin.aws.amazon.com/console>

[Download .csv](#)

User	Access key ID	Secret access key
msan694_dwoodbridge	[REDACTED]	<a href="#">Hide</a>

[Close](#)



# Running on a YARN Cluster

Running on Amazon EMR.

- On your .profile or .bash\_profile store  
AWS\_ACCESS\_KEY\_ID and  
AWS\_SECRET\_ACCESS\_KEY.

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

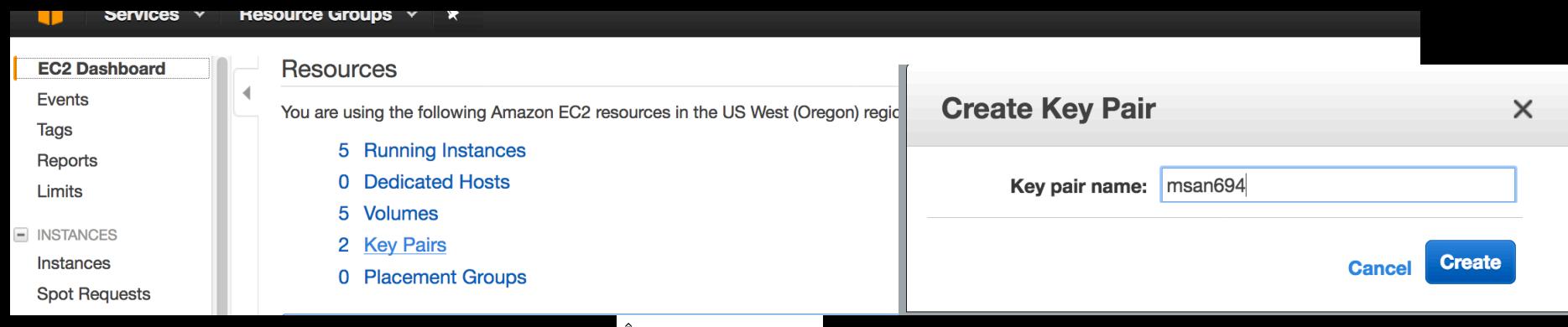
AWS **Access Key ID** and **Secret Access Key**.

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.

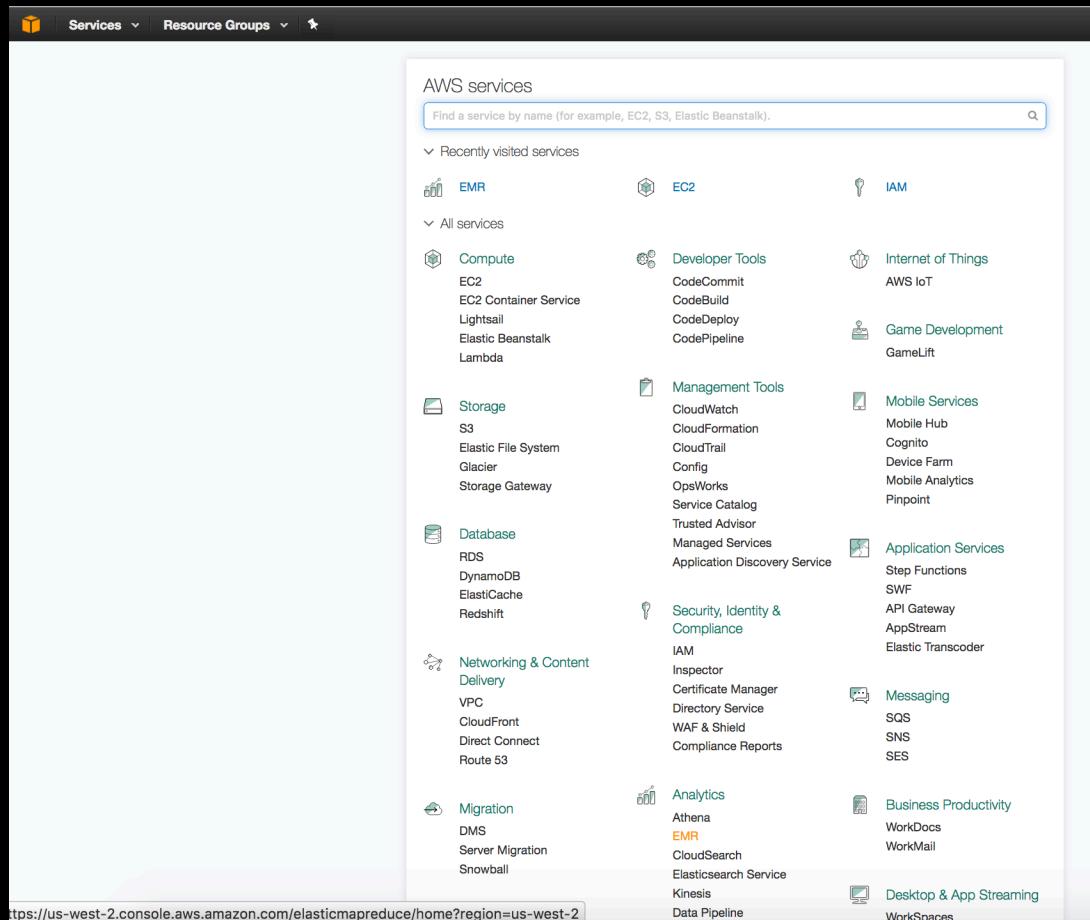
- On EC2 Console  
(<https://console.aws.amazon.com/ec2/>), Choose Key Pairs → Create Key Pairs , download the .pem and store somewhere on your machine\*.
- Make sure that you set the permissions for the private key file to 600 (i.e. only you can read and write it) so that ssh will work.
  - `chmod 600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

The screenshot shows the Amazon EMR service dashboard. At the top, there are navigation links for 'Services' and 'Resource Groups'. On the left, a sidebar menu includes 'Amazon EMR' (selected), 'Cluster list', 'Security configurations', 'VPC subnets', and 'Help'. The main content area is titled 'Welcome to Amazon Elastic MapReduce'. It explains that Amazon EMR is a web service for processing large amounts of data. Below this, a message says 'You do not appear to have any clusters. Create one now:' followed by a blue 'Create cluster' button. A section titled 'How Elastic MapReduce Works' is divided into three steps: 'Upload' (cloud icon with an orange arrow pointing up), 'Create' (cluster icon with a gear and an orange arrow pointing right), and 'Monitor' (monitor icon with a graph and an orange arrow pointing down). Each step has a corresponding description and a 'Learn more' link.

Welcome to Amazon Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR) is a web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data.

You do not appear to have any clusters. Create one now:

Create cluster

How Elastic MapReduce Works

Upload

Create

Monitor

Upload your data and processing application to S3.

Configure and create your cluster by specifying data inputs, outputs, cluster size, security settings, etc.

Monitor the health and progress of your cluster. Retrieve the output in S3.

[Learn more](#)

[Learn more](#)

[Learn more](#)

# Running on a YARN Cluster

Running on Amazon EMR.

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

General Configuration

Cluster name: msan694  
 Logging  
S3 folder: s3://aws-logs-374226464461-us-west-2/elastict  
Launch mode: Cluster  Step execution

Software configuration

Release: emr-5.10.0  
 Core Hadoop: Hadoop 2.7.3 with Ganglia 3.7.2, Hive 2.3.1, Hue 4.0.1, Mahout 0.13.0, Pig 0.17.0, and Tez 0.8.4  
 HBase: HBase 1.3.1 with Ganglia 3.7.2, Hadoop 2.7.3, Hive 2.3.1, Hue 4.0.1, Phoenix 4.11.0, and ZooKeeper 3.4.10  
 Presto: Presto 0.187 with Hadoop 2.7.3 HDFS and Hive 2.3.1 Metastore  
 Spark: Spark 2.2.0 on Hadoop 2.7.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.3  
 Use AWS Glue Data Catalog for table metadata

Hardware configuration

Instance type: m3.xlarge  
Number of instances: 3 (1 master and 2 core nodes)

Security and access

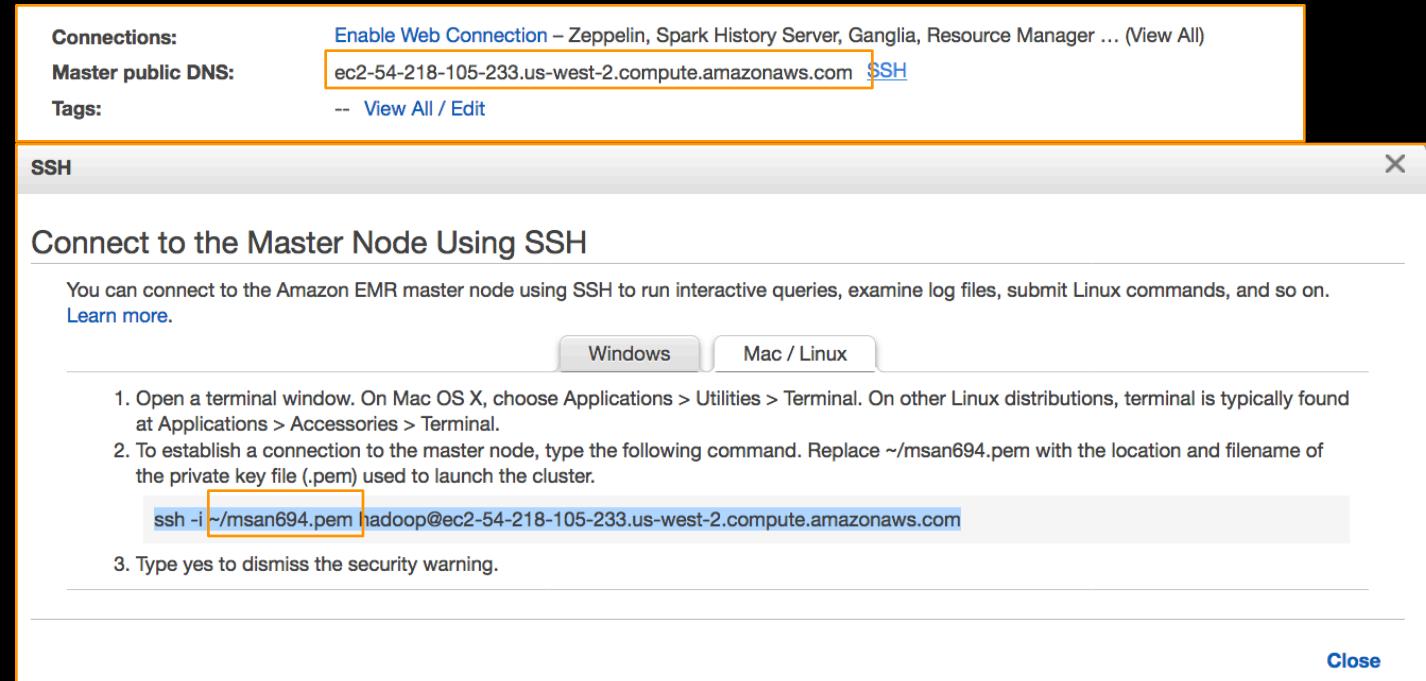
EC2 key pair: msan694   
Permissions: Default  Custom  
Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.  
EMR role: EMR\_DefaultRole   
EC2 instance profile: EMR\_EC2\_DefaultRole

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:lvuUviA9QfdsQoyEngU4aEQ8eqyN0a21uv1g2BWK16I.
Are you sure you want to continue connecting (yes/no)? yes]
```

# 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:lvuUviA9QfdsQoyEngU4aEQ8eqyN0a21uv1g2BWK16I.
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/

EEEEEEEEEEEEEEEEEE MMMMMMMM      MMMMMMM RRRRRRRRRRRRRR
E:::::::E:::::E M:::::::M      M:::::::M R:::::::R:::::R
EE:::::E:::::E 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:::::E:::::E E M::::::M M:::::M::::::M M::::::M R:::::RRRRRR:::R
  E:::::E:::::E E M::::::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:::::E:::::E E M::::::M      M::::::M R:::::R    R:::::R
E:::::::E:::::E E M::::::M      M::::::M RR:::::R    R:::::R
EEEEEEEEEEEEEEEEEE MMMMMMM      MMMMMMM RRRRRRRR      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

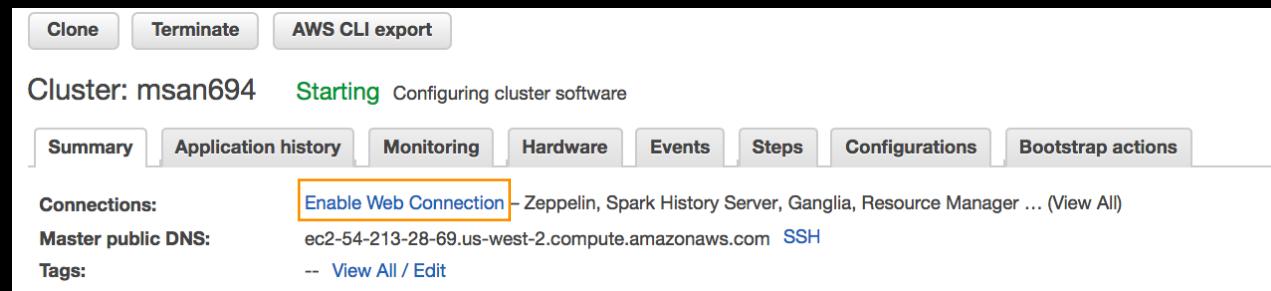
   /   /_ \
  / \  _ \ - \
 /_ / .__\_,/_/ /_/\_\_/
 /_/
version 2.0.2

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.



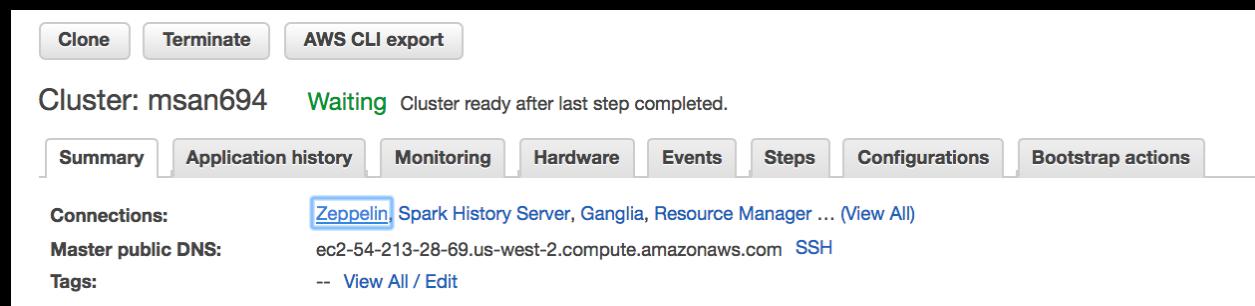
- You can also install Jupyter notebook (The script is in the Example repository.)

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

# 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.



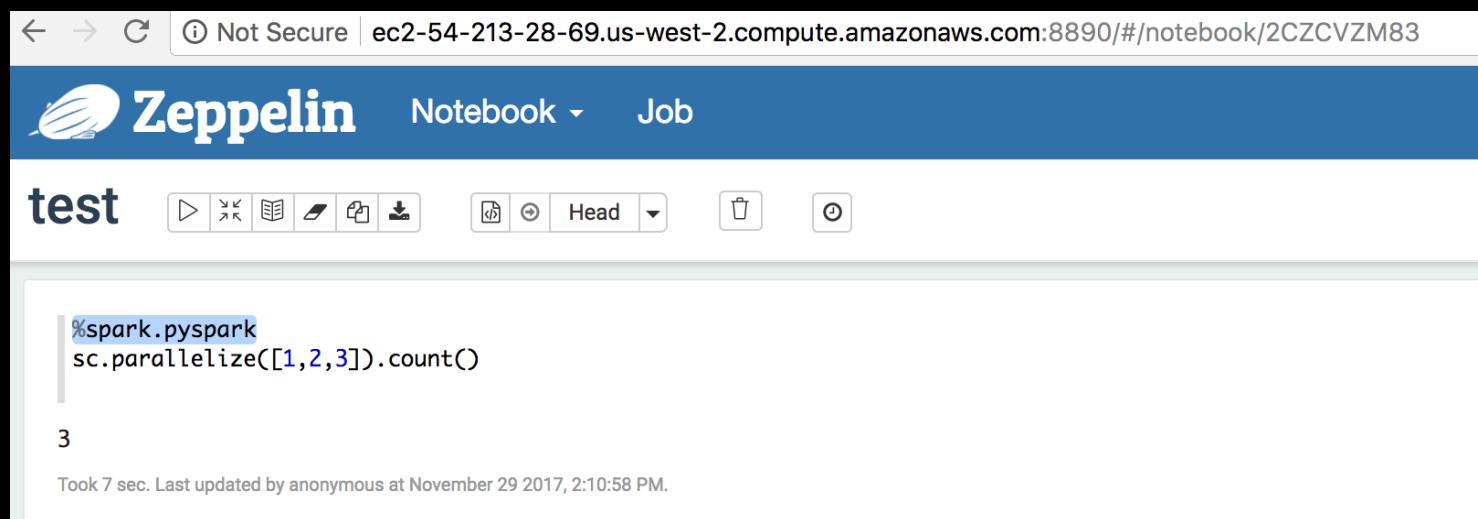
- You can also install Jupyter notebook (The script is in the Example repository.)

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

# 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.



The screenshot shows a browser window for a Zeppelin notebook. The URL is `ec2-54-213-28-69.us-west-2.compute.amazonaws.com:8890/#/notebook/2CZCVZM83`. The page title is "Zeppelin". The top navigation bar includes "Notebook" and "Job". Below the header, there's a toolbar with icons for back, forward, search, and file operations. The main content area contains a code cell labeled "test". The code in the cell is:

```
%spark.pyspark
sc.parallelize([1,2,3]).count()
```

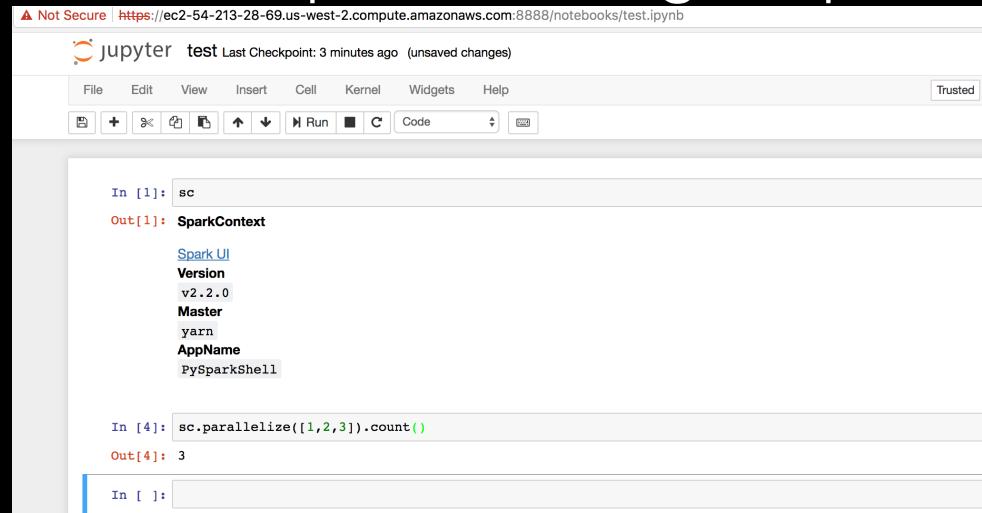
The output of the code is "3". At the bottom of the page, a timestamp indicates it was "Took 7 sec. Last updated by anonymous at November 29 2017, 2:10:58 PM."

<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.)



A screenshot of a Jupyter Notebook interface. The title bar says "Not Secure | https://ec2-54-213-28-69.us-west-2.compute.amazonaws.com:8888/notebooks/test.ipynb". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar below has icons for New, Open, Save, Run, Cell, Kernel, Help, and Code. A "Trusted" button is on the right. The main area shows two code cells:

```
In [1]: sc
Out[1]: SparkContext
          Spark UI
          Version
          v2.2.0
          Master
          yarn
         AppName
          PySparkShell
```

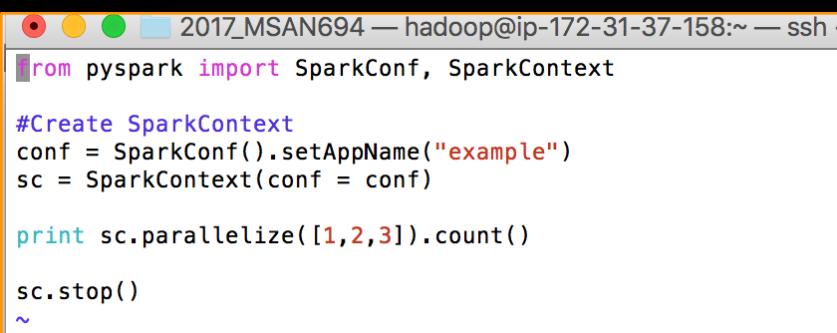
```
In [4]: sc.parallelize([1,2,3]).count()
Out[4]: 3
```

The interface is light gray with dark gray toolbars and a white main content area.

# Running on a YARN Cluster

Running on Amazon EMR.

- Run an application to cluster.
  - spark-submit code\_name.py



```
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 disabled
```

<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

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...

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

# Running on a YARN Cluster

YARN UI (Resource Manager, Port 8088)

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

The screenshot shows the YARN Resource Manager UI at port 8088. At the top, there is a navigation bar with tabs: Summary (selected), Application history, Monitoring, Hardware, Events, Steps, and Configurations. Below the tabs, there are three sections: 'Connections' listing Zeppelin, Spark History Server, Ganglia, and Resource Manager (with a 'View All' link); 'Master public DNS' showing ec2-34-210-83-121.us-west-2.compute.amazonaws.com with SSH; and 'Tags' with a 'View All / Edit' link.



## All Applications

The screenshot shows the Hadoop YARN Application Overview page. On the left, there is a sidebar with a tree view of the cluster structure and a 'Scheduler' section. The main area has two tabs: 'Cluster Metrics' and 'Scheduler Metrics'. The 'Cluster Metrics' tab displays various resource statistics. The 'Scheduler Metrics' tab shows the scheduler type (Capacity Scheduler), scheduling resource type ([MEMORY]), and minimum/maximum allocation details (<memory:32, vCores:1> and <memory:12288, vCores:16>). Below these tabs is a table for 'All Applications'. The table includes columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, and Tracking UI. One application entry is shown: application\_1512070821378\_0001, run by hadoop, using PySparkShell, SPARK application type, default queue, started on Thu Nov 30 14:02:48 -0800 2017, still RUNNING with undefined final status, 0% progress, and no tracking UI. At the bottom, it says 'Showing 1 to 1 of 1 entries' and has links for 'First' and 'Previous'.

# Contents

Running on a YARN Cluster

**Running on a Mesos Cluster**

Loading and Saving Data

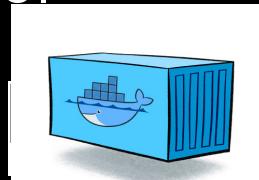
# Running on a Mesos Cluster

## 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

Running on a YARN Cluster

Running on a Mesos Cluster

**Loading and Saving Data**

# 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(lambda x: json.loads(x))
```

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

```
json_output = data.map(lambda 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()
```

# References

Distributed Computing with Spark, Reza Zadeh,  
[http://stanford.edu/~rezab/slides/bayacm\\_spark.pdf](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

---



UNIVERSITY OF SAN FRANCISCO  
CHANGE THE WORLD FROM HERE

# 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' # Notebook config this is where you saved
c.NotebookApp.ip = '*' # Run on all IP addresses of your instance your pem cert
c.NotebookApp.open_browser = False # Don't open browser by default
c.NotebookApp.port = 8888 # Use port 8888 for notebook
```



# 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.

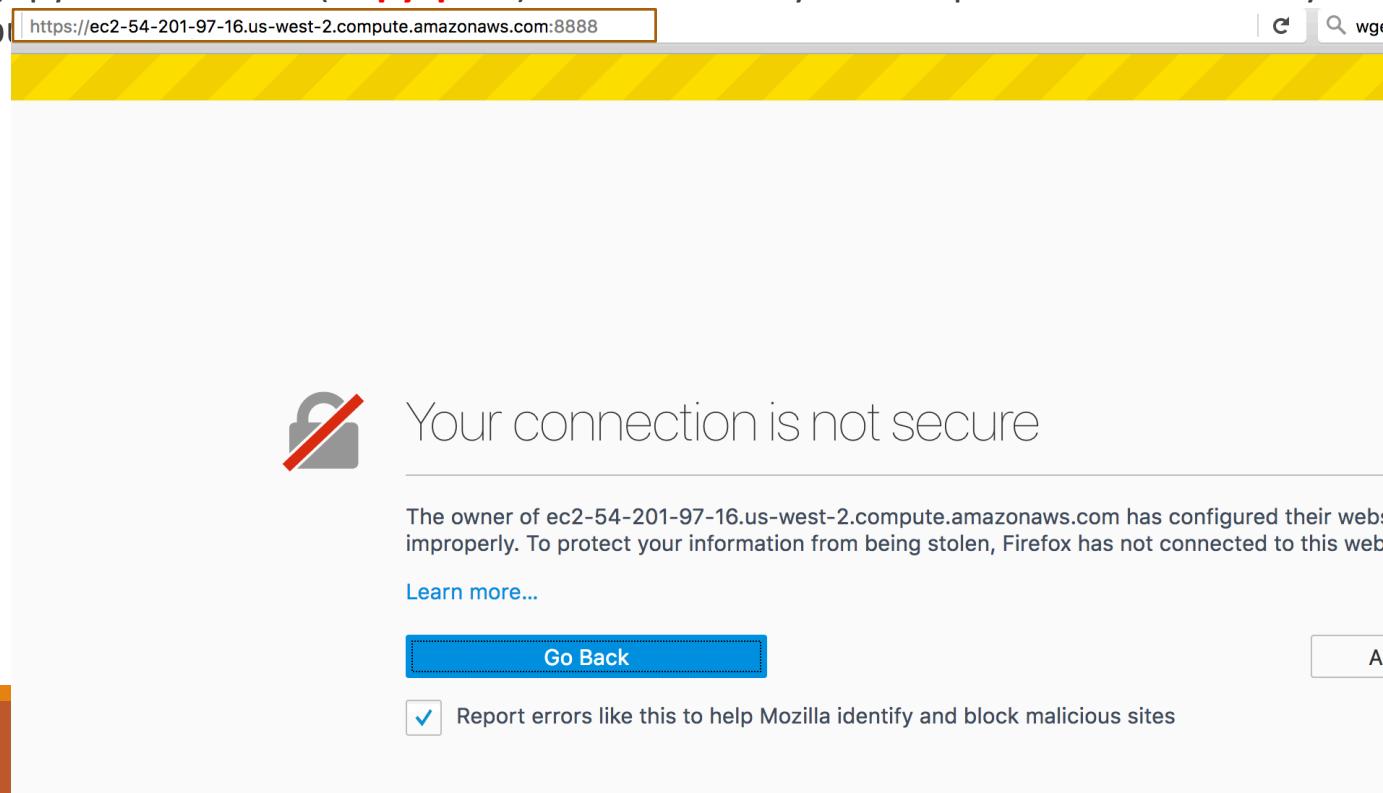


# Running on YARN

---

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 computer



# 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      pi~  Untitled1.ipynb  Untitled.ipynb
r.....
```



# 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 permission 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.dynamicAllocation.maxExecutors
17/01/13 20:58:16 INFO RMProxy: Connecting to ResourceManager at ip-172-31-32-214.us-west-2.compute.internal/172.31.32.214:8032
17/01/13 20:58:16 INFO Client: Requesting a new application from cluster with 2 NodeManagers
```

