Exercise 4

Spark SQL on AWS

Prior Knowledge
Unix Command Line Shell
Simple Python

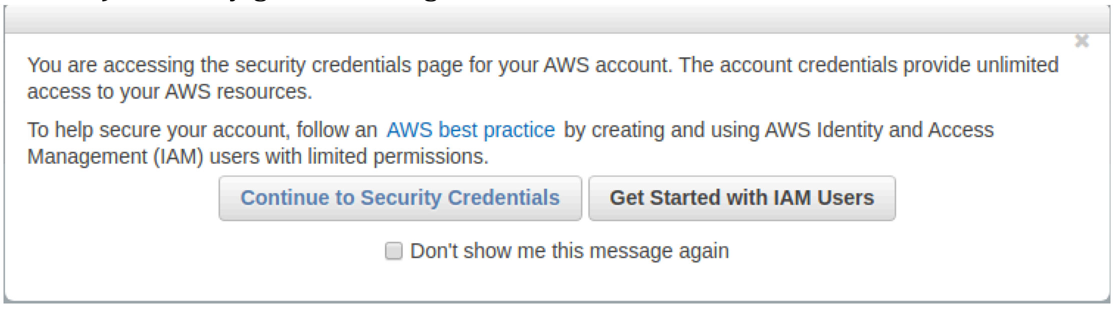
Learning Objectives
Understanding how to run Spark on AWS using Flintrock

Software Requirements
(see separate document for installation of these)

- EC2 credentials
- Flintrock

Part A. Starting Spark in EC2

1. Log into your AWS account in a browser, and go to **My Security Credentials** (in a dropdown menu from your name in the top right end corner). You may get a warning like this:



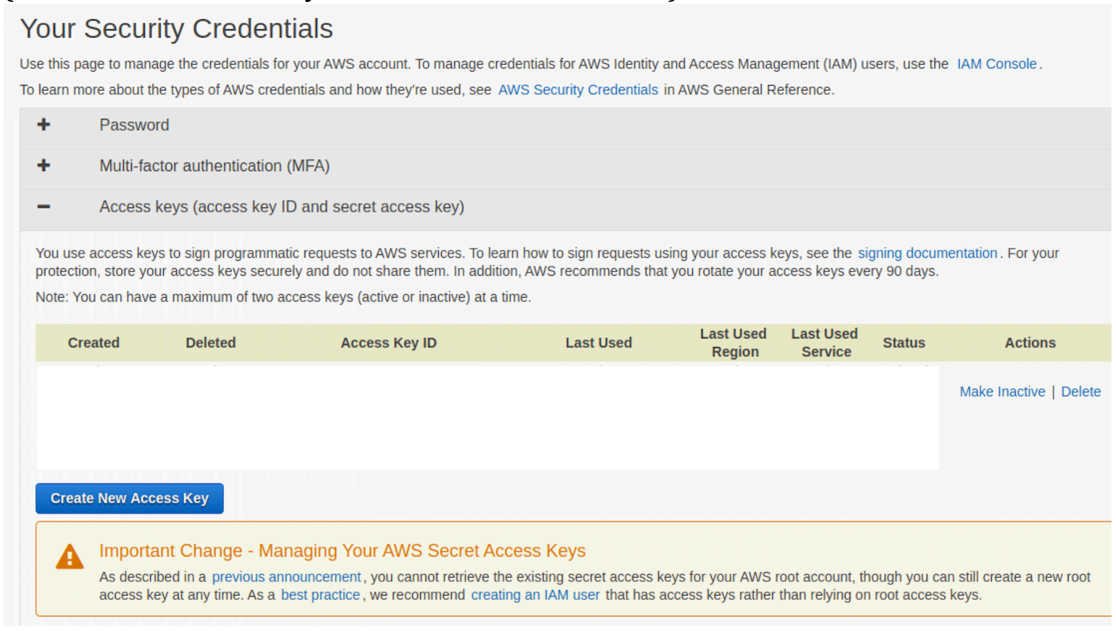
You are accessing the security credentials page for your AWS account. The account credentials provide unlimited access to your AWS resources.

To help secure your account, follow an AWS best practice by creating and using AWS Identity and Access Management (IAM) users with limited permissions.

Continue to Security Credentials
Get Started with IAM Users
Don't show me this message again

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

2. Expand the section labeled **Access Keys**
   (I have blanked out key details in the screen shot)



Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console.

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

+ Password
+ Multi-factor authentication (MFA)
− Access keys (access key ID and secret access key)

You use access keys to sign programmatic requests to AWS services. To learn how to sign requests using your access keys, see the signing documentation. For your protection, store your access keys securely and do not share them. In addition, AWS recommends that you rotate your access keys every 90 days.

Note: You can have a maximum of two access keys (active or inactive) at a time.

Created	Deleted	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Make Inactive | Delete

Important Change - Managing Your AWS Secret Access Keys

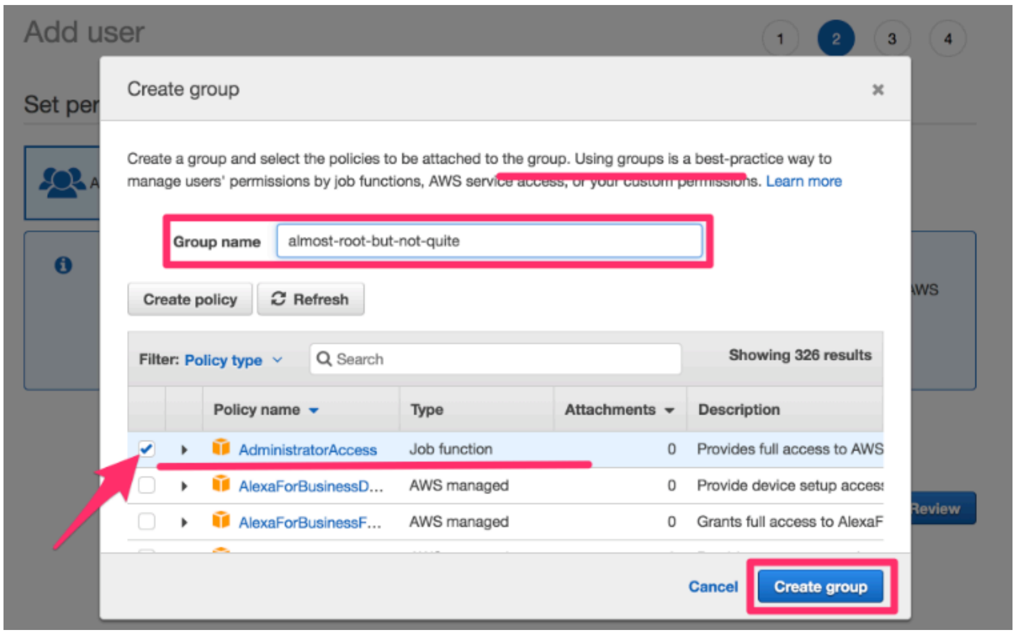
As described in a previous announcement, you cannot retrieve the existing secret access keys for your AWS root account, though you can still create a new root access key at any time. As a best practice, we recommend creating an IAM user that has access keys rather than relying on root access keys.

3. Its now considered best practice to create an IAM (Identity and Authorisation Management) user with its own keys for each application rather than relying on root access keys. You can find out more about this by clicking on "creating an IAM user" (see screenshot above). Basically, you need to go to the "all services" list in the AWS console, selecting **IAM**, selecting Users and then selecting **Add user.**

4. On the page that appears
   a. Provide a username (e.g., sparkuser)
   b. Check the box next to both access types: "*Programmatic access*" as well as "*AWS Management Console access*"
   c. Click the **Next:Permissions** button

5. Select "**Add user to group**" and click "**Create group**". Create a group with the following policy selected:

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/



Create group

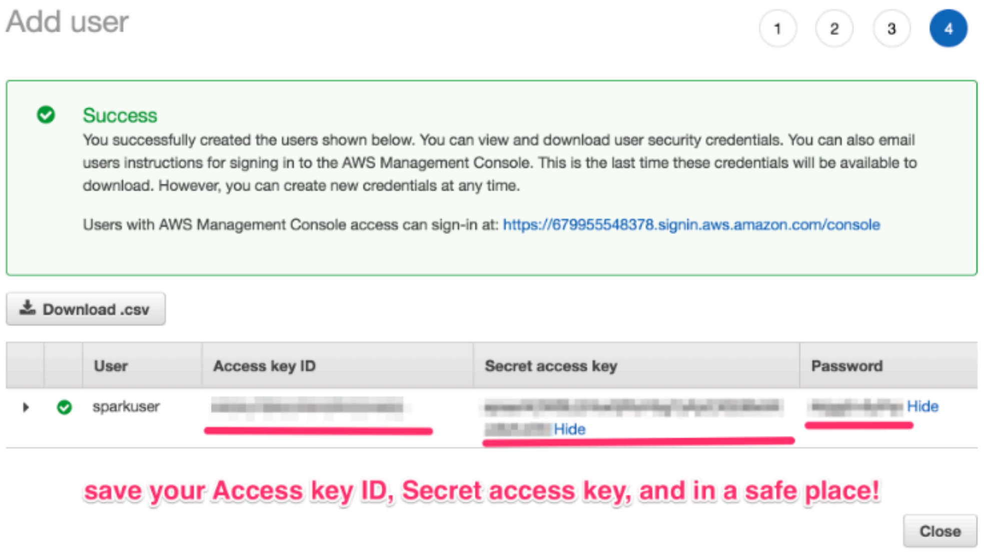
Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Learn more

Group name: almost-root-but-not-quite

Showing 326 results

		Policy name	Type	Attachments	Description
✓	▸	AdministratorAccess	Job function	0	Provides full access to AWS
	▸	AlexaForBusinessD...	AWS managed	0	Provide device setup acces
	▸	AlexaForBusinessF...	AWS managed	0	Grants full access to AlexaF

6. Add your user to this newly created group and click the **Next: Review** button. Having checked that the details look correct, click **Create user**.



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://679955548378.signin.aws.amazon.com/console

		User	Access key ID	Secret access key	Password
▸	✓	sparkuser	[redacted]	[redacted] Hide	[redacted] Hide

save your Access key ID, Secret access key, and in a safe place!

7. **Download the key file**

8. Move or copy the `credentials.csv` file out of your Downloads directory into a safe place where you will be able to find it (e.g., a directory called `keys`). Display the keyfile (e.g. use `cat` or `less`)

9. On a command line type:
```
aws configure
```
You should see:
```
AWS Access Key ID [****************J3EA]:
```

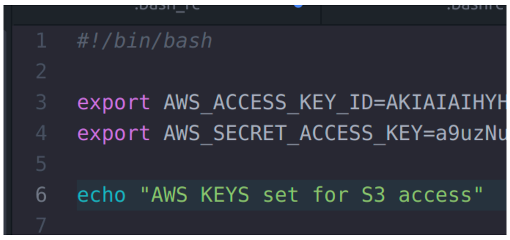
10. Copy the Access Key ID from the keyfile, and then hit Enter

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

11. Do the same for the Secret Access Key
12. Set the default region to **eu-west-2**
13. Set the output to **json**
14. It should look something like this (but with your keys):

```
AWS Access Key ID [****************J3EA]: AKIASF22343434UNM33UVIA
Secret Access Key [****************JXb7]: 8z1rtTbU3Ur/llksdafkjhd398u34msndHnGaDY
Default region name [eu-west-1]: eu-west-2
Default output format [json]: json
```

15. Some applications also need these keys to be in your environment variables. Use an editor (e.g. Atom) to create a bash script called awskeys as follows



```
#!/bin/bash

export AWS_ACCESS_KEY_ID=AKIAIAIHYH
export AWS_SECRET_ACCESS_KEY=a9uzNu

echo "AWS KEYS set for S3 access"
```

Make this script executable and run it by typing at the command line:

```
$chmod u+x awskeys
$source ./awskeys
```

16. In addition to these "Access Keys", we also need an SSH key to continue. If you successfully completed the pre-course Amazon lab, you should have a file `~/keys/bigkp.pem`. If not, look at the pre-course instructions or grab one of the instructors to help you create one.

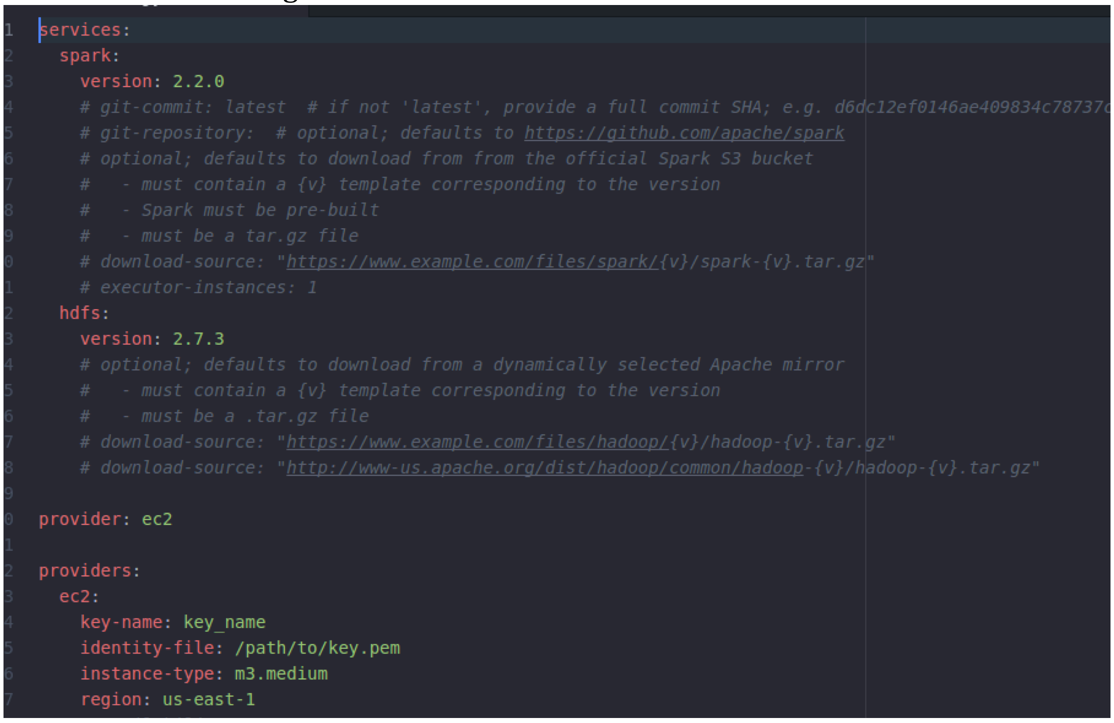
17. There is a project from the creators of Spark to run it in EC2, but it is not very good! Instead we will use a tool called **flintrock,** which can configure Spark clusters in AWS for you.

18. Before we can use flintrock, you need to modify the config file for flintrock so that it uses your own keys. Edit the flintrock config file:

```
atom ~/.config/flintrock/config.yaml
```

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

It will look something like:



```
services:
  spark:
    version: 2.2.0
    # git-commit: latest  # if not 'latest', provide a full commit SHA; e.g. d6dc12ef0146ae409834c78737c
    # git-repository:  # optional; defaults to https://github.com/apache/spark
    # optional; defaults to download from from the official Spark S3 bucket
    #   - must contain a {v} template corresponding to the version
    #   - Spark must be pre-built
    #   - must be a tar.gz file
    # download-source: "https://www.example.com/files/spark/{v}/spark-{v}.tar.gz"
    # executor-instances: 1
  hdfs:
    version: 2.7.3
    # optional; defaults to download from a dynamically selected Apache mirror
    #   - must contain a {v} template corresponding to the version
    #   - must be a .tar.gz file
    # download-source: "https://www.example.com/files/hadoop/{v}/hadoop-{v}.tar.gz"
    # download-source: "http://www-us.apache.org/dist/hadoop/common/hadoop-{v}/hadoop-{v}.tar.gz"

provider: ec2

providers:
  ec2:
    key-name: key_name
    identity-file: /path/to/key.pem
    instance-type: m3.medium
    region: us-east-1
```

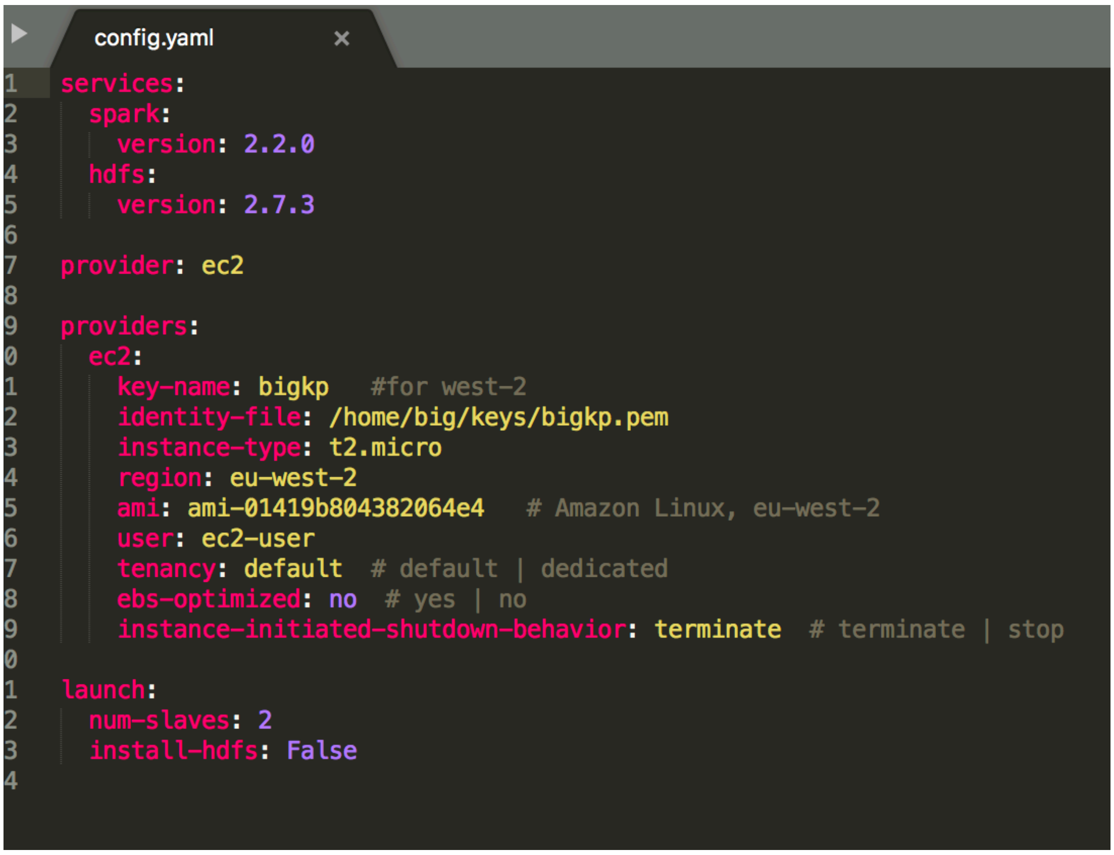
We need to replace the contents of this file, with one that will work for us.

The source for this is in the github repository

~/BigData/code_jw/flintrock-config/config.yaml

It should look like:

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/



```yaml
services:
  spark:
    version: 2.2.0
  hdfs:
    version: 2.7.3

provider: ec2

providers:
  ec2:
    key-name: bigkp   #for west-2
    identity-file: /home/big/keys/bigkp.pem
    instance-type: t2.micro
    region: eu-west-2
    ami: ami-01419b804382064e4   # Amazon Linux, eu-west-2
    user: ec2-user
    tenancy: default  # default | dedicated
    ebs-optimized: no  # yes | no
    instance-initiated-shutdown-behavior: terminate  # terminate | stop

launch:
  num-slaves: 2
  install-hdfs: False
```

This is modified in a couple of ways. Firstly, it gives the London region and AMI (Amazon Machine Image) that is available in that region. This do change from time to time, so it is worth checking what instances are available from the AWS console. The instance-type should also match the type given by for the AMI in the console.

Finally, I've changed the key name and identity file to match your key name and identity file.

19. Replace the file in ~/.config/flintrock with the one from the github repository
20. Update it if necessary by typing
    `flintrock configure`

21. You should now be able to launch a cluster in Amazon. From a new terminal window

    ```
    flintrock launch big
    ```

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

22. Now you should see something like (except with more lines):

```
 Launching 3 instances...
 [54.154.17.100] SSH online.
 [54.154.17.100] Configuring ephemeral storage...
 [54.154.17.100] Installing Java 1.8...
 [34.253.201.139] SSH online.
 [34.253.201.139] Configuring ephemeral storage...
 [34.253.201.139] Installing Java 1.8...
 [54.154.17.100] Installing Spark...
 [34.253.201.139] Installing Spark...
 [34.253.201.139] Configuring Spark master...
 Spark Health Report:
   * Master: ALIVE
   * Workers: 1
   * Cores: 1
   * Memory: 2.7 GB
 launch finished in 0:03:49.
```

23. In the meantime, you could start yet another terminal window and prepare your code to run on AWS.

    ```
    cd sql
    cp wind.py wind-s3.py
    ```

24. Change the URL so that instead of loading the data from the local filesystem, it reaches out to S3 to do it:

    ```
    atom wind-s3.py
    ```

    Instead of reading from '/home/big/sql/*.csv' change it to read from:

    ```
    's3a://discnet-big/wind2015/*'
    ```

    This is a public S3 bucket I have created to store the data. Alternatively you can upload the data to your own S3 bucket using the aws client as follows:

    ```
    $cd ~/BigData/datafiles/wind/2015
    $aws s3 mb s3://my_bucket_name
    $aws s3 cp . s3://my_bucket_name --recursive
    ```

    You then just need to read from '`s3a://my_bucket_name`'
    Note that bucket names on S3 need to be globally unique so you will have to call it something more distinctive than my_bucket_name!

25. In wind-s3.py, delete or comment the first two lines (import findspark and findspark.init()).

26. Save the file

27. Once the launch of your cluster has finished, we need to copy the code into the cluster:

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

```
flintrock copy-file big ~/sql/wind-s3.py /home/ec2-user/
```

28. We also need to copy the AWS access keys to the cluster so that the cluster can access S3 storage

```
flintrock copy-file big ~/awskeys /home/ec2-user/
```

29. Let's login to the master (all one line):

```
flintrock login big
```

You see something like:

```
Warning: Permanently added '34.253.201.139' (ECDSA) to the list
of known hosts.
Last login: Mon Jul 10 18:55:35 2017 from host109-156-251-
208.range109-156.btcentralplus.com

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

https://aws.amazon.com/amazon-linux-ami/2017.03-release-notes/
1 package(s) needed for security, out of 1 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-6-32 ~]$
```

This basically has just SSH'ed you into the master.

30. Now set your awskeys

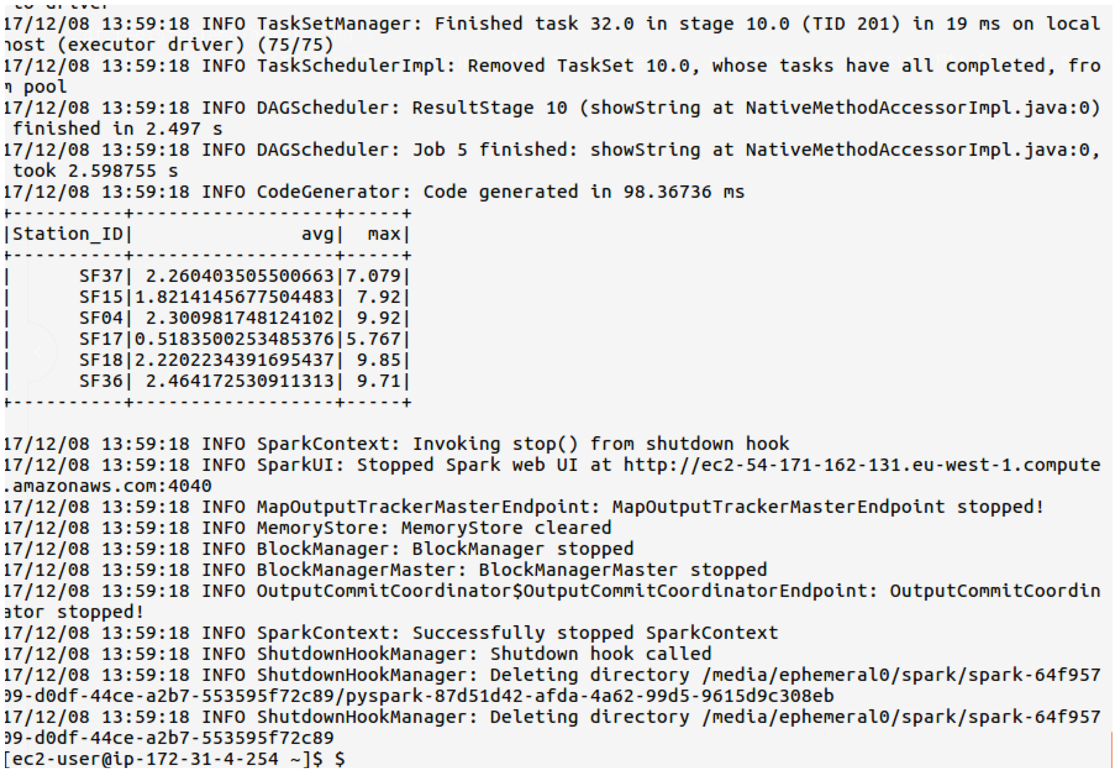
```
$chmod u+x awskeys
$source ./awskeys
```

31. Now launch your code:

```
~/spark/bin/spark-submit wind-s3.py
```

32. You should see a lot of logging, eventually ending with output like in the screenshot below. In this case, it actually took us longer to run on the cluster than on our local machines. However, you will note that this would speed up for bigger problems where the parallelization would add benefits

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/



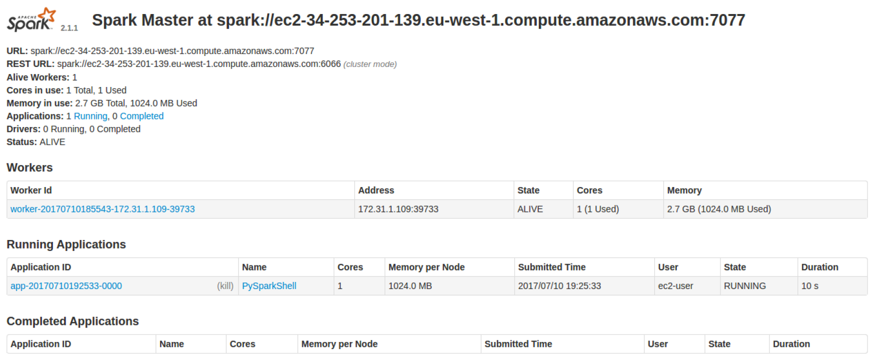
```
17/12/08 13:59:18 INFO TaskSetManager: Finished task 32.0 in stage 10.0 (TID 201) in 19 ms on local
host (executor driver) (75/75)
17/12/08 13:59:18 INFO TaskSchedulerImpl: Removed TaskSet 10.0, whose tasks have all completed, fro
m pool
17/12/08 13:59:18 INFO DAGScheduler: ResultStage 10 (showString at NativeMethodAccessorImpl.java:0)
 finished in 2.497 s
17/12/08 13:59:18 INFO DAGScheduler: Job 5 finished: showString at NativeMethodAccessorImpl.java:0,
 took 2.598755 s
17/12/08 13:59:18 INFO CodeGenerator: Code generated in 98.36736 ms
+----------+------------------+-----+
|Station_ID|               avg|  max|
+----------+------------------+-----+
|      SF37| 2.260403505500663|7.079|
|      SF15|1.8214145677504483| 7.92|
|      SF04| 2.300981748124102| 9.92|
|      SF17|0.5183500253485376|5.767|
|      SF18|2.2202234391695437| 9.85|
|      SF36| 2.464172530911313| 9.71|
+----------+------------------+-----+

17/12/08 13:59:18 INFO SparkContext: Invoking stop() from shutdown hook
17/12/08 13:59:18 INFO SparkUI: Stopped Spark web UI at http://ec2-54-171-162-131.eu-west-1.compute
.amazonaws.com:4040
17/12/08 13:59:18 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
17/12/08 13:59:18 INFO MemoryStore: MemoryStore cleared
17/12/08 13:59:18 INFO BlockManager: BlockManager stopped
17/12/08 13:59:18 INFO BlockManagerMaster: BlockManagerMaster stopped
17/12/08 13:59:18 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordin
ator stopped!
17/12/08 13:59:18 INFO SparkContext: Successfully stopped SparkContext
17/12/08 13:59:18 INFO ShutdownHookManager: Shutdown hook called
17/12/08 13:59:18 INFO ShutdownHookManager: Deleting directory /media/ephemeral0/spark/spark-64f957
99-d0df-44ce-a2b7-553595f72c89/pyspark-87d51d42-afda-4a62-99d5-9615d9c308eb
17/12/08 13:59:18 INFO ShutdownHookManager: Deleting directory /media/ephemeral0/spark/spark-64f957
99-d0df-44ce-a2b7-553595f72c89
[ec2-user@ip-172-31-4-254 ~]$ $
```

33. It is perfectly possible to get Jupyter to talk to Spark on our cluster, but it is slightly complex, so we will just use the normal Python command-line for the moment.

34. Find the IP address of the Spark Master. There are two ways. Firstly, it showed up in the console when you first launched the flintrock cluster:
    `[34.253.201.139] Configuring Spark master...`

    Alternatively, you can find it by typing
    `flintrock describe big`

35. Go to http://xx.xx.xx.xx:8080 using the master's IP address. You should see something like:



Spark 2.1.1 **Spark Master at spark://ec2-34-253-201-139.eu-west-1.compute.amazonaws.com:7077**

URL: spark://ec2-34-253-201-139.eu-west-1.compute.amazonaws.com:7077
REST URL: spark://ec2-34-253-201-139.eu-west-1.compute.amazonaws.com:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 1 Total, 1 Used
Memory in use: 2.7 GB Total, 1024.0 MB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20170710185543-172.31.1.109-39733	172.31.1.109:39733	ALIVE	1 (1 Used)	2.7 GB (1024.0 MB Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20170710192533-0000 (kill)	PySparkShell	1	1024.0 MB	2017/07/10 19:25:33	ec2-user	RUNNING	10 s

Completed Applications

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

36. You can explore your Spark cluster here.

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

37. Exit the SSH session:
    ```
    exit
    ```

38. We must remember to stop our cluster as well (its costing money…)
    From Ubuntu terminal where you started the Spark cluster

    ```
    flintrock destroy big
    ```
    Type y when prompted.

39. Congratulations, this lab is complete.

© Julie Weeds 2019 (adapted from Paul Fremantle 2015). Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/