**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

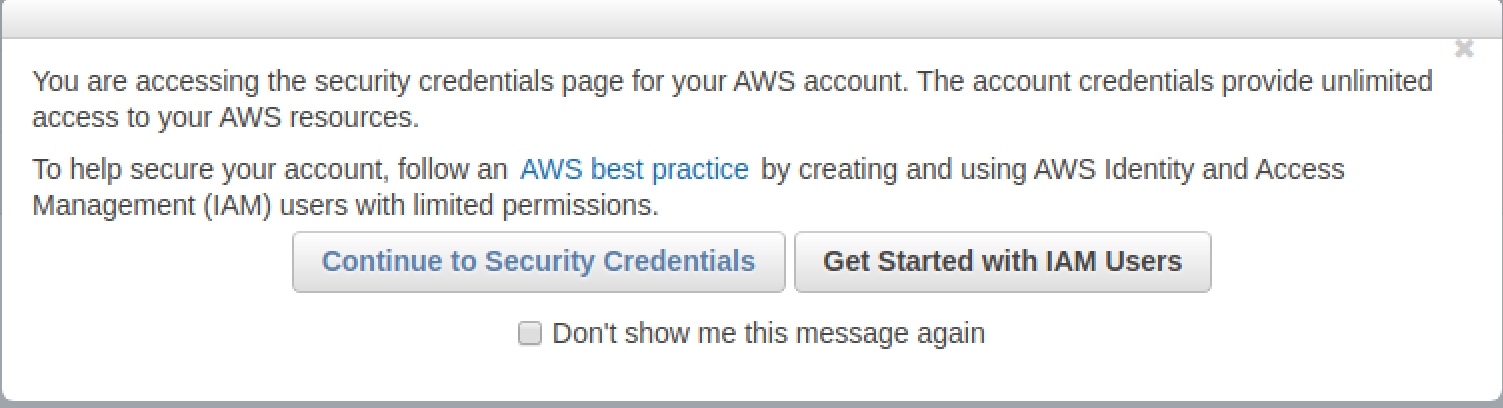
**Software Requirements**

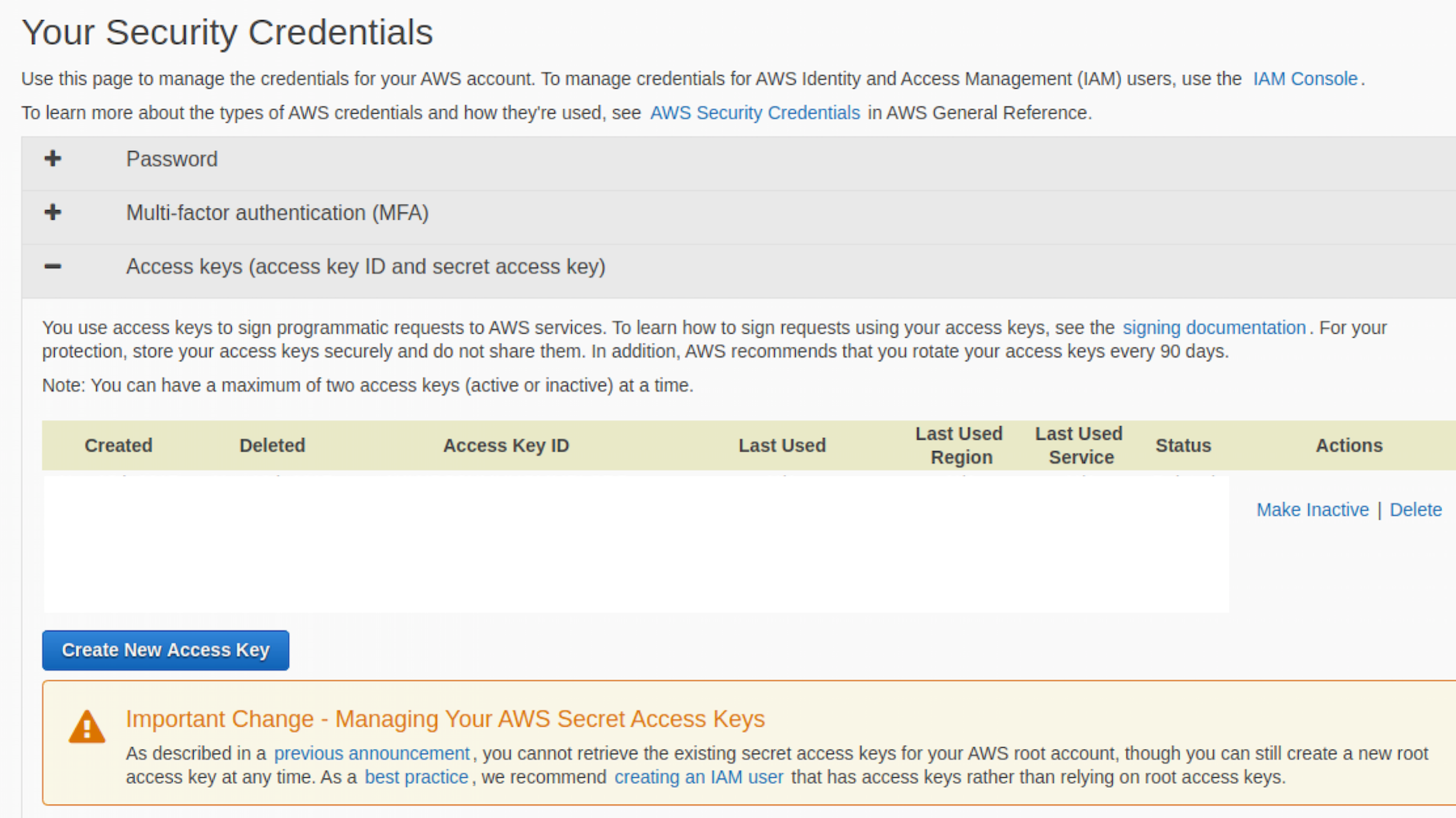
(see separate document for installation of these)

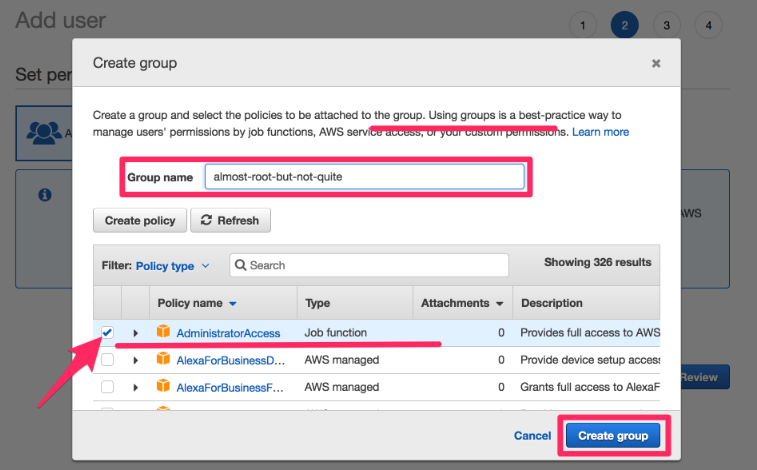
* EC2 credentials
* Flintrock

**Part A. Starting Spark in EC2**

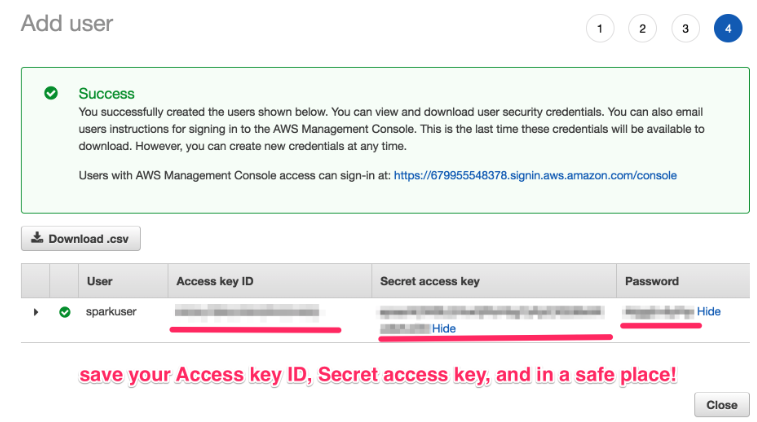
1. Log into your AWS account in a browser, and go to **My Security Credentials** (in a dropdown menu from your name in the top right end corner). You may get a warning like this:



1. Expand the section labeled **Access Keys**(I have blanked out key details in the screen shot)  
   
2. 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**.
3. On the page that appears
   1. Provide a username (e.g., sparkuser)
   2. Check the box next to both access types: “*Programmatic access*” as well as “*AWS Management Console access*”
   3. Click the **Next:Permissions** button
4. Select “**Add user to group**” and click “**Create group**”. Create a group with the following policy selected:



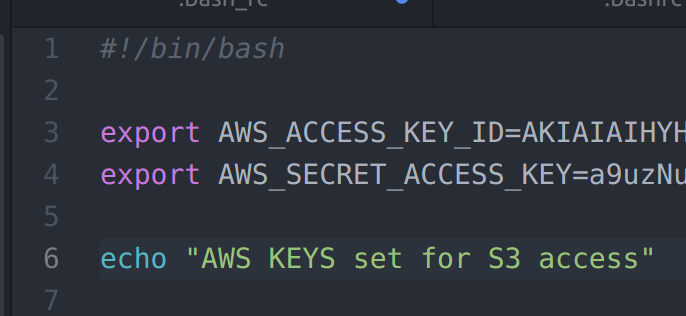
1. Add your user to this newly created group and click the **Next: Review** button. Having checked that the details look correct, click **Create user**.

****

1. **Download the key file**
2. 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)
3. On a command line type:  
   aws configure  
     
   You should see:  
   AWS Access Key ID [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*J3EA]:
4. Copy the Access Key ID from the keyfile, and then hit Enter
5. Do the same for the Secret Access Key
6. Set the default region to **eu-west-2**
7. Set the output to **json**
8. It should look something like this (but with your keys):
9. 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

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



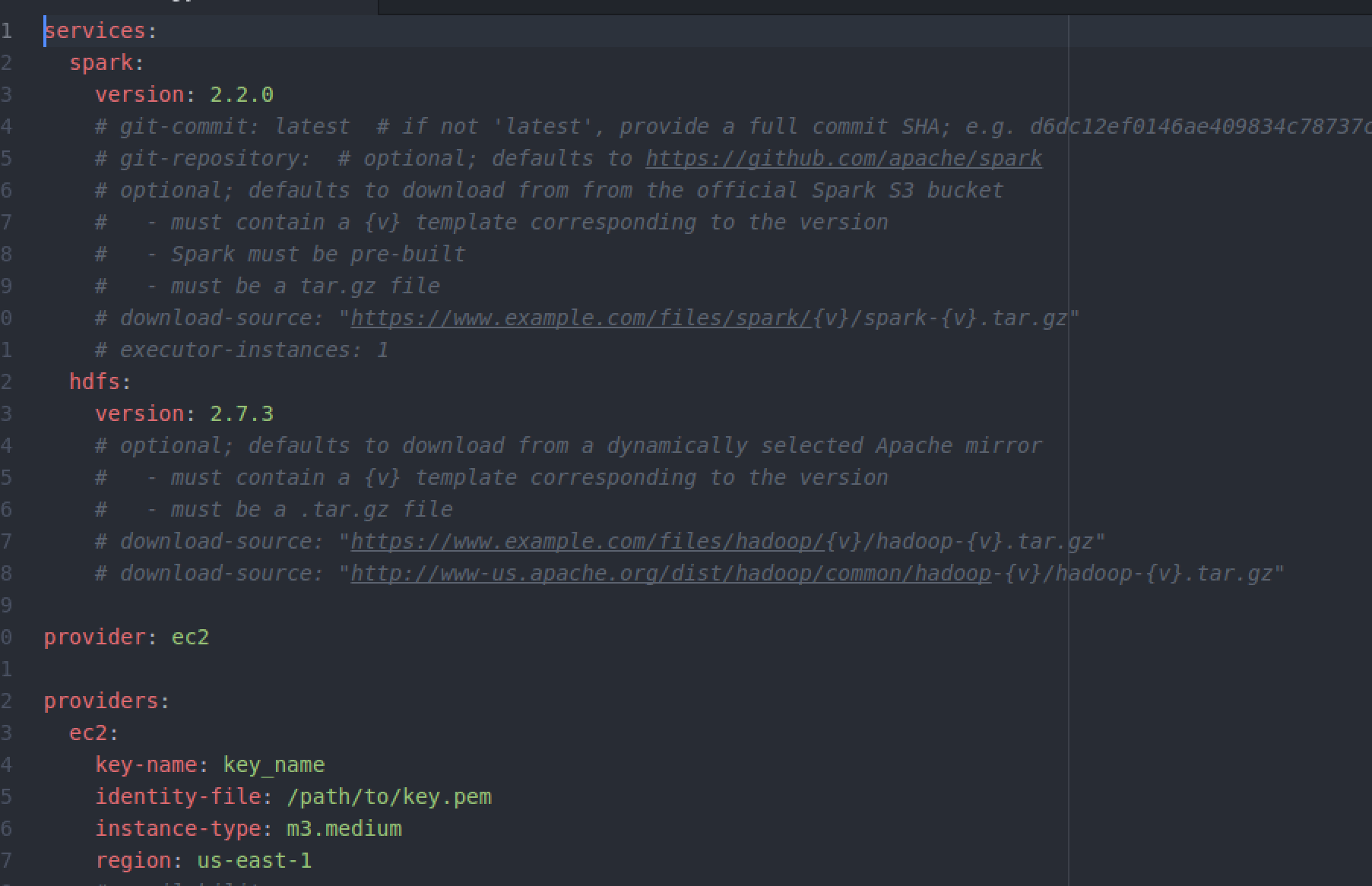
Make this script executable and run it by typing at the command line:

$chmod u+x awskeys

$source ./awskeys

1. 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.
2. 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.
3. 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

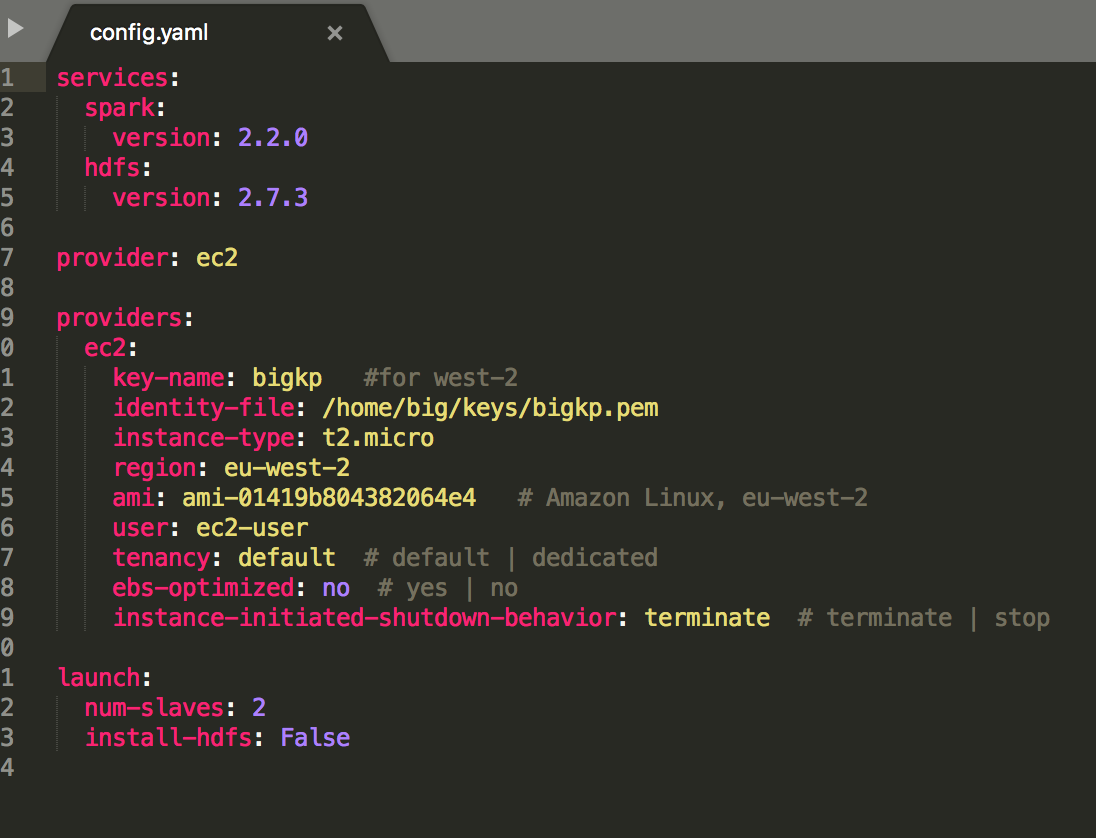
It will look something like:  


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:



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.

1. Replace the file in ~/.config/flintrock with the one from the github repository
2. Update it if necessary by typing

flintrock configure

1. You should now be able to launch a cluster in Amazon. From a new terminal window  
     
   flintrock launch big
2. 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.

1. 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
2. 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!

1. In wind-s3.py, delete or comment the first two lines (import findspark and findspark.init()).
2. Save the file
3. Once the launch of your cluster has finished, we need to copy the code into the cluster:  
     
   flintrock copy-file big ~/sql/wind-s3.py /home/ec2-user/
4. 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/

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 ~]$

1. Let’s login to the master (all one line):  
     
   flintrock login big  
   You see something like:

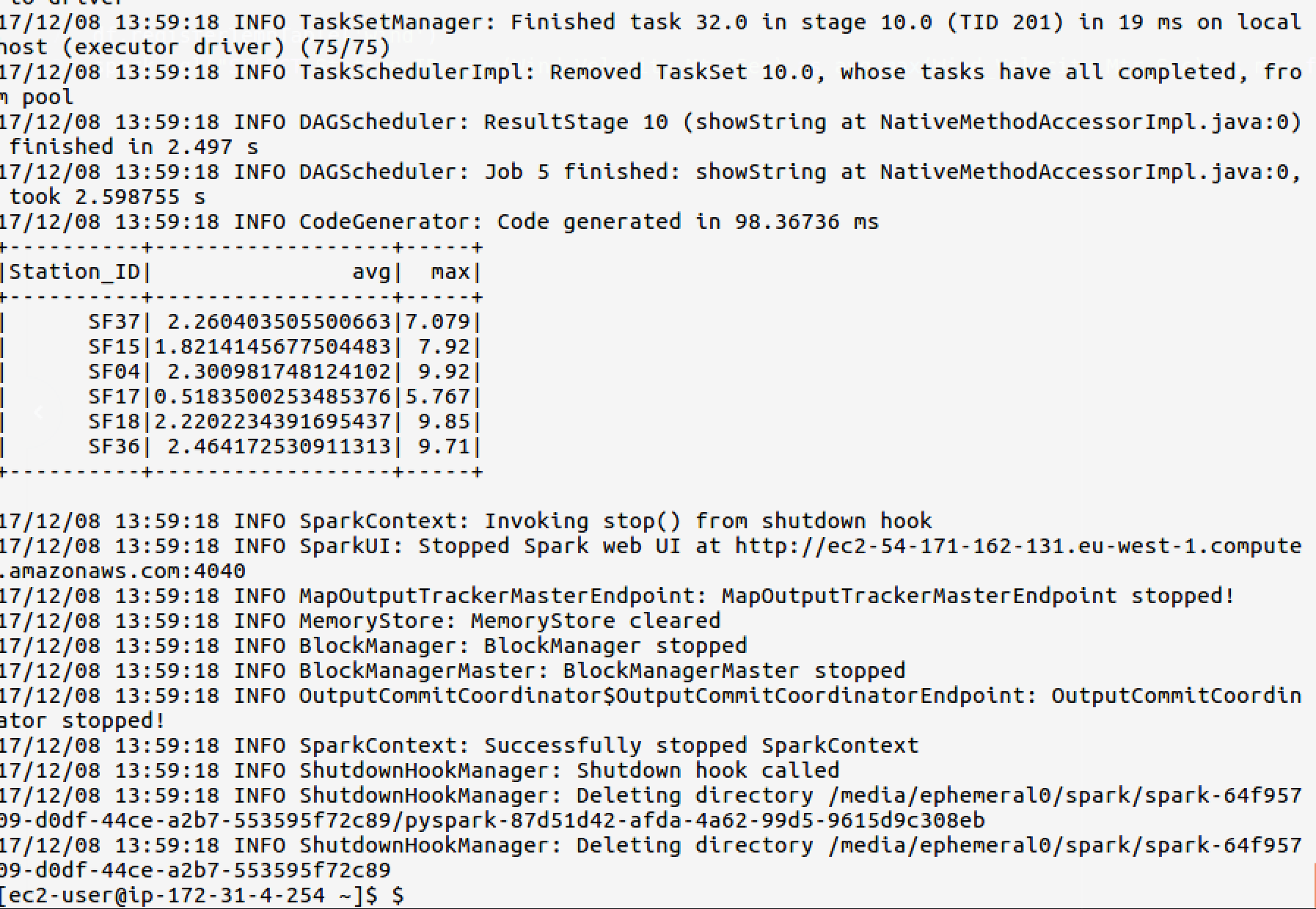
This basically has just SSH’ed you into the master.

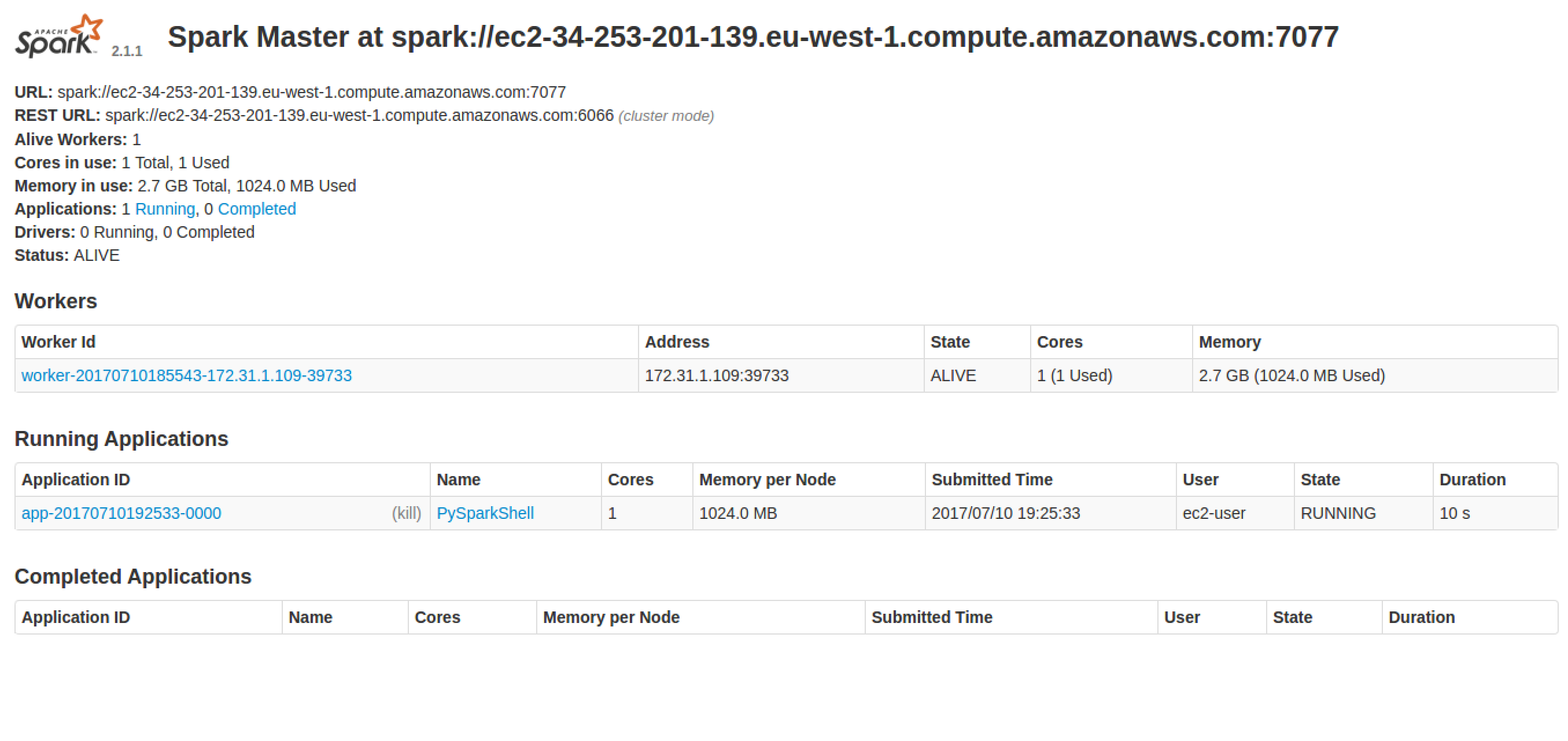
1. Now set your awskeys

$chmod u+x awskeys

$source ./awskeys

1. Now launch your code:  
     
   ~/spark/bin/spark-submit wind-s3.py
2. 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



1. 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.
2. 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
3. Go to <http://xx.xx.xx.xx:8080> using the master’s IP address.  
   You should see something like:
4. You can explore your Spark cluster here.
5. Exit the SSH session:  
   exit
6. 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.
7. Congratulations, this lab is complete.