**Exercise 4**

*Spark SQL on AWS*

**Prior Knowledge**

Unix Command Line Shell

Simple Python

**Learning Objectives**

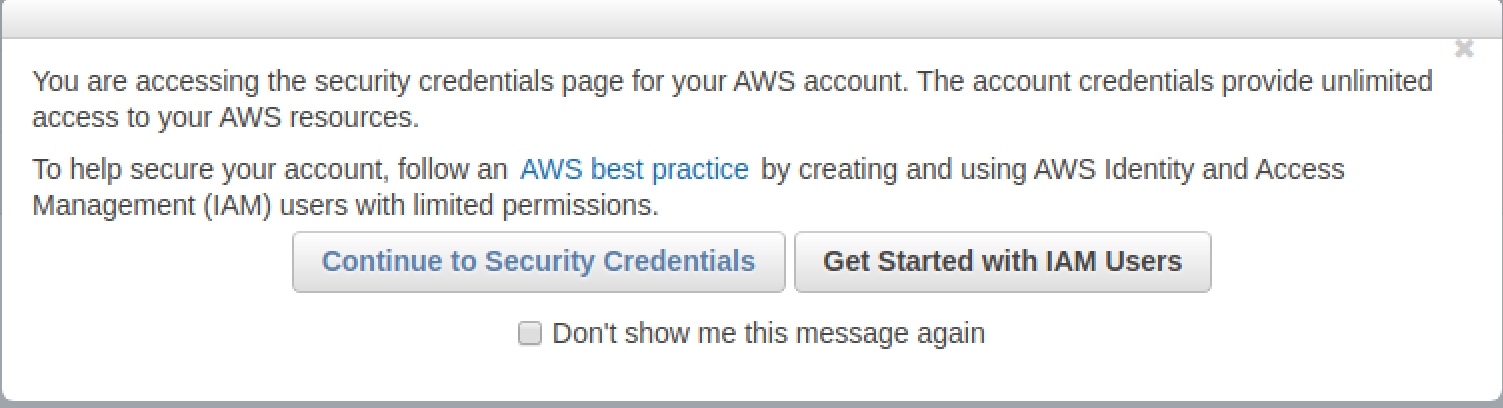
Understanding how to run Spark on AWS-EMR

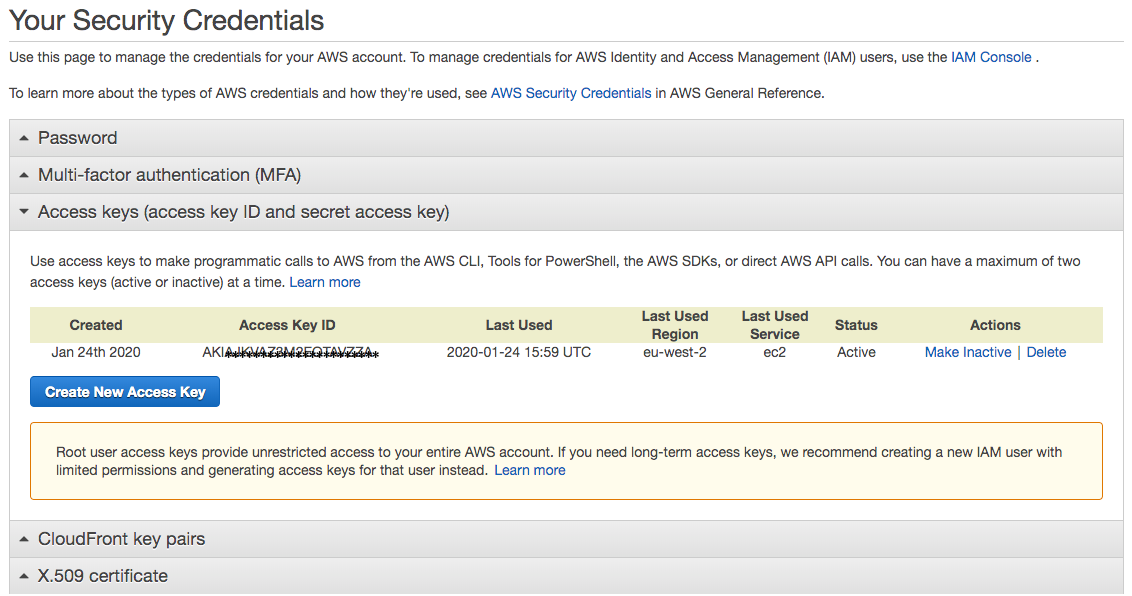
**Pre-requisites**

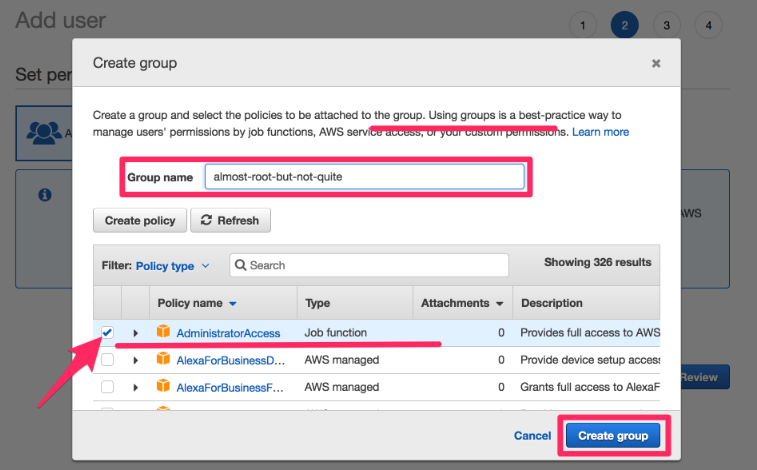
* AWS account with EC2 credentials

**Part A. Starting Spark in EC2**

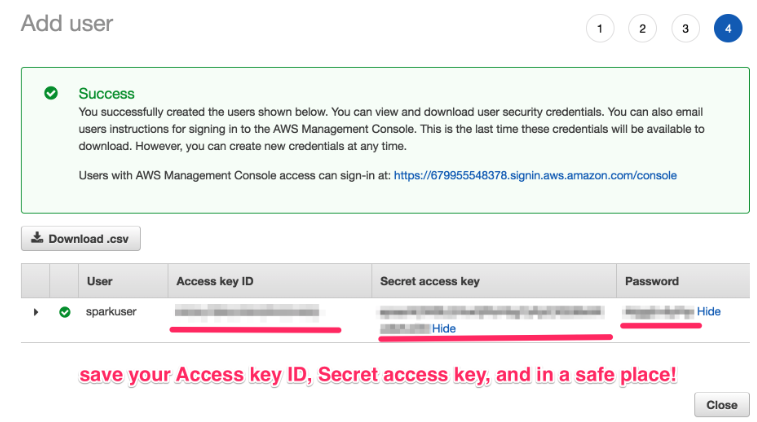
1. Log into your AWS account in a browser, and go to **My Security Credentials** (in a dropdown menu from your name in the top right end corner). You may get a warning like this:



1. Expand the section labeled **Access Keys**(I have blanked out key details in the screen shot)  
   
2. It’s 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. Leave the password options as they are
   4. 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:Tags** button and then **Next: Review** button. Having checked that the details look correct, click **Create user**.

****

1. **Download the key file (click Download .csv)**
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. Now we need to install and configure the AWS command line tool. Make sure you activate your bigdata conda environment and have installed aws and awscli via **pip**

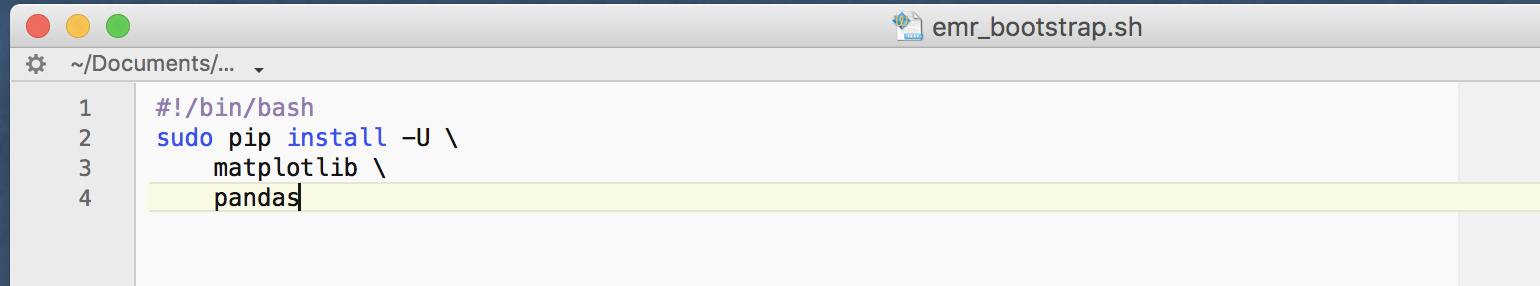
* conda activate bigdata
* pip install aws
* pip install awscli

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

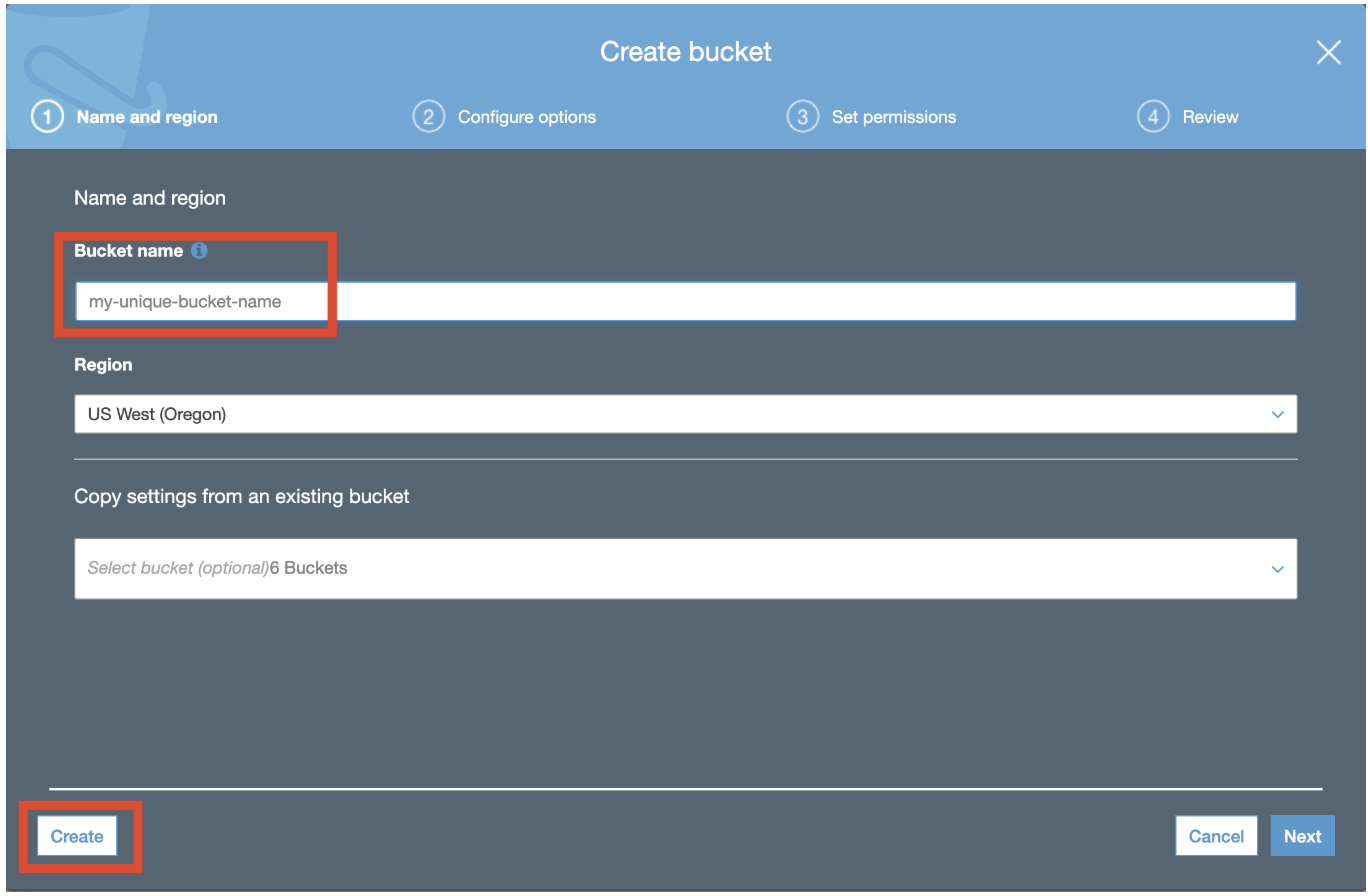
AWS Access Key ID [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*J3EA]: AKIASF22343434UNM33UVIA   
Secret Access Key [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*JXb7]: 8z1rtTbU3Ur/llksdafkjhd398u34msndHnGaDY   
Default region name [eu-west-1]: eu-west-2

Default output format [json]: json

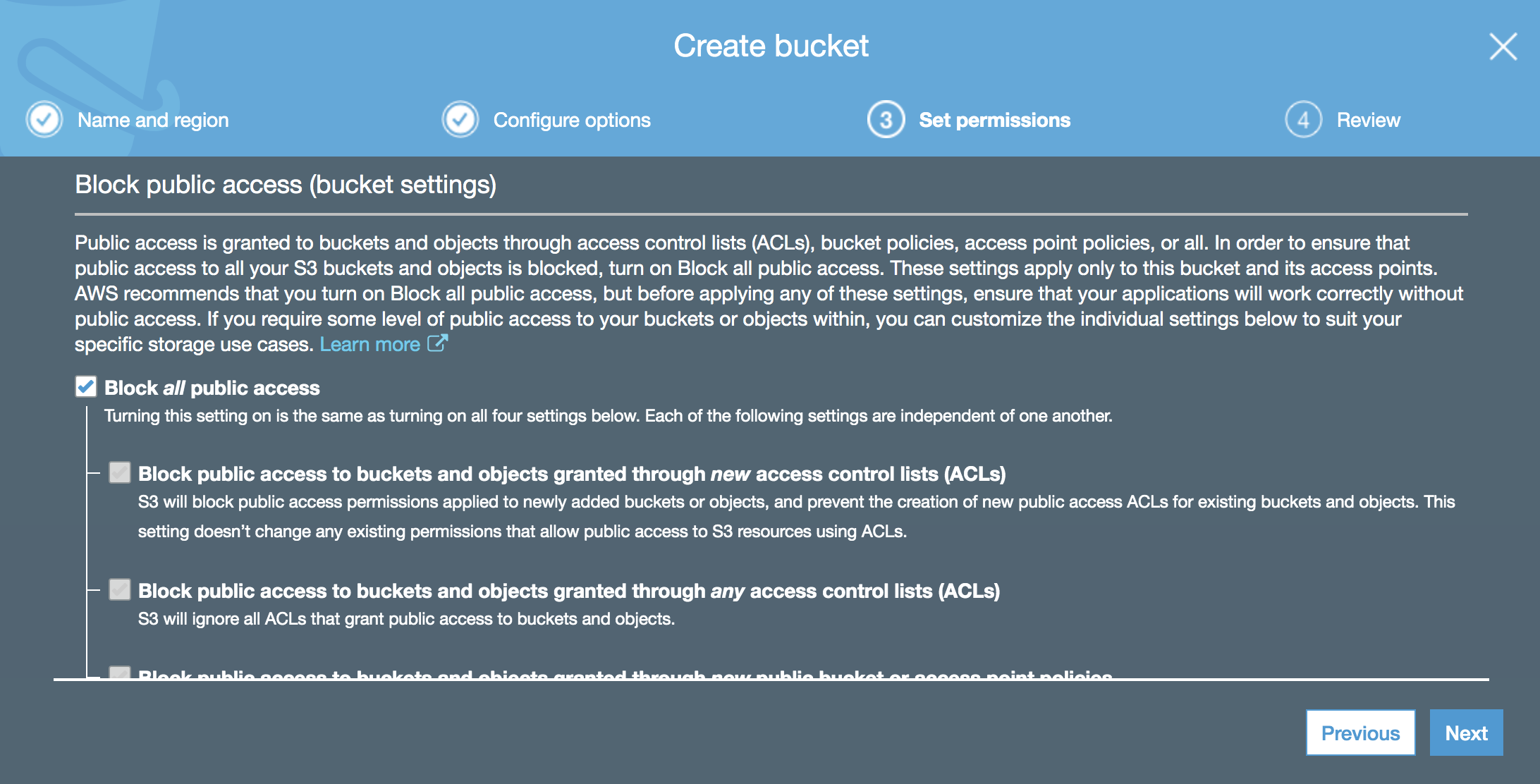
1. In addition to these “Access Keys”, we also need an SSH key to continue.   
   If you successfully completed the pre-course Amazon lab,   
   you should have a file ~/keys/bigkp.pem . If not, look at the pre-course instructions or grab one of the instructors to help you create one.
2. To install useful python packages on all nodes of our cluster, we need to create a file emr\_bootstrap.sh, create a bucket on S3 and store the file emr\_bootstrap.sh there. First, create the file:



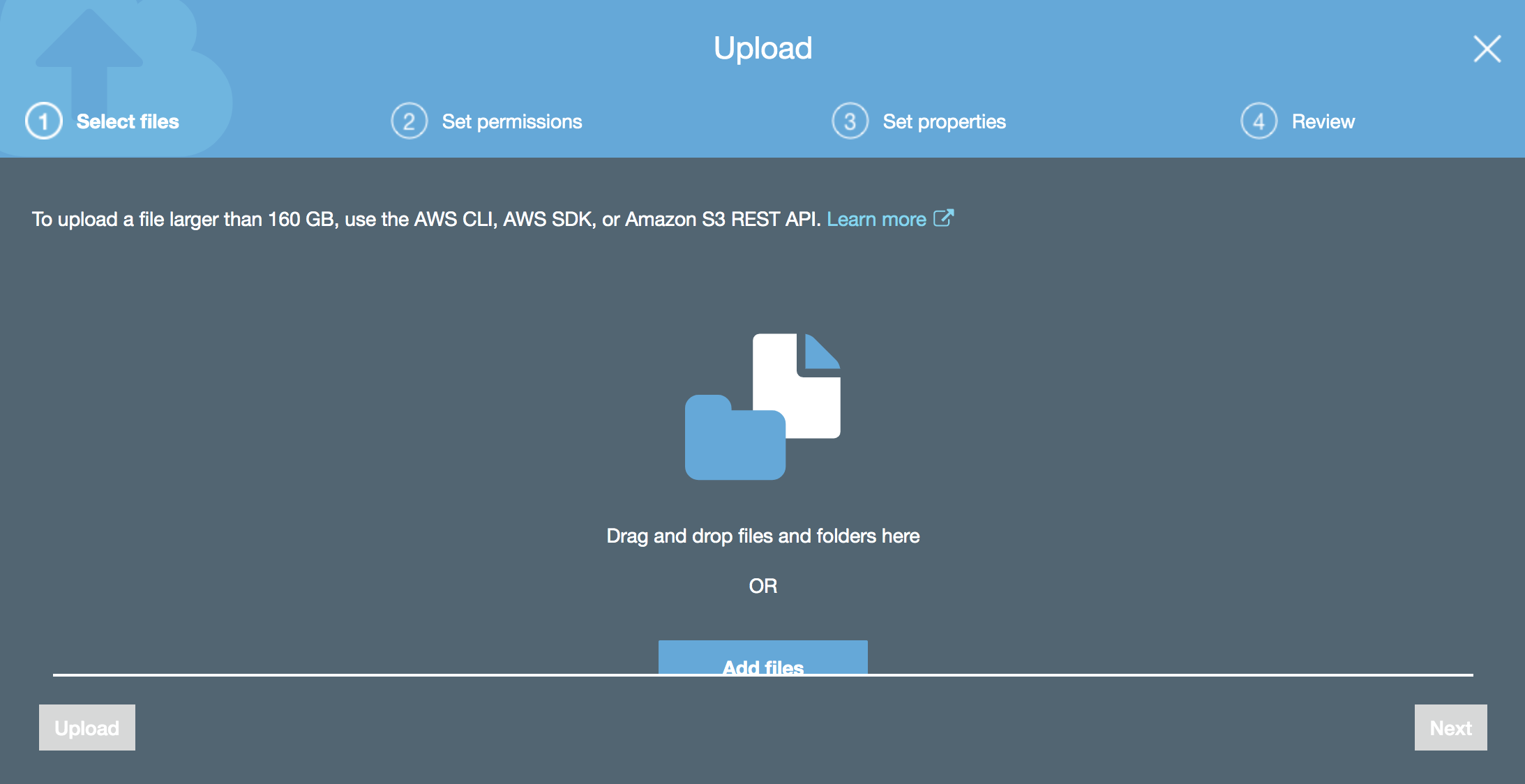
1. Login to the AWS console in a web browser. Try using the IAM user credentials you created earlier. There is a weblink provided as the final entry in the credentials file (which contains this unique user’s id).
2. Navigate to S3.
3. Now we are going to create a bucket to store your data and code. Click **Create Bucket**. Give your bucket a name, select the region **EU (London**) and then click **Next**



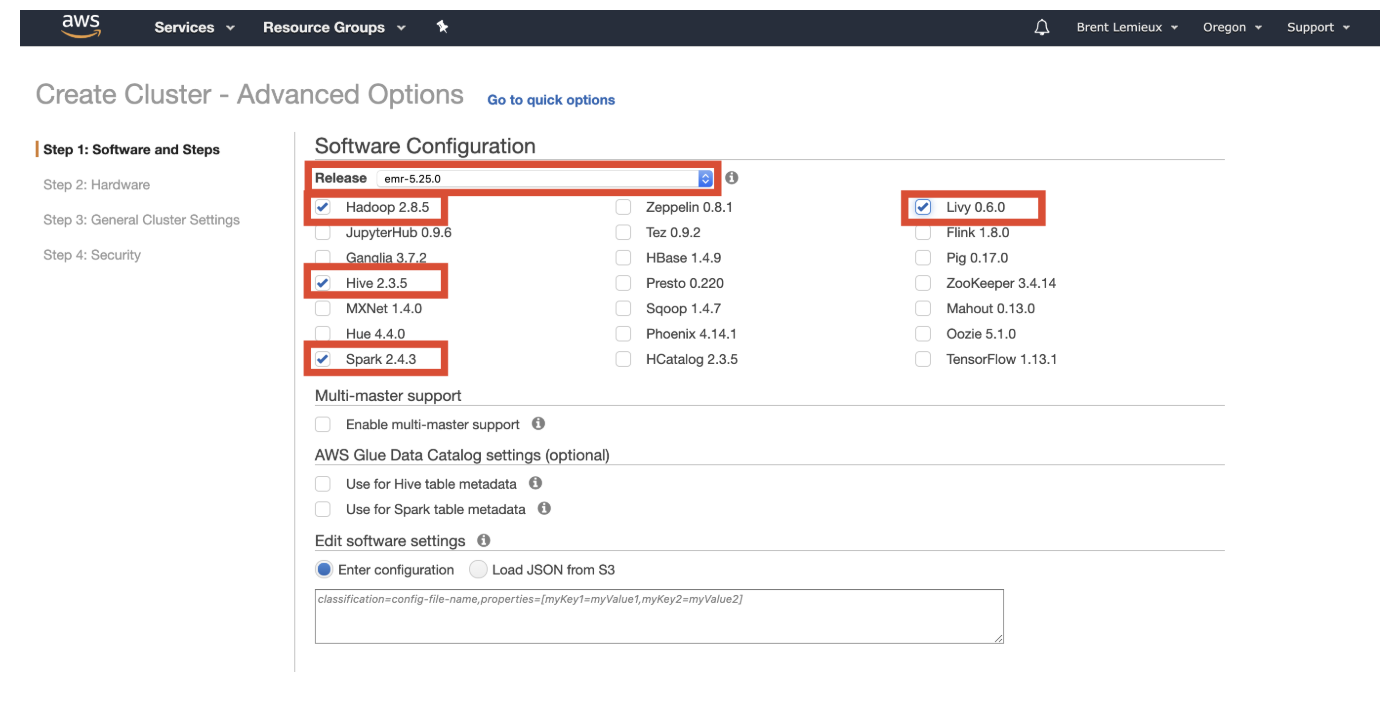
1. You can leave the options as they are (unless you want other people to be able to access this bucket). Click **Next** on the **Configure options** tab and **Set permissions** tab. Then on **Review**, click **Create**



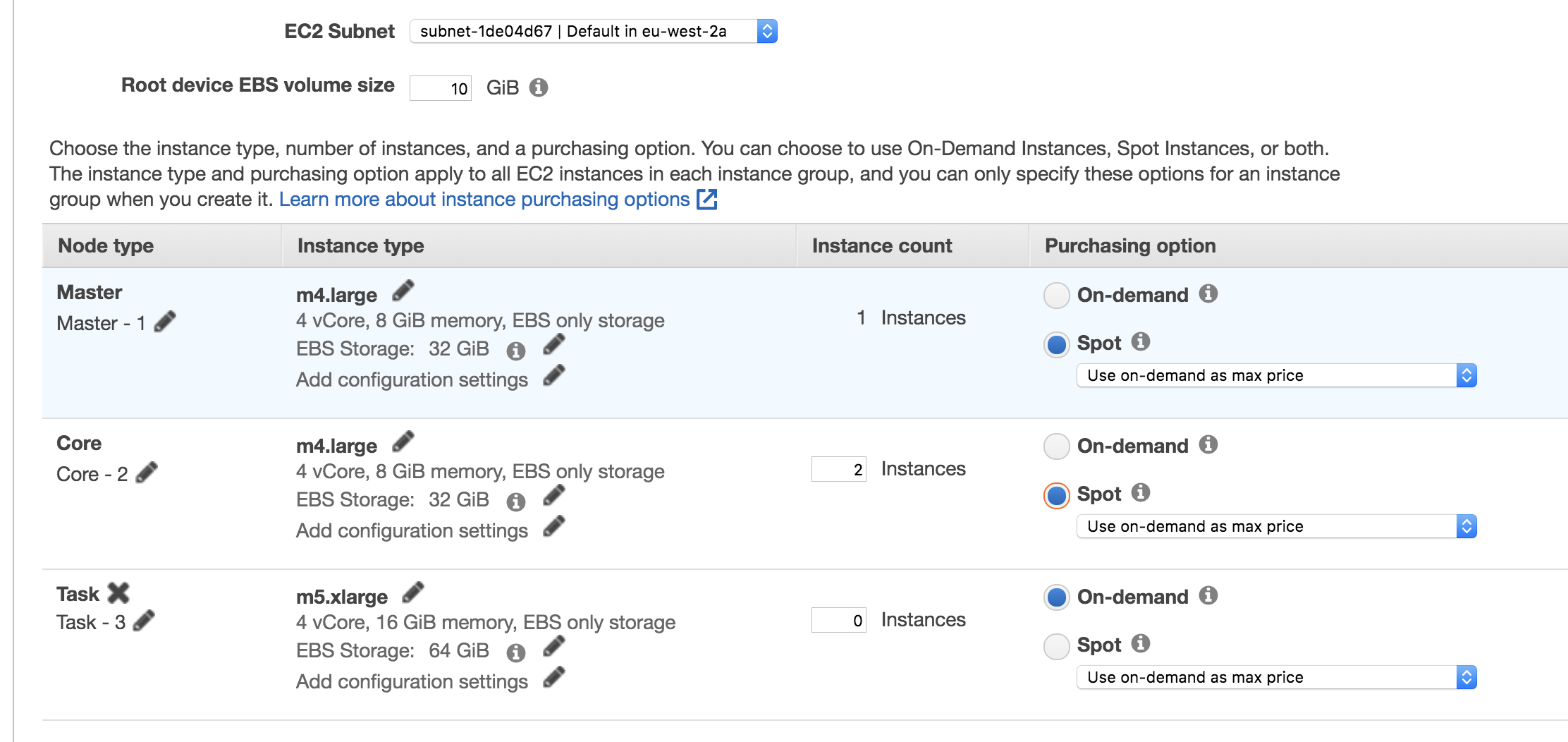
1. Then we need to upload the file. Select your newly created bucket, then click **Upload**. You can then drag-and-drop the emr\_bootstrap.sh file



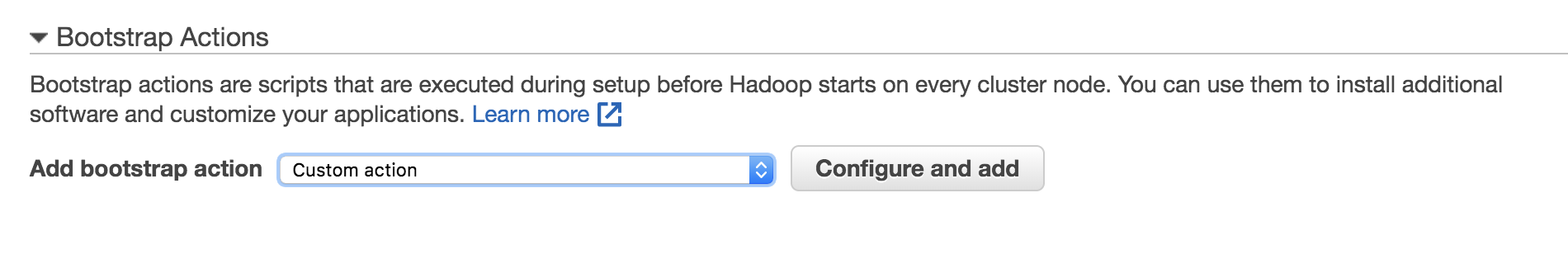
1. Now we are ready to set up our first cluster. We are going to do this using Amazon’s EMR service. So, first navigate to EMR on your console. Click **Create cluster** and **Advanced Options**.
2. Within the **Advanced Options Software Configuration**, choose the latest release from the release dropdown menu and then make sure you check **Spark** before clicking Next.



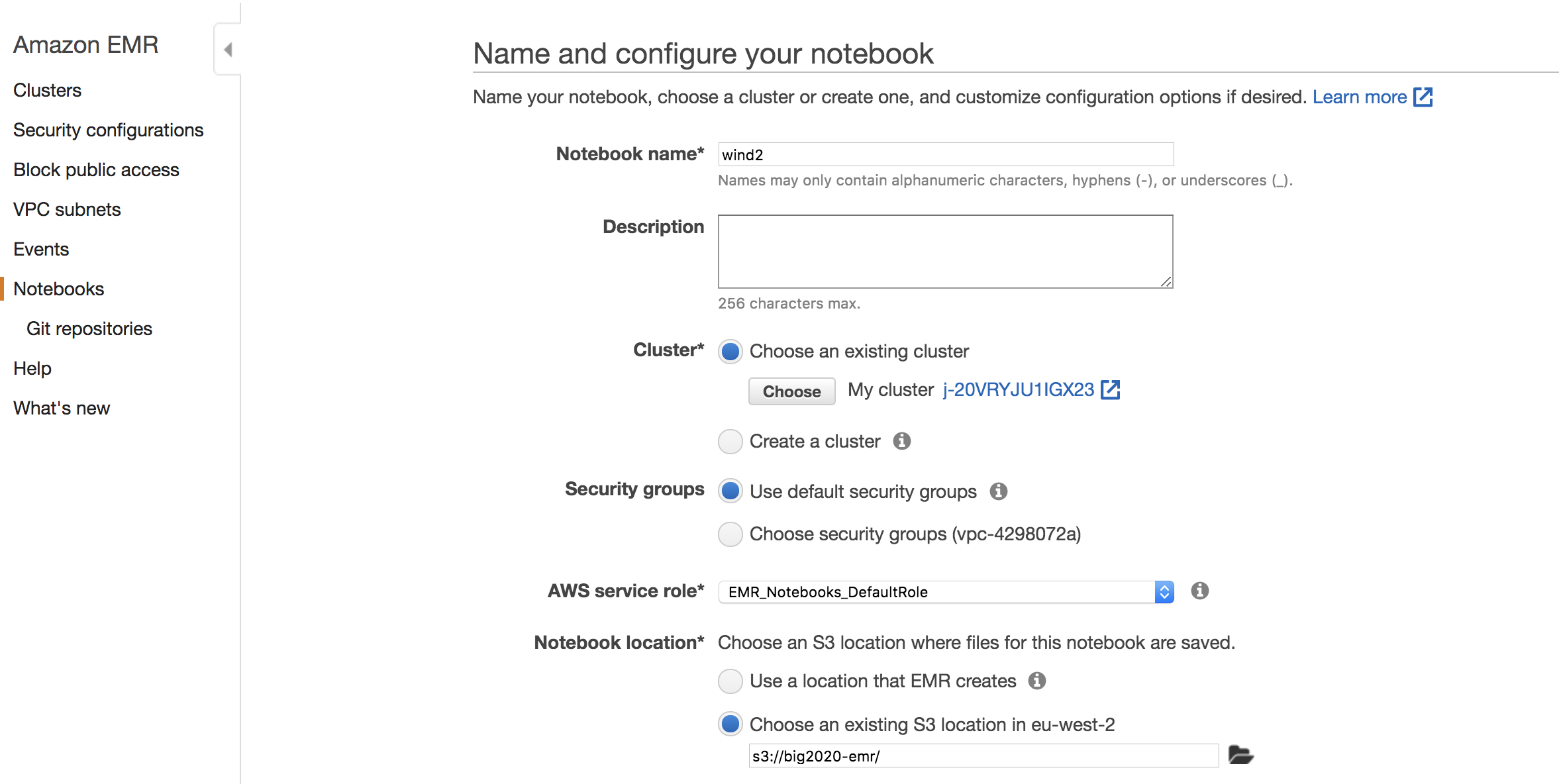
1. On the **Hardware Configuration**, choose ‘*Default on EU-West-2a*’ from the EC2 Subnet dropdown menu. Choose **m4.large** instance types whilst you are just learning how to use EMR (as these are the cheapest!). Choosing Spot instances should also reduce costs.



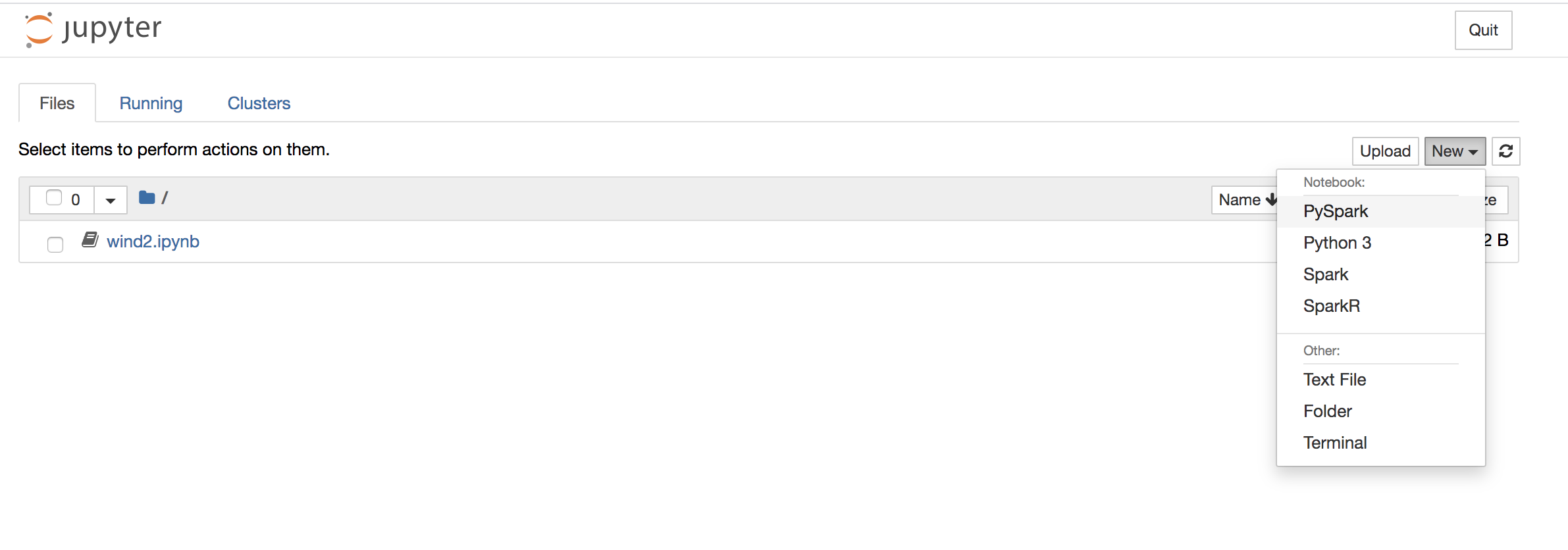
1. On **General options**, you need to add a custom bootstrap action (in order to run the code we uploaded earlier on each node in the cluster on start-up).



1. Select and the file from your S3 storage and add it to the cluster.
2. On **Security**, select your bigkp key pair that you created earlier (see step 16).
3. You can then click **Create cluster** and AWS will provision your cluster (this may take 5-10 minutes).
4. Once it is ready, we are going to run a Jupyter Notebook on the master node. We will use the same code as previously with a few minor modifications (regarding packages and file locations).
5. Navigate to **Notebooks** in the left hand panel. Click **Create notebook** and then configure it. You will need to give the notebook a name and choose the cluster you have created to run it on.



1. Clicking **Create notebook** will start the notebook once the cluster is ready. Click **Open in Jupyter**
2. This will open the familiar Jupyter notebook system in our browser. However, annoyingly, the notebook you have just created won’t actually have pyspark available. If you want to use pyspark (rather than just standard python libraries), you need to create a new notebook here – choosing a **PySpark** notebook. Note, you will not be able to successfully run the notebook until the cluster has completed bootstrapping.

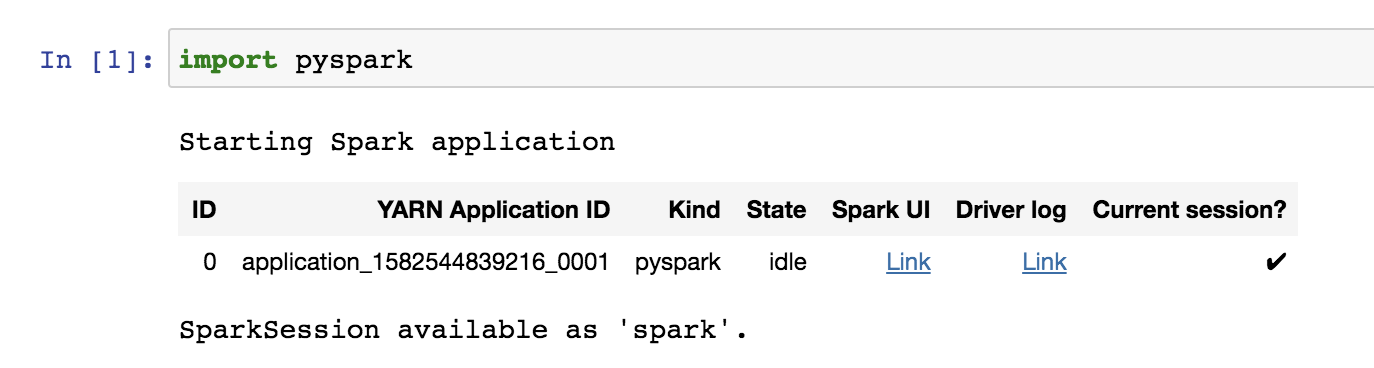


1. In the meantime, you could start yet another terminal window and prepare your code to run on AWS.  
     
   cd sql  
   cp wind.py wind-s3.py
2. Change the URL so that instead of loading the data from the local filesystem, it reaches out to S3 to do it:   
     
   Instead of reading from ‘/home/big/sql/\*.csv’ change it to read from:  
     
   's3a://discnet-big/wind2015/\*'

This is a public S3 bucket I have created to store the data. [[1]](#footnote-1)

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 for future reference.
3. Now, once your notebook is ready on the cluster, check that you can import pyspark
4. Now, copy the rest of the code from wind-s3.py into your jupyter notebook on the cluster.



1. Run the notebook. You could carry out some other queries too.
2. Stop the notebook in Jupyter and also within the AWS-EMR console.
3. **Finally, remember to stop the cluster as well (its costing money…**). You will probably need to remove termination protection before AWS allows you to shut the cluster down.
4. Congratulations, this lab is complete.

1. Alternatively, you can upload the data to your own S3 bucket using the AWS console or 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 [↑](#footnote-ref-1)