

How to run Spark on EC2

Monday, March 15, 2021 8:01 PM

After having your Apache Spark up and running on your local machine, now it is time to get more advance and enjoy the real power of Apache Spark computation on Amazon EC2. It is recommended to get more official details on [Spark Official Guidance](#).

✓ Download spark-ec2 script under Spark's ec2 directory

First, make sure you have already downloaded Apache Spark on your local machine. Spark does no longer provide the ec2 directory as part of the official distribution. So we need to download or use git clone the ec2 directory [here](#).

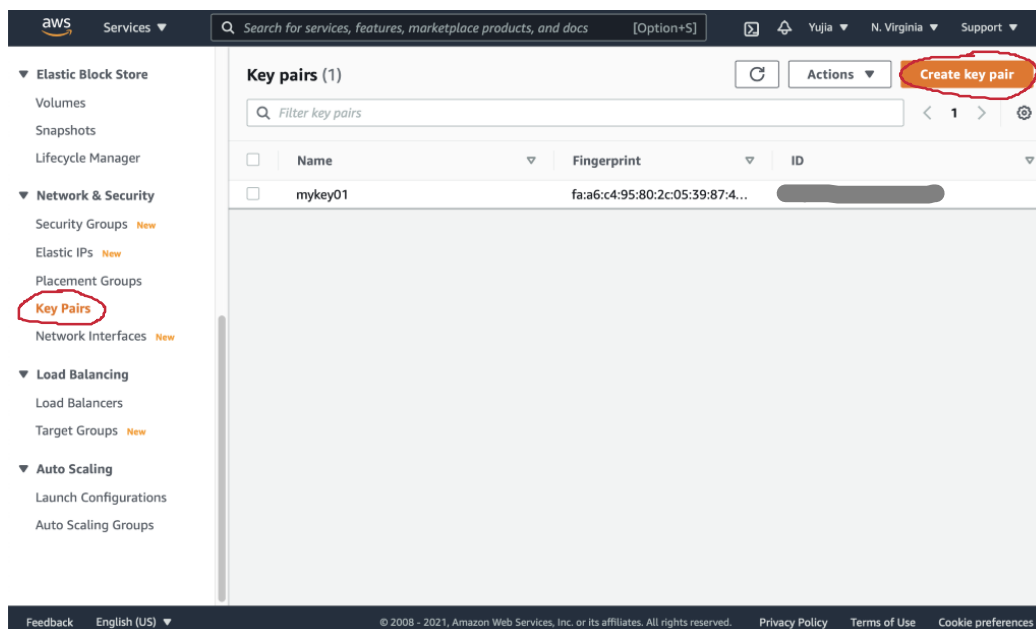
```
$git clone https://github.com/amplab/spark-ec2
```

✓ Preparation on the Amazon Web Services site

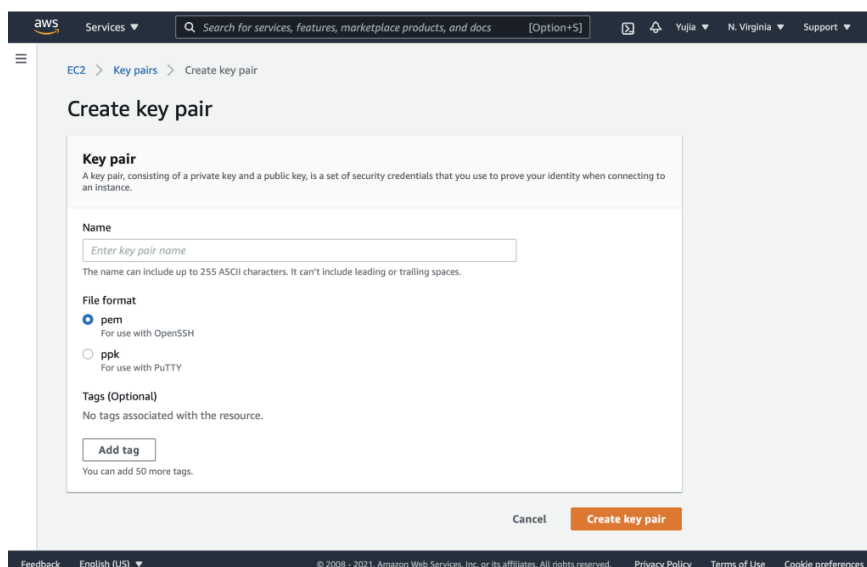
Sign up for an account on the Amazon Web Services site. You can try an AWS Free Tier for 12 months.

✓ Create an Amazon EC2 key pair

- Log into your Amazon Web Services account through the AWS console, clicking Key Pairs on the left sidebar, and creating a key.



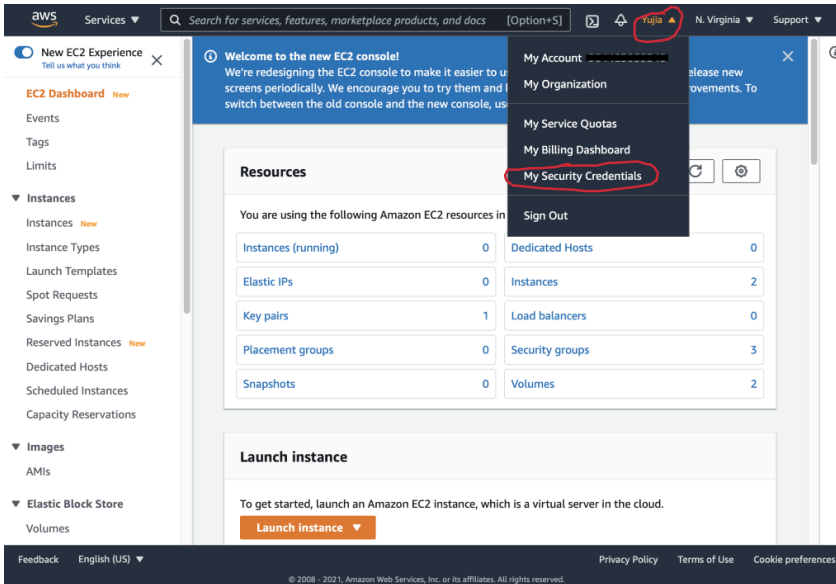
- Create a key pair and download the .pem file. Remember the path you saved the .pem file.



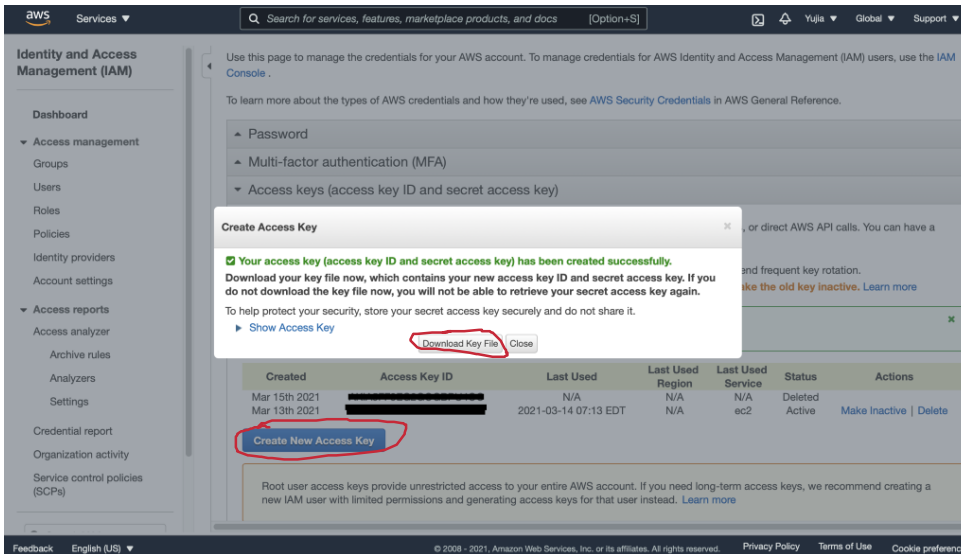
! Pay attention to the region and time zone when you create key pair. It will cause error when you set improper region. If you locate in New Jersey, it is recommended to choose "US East N. Virginia" (us-east-1).

✓ Create an IAM user

- An IAM user is an entity that you create in AWS. It can give you the ability to sign in to the AWS Management Console for interactive tasks and to make programmatic requests to AWS services using the API or CLI.
- Open your AWS console. Choose your account -> "My Security Credentials".



- Create and remember your access account and ID. You will not be able to see it again after closing this window.



✓ Launch a Cluster

- Go into the ec2 directory in the release of Spark.
- Set the permissions for the .pem file to 600(i.e., only you can read and write it) so that ssh will work.

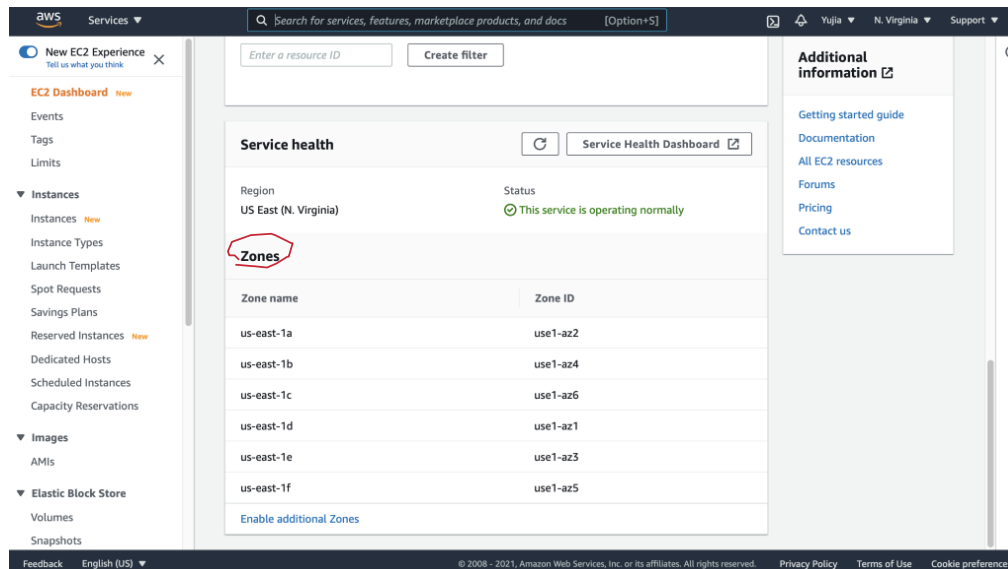
```
$chmod 600 {your_key_file}.pem
```
- Add the "Access key ID" and "Secret access key" to your ~/.profile or ~/.bash_profiles so that you can sign into the AWS Management Console as an IAM user from the terminal.

```
$export AWS_ACCESS_KEY_ID={your_access_key_id}
$export AWS_SECRET_ACCESS_KEY={your_secret_access_key}
```
- Now, we can use ./spark-ec2 to launch our EC2 instances. Run ./spark-ec2 -k <keypair> -i <key-file> -s <num-slaves> launch <cluster-name>, where <keypair> is the name of your EC2 key pair (that you gave it when you created it), <key-file> is the private key file for your key pair, <num-slaves> is the number of slave nodes to launch (default number is 1), and <cluster-name> is the name to give to

your cluster. For example,

```
$. /spark-ec2 --key-pair=awskey --identity-file=awskey.pem --region=us-east-1 --zone=us-east-1a  
launch my-spark-cluster
```

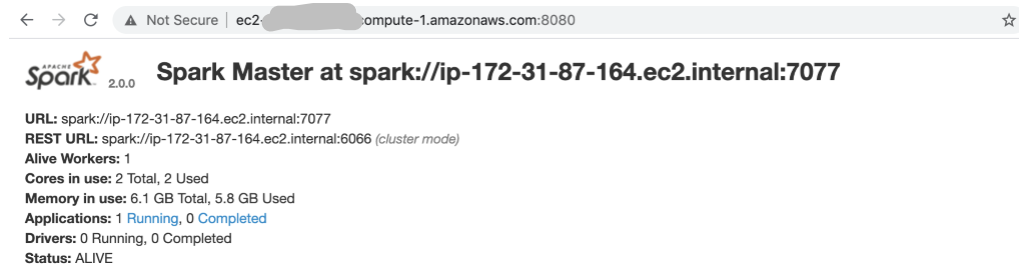
- Check your region and available zones on your EC2 Dashboard.



Zone name	Zone ID
us-east-1a	use1-az2
us-east-1b	use1-az4
us-east-1c	use1-az6
us-east-1d	use1-az1
us-east-1e	use1-az3
us-east-1f	use1-az5

- After you launch your instances, you will see information like below in your terminal. You can use the highlighted url to open your web UI.

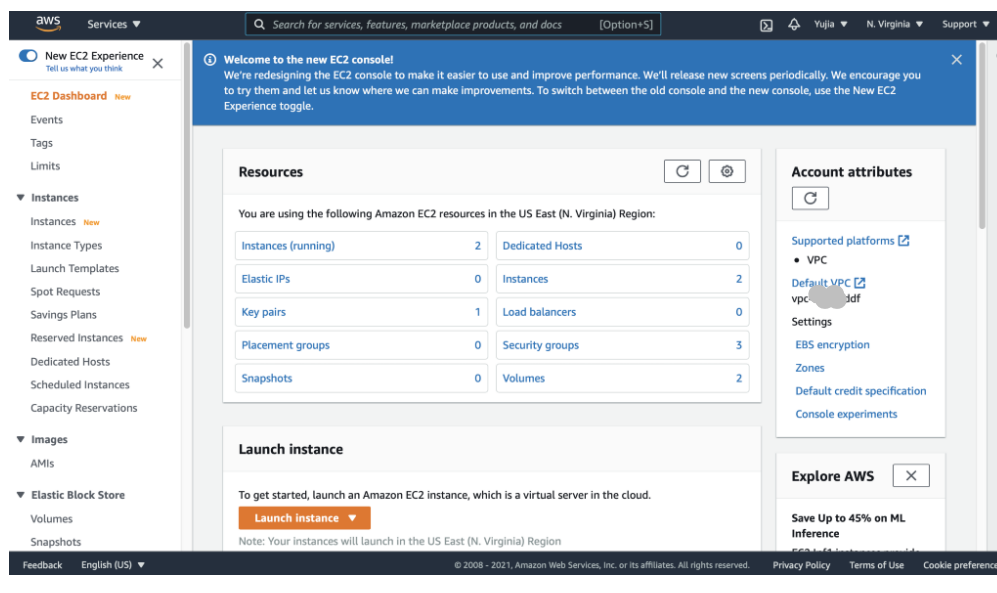
```
Connection to ec2-3-86-247-154.compute-1.amazonaws.com closed.  
Spark standalone cluster started at http://ec2-3-86-247-154.compute-1.amazonaws.com:8080  
Ganglia started at http://ec2-3-86-247-154.compute-1.amazonaws.com:5080/ganglia  
Done!  
./ec2/spark_ec2.py:1565: ResourceWarning: unclosed <ssl.SSLSocket fd=6, family=AddressFam  
ily.AF_INET, type=SocketKind.SOCK_STREAM, proto=6, laddr=('192.168.1.119', 62321)>  
  real_main()  
hanukas-MacBook-puro:2.4.4 fanyujia$
```



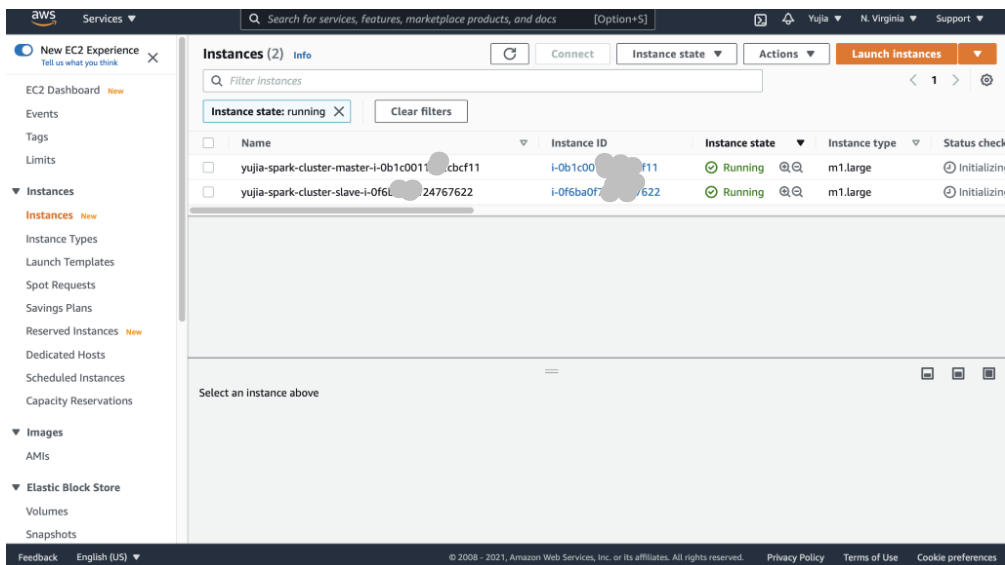
Spark Master at spark://ip-172-31-87-164.ec2.internal:7077

URL: spark://ip-172-31-87-164.ec2.internal:7077
REST URL: spark://ip-172-31-87-164.ec2.internal:6066 (cluster mode)
Alive Workers: 1
Cores in use: 2 Total, 2 Used
Memory in use: 6.1 GB Total, 5.8 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

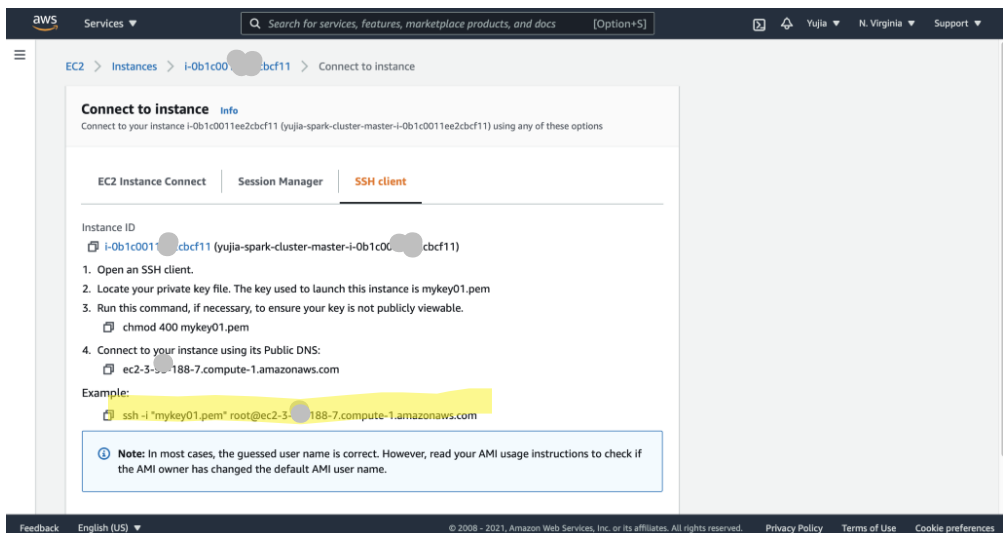
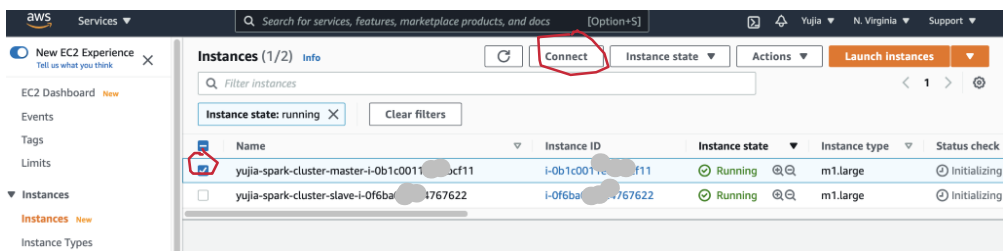
- If you successfully launch a cluster with 1 master and 1 slave, you can see there are 2 instances running on your EC2 Dashboard.



Resources	
Instances (running)	2
Elastic IPs	0
Key pairs	1
Placement groups	0
Snapshots	0
Dedicated Hosts	0
Instances	2
Load balancers	0
Security groups	3
Volumes	2



- Select your master instance, and click Connect, you can find an option to ssh to your master.



- Once you log in, you will find some native libraries has been installed for you.

```
--|  --|  )
--|  (  /   Amazon Linux AMI
--|\_---|---
```

<https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/>
Amazon Linux version 2018.03 is available.

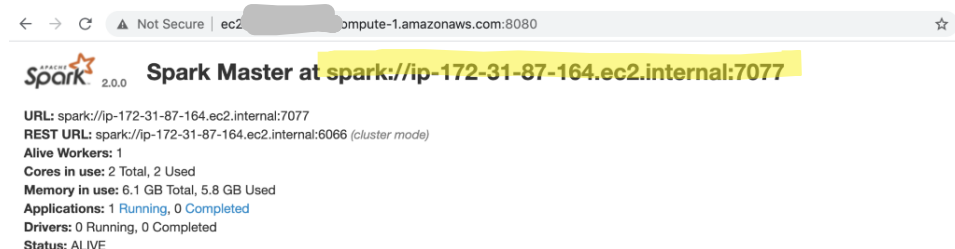
```
root@ip-172-31-87-164 ~]$ ls
ephemeral-hdfs  hadoop-native  mapreduce  persistent-hdfs  scala  spark  spark-ec2  tachyon
```

- ! If you find ERROR:Peer reports incompatible or unsupported protocol version, try to update Linux nss, curl on your master.
`$yum update nss curl`

✓ Run your application on the Spark cluster

After log in to the cluster, you can run your application now. Here we use spark-submit to run either .py/.jar files.

- You can either write a script directly in the remote machine using the terminal or upload a local file from your local machine. It is recommended to upload a local file to save time and money running EC2 instances.
- To upload a local file to the remote machine, make sure Master is set the same as the IP address in the web UI in your code.



- In the remote machine, create a new directory in your master node.

```
$mkdir example_code
```

- In your local machine, use command scp to upload the file into the remote machine

```
$scp -i mykey01.pem -r localDir root@ec2-***.compute-1.amazonaws.com:/root/example_code
```

-i mykey.pem	if you are not in the directory of the key pair, make sure you write the absolute path of the .pem file
-r localDir	it is the directory in your local machine where you store your .py/.jar file
root@ec2-***.compute-1.amazonaws.com	it is the Public DNS of your master instance. You can find it on your AWS console.

- After we upload the file to the master node, we need to copy it to all the slave nodes. We do this in the remote machine using the command below.

```
$~/spark-ec2/copy-dir /root/example_code
```

- Now we can run spark-submit.

```
$/root/spark/bin/spark-submit --deploy-mode client {your_file} > output.txt
```

- The output results will be stored in output.txt. Remember to download the output to your local machine. Once you terminate your instance, it will no longer exist.
- You can also check your online Web UI to check your application status.

☑ Stop or terminate your instance

- Never forget this step or you will be charged. You can do it either manually from the AWS console or use commands in your terminal.
- Below are some example commands to start/stop/terminate the cluster in your terminal.

- To start your cluster:

```
$/ec/spark-ec2 --identity-file=mykey01.pem --region=us-east-1 --zone=us-east-1a start my-spark-cluster
```

- To stop your cluster:

```
$/ec/spark-ec2 --region=us-east-1 --zone=us-east-1a stop my-spark-cluster
```

- To terminate your cluster:

```
$/ec/spark-ec2 --identity-file=mykey01.pem --key-pair=mykey01 --region=us-east-1 --zone=us-east-1a destroy my-spark-cluster
```

- Double check you instances are truly stopped or terminated in your AWS console before you close it.