**Spinning Spark Cluster on EC2**

**Amazon Elastic Compute Cloud (Amazon EC2**) provides **scalable** **computing** **capacity** in the Amazon Web Services (AWS) cloud. Using Amazon EC2 **eliminates** your need to **invest** in **hardware** up front, so you can **develop** and **deploy** **applications** **faster**.

**This document walks you through the end-to-end steps involved in setting up an EC2 spark cluster in fully distributed mode, using spark-ec2 script.**

### **Prerequisites**

You need an [AWS account](https://via.hypothes.is/https:/www.amazon.com/ap/signin?openid.assoc_handle=aws&openid.return_to=https%3A%2F%2Fsignin.aws.amazon.com%2Foauth%3Fresponse_type%3Dcode%26client_id%3Darn%253Aaws%253Aiam%253A%253A015428540659%253Auser%252Fhomepage%26redirect_uri%3Dhttps%253A%252F%252Fconsole.aws.amazon.com%252Fconsole%252Fhome%253Fstate%253DhashArgs%252523%2526isauthcode%253Dtrue%26noAuthCookie%3Dtrue&openid.mode=checkid_setup&openid.ns=http://specs.openid.net/auth/2.0&openid.identity=http://specs.openid.net/auth/2.0/identifier_select&openid.claimed_id=http://specs.openid.net/auth/2.0/identifier_select&openid.pape.preferred_auth_policies=MultifactorPhysical&openid.pape.max_auth_age=43200&openid.ns.pape=http://specs.openid.net/extensions/pape/1.0&server=/ap/signin&forceMobileApp=&forceMobileLayout=&pageId=aws.ssop&ie=UTF8) to access AWS service offerings. If you are creating a new account, Amazon requires a credit card number on file and telephone verification.

**Key Takeaways**

1. Spinning an EC2 spark cluster in fully distributed mode.
2. Launching. starting and stopping a spark cluster.
3. Creating a working pyspark environment integrated with a Jupyter notebook.
4. Uploading and accessing data from the S3 bucket.
5. Running pyspark jobs over the EC2 spark cluster.

## **Why EC2?**

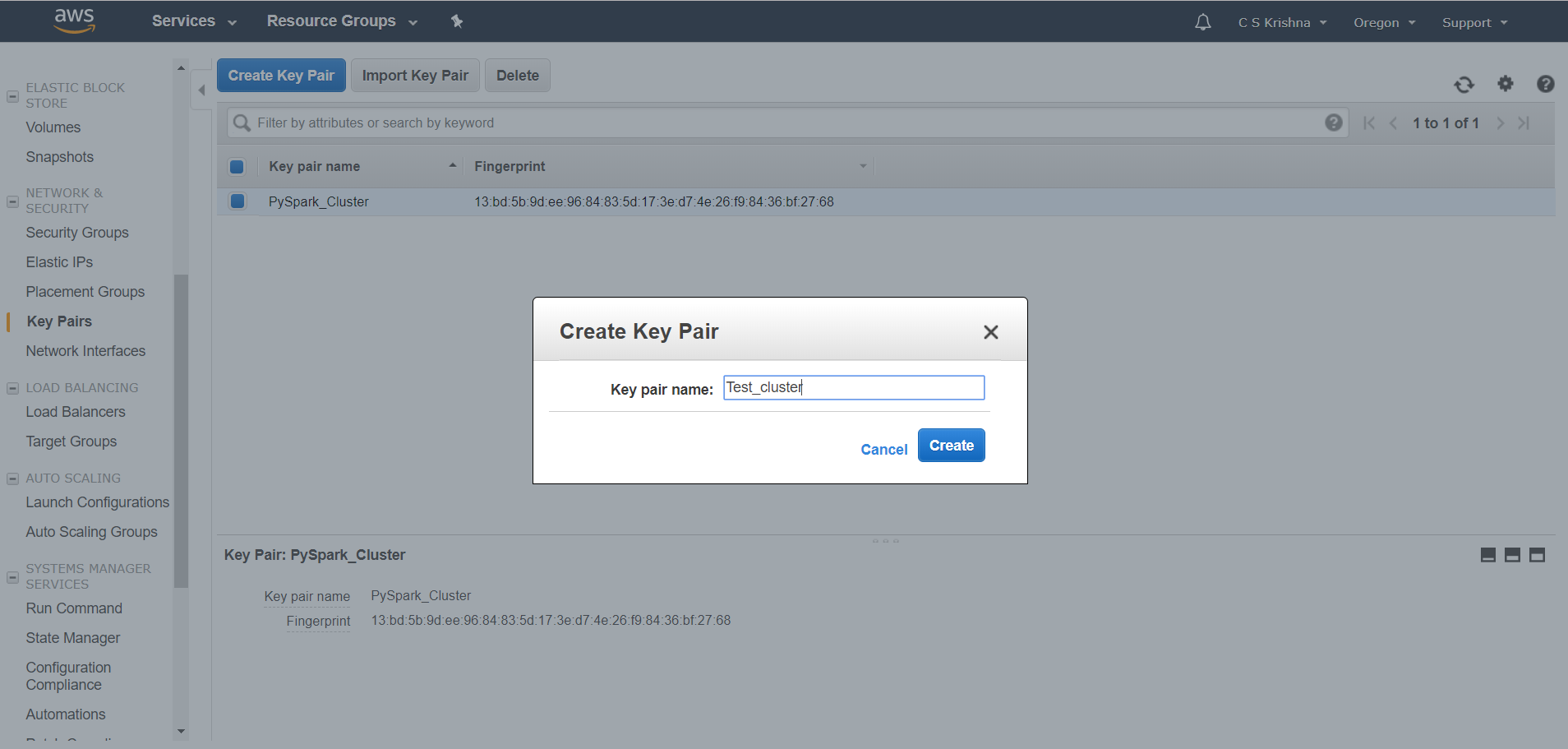
Amazon EC2 is a cloud-based service that allows you to quickly configure and launch new servers in a pay-as-you-go model. Once it's running, an EC2 instance can be used just like a physical machine in your office. The only difference is that it lives somewhere on the Internet, or "in the cloud". Although we could choose to install Spark on our local development environment, but EC2 is preferred for the following reasons:

1. EC2 results in a clean and consistent baseline, independent of whatever is already installed in your development environment.
2. Your Spark experimentation on an EC2 instance is isolated from your local development work.
3. Becoming familiar with EC2 puts you in a better position to work with Spark clusters spanning multiple servers in the future.

**Before** **spinning** Spark cluster, we **need** an **EC2 instance** with **spark installation** from where we can **spawn** **spark** **cluster** using **Spark-EC2 script**.

**Before You Start**

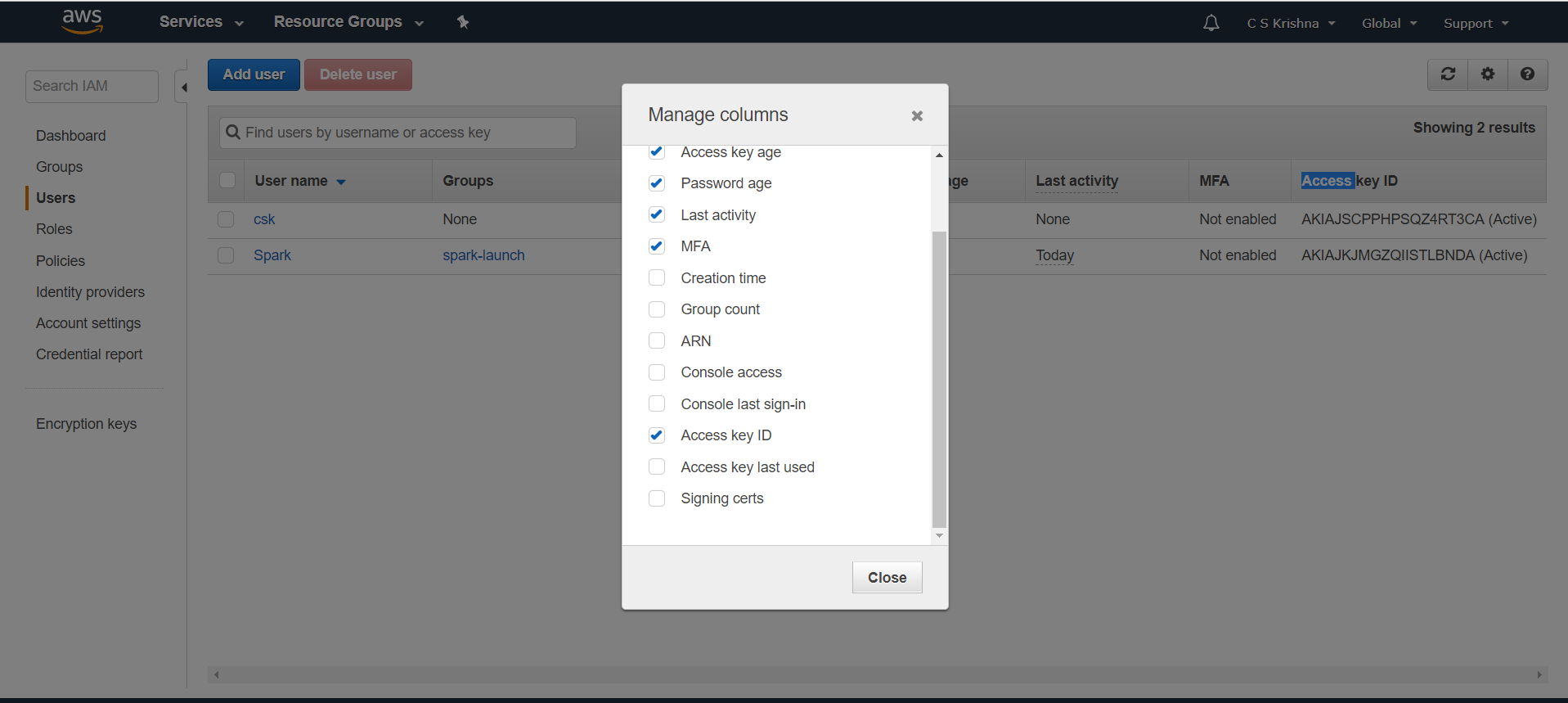
1. Generate Key/Pair in EC2 section of AWS Console. Click “Key Pairs” in left navigation and then Create Key Pair button.



1. Download the resulting key/pair PEM file.
2. **Convert** the key from **pem** format to **ppk** using **PuttyGen**.
3. Navigate to the link : <https://console.aws.amazon.com/iam/home?#/users>
4. In the navigation pane, choose Users.
5. Add the Access key ID column to the users table by completing the following steps:

* Above the table on the far right, choose the settings icon (Settings icon).
* In Manage Columns, select Access key ID.
* Choose Close to return to the list of users.

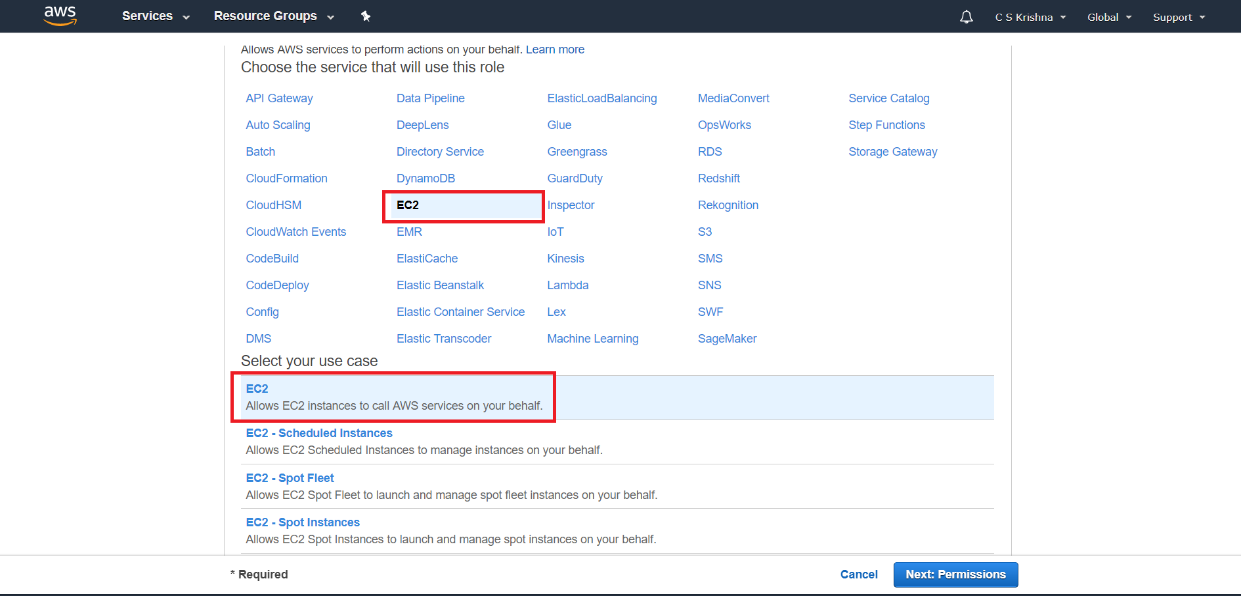
1. Create a User with any name let’s say 'Spark' and download its credentials file in csv format.



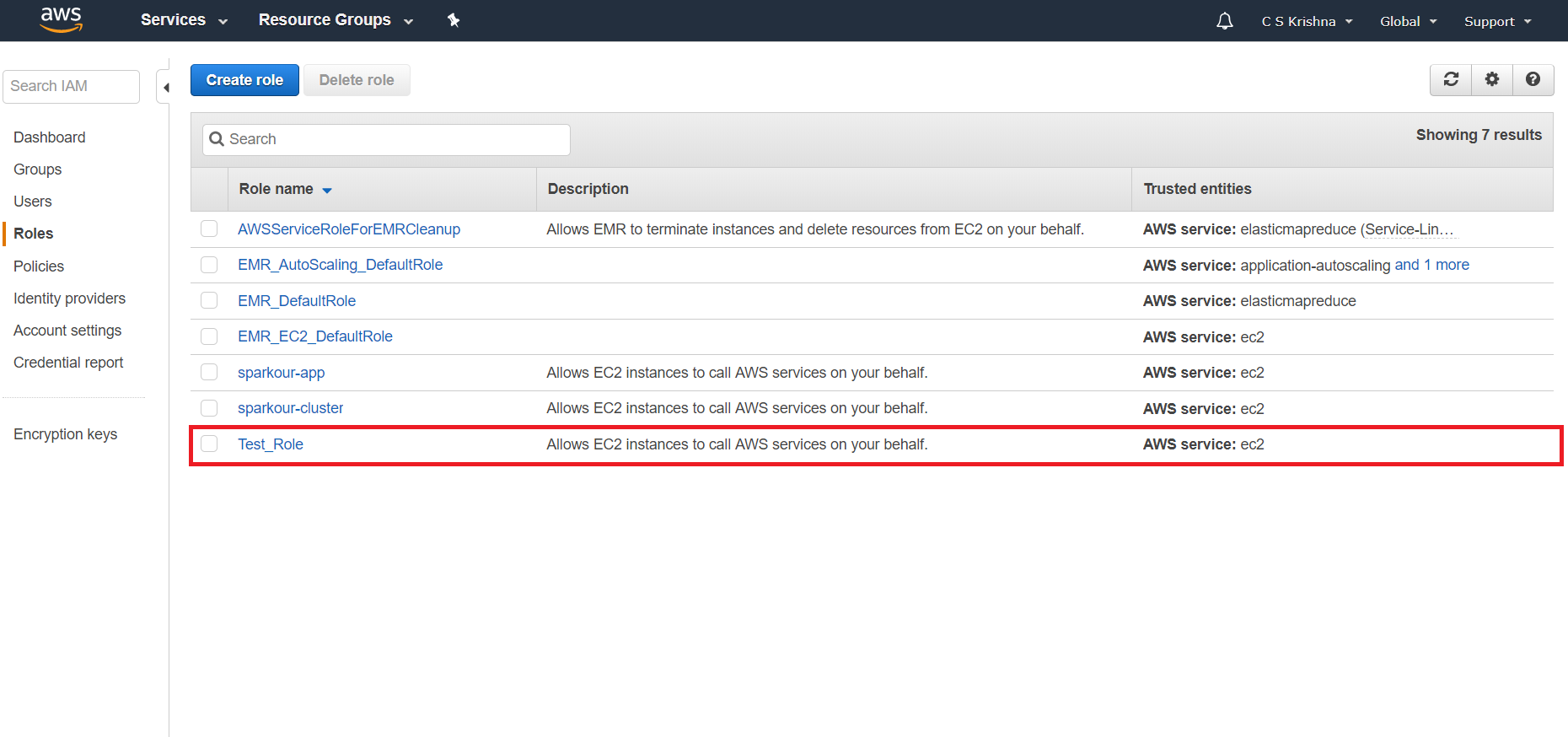
**Creating an IAM Role**

IAM Roles grant permissions for an instance to work with other AWS services (such as Simple Storage Service, or S3). It's a good practice to always assign an IAM Role to new instances, because a new Role cannot be assigned after the instance is launched.

1. **Go to the link :** <https://console.aws.amazon.com/iam>
2. **Navigate to Roles in the left side menu, and then select Create New Role at the top of the page. This starts a wizard workflow to create a new role.**
3. **Select RoleType, select Amazon EC2 to establish that this role will be applied to EC2 instances. Go to the Next Step.**

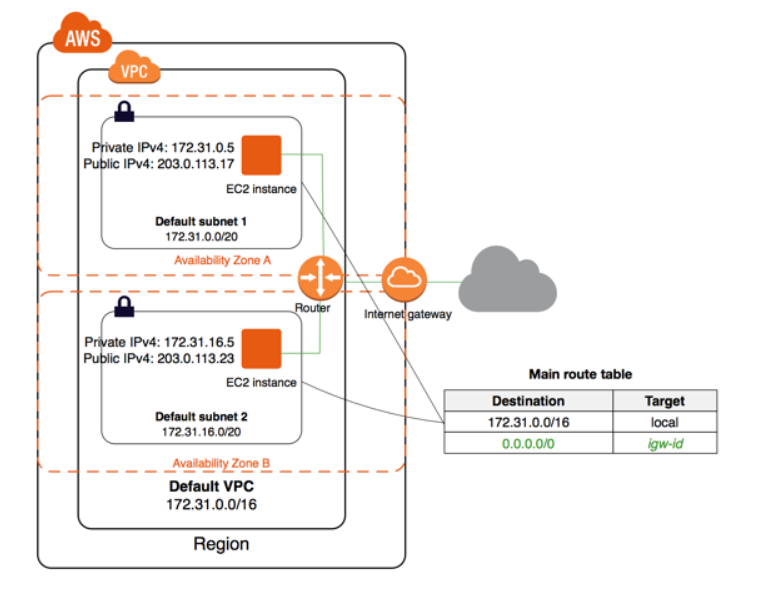
****

1. **Do not select any policies. (We will add policies in other recipes when we need our instance to access other services). Go to the Next Step.**
2. **Review, select Create Role. Enter the Role name and return to the Roles dashboard, and should see your new role listed on the dashboard.**

****

### **Creating VPC and Subnets**

A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch your AWS resources, such as Amazon EC2 instances, into your VPC. You can configure your VPC by modifying its IP address range, create subnets, and configure route tables, network gateways, and security settings.



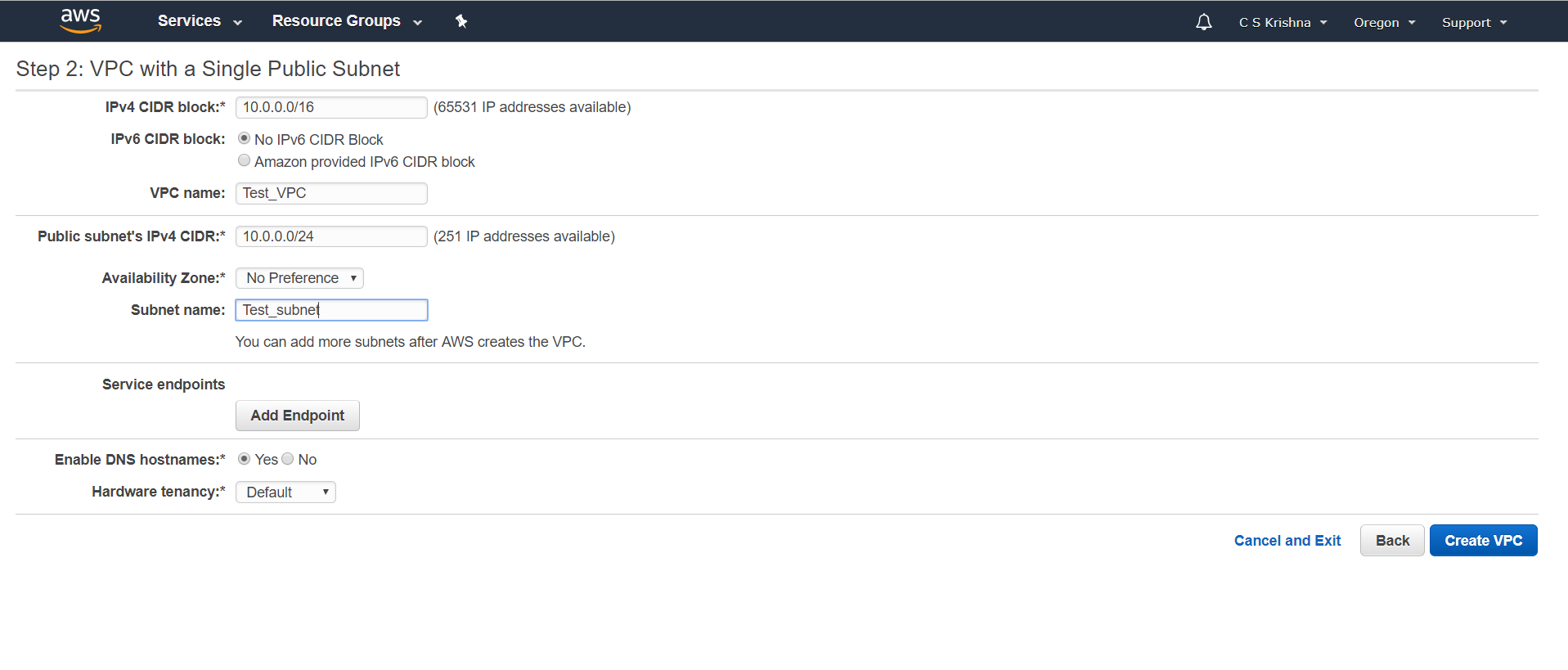
**VPC and Subnets in AWS**

A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet.

To protect the AWS resources in each subnet, you can use multiple layers of security, including security groups and network access control lists (ACL).

**To create a VPC using the Amazon VPC Wizard**

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose VPC dashboard, and then choose **Start VPC Wizard**.
3. Choose the first option, VPC with a Single Public Subnet, and then choose Select.
4. On the configuration page, enter a name for your VPC in the VPC name field; for example, my-vpc, and enter a name for your subnet in the Subnet name field. For this exercise, you can leave the rest of the configuration settings on the page, and choose Create VPC.

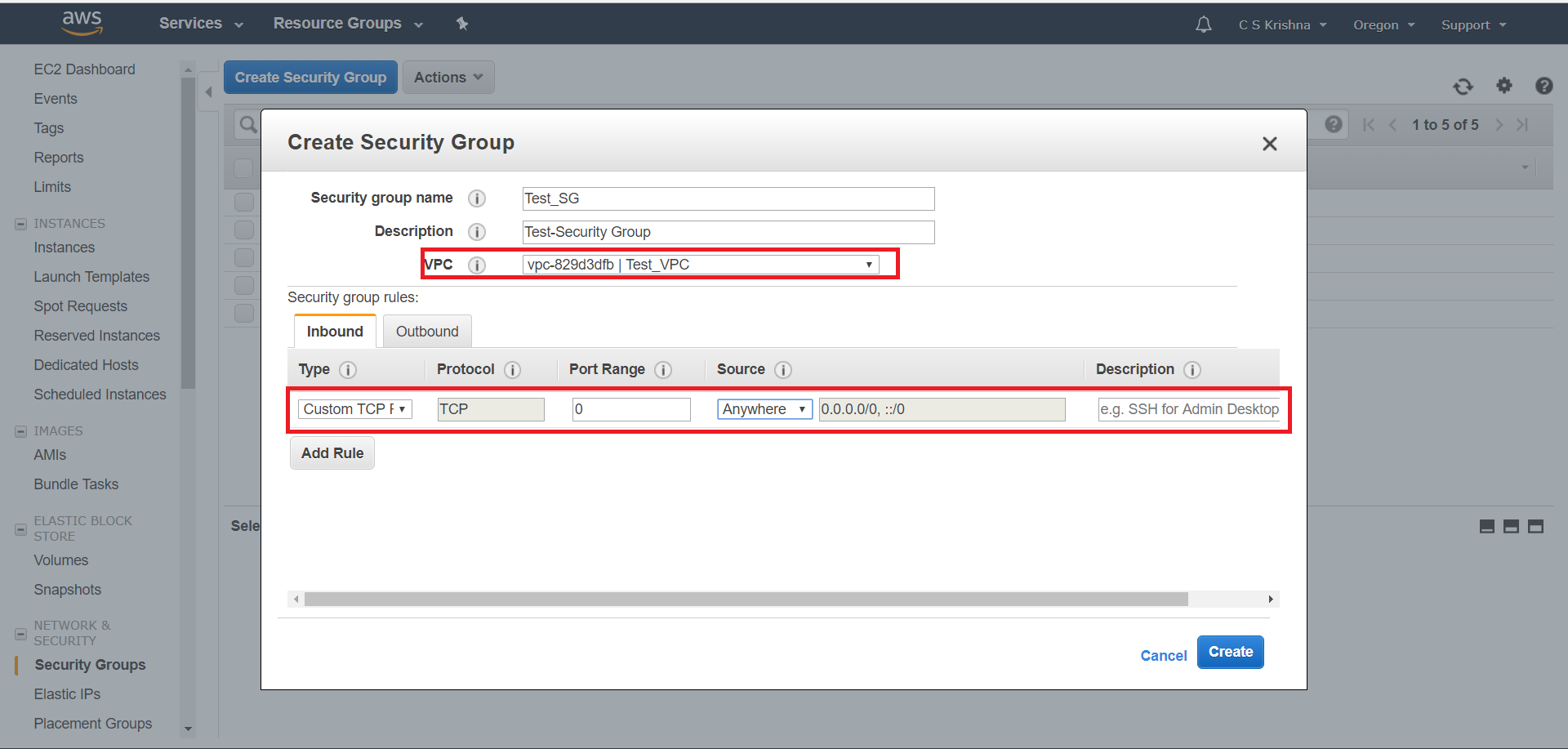


**\*\* *In subnet select, Auto assign IP to true***

### **Creating a Security Group**

**Next, we create a Security Group to protect our instance. Security Groups contain inbound and outbound rules to allow certain types of network traffic between our instance and the Internet.**

1. **Navigate to Network & Security: Security Groups and select Create Security Group. This opens a popup dialog to create the Security Group.**
2. **Set the Security group name to a value like sparkour-app-sg and set the Description to Security Group protecting Spark instance. Select the VPC create by you.**

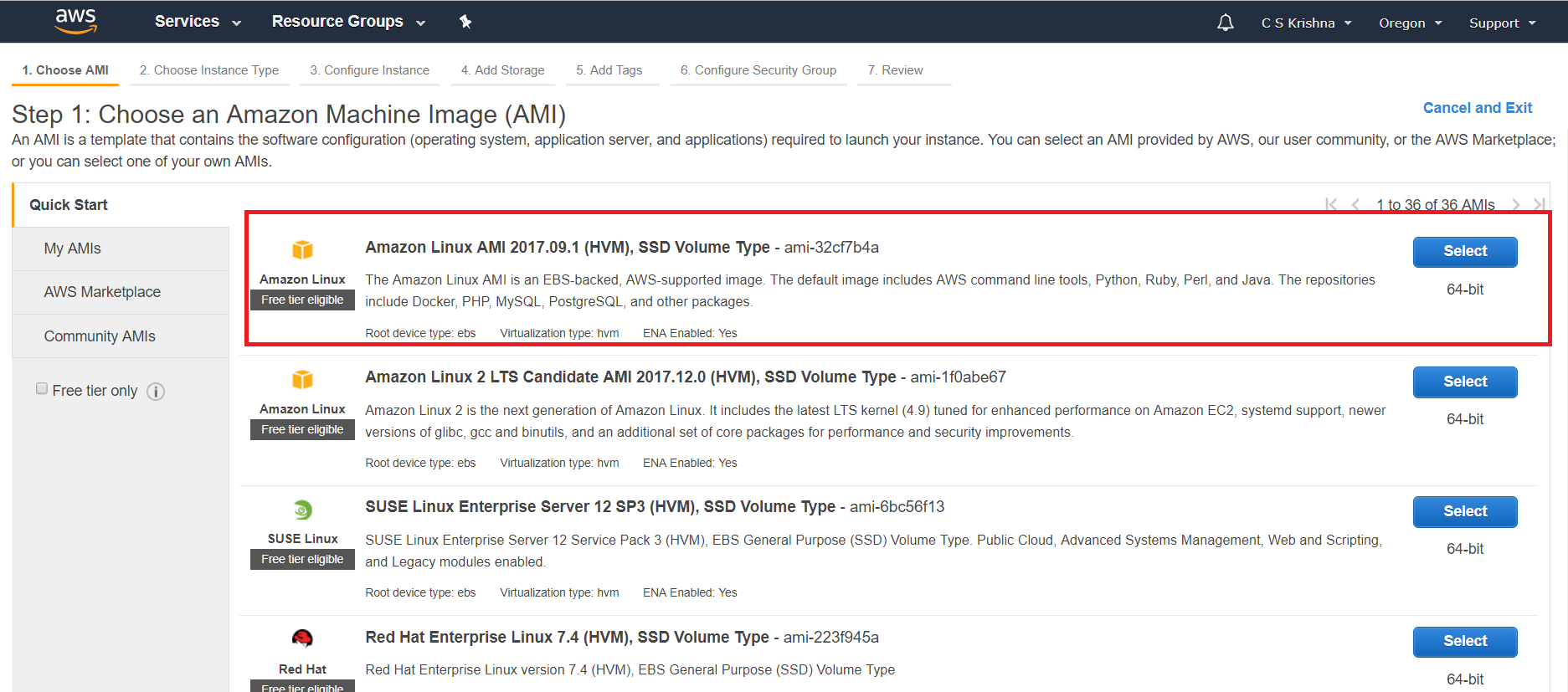
****

1. **Set the Inbound and the Outbound Rules and click on Create button.**

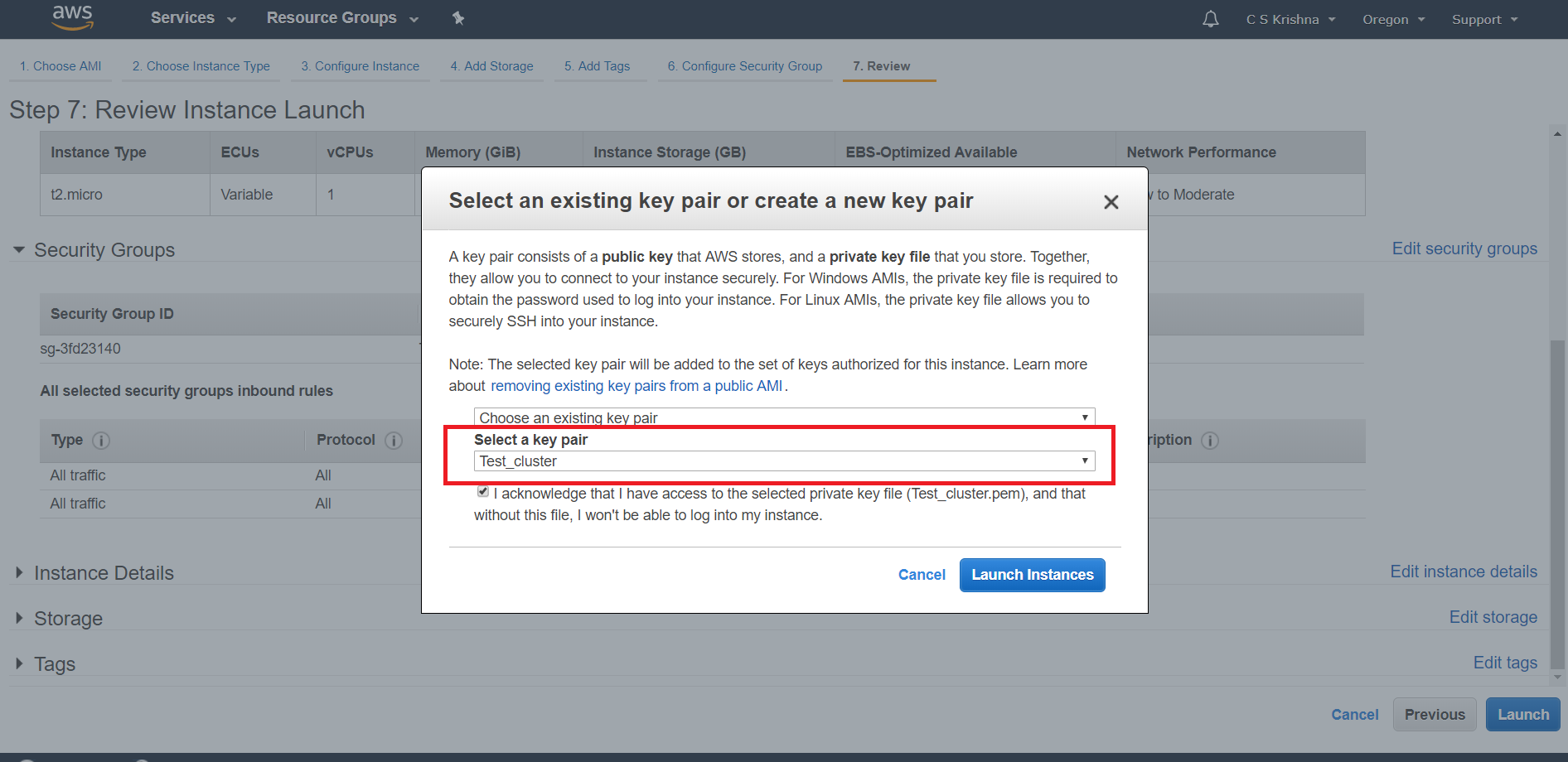
**Creating the EC2 Instance**

**We now have everything we need to create the instance itself. We select an appropriate instance type from the wide variety of available options tailored for different workloads and initialize it with Amazon's free distribution of Linux (delivered in the form of an Amazon Machine Image, or AMI).**

1. **Select the EC2 service from the AWS Management Console.**
2. **Navigate to Instances: Instances and select Launch Instance. This starts a wizard workflow to create the new instance.**
3. **On Step 1: Choose an Amazon Machine Image (AMI), Select the Amazon Linux AMI option. This image contains Amazon's free distribution of Linux (based on Red Hat) and comes preinstalled with useful extras like Java, Python, and the AWS command line tools.**

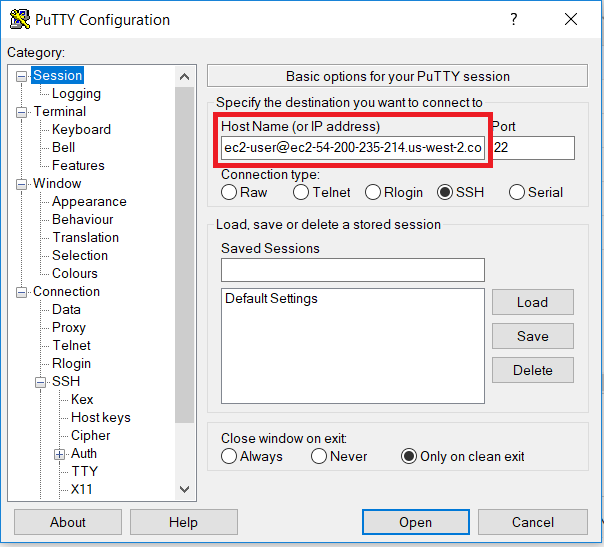
****

1. **On Step 2: Choose Instance Type, we will use t2.micro which is part of free tier with 1 core and 1 GB of memory.**
2. **Select Next: Configure Instance Details after you have made your selection.**
3. **Configure Instance Details, set IAM role to the IAM Role you created earlier. Change Auto-Assign Public IP to Enable. You can leave all of the other details with their default values. Select Next: Add Storage.**
4. **Add Storage, keep all of the default values. Elastic Block Store (EBS) is Amazon's network-attached storage solution, and the Amazon Linux AMI requires at least 8 GB of storage.**
5. **Configure Security Group, choose Select an existing security group and pick the Security Group you created earlier. Finally, select Review and Launch.**
6. **Select the key pair for the cluster, choose the one already created by you, in ‘pem’ format.**

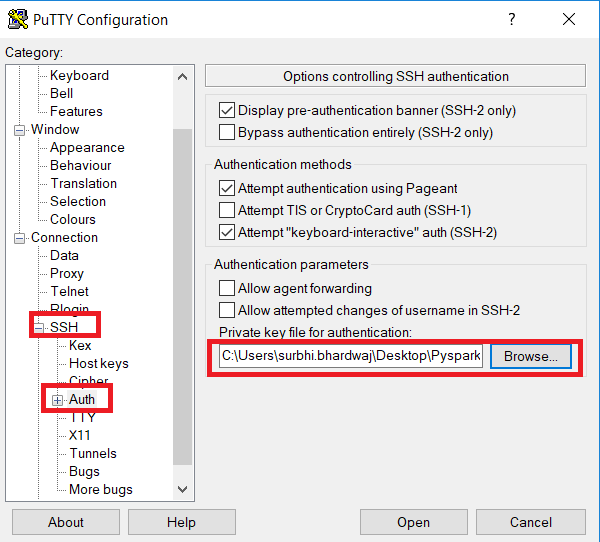
****

**Connecting to the EC2 Instance**

1. **While the EC2 instance is starting up, select it in the dashboard. Details about the instance appear in the lower pane.**
2. **Record the Public IP address of the instance. You use this to SSH into the instance, or access it via a web browser. The Public IP is not static, and changes every time you stop and restart the instance. Advanced users can attach an Elastic IP address to work around this minor annoyance whereas Elastic IP is chargeable.**
3. **Open putty and enter Public-DNS of the instance, in the hostname.**

****

1. **Select the SSH🡪 Auth and browse the Key for Authentication, created by you.**

****

1. **You will be connected as ec2-user**
2. **On your first login, you may see a welcome message notifying you of new security updates. You can apply these updates with the yum utility.**
3. **Do a ‘sudo yum update’ at first.**

**Installing Apache Spark**

**To install spark run the following commands on the console:**

*# Download Spark to the ec2-user's home directory*

cd ~

wget https://d3kbcqa49mib13.cloudfront.net/spark-2.2.0-bin-hadoop2.7.tgz

*# Unpack Spark in the /opt directory*

sudo tar zxvf spark-2.2.0-bin-hadoop2.7.tgz -C /opt

*# Create a symbolic link to make it easier to access*

sudo ln -fs spark-2.2.0-bin-hadoop2.7 /opt/spark

**To complete your installation, set the SPARK\_HOME environment variable so it takes effect when you login to the EC2 instance.**

*# Insert these lines into your ~/.bash\_profile:*

export SPARK\_HOME=/opt/spark

PATH=$PATH:$SPARK\_HOME/bin

export PATH

*# Then exit the text editor and return to the command line.*

You need to reload the environment variables (or logout and login again) so they take effect.

*# Reload environment variables*

source ~/.bash\_profile

*# Confirm that spark-submit is now in the PATH.*

spark-submit --version

*# (Should display a version number)*

Install Java 8 version as the current image, has Java 7 version by default.

*# Install Java 1.8*

sudo yum install java-1.8.0

*# Change the current Java version*

sudo alternatives --config java

*#Again run the command*

spark-submit --version

**Managing Cluster**

**This section describes how to automatically launch, start, stop, or destroy a Spark cluster running in Amazon Elastic Cloud Compute (EC2) using the spark-ec2 script. It steps through the pre-launch configuration, explains the script's most common parameters, and points out where specific parameter values can be found in the Amazon Web Services (AWS) Management Console.**

**Downloading the Script**

* **The spark-ec2 script was detached from the main Spark distribution in Spark 2.0.0 and needs to be downloaded separately. If you are using an older version of Spark, you will find the script in your Spark distribution at $SPARK\_HOME/ec2.**
* **Download the script to your launch environment and place the files into your Spark distribution.**

*# Download a ZIP of the script (you can also use Git if you prefer)*

cd ~

wget https://github.com/amplab/spark-ec2/archive/branch-2.0.zip

*# Unpack the script into your Spark distribution and set permissions*

sudo unzip branch-2.0.zip -d $SPARK\_HOME

mv $SPARK\_HOME/spark-ec2-branch-2.0 $SPARK\_HOME/ec2

sudo chown -R ec2-user:ec2-user $SPARK\_HOME/ec2

*#Confirm that script exists by checking the version.*

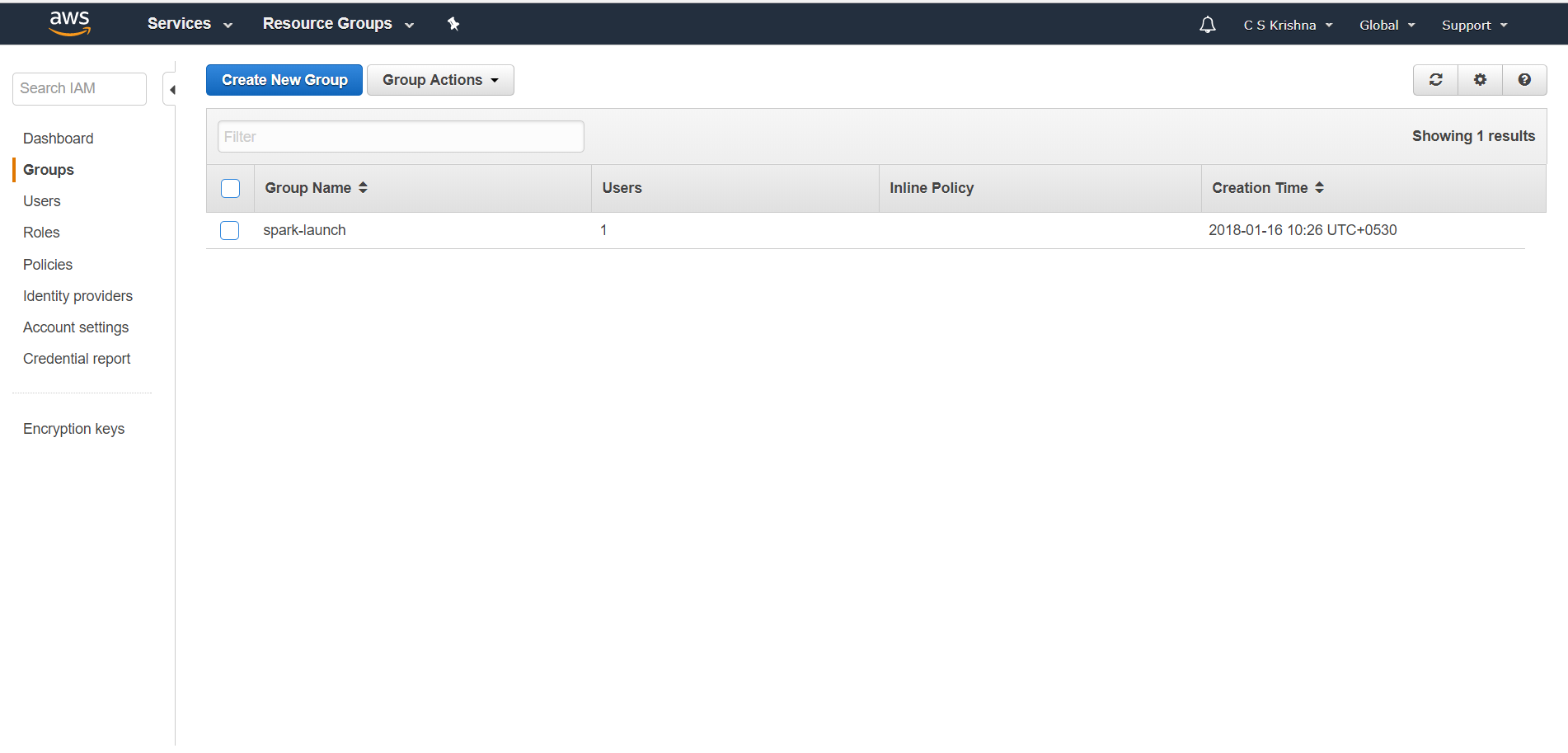
$SPARK\_HOME/ec2/spark-ec2 --version

*# Output should be "spark-ec2 2.0.0". "2.0.0" is the version of the script itself, not Spark's version.*

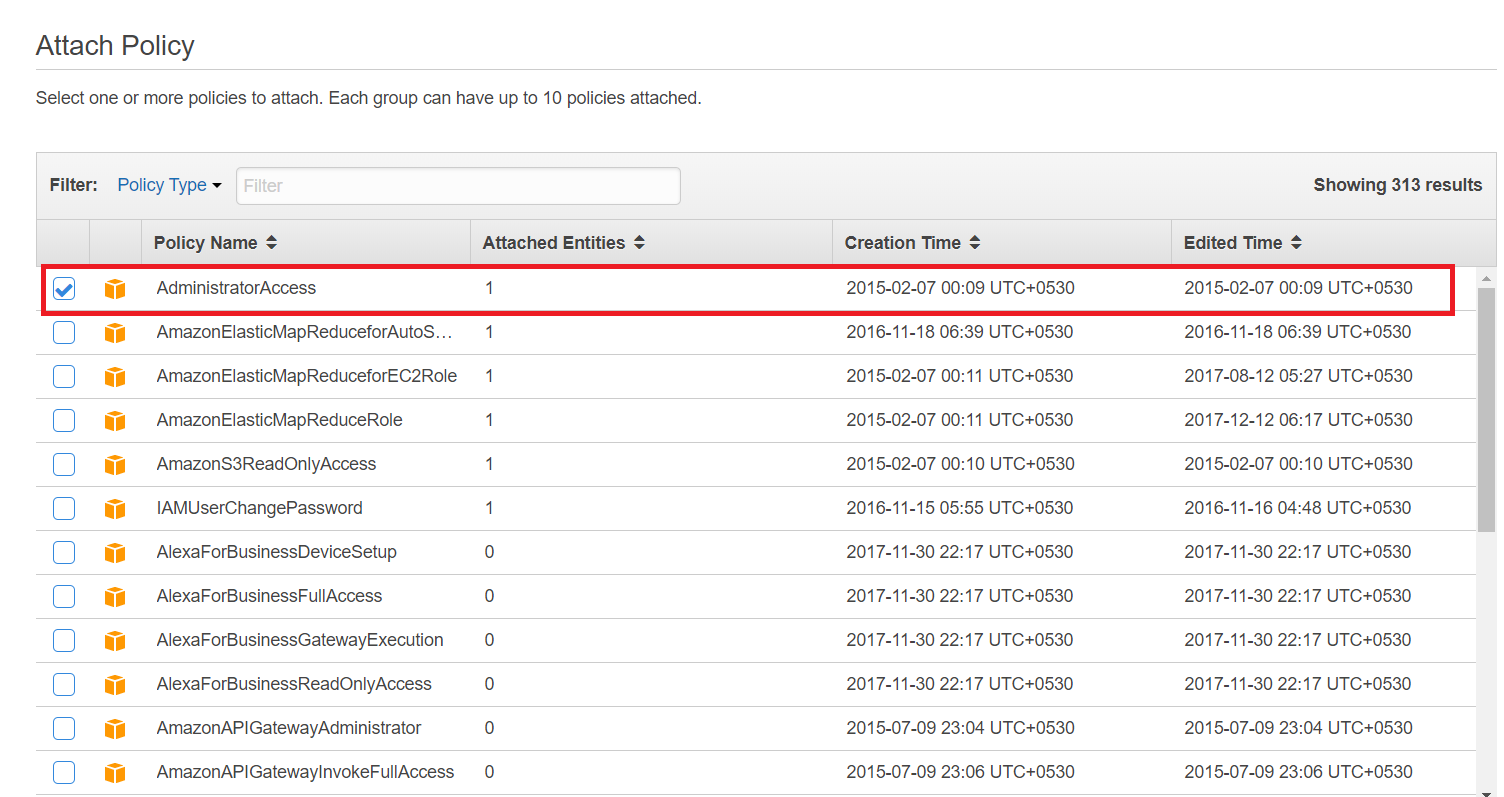
### **Adding Launch Permissions**

**The spark-ec2 script requires permission to execute multiple AWS Actions, such as creating new EC2 instances and configuring security groups. These permissions are checked against the AWS user account used to run the script, and can be attached to a user account through the IAM service.**

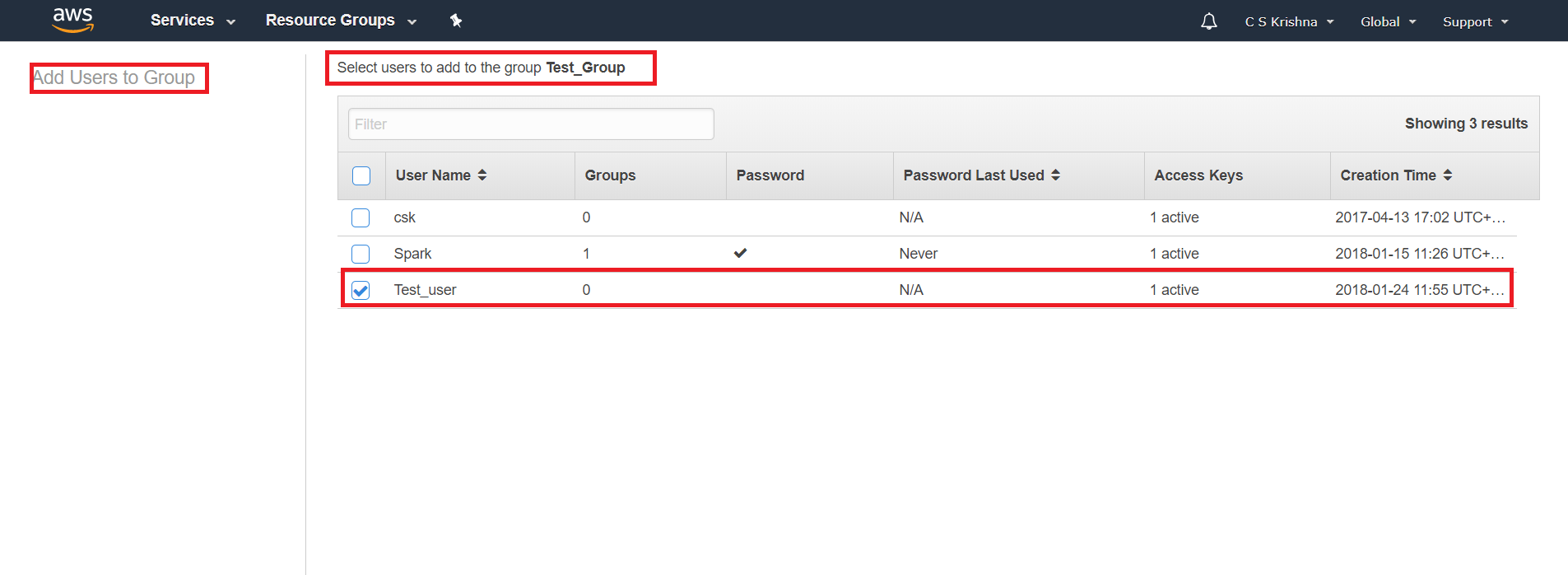
1. **Login to your AWS Management Console and select the Identity & Acces Management service.**
2. **Navigate to Groups in the left side menu, and then select Create New Group at the top of the page. This starts a wizard workflow to create a new Group.**

****

1. **On Step 1. Group Name, set the Group Name to a value like spark-launch and go to the Next Step.**
2. **On Step 2. Attach Policy, select AdministratorAccess to grant a pre-defined set of administrative permissions to this Group. Go to the Next Step.**

****

1. **On Step 3. Review, select Create Group. You return to the Groups dashboard, and should see your new Group listed on the dashboard.**
2. **Next, click on the name of your group to show summary details. Go to the Users tab and select Add Users to Group.**
3. **On the Add Users to Group page, select your IAM user account and then Add Users. You return to the Groups summary detail page, and should see your user account listed in the Group.**

****

**Your account can now be used to run the spark-ec2 script.**

**Gathering VPC Details**

**Next, you need to gather some information about the Virtual Private Cloud (VPC) where the cluster will be deployed.**

**1. Which AWS Region will I deploy into? You need the unique Region key for your selected region. [US West (Oregon) region]**

**2. Which VPC will I deploy into? You need the VPC ID of your VPC (the default is fine if you do not want to create a new one). From the VPC dashboard in the AWS Management Console, you can find this in the Your VPCs table.**

**Script Parameters**

**The spark-ec2 script exposes a variety of configuration options. The most commonly used options are described below, and there are other options available for advanced users.**

#########VPC Options

--region: Optionally set the key of the region where the cluster will live (defaults to US East).

--vpc-id: The ID of the VPC where the instances will be launched.

--subnet-id: Optionally set the ID of the Subnet within the VPC where the instances will be launched.

--zone: Optionally set the ID of an Availability Zone(s) to launch in (defaults to a randomly chosen zone). You can also set to all for higher availability at greater cost.

--authorized-address: The whitelisted IP range used in the generated Security Group (defaults to the public Internet as 0.0.0.0/0, so you should always specify a value for better security). [ For now we will use default only]

--private-ips: Optionally set to True for VPCs in which public IP addresses are not automatically assigned to instances. The script tries to connect to the cluster to configure it through Public DNS Names by default, and setting this to True forces the use of Private IPs instead (defaults to False). [Use False]

*#######Cluster Options*

--slaves: Optionally set the number of slaves to launch (defaults to 1).

--instance-type: Optionally set the EC2 instance type from a wide variety of options based on your budget and workload needs (defaults to m1.large, which is a deprecated legacy type).

[t2.micro]

--master-instance-type: Optionally set a different instance type for the master (defaults to the overall instance type for the cluster). [t2.micro]

--hadoop-major-version: Optionally set to 1 for Hadoop 1.0.4, 2 for CDH 4.0.2, or yarn for Hadoop 2.4.0 (defaults to 1). No other versions are currently supported [yarn by deafult]

--instance-profile-name: Optionally set to the unique name of an IAM Role to assign to each instance. [sparkour-cluster]

**Upload the pem key file to the current node using WinScp and change the permission of key to 400.**

**Run the following commands to launch the cluster :**

export AWS\_SECRET\_ACCESS\_KEY=d6+EEmci3KdXELF8UDOTxhqG202zttQ8vP7igot7

export AWS\_ACCESS\_KEY\_ID=AKIAJAYKK72QAP66NH3A

$SPARK\_HOME/ec2/spark-ec2 \

*--key-pair=Test\_cluster \*

*--identity-file=/home/ec2-user/Test\_cluster.pem \*

*--region=us-west-2 \*

*--vpc-id=vpc-829d3dfb \*

*--slaves=2 \*

*--instance-type=t2.micro \*

*--spark-version=2.0.0 \*

*--hadoop-major-version=yarn \*

launch sparktest-cluster

**To ssh to other nodes use :**

*######To ssh to some other node :*

ssh -i /opt/spark/PySpark\_Cluster.pem ec2-54-212-222-145.us-west-2.compute.amazonaws.com

**To open the pyspark shell use below command on master node:**

/root/spark/bin/pyspark --master spark://ec2-54-218-178-139.us-west-2.compute.amazonaws.com:7077

**To start the cluster, run these commands (from your launch environment, not the master):**

*### Start the Cluster*

$SPARK\_HOME/ec2/spark-ec2 \

--key-pair=Test\_cluster \

--identity-file=/home/ec2-user/Test\_cluster.pem \

--region=us-west-2 \

--vpc-id=vpc-829d3dfb \

--slaves=2 \

--instance-type=t2.micro \

--spark-version=2.0.0 \

--hadoop-major-version=yarn \

start sparktest-cluster

**For stopping and terminating the cluster use below :**

*### To stop the cluster*

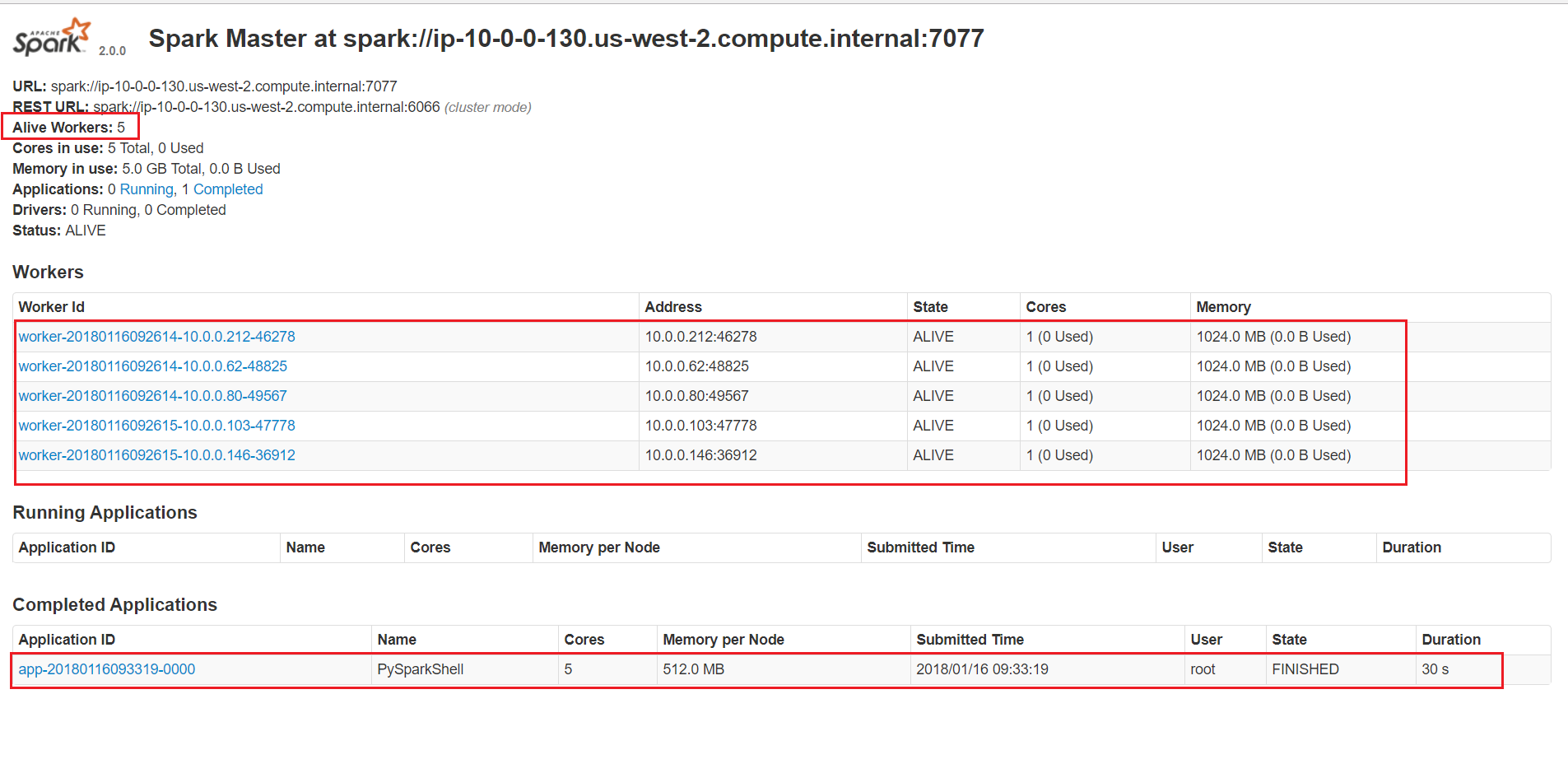
$SPARK\_HOME/ec2/spark-ec2 stop sparktest-cluster

*# (Hit 'y' to confirm)*

*### To terminate the cluster*

$SPARK\_HOME/ec2/spark-ec2 destroy sparktest-cluster

**Access the Spark web UI on port 8080 and ganglia on 5080.**

****

**Installing Anaconda/Jupyter**

**Installing Jupyter Notebook and its dependencies can be easily achieved by installing the Anaconda Python distribution. This will also include many important libraries such as numpy, scipy, matplotlib, pandas, etc. The installations are done on Spark master :**

*## Downloading Anaconda #########*

wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86\_64.sh

*## Extracting the package##*

bash Anaconda3–4.1.1-Linux-x86\_64.sh

*##set the path in ~/.bashrc file*

export PATH=$PATH:/root/anaconda3/bin

*## Load the variables*

source ~/.bashrc

*## Configure the Jupyter notebook*

jupyter notebook --generate-config

**Create certifications for the jupyter :**

*## Create Certifications*

mkdir certs

$ cd certs

$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem

**Edit the jupyter configuration file :**

*## Change the directory to :*

cd ~/.jupyter/

vi jupyter\_notebook\_config.py

**Enter the text below at the top of config file.**

c = get\_config()

*# Notebook config this is where you saved your pem cert*

c.NotebookApp.certfile = u'/root/certs/mycert.pem'

*# Run on all IP addresses of your instance*

c.NotebookApp.ip = '\*'

*# Don't open browser by default*

c.NotebookApp.open\_browser = False

*# Fix port to 8888*

c.NotebookApp.port = 8888

**Check that jupyter notebook is running using command :**

jupyter notebook jupyter

**And open the UI on master node with 8888 port.**

**Install py4j on the master node.**

**To get Spark Context in Jupyter notebook, run the following command :**

import os

import sys

os.environ['SPARK\_HOME'] = "/root/spark/"

sys.path.append("/root/spark/python/")

from pyspark import SparkContext

sc = SparkContext()

*## Set the following in ~/.bashrc file*

export PATH=/root/anaconda3/bin:/root/spark:$PATH

export PYSPARK\_PYTHON=/root/anaconda3/bin/python

**Accessing Data from S3 Bucket**

**Amazon S3 is cloud storage for the Internet. To upload your data (photos, videos, documents etc.), you first create a bucket in one of the AWS Regions. You can then upload any number of objects to the bucket.**

**Create an S3 bucket using Create Bucket wizard under EC2 services in AWS. S3 bucket takes a unique name. Data can be uploaded using Upload wizard. Following is the script to access data placed in S3 bucket using pyspark.**

sc.\_jsc.hadoopConfiguration().set("fs.s3n.awsAccessKeyId", "AKIAJKJMGZQIISTLBNDA")

sc.\_jsc.hadoopConfiguration().set("fs.s3n.awsSecretAccessKey", "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")

*### Reading the Data in bucket*

x = sc.textFile("s3n://esters/\*")

*## Applying count operation*

x.count()

**References**

1. <https://sparkour.urizone.net/recipes/installing-ec2/>
2. <https://sparkour.urizone.net/recipes/spark-ec2/>
3. <https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297>
4. <http://www.sparktutorials.net/spark-clusters-on-aws-ec2---reading-and-writing-s3-data---predicting-flight-delays-with-spark-part-1>
5. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
6. <https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html>