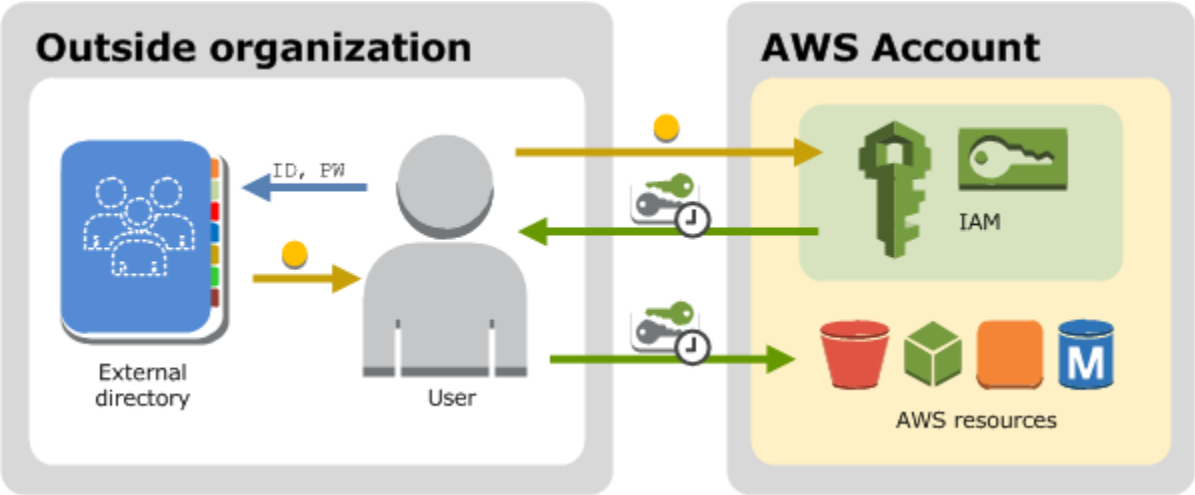
AWS Lab Work - IAM Roles

*Prepared by Sriganesh Pera

Intro to IAM [Identity Access Management]

IAM is a web service that lets users' access AWS features in a secured way. Basic illustration is given in the diagram



IAM gives you the following features:

1. **Shared Access**: **You can grant permissions to other users to use AWS without sharing password.**

2. **Granular permissions:** **You can grant different permissions to different people for using the AWS resources (In our group some can only use EC2and some others can only use Amazon S3).**

3. **Secure access to AWS resources for applications that run on Amazon EC2:** **Secure access can be given to applications that run on EC2 and use other resources like Amazon S3 buckets or Dynamo DB.**

4. **Identity federation:** You can allow users who already have passwords elsewhere and give them temporary access.

5. Identity information for assurance : If you use AWS CloudTrail, you receive log records that include information about those who made requests for resources in your account. That information is based on IAM identities.

IAM can be accessed in 4 ways:

1. **AWS Management Console:** Web interface in AWS homepage when you logged in.

2. **AWS Command Line Tools:** Jus CLI

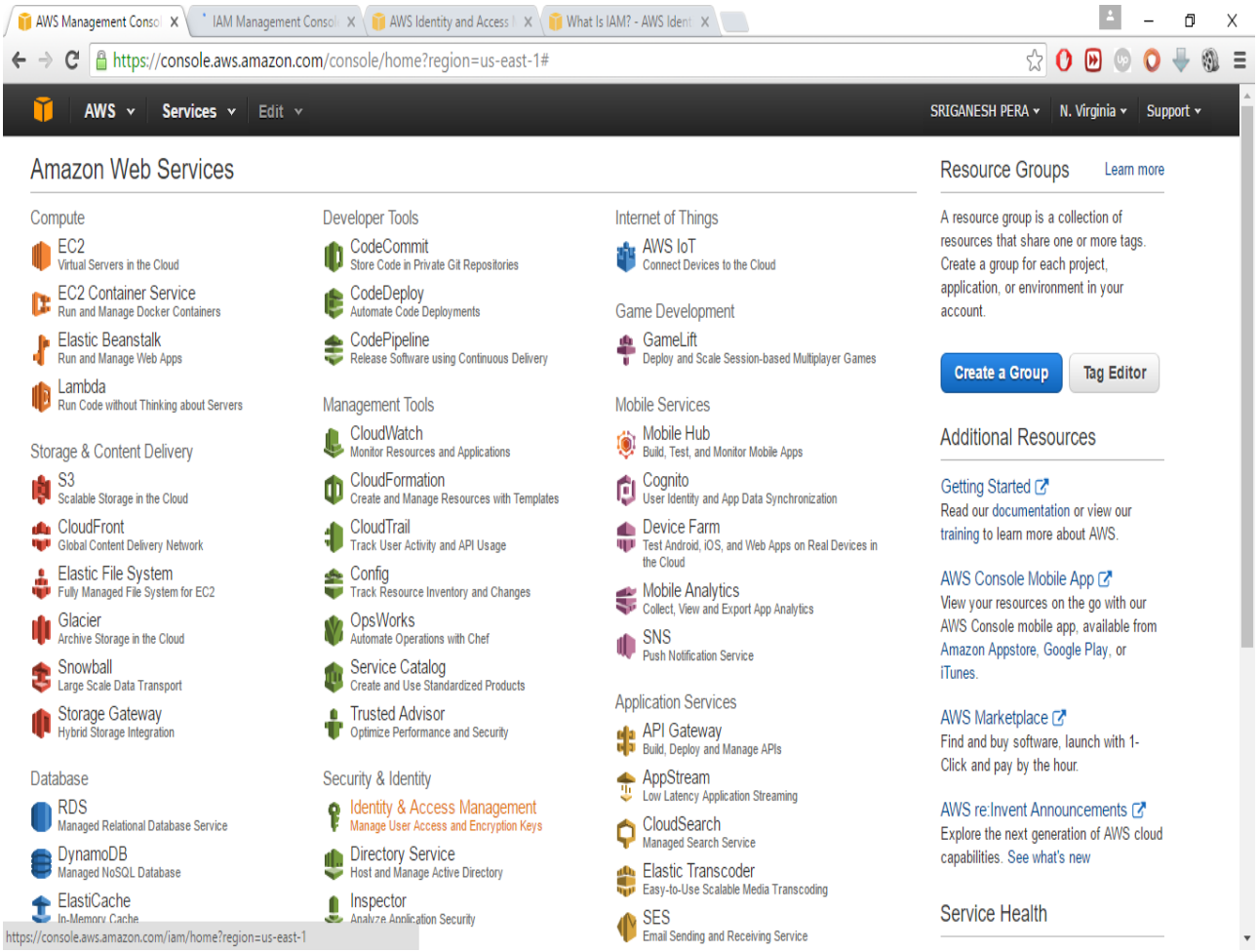
3. **AWS SDKs:** AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, etc.). The SDKs provide a convenient way to create programmatic access to IAM and AWS.

4. **IAM HTTPS API:** You can access IAM and AWS programmatically by using the IAM HTTPS API, which lets you issue HTTPS requests directly to the service. When you use the HTTPS API, you must include code to digitally sign requests using your credentials

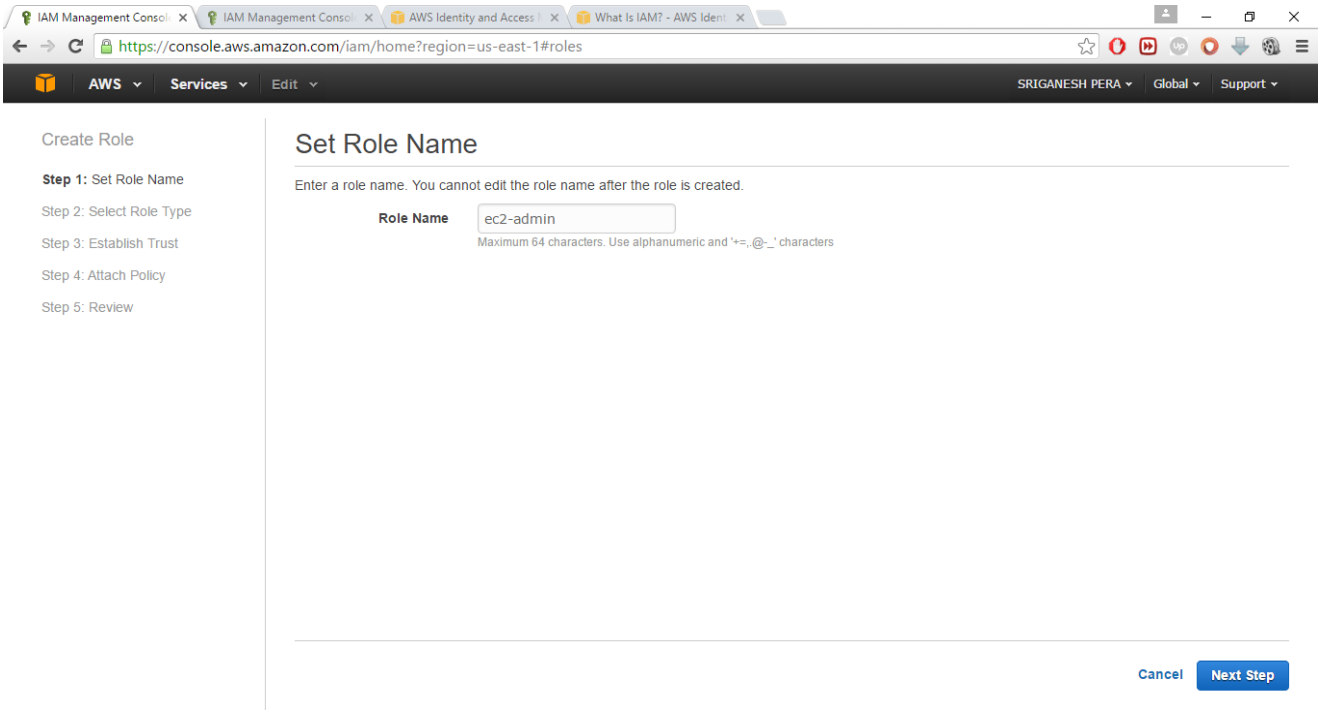
Lab 1 Objective: Accessing S3 bucket on AWS using Python SDK set up on an EC2 machine with IAM roles

Lab 1: Setting up an EC2 Instance with IAM Roles and the Python Boto3 SDK

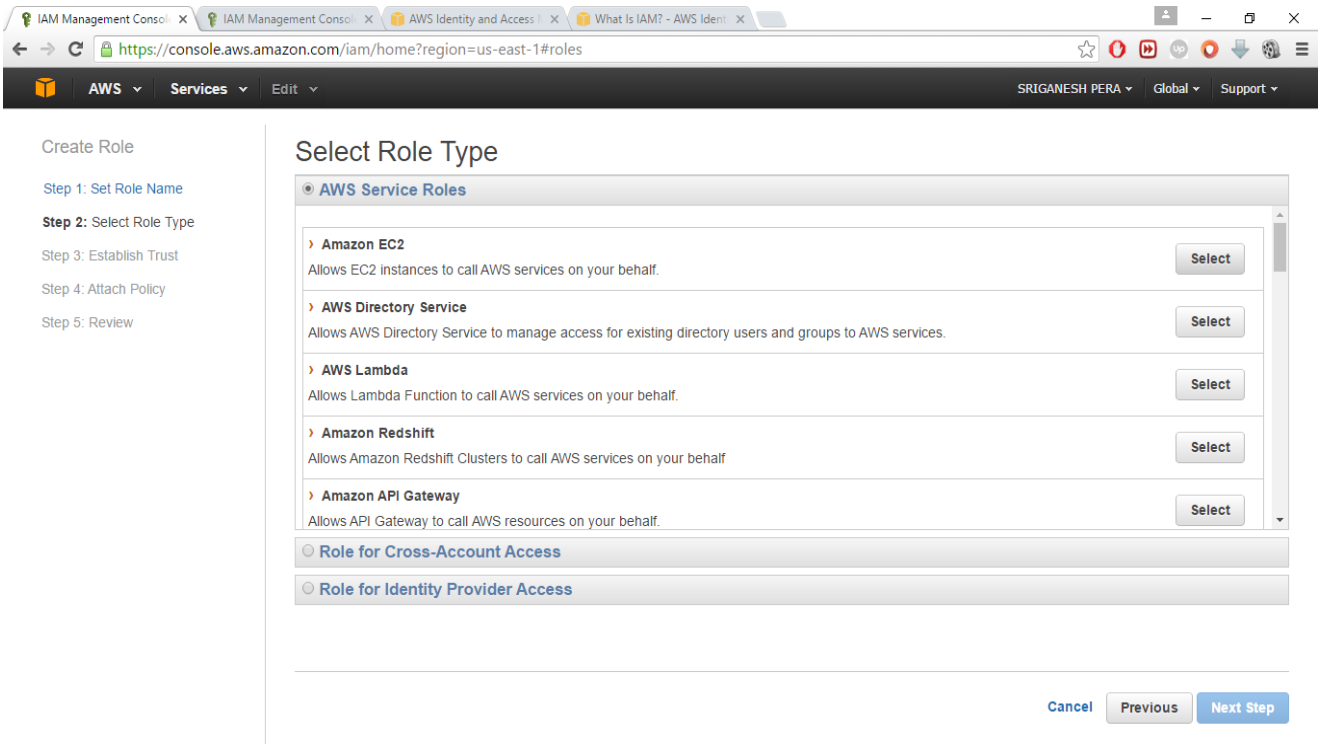
1. Click on IAM on AWS console



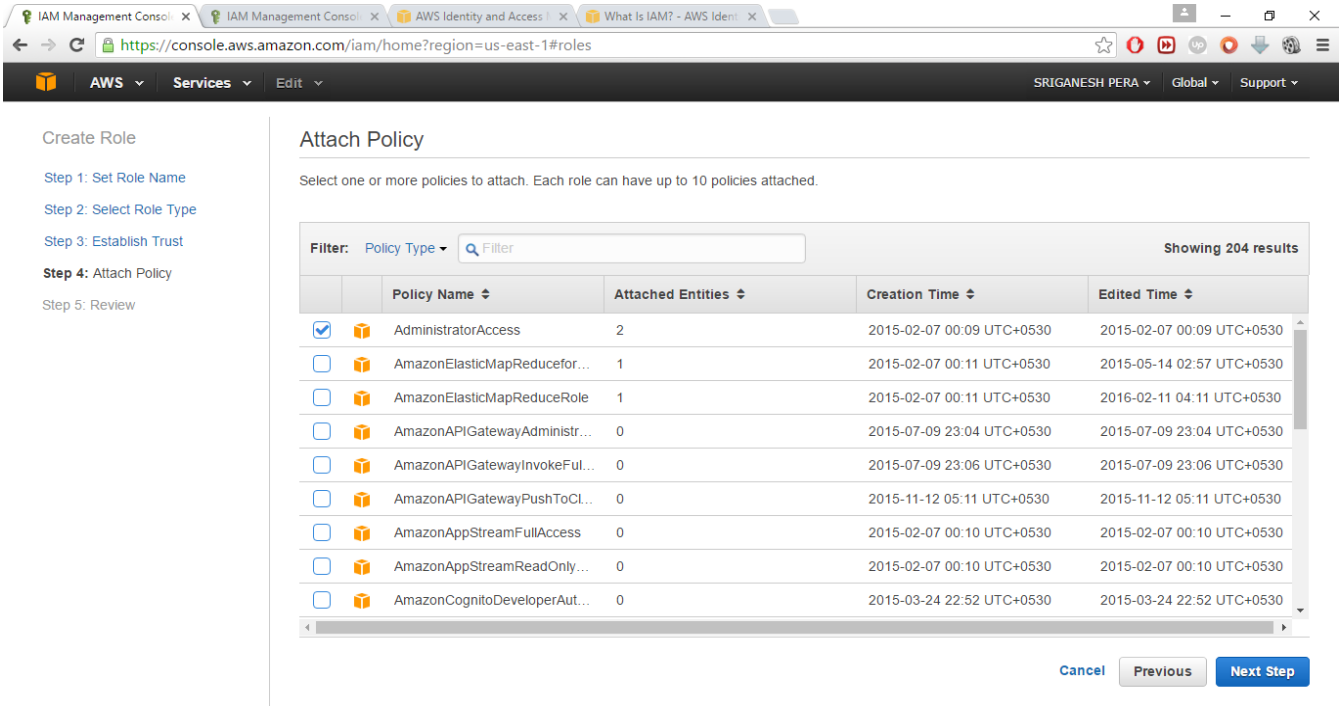
2. Then Click on Role -> Create Role (blue button) -> Name it as ‘ec2-admin’ and click next step



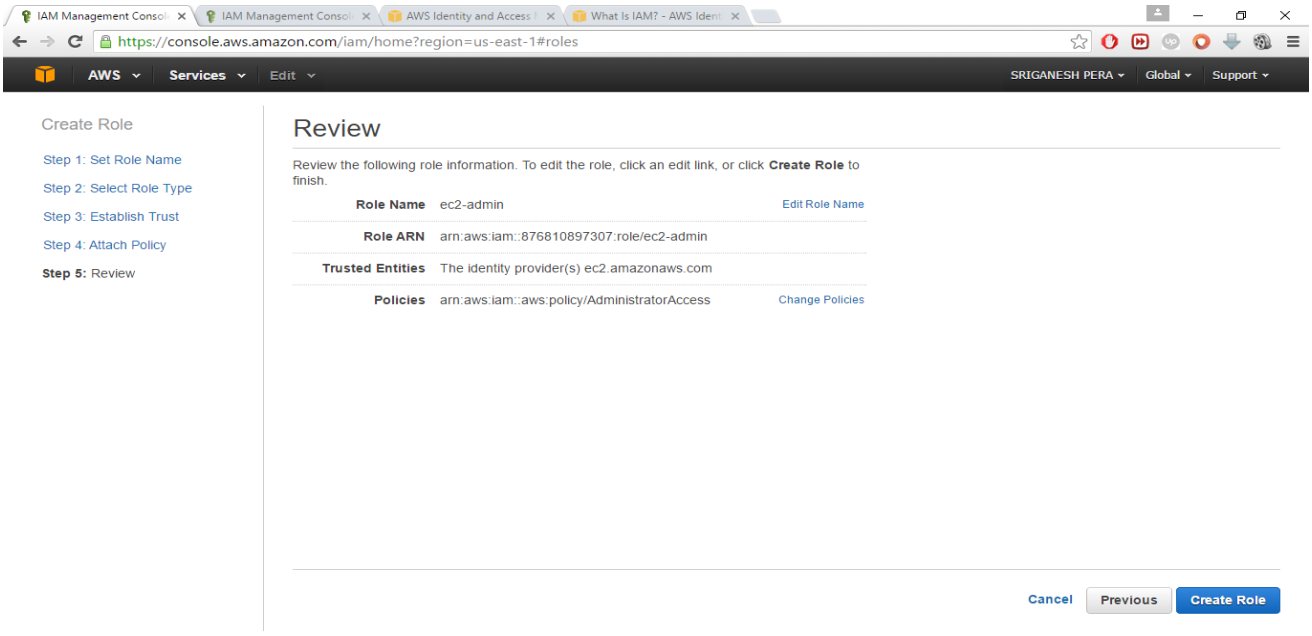
3. Select role type as Amazon EC2



4. Select Admin Access (so that EC2 instance that we'll set up will have access to all AWS services) and click next step

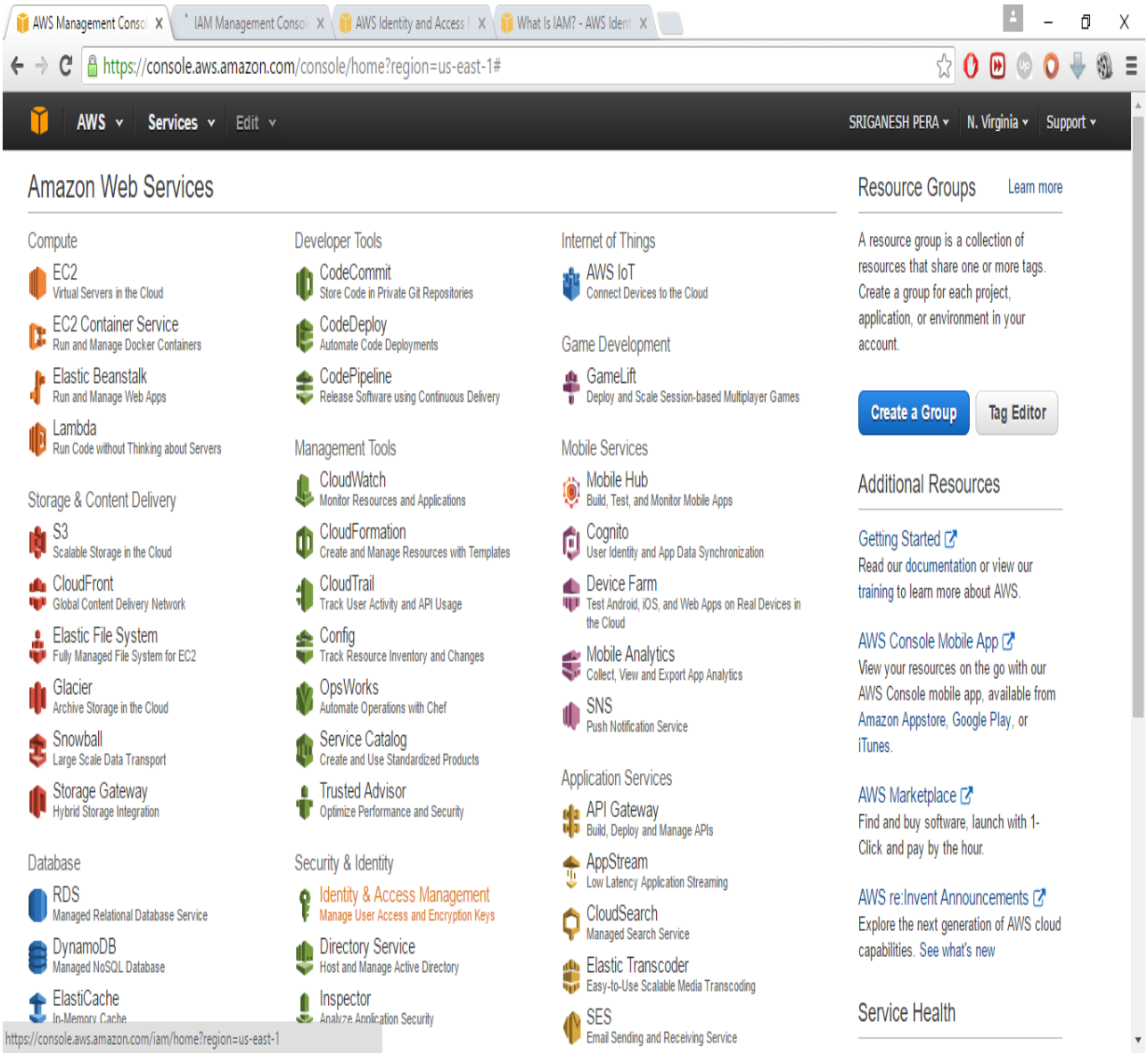


5. You can review the role we just created and click on Create Role

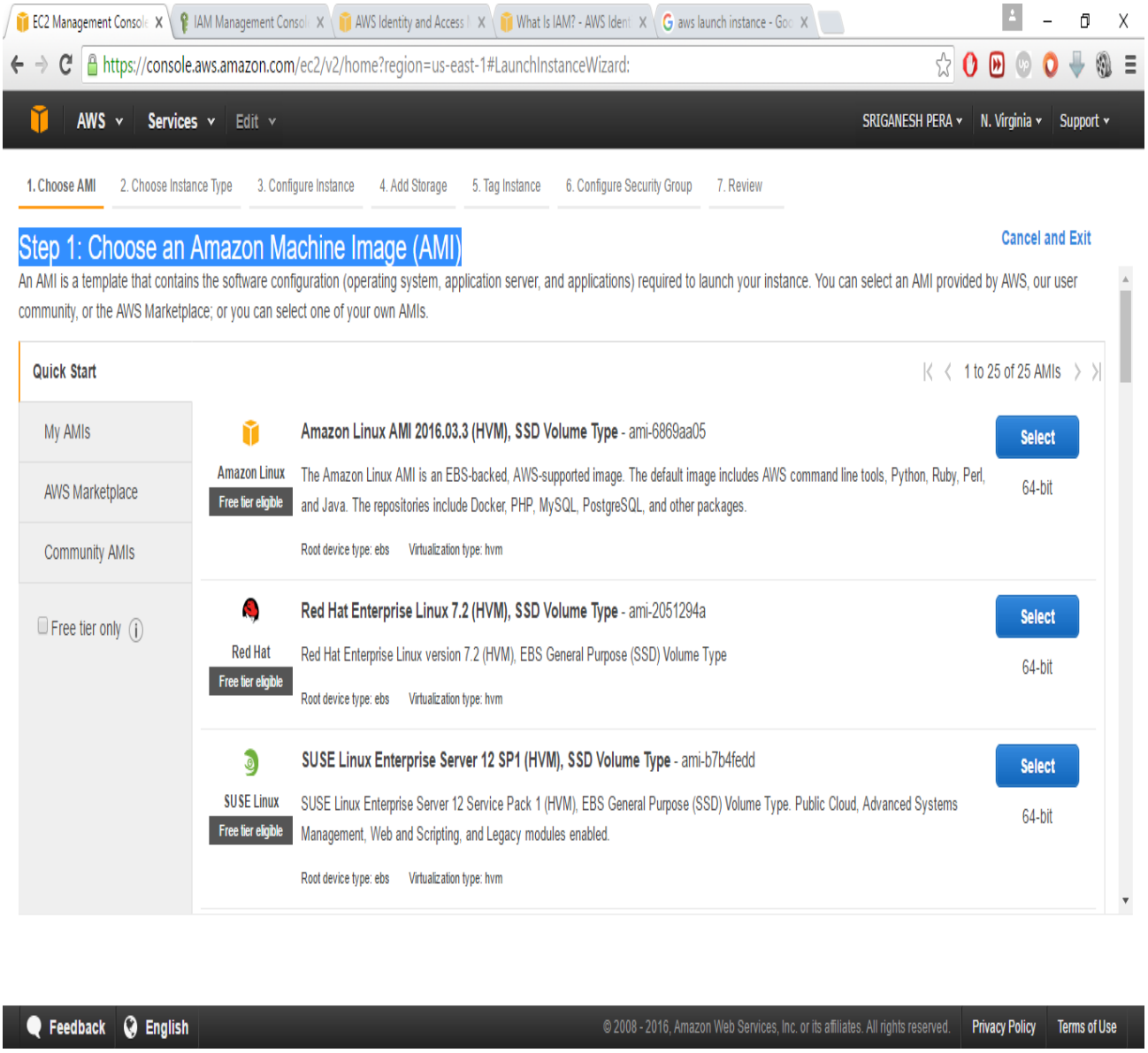


Now launch EC2 instance and see how this 'ec2-admin' role gives us access in using all AWS resources

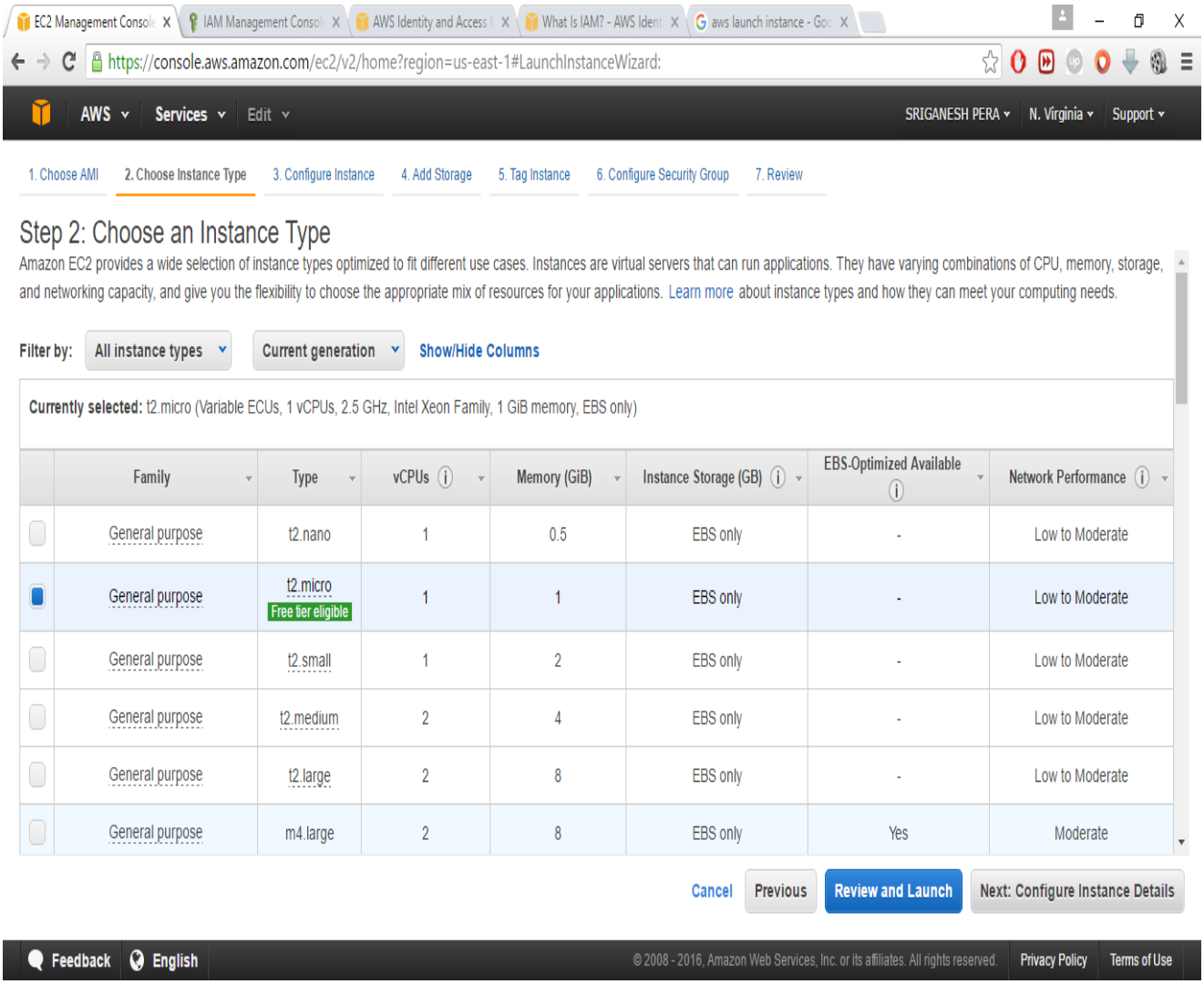
6.Go to EC2 on the AWS console and click EC2



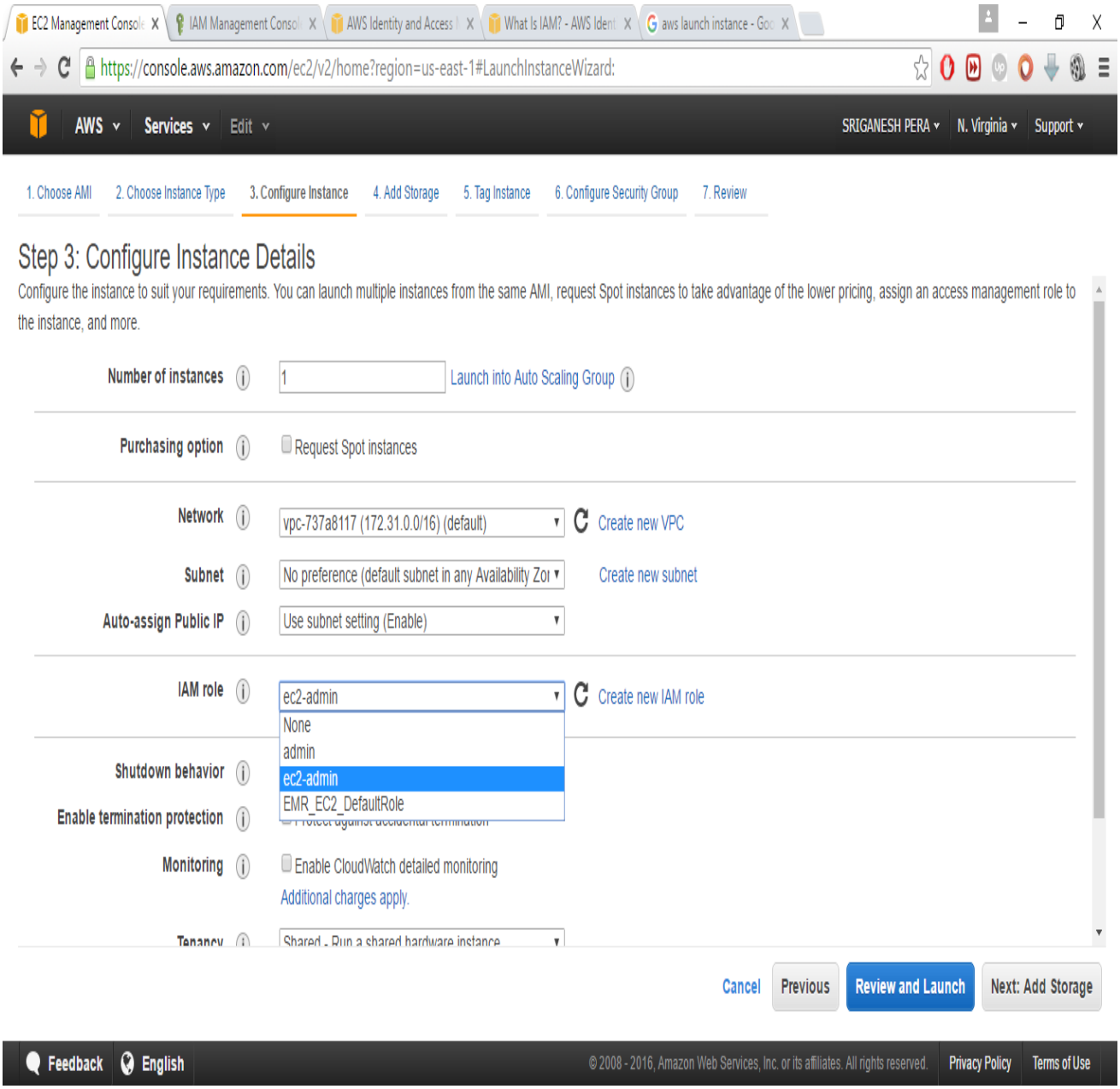
7.Click on Launch instance which takes you to Step 1: Choose an Amazon Machine Image (AMI).Now click on Amazon Linux Image(the very first block)



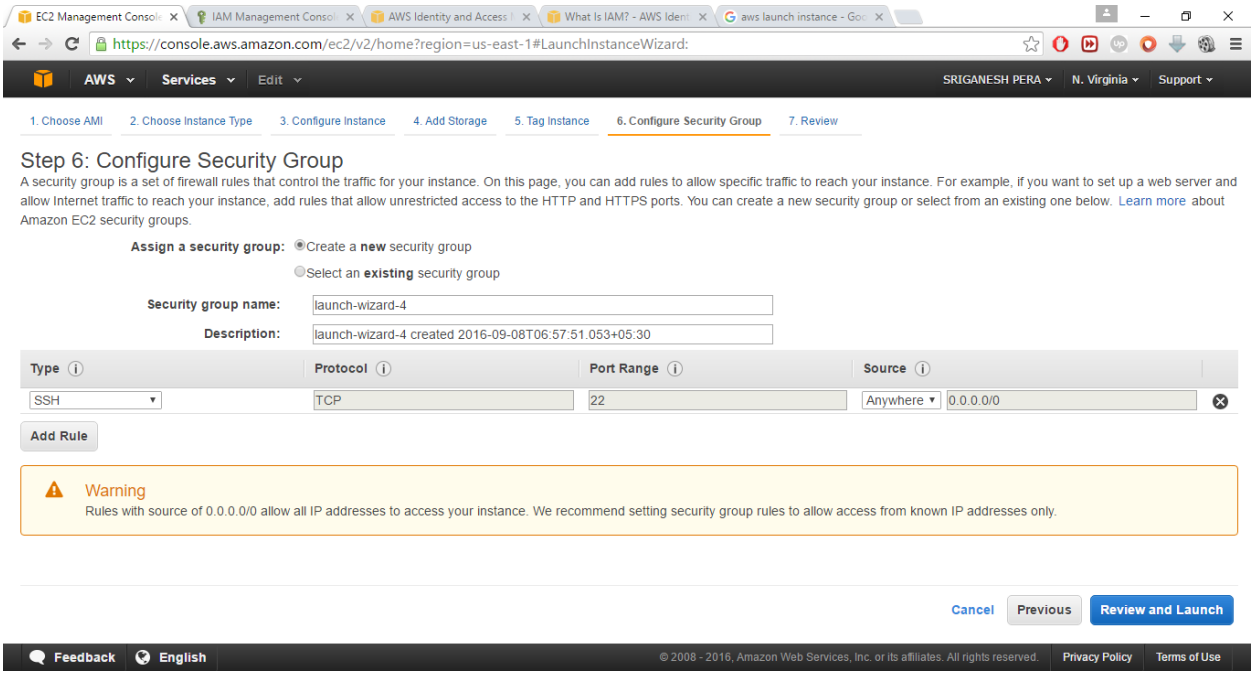
8. It takes you to Choose instance type. Now click on Configure instance



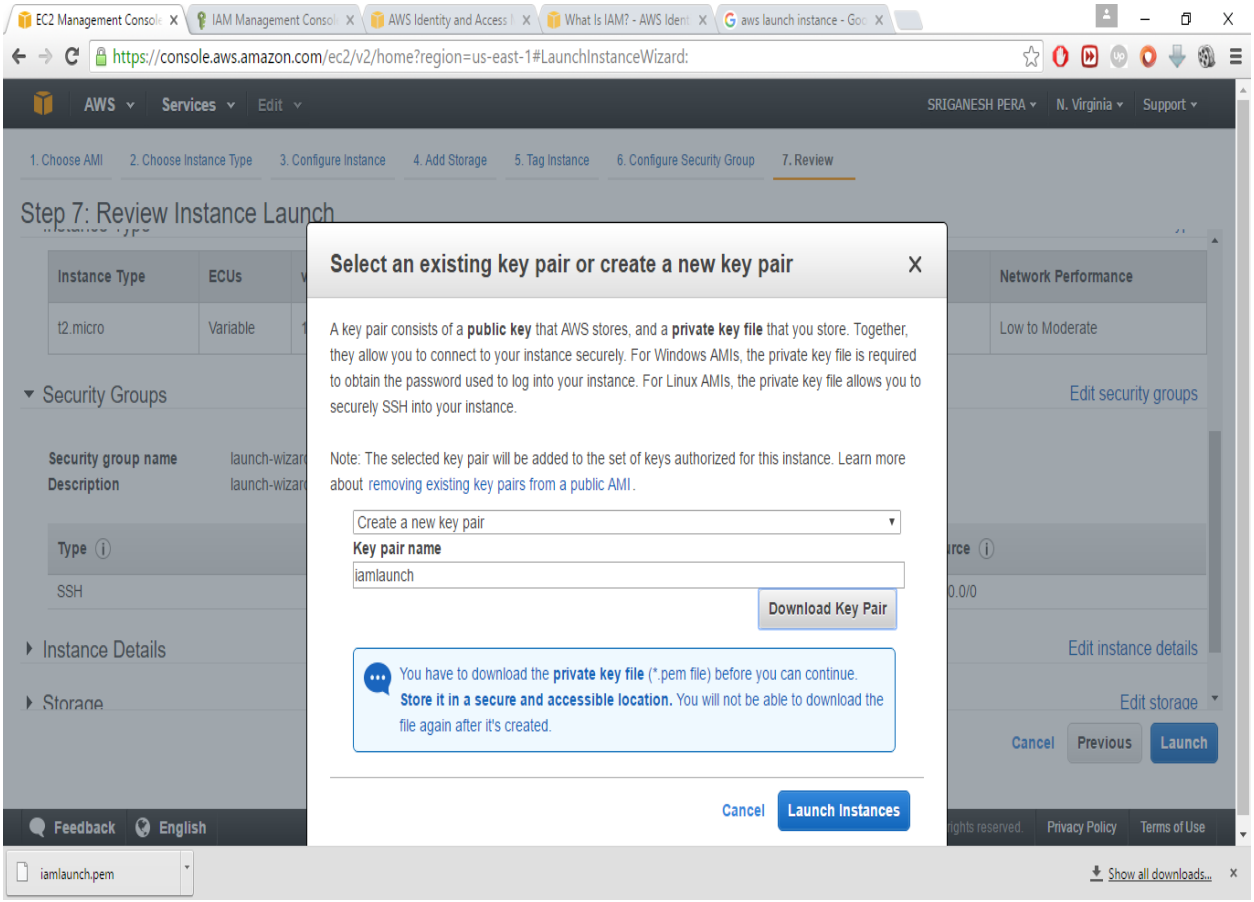
9. Select IAM role as ec2-admin that we have created in the first step:



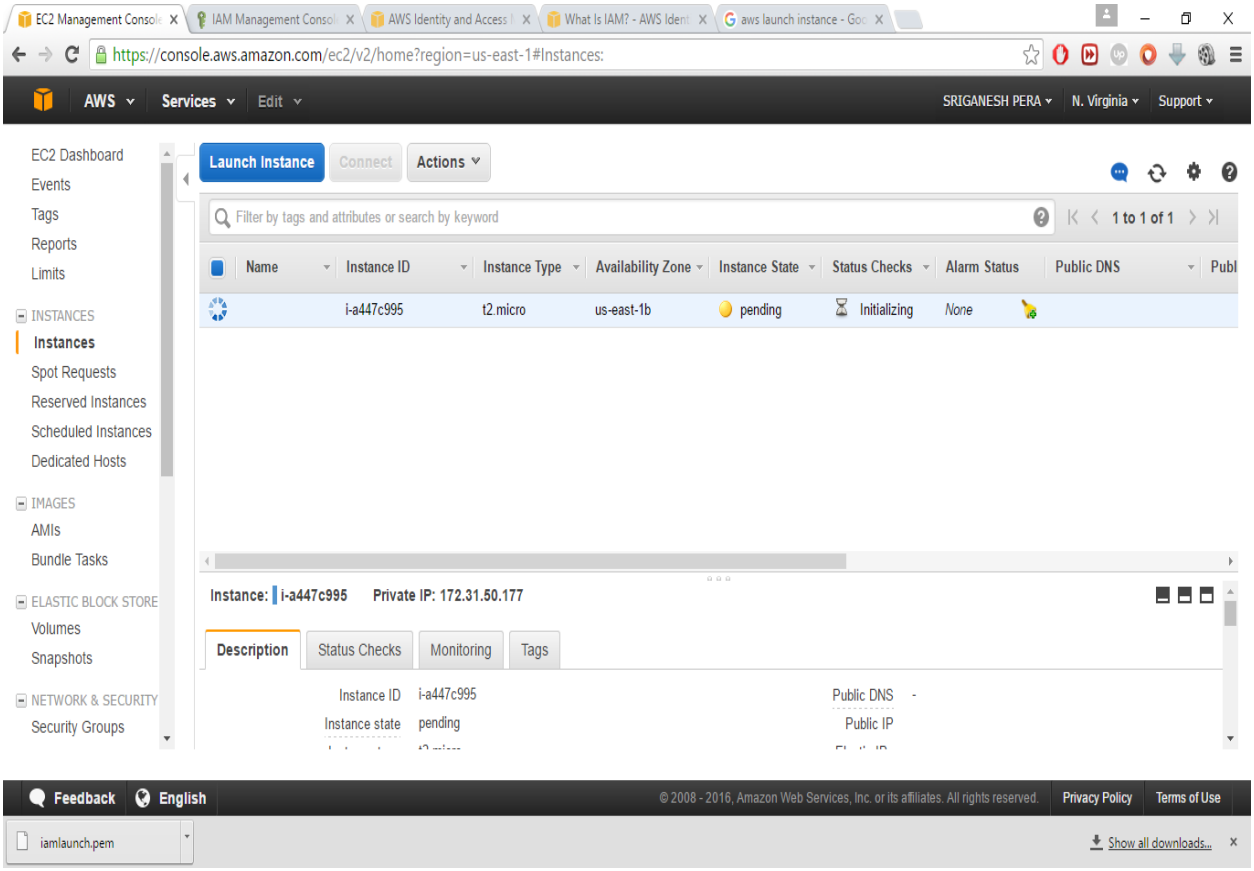
10. Click on add storage, tag instance, and configure security group. Make sure SSH is selected (by default you'll have it so don't change anything) and click in 'Review and Launch' and 'Launch'



11. You'll be asked to select a private key air as you need to SSH into AMI instance that we created in 7th step. Create a private key and name it as you like and the key with the extension '.pem' will be downloaded. Save it in a folder. Then click on Launch instances and then click on View

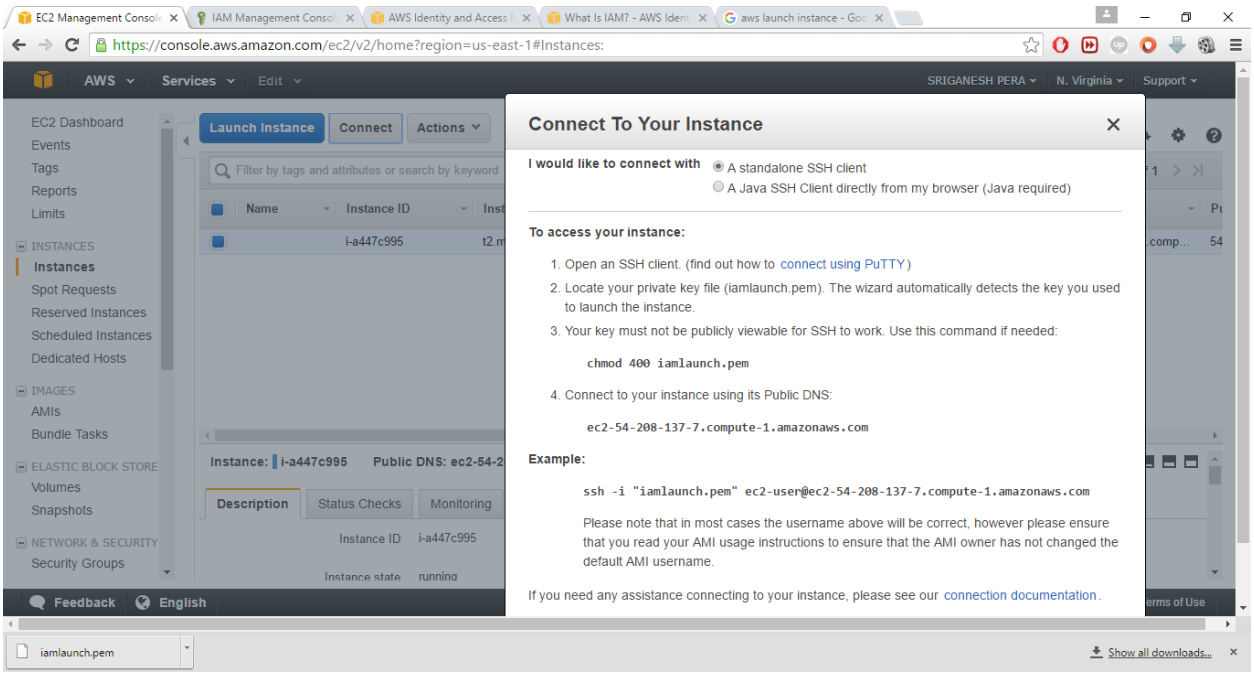


12. As we know it takes some time for the instance to be launched



Now SSH the instance using Command Line

13. Now SSH that instance using your Command line. First click on connect button, you will notice a template pops out which gives you instruction on how to connect



14.Now open your Command Line terminal(either windows or OSX or Linux)

a. First cd to the folder where you've stored your .pem file. I've stored mine in 'Downloads' folder.

b.Then follow the third step in the popped our window which shows chmod 400 'your key filename'.

This needs to be done as you're securely accessing AWS using the private key(as this is not for public access).

c.Then follow the 4th step example as it is given in your AWS popped out screen.

Now you're connect to your instance securely

```
C:\Users\spbg7\Downloads>chmod 400 iamlaunch.pem

C:\Users\spbg7\Downloads>ssh -i "iamlaunch.pem" ec2-user@ec2-54-208-137-7.compute-1.amazonaws.com
The authenticity of host 'ec2-54-208-137-7.compute-1.amazonaws.com (54.208.137.7)' can't be established.
ECDSA key fingerprint is SHA256:VlmKUs9x36n3AWFX0wBHcC+k8sY8lY8ufobDYjnidjM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-208-137-7.compute-1.amazonaws.com,54.208.137.7' (ECDSA) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-ami/2016.03-release-notes/
10 package(s) needed for security, out of 22 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-50-177 ~]$
```

15. Now type in Command line:

1. Sudo yum update. After the process is complete

2.Type Sudo yum python-pip. After the process is complete.

3.Type sudo pip install boto3.

What is Boto3?

***Boto** is the Amazon Web Services (AWS) SDK for Python, which allows Python developers to write software that makes use of Amazon services like S3 and EC2. **Boto** provides an easy to use, object-oriented API as well as low-level direct service access.*

```
Command Prompt - ssh -i "iamlaunch.pem" ec2-user@ec2-54-208-137-7.compute-1.amazonaws.com
  Installing : python26-setuptools-12.2-1.30.amzn1.noarch                                                    5/6

  Installing : python26-pip-6.1.1-1.21.amzn1.noarch                                                          6/6

  Verifying  : python26-setuptools-12.2-1.30.amzn1.noarch                                                    1/6

  Verifying  : python26-pip-6.1.1-1.21.amzn1.noarch                                                          2/6

  Verifying  : python26-libs-2.6.9-2.88.amzn1.x86_64                                                         3/6

  Verifying  : python26-2.6.9-2.88.amzn1.x86_64                                                              4/6

  Verifying  : python26-backports-ssl_match_hostname-3.4.0.2-1.12.amzn1.noarch                               5/6

  Verifying  : python26-backports-1.0-3.14.amzn1.x86_64                                                      6/6


Installed:
  python26-pip.noarch 0:6.1.1-1.21.amzn1


Dependency Installed:
  python26.x86_64 0:2.6.9-2.88.amzn1          python26-backports.x86_64 0:1.0-3.14.amzn1       python26-backports-ssl_match_hostname.noarch 0:3.4.0.2-1.12.amzn1
  python26-libs.x86_64 0:2.6.9-2.88.amzn1     python26-setuptools.noarch 0:12.2-1.30.amzn1

Complete!
[ec2-user@ip-172-31-50-177 ~]$ sudo pip install boto3
You are using pip version 6.1.1, however version 8.1.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Collecting boto3
  Downloading boto3-1.4.0-py2.py3-none-any.whl (117kB)
    100% |################################| 118kB 2.6MB/s
Requirement already satisfied (use --upgrade to upgrade): jmespath<1.0.0,>=0.7.1 in /usr/lib/python2.7/dist-packages (from boto3)
Requirement already satisfied (use --upgrade to upgrade): botocore<1.5.0,>=1.4.1 in /usr/lib/python2.7/dist-packages (from boto3)
Collecting s3transfer<0.2.0,>=0.1.0 (from boto3)
  Downloading s3transfer-0.1.3-py2.py3-none-any.whl (50kB)
    100% |################################| 53kB 6.9MB/s
Requirement already satisfied (use --upgrade to upgrade): python-dateutil<3.0.0,>=2.1 in /usr/lib/python2.7/dist-packages (from botocore<1.5.0,>=1.4.1->boto3)
Requirement already satisfied (use --upgrade to upgrade): docutils>=0.10 in /usr/lib/python2.7/dist-packages (from botocore<1.5.0,>=1.4.1->boto3)
Requirement already satisfied (use --upgrade to upgrade): futures<4.0.0,>=2.2.0 in /usr/lib/python2.7/dist-packages (from s3transfer<0.2.0,>=0.1.0->boto3)
Requirement already satisfied (use --upgrade to upgrade): six in /usr/lib/python2.7/dist-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.5.0,>=1.4.1->boto3)
Installing collected packages: s3transfer, boto3
Successfully installed boto3-1.4.0 s3transfer-0.1.3
[ec2-user@ip-172-31-50-177 ~]$
Skype
```

16. Now let’s write a simple python script about we can securely access S3 buckets .

a. Type vi dev.py (Name of your python script)

b. Type ‘I’ so that you can insert some code into it.

c. Write the following code:

**make sure that the bucket name you create must be uniquely named.

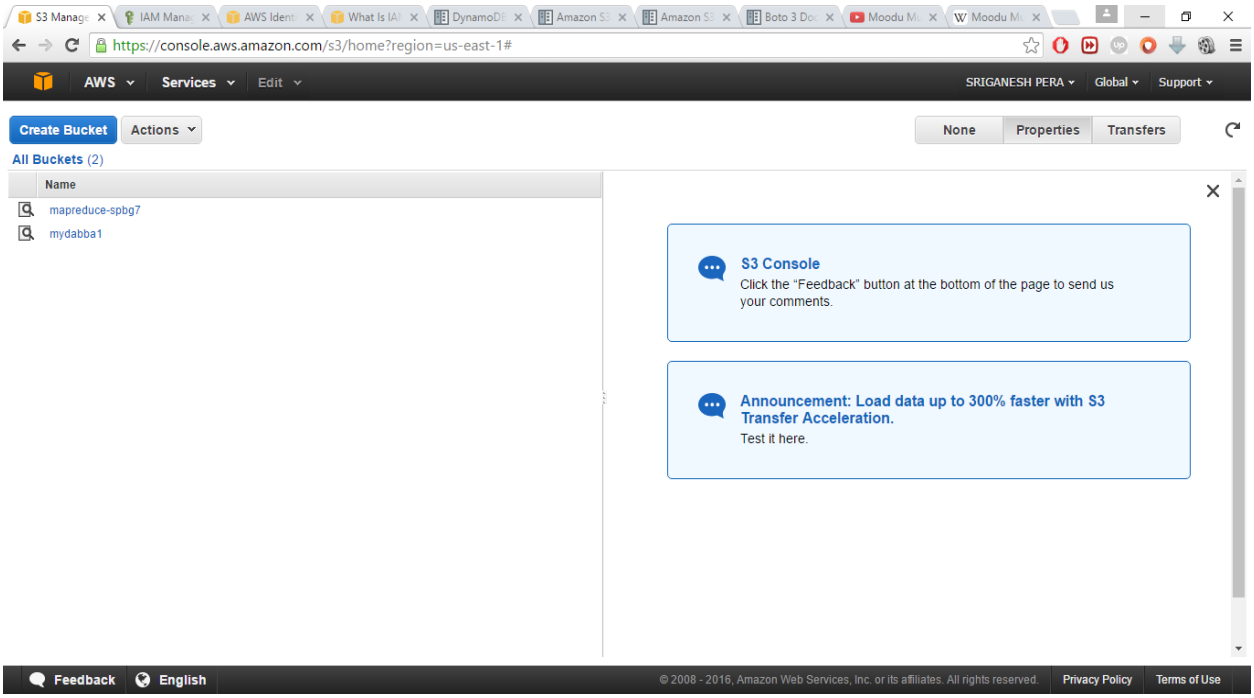
```
#!/usr/bin/python
#imported boto3
    import boto3
 #create connection to S3 using
            boto S3=boto.resource('s3')
     # Create a S3 bucket using boto3
          S3.create_bucket (Bucket='mydabba1')
```

d.Then click 'esc' button and type wq!,which save the code.

17.Now set the executable permission for our script dev.py by typing chmod u+x dev.py and run the script by typing /.dev.py

```
https://aws.amazon.com/amazon-linux-ami/2016.03-release-notes/
[ec2-user@ip-172-31-50-177 ~]$ vi dev.py
[ec2-user@ip-172-31-50-177 ~]$ vi dev.py
[ec2-user@ip-172-31-50-177 ~]$ chmod u+x dev.py
[ec2-user@ip-172-31-50-177 ~]$ /.dev.py
```

18. Now got to AWS console and click on S3 and you'll find the bucket created with the name 'mydabba1'

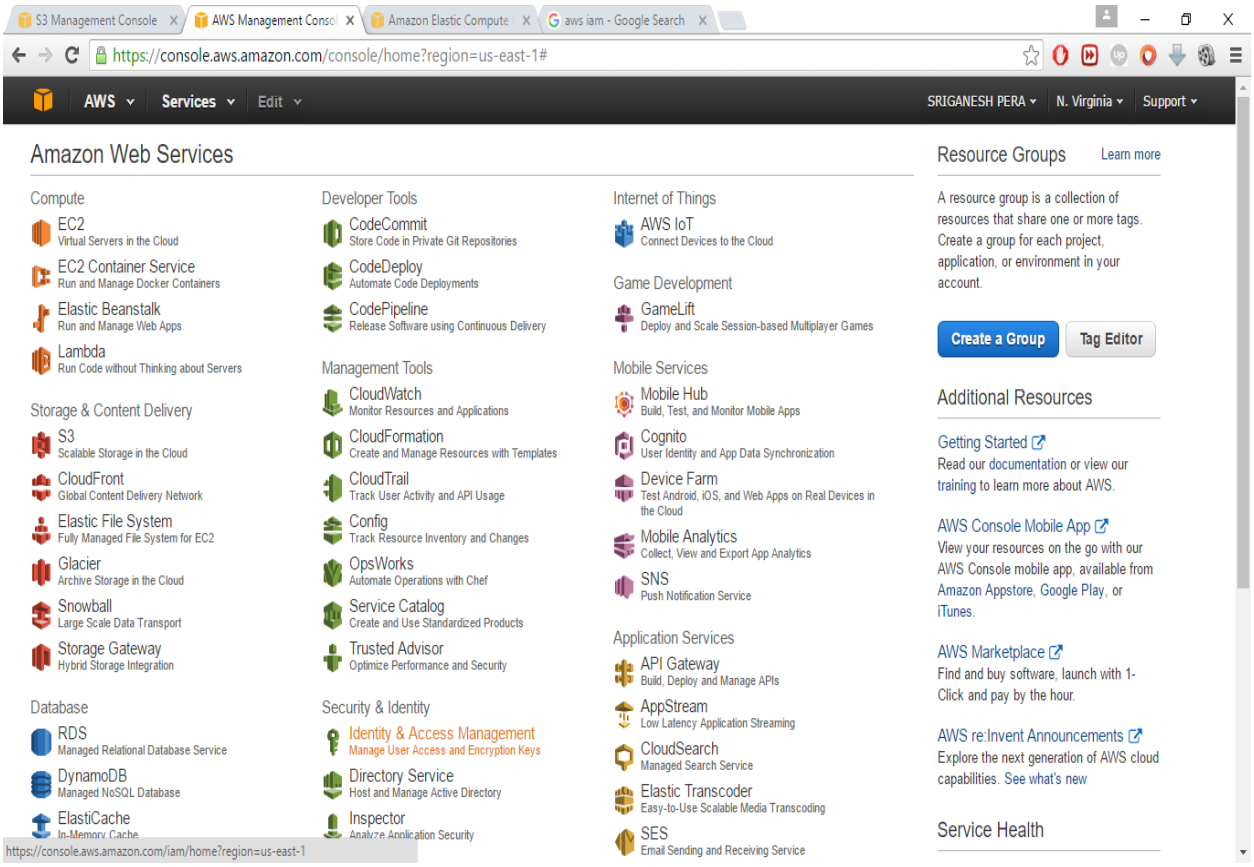


This is how you can access S3 buckets using Python code running on EC2 machine with IAM roles.

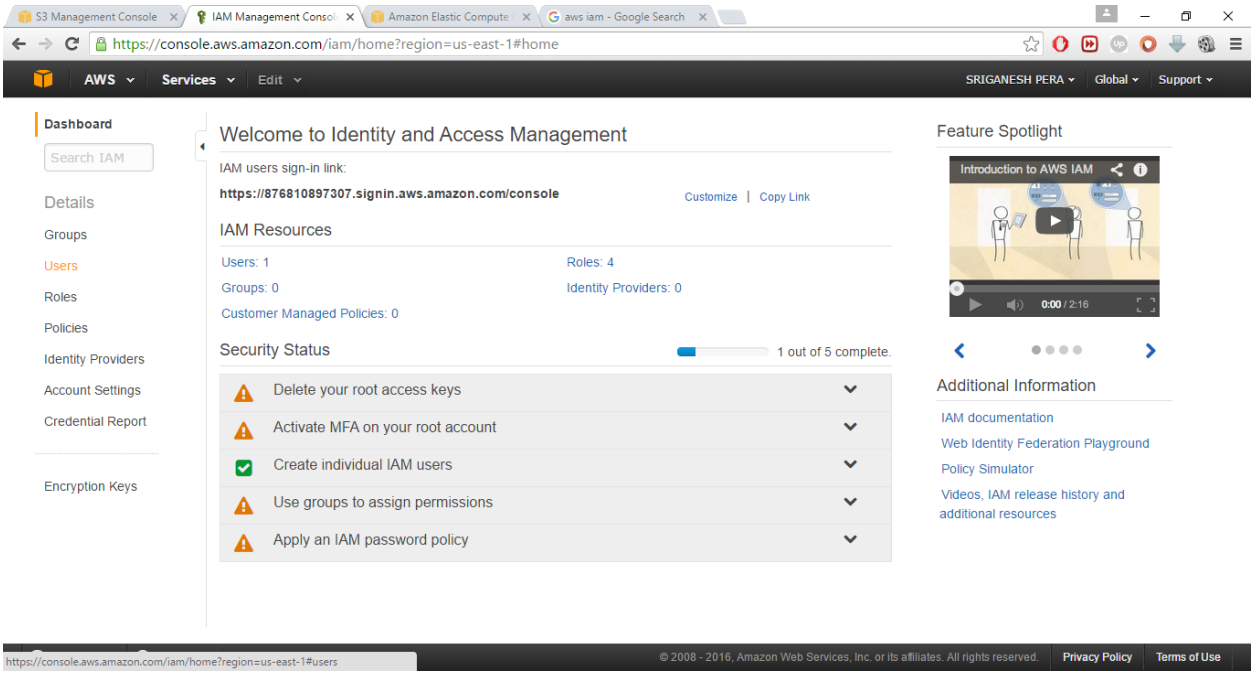
Lab 2: Configuring the Boto3 SDK with API Credentials

This time AWS access is given to users using API credentials

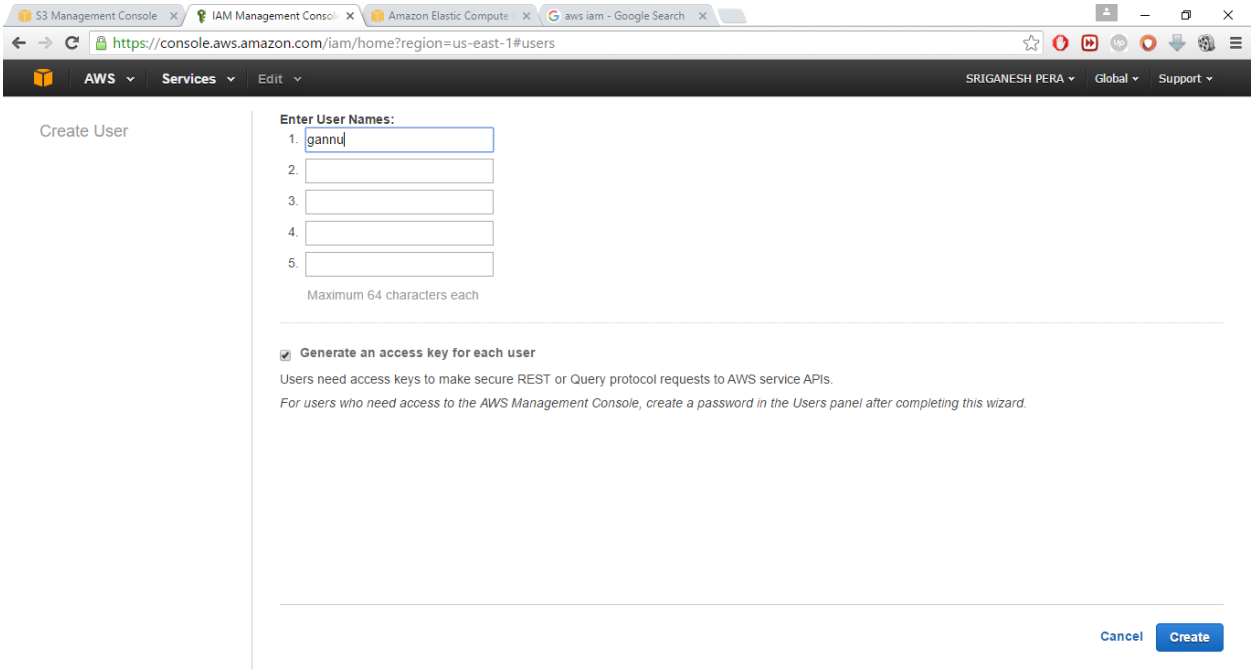
1. Go to AWS IAM



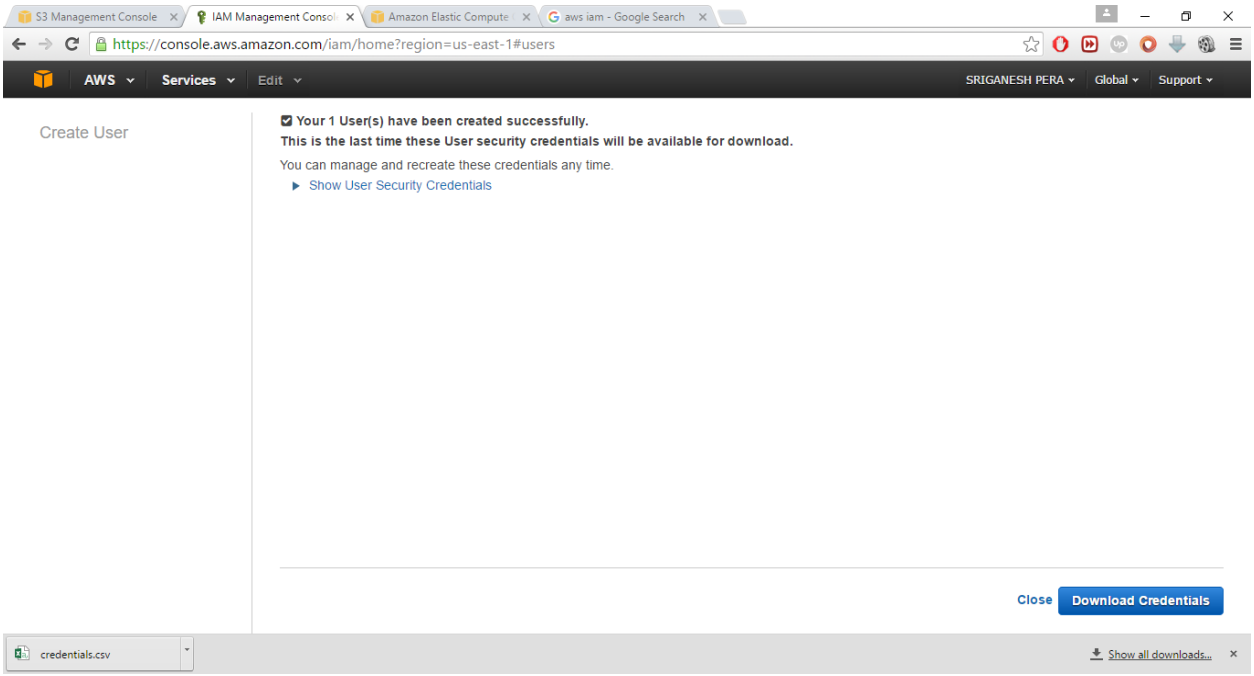
2. Click on Users



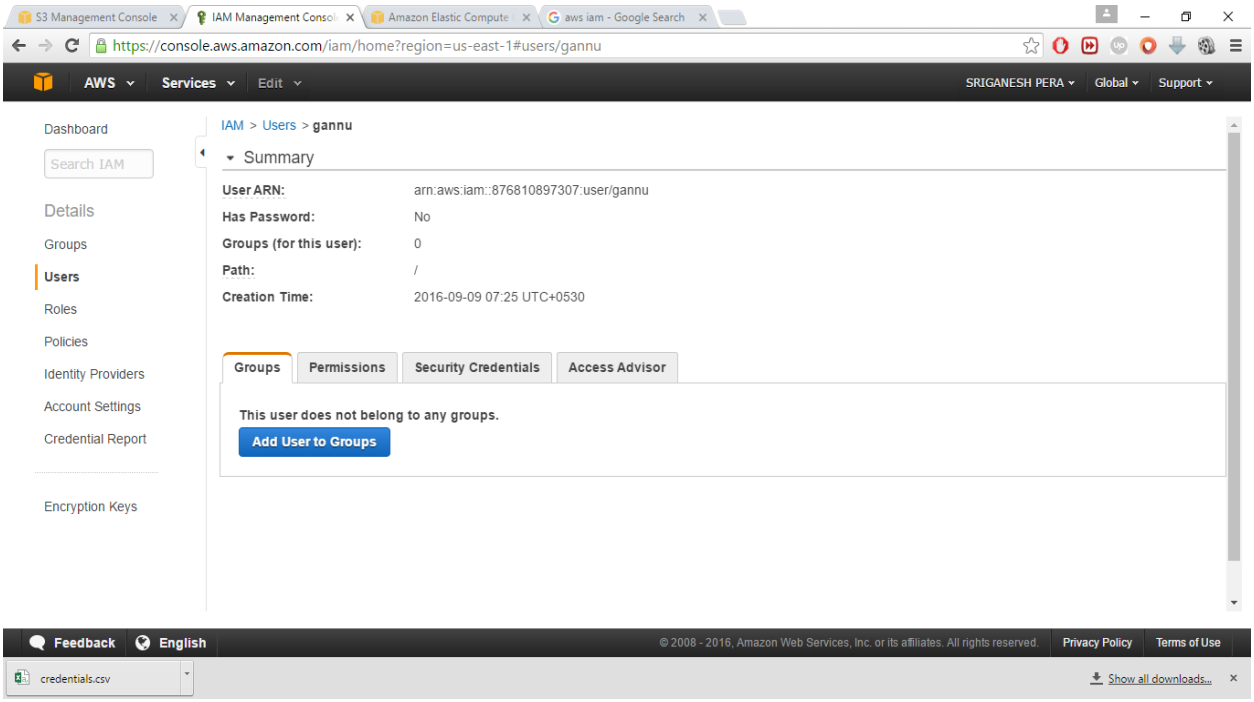
3. Click on create new user and type in the name of the user.



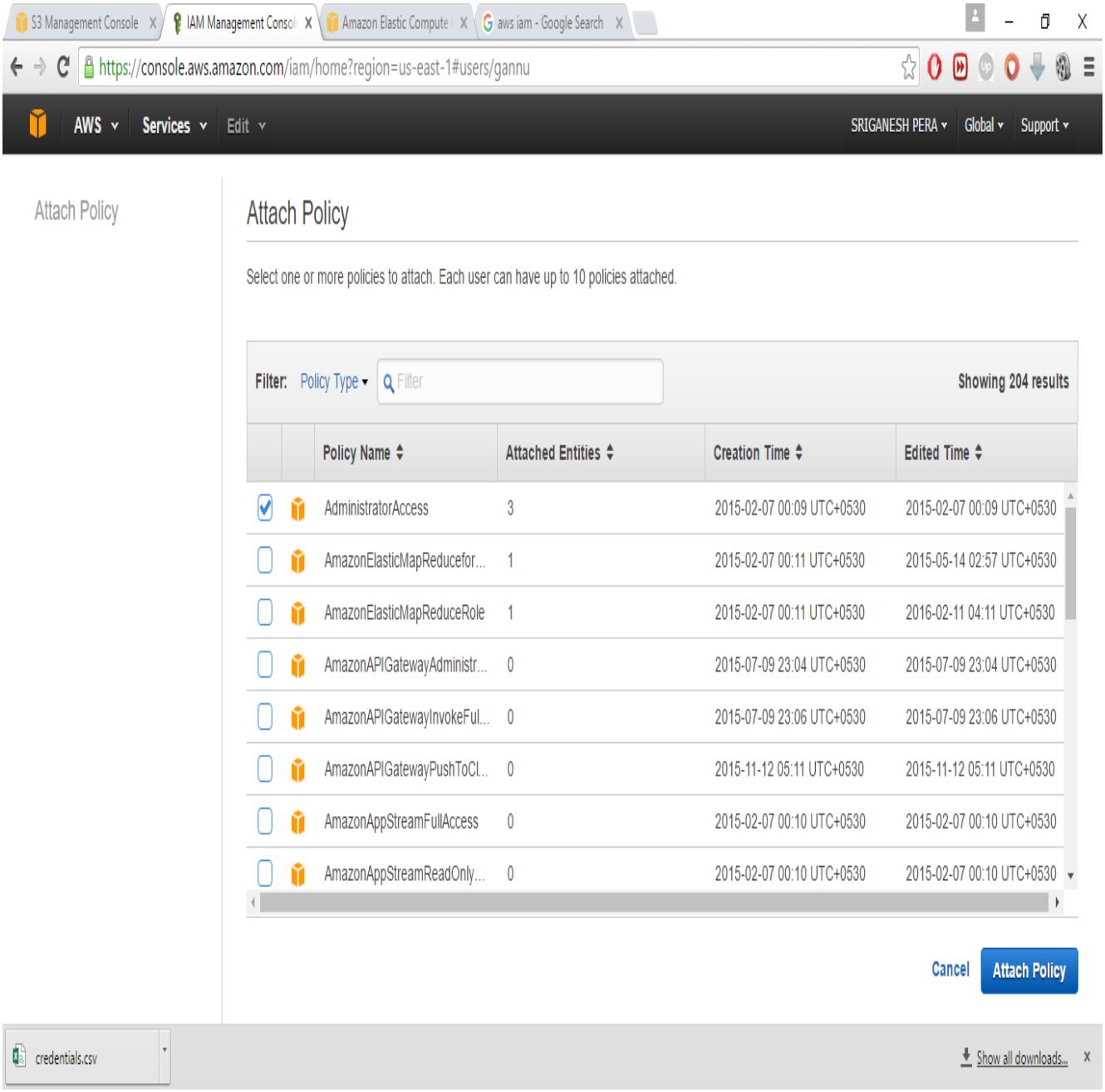
4. Click on Create Users and then Download Credentials. You'll get a CSV file.



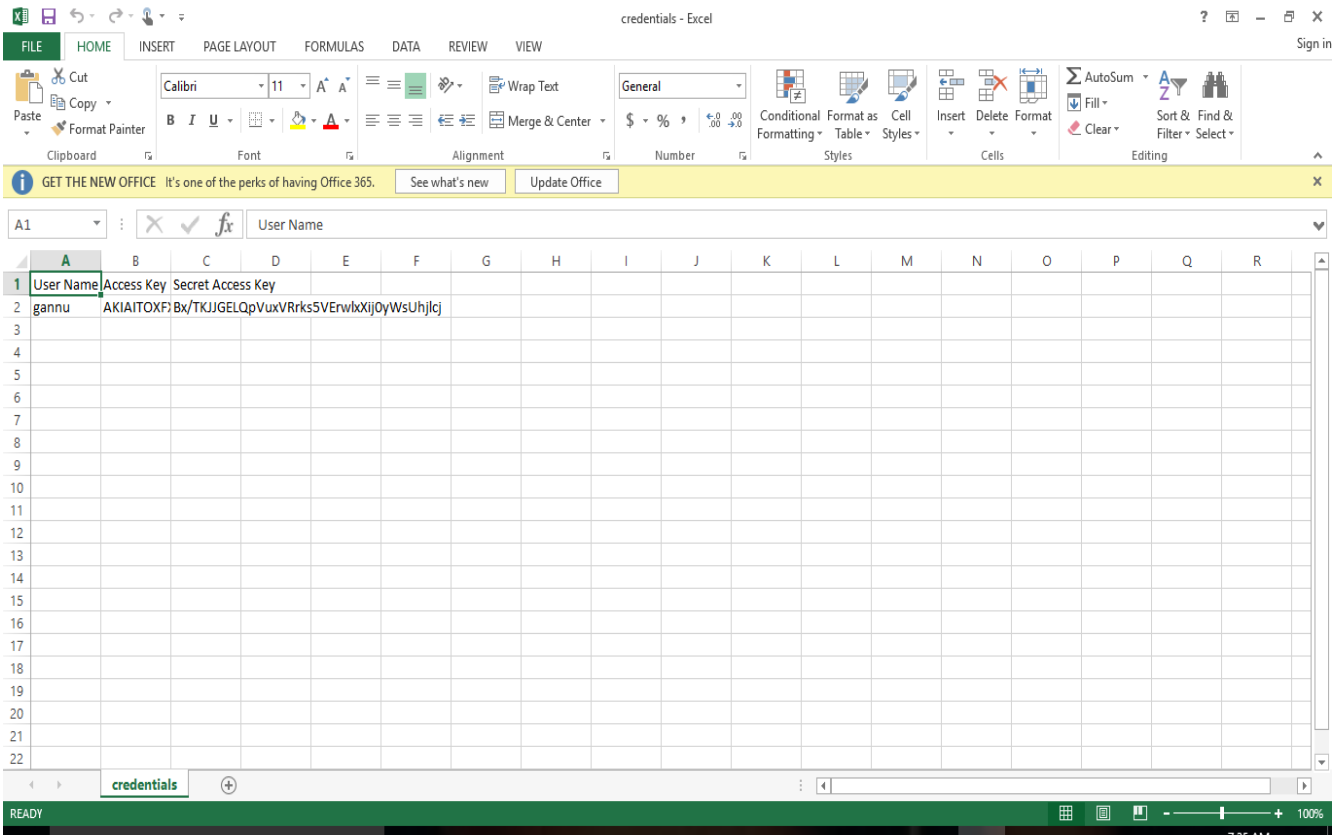
5. Now click on close and select the username you just created and click on the username.



5. **Now click on attach policy and give access to whatever the services you want to give to that user In this case I'm giving admin access.**

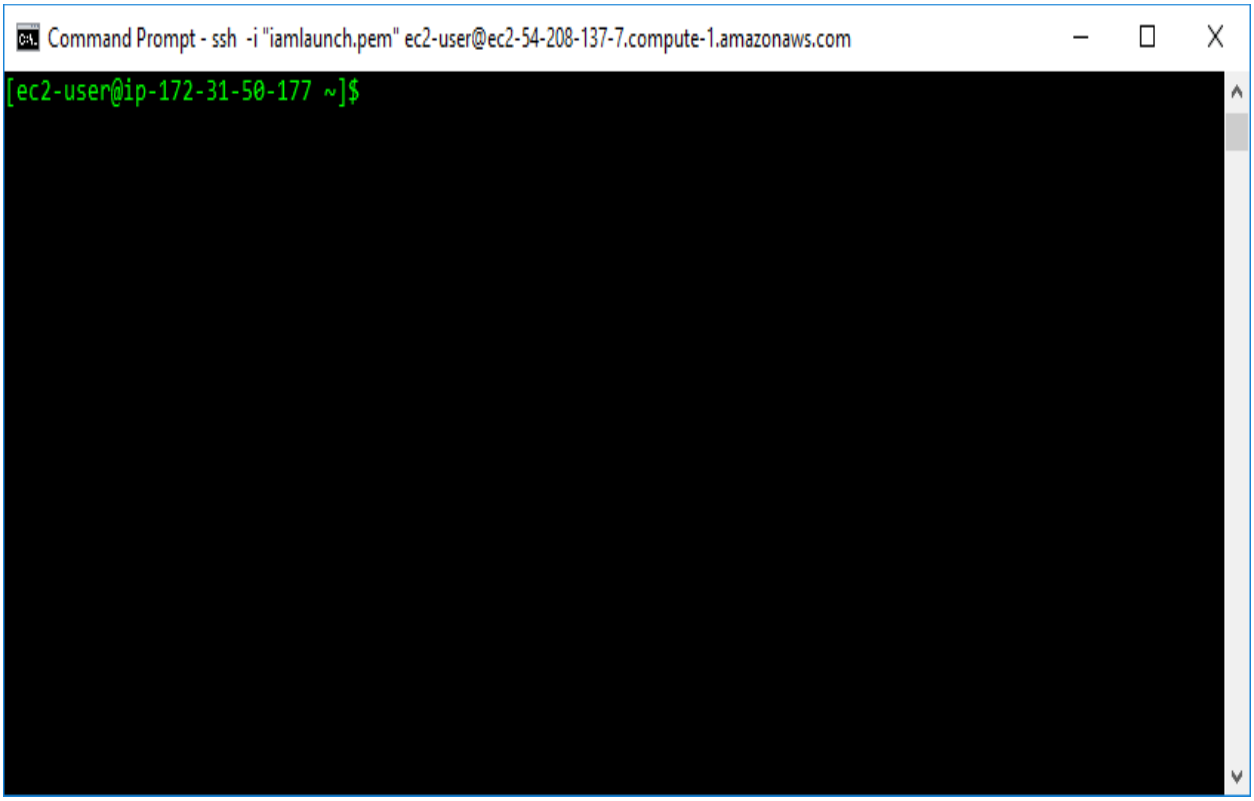


6.Now you've attached policy. Then click on the credentials we downloaded earlier to see what it has to offer



As you can see you can find username 'gannu'(which I have created along with the access key and secret access key)

7. Now let's get back to our terminal window and connect it back to the AMI instance we have launched. If you're already connected then don't worry.



8. Now write the following Python Boto3 code in your favorite editor (for me it's Vim) and run the code the same way we have done it in previous lab.

In you cmd prompt type: vi dev 1.py

Then type i to insert the following code:

#!/usr/bin/python

from boto3.session import Session

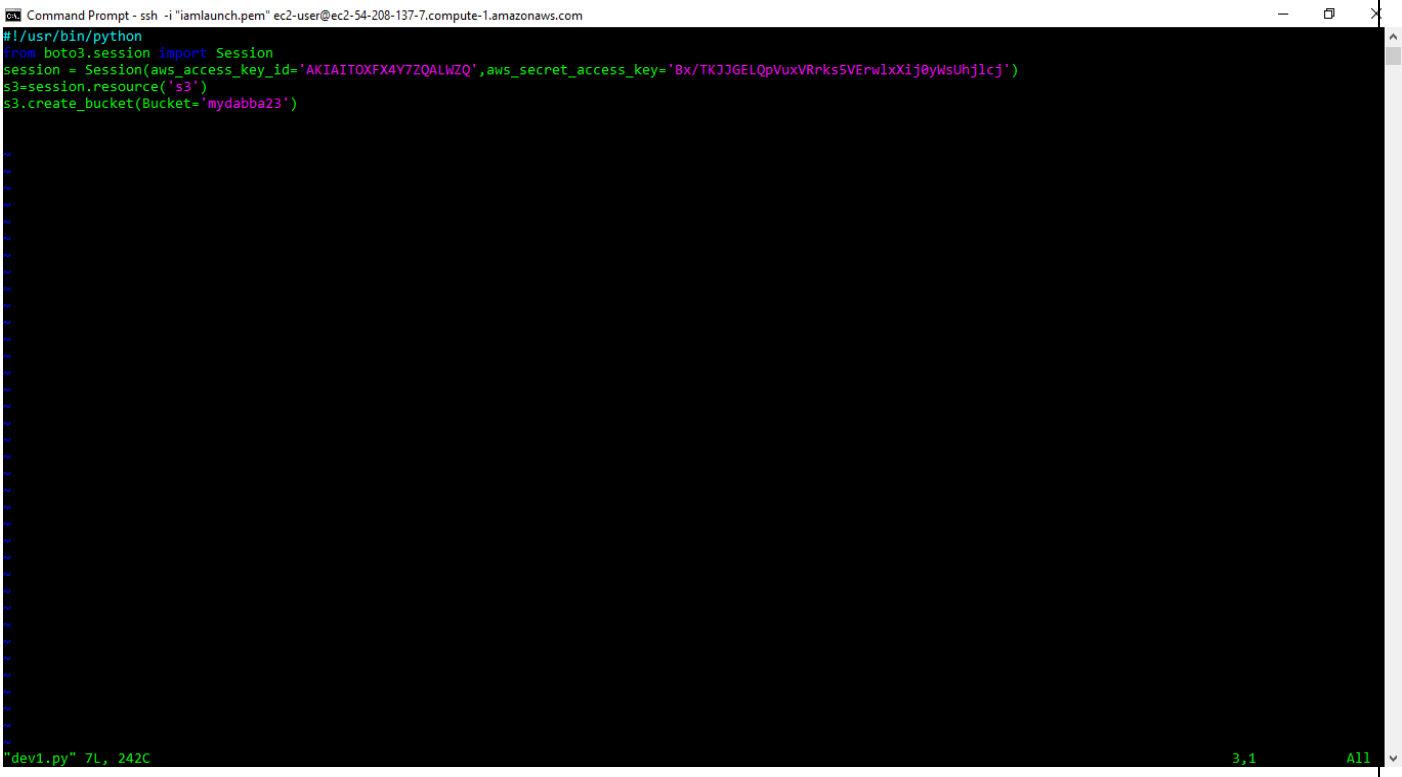
** rather than importing boto3 we import a session from boto3**

session = Session (aws_access_key_id=’AKIAITOXFX4Y7ZQALWZQ’,aws_secret_access_key=’ Bx/TKJJGELQpVuxVRrks5VErwlxXij0yWsUhjlcj’)

s3 = session.resource(‘s3’) **#creating a session to S3**

s3.create_bucket(Bucket=’mydabba23’) **#creating a bucket**

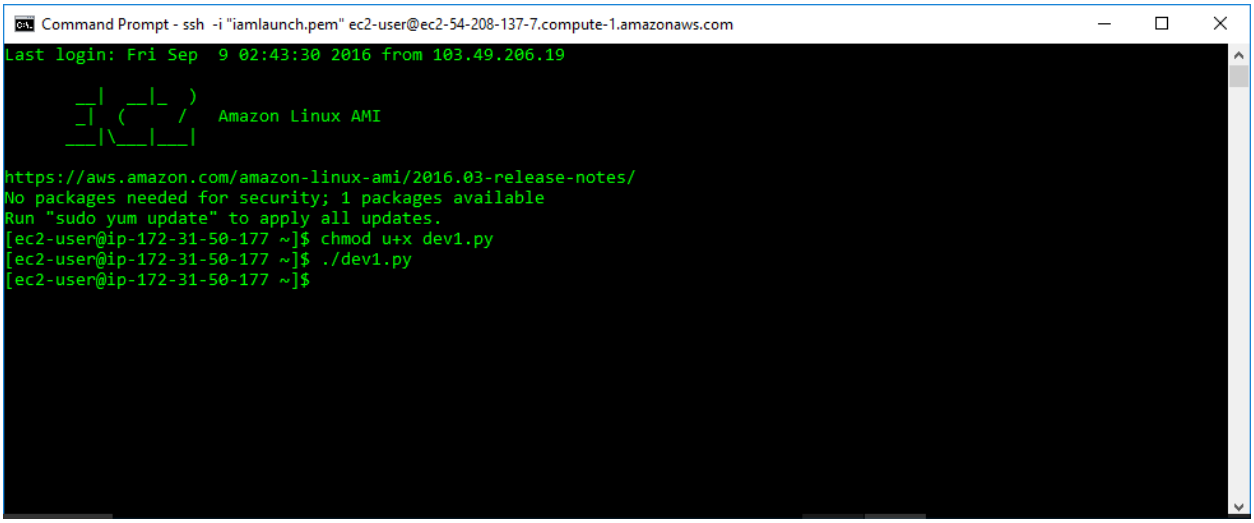
In the parameters for Session i.e aws_access_key_id and aws_secret_Access_key.You need to add the secret keys you downloaded in the step



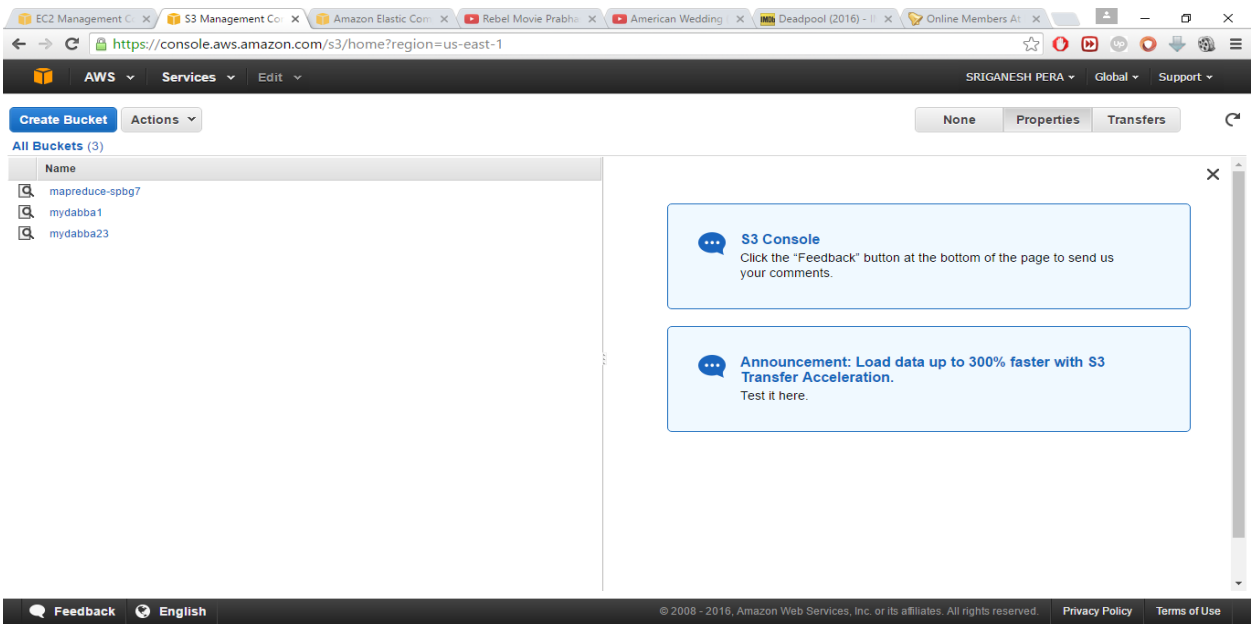
Now run the code :

chmod u+x dev1.py

./dev1.py

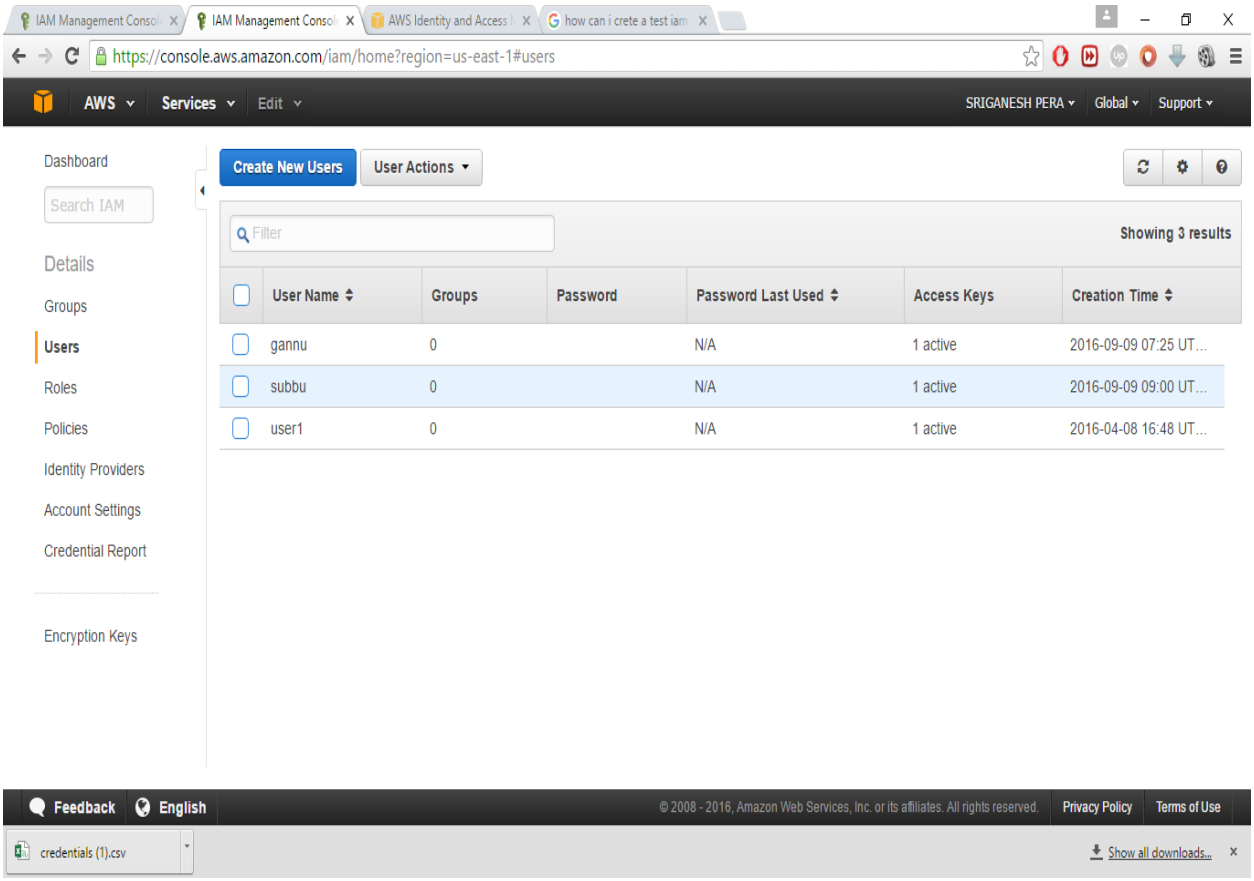


Now check your S3 in AWS console.You'll find the new bucket 'mydabba23' .

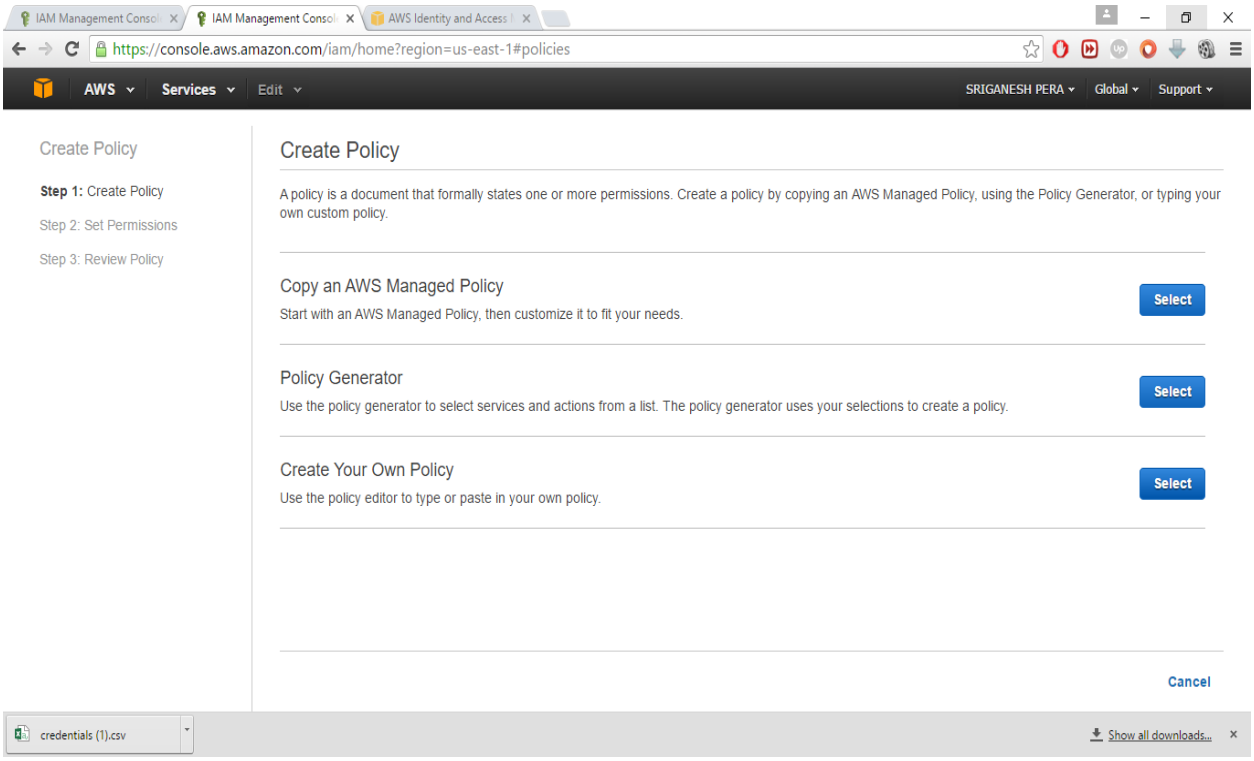


LAB 3 : Create policy for users for restricted access to AWS.

1.Create a new user in IAM console:



2.Click on Policies and then click on Create your own policy

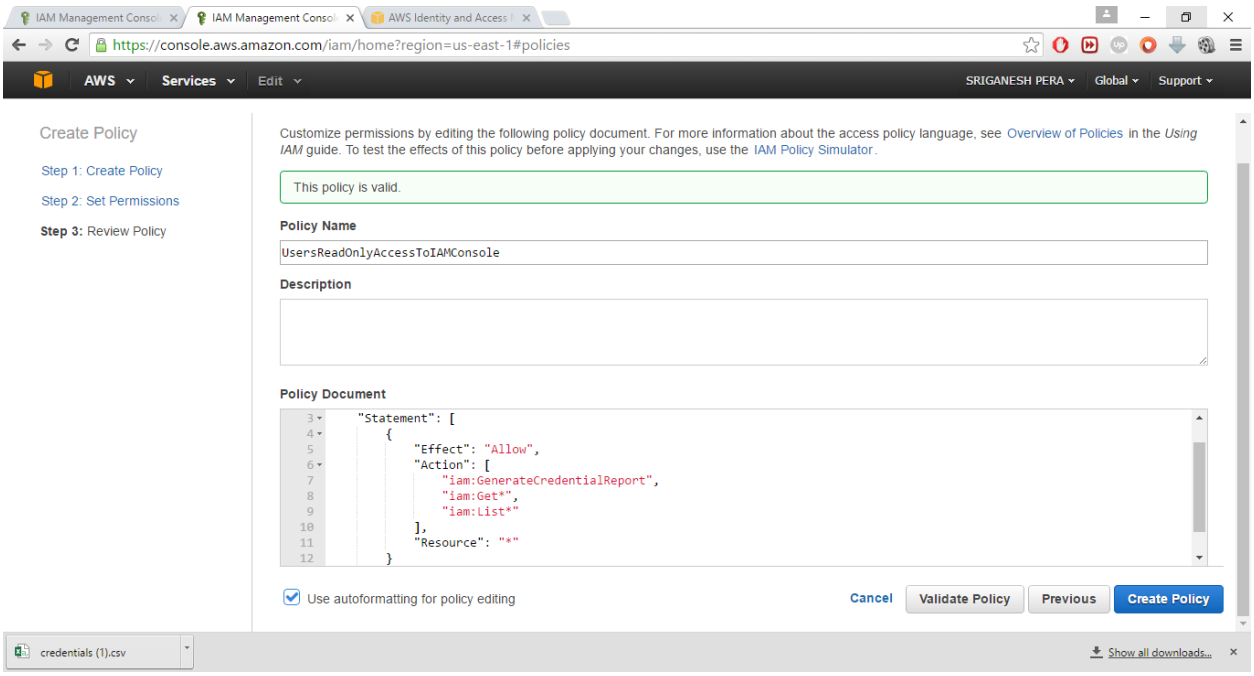


3.In the Policyname type : UsersReadOnlyAccessToIAMConsole and in the Policy Document type the following code:

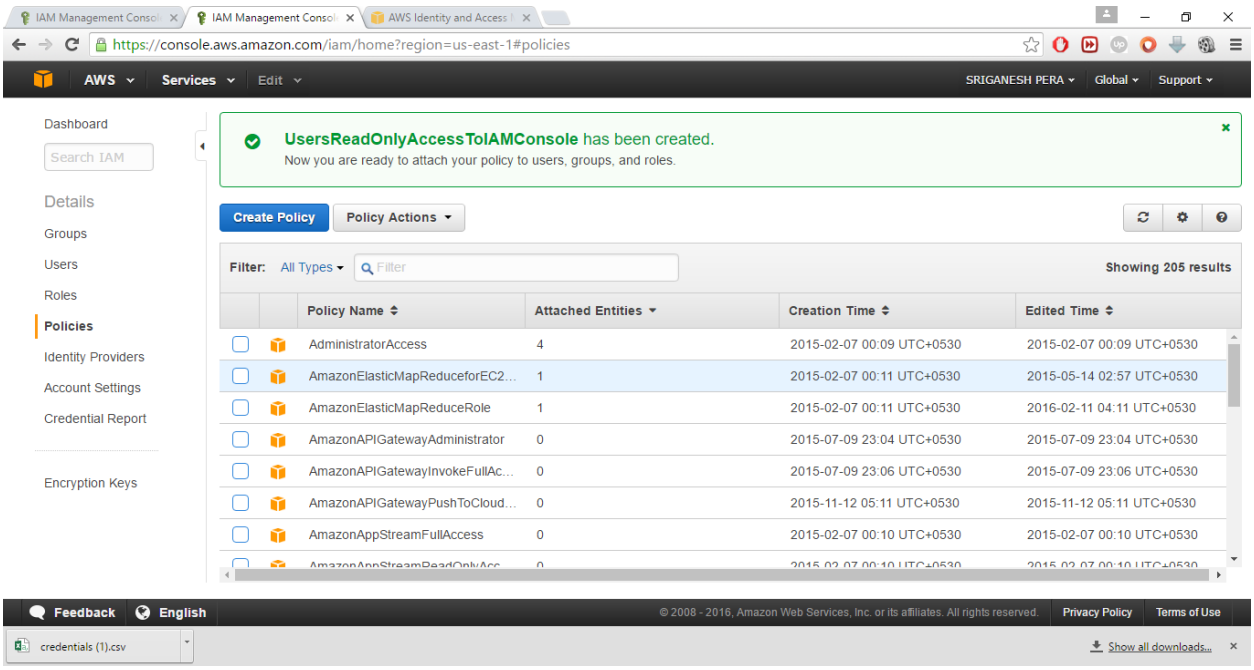
```
{
 "Version": "2012-10-17",
 "Statement": [ {
 "Effect": "Allow",
 "Action": [
 "iam:GenerateCredentialReport",
```

```
"iam:Get*",
"iam:List*"
],
"Resource": "*"
} ]
}
```

Click on **Validate now** to make sure the code is error free and if you get '**policy is valid**' in green color at the top

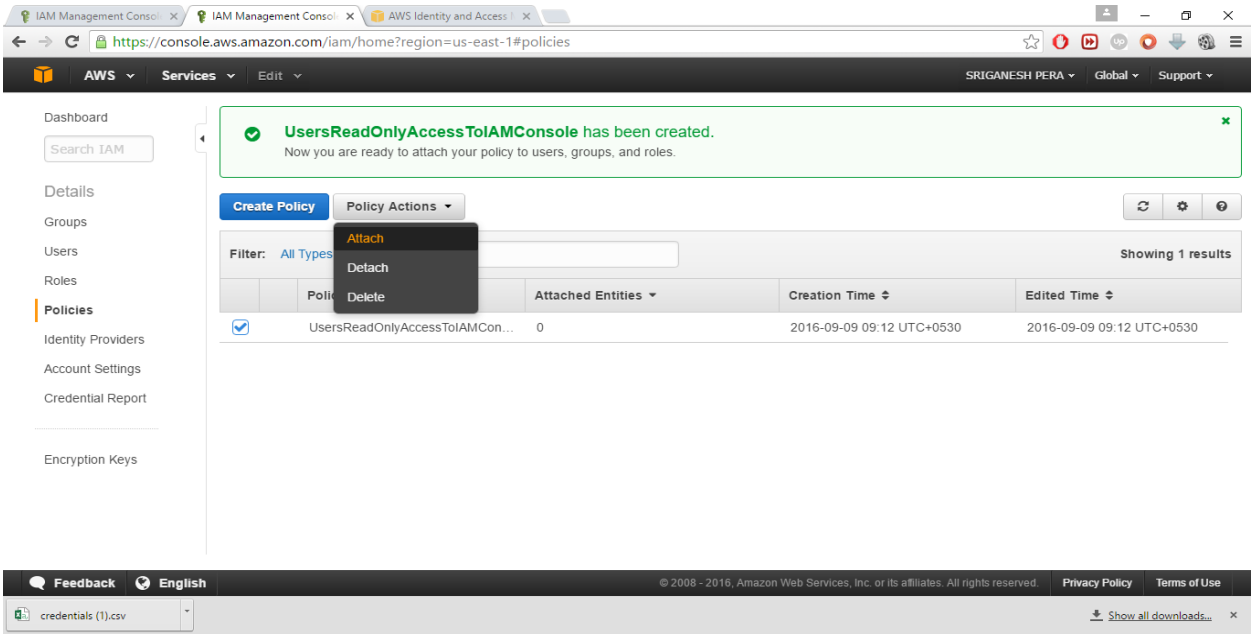


Click on **create policy**.

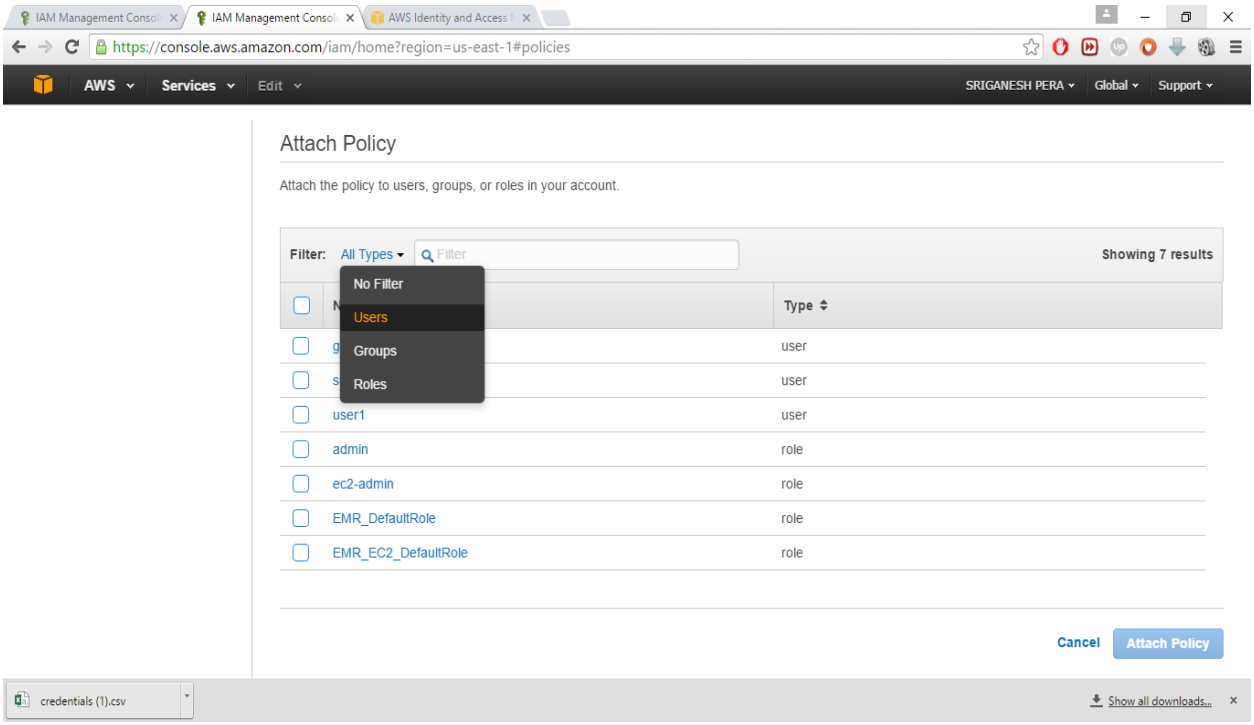


If you can search the policy you just created in search bar. You'll find it.

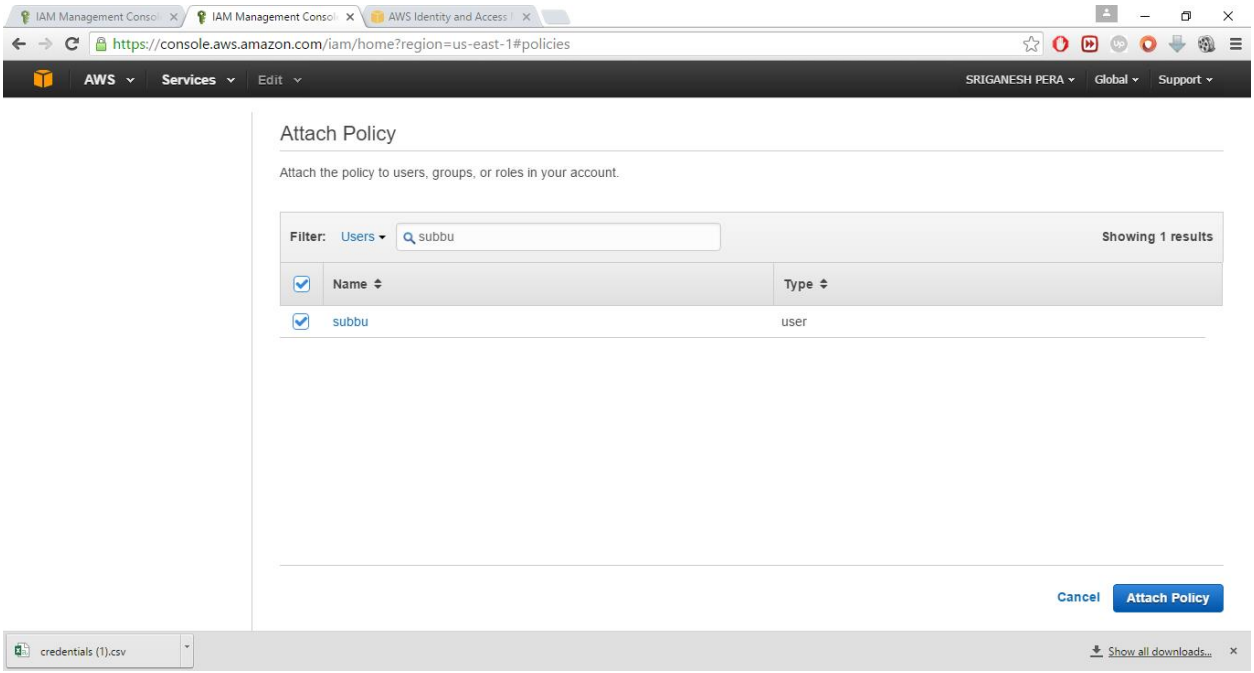
4.Select your policy and click on policy actions and choose Attach policy form the drop-down list



5.Now Choose **All Types** to display the filter menu, and then choose **Users.**



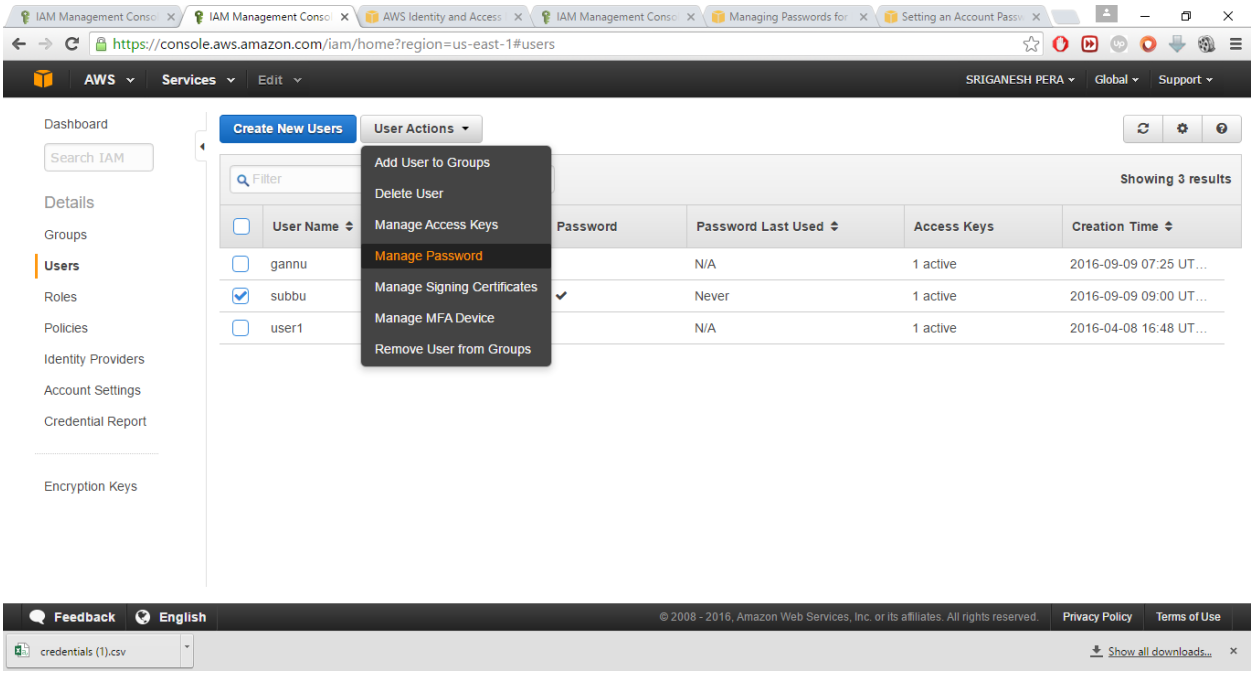
6.Now select your user and click on attach your policy



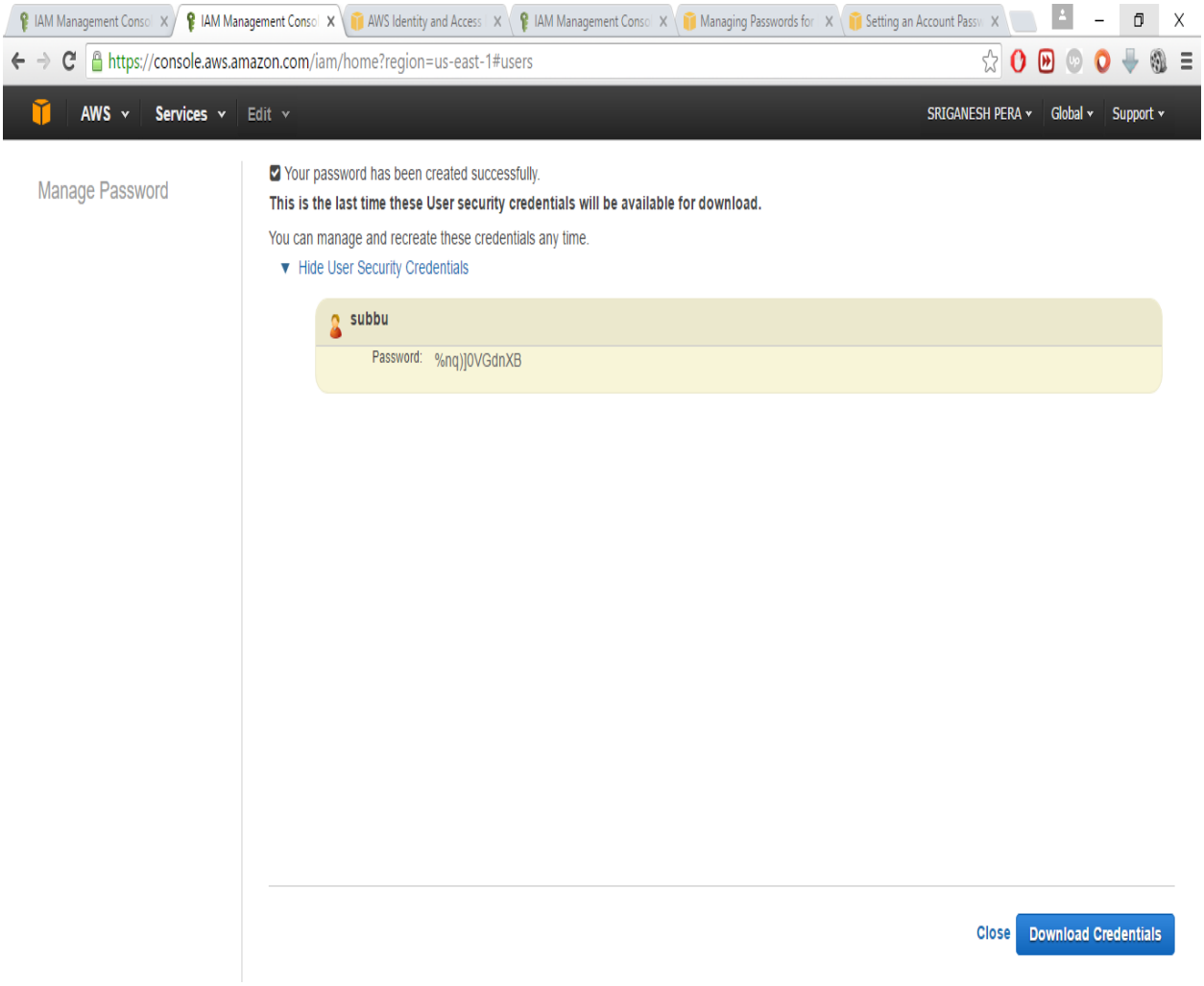
Now You have attached the policy to your IAM test user('subbu' in my case), which means that user now has read-only access to the IAM console.

Now we need to test the user whether he can access as read only user. Before we test it we need to create a password for the user

7.In the IAM console Select **users** and check the respective user you would like to create a password and the click on **user actions** and then click on **Manage passwords**



8. Then click on auto generate password and click on apply. Now click on show user security credentials. Copy the password and save it in a file

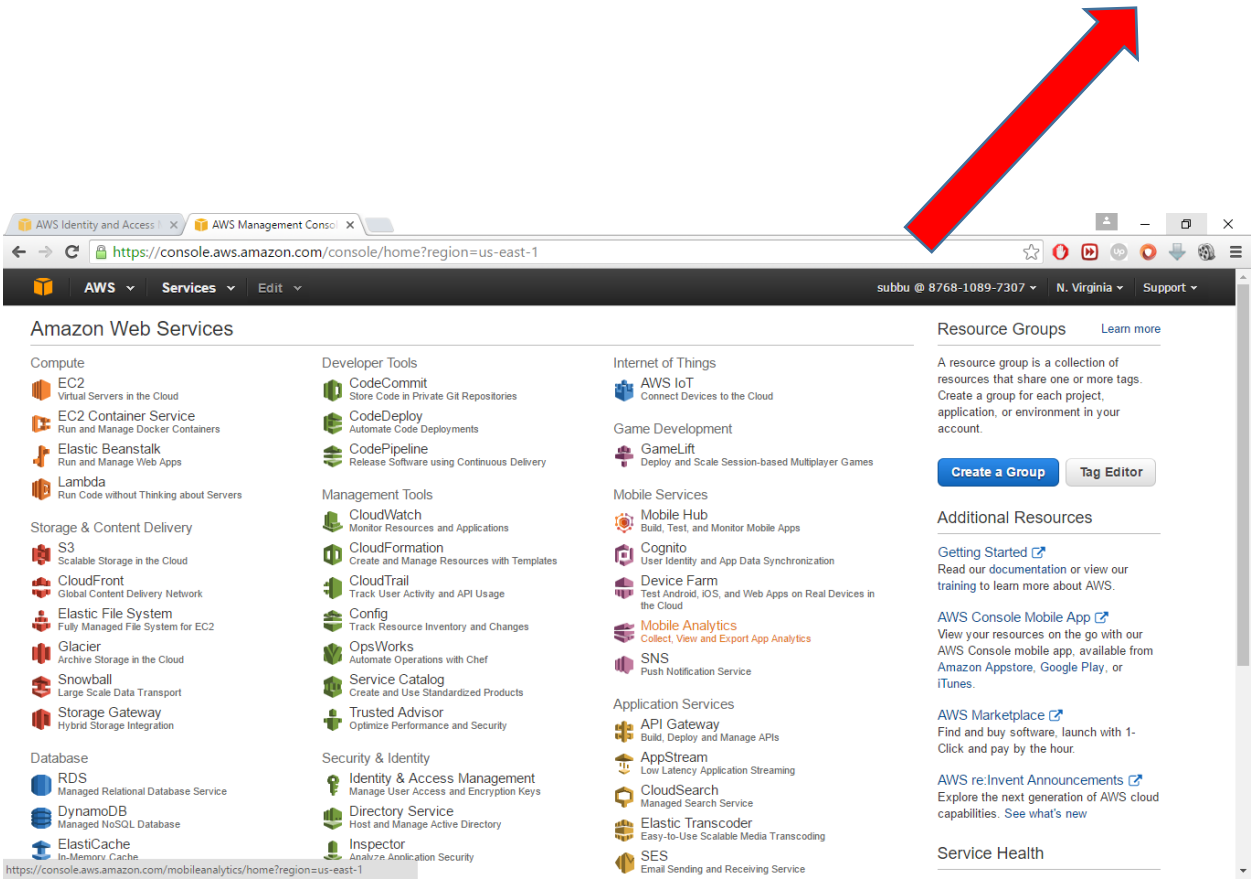


qyjEgZc}ptYB

9.Now its quite simple.Go back to IAM console Dashboard.On the top of the screen you'll find **IAM users sign-in link:** Copy that link and go to that link.Now you'll get the login page asking for username and password.

In the username box type the name of the **user** you created in the **IAM console** and in the **password** field type the password you just saved.

If you notice I 've logged in as the IAM user 'Subbu' I've created.

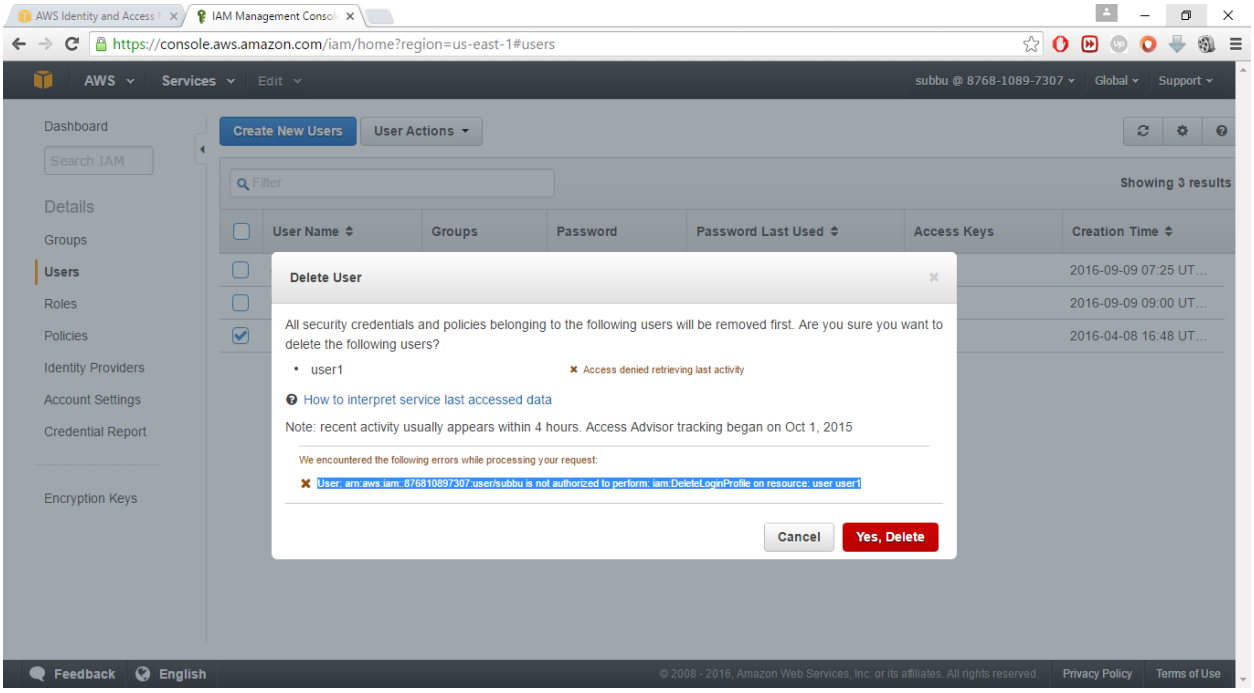


Now lets try to make changes with IAM and see whtehr the user Subbu has the read-only access or not.

I've gone to IAM Console and tried deleting an user there.

Bingo!

I received an error message that I'm not authorized to do that!



----------End of IAM tutorial----------