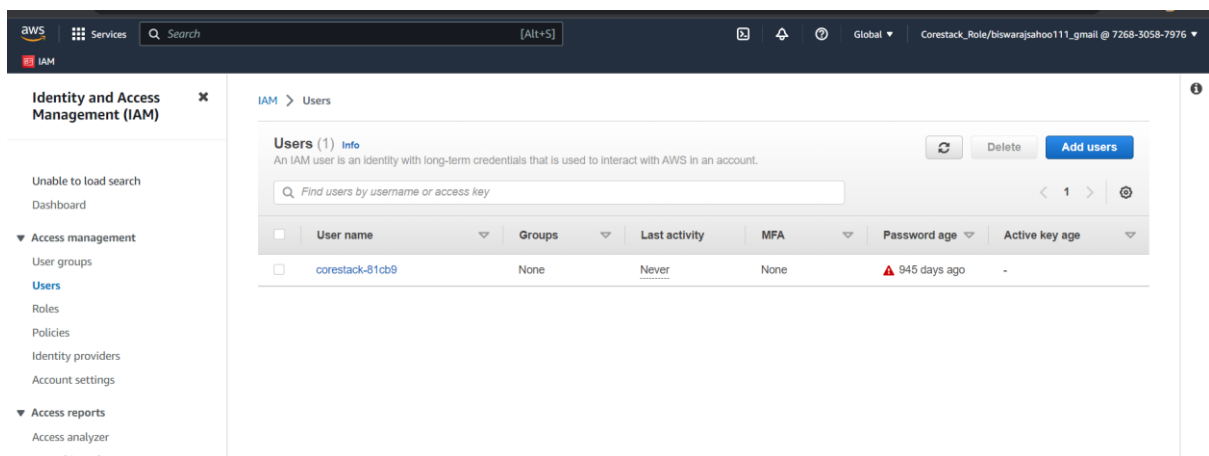


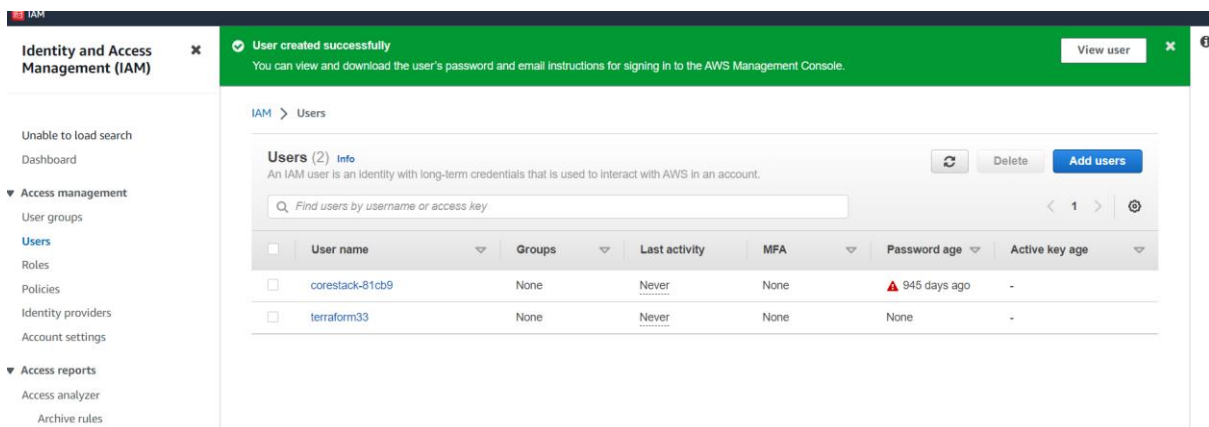
Automating Infrastructure using Terraform

Create EC2 instance

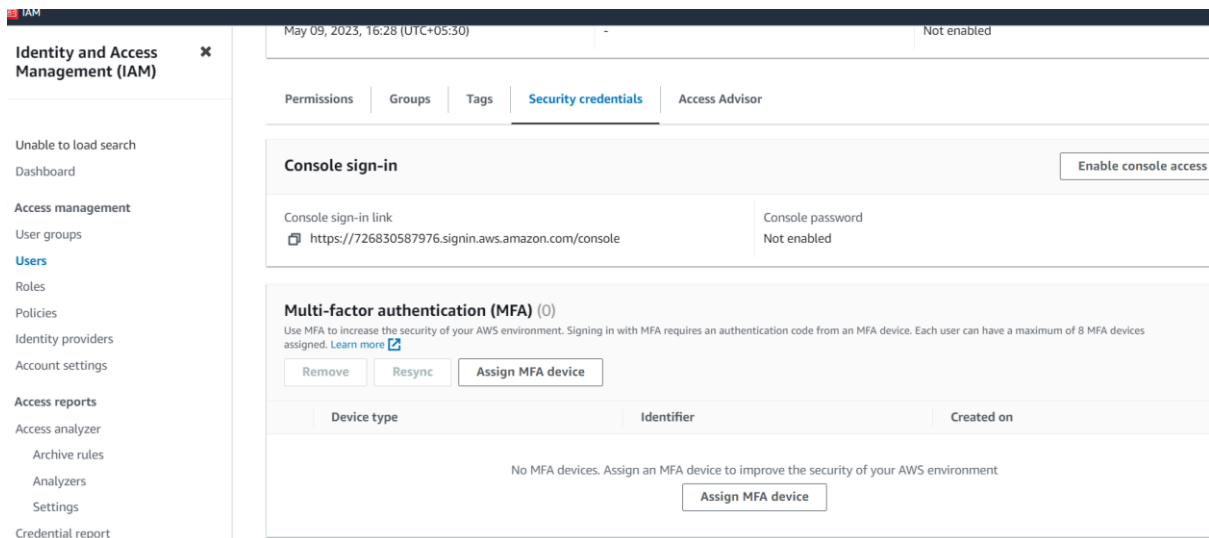
First create a user in aws services



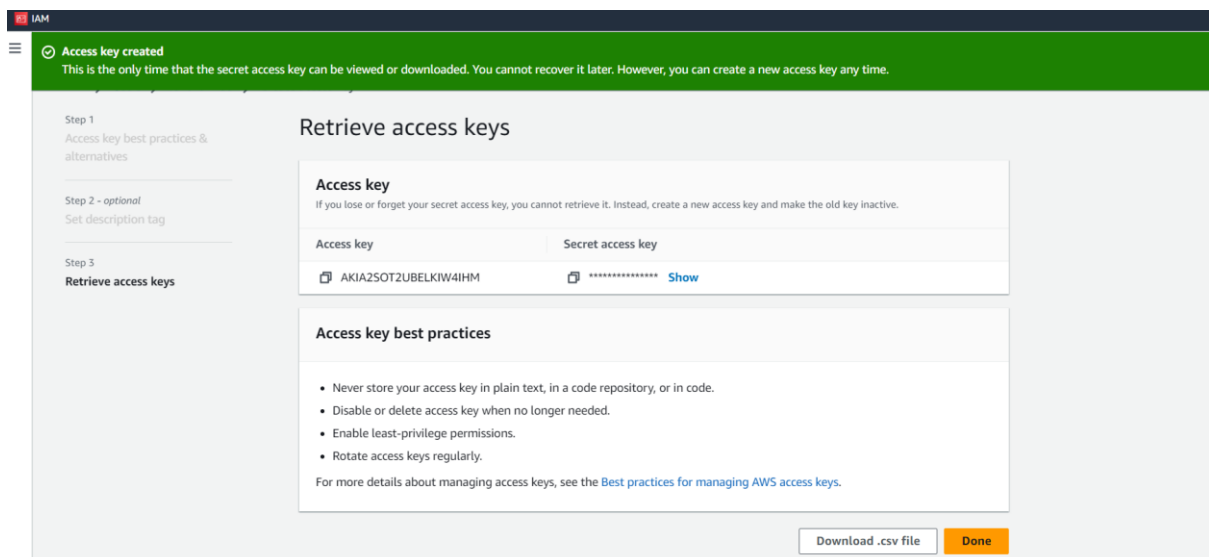
Name of the user = Terraform33



Now we will go to user (Terraform33) security credentials to create access key and security key



Create access key

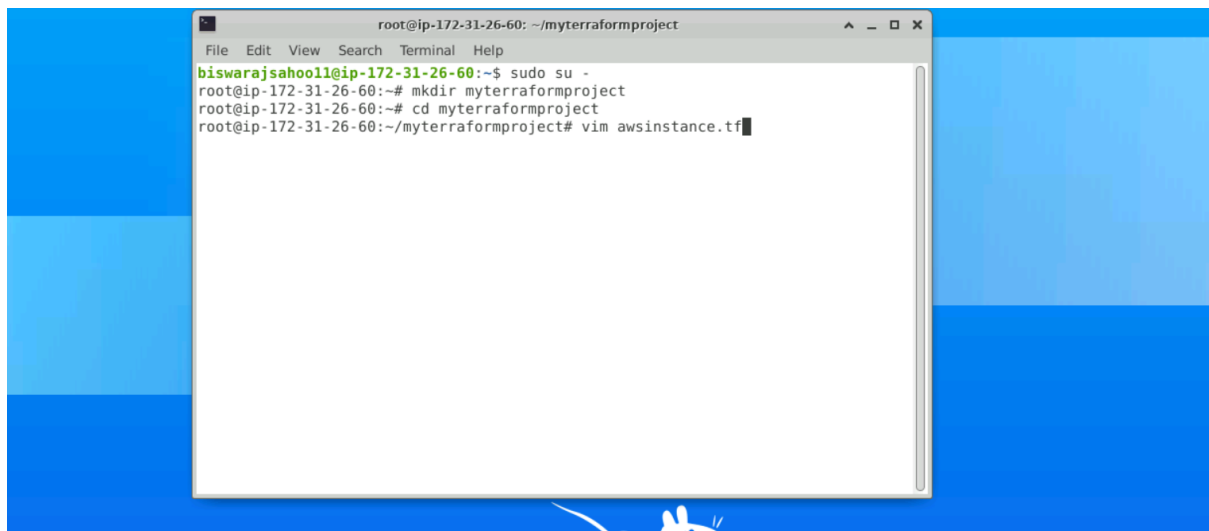


Access key and secret access key created

Connect to your lab terminal

Create a new directory (myterraformproject)

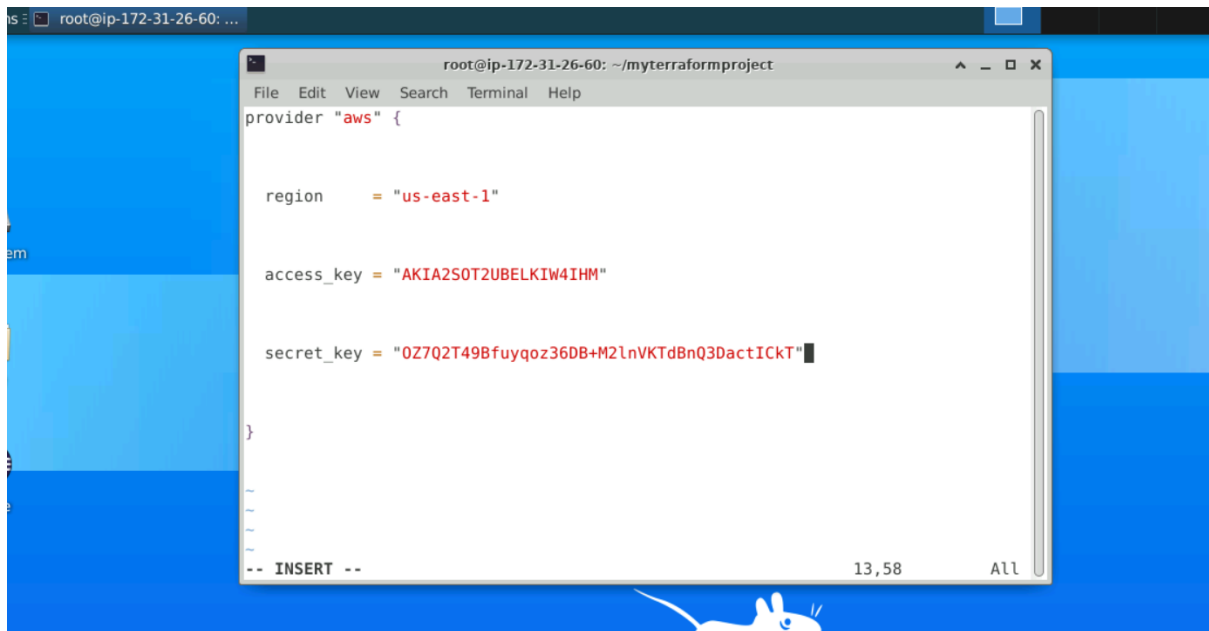
In this directory , we will mention AWS Infrastructure is our project code

A screenshot of a terminal window with a blue background. The terminal title bar reads 'root@ip-172-31-26-60: ~/myterraformproject'. The terminal content shows the following commands and output:

```
biswarajsahoo1@ip-172-31-26-60:~$ sudo su -
root@ip-172-31-26-60:~# mkdir myterraformproject
root@ip-172-31-26-60:~# cd myterraformproject
root@ip-172-31-26-60:~/myterraformproject# vim awsinstance.tf
```

The terminal window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. A white mouse cursor is visible at the bottom center of the terminal window.

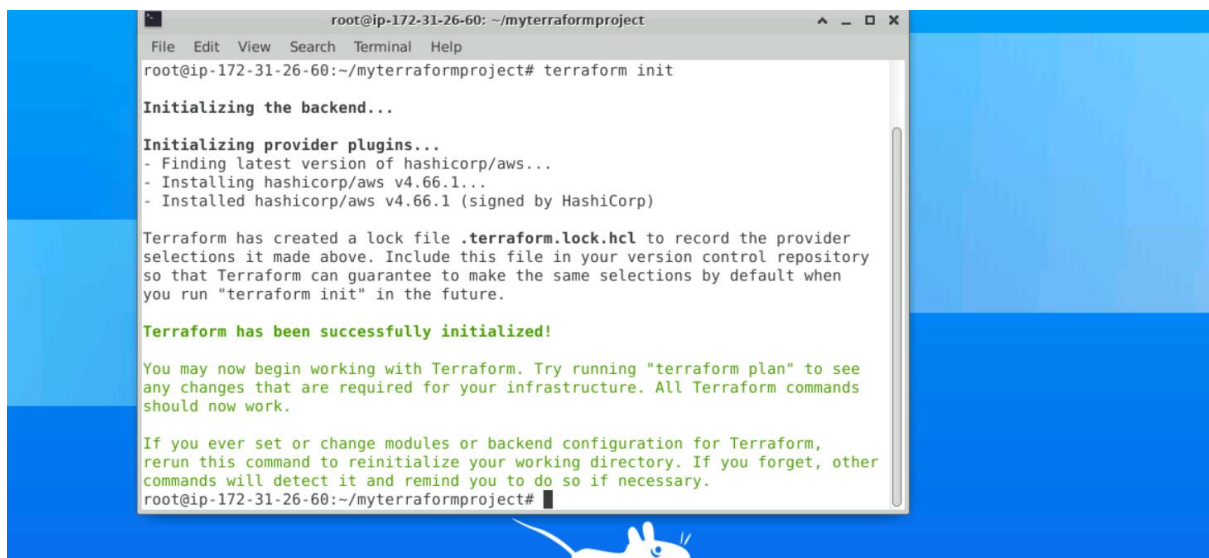
It's a provider block. We have to add region, access key secret access key



A screenshot of a terminal window titled "root@ip-172-31-26-60: ~/myterraformproject". The window shows the configuration for the AWS provider in a Terraform file. The configuration includes the provider name "aws", the region "us-east-1", and the access and secret keys. The terminal output shows the configuration being edited in a text editor.

```
provider "aws" {  
  
    region = "us-east-1"  
  
    access_key = "AKIA2SOT2UBELKIW4IHM"  
  
    secret_key = "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"  
}
```

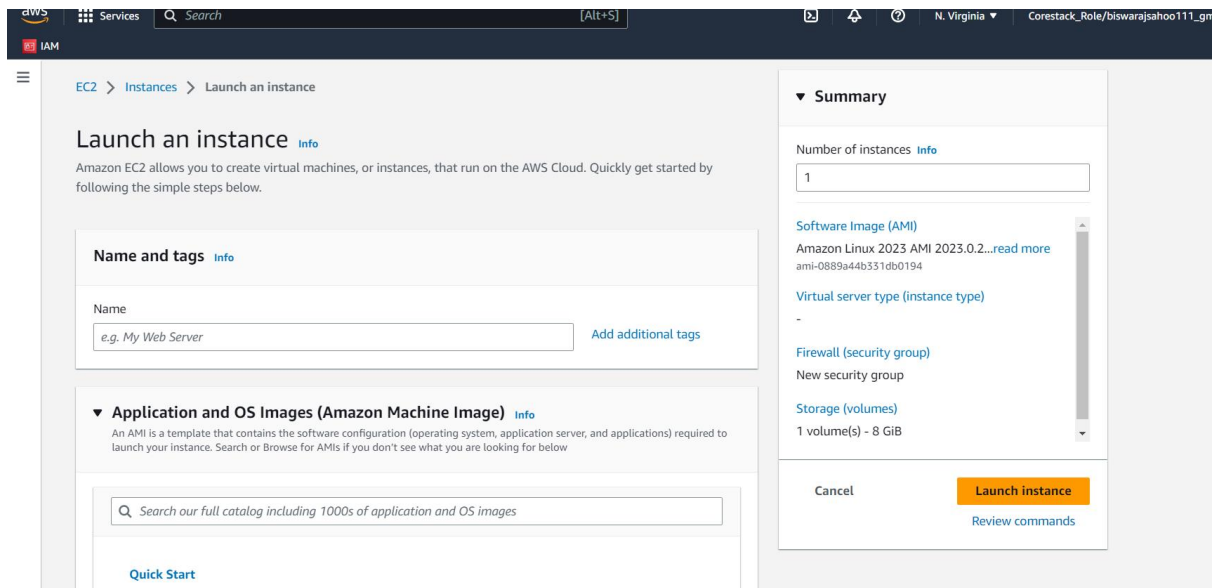
Then back to terminal and terraform init



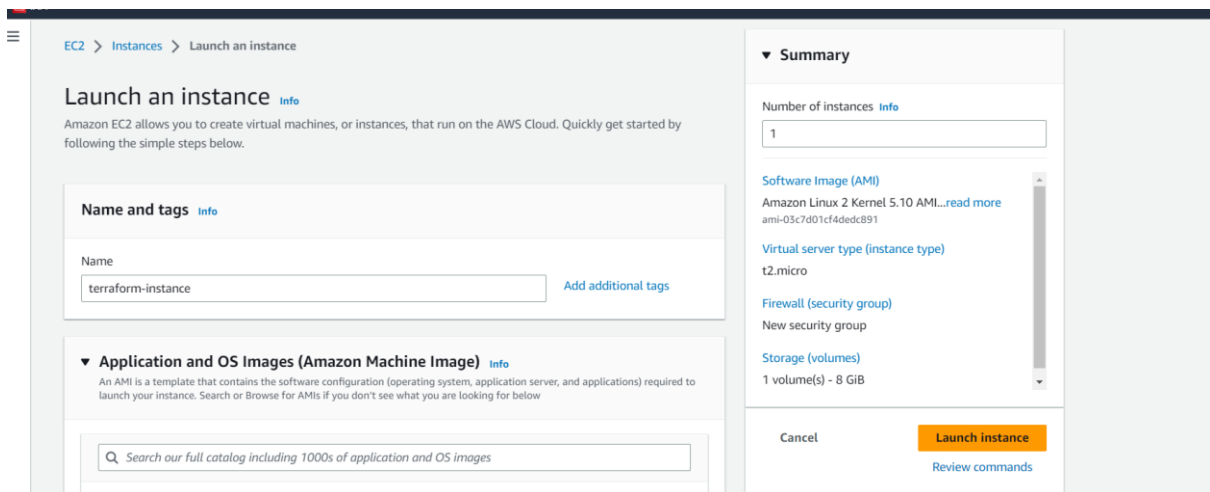
A screenshot of a terminal window titled "root@ip-172-31-26-60: ~/myterraformproject". The window shows the output of the "terraform init" command. The output indicates that the backend is being initialized, the AWS provider plugin is being installed, and the configuration is being saved to a lock file. The terminal output shows the command being executed and the resulting messages.

```
root@ip-172-31-26-60:~/myterraformproject# terraform init  
  
Initializing the backend...  
  
Initializing provider plugins...  
- Finding latest version of hashicorp/aws...  
- Installing hashicorp/aws v4.66.1...  
- Installed hashicorp/aws v4.66.1 (signed by HashiCorp)  
  
Terraform has created a lock file .terraform.lock.hcl to record the provider  
selections it made above. Include this file in your version control repository  
so that Terraform can guarantee to make the same selections by default when  
you run "terraform init" in the future.  
  
Terraform has been successfully initialized!  
  
You may now begin working with Terraform. Try running "terraform plan" to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.  
  
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.  
root@ip-172-31-26-60:~/myterraformproject#
```

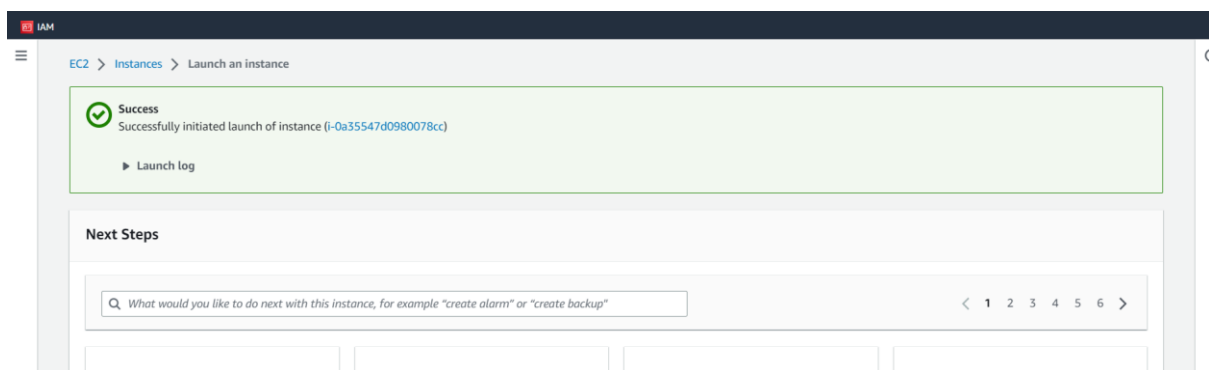
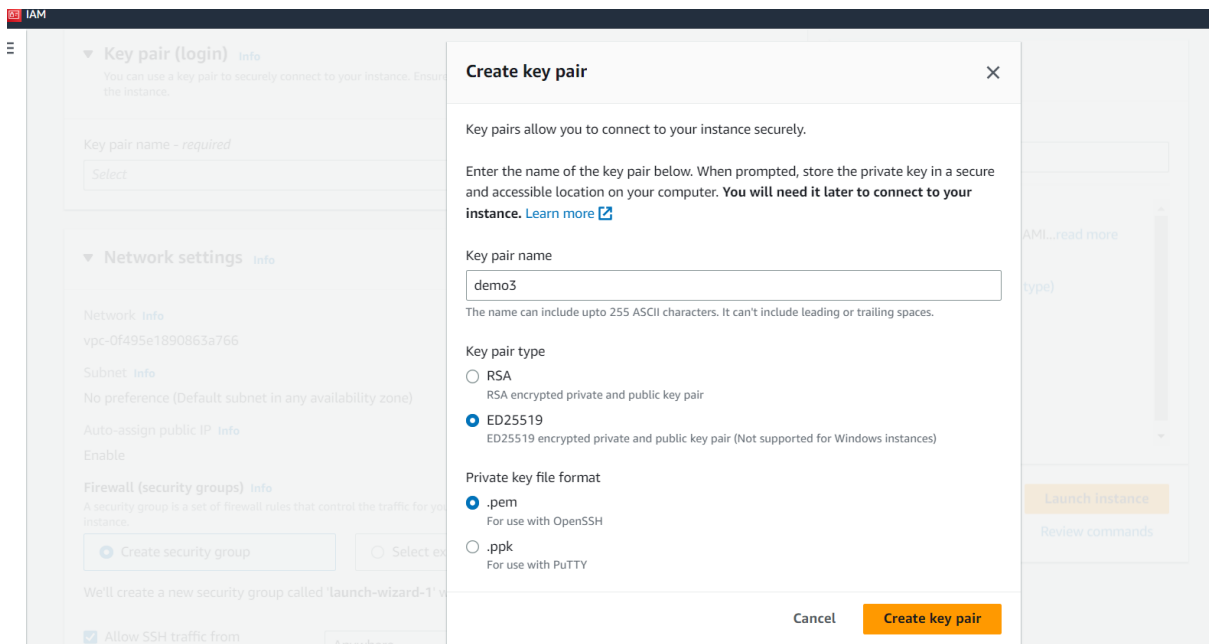
Now we are going to create EC2 instance and security group



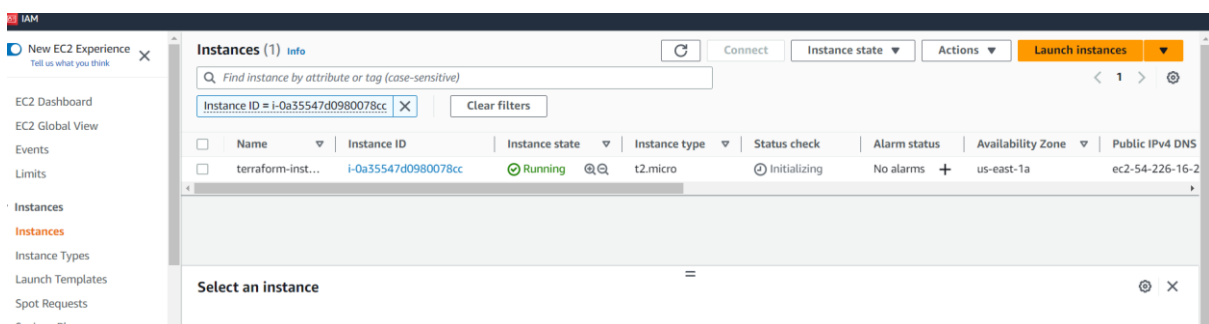
Instance name = terraform-instance



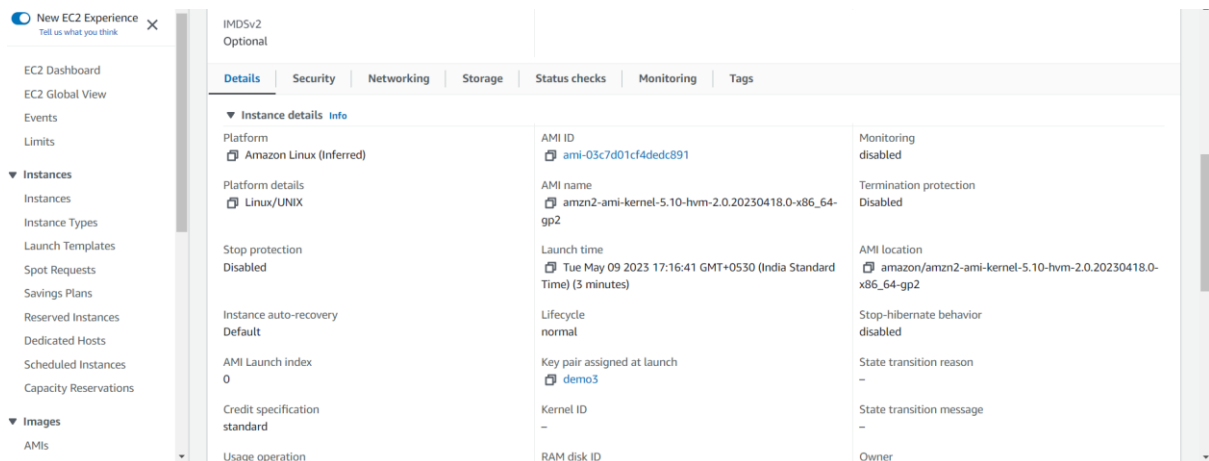
Create key pair



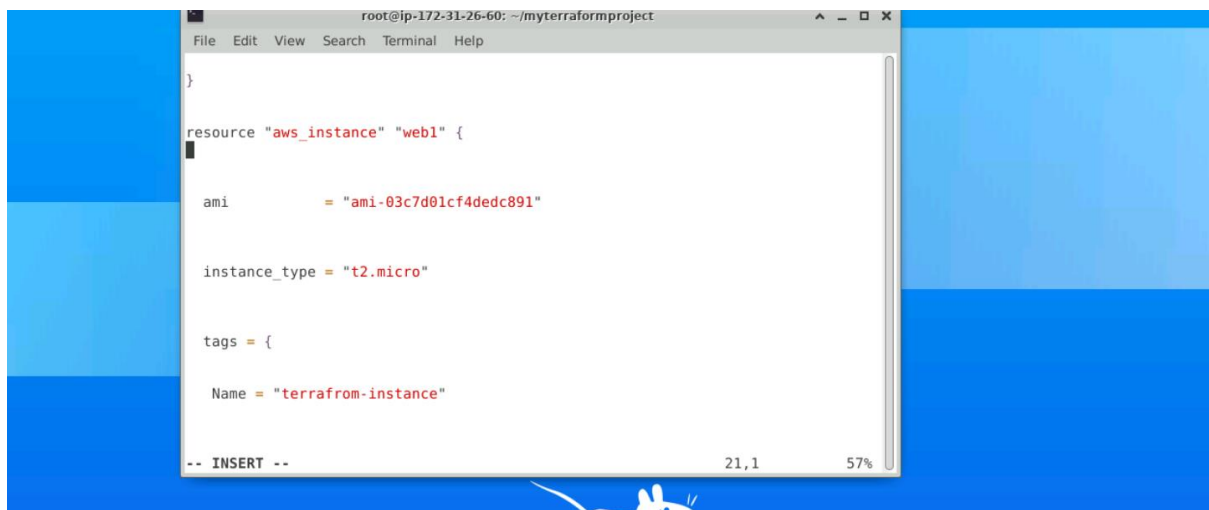
Launch instance successfully



Instance details



Now we will go to terminals and create terraform resource which will create AWS security group for attach with EC2 instance



Now Create a terraform resource which will create AWS security group rules(firewall rules) for the ec2 instance ,so that we can connect and install applications on it.

```
root@ip-172-31-26-60: ~/myterraformproject
File Edit View Search Terminal Help

}
resource "aws_security_group" "test3" {
  name      = "test3"
  description = "Allow inbound SSH"

  ingress {
    from_port      = 22
    to_port        = 22
    protocol        = "tcp"
    cidr_blocks     = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [ "::/0" ]
  }
}

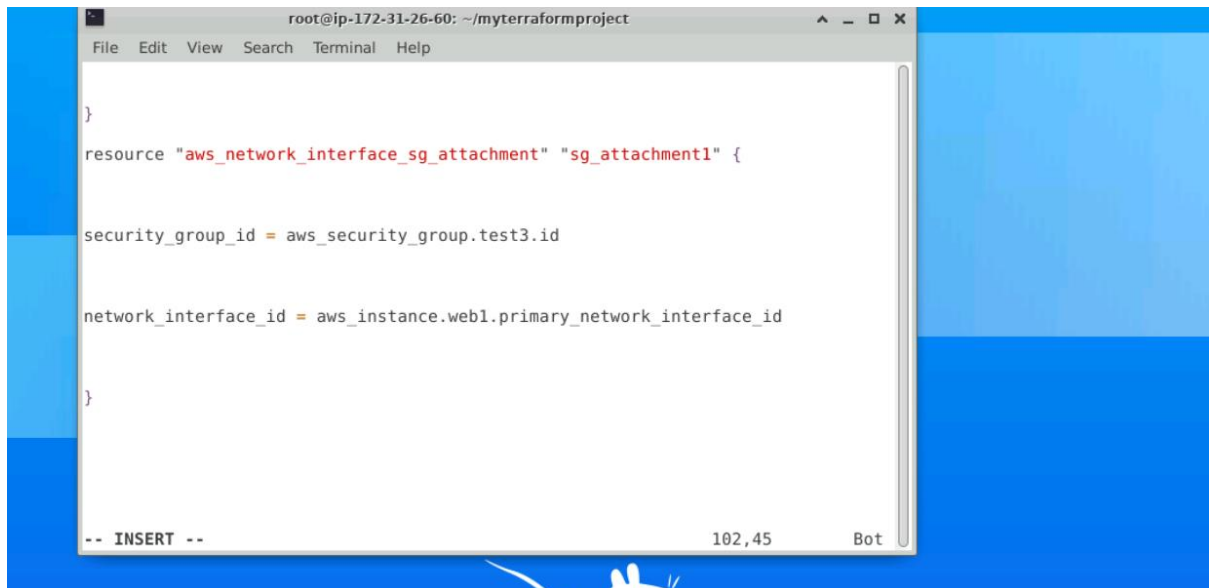
-- INSERT --                                     33,23      18%
```

```
root@ip-172-31-26-60: ~/myterraformproject
File Edit View Search Terminal Help

}
ingress {
  description = "HTTP"
  from_port   = 8080
  to_port     = 8080
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
egress {
  from_port = 0
  to_port   = 0
}

-- INSERT --                                     55,14      46%
```

Now we need to attach the security group to AWS instance .



```
root@ip-172-31-26-60: ~/myterraformproject
File Edit View Search Terminal Help

}
resource "aws_network_interface_sg_attachment" "sg_attachment1" {

security_group_id = aws_security_group.test3.id

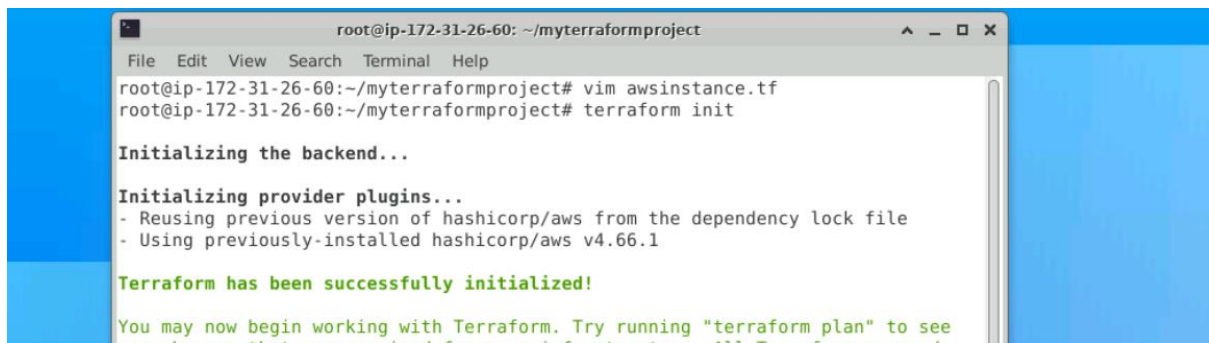
network_interface_id = aws_instance.web1.primary_network_interface_id

}

-- INSERT --                               102,45   Bot
```

Save the file.

Now terraform init



```
root@ip-172-31-26-60: ~/myterraformproject
File Edit View Search Terminal Help
root@ip-172-31-26-60:~/myterraformproject# vim awsinstance.tf
root@ip-172-31-26-60:~/myterraformproject# terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.66.1

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
```

Terraform initialise successfully

Now apply Terraform

```
File Edit View Search Terminal Help
+ from_port      = 8080
+ ipv6_cidr_blocks = []
+ prefix_list_ids = []
+ protocol       = "tcp"
+ security_groups = []
+ self           = false
+ to_port        = 8080
},
+ name           = "test3"
+ name_prefix    = (known after apply)
+ owner_id       = (known after apply)
+ revoke_rules_on_delete = false
+ tags_all       = (known after apply)
+ vpc_id         = (known after apply)
}

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: 
```

Type yes to perform action

```
File Edit View Search Terminal Help
}

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

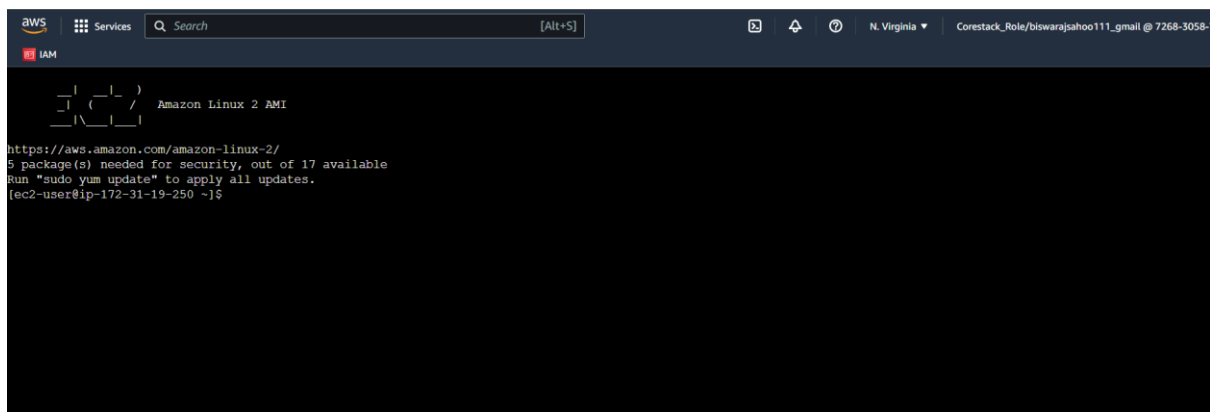
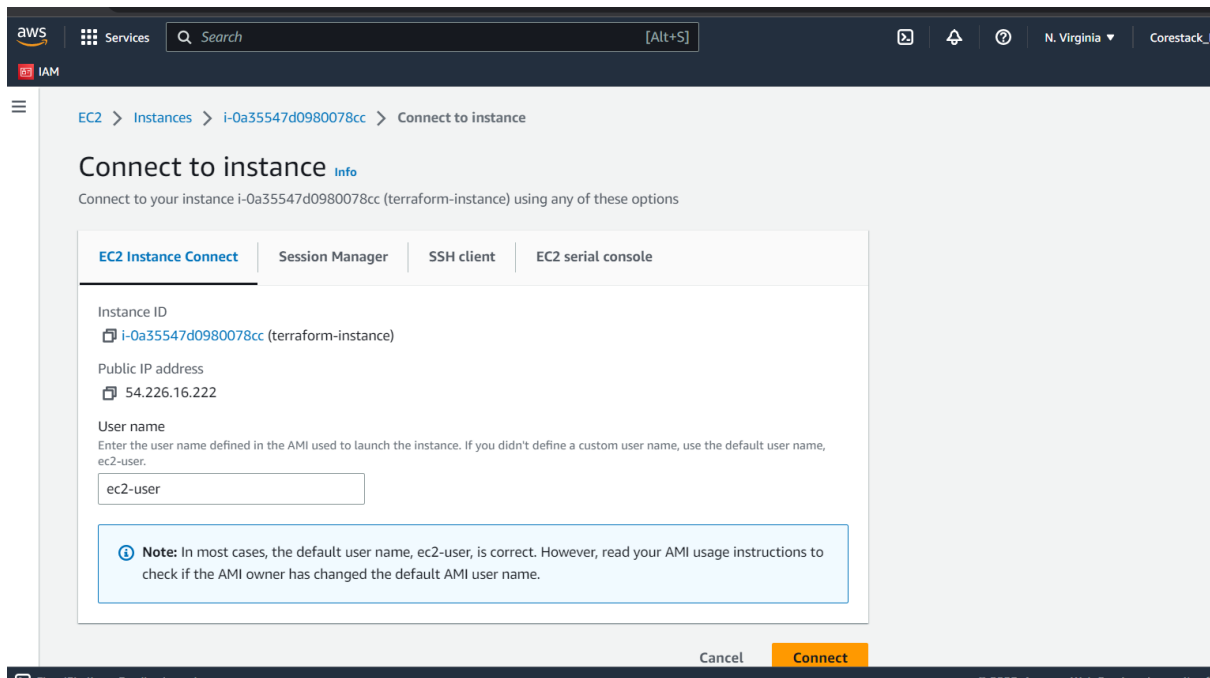
Enter a value: yes

aws_security_group.test3: Creating...
aws_instance.web1: Creating...
aws_security_group.test3: Creation complete after 3s [id=sg-0ef1971fb9899b7b6]
aws_instance.web1: Still creating... [10s elapsed]
aws_instance.web1: Still creating... [20s elapsed]
aws_instance.web1: Still creating... [30s elapsed]
aws_instance.web1: Still creating... [40s elapsed]
aws_instance.web1: Creation complete after 42s [id=i-02f1de36150bda886]
aws_network_interface_sg_attachment.sg_attachment1: Creating...
aws_network_interface_sg_attachment.sg_attachment1: Creation complete after 0s [id=sg-0ef1971fb9899b7b6_eni-08f53108e06bd5fae]

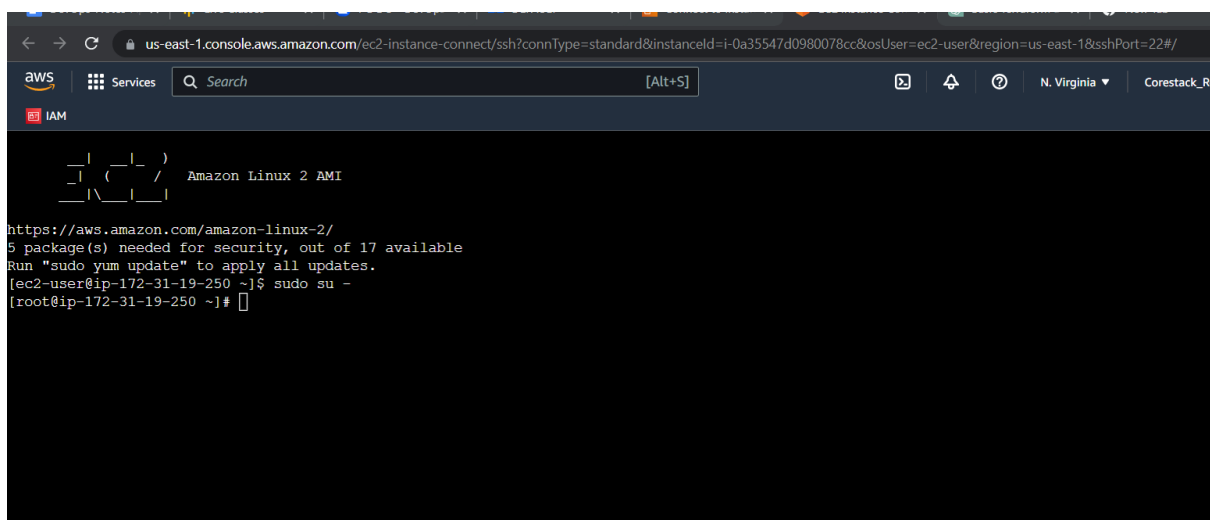
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
root@ip-172-31-26-60:~/myterraformproject# 
```

Terraform apply completion

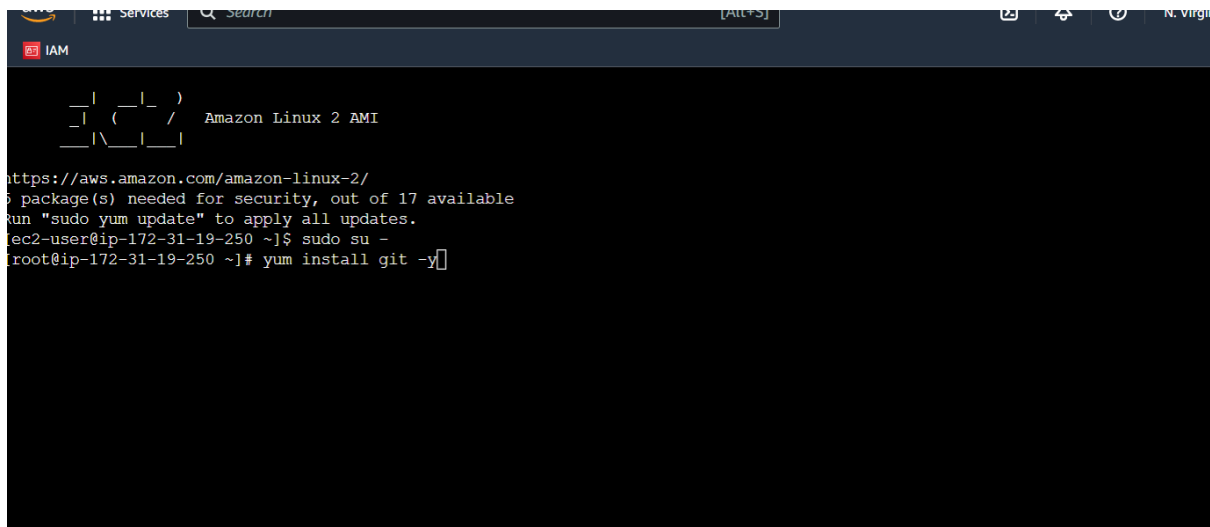
Now connect it to instance



Connected via browser

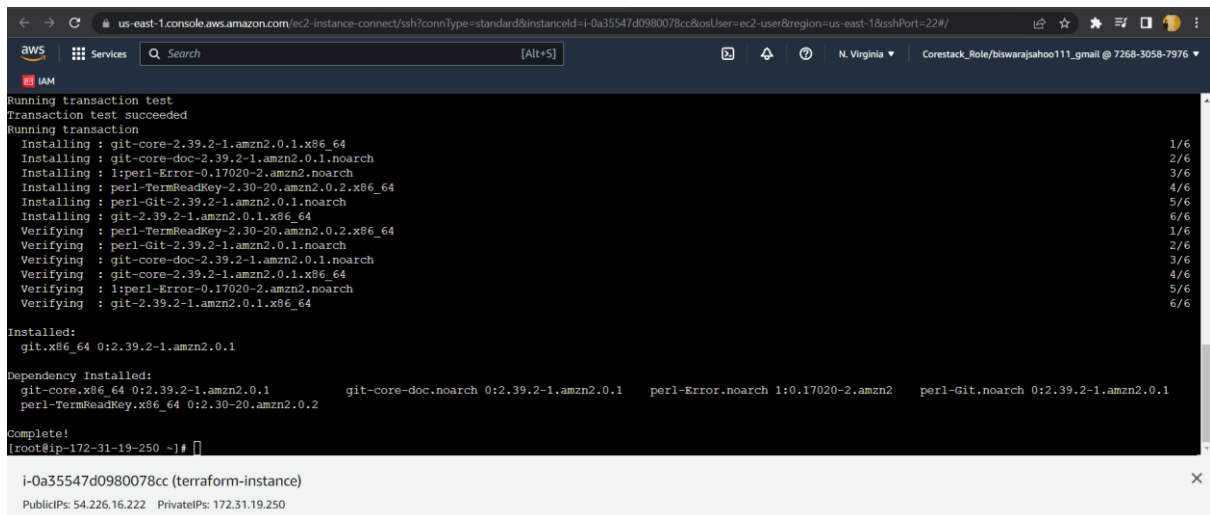


Now in EC2 we are going to install GIT



The screenshot shows the AWS IAM console with the Amazon Linux 2 AMI page. The page displays the Amazon Linux 2 logo and the URL <https://aws.amazon.com/amazon-linux-2/>. Below the URL, it states that 0 package(s) are needed for security, out of 17 available. It instructs to run "sudo yum update" to apply all updates. The terminal output shows the user running "sudo su -" and then "yum install git -y".

```
https://aws.amazon.com/amazon-linux-2/
0 package(s) needed for security, out of 17 available
Run "sudo yum update" to apply all updates.
ec2-user@ip-172-31-19-250 ~]$ sudo su -
root@ip-172-31-19-250 ~]# yum install git -y
```



The screenshot shows the AWS IAM console with the transaction test results for installing git. The terminal output displays the transaction test results, including the installation of git-core, git-core-doc, perl-Error, perl-TermReadKey, perl-Git, and git. The results show that the transaction test succeeded and the packages were installed successfully.

```
Running transaction test
Transaction test succeeded
Running transaction
Installing : git-core-2.39.2-1.amzn2.0.1.x86_64 1/6
Installing : git-core-doc-2.39.2-1.amzn2.0.1.noarch 2/6
Installing : 1:perl-Error-0.17020-2.amzn2.noarch 3/6
Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 4/6
Installing : perl-Git-2.39.2-1.amzn2.0.1.noarch 5/6
Installing : git-2.39.2-1.amzn2.0.1.x86_64 6/6
Verifying : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 1/6
Verifying : perl-Git-2.39.2-1.amzn2.0.1.noarch 2/6
Verifying : git-core-doc-2.39.2-1.amzn2.0.1.noarch 3/6
Verifying : git-core-2.39.2-1.amzn2.0.1.x86_64 4/6
Verifying : 1:perl-Error-0.17020-2.amzn2.noarch 5/6
Verifying : git-2.39.2-1.amzn2.0.1.x86_64 6/6
Installed:
git.x86_64 0:2.39.2-1.amzn2.0.1
Dependency Installed:
git-core.x86_64 0:2.39.2-1.amzn2.0.1 git-core-doc.noarch 0:2.39.2-1.amzn2.0.1 perl-Error.noarch 1:0.17020-2.amzn2 perl-Git.noarch 0:2.39.2-1.amzn2.0.1
perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2
Complete!
[root@ip-172-31-19-250 ~]#
```

Git installed

Now we are going to install python

Sudo yum install python

Dependencies Resolved

Package	Arch	Version	Repository	Size
Installing:				
java-1.8.0-openjdk-devel	x86_64	1:1.8.0.362.b08-1.amzn2.0.1	amzn2-core	9.8 M

Transaction Summary

Install 1 Package

Total download size: 9.8 M
Installed size: 40 M
Is this ok [y/d/N]: y

Downloading packages:

java-1.8.0-openjdk-devel-1.8.0.362.b08-1.amzn2.0.1.x86_64.rpm | 9.8 MB 00:00:00

Running transaction check
Running transaction test
Transaction test succeeded
Running transaction

Installing : 1:java-1.8.0-openjdk-devel-1.8.0.362.b08-1.amzn2.0.1.x86_64 1/1

i-0a35547d0980078cc (terraform-instance)

PublicIPs: 54.226.16.222 PrivateIPs: 172.31.19.250

Install 1 Package

Total download size: 94 M
Installed size: 94 M
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
jenkins-2.387.3-1.1.noarch.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction

Installing : jenkins-2.387.3-1.1.noarch
Verifying : jenkins-2.387.3-1.1.noarch

Installed:
jenkins.noarch 0:2.387.3-1.1

Complete!
[ec2-user@ip-172-31-19-250 ~]\$

Finished Dependency Resolution

Dependencies Resolved

Package	Arch	Version	Repository	Size
Installing:				
jenkins	noarch	2.404-1.1	jenkins	90 M

Transaction Summary

Install 1 Package

Total size: 90 M
Installed size: 90 M
Is this ok [y/d/N]: Exiting on user command

```
May 09 15:03:16 ip-172-31-19-250.ec2.internal jenkins[11354]: *****
May 09 15:03:16 ip-172-31-19-250.ec2.internal jenkins[11354]: *****
May 09 15:03:16 ip-172-31-19-250.ec2.internal jenkins[11354]: *****
May 09 15:03:43 ip-172-31-19-250.ec2.internal jenkins[11354]: 2023-05-09 15:03:43.054
May 09 15:03:43 ip-172-31-19-250.ec2.internal jenkins[11354]: 2023-05-09 15:03:43.088
May 09 15:03:43 ip-172-31-19-250.ec2.internal systemd[1]: Started Jenkins Continuous
May 09 15:03:43 ip-172-31-19-250.ec2.internal jenkins[11354]: 2023-05-09 15:03:43.217
May 09 15:03:43 ip-172-31-19-250.ec2.internal jenkins[11354]: 2023-05-09 15:03:43.218
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-19-250 ~]$ -l
-bash: -l: command not found
[ec2-user@ip-172-31-19-250 ~]$ jenkins --version
2.387.3
[ec2-user@ip-172-31-19-250 ~]$
```

Jenkins installed.

infrastructure automation is critical. We tend to put the most emphasis on software development processes, but infrastructure deployment strategy is just as important. Infrastructure automation not only aids disaster recovery, but it also facilitates testing and development.

Terraform is an open-source infrastructure as code (IAC) tool that can be used to automate the provisioning and management of infrastructure in the cloud. AWS provides a rich set of services that can be leveraged with Terraform to automate the deployment and management of infrastructure.