PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)

Semester: V **Name of Student: Paresh Gupta**
Academic Year: 2022-23 **Student ID:22104089**
Class / Branch: TE IT
Subject: Advanced Devops Lab (ADL)
Name of Instructor: Prof. Sonal Jain

EXPERIMENT NO. 06

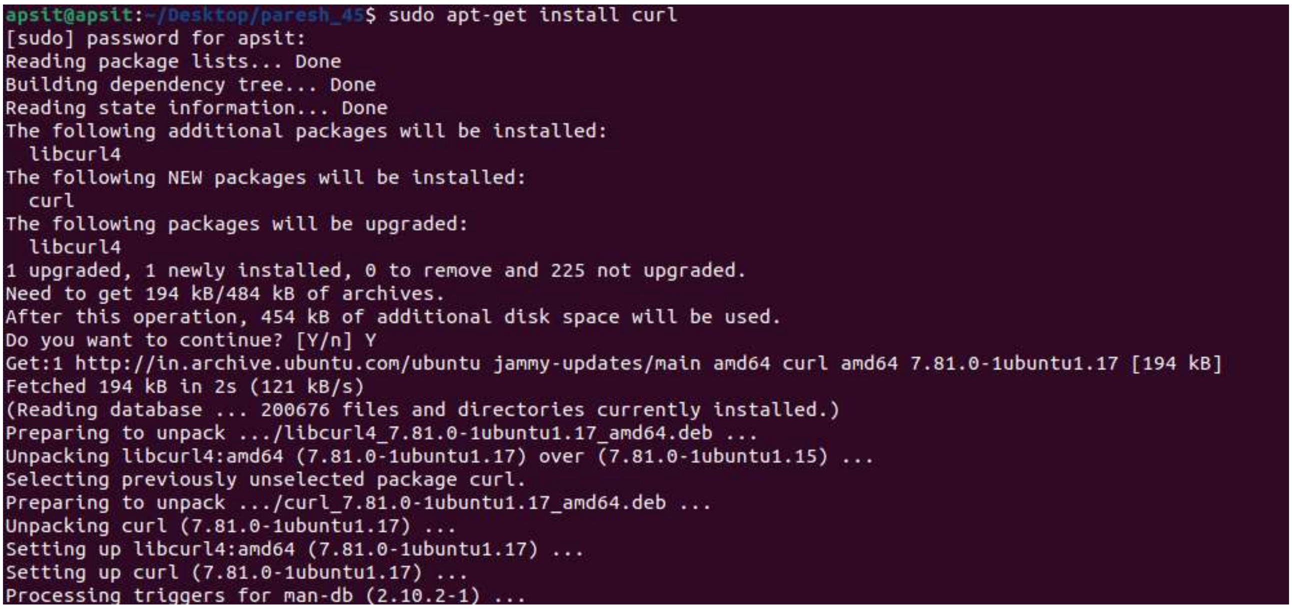
Aim: To Build, change, and destroy AWS infrastructure Using Terraform.

Pre-requistes:

1. Install the AWS CLI version 2 on Linux

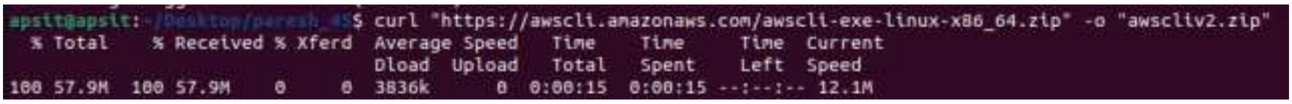
Follow these steps from the command line to install the AWS CLI on Linux.

Install curl on linux



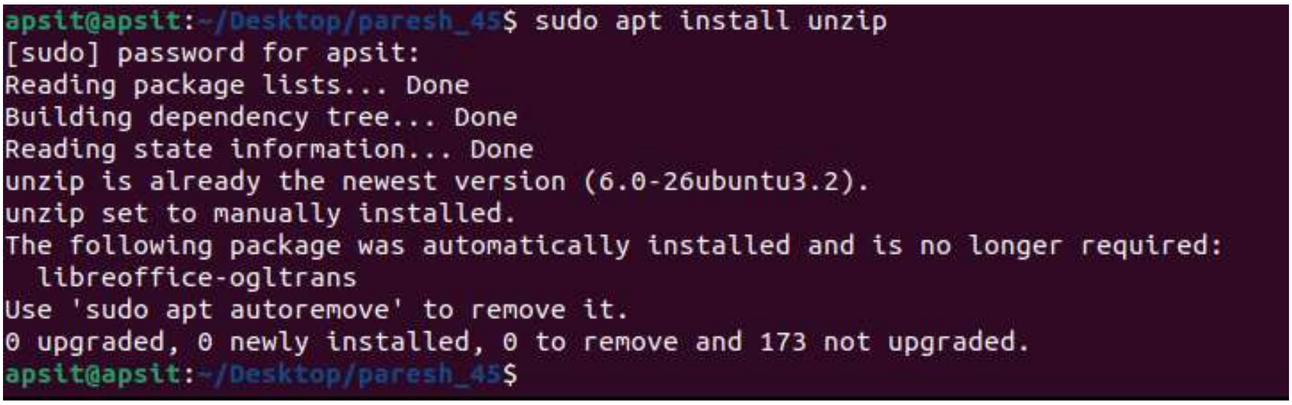
```
apsit@apsit:~/Desktop/paresh_45$ sudo apt-get install curl
[sudo] password for apsit:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl
The following packages will be upgraded:
  libcurl4
1 upgraded, 1 newly installed, 0 to remove and 225 not upgraded.
Need to get 194 kB/484 kB of archives.
After this operation, 454 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.17 [194 kB]
Fetched 194 kB in 2s (121 kB/s)
(Reading database ... 200676 files and directories currently installed.)
Preparing to unpack .../libcurl4_7.81.0-1ubuntu1.17_amd64.deb ...
Unpacking libcurl4:amd64 (7.81.0-1ubuntu1.17) over (7.81.0-1ubuntu1.15) ...
Selecting previously unselected package curl.
Preparing to unpack .../curl_7.81.0-1ubuntu1.17_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.17) ...
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.17) ...
Setting up curl (7.81.0-1ubuntu1.17) ...
Processing triggers for man-db (2.10.2-1) ...
```

vishal@apsit:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"



```
apsit@apsit:~/Desktop/paresh_45$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 57.9M  100 57.9M    0     0  3836k      0  0:00:15  0:00:15 --:--:-- 12.1M
```

vishal@apsit:~$ sudo apt install unzip



```
apsit@apsit:~/Desktop/paresh_45$ sudo apt install unzip
[sudo] password for apsit:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
unzip is already the newest version (6.0-26ubuntu3.2).
unzip set to manually installed.
The following package was automatically installed and is no longer required:
  libreoffice-ogltrans
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 173 not upgraded.
apsit@apsit:~/Desktop/paresh_45$
```

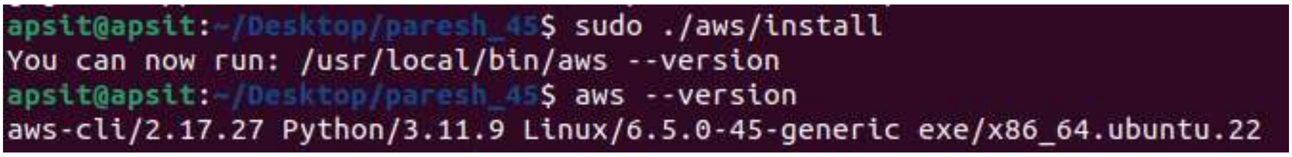
vishal@apsit:~$ sudo unzip awscliv2.zip

vishal@apsit:~$ sudo ./aws/install

vishal@apsit:~$ aws --version

it should display the below outout.

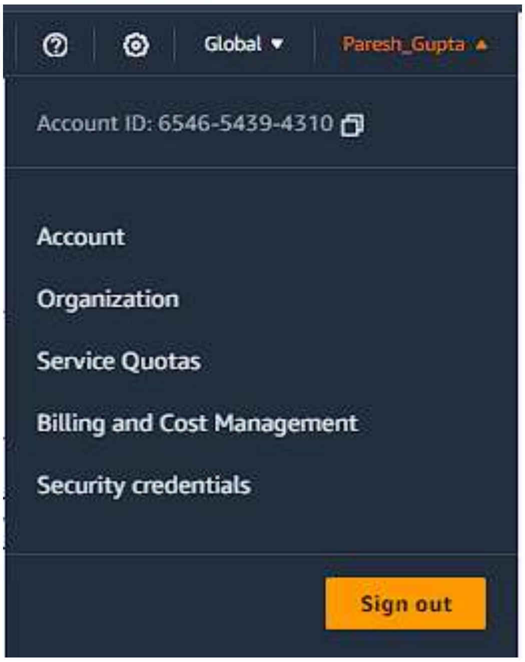
aws-cli/2.1.29 Python/3.8.8 Linux/5.4.0-1038-aws exe/x86_64.ubuntu.18 prompt/off



```
apsit@apsit:~/Desktop/paresh_45$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
apsit@apsit:~/Desktop/paresh_45$ aws --version
aws-cli/2.17.27 Python/3.11.9 Linux/6.5.0-45-generic exe/x86_64.ubuntu.22
```

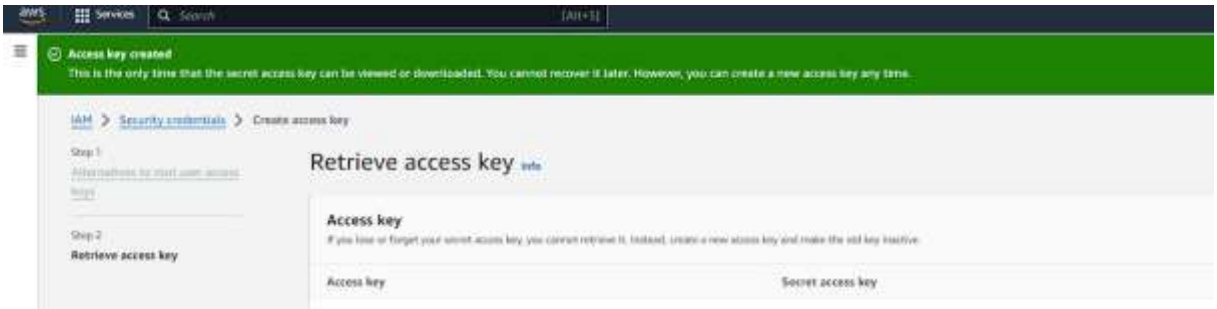
2. Create a new access key if you don't have one. Make sure you download the keys in your local machine.

Login to AWS console, click on username and go to My security credentials.



Account ID: 6546-5439-4310

Continue on security credentials, click on access keys


Services
Search
[Alt+S]
IAM > Security credentials > Create access key
Step 1
Alternatives to root user access keys
Step 2
Retrieve access key
Info

Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Retrieve access key

Access key

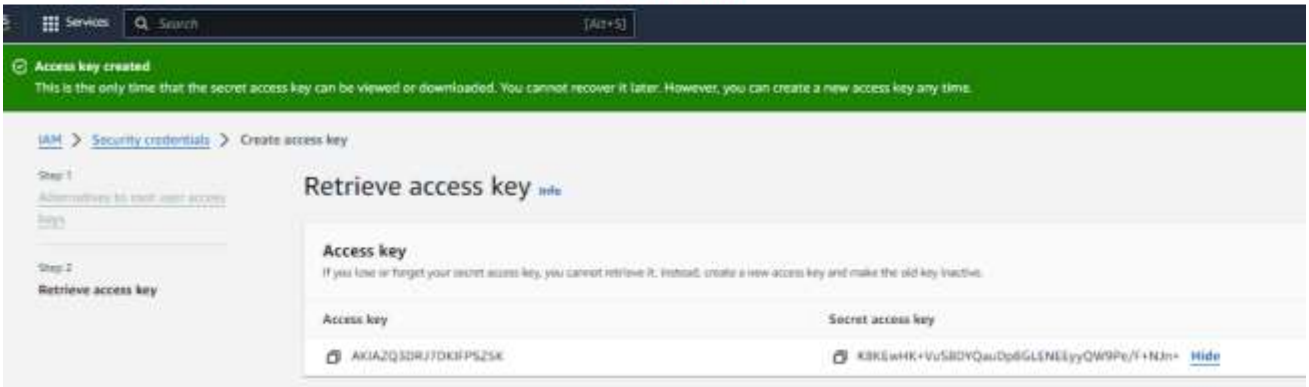
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key

Perform below commands in Linux where you have installed Terraform

First setup your access keys, secret keys and region code locally.

```
vishal@apsit:~$aws configure
```



Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Security credentials > Create access key

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

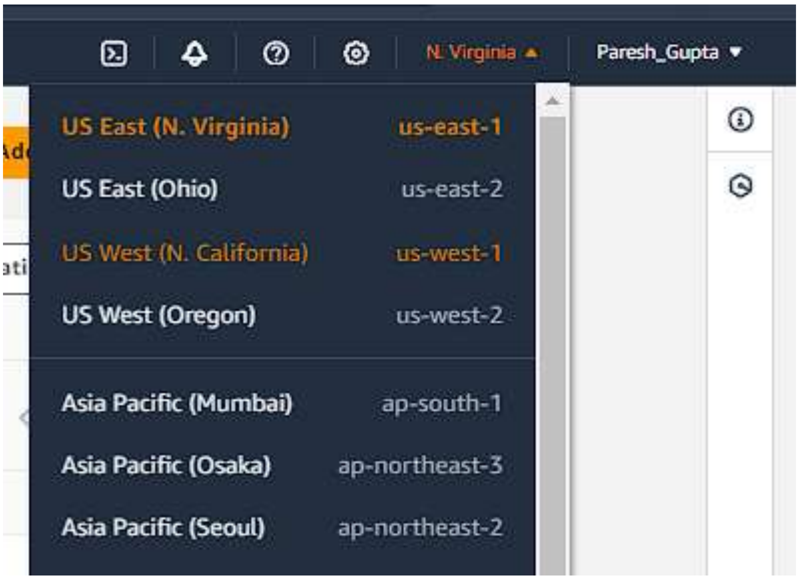
Retrieve access key Info

Access key

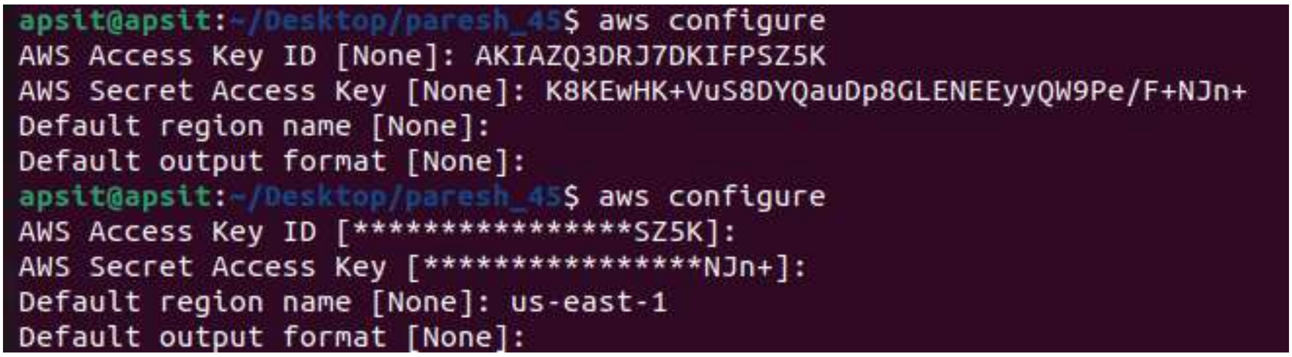
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIA2Q3DRJ7DKIFPSZSK	K8KEwHK+VuSBDYQauDp8GLENEEyyQW9Pe/Y+NJn+ Hide

You can check region as shown in below image :



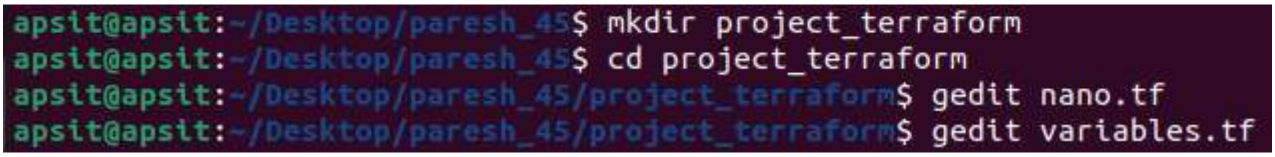
US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2



```
apsit@apsit:~/Desktop/paresh_45$ aws configure
AWS Access Key ID [None]: AKIAZQ3DRJ7DKIFPSZ5K
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
apsit@apsit:~/Desktop/paresh_45$ aws configure
AWS Access Key ID [****************SZ5K]:
AWS Secret Access Key [****************NJn+]:
Default region name [None]: us-east-1
Default output format [None]:
```

Create one Directory for Terraform project in which all files of terraform we can save

vishal@apsit:~$ cd ~
vishal@apsit:~$ mkdir project-terraform
vishal@apsit:~$ cd project-terraform

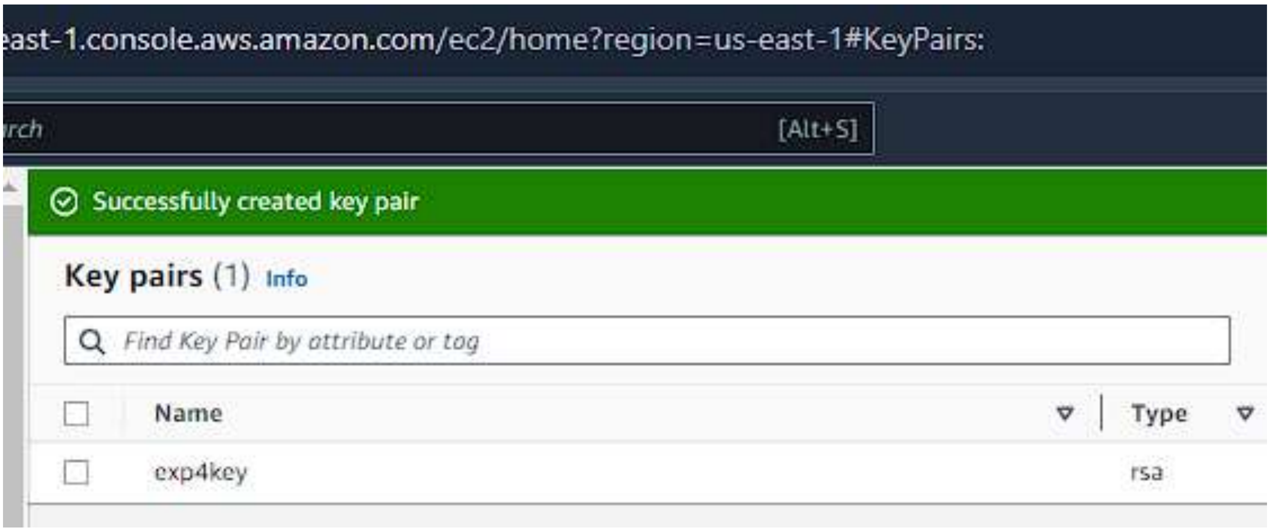


```
apsit@apsit:~/Desktop/paresh_45$ mkdir project_terraform
apsit@apsit:~/Desktop/paresh_45$ cd project_terraform
apsit@apsit:~/Desktop/paresh_45/project_terraform$ gedit nano.tf
apsit@apsit:~/Desktop/paresh_45/project_terraform$ gedit variables.tf
```

Create Terraform Files

```
vishal@apsit:~$ sudo nano variables.tf
```

In order to provide key name in variables first create key pair as shown:

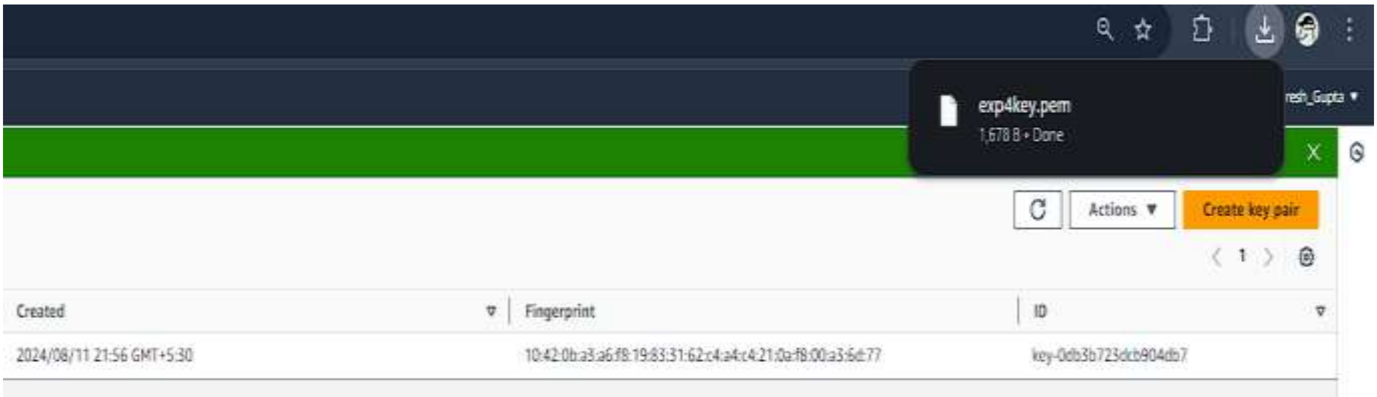


east-1.console.aws.amazon.com/ec2/home?region=us-east-1#KeyPairs:

Successfully created key pair

Key pairs (1) Info

	Name	Type
□	exp4key	rsa



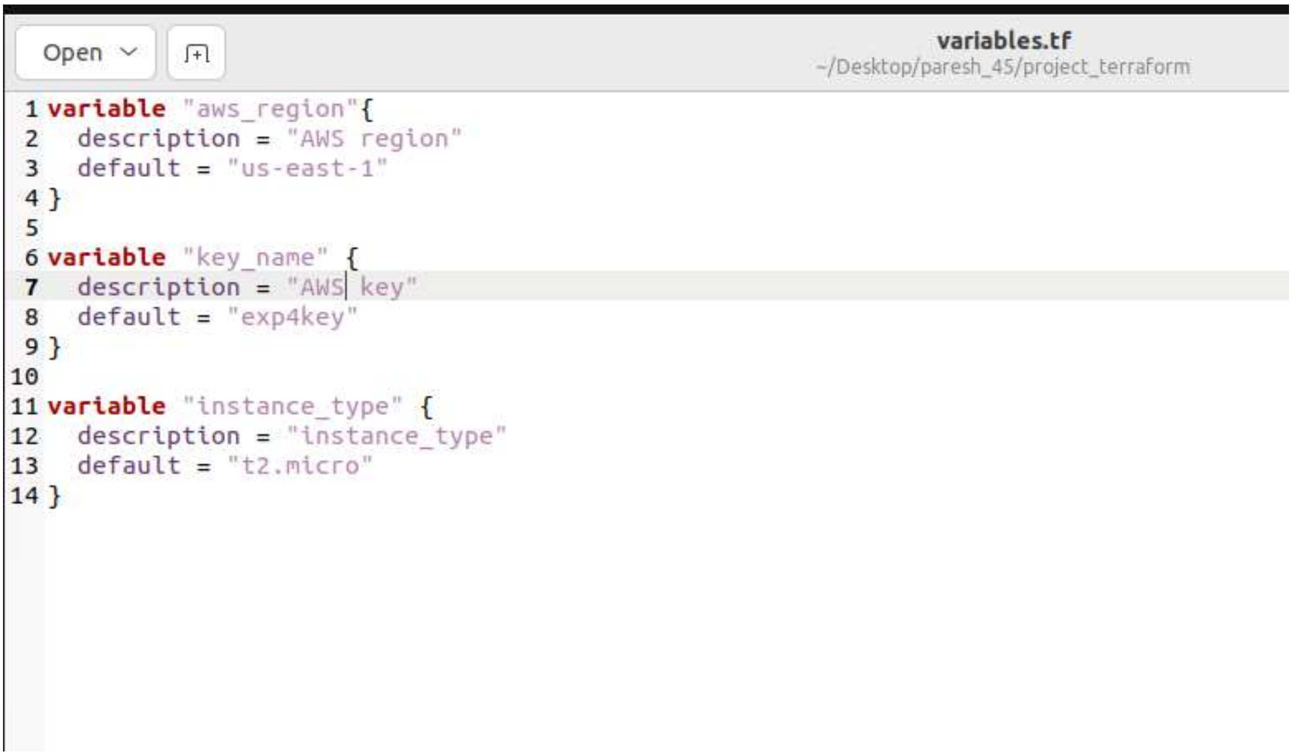
exp4key.pem
1,678 B • Done

Created	Fingerprint	ID
2024/08/11 21:56 GMT+5:30	10:42:0b:a3:a6:f8:19:83:31:62:c4:a4:c4:21:0a:f8:00:a3:6d:77	key-0db3b723dcb904db7

Give name to key pair file as **terraform**

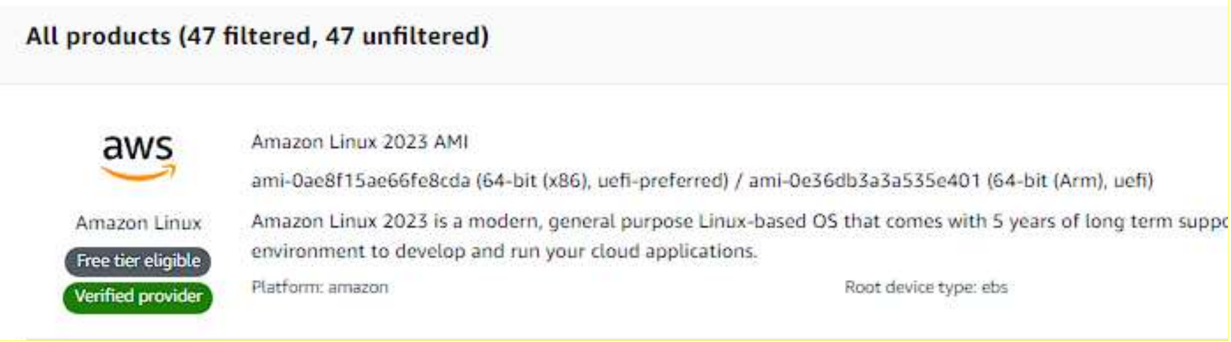
Key pair is generated

Use your Region and Key name in variable.tf as shown and provide instance type which you want to create.



```
variable "aws_region"{
  description = "AWS region"
  default = "us-east-1"
}

variable "key_name" {
  description = "AWS key"
  default = "exp4key"
}

variable "instance_type" {
  description = "instance_type"
  default = "t2.micro"
}
```

After creating variable terraform file note down the AMI ID of instance which u want to create which we will use to configure our instance in main.tf file.



All products (47 filtered, 47 unfiltered)

Amazon Linux 2023 AMI

ami-0ae8f15ae66fe8cda (64-bit (x86), uefi-preferred) / ami-0e36db3a3a535e401 (64-bit (Arm), uefi)

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term suppc environment to develop and run your cloud applications.

Platform: amazon

Root device type: ebs

<u>Now create main.tf file:</u>

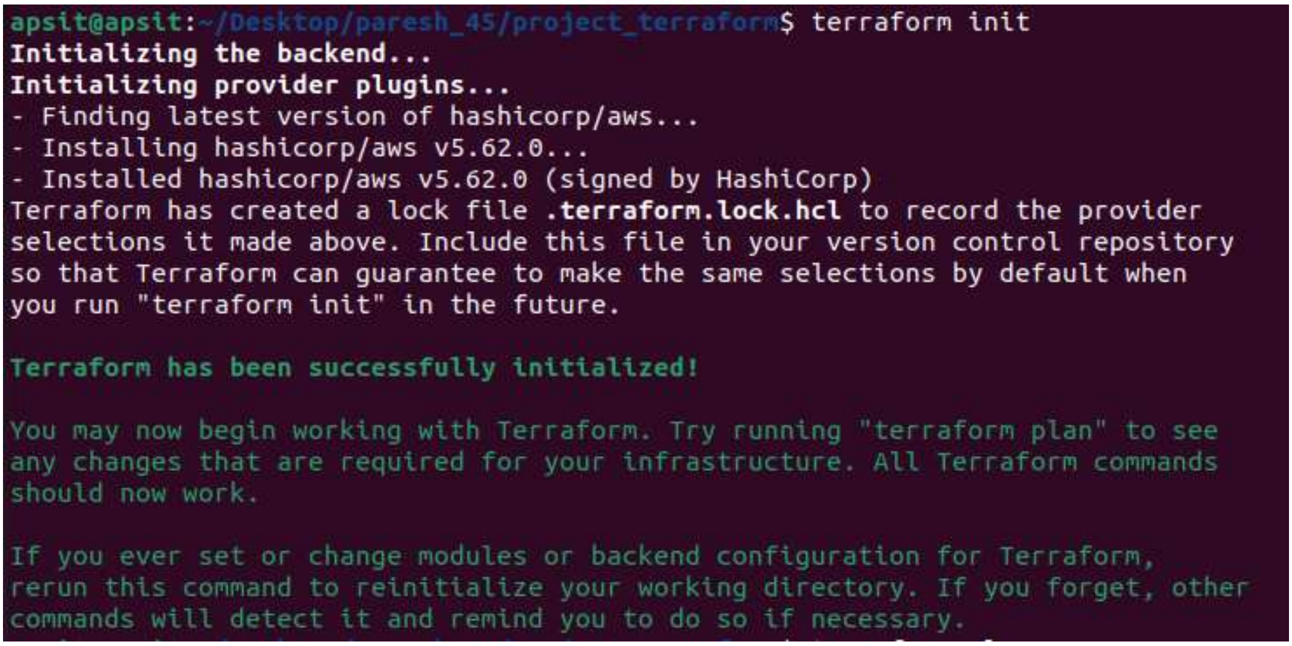
```
apsit@apsit:~/Desktop/paresh_45/project_terraform$ gedit main.tf
```

```
provider "aws" {
  region = var.aws_region
}


# Create security group with firewall rules
resource "aws_security_group" "security_jenkins_port" {
  name        = "security_jenkins_port"
  description = "security group for jenkins"
```

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

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

 # Outbound from Jenkins server
 egress {
   from_port   = 0
   to_port     = 65535
   protocol    = "tcp"
   cidr_blocks = ["0.0.0.0/0"]
 }

 tags = {
   Name = "security_jenkins_port"
 }
}
```

```
resource "aws_instance" "myFirstInstance" {
  ami           = "ami-0ae8f15ae66fe8cda"
  key_name      = var.key_name
  instance_type = var.instance_type
  vpc_security_group_ids = [aws_security_group.security_jenkins_port.id] #changed here
  tags = {
    Name = "jenkins_instance"
  }
}

# Create Elastic IP address
resource "aws_eip" "myFirstInstance" {
  instance = aws_instance.myFirstInstance.id
  domain   = "vpc"
  tags = {
    Name = "jenkins_elstic_ip"
  }
}
```

Put AMI-ID in above highlighted space and Now execute the below command:

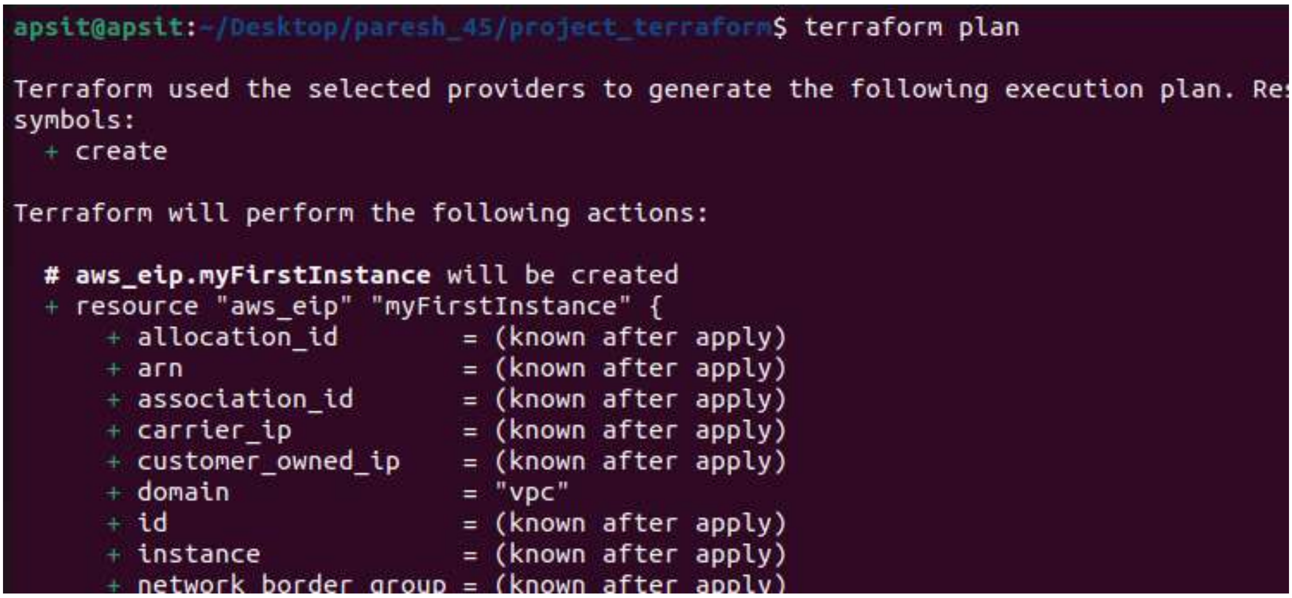


```
apsit@apsit:~/Desktop/paresh_45/project_terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.62.0...
- Installed hashicorp/aws v5.62.0 (signed by HashiCorp)
Terraform has created a lock file .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.
```

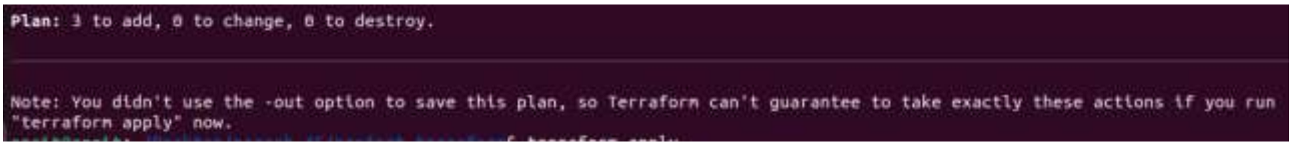
you should see like below screenshot.

Execute the below command

the above command will show how many resources will be added.
Plan: 3 to add, 0 to change, 0 to destroy.

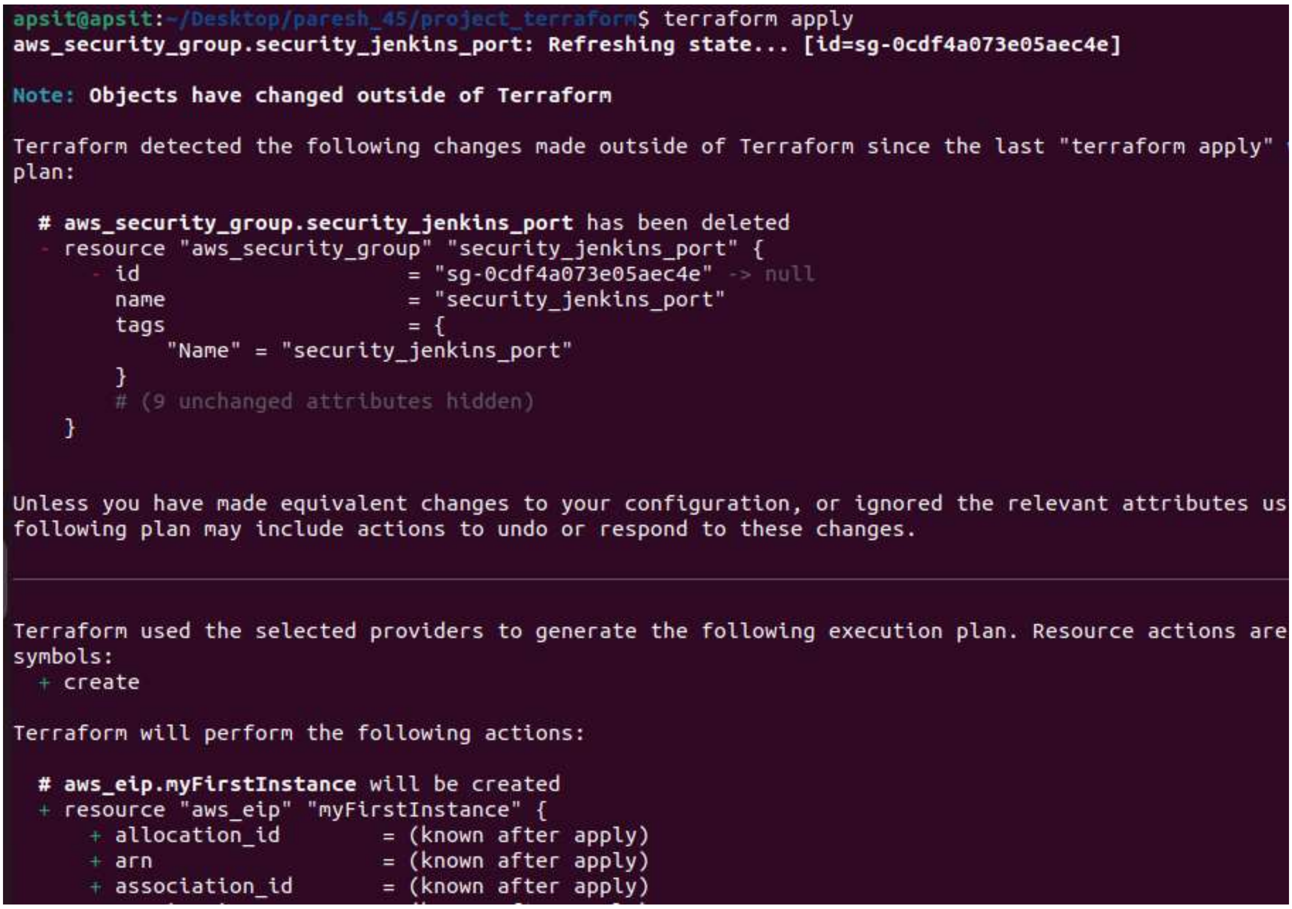


```
apsit@apsit:~/Desktop/paresh_45/project_terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan. Re
symbols:
  + create

Terraform will perform the following actions:

  # aws_eip.myFirstInstance will be created
  + resource "aws_eip" "myFirstInstance" {
      + allocation_id        = (known after apply)
      + arn                  = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = "vpc"
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
```



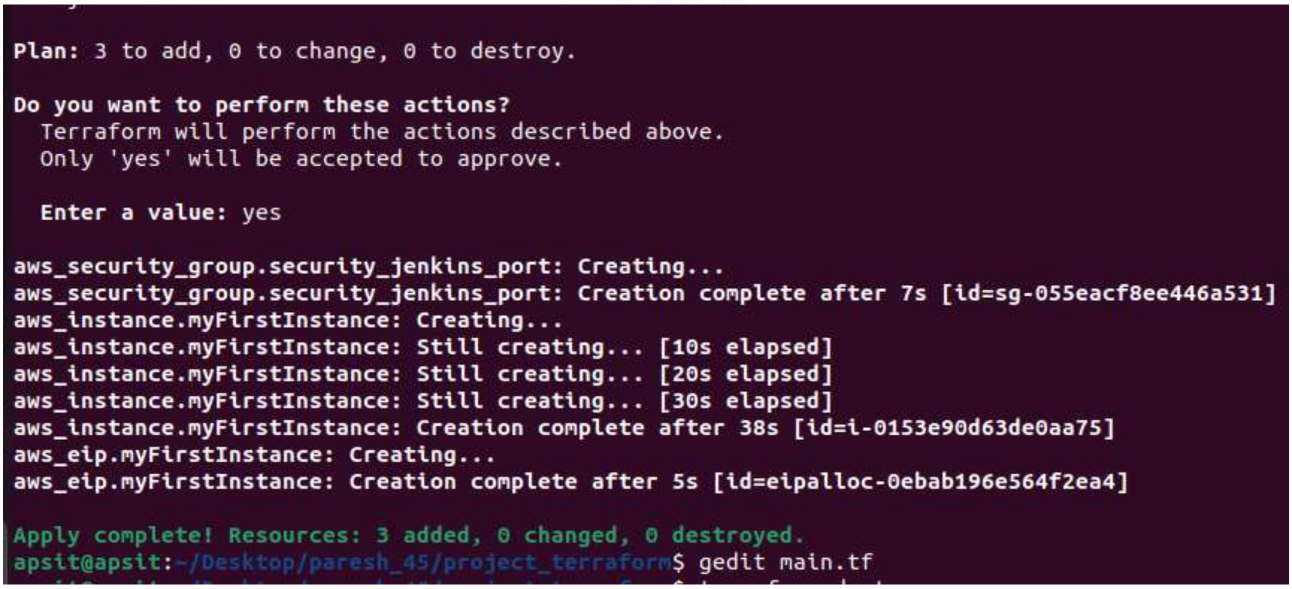
```
Plan: 3 to add, 0 to change, 0 to destroy.

───────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run
"terraform apply" now.
```

Execute the below command



```
apsit@apsit:~/Desktop/paresh_45/project_terraform$ terraform apply
aws_security_group.security_jenkins_port: Refreshing state... [id=sg-0cdf4a073e05aec4e]

Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform since the last "terraform apply"
plan:

  # aws_security_group.security_jenkins_port has been deleted
  - resource "aws_security_group" "security_jenkins_port" {
      - id                      = "sg-0cdf4a073e05aec4e" -> null
        name                    = "security_jenkins_port"
        tags                    = {
            "Name" = "security_jenkins_port"
        }
        # (9 unchanged attributes hidden)
    }


Unless you have made equivalent changes to your configuration, or ignored the relevant attributes us
following plan may include actions to undo or respond to these changes.

───────────────────────────────────────────────────────────────────────

Terraform used the selected providers to generate the following execution plan. Resource actions are
symbols:
  + create

Terraform will perform the following actions:

  # aws_eip.myFirstInstance will be created
  + resource "aws_eip" "myFirstInstance" {
      + allocation_id        = (known after apply)
      + arn                  = (known after apply)
      + association_id       = (known after apply)
```

Provide the value as Yes for applying terraform



```
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.security_jenkins_port: Creating...
aws_security_group.security_jenkins_port: Creation complete after 7s [id=sg-055eacf8ee446a531]
aws_instance.myFirstInstance: Creating...
aws_instance.myFirstInstance: Still creating... [10s elapsed]
aws_instance.myFirstInstance: Still creating... [20s elapsed]
aws_instance.myFirstInstance: Still creating... [30s elapsed]
aws_instance.myFirstInstance: Creation complete after 38s [id=i-0153e90d63de0aa75]
aws_eip.myFirstInstance: Creating...
aws_eip.myFirstInstance: Creation complete after 5s [id=eipalloc-0ebab196e564f2ea4]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
apsit@apsit:~/Desktop/paresh_45/project_terraform$ gedit main.tf
```

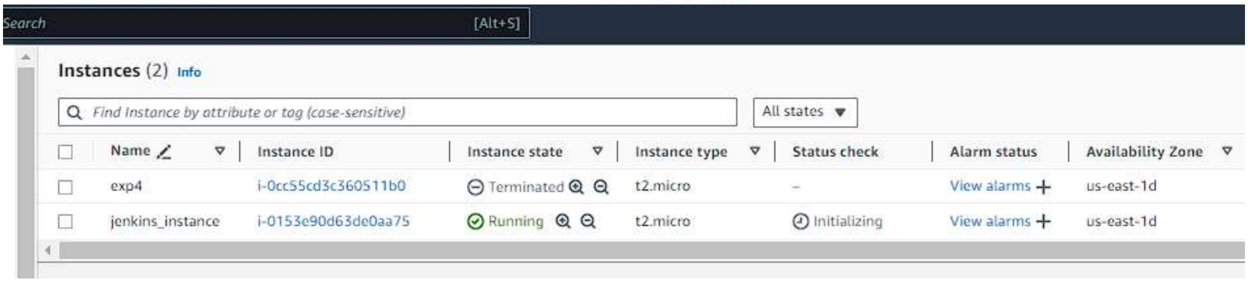
Plan: 3 to add, 0 to change, 0 to destroy.

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

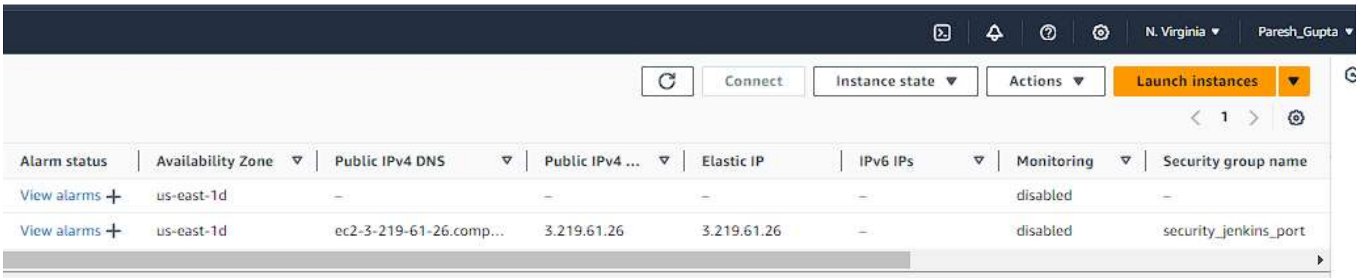
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Now login to EC2 console, to see the new instances up and running, you can see Jenkins_instance is up and running which we deploy from terraform.

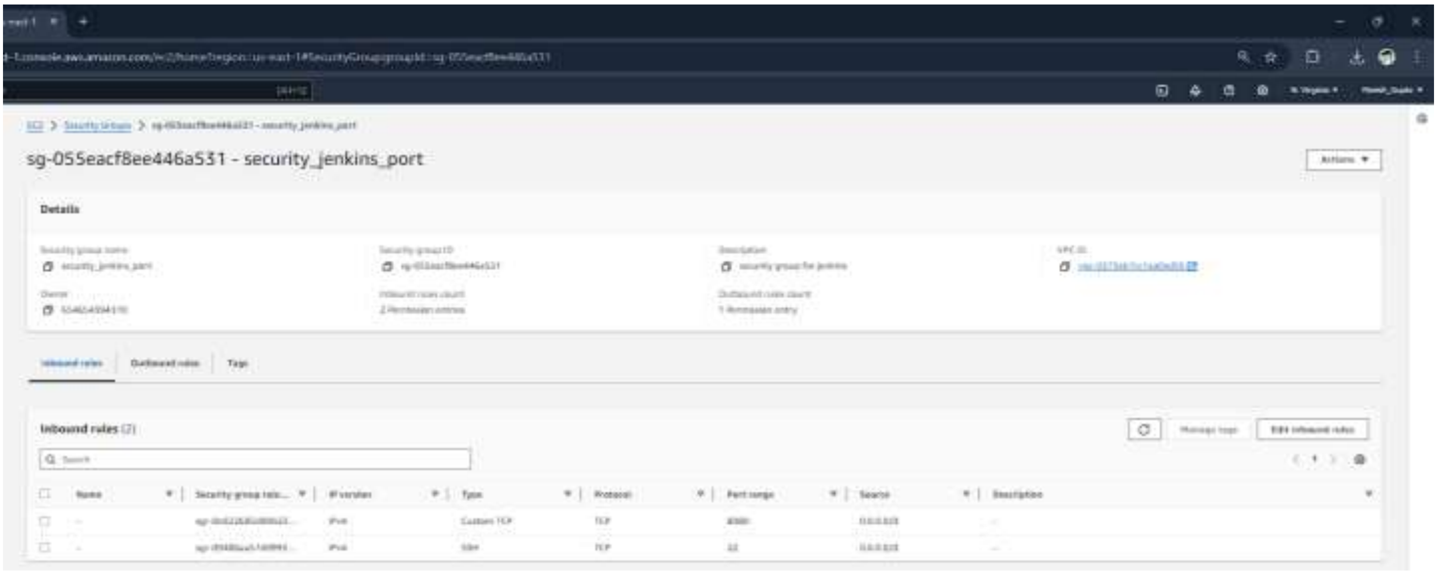


Instances (2) Info

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
□	exp4	i-0cc55cd3c360511b0	Terminated	t2.micro	–	View alarms +	us-east-1d
□	jenkins_instance	i-0153e90d63de0aa75	Running	t2.micro	Initializing	View alarms +	us-east-1d



Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security group name
View alarms +	us-east-1d	–	–	–	–	disabled	–
View alarms +	us-east-1d	ec2-3-219-61-26.comp...	3.219.61.26	3.219.61.26	–	disabled	security_jenkins_port



sg-055eacf8ee446a531 - security_jenkins_port

Details

Security group name: security_jenkins_port
Security group ID: sg-055eacf8ee446a531
Description: security group for jenkins
VPC ID: [illegible]
Owner: [illegible]
Inbound rules count: 2 Permission entries
Outbound rules count: 1 Permission entry

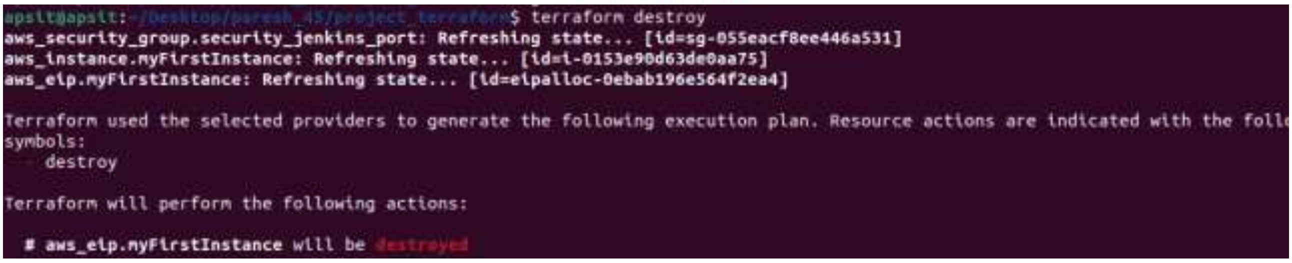
Inbound rules (2)

□	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
□	–	[illegible]	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	–
□	–	[illegible]	IPv4	SSH	TCP	22	0.0.0.0/0	–

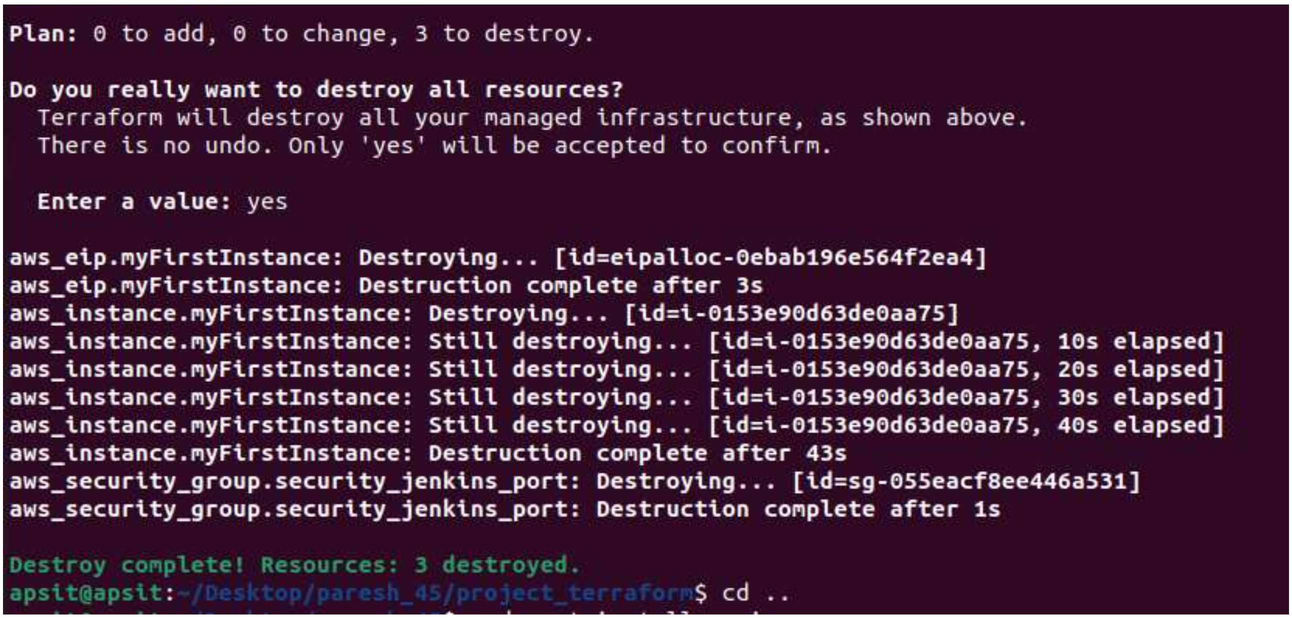
You can also check the security group resource details which you created from terraform :

Terraform destroy

you can also destroy or delete your instance by using terraform destroy command :

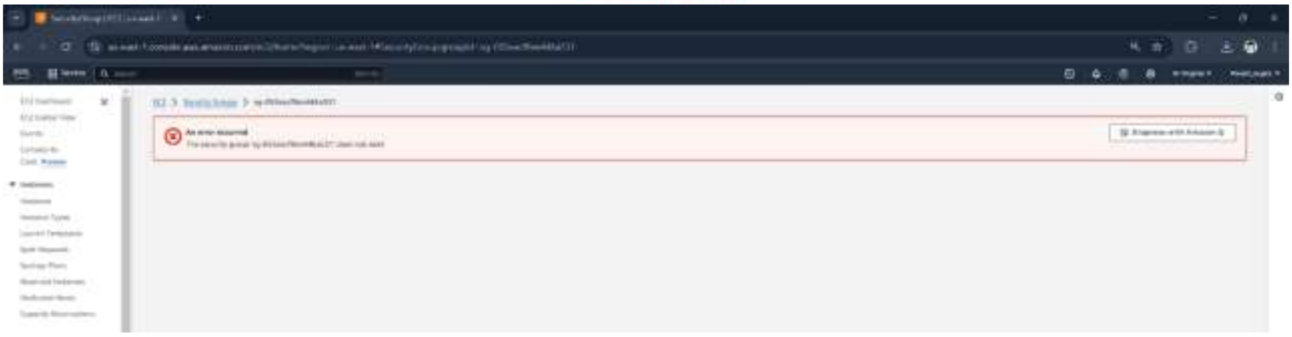


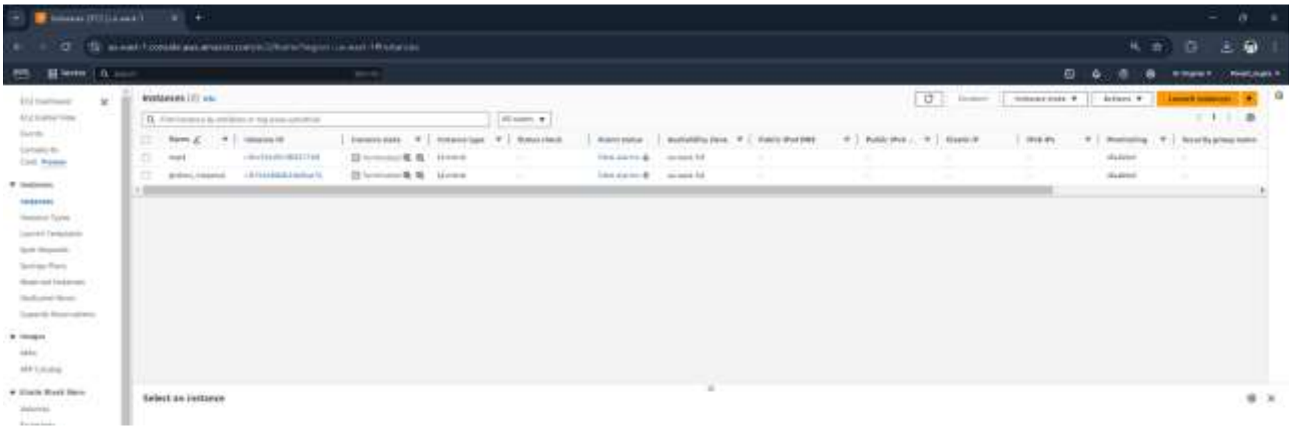
```
apsit@apsit:~/Desktop/paresh_45/project_terraform$ terraform destroy
aws_security_group.security_jenkins_port: Refreshing state... [id=sg-055eacf8ee446a531]
aws_instance.myFirstInstance: Refreshing state... [id=i-0153e90d63de0aa75]
aws_eip.myFirstInstance: Refreshing state... [id=eipalloc-0ebab196e564f2ea4]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the foll
symbols:
  - destroy

Terraform will perform the following actions:

  # aws_eip.myFirstInstance will be destroyed
```



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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_eip.myFirstInstance: Destroying... [id=eipalloc-0ebab196e564f2ea4]
aws_eip.myFirstInstance: Destruction complete after 3s
aws_instance.myFirstInstance: Destroying... [id=i-0153e90d63de0aa75]
aws_instance.myFirstInstance: Still destroying... [id=i-0153e90d63de0aa75, 10s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0153e90d63de0aa75, 20s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0153e90d63de0aa75, 30s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0153e90d63de0aa75, 40s elapsed]
aws_instance.myFirstInstance: Destruction complete after 43s
aws_security_group.security_jenkins_port: Destroying... [id=sg-055eacf8ee446a531]
aws_security_group.security_jenkins_port: Destruction complete after 1s

Destroy complete! Resources: 3 destroyed.
apsit@apsit:~/Desktop/paresh_45/project_terraform$ cd ..
```

Now you can see instance which you created by using terraform is deleted successfully from aws console also you can check it will removed successfully:





All the Resources including Security groups, EC2 instances using terraform will be deleted. In this way we can automate infrastructure set up using terrform in aws cloud.

Conclusion: Hence we had successfully created EC2 Instance and Security Group without AWS GUI using Terraform through Linux Terminal.