Semester: V
Academic Year: 2024-25
Class / Branch: TE IT
Subject: Advanced Devops Lab (ADL)
Name of Instructor:Prof. Manjusha K.

Name of Student:Mustqeem Masuldar
Student ID:22104024

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

```
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~$ sudo apt-get install curl
```

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

```
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads$ 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  59.0M      0 --:--:-- --:--:-- --:--:-- 58.9M
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads$
```

```
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads$ sudo apt install unzip
```

vishal@apsit:~$ sudo unzip awscliv2.zip

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

vishal@apsit:~$ aws --version

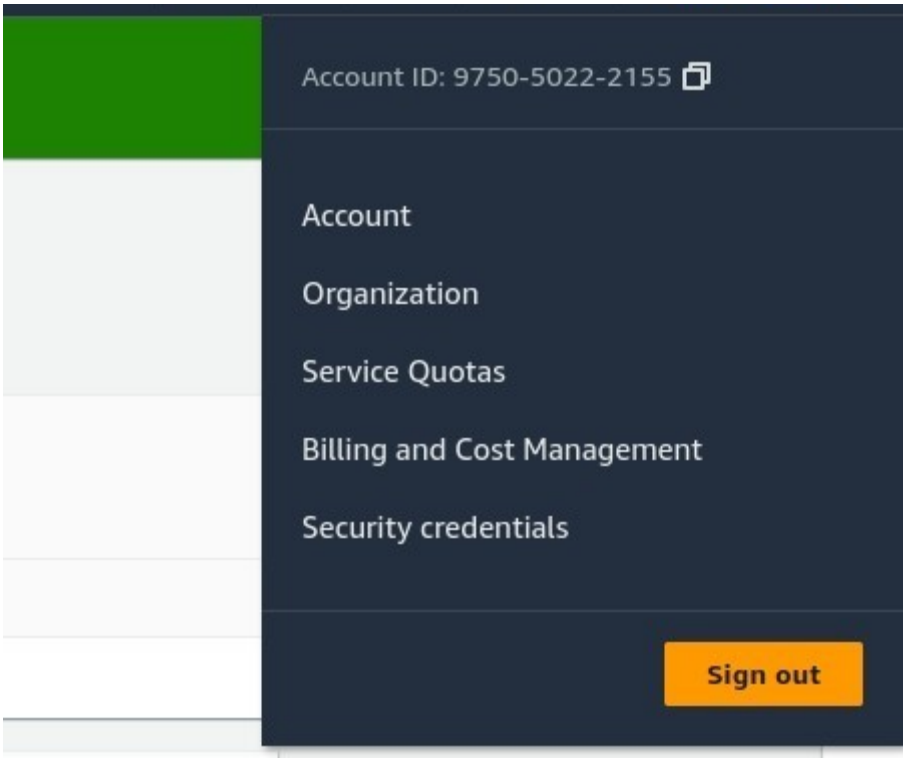
it should display the below outout.

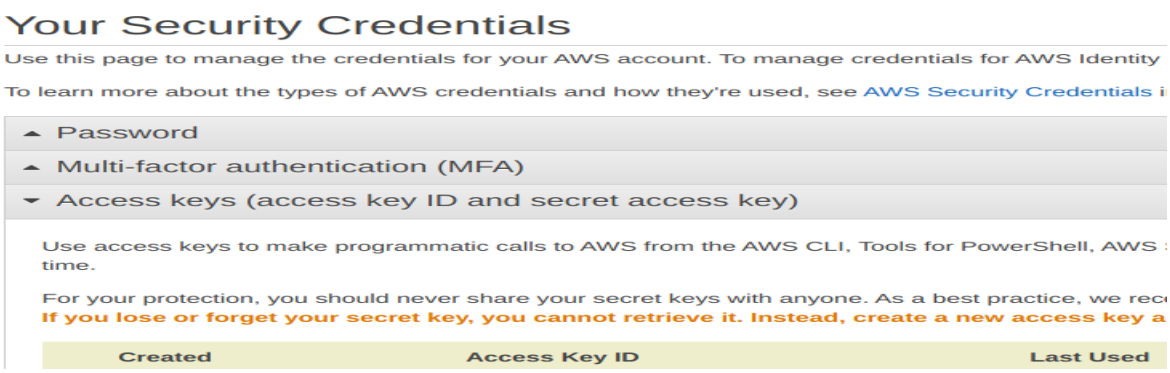
aws-cli/2.1.29 Python/3.8.8 Linux/5.4.0-1038-aws exe/x86_64.ubuntu.18 prompt/off

```
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads$ aws --version
aws-cli/1.18.69 Python/3.6.9 Linux/5.4.0-150-generic botocore/1.16.19
```

local machine.

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





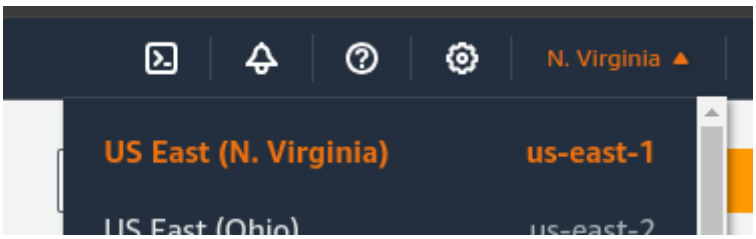
Continue on security credentials, click on access keys

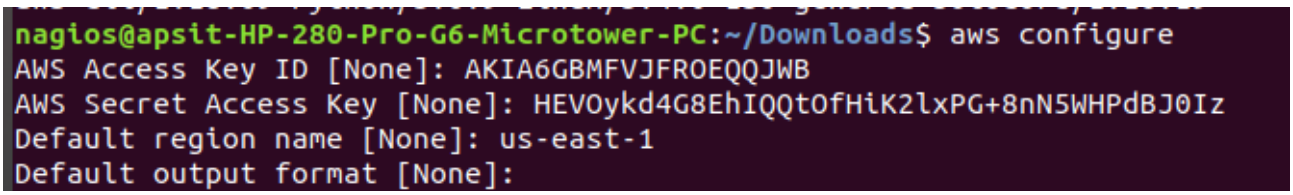
Perform below commands in Linux where you have installed Terraform

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

vishal@apsit:~$aws configure

You can check region as shown in below image :





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

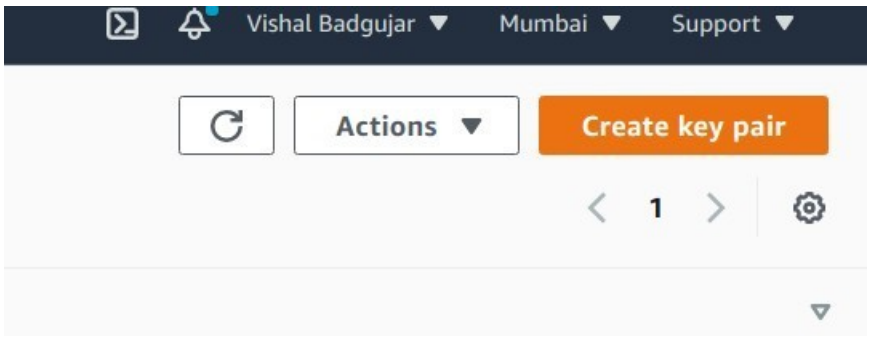
vishal@apsit:~$ cd ~
vishal@apsit:~$ mkdir project-terraform
vishal@apsit:~$ cd project-terraform

```
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads$ cd ~
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~$  mkdir project-terraform
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~$  cd project-terraform
nagios@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform$
```

Create Terraform Files

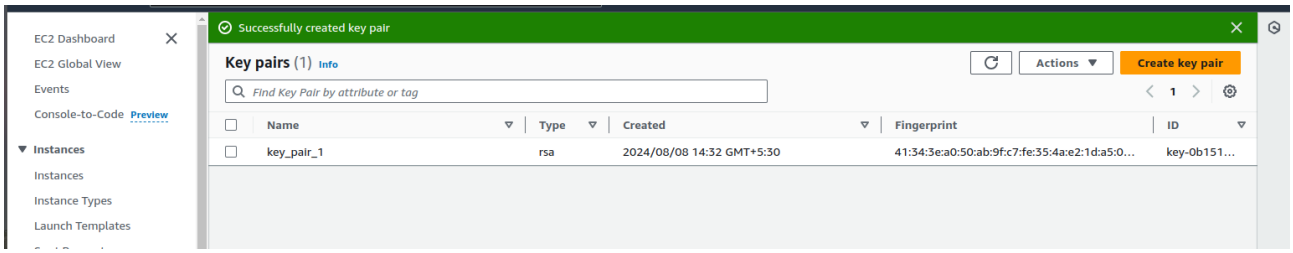
vishal@apsit:~$ sudo nano variables.tf

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



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

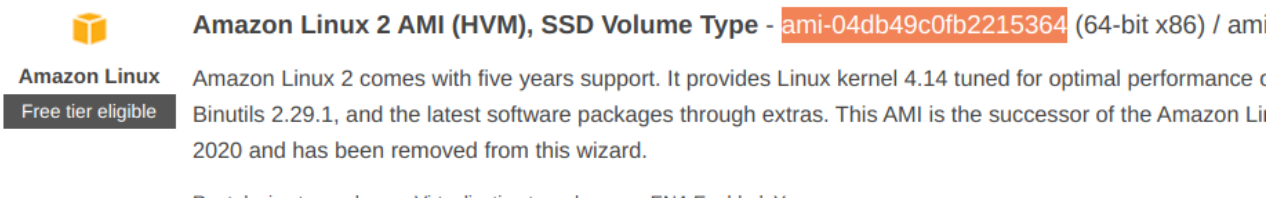
Key pair is generated



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

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                        variables.tf                          Modified

variable "aws_region" {
  description = "The AWS region to create things in."
  default     = "ap-south-1"
}

variable "key_name" {
  description = " SSH keys to connect to ec2 instance"
  default     =  "terraform"
}

variable "instance_type" {
  description = "instance type for ec2"
  default     =  "t2.micro"
}

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  M-E Redo
```

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.



Now create main.tf file:

```
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 jenkis 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-0b9064170e32bde34"
  key_name = var.key_name
  instance_type = var.instance_type
  security_groups= [ "security_jenkins_port"]
```

```
  tags= {
    Name = "jenkins_instance"
  }
}

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

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

you should see like below screenshot.

```
vishal@apsit:~/project-terraform$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.52.0...
- Installed hashicorp/aws v3.52.0 (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.
```

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.

Execute the below command

Provide the value as Yes for applying terraform

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

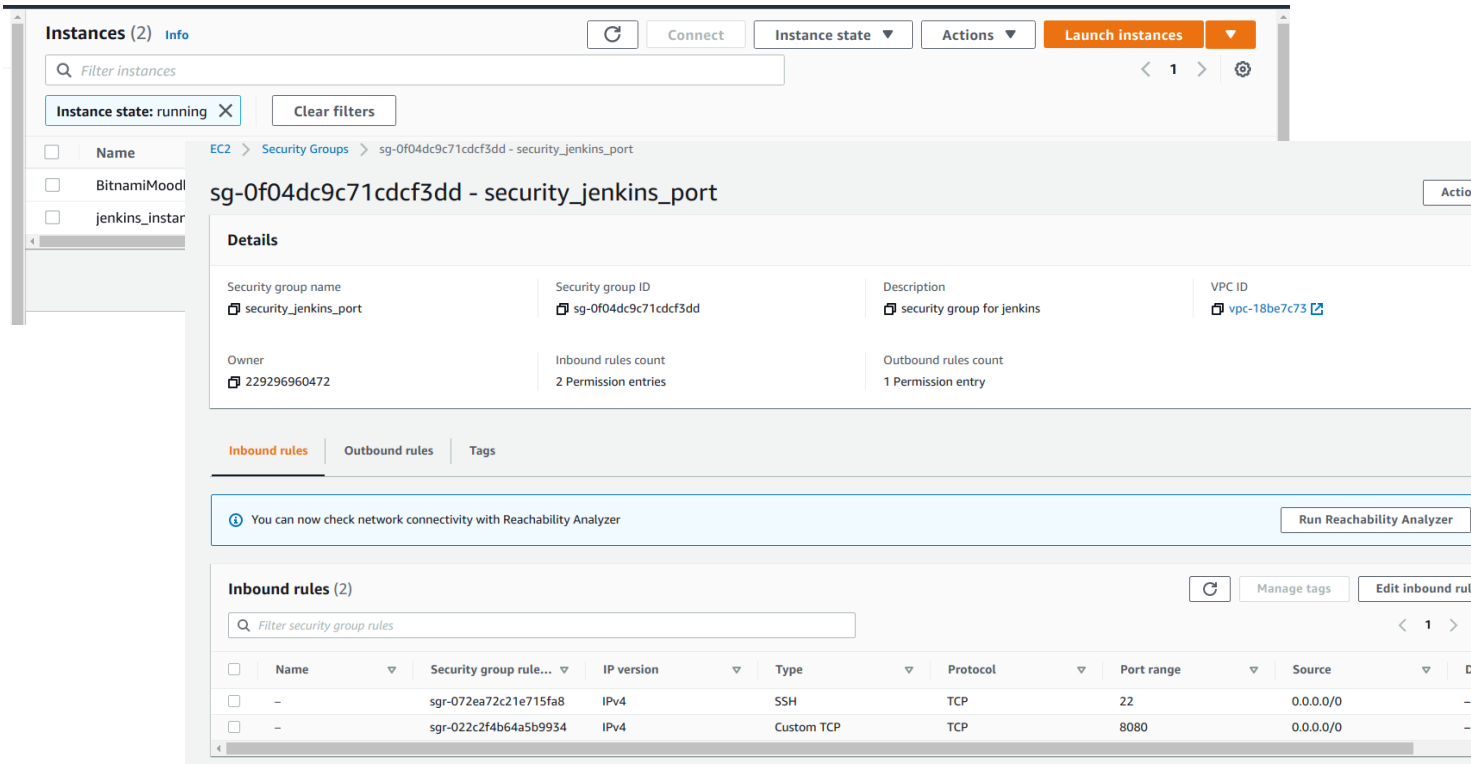
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

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

```
Plan: 2 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_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 32s [id=i-0a4a0fb7e55252d0f]
aws_eip.myFirstInstance: Creating...
aws_eip.myFirstInstance: Creation complete after 1s [id=eipalloc-0fd8f60524b10fc93]

Apply complete! Resources: 2 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.



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

Terraform destroy

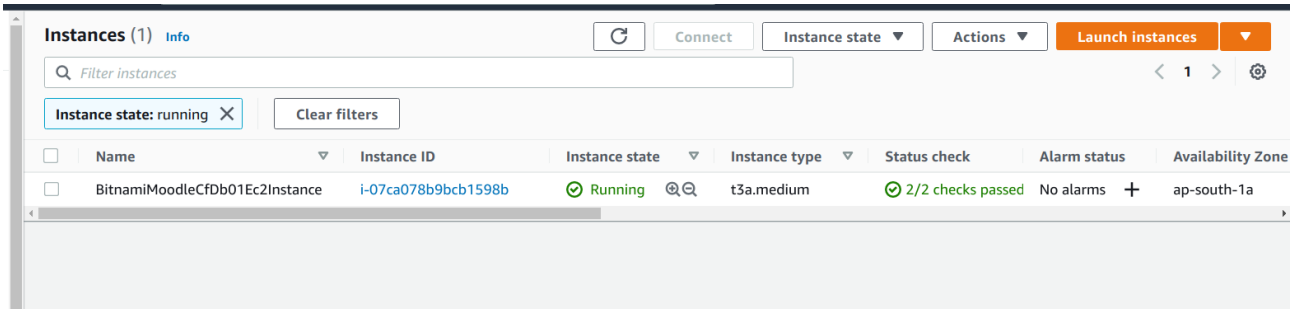
you can also destroy or delete your instance by using terraform destroy command :

terraform destroy

```
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
```

```
  Enter a value: yes

aws_eip.myFirstInstance: Destroying... [id=eipalloc-0fd8f60524b10fc93]
aws_security_group.security_jenkins_port: Destroying... [id=sg-0f04dc9c71cdcf3dd]
aws_eip.myFirstInstance: Destruction complete after 2s
aws_instance.myFirstInstance: Destroying... [id=i-0a4a0fb7e55252d0f]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdcf3dd, 10s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 10s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdcf3dd, 20s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 20s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdcf3dd, 30s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 30s elapsed]
aws_security_group.security_jenkins_port: Destruction complete after 38s
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 40s elapsed]
aws_instance.myFirstInstance: Destruction complete after 40s

Destroy complete! Resources: 3 destroyed.
```

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: In this experiment To Build, change, and destroy AWS infrastructure Using Terraform.