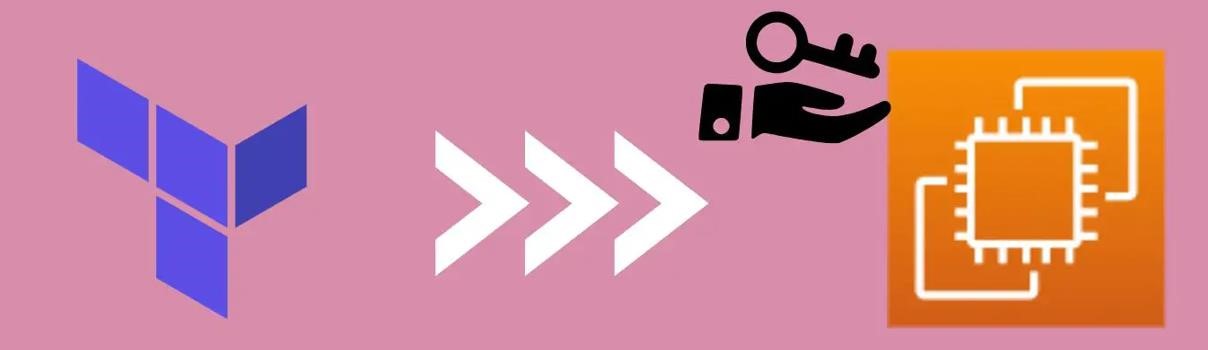
TERRAFORM DOCUMENTATION

Terraform

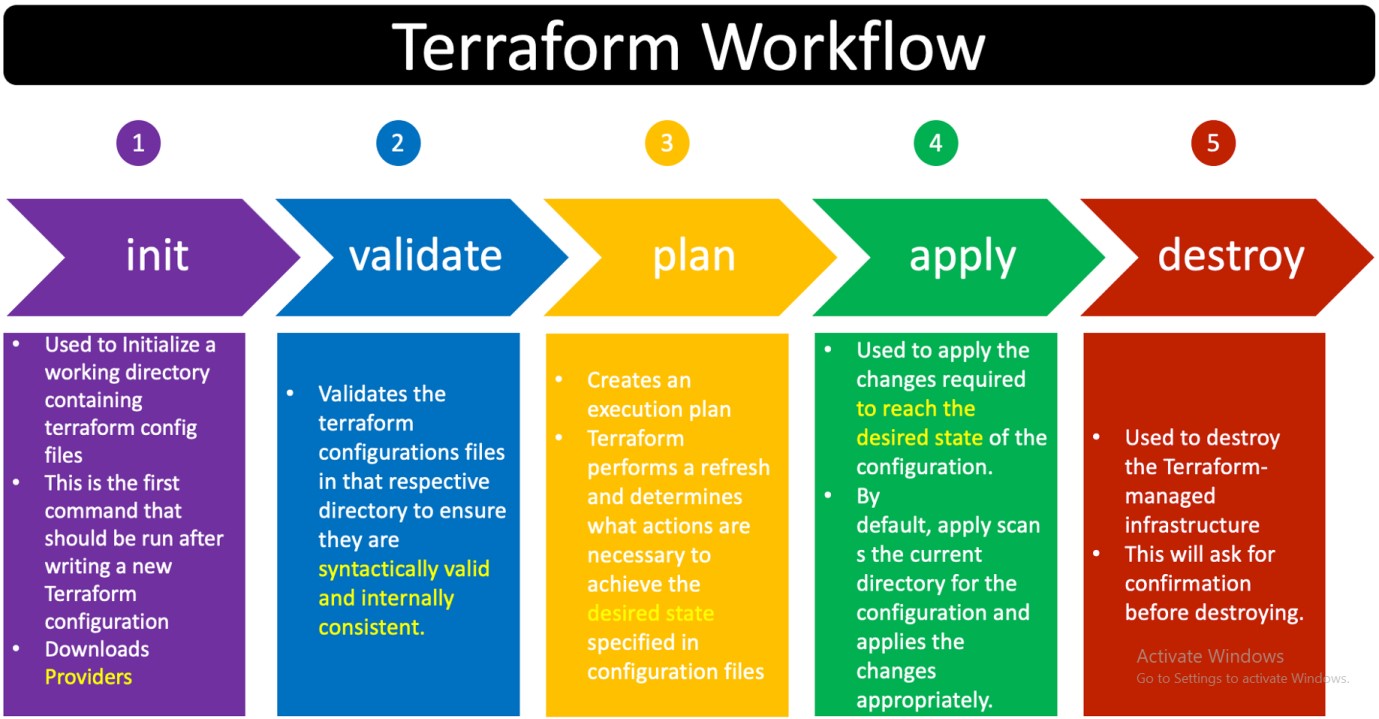


Introduction:-

1.What is Terrafrom?

Terraform is an open-source infrastructure as code (IaC) tool created by HashiCorp. It is used for building, changing, and versioning infrastructure efficiently. Terraform allows you to define your infrastructure in a declarative configuration language and then create, update, or delete infrastructure resources to match that configuration.

2.What is the workflow of Terraform?



Terraform is an infrastructure as a code tool that allows you to define and configure your infrastructure using your declarative code. Its syntax is very simple. Terraform allows to codify the infrastructure, and so it's also known as Infrastructure as Code (IaC).

Following are the commands which are being covered:

terraform init terraform plan terraform apply terraform destroy terraform validate terraform output terraform refresh terraform import terraform taint terraform untaint terraform state list terraform state show terraform state rm terraform state mv terraform fmt terraform get terraform graph terraform import terraform providers terraform show terraform force-unlock terraform workspace terraform version terraform console terraform debug terraform state pull terraform state push terraform state replace-provider terraform workspace new terraform workspace list terraform workspace show terraform workspace select terraform workspace delete

1. **terraform init:**

**Initialize a new or existing Terraform working directory.**

1. **terraform plan:**

**Generate and show an execution plan for changes to infrastructure.**

1. **terraform apply:**

**Apply the changes required to reach the desired state of the configuration.**

1. **terraform destroy:**

**Destroy the Terraform-managed infrastructure.**

1. **terraform validate:**

**Validate the configuration files in a directory.**

1. **terraform output:**

**Read an output variable from a Terraform state file.**

1. **terraform refresh:**

**Update the state file against real resources.**

1. **terraform import:**

**Import existing infrastructure into Terraform.**

1. **terraform taint:**

**Mark a resource instance as tainted, forcing it to be destroyed and recreated on the next `apply`.**

1. **terraform untaint:**

**Remove the tainted state from a resource instance.**

1. **terraform state list:**

**List resources within a Terraform state.**

1. **terraform state show:**

**Show the attributes of a single resource in the Terraform state.**

1. **terraform state rm:**

**Remove a resource from the Terraform state.**

1. **terraform state mv:**

**Move an item within the Terraform state.**

1. **terraform fmt:**

**Rewrites Terraform configuration files to a canonical format.**

1. **terraform get:**

**Download and install modules for the configuration in the current directory.**

1. **terraform graph:**

**Create a visual representation of a Terraform configuration or execution plan.**

1. **terraform import:**

**Import existing infrastructure into Terraform.**

1. **terraform providers:**

**Prints a tree of the providers used in the configuration.**

1. **terraform show:**

**Inspect Terraform state or plan.**

1. **terraform force-unlock:**

**Release a stuck lock on the current workspace/state.**

1. **terraform workspace:**

**Workspace management.**

1. **terraform version:**

**Prints the Terraform version.**

1. **terraform console:**

**Interactive console for evaluating Terraform expressions.**

1. **terraform debug:**

**Debug output management.**

1. **terraform state pull:**

**Fetch and output the state from a remote backend.**

1. **terraform state push:**

**Update remote state from a local state file.**

1. **terraform state replace-provider:**

**Remove provider instances from the terraform.tfstate file.**

1. **terraform workspace new:**

**Create a new workspace.**

1. **terraform workspace list:**

**List workspaces.**

1. **terraform workspace show:**

**Show the current workspace.**

1. **terraform workspace select:**

**Select a workspace.**

1. **terraform workspace delete:**

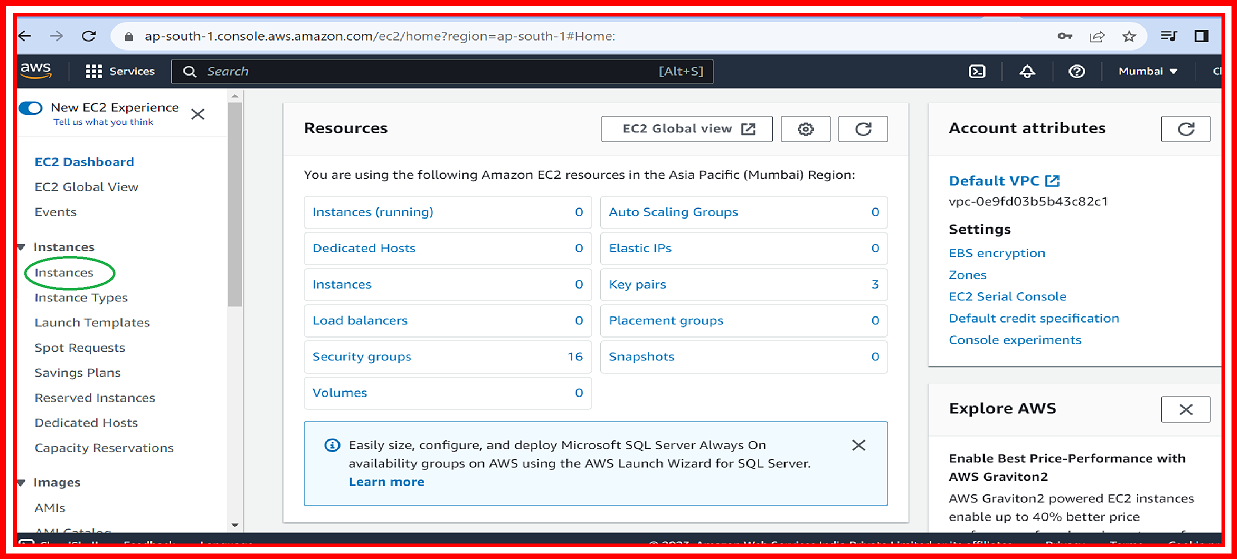
**Delete a workspace.**

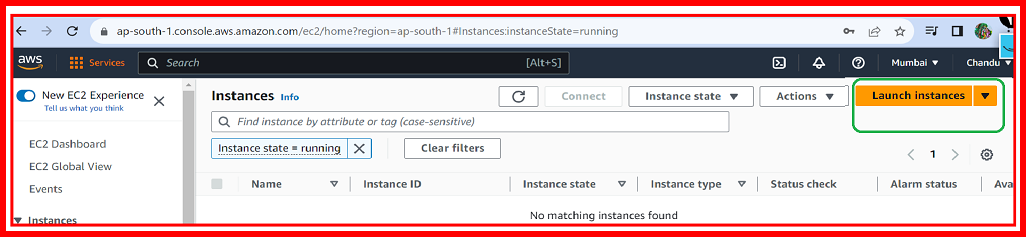
1. **terraform graph:**

**Create a visual representation of a Terraform configuration or execution plan.**

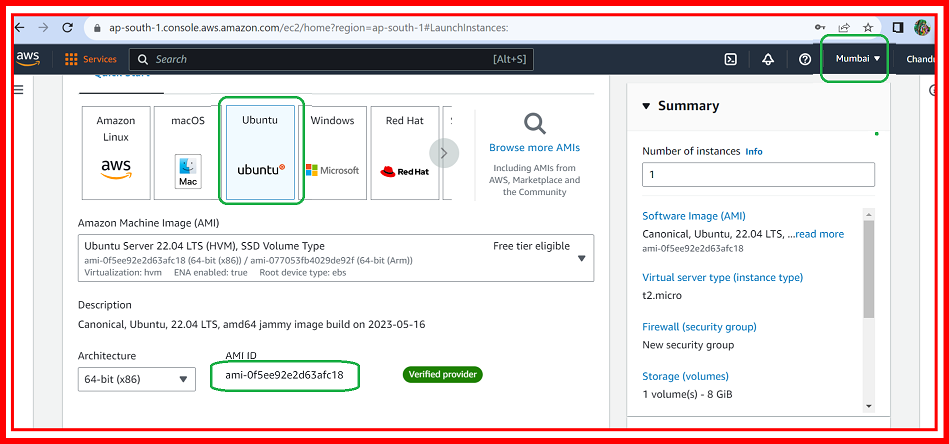
Steps to launch a AWS instance by using Terraform

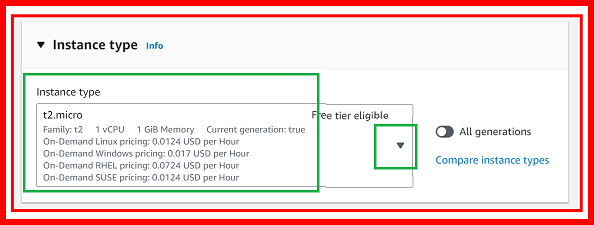
a>Login to AWS Account and Create EC2 Instance

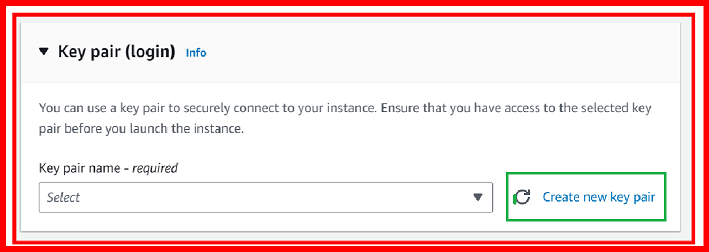


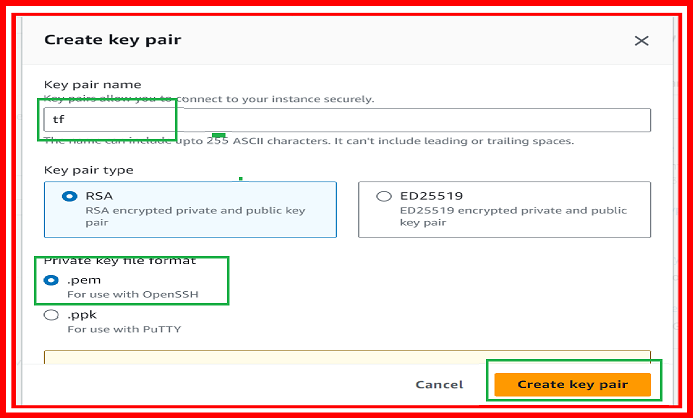


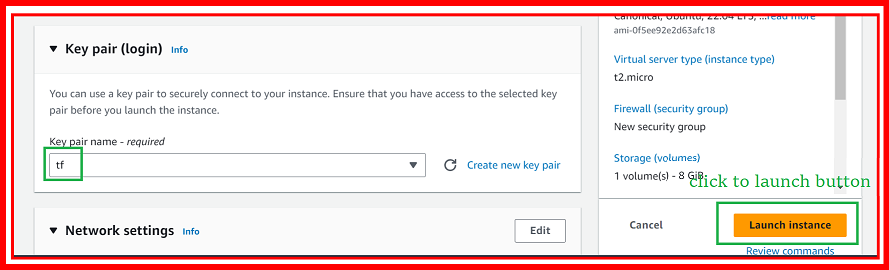


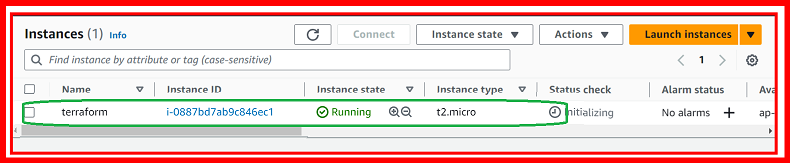


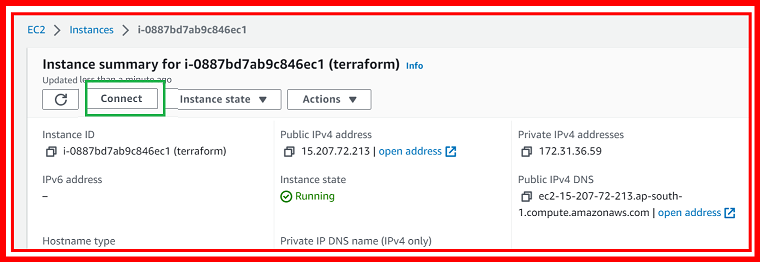


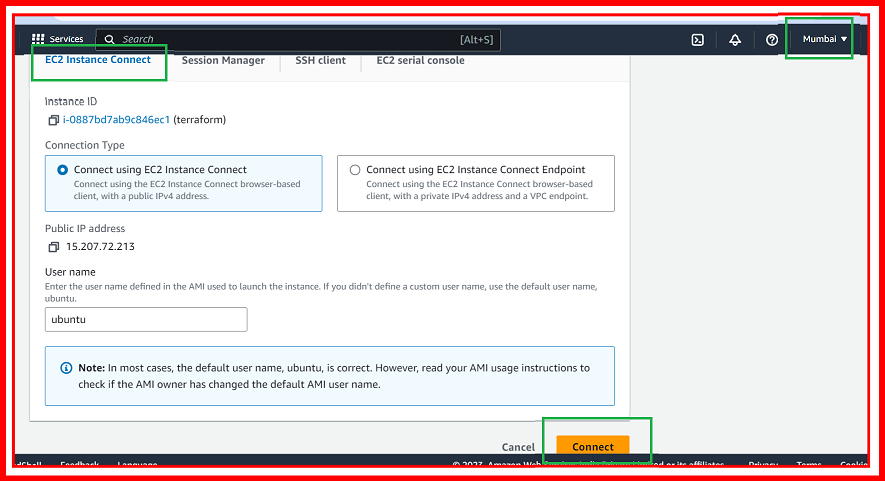


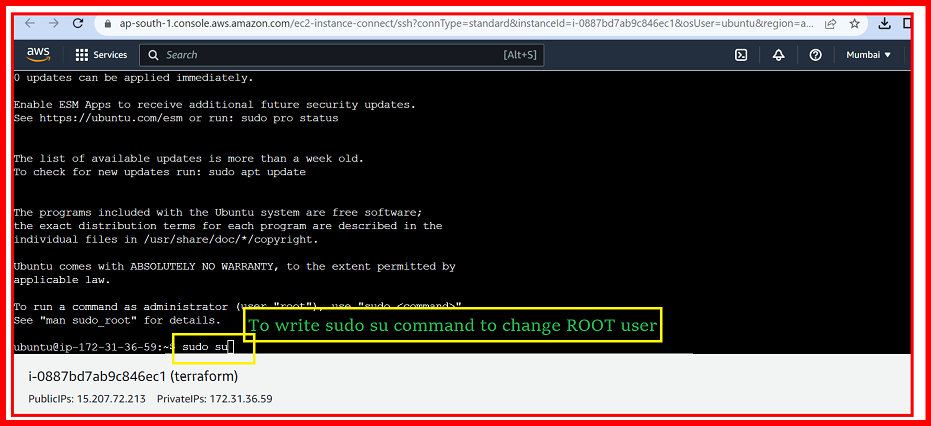


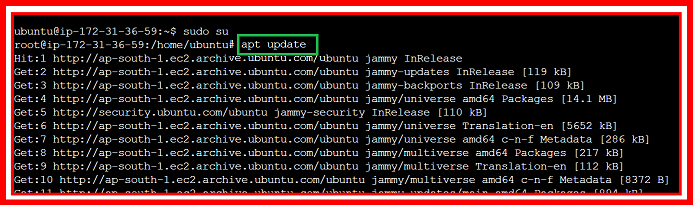












TERRAFORM/DOCUMENTATION

1.Install Terraform:

If you haven't already, you need to install Terraform on your local machine. You can download it from the official Terraform website.

2.Create a Terraform Configuration File:

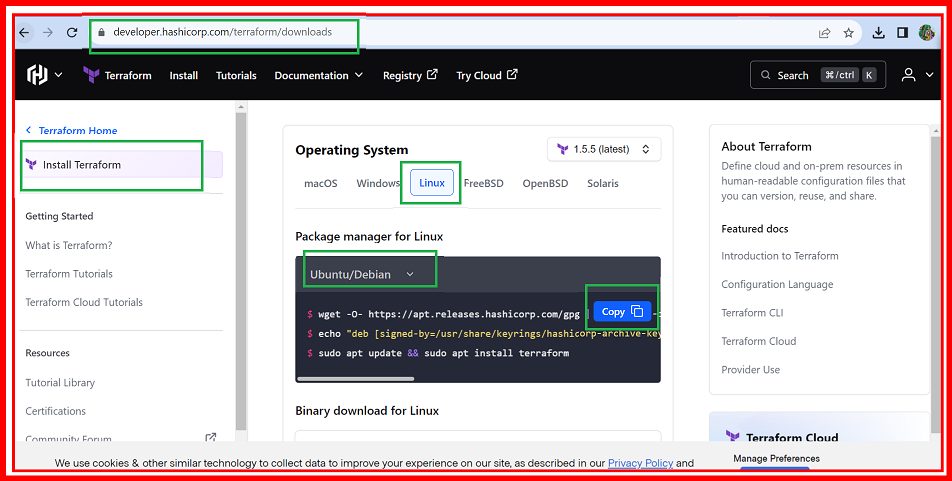
Create a file with a .tf extension, for example, main.tf. This is where you'll define your infrastructure using HashiCorp Configuration Language (HCL).

INSTALL TERAFORM

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

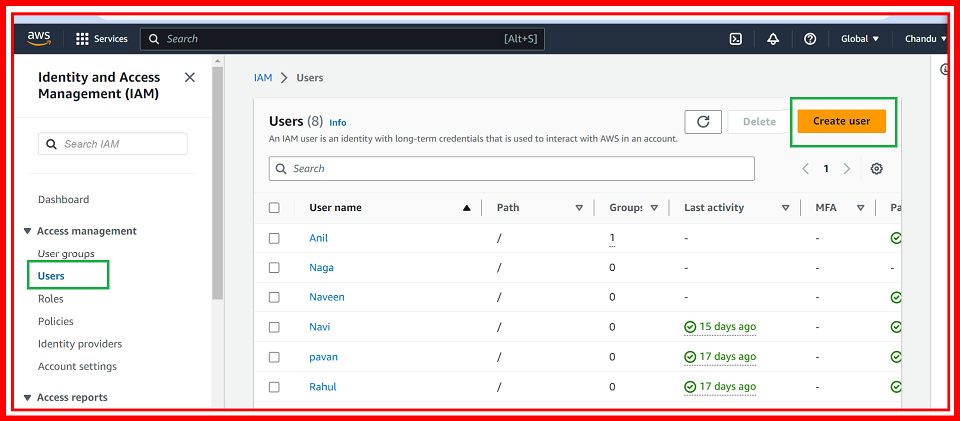
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform

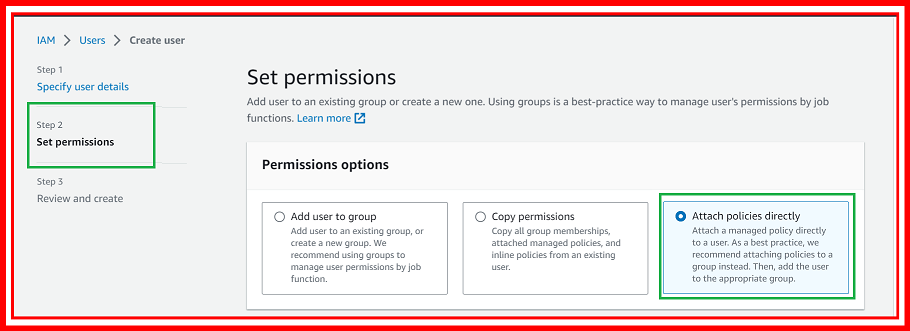


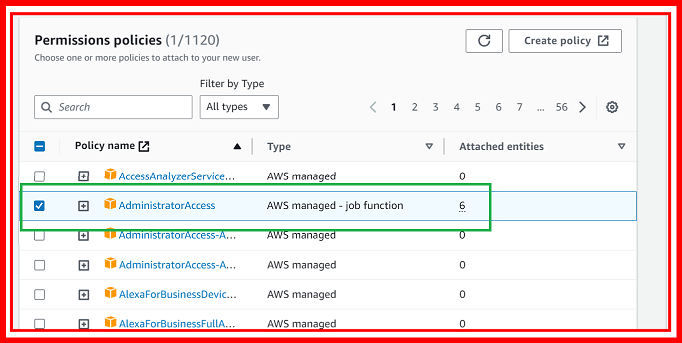


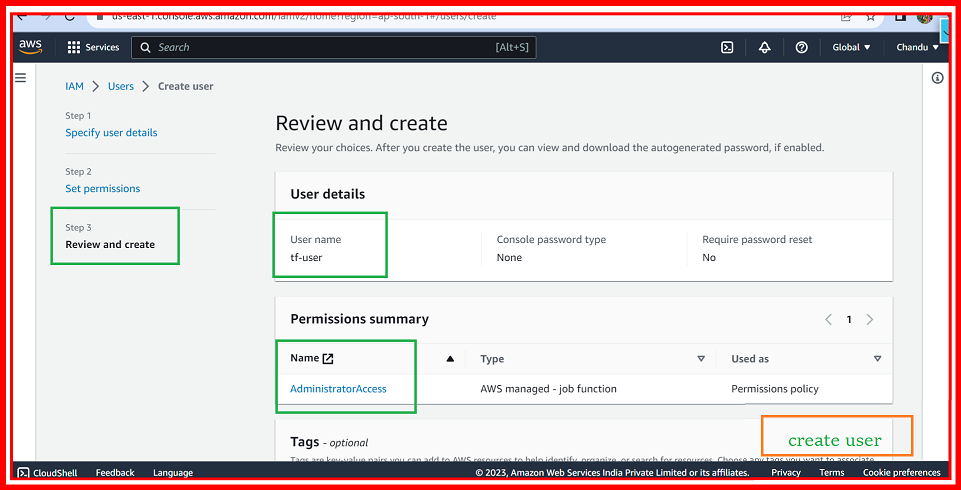
Create IAM user:



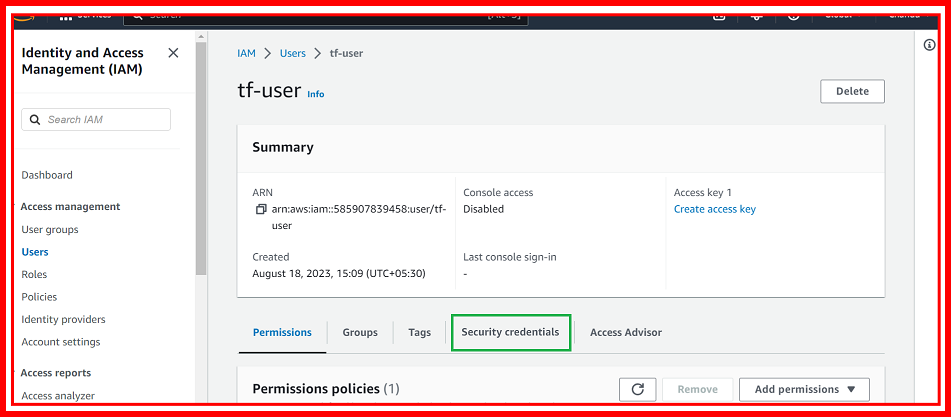


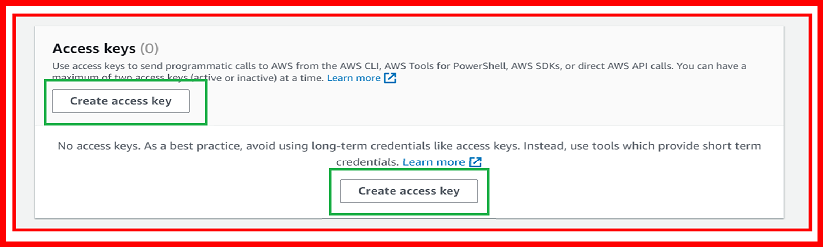


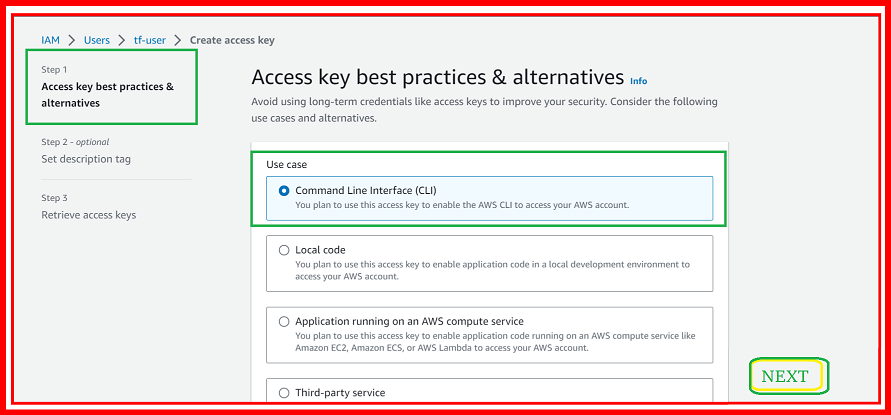


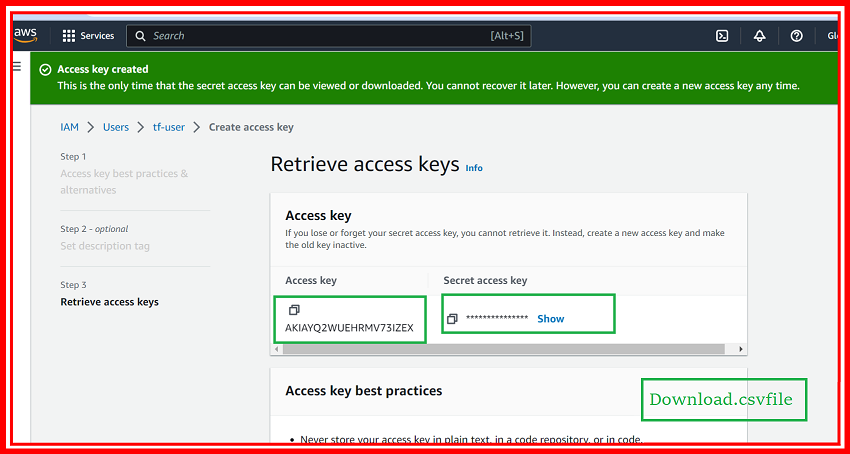


Create Access Key



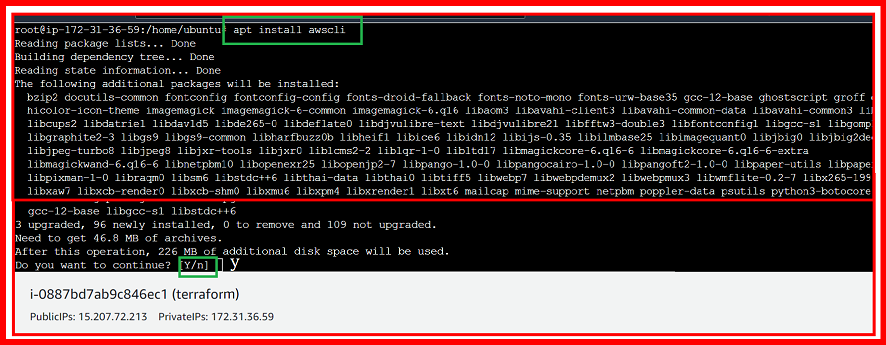






Install AWSCLI:

apt install awscli



AWS CONFIGURE

aws configure



TERRAFORM EC2

Configure AWS Provider:

You need to configure the AWS provider to let Terraform know where to provision resources. You'll need your AWS access key and secret key for this.

provider "aws" {

region = "us-east-1"

}

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-buil

vi main.tf

provider "aws" {

region = "us-west-2"

}

resource "aws\_instance" "app\_server" {

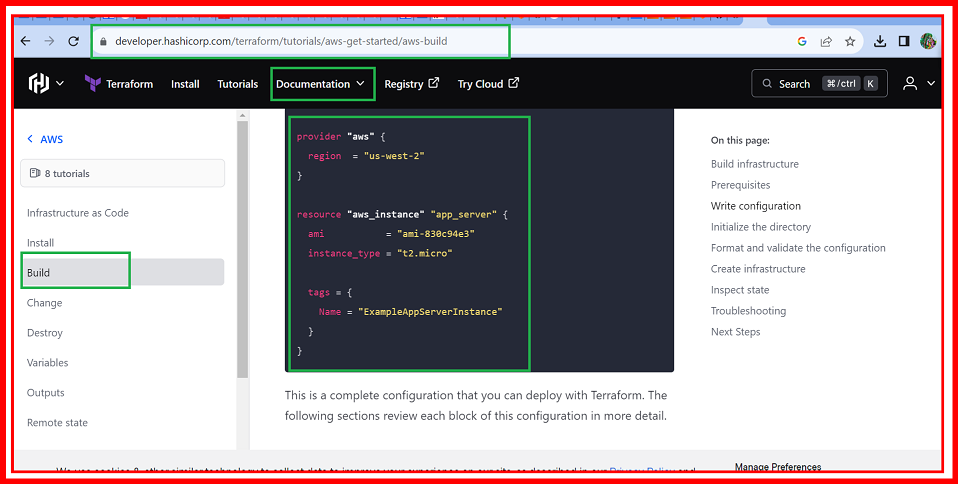
ami = "ami-830c94e3"

instance\_type = "t2.micro"

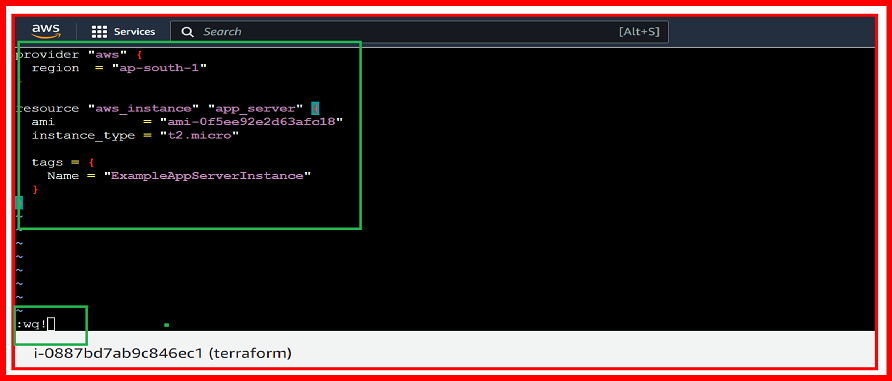
tags = {

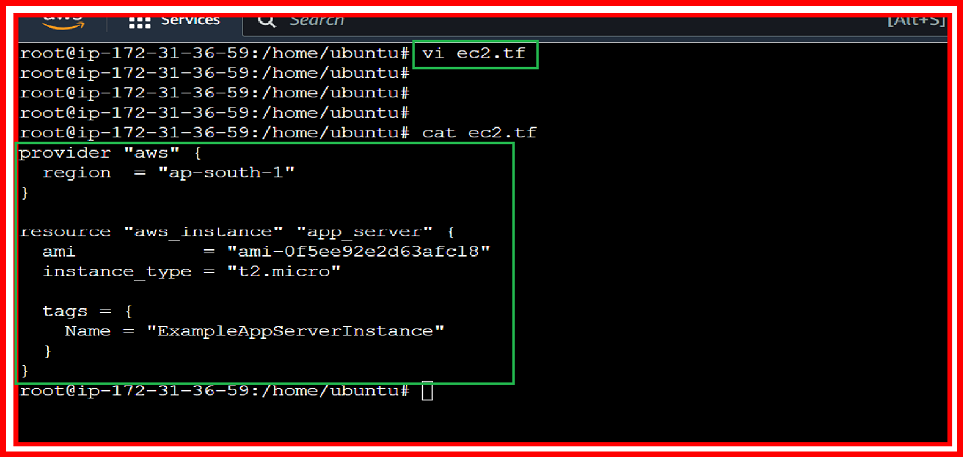
Name = "ExampleAppServerInstance"

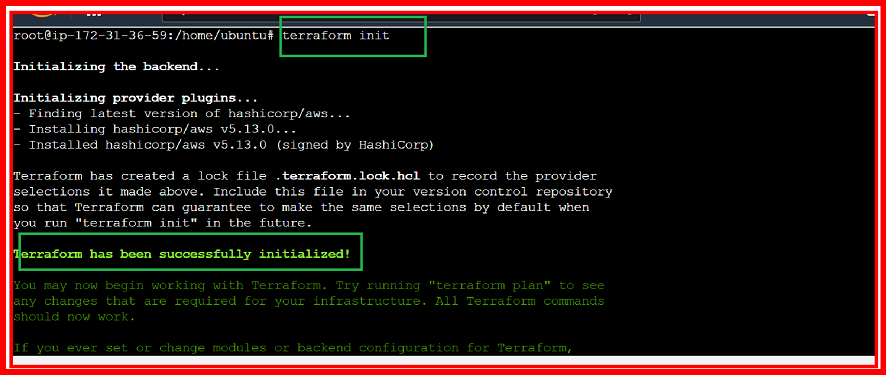
}

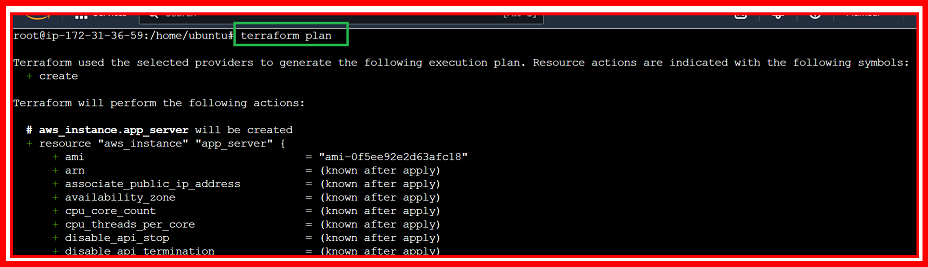
****

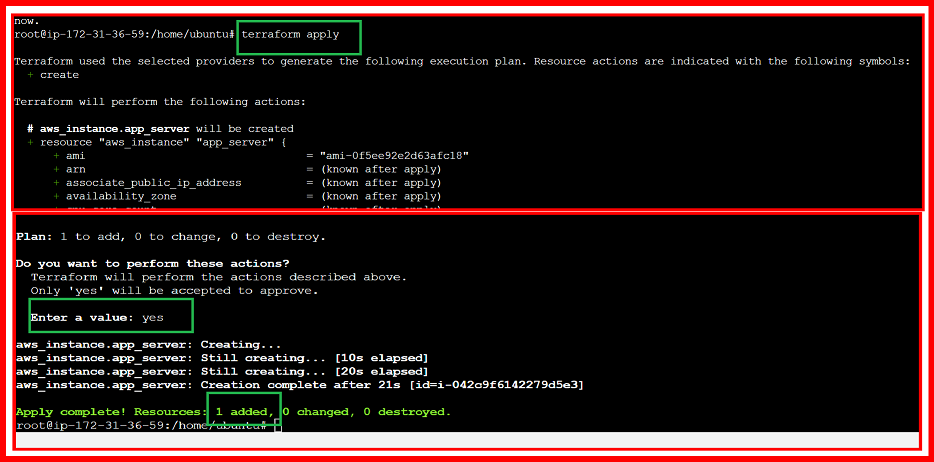


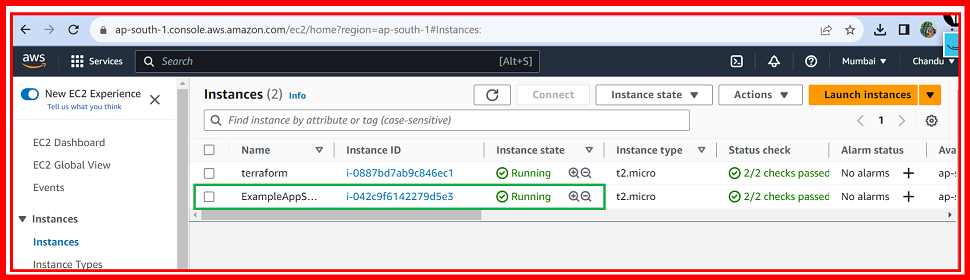


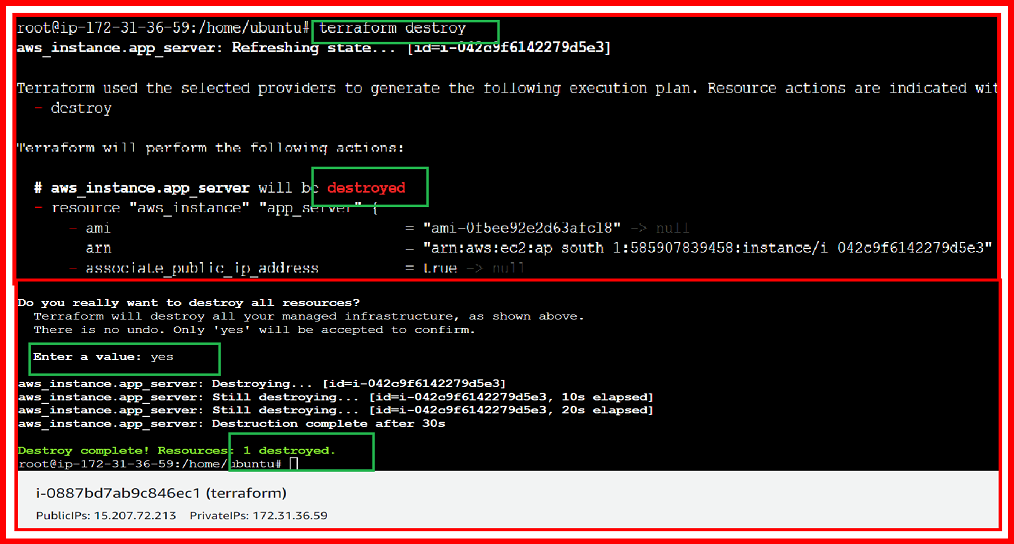


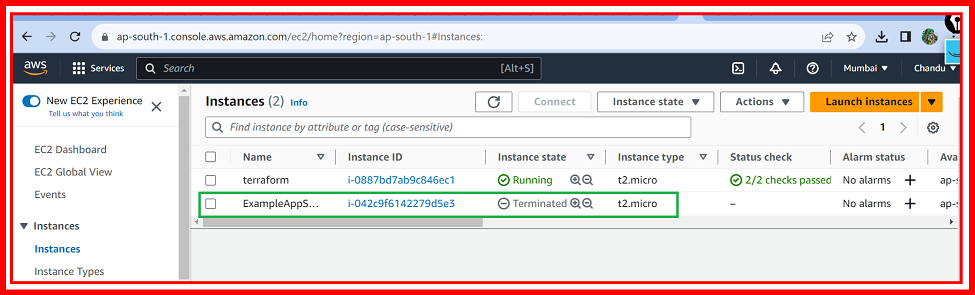












TERRAFORM AWS S3

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket>

TERRAFORM S3

resource "aws\_s3\_bucket" "example" {

bucket = "my-tf-test-bucket"

tags = {

Name = "My bucket"

Environment = "Dev"

}

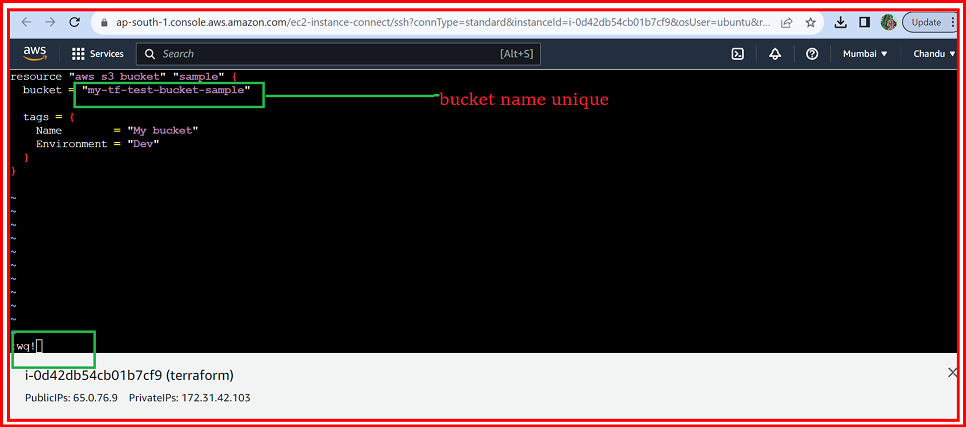
}

Buckets [key point]:

S3 uses a flat namespace called "buckets" to organize and store data. Each bucket has a globally unique name and is used to store objects (files) and metadata. Buckets can be created in different AWS regions

Create s3.tf file

Vi s3.tf

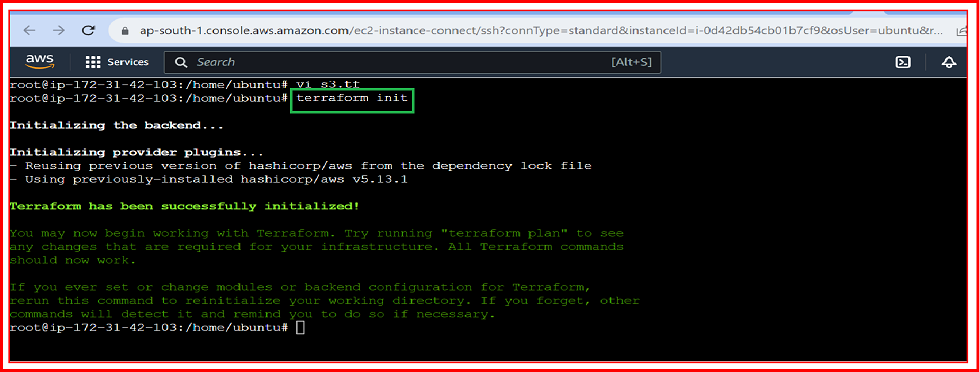


Initialize Terraform:

Open a terminal, navigate to the directory containing your main.tf file, and run the following command to initialize Terraform:

Command:

terraform init

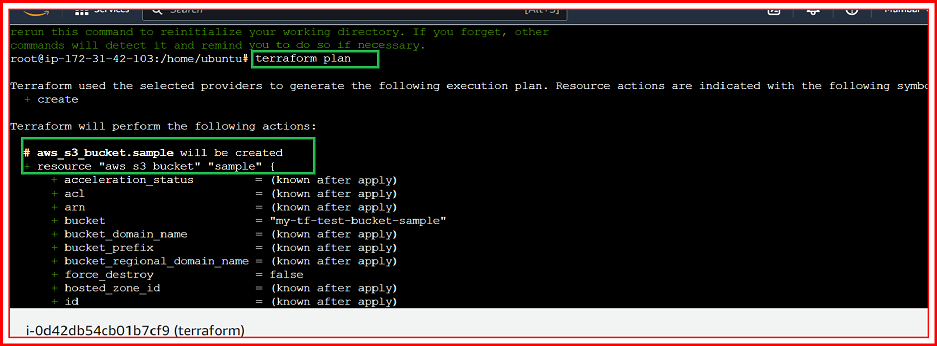


Review and Apply Changes:

After initialization, you can review the planned changes by running:

Command:

terraform plan



Apply Changes:

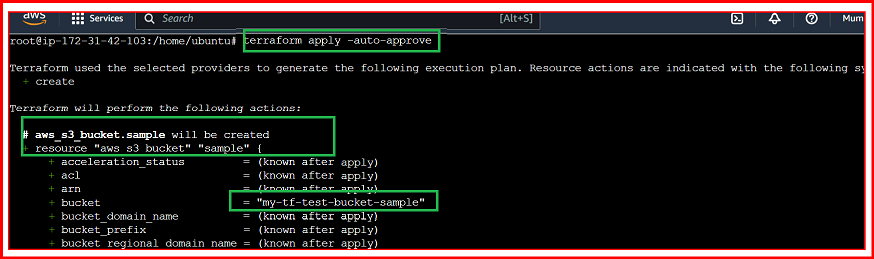
If the preview looks good, you can apply the changes and create your S3 bucket:

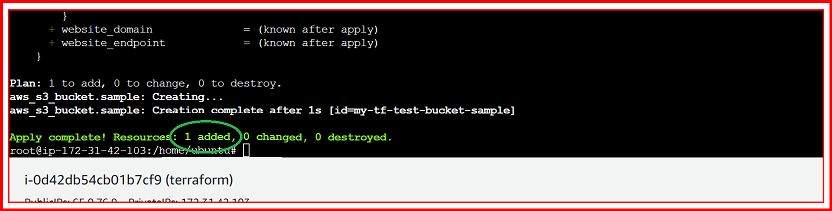
Command:

terraform apply

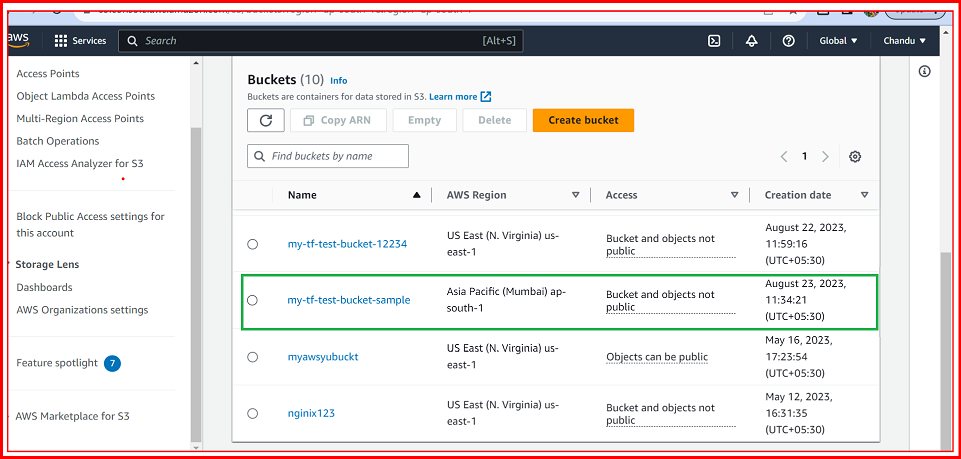
OR

Terraform apply –auto-approve





Go and check aws s3:

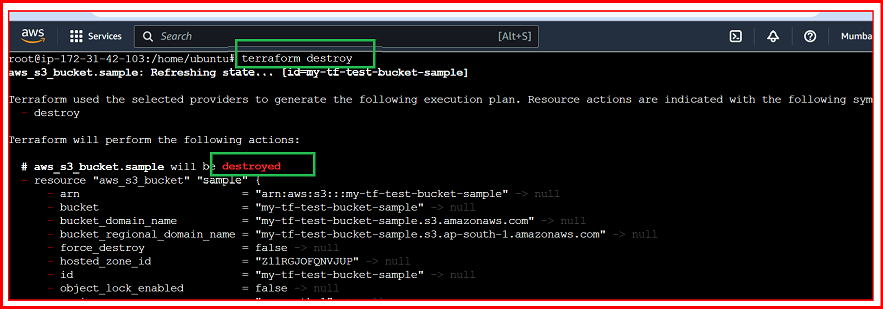


HOW TO DESTROY RESOURCES:

When you're done experimenting, you can remove the resources you've created by running:

Command:

terraform destroy



TERRAFORM VPC

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/vpc>

Create vi vpc.tf file

Vi vpc.tf:

resource "aws\_vpc" "main" {

cidr\_block = "10.0.0.0/16"

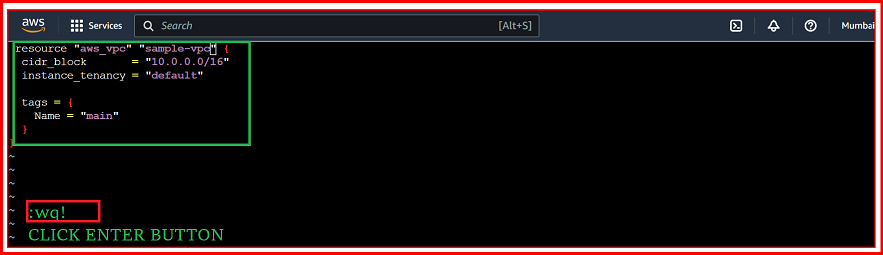
instance\_tenancy = "default"

tags = {

Name = "main"

}

}

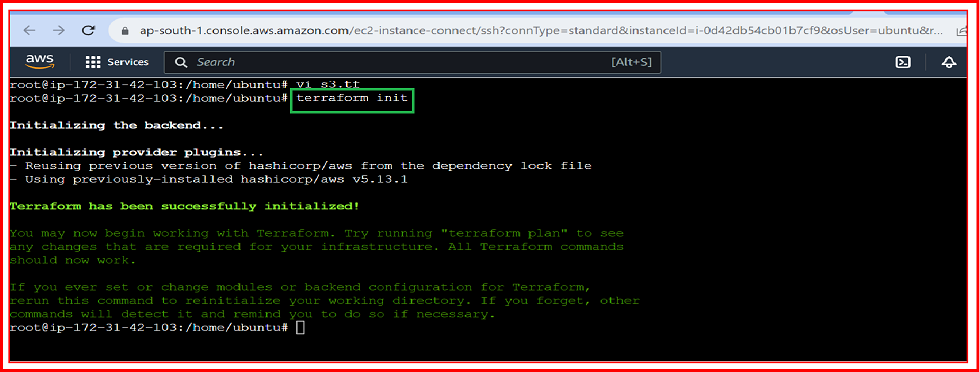


Initialize Terraform:

Open a terminal, navigate to the directory containing your main.tf file, and run the following command to initialize Terraform:

Command:

terraform init

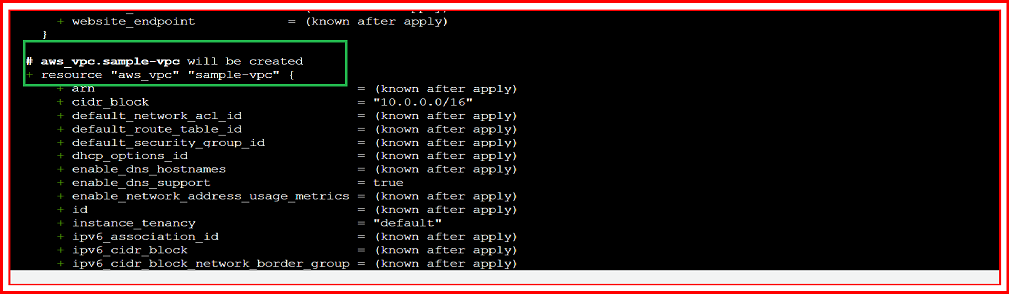


Review and Apply Changes:

After initialization, you can review the planned changes by running:

Command:

terraform plan



Apply Changes:

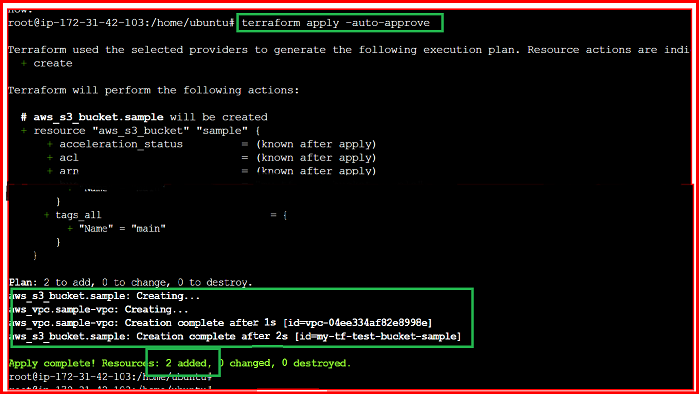
If the preview looks good, you can apply the changes and create your S3 bucket:

Command:

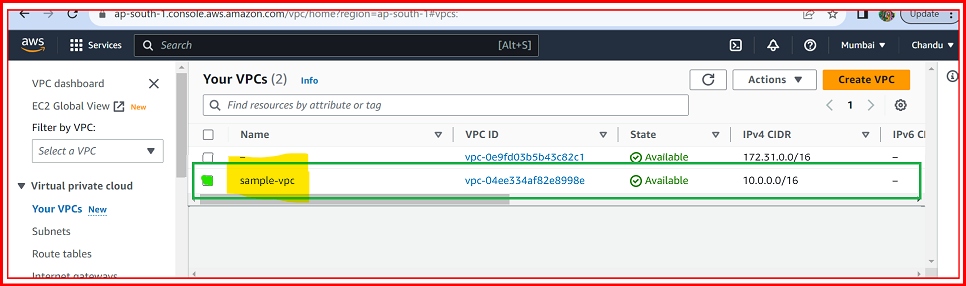
terraform apply

OR

Terraform apply –auto-approve



Go to check VPC:



Create Subnet:

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/subnet>

resource "aws\_subnet" "public" {

vpc\_id = aws\_vpc.sample-vpc.id

cidr\_block = "10.0.1.0/24"

tags = {

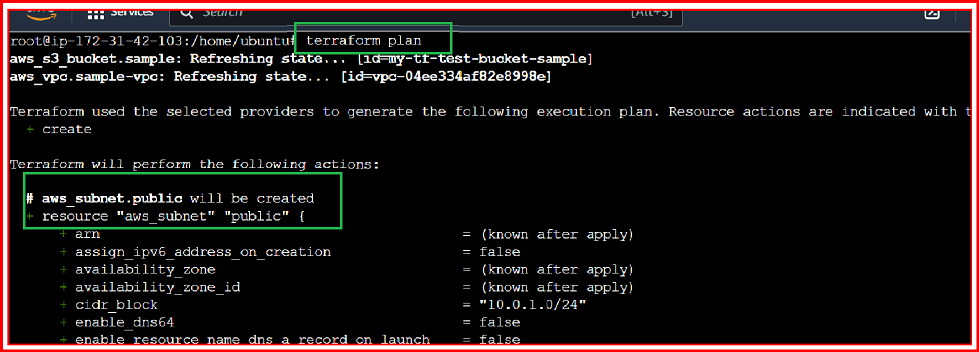
Name = "public"

}

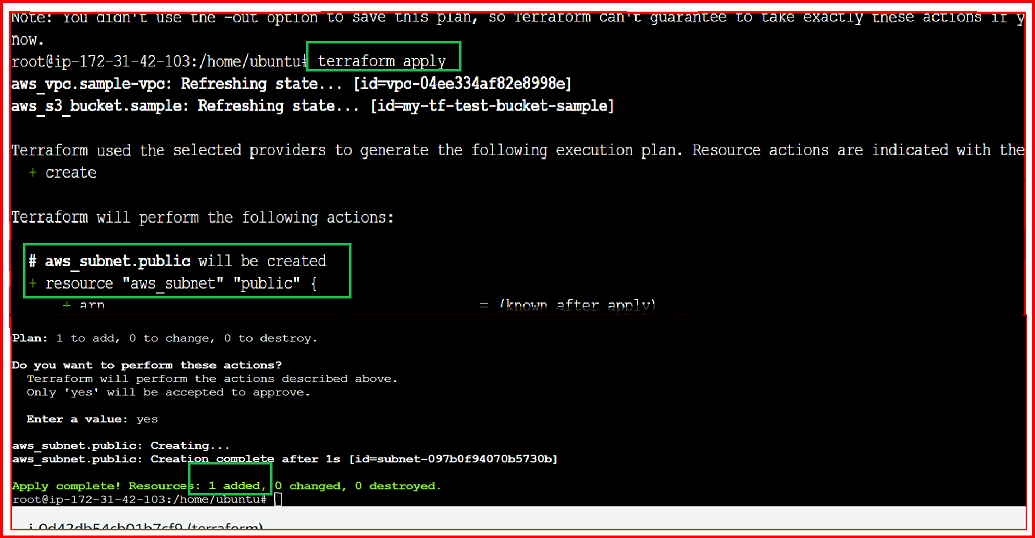
}



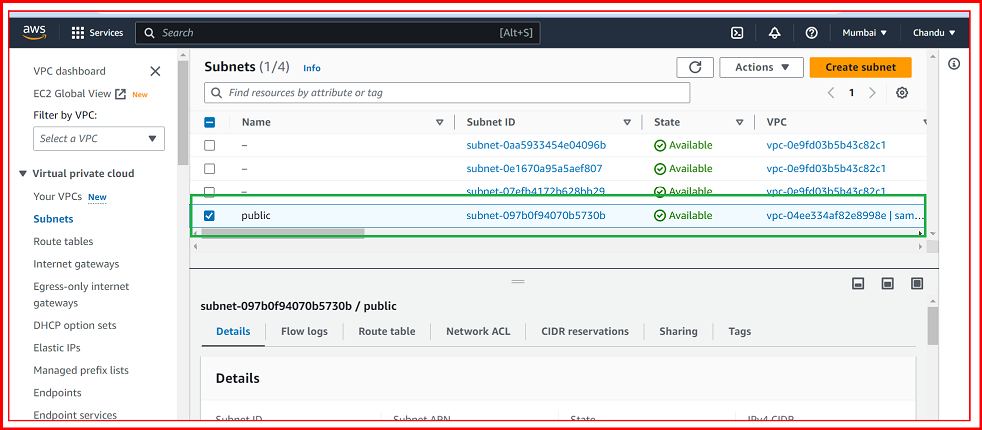
Terraform plan:



Terraform apply:



Go to check SUBNET:



Create private subnet:

resource "aws\_subnet" "private" {

vpc\_id = aws\_vpc.sample-vpc.id

cidr\_block = "10.0.1.0/24"

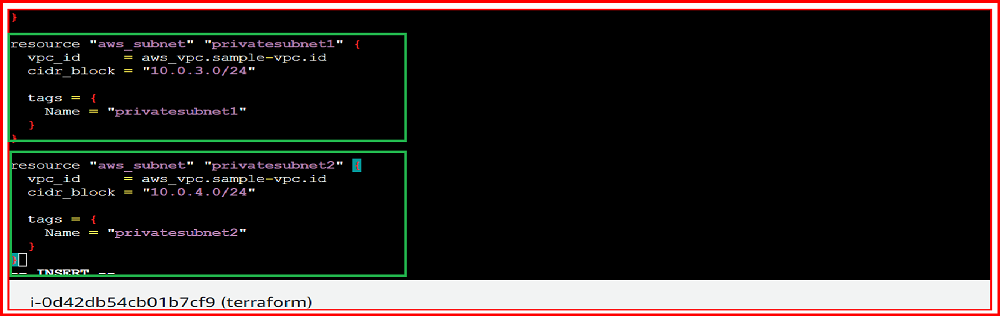
tags = {

Name = "private"

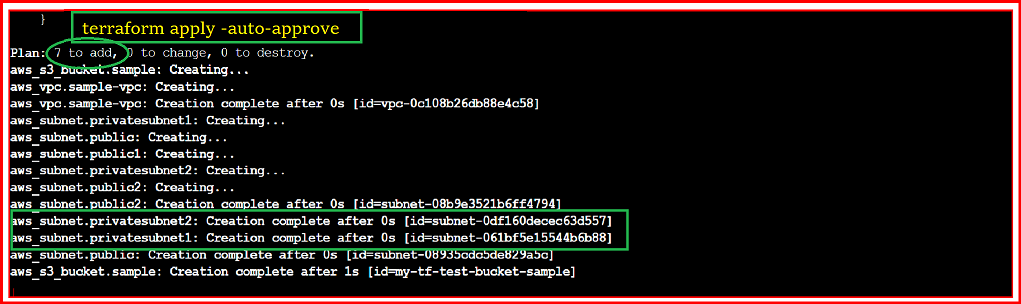
}

}

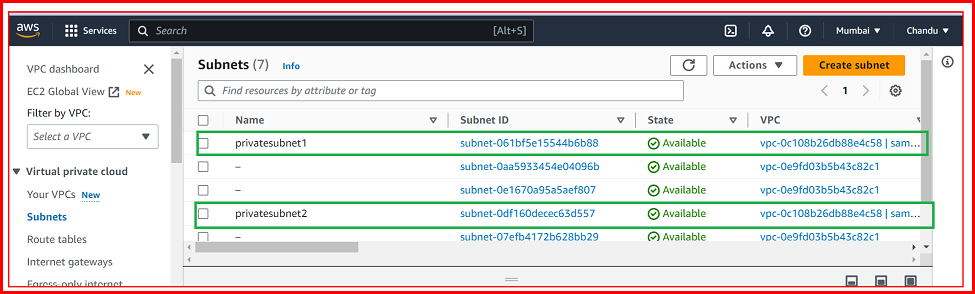
Vi vpc.tf file:



**TERRAFORM APPLY:**



**AWS ACCOUNT AND CHECK SUBNETS OPEN:**



**INTERNET GATEWAY ADD:**

**https link:**

[**https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/internet\_gateway**](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/internet_gateway)

**resource "aws\_internet\_gateway" "igw" {**

**vpc\_id = aws\_vpc.sample-vpc.id**

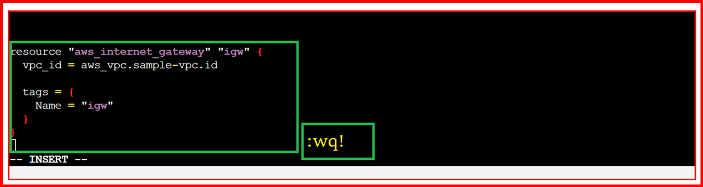
**tags = {**

**Name = "igw"**

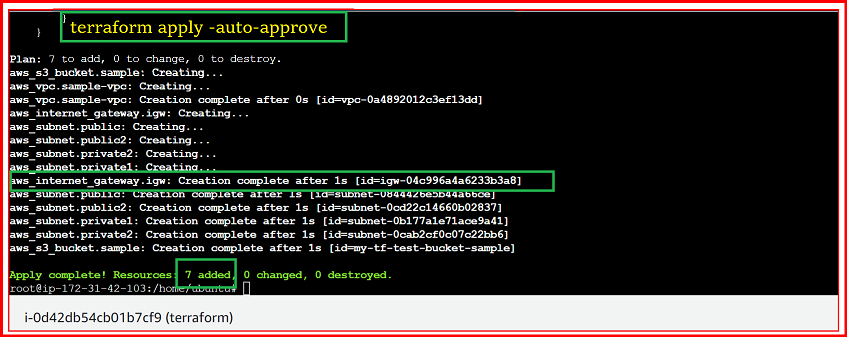
**}**

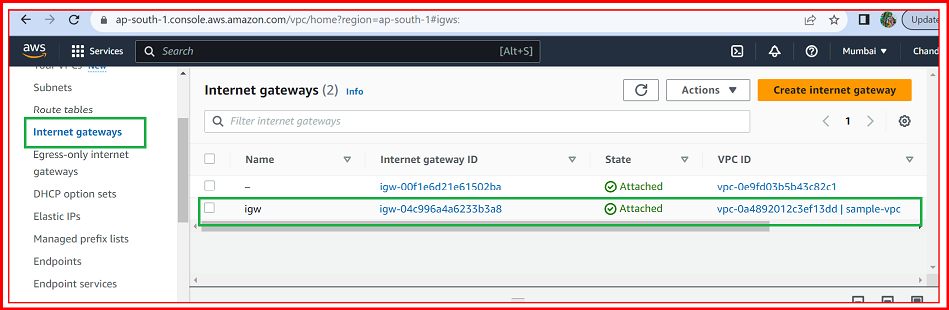
**}**

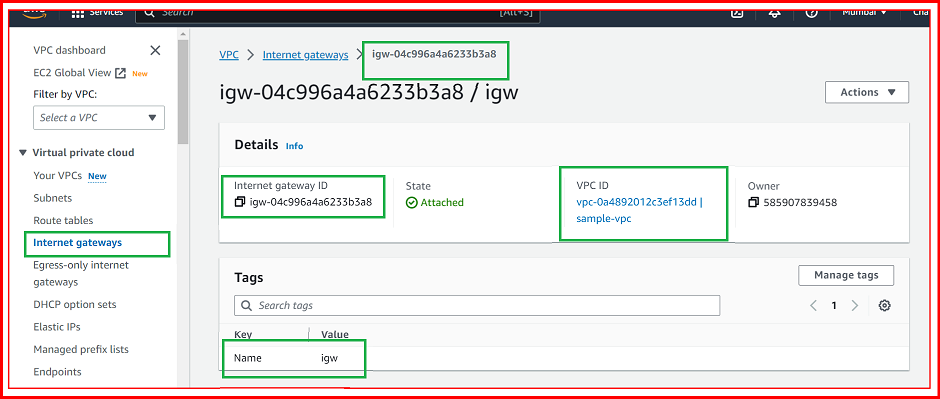
**Vi vpc.tf file:**

****

**TERRAFORM APPLY:**

****

****

****

**ELASTIC IPs:**

**https link:**

[**https://registry.terraform.io/providers/figma/aws-4-49-0/latest/docs/resources/eip\_association**](https://registry.terraform.io/providers/figma/aws-4-49-0/latest/docs/resources/eip_association)

**ex:**

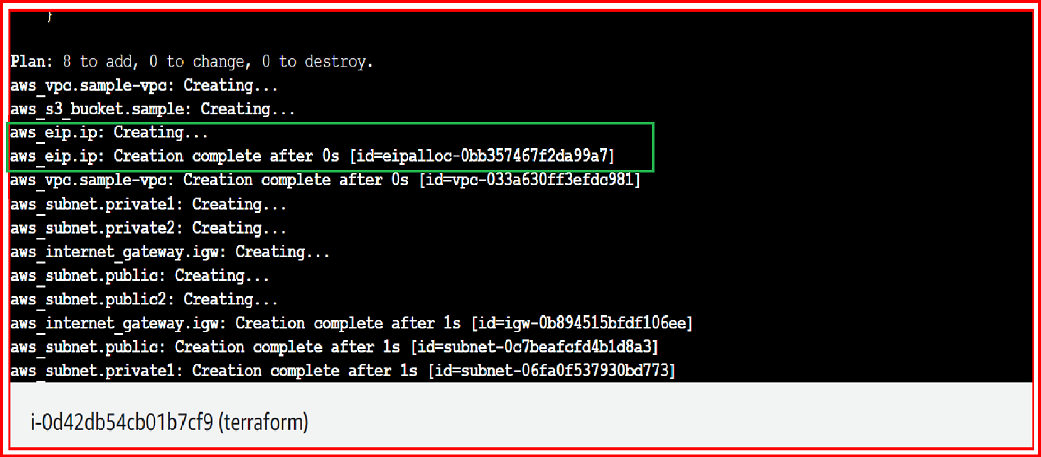
**resource "aws\_eip" "ip" {**

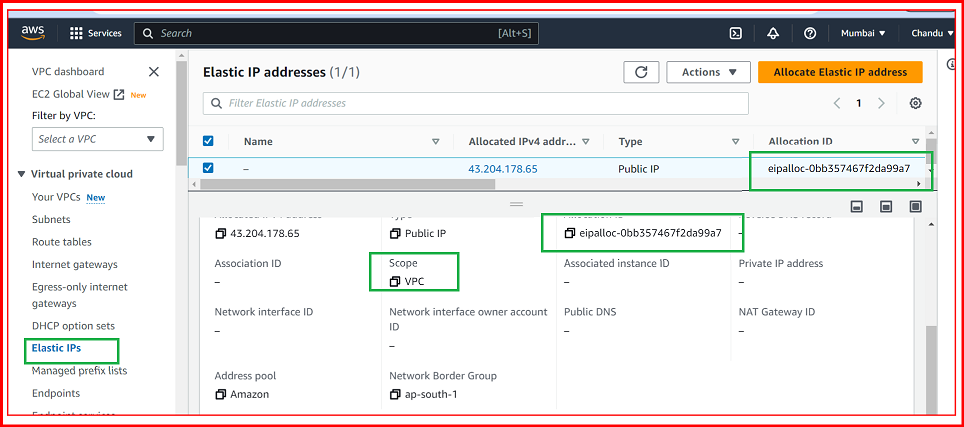
**#vpc = true**

**}**

**Vi vpc.tf file:**

****

****

****

**NAT GATEWAY:**

**https link:**

[**https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/nat\_gateway**](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/nat_gateway)

**resource "aws\_nat\_gateway" "sample-nat-gateway" {**

**allocation\_id = aws\_eip.ip.id**

**subnet\_id = aws\_subnet.privatesubnet1.id**

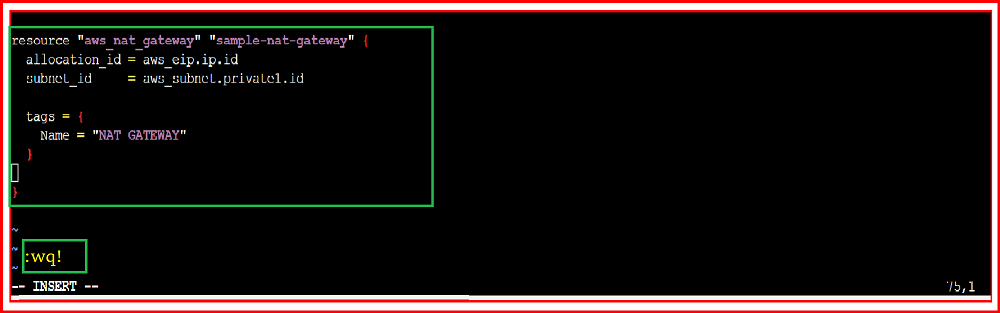
**tags = {**

**Name = "NAT GATEWAY"**

**}**

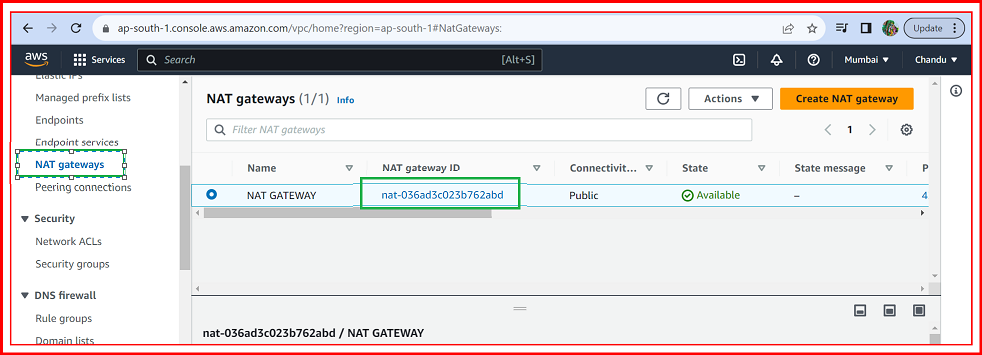
**}**

**Vi vpc.tf file:**

****

TERRAFORM APPLY:





**ROUTE TABLE:**

[**https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route\_table**](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table)

**resource "aws\_route\_table" "mypubrt" {**

**vpc\_id = aws\_vpc.sample-vpc.id**

**route {**

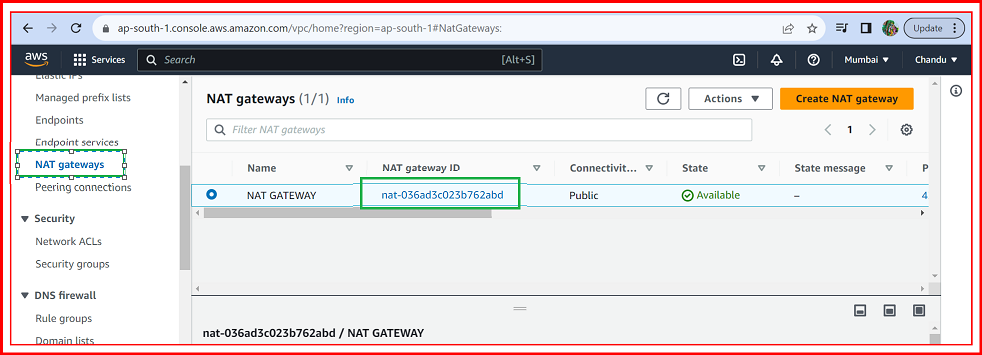
**cidr\_block = "0.0.0.0/0"**

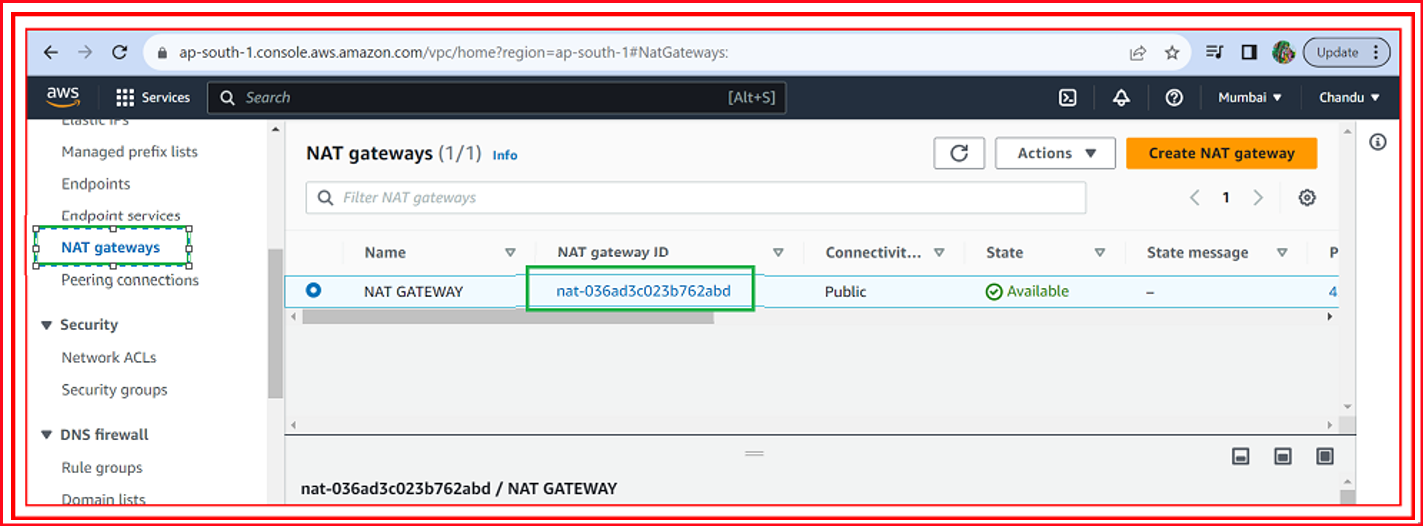
**gateway\_id = aws\_internet\_gateway.igw.id**

**}**

**}**

**Bvhjgjg**



****