**Terraform Task**

**Task Description:**

Launch Linux EC2 instances in two regions using a single Terraform file.

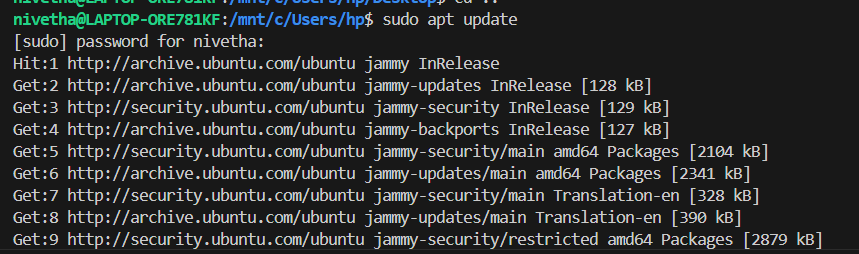
Explanation:

1. Prerequisites:

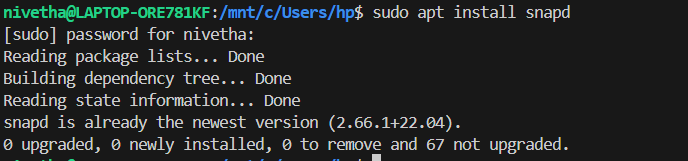
Refer documentation to install aws cli and terraform.

**AWS CLI** - [Installing or updating to the latest version of the AWS CLI - AWS Command Line Interface](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)

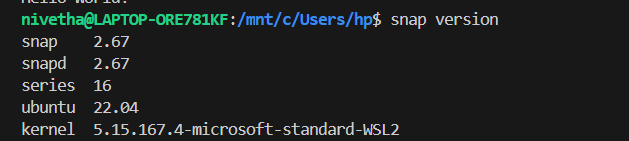
Install snap using the command : sudo apt update



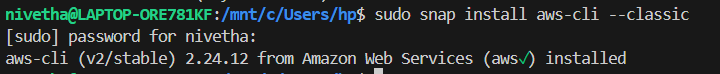
sudo apt install snapd



snap version



To Install AWS CLI, sudo snap install aws-cli –classic



To verify, use aws –version



**Terraform –** Refer doc [Install Terraform | Terraform | HashiCorp Developer](https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli)

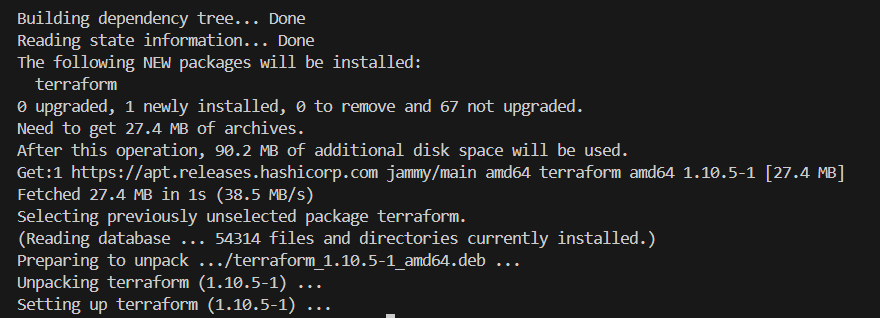
[Install | Terraform | HashiCorp Developer](https://developer.hashicorp.com/terraform/install)

use this command to install terraform on ubuntu :

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

echo "deb [arch=$(dpkg --print-architecture) 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

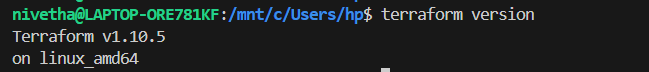
sudo apt update && sudo apt install terraform



Ensure terraform is in correct path else move it to /usr/local/bin. To check use

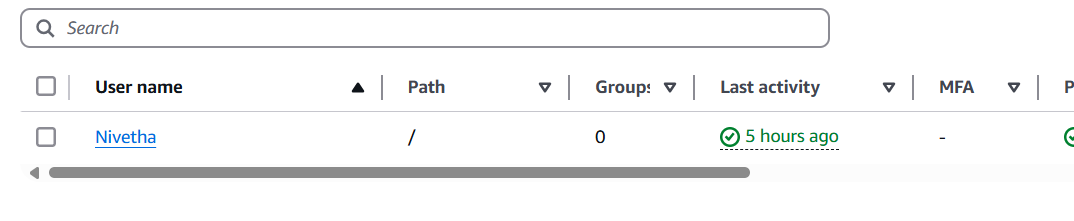
Which terraform



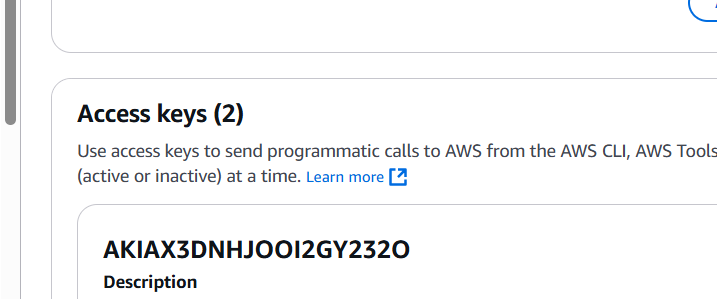


1. Aws configure : to configure aws in ubuntu,

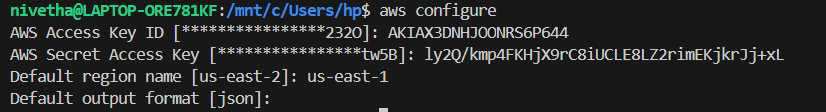
Create user with IAM role and give EC2 full access and vpc full access.

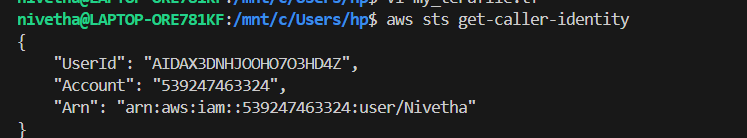


Post creation, go to security tab and create Access key, copy the access key and secret accesskey.



Use aws configure to configure your credentials.



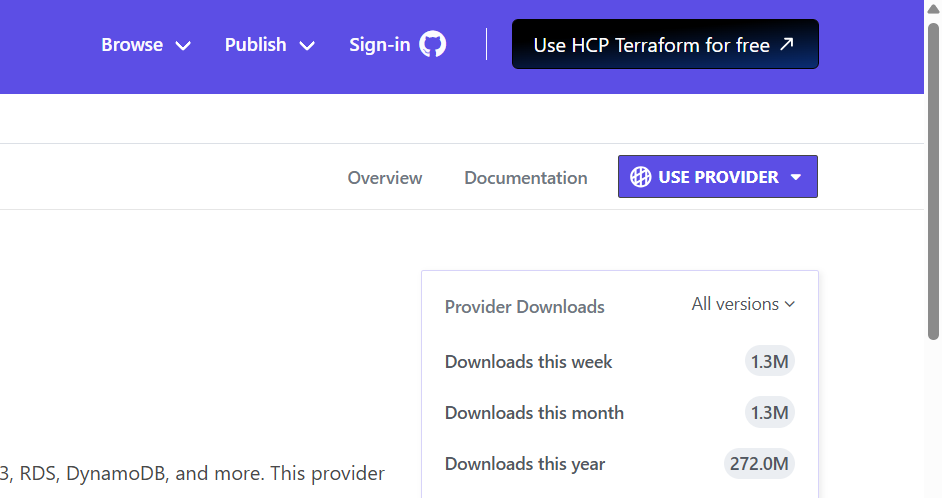


Use aws sts get-caller-identity to verify the credentials have been configured

1. Create Terraform config file.

Refer to terraform document, go to terraform hashicorp link - [Terraform by HashiCorp](https://www.terraform.io/)

Visit Registry tab > browse provider > choose AWS provider> click documentation.



Create terraform file using touch nano my\_terafile.tf



Give the following contents, - here I have created not only instance (though the task is to create instance only), created VPC, Subnets, routetable , internet gateway, Nat gateway and attached the vpc and subnet to the created instance.

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 5.0"

}

}

}

# Configure the AWS Provider

provider "aws" {

region = "us-east-1"

}

# VPC

resource "aws\_vpc" "terra\_vpc" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "Terraform\_Vpc"

}

}

#Subnet

resource "aws\_subnet" "pub\_sub" {

vpc\_id = aws\_vpc.terra\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

tags = {

Name = "Terra\_Public\_Subnet"

}

}

resource "aws\_subnet" "pri\_sub" {

vpc\_id = aws\_vpc.terra\_vpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-1b"

tags = {

Name = "Terra\_Private\_Subnet"

}

}

#Internet Gateway

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

vpc\_id = aws\_vpc.terra\_vpc.id

tags = {

Name = "Terra\_IG"

}

}

#Pub Route table

resource "aws\_route\_table" "pub\_rt" {

vpc\_id = aws\_vpc.terra\_vpc.id

route {

cidr\_block = "0.0.0.0/0"

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

}

tags = {

Name = "Terraform\_Public\_RT"

}

}

#Association of route table to pub sub

resource "aws\_route\_table\_association" "pubsubrt" {

subnet\_id = aws\_subnet.pub\_sub.id

route\_table\_id = aws\_route\_table.pub\_rt.id

}

#Eip

resource "aws\_eip" "teip" {

domain = "vpc"

}

#NAT gateway

resource "aws\_nat\_gateway" "nat\_gw" {

allocation\_id = aws\_eip.teip.id

subnet\_id = aws\_subnet.pub\_sub.id

tags = {

Name = "Terra\_NatGW"

}

# To ensure proper ordering, it is recommended to add an explicit dependency

# on the Internet Gateway for the VPC.

depends\_on = [aws\_internet\_gateway.igw]

}

#Pri Route table

resource "aws\_route\_table" "pri\_rt" {

vpc\_id = aws\_vpc.terra\_vpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_nat\_gateway.nat\_gw.id

}

tags = {

Name = "Terraform\_Pri\_RT"

}

}

#Association of route table to pri sub

resource "aws\_route\_table\_association" "prisubrt" {

subnet\_id = aws\_subnet.pri\_sub.id

route\_table\_id = aws\_route\_table.pri\_rt.id

}

#Security group-public

resource "aws\_security\_group" "allow\_all" {

name = "terra\_allow\_all"

description = "Allow TLS inbound traffic and all outbound traffic"

vpc\_id = aws\_vpc.terra\_vpc.id

tags = {

Name = "terra\_allow\_all"

}

}

resource "aws\_vpc\_security\_group\_ingress\_rule" "allow\_tls\_ipv4" {

security\_group\_id = aws\_security\_group.allow\_all.id

cidr\_ipv4 = "0.0.0.0/0"

from\_port = 22

ip\_protocol = "tcp"

to\_port = 22

}

resource "aws\_vpc\_security\_group\_ingress\_rule" "allow\_tlsh\_ipv4" {

security\_group\_id = aws\_security\_group.allow\_all.id

cidr\_ipv4 = "0.0.0.0/0"

from\_port =80

ip\_protocol = "tcp"

to\_port = 80

}

resource "aws\_vpc\_security\_group\_egress\_rule" "allow\_all\_traffic\_ipv4" {

security\_group\_id = aws\_security\_group.allow\_all.id

cidr\_ipv4 = "0.0.0.0/0"

ip\_protocol = "-1" # semantically equivalent to all ports

}

#Security group-private

resource "aws\_security\_group" "allow\_inbound" {

name = "terra\_allow\_inbound"

description = "Allow TLS inbound traffic "

vpc\_id = aws\_vpc.terra\_vpc.id

tags = {

Name = "terra\_allow\_inbound"

}

}

resource "aws\_vpc\_security\_group\_ingress\_rule" "allow\_intls\_ipv4" {

security\_group\_id = aws\_security\_group.allow\_inbound.id

referenced\_security\_group\_id = aws\_security\_group.allow\_all.id

from\_port = 0

ip\_protocol = "tcp"

to\_port = 65535

}

resource "aws\_vpc\_security\_group\_egress\_rule" "allow\_inbound\_traffic\_ipv4" {

security\_group\_id = aws\_security\_group.allow\_inbound.id

cidr\_ipv4 = "0.0.0.0/0"

ip\_protocol = "-1" # semantically equivalent to all ports

}

#EC2 Instance1

resource "aws\_instance" "Terra1\_inst" {

ami = "ami-04b4f1a9cf54c11d0"

instance\_type = "t2.micro"

key\_name = "Nivetha"

associate\_public\_ip\_address = true

subnet\_id = aws\_subnet.pub\_sub.id

vpc\_security\_group\_ids = [aws\_security\_group.allow\_all.id]

}

#EC2 Instance2

resource "aws\_instance" "Terra2\_inst" {

ami = "ami-04b4f1a9cf54c11d0"

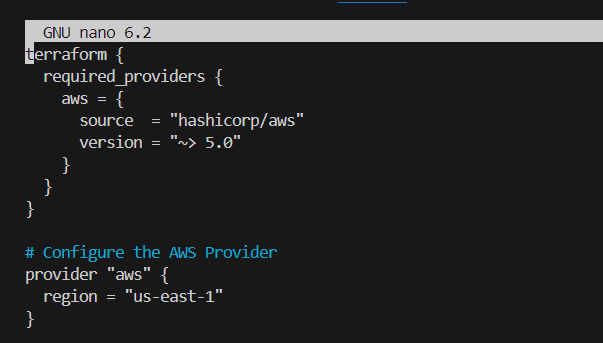
instance\_type = "t2.micro"

key\_name = "Nivetha"

subnet\_id = aws\_subnet.pri\_sub.id

vpc\_security\_group\_ids = [aws\_security\_group.allow\_inbound.id]

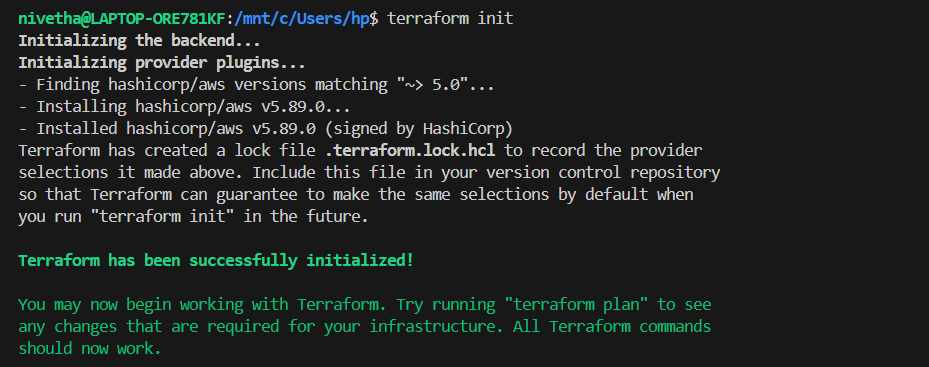
}



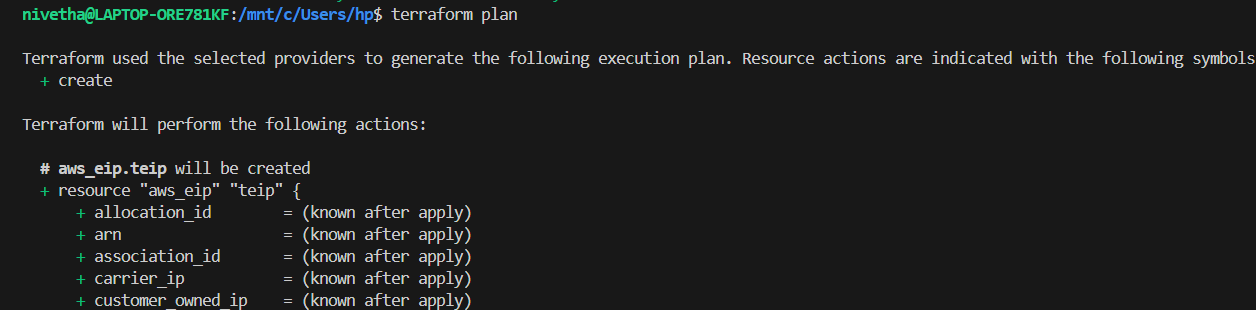
Save and exit.

1. Initialize and apply terraform file.

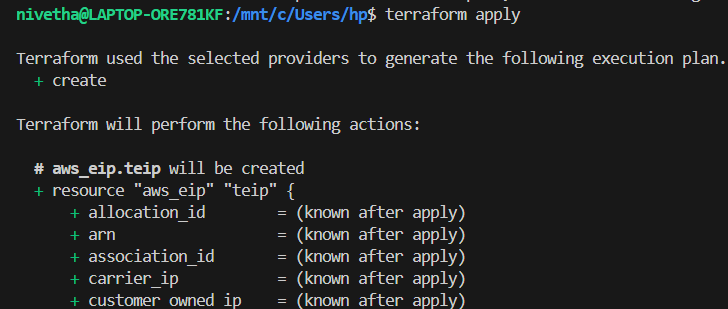
Initialize terraform using terraform init

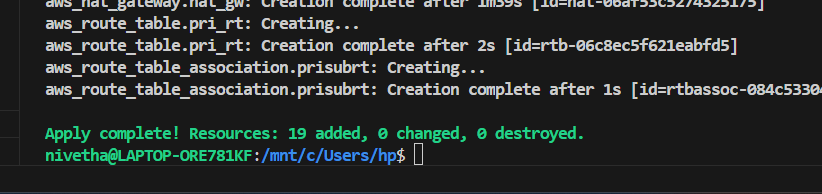


Then validate the given file using terraform plan



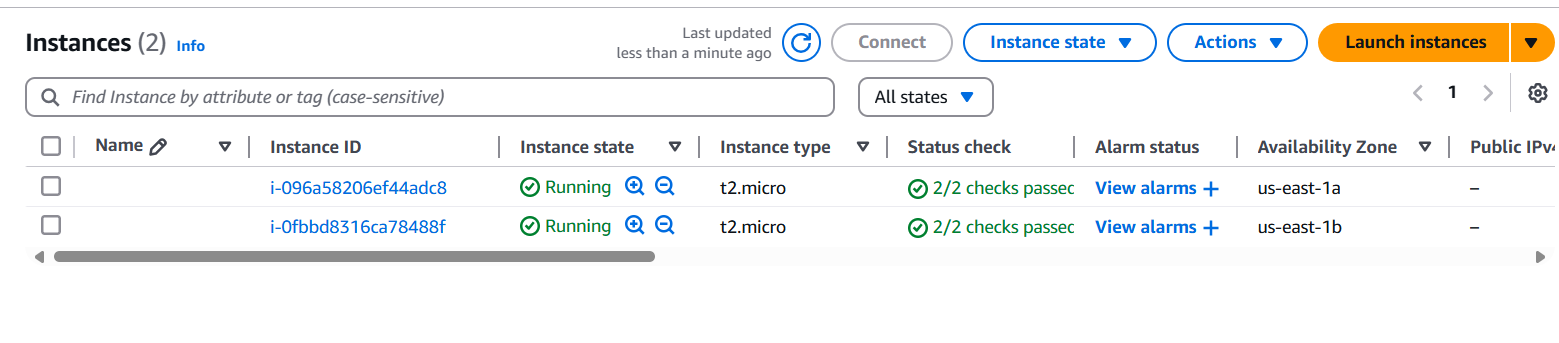
Give terraform apply to launch the instance using terraform file



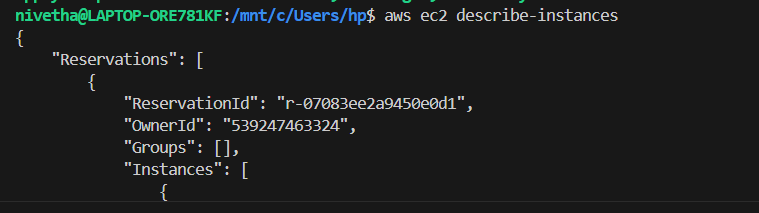


1. Verification

Instances have been successfully launched in my ec2 dashboard.



We can also verfify using aws ec2 describe-instances



Give terraform destroy to remove the instances

