**Terraform Task -2**

**Task Description:**

Create 2 EC2 instances on 2 different regions and install nginx using terraform script.

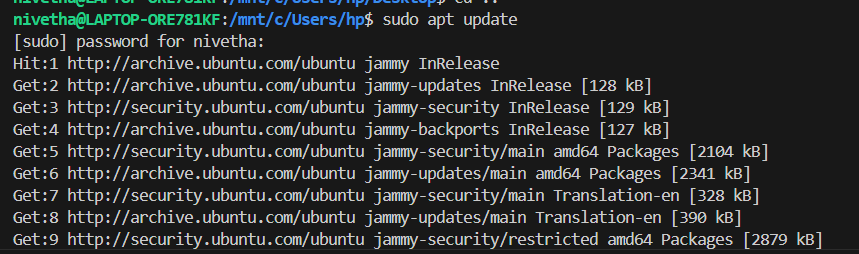
Explanation:

1. Prerequisites:

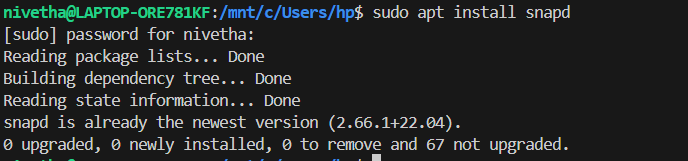
Refer documentation to install aws cli and terraform.

**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)

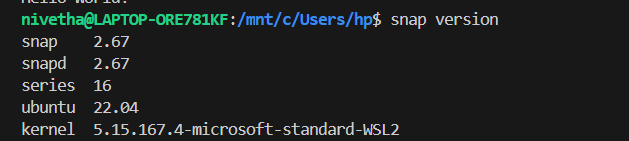
Install snap using the command : sudo apt update



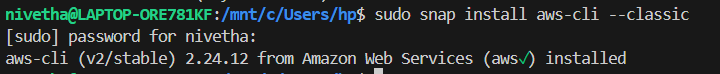
sudo apt install snapd



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)

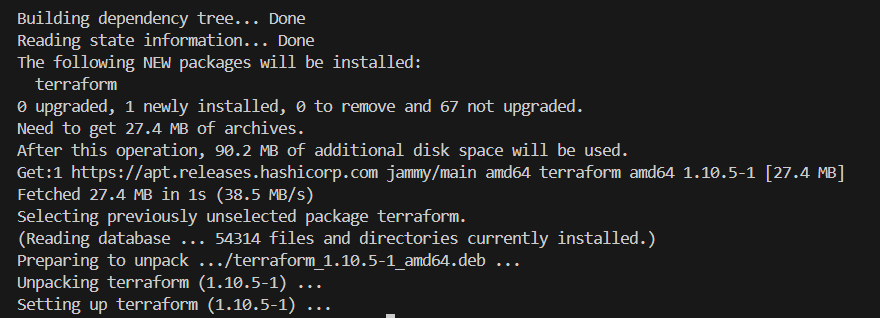
[Install | Terraform | HashiCorp Developer](https://developer.hashicorp.com/terraform/install)

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

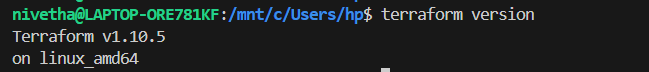
sudo apt update && sudo apt install terraform



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

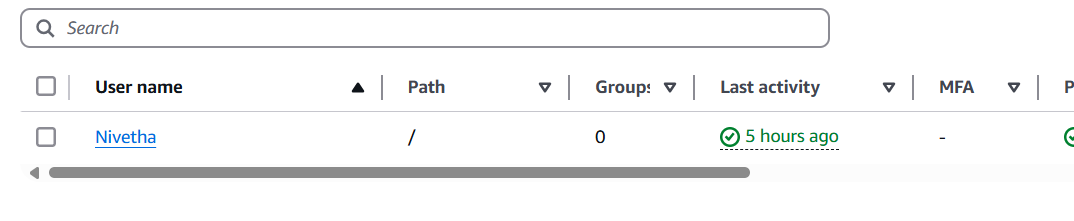
Which terraform



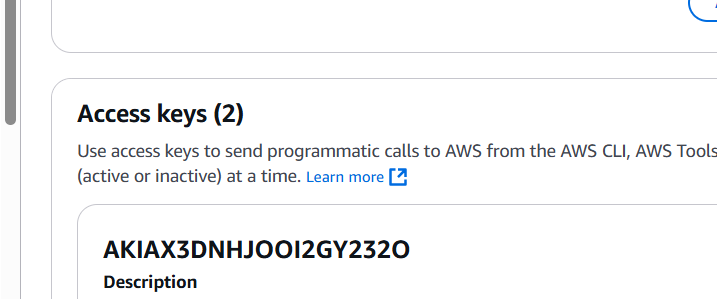


1. Aws configure : to configure aws in ubuntu,

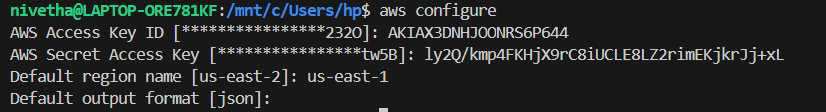
Create user with IAM role and give EC2 full access and vpc full access.

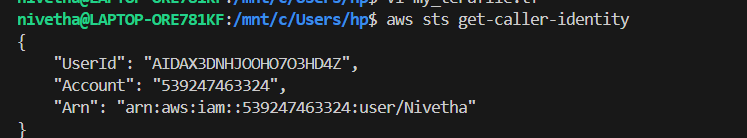


Post creation, go to security tab and create Access key, copy the access key and secret accesskey.



Use aws configure to configure your credentials.



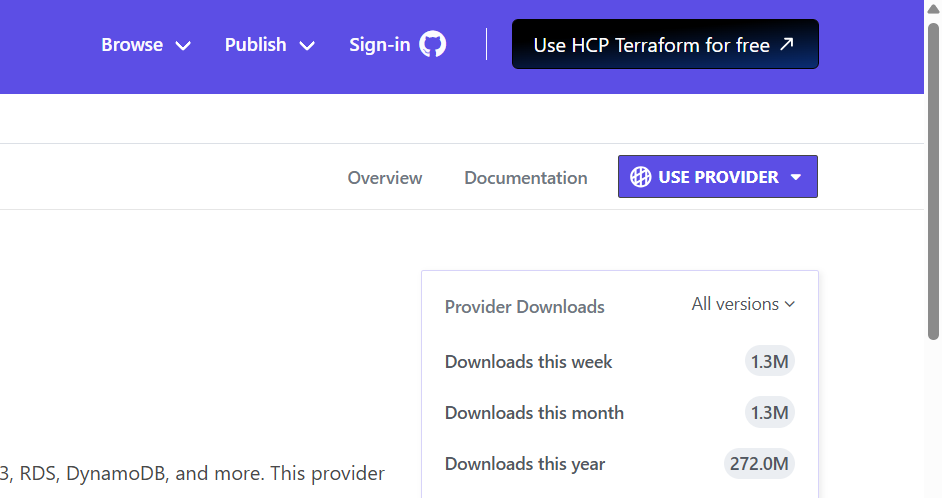


Use aws sts get-caller-identity to verify the credentials have been configured

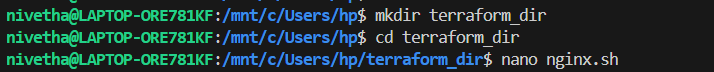
1. Create Terraform config file.

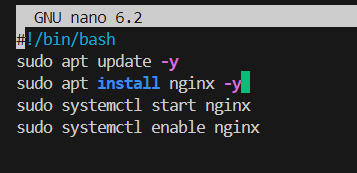
Refer to terraform document, go to terraform hashicorp link - [Terraform by HashiCorp](https://www.terraform.io/)

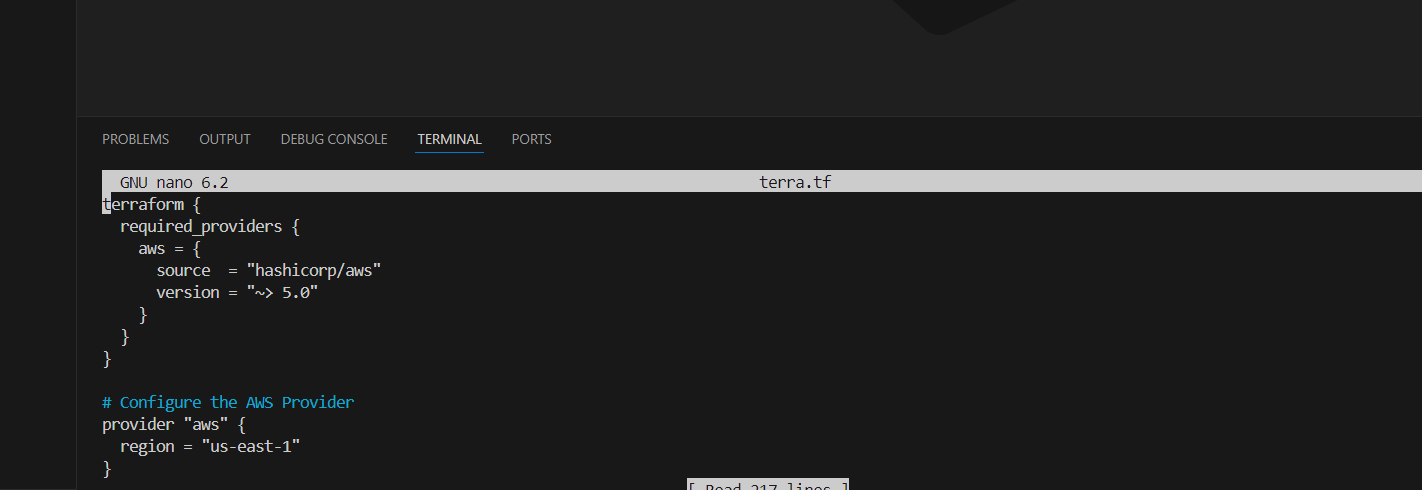
Visit Registry tab > browse provider > choose AWS provider> click documentation.



Create terraform directory and put the nginx file and terraform file using nano.







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" "myvpc1" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "my\_VPC1"

}

}

#subnet

resource "aws\_subnet" "pubsub1" {

vpc\_id = aws\_vpc.myvpc1.id

cidr\_block = "10.0.2.0/24"

tags = {

Name = "My\_Pub\_Sub1"

}

}

#IGW

resource "aws\_internet\_gateway" "igw1" {

vpc\_id = aws\_vpc.myvpc1.id

tags = {

Name = "My\_IGW1"

}

}

#Route table

resource "aws\_route\_table" "pubrt1" {

vpc\_id = aws\_vpc.myvpc1.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.igw1.id

}

tags = {

Name = "Pub\_RT1"

}

}

#Route table association

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

subnet\_id = aws\_subnet.pubsub1.id

route\_table\_id = aws\_route\_table.pubrt1.id

}

#security groups

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

name = "allow\_tls"

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

vpc\_id = aws\_vpc.myvpc1.id

tags = {

Name = "allow\_tls"

}

}

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

security\_group\_id = aws\_security\_group.allow\_tls.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\_http\_ipv4" {

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

cidr\_ipv4 = "0.0.0.0/0"

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

}

resource "aws\_instance" "terra\_instance1" {

ami = "ami-04b4f1a9cf54c11d0"

instance\_type = "t2.micro"

key\_name = "Nivetha"

associate\_public\_ip\_address = true

subnet\_id = aws\_subnet.pubsub1.id

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

user\_data = file("nginx.sh")

}

# Configure the AWS Provider

provider "aws" {

alias = "ohio"

region = "us-east-2"

}

#VPC

resource "aws\_vpc" "myvpc2" {

provider = aws.ohio

cidr\_block = "10.1.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "my\_VPC2"

}

}

#subnet

resource "aws\_subnet" "pubsub2" {

provider = aws.ohio

vpc\_id = aws\_vpc.myvpc2.id

cidr\_block = "10.1.2.0/24"

tags = {

Name = "My\_Pub\_Sub2"

}

}

#IGW

resource "aws\_internet\_gateway" "igw2" {

provider = aws.ohio

vpc\_id = aws\_vpc.myvpc2.id

tags = {

Name = "My\_IGW2"

}

}

#Route table

resource "aws\_route\_table" "pubrt2" {

provider = aws.ohio

vpc\_id = aws\_vpc.myvpc2.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.igw2.id

}

tags = {

Name = "Pub\_RT2"

}

}

#Route table association

resource "aws\_route\_table\_association" "rtasps2" {

provider = aws.ohio

subnet\_id = aws\_subnet.pubsub2.id

route\_table\_id = aws\_route\_table.pubrt2.id

}

#security groups

resource "aws\_security\_group" "allow\_tls2" {

provider = aws.ohio

name = "allow\_tls2"

description = "Allow TLS2 inbound traffic and all outbound traffic2"

vpc\_id = aws\_vpc.myvpc2.id

tags = {

Name = "allow\_tls2"

}

}

resource "aws\_vpc\_security\_group\_ingress\_rule" "allow\_tls\_ipv42" {

provider = aws.ohio

security\_group\_id = aws\_security\_group.allow\_tls2.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\_http\_ipv42" {

provider = aws.ohio

security\_group\_id = aws\_security\_group.allow\_tls2.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\_ipv42" {

provider = aws.ohio

security\_group\_id = aws\_security\_group.allow\_tls2.id

cidr\_ipv4 = "0.0.0.0/0"

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

}

resource "aws\_instance" "terra\_instance2" {

provider = aws.ohio

ami = "ami-0cb91c7de36eed2cb"

instance\_type = "t2.micro"

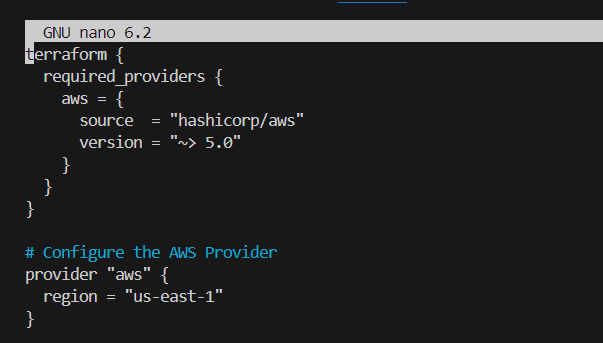
associate\_public\_ip\_address = true

subnet\_id = aws\_subnet.pubsub2.id

vpc\_security\_group\_ids = [aws\_security\_group.allow\_tls2.id]

user\_data = file("nginx.sh")

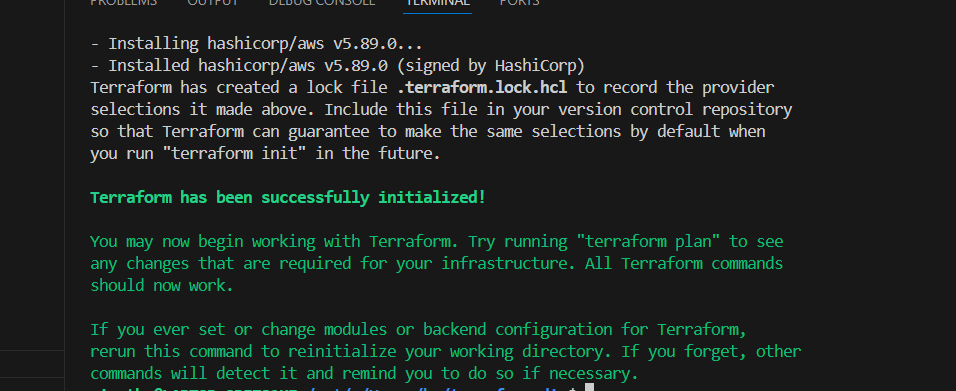
}



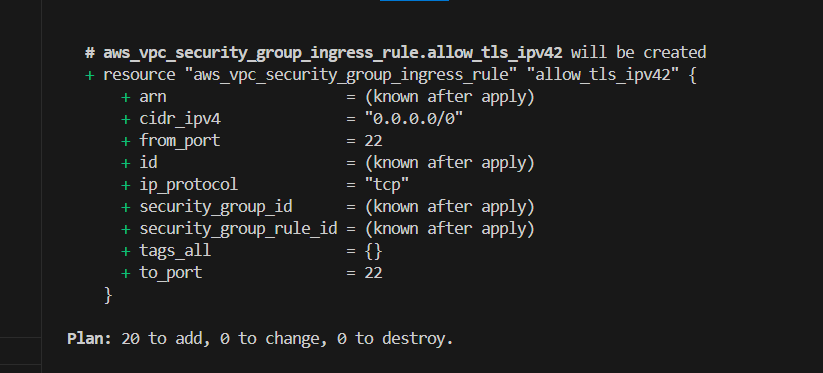
Save and exit.

1. Initialize and apply terraform file.

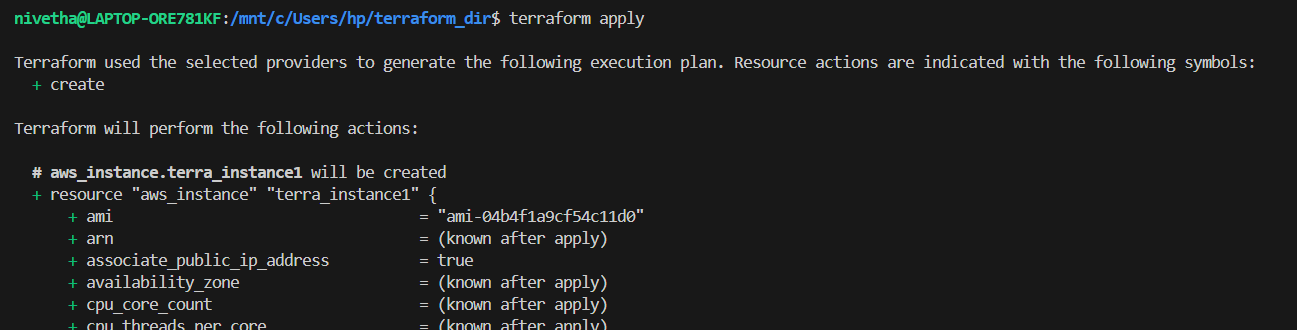
Initialize terraform using terraform init

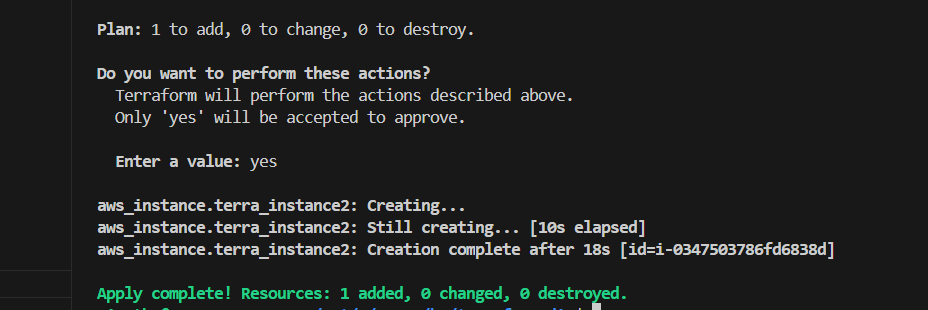


Then validate the given file using terraform plan



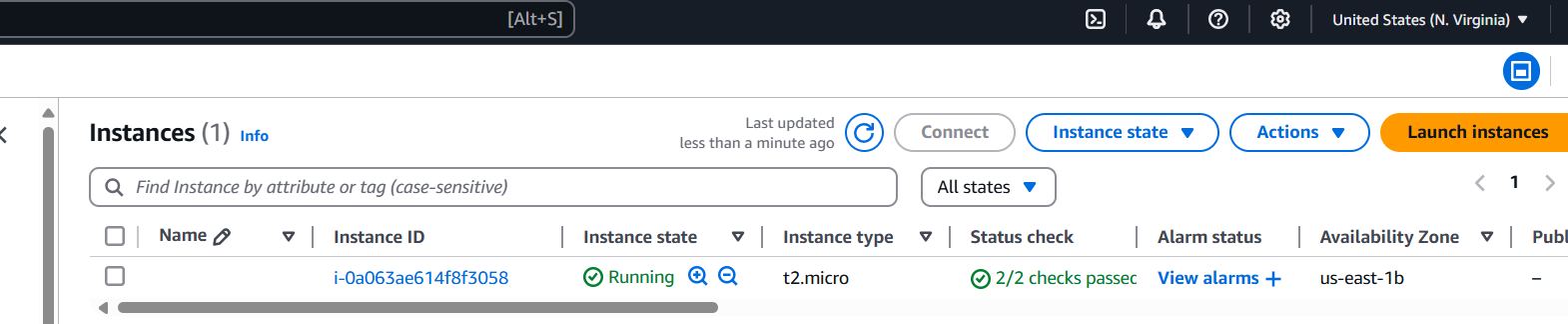
Give terraform apply to launch the instance using terraform file

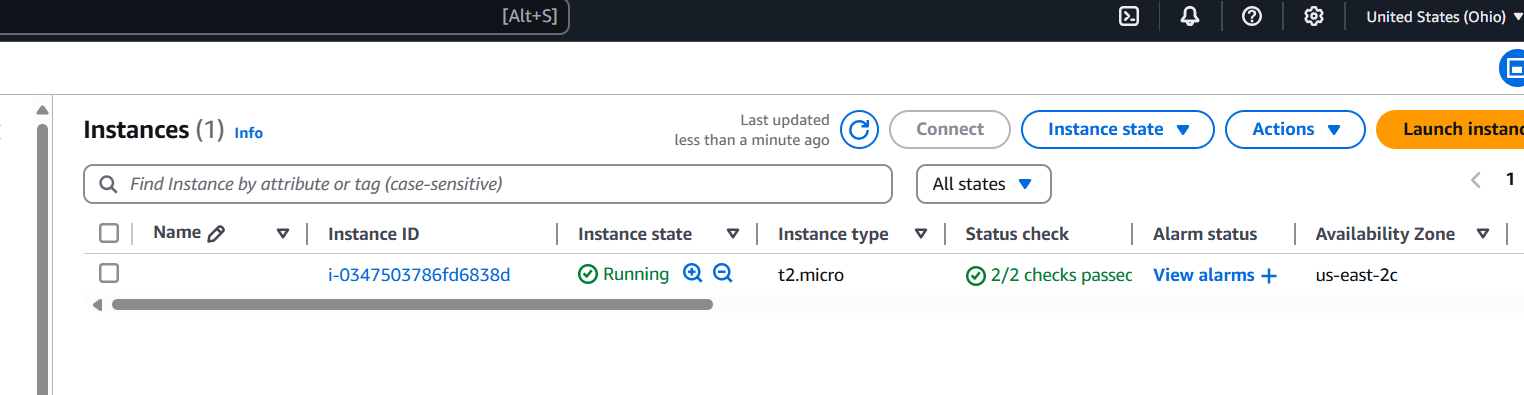




1. Verification

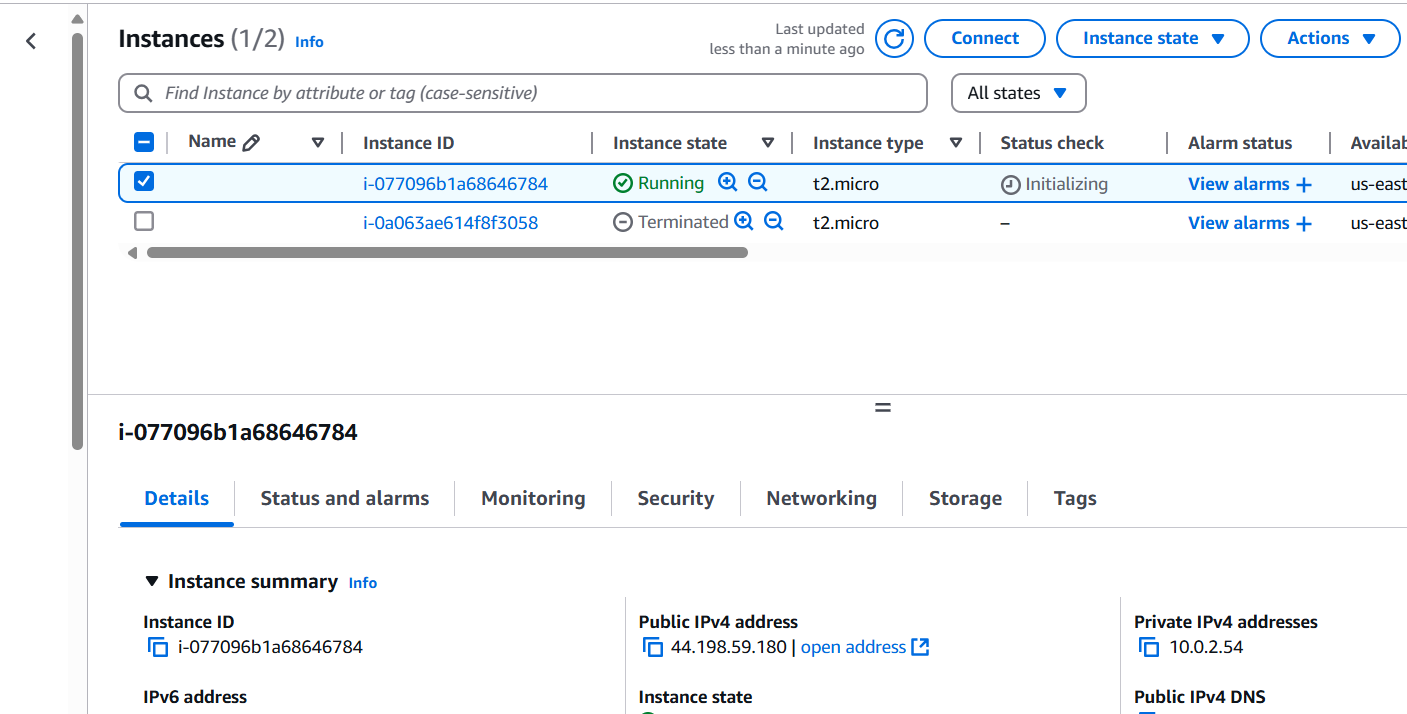
Two Instances from different regions have been successfully launched in my ec2 dashboard.

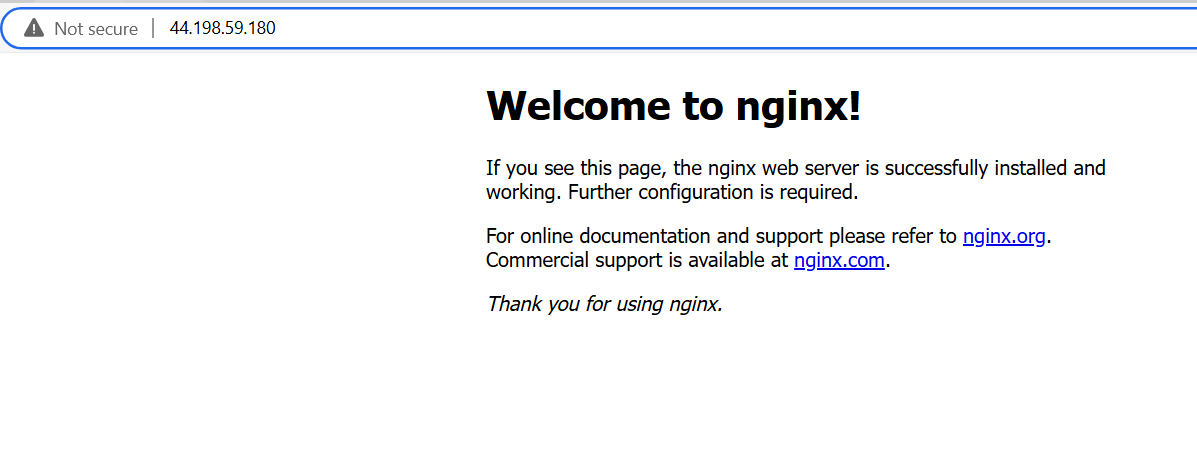




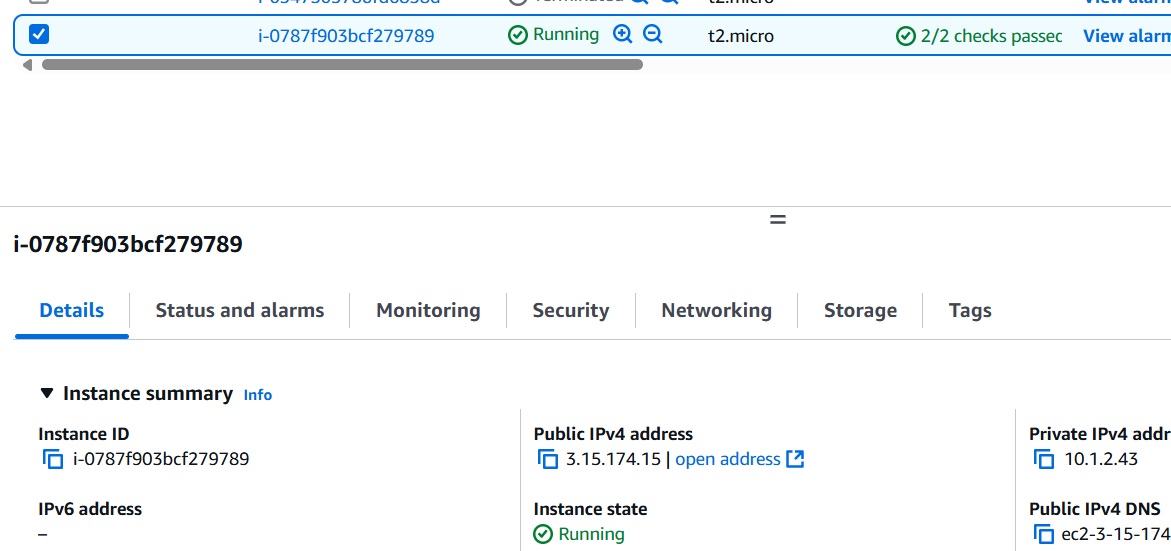
Copy the public v4 and verify in browser using http://<publicv4>

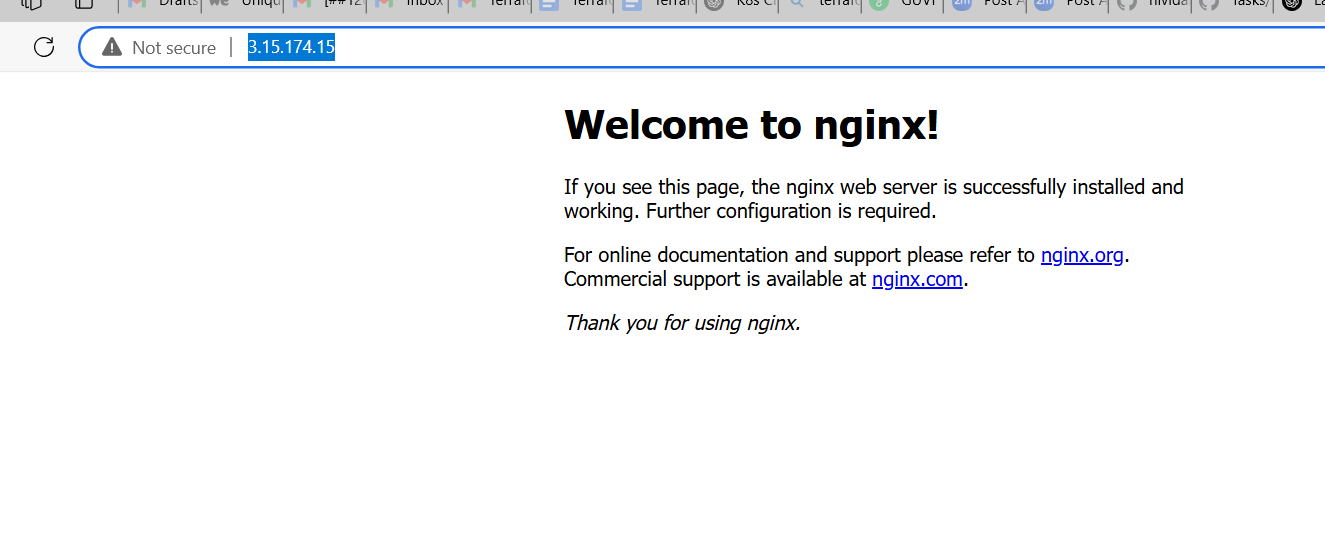
Instance 1





Instance 2



Give terraform destroy to remove the instances

