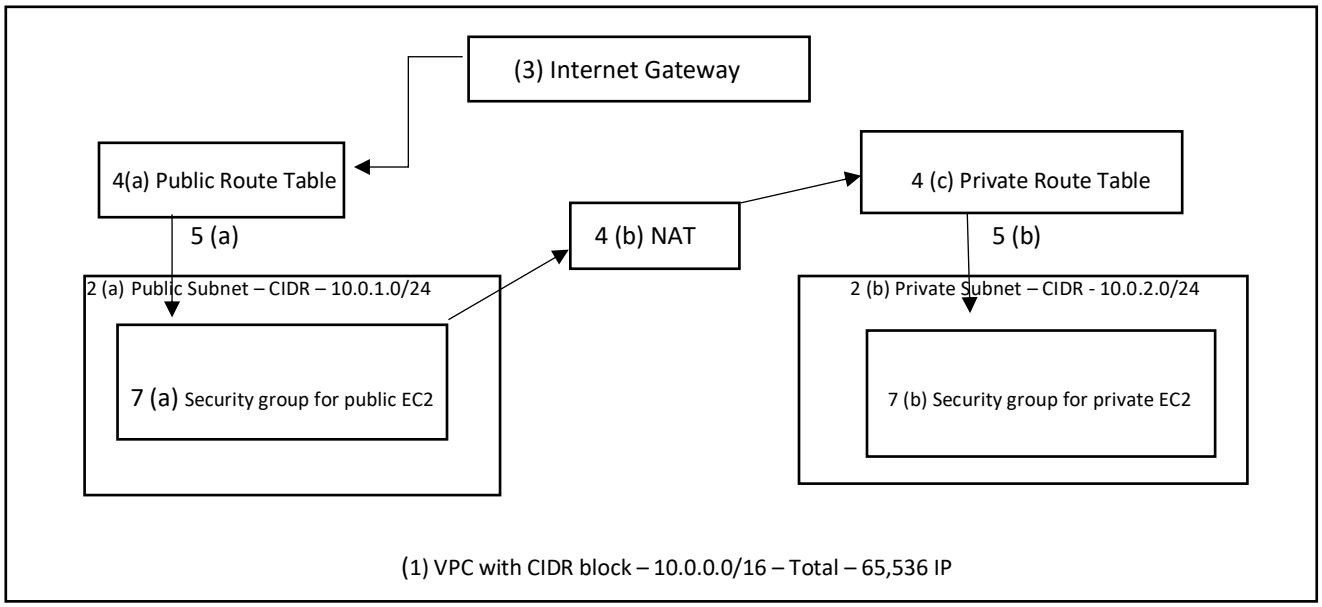
Automate AWS Network Infrastructure Virtual Private Cloud Deployment using Terraform DevOps Tool

TERRAFORM - It is an Infrastructure as a code tool, used primarily by DevOps teams to automate infrastructure tasks.

To understand the concept of Virtual private cloud before writing the terraform code for VPC creation.



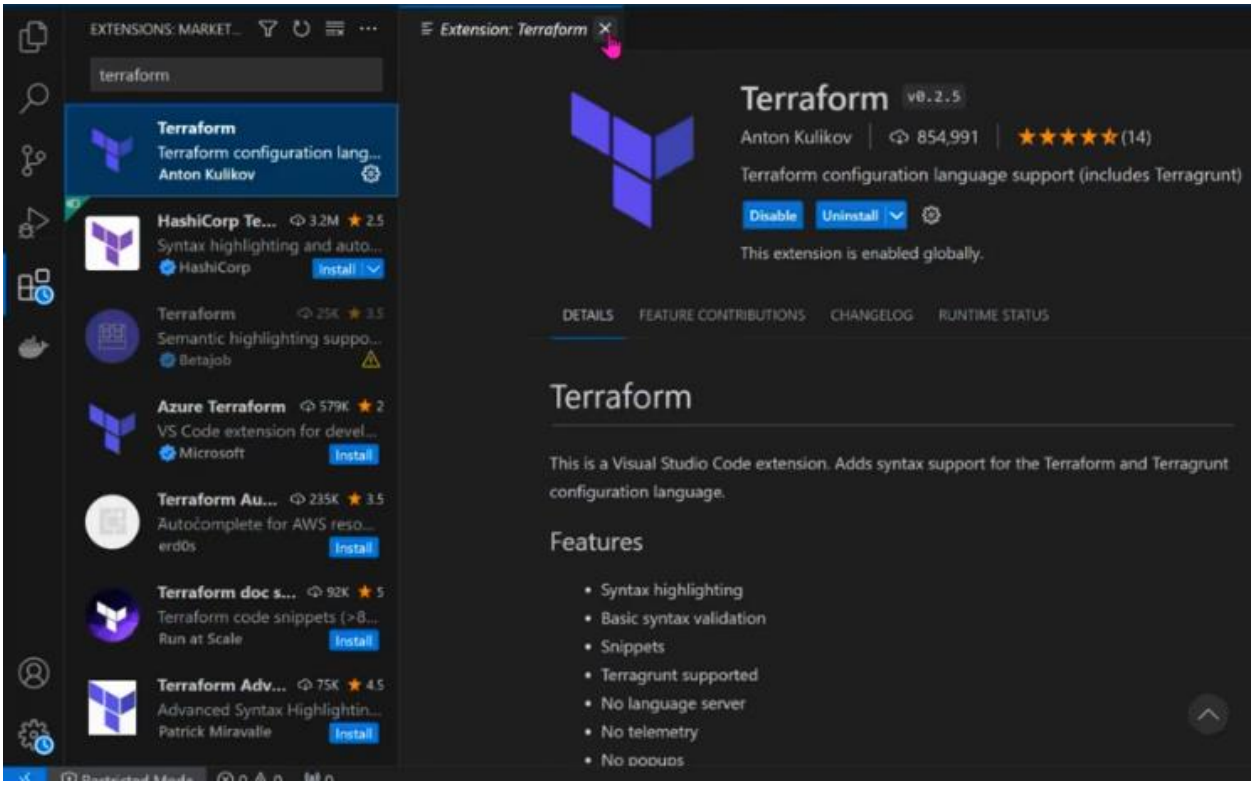
Virtual Private Cloud

Now we have to write the terraform hashicorp programming language to automate AWS Network Infrastructure Virtual Private Cloud Deployment. Refer the terraform hashicorp programming language from terraform registry official website.

We have to write the code in **visual studio code.**

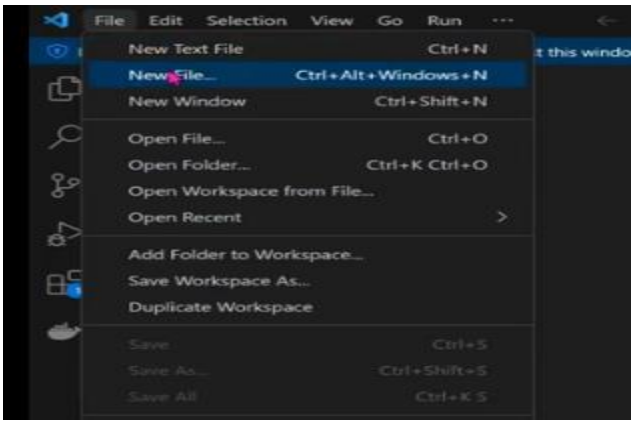
The Main purpose of using Visual studio code: Syntax highlighting, Basic syntax validation, Terragrunt supported.

Before writing the code in visual studio code, we have to complete terraform setup in visual studio code with file name extension as ".tf "



Create the new file with file extension as ".tf"

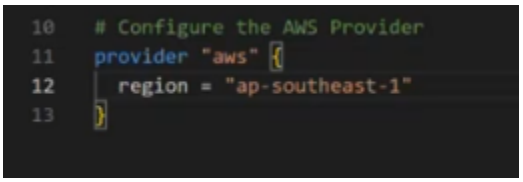
tf -> **T**erra**f**orm



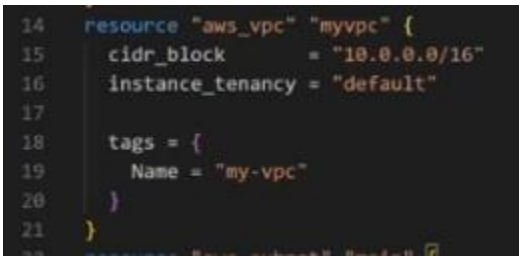
Now we will start to write the terraform hashicorp programming code to automate AWS Network Infrastructure Virtual Private Cloud Deployment.

Step: 1 -> we will setup terraform cloud provider plugins such as Aws, Azure, GCP cloud etc. Now we will select terraform AWS cloud plugins

Step: 2 -> # Configure the AWS Provider with region = “ap-southeast-1”



Step: 3 -> # Create VPC with CIDR = “10.0.0.0/16” (10.0.0.0 to 10.0.255.255 = Total -> 65,536 Ip Address)



Step: 4 -> To create two subnets. One is for public subnet and another one is private subnets with two different availability zones and CIDR_block.

Step: 5 -> To create internet gateway and attach it to VPC and Public route table.

```
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.myvpc.id

  tags = {
    Name = "myinternet"
  }
}
```

Step: 6 -> To create two route table. One is for public route table and another one is for private route table.

(a): To create public route table.

```
resource "aws_route_table" "pubrt" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }

  tags = {
    Name = "my-pubrt"
  }
}
```

(b): We have to create Elastic IP address and Network Address Translation (NAT) before private route table creation. After that we have attach Elastic IP address and Public subnet to NAT.

```
resource "aws_eip" "elasticip" {
  domain   = "vpc"
}
resource "aws_nat_gateway" "mynat" {
  allocation_id = aws_eip.elasticip.id
  subnet_id     = aws_subnet.pubsub.id

  tags = {
    Name = "my-NAT"
  }
}
```

(c): To create Private Route Table and attach it to NAT to give internet to private network.

```
resource "aws_route_table" "prirt" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.mynat.id
  }

  tags = {
    Name = "my-pubrt"
  }
}
```

Step: 7 -> Route Table association -> To attach public and private subnets to the route table

```
resource "aws_route_table_association" "pubass" {
  subnet_id      = aws_subnet.pubsub.id
  route_table_id = aws_route_table.pubrt.id
}
resource "aws_route_table_association" "priass" {
  subnet_id      = aws_subnet.prisub.id
  route_table_id = aws_route_table.prirt.id
}
```

Step: 8 -> To create Security Group and attach it to VPC.

```
resource "aws_security_group" "allowall" {
  name        = "allow_tls"
  description = "Allow TLS inbound traffic"
  vpc_id      = aws_vpc.myvpc.id
```

```
  ingress {
    description      = "TLS from VPC"
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }
  ingress {
    description      = "TLS from VPC"
    from_port        = 80
    to_port          = 80
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }


  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
  }
```

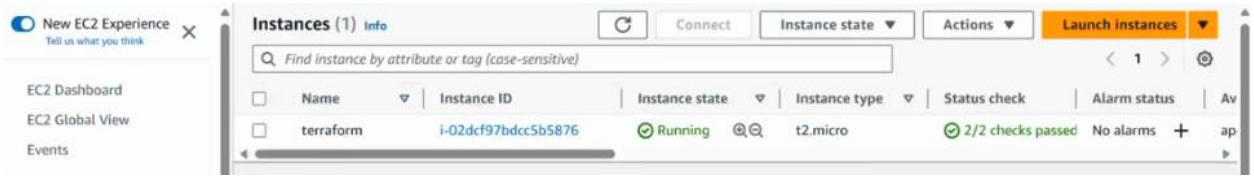
```
  tags = {
    Name = "security"
  }
}
```

Step: 9 -> To create 2 virtual Machines. One is for public and another one is for private.

```
resource "aws_instance" "public" {
    ami                         = "ami-0fa7190e664488b99"
    instance_type               = "t2.micro"
    subnet_id                   = aws_subnet.pubsub.id
    vpc_security_group_ids      = [aws_security_group.allowall.id]
    key_name                    = "ppksing"
    associate_public_ip_address = true
}
resource "aws_instance" "private" {
    ami                         = "ami-0fa7190e664488b99"
    instance_type               = "t2.micro"
    subnet_id                   = aws_subnet.prisub.id
    vpc_security_group_ids      = [aws_security_group.allowall.id]
    key_name                    = "ppksing"
}
```

Step: 10 -> To create virtual server (EC2 Instance) in AWS for terraform setup in linux.

To login as Ubuntu linux by using public key




```
login as: ubuntu
Authenticating with public key "ppksing"
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Oct  5 05:52:50 UTC 2023
```

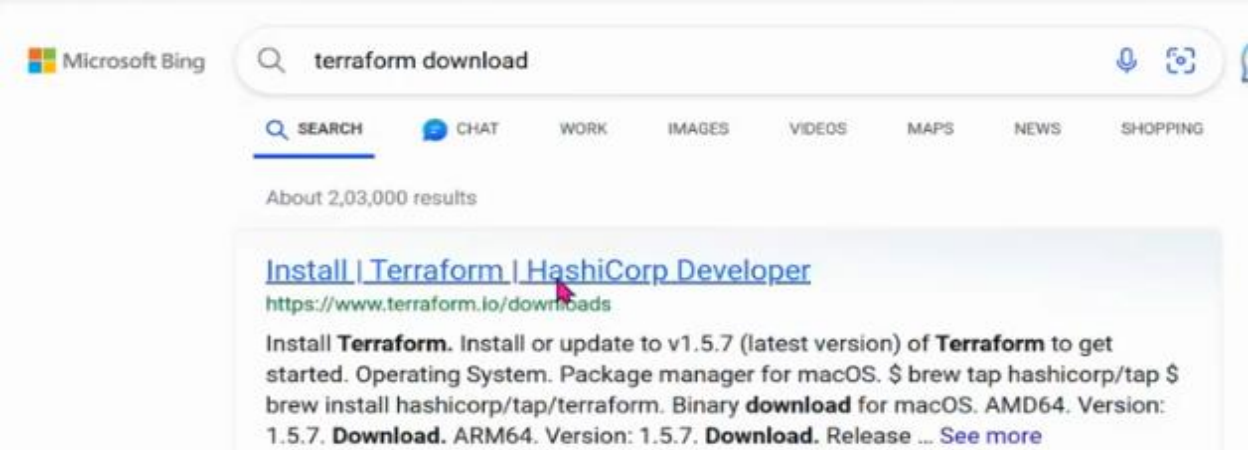
```
  System load:  0.0                Processes:              94
  Usage of /:   20.6% of 7.57GB    Users logged in:        0
  Memory usage: 23%                IPv4 address for eth0: 172.31.35.78
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

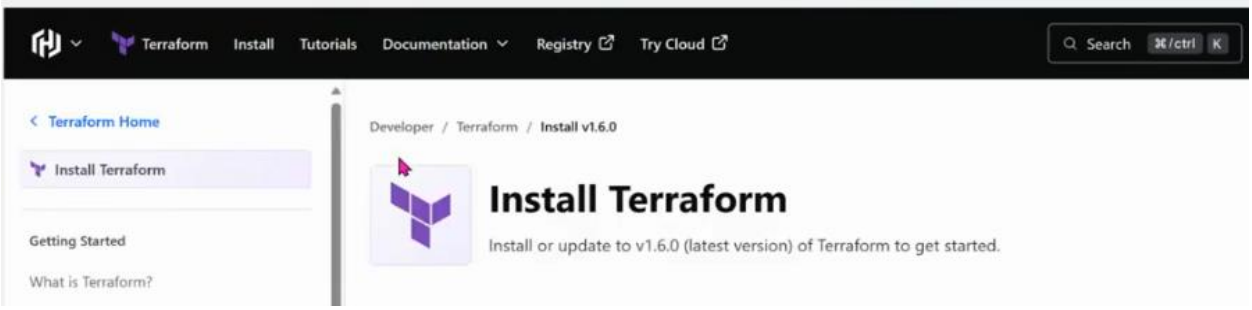
To convert root user / admin user using command as “sudo –i”

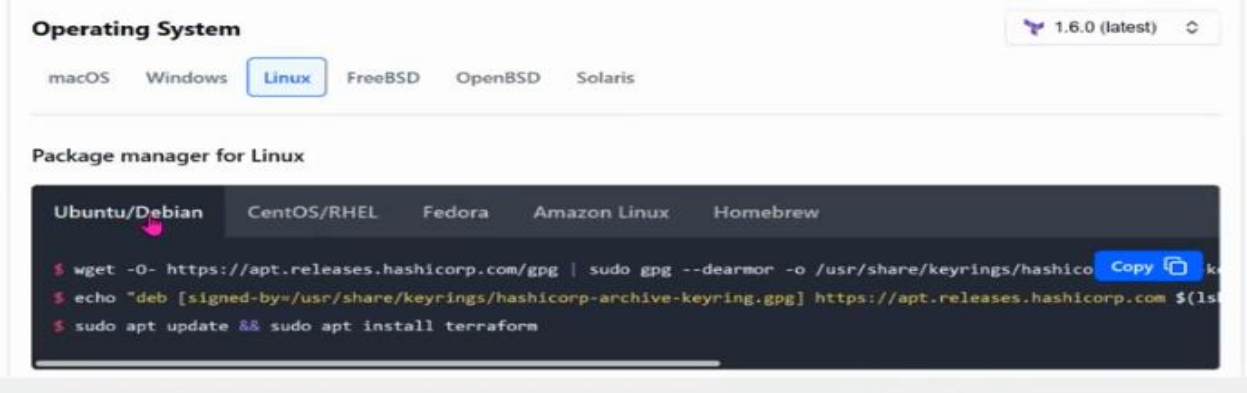
```
ubuntu@ip-172-31-35-78:~$ sudo -i
root@ip-172-31-35-78:~# clear
```

Now we have to install terraform in Ubuntu linux after logged in Ubuntu linux.

Go to Terraform Official website







Copy the above commands and paste in Ubuntu linux

```
root@ip-172-31-35-78:~#
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyr
ing.gpg
--2023-10-05 05:53:14--  https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 18.155.68.89, 18.155.68.116, 18.155.68.43, ..
```

```
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|18.155.68.89|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'

-                           100%[=========================================>]   3.89K  --.-KB/s    in 0s

2023-10-05 05:53:14 (1.21 GB/s) - written to stdout [3980/3980]
```

```
root@ip-172-31-35-78:~# echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releas
es.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com jammy main
```

The sudo apt update command is a Linux system administration command that updates the list of available packages and their versions stored in the system's package index

```
root@ip-172-31-35-78:~#
sudo apt update
```

```
Get:4 https://apt.releases.hashicorp.com jammy InRelease [12.9 kB]
Get:5 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:7 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:8 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:9 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:10 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:11 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:12 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1057 kB]
Get:13 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [232 kB]
Get:14 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [15.6 kB]
Get:15 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [969 kB]
Get:16 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [156 kB]
Get:17 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [532 B]
Get:18 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [989 kB]
Get:19 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [216 kB]
Get:20 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [21.9 kB]
Get:21 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [41.6 kB]
Get:22 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [9768 B]
Get:23 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 B]
Get:24 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [41.7 kB]
Get:25 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [10.5 kB]
Get:26 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:27 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:28 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [24.3 kB]
Get:29 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.4 kB]
Get:30 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [640 B]
Get:31 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:32 https://apt.releases.hashicorp.com jammy/main amd64 Packages [100 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [856 kB]
```

```
Get:34 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [175 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [11.4 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [953 kB]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [154 kB]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [532 B]
Get:39 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [788 kB]
Get:40 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [144 kB]
Get:41 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.7 kB]
Get:42 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [36.5 kB]
Get:43 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7060 B]
Get:44 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 27.8 MB in 5s (5467 kB/s)
```

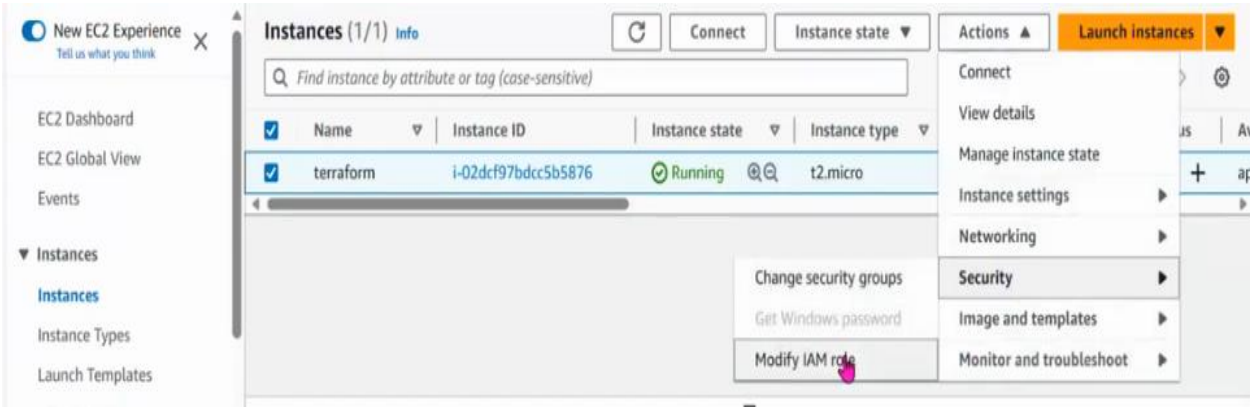
To install terraform by using command as # sudo apt install terraform

```
root@ip-172-31-35-78:~# sudo apt install terraform
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  terraform
0 upgraded, 1 newly installed, 0 to remove and 129 not upgraded.
Need to get 25.6 MB of archives.
After this operation, 80.7 MB of additional disk space will be used.
Get:1 https://apt.releases.hashicorp.com jammy/main amd64 terraform amd64 1.6.0-1 [25.6 MB]
Fetched 25.6 MB in 0s (76.1 MB/s)
Selecting previously unselected package terraform.
(Reading database ... 64295 files and directories currently installed.)
Preparing to unpack .../terraform_1.6.0-1_amd64.deb ...
Unpacking terraform (1.6.0-1) ...
Setting up terraform (1.6.0-1) ...
Scanning processes...
Scanning linux images...
```

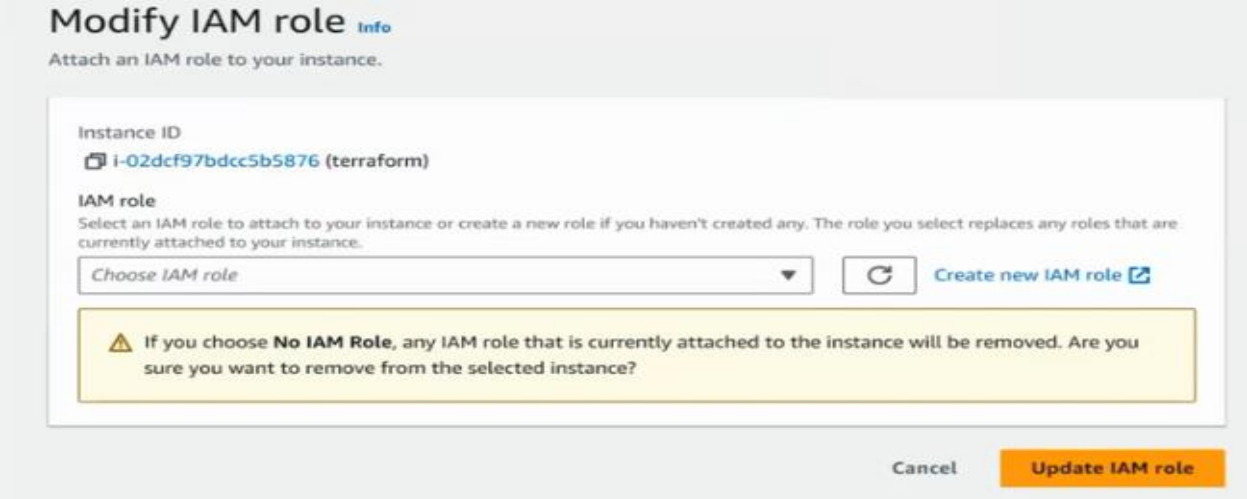
Now we have to check whether the terraform installed or not

```
root@ip-172-31-35-78:~# terraform --version
Terraform v1.6.0
on linux_amd64
```

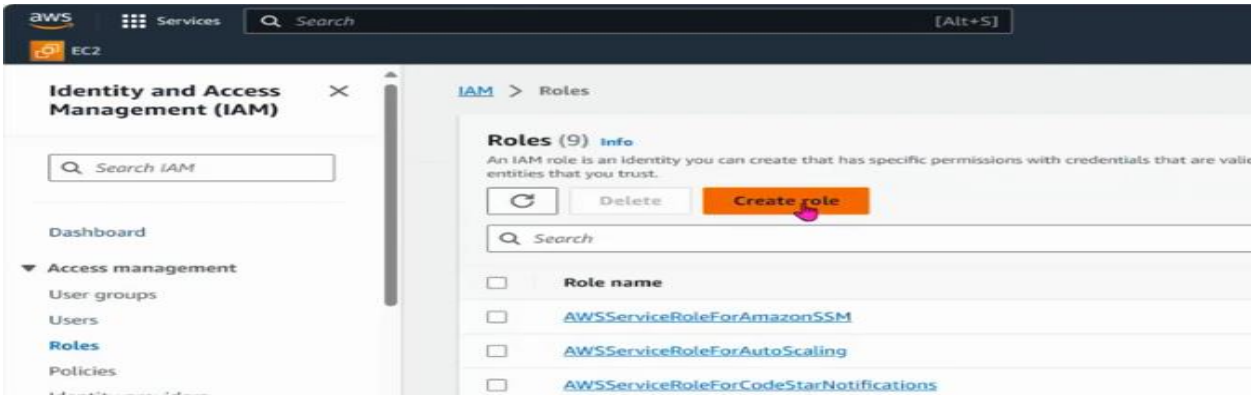
Now we have to create IAM role to give permission to terraform virtual machine to build VPC infrastructure.



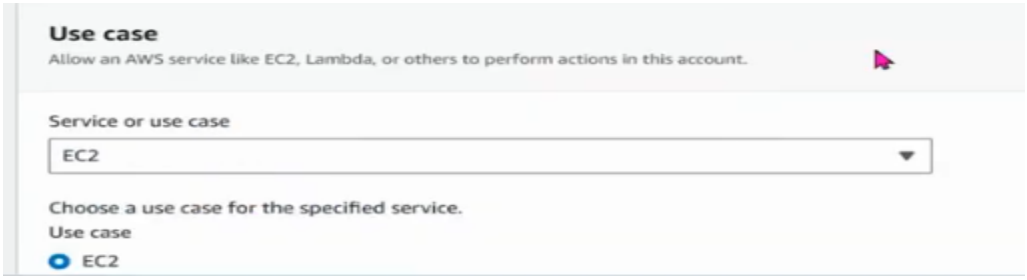
Click on “Create new IAM role”

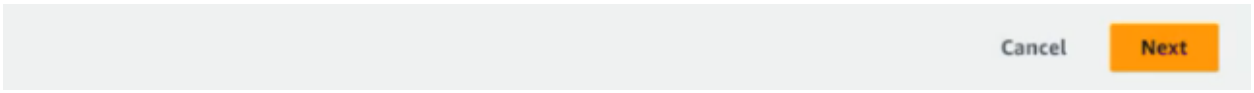


Click on create role

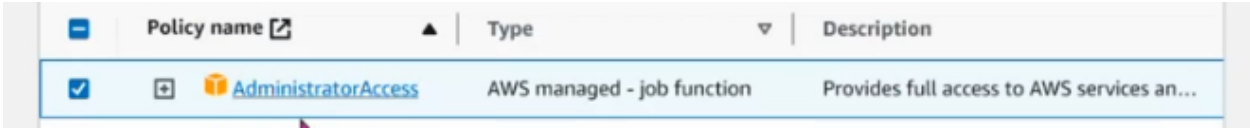


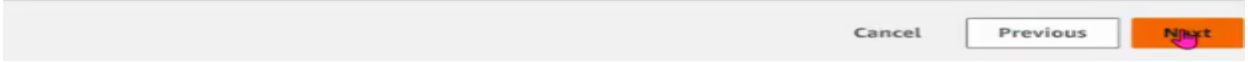
Select EC2 and click on next



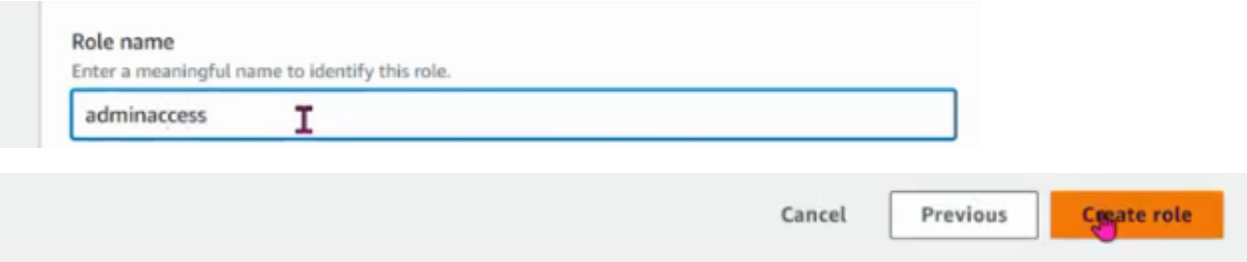


Now we have to give full admin access and click on “Next”

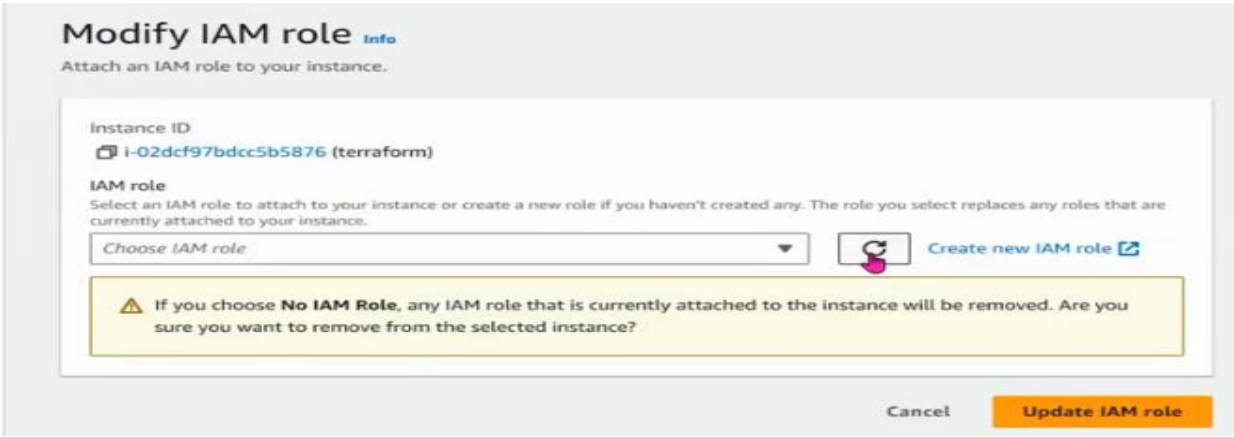


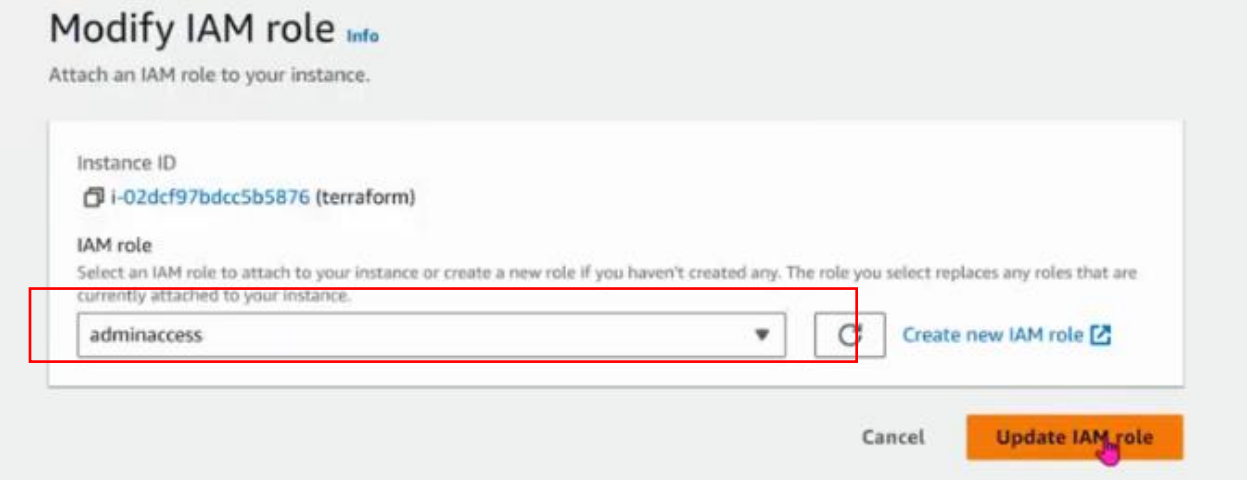


Set name of the role and click on “Create Role”



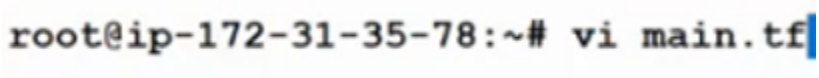
Now refresh IAM role and select “adminaccess” role which we have created and click on “Update IAM role”





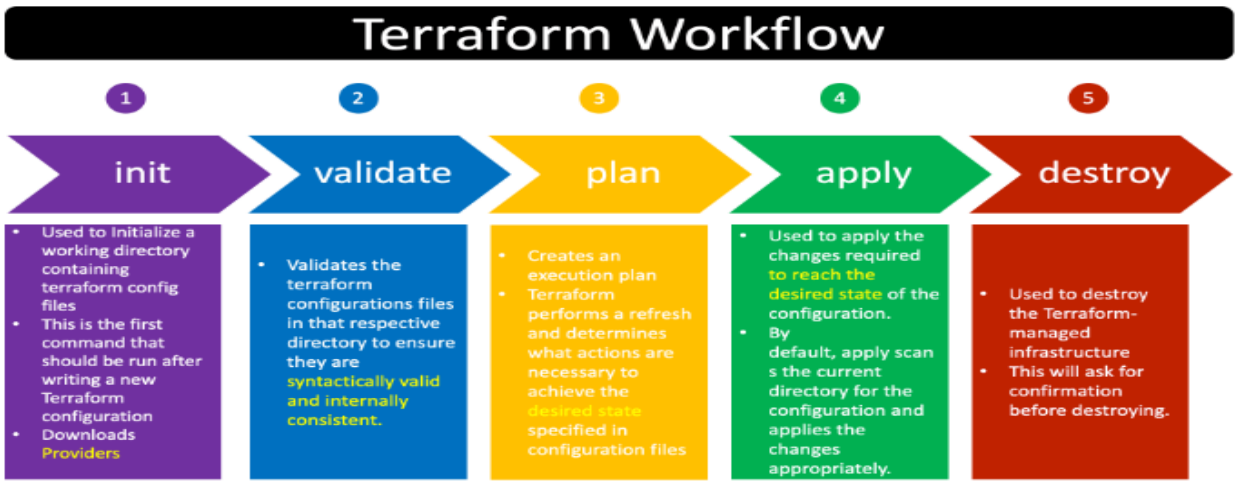
Now, we have successfully attached IAM role to EC2 instance.

Now, we have to create the file in Ubuntu linux then copy the entire code from visual studio code and paste in **main.tf file** then save and quit.



```
  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  tags = {
    Name = "security"
  }
}
resource "aws_instance" "public" {
    ami                         = "ami-0fa7190e664488b99"
    instance_type               = "t2.micro"
    subnet_id                   = aws_subnet.pubsub.id
    vpc_security_group_ids      = [aws_security_group.allowall.id]
    key_name                    = "ppksing"
    associate_public_ip_address = true
}
resource "aws_instance" "private" {
    ami                         = "ami-0fa7190e664488b99"
    instance_type               = "t2.micro"
    subnet_id                   = aws_subnet.prisub.id
    vpc_security_group_ids      = [aws_security_group.allowall.id]
    key_name                    = "pemsing"
}

:wq!
```

Now we need to apply terraform workflow command one by one in linux.



- **Terraform init** command to initiazing provider plugins in backend.

```
root@ip-172-31-35-78:~# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.19.0...
- Installed hashicorp/aws v5.19.0 (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.
```

- **terraform validate command** is used to verify the correctness of Terraform configuration files

```
terraform validate
```

```
Success! The configuration is valid.
```

- The **terraform plan command** creates a plan consisting of a set of changes that will make your resources match your configuration.

```
terraform plan
```

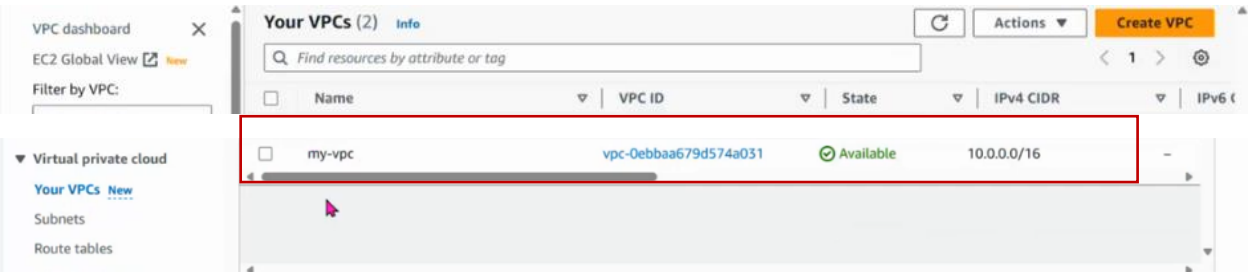
```
root@ip-172-31-35-78:~# terraform plan

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

Terraform will perform the following actions:
```

```
      + arn                                    = (known after apply)
      + cidr_block                             = "10.0.0.0/16"
      + default_network_acl_id                 = (known after apply)
      + default_route_table_id                 = (known after apply)
      + default_security_group_id              = (known after apply)
      + dhcp_options_id                        = (known after apply)
      + enable_dns_hostnames                   = (known after apply)
      + enable_dns_support                     = true
      + enable_network_address_usage_metrics   = (known after apply)
      + id                                     = (known after apply)
      + instance_tenancy                       = "default"
      + ipv6_association_id                    = (known after apply)
      + ipv6_cidr_block                        = (known after apply)
      + ipv6_cidr_block_network_border_group   = (known after apply)
      + main_route_table_id                    = (known after apply)
      + owner_id                               = (known after apply)
      + tags                                   = {
          + "Name" = "my-vpc"
        }
      + tags_all                               = {
          + "Name" = "my-vpc"
        }
    }

Plan: 13 to add, 0 to change, 0 to destroy.
```

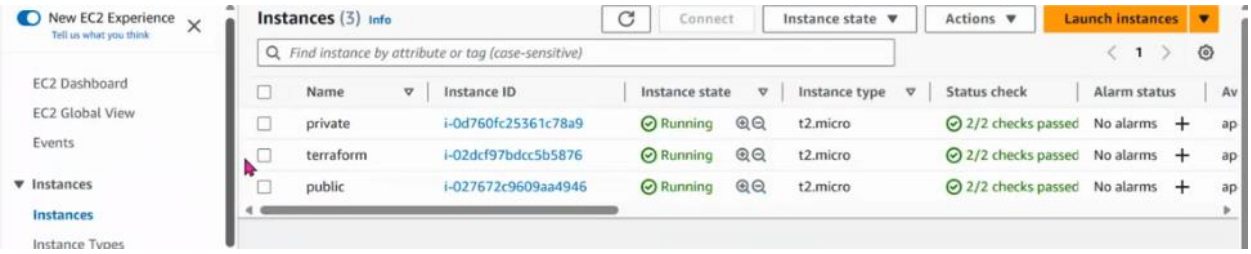
- **terraform apply –auto-approve is the final command for execution**

```
terraform apply --auto-approve
```

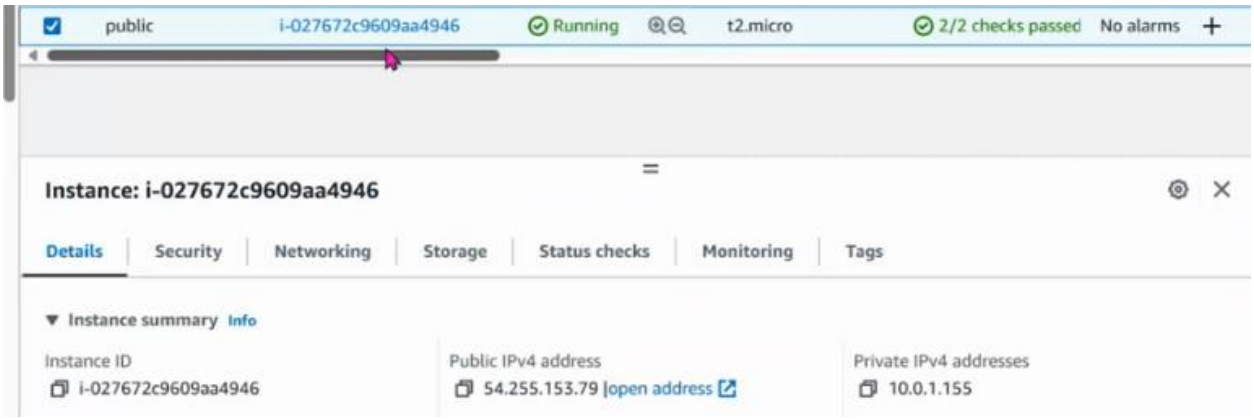
Now Virtual Private Cloud has been successfully created on AWS cloud.



Now two virtual machines (Public and private) has been successfully created



Now we have to connect public virtual machine and to check whether the internet is working or not.

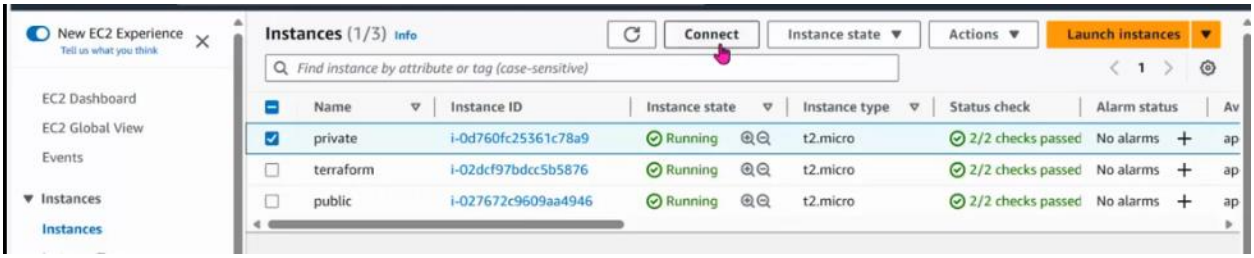


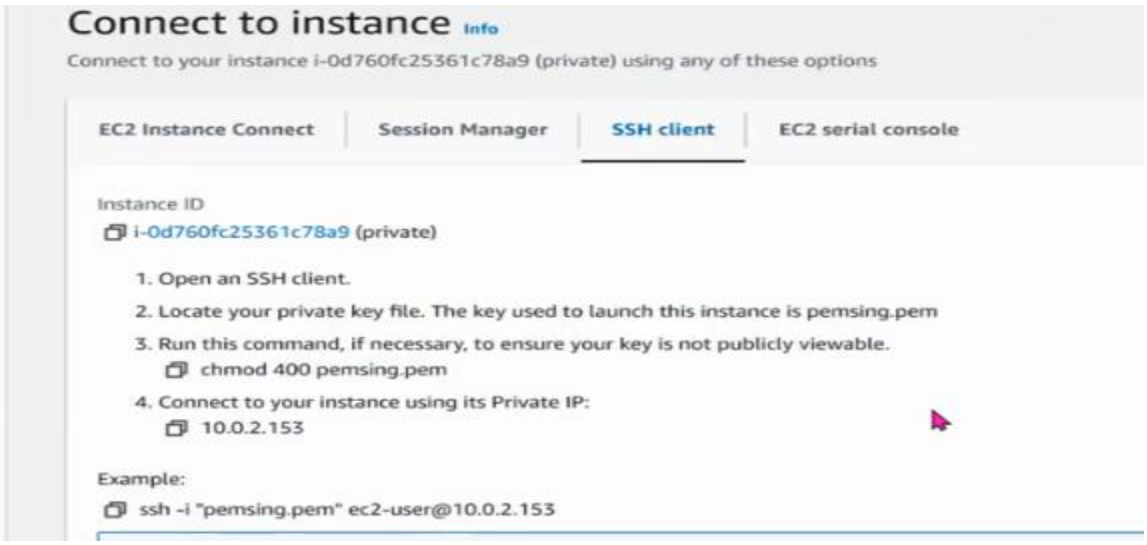

```
[root@ip-10-0-1-155 ~]# ping google.com
PING google.com (74.125.130.101) 56(84) bytes of data.
64 bytes from sb-in-f101.1e100.net (74.125.130.101): icmp_seq=1 ttl=51 time=1.93 ms
64 bytes from sb-in-f101.1e100.net (74.125.130.101): icmp_seq=2 ttl=51 time=1.92 ms
64 bytes from sb-in-f101.1e100.net (74.125.130.101): icmp_seq=3 ttl=51 time=1.98 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.927/1.947/1.984/0.026 ms
[root@ip-10-0-1-155 ~]# clear
```

Now the internet is available and working fine with public network.

After that we have to check whether the internet is working or not in private network.

Now we need to connect private network and click on connect






```
[root@ip-10-0-1-155 ~]# vi pemsing.pem
```

Open the file and paste the private key credential then save and quite then give the permission using the command as “chmod 400 pemsing.pem”

```
[root@ip-10-0-1-155 ~]# vi pemsing.pem
```

```
[root@ip-10-0-1-155 ~]# chmod 400 pemsing.pem
```

```
[root@ip-10-0-1-155 ~]# ssh -i "pemsing.pem" ec2-user@10.0.2.153
The authenticity of host '10.0.2.153 (10.0.2.153)' can't be established.
```

Now the internet is working fine with private network.

```
[ec2-user@ip-10-0-2-153 ~]$ ping google.com
PING google.com (142.251.10.100) 56(84) bytes of data.
64 bytes from sd-in-f100.1e100.net (142.251.10.100): icmp_seq=1 ttl=50 time=3.59 ms
64 bytes from sd-in-f100.1e100.net (142.251.10.100): icmp_seq=2 ttl=50 time=3.26 ms
64 bytes from sd-in-f100.1e100.net (142.251.10.100): icmp_seq=3 ttl=50 time=3.90 ms
64 bytes from sd-in-f100.1e100.net (142.251.10.100): icmp_seq=4 ttl=50 time=3.26 ms
```