CASE STUDY -CREATING AN ARCHITECTURE USING TERRAFORM ON AWS

You work as a DevOps Engineer in leading Software Company. You have been asked to build an infrastructure safely and efficiently.

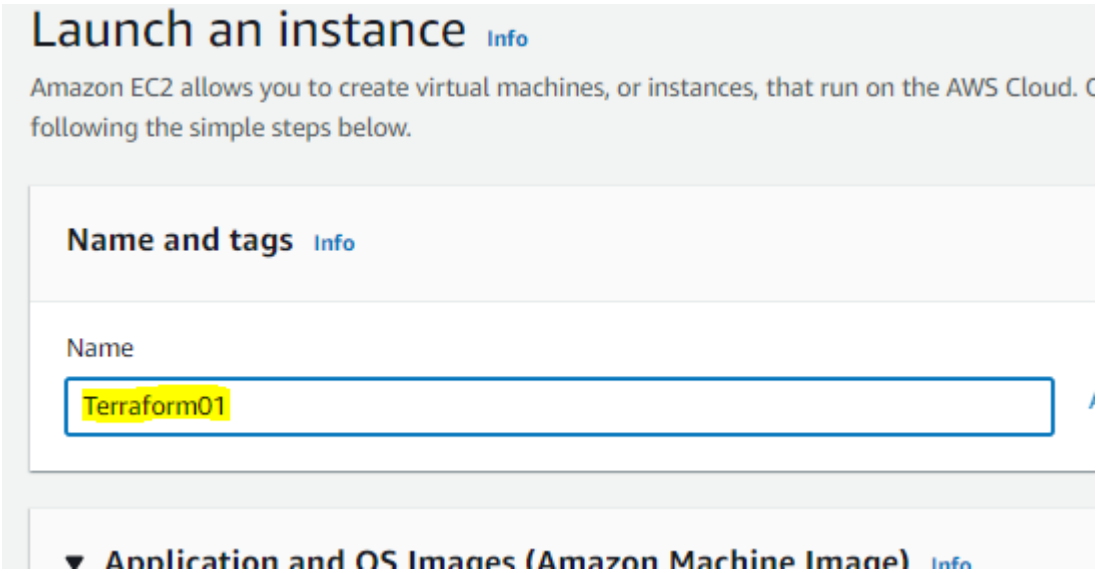
The company Requirements:

1. Use AWS cloud Provider and the software to be installed is Apache2
2. Use Ubuntu AMI

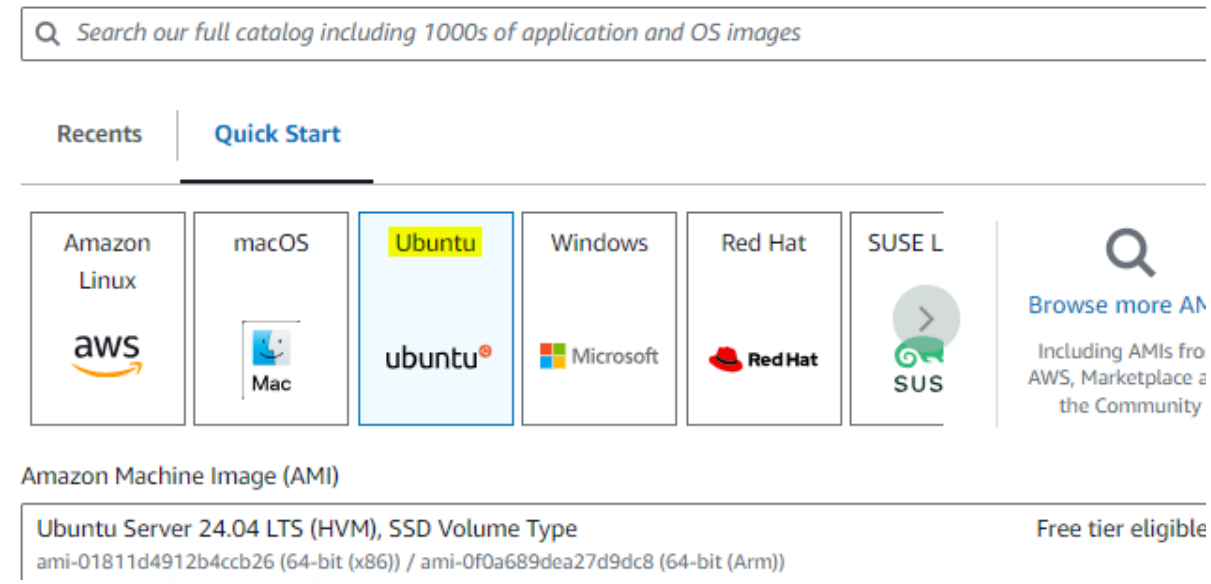
The company wants the Architecture to have the following services:

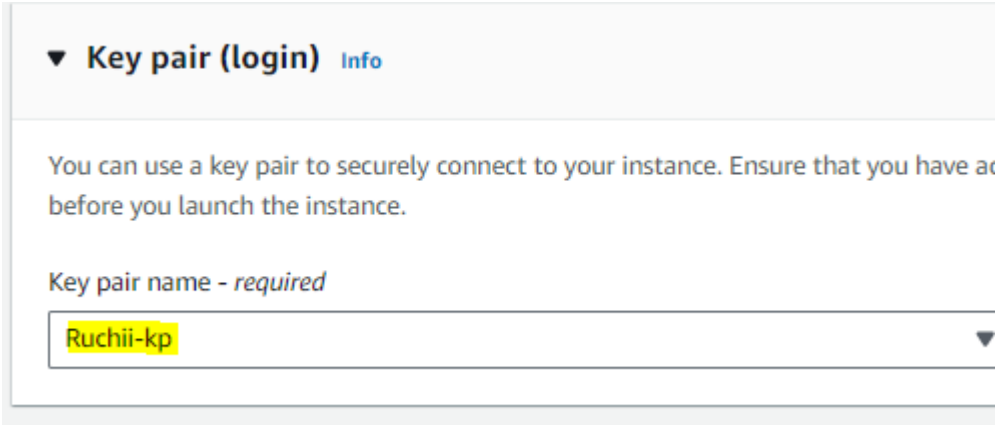
1. Create a template with a VPC, 2 subnets and 1 instance in each subnet
2. Attach Security groups, internet gateway and network interface to the instance

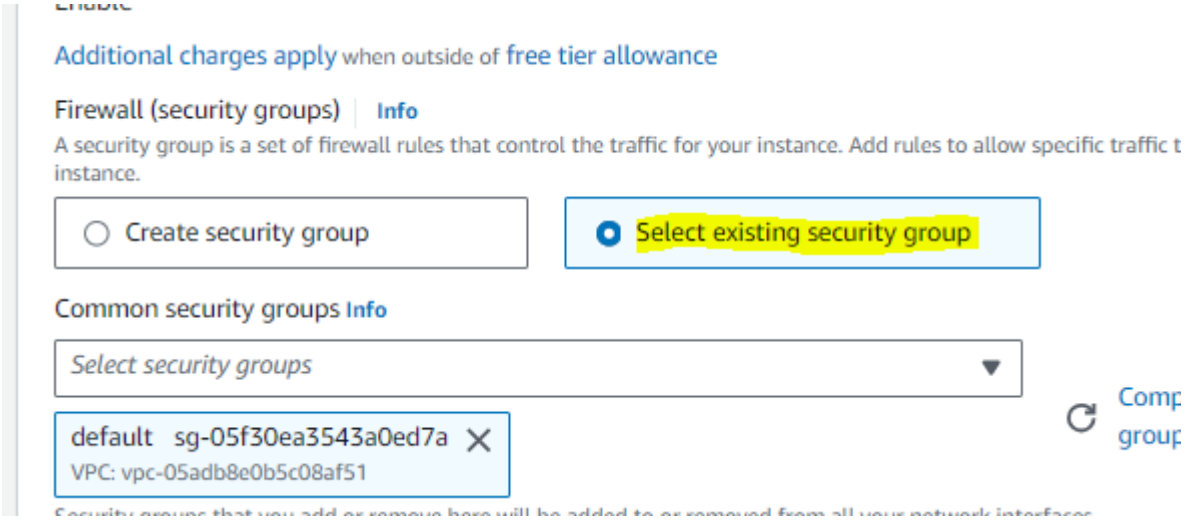
Launch an instance – terraform01



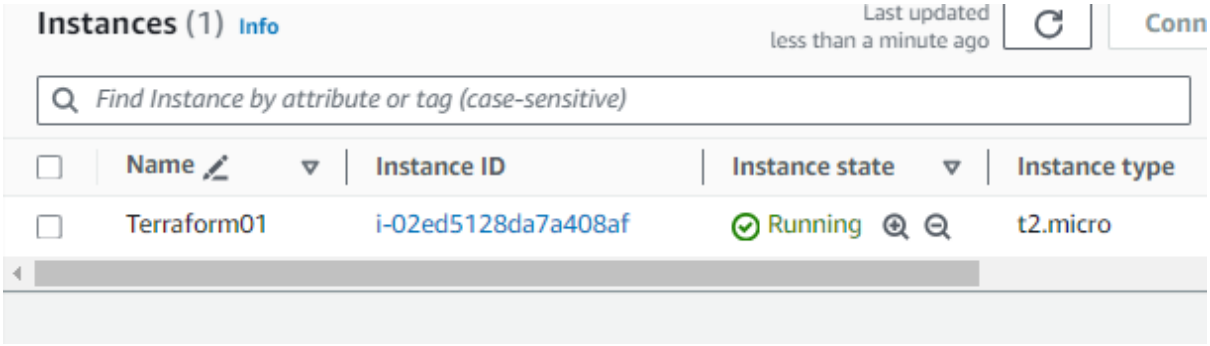
Select Ubuntu image



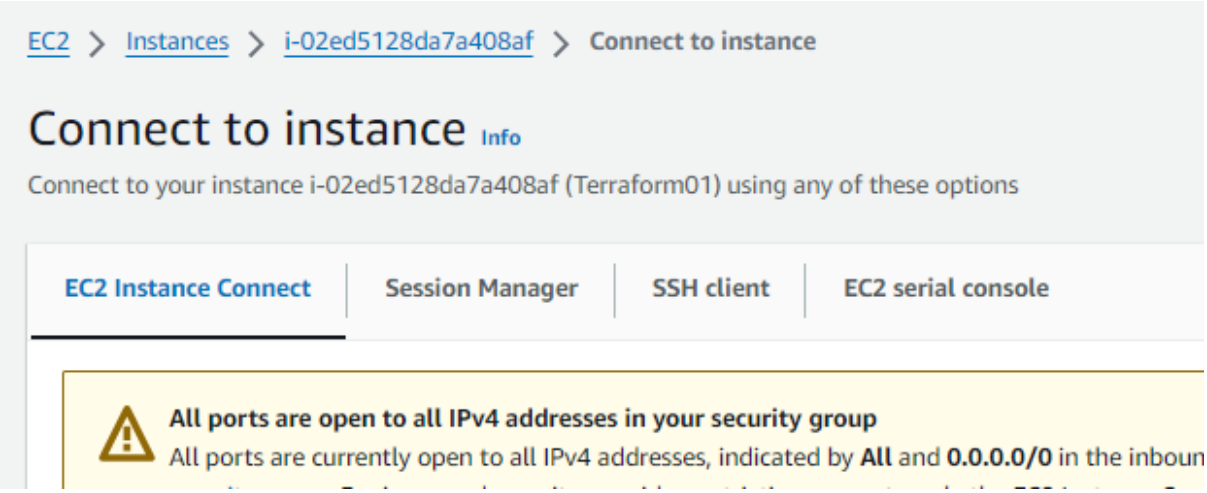




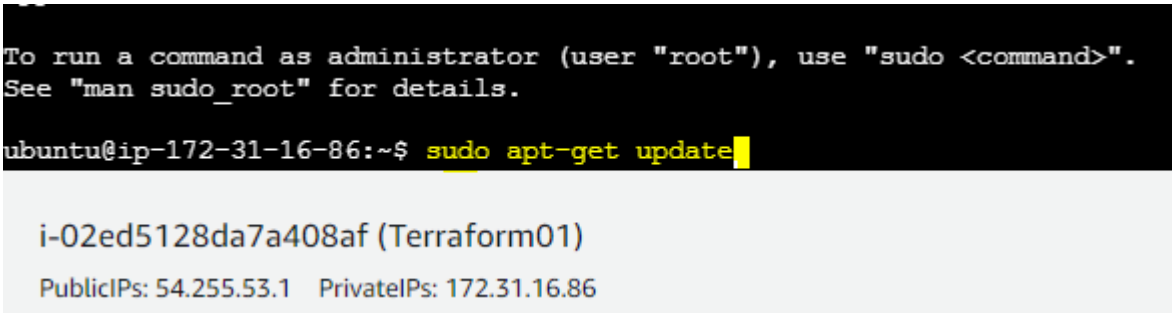
Successfully created instance



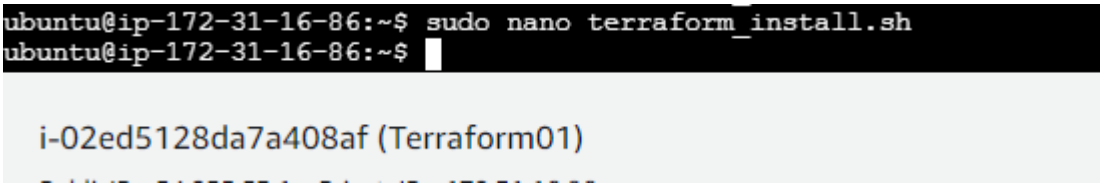
Connect to the instance



Update the machine



Create a shell script for installing terraform

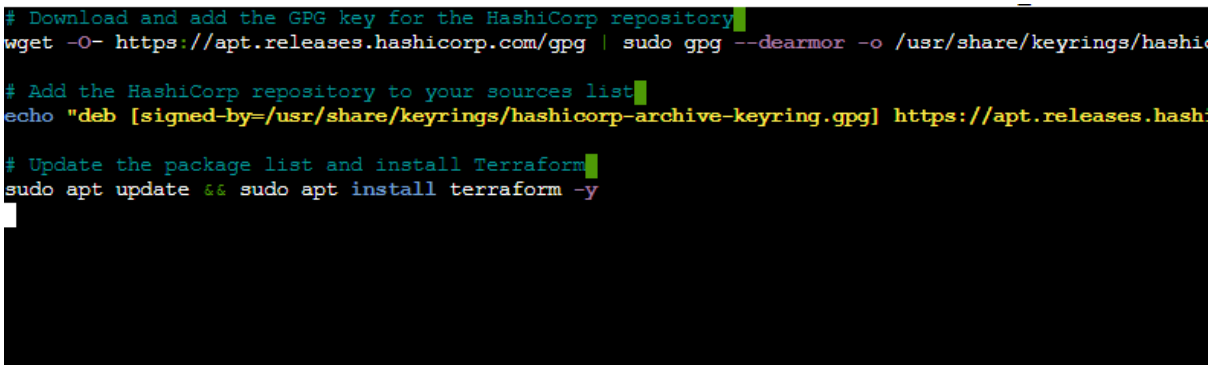


Copy the following commands for installing terraform

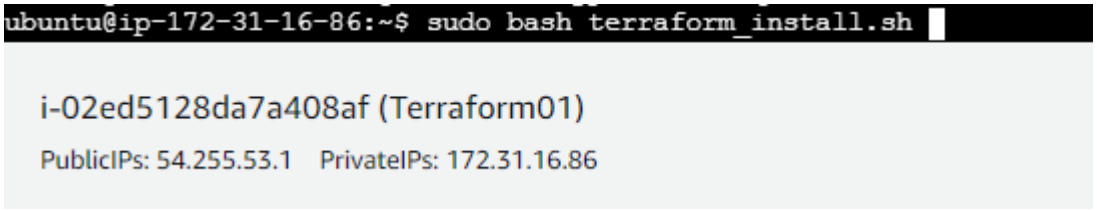
Download and add the GPG key for the HashiCorp repository
wget -O- https://apt.releases.hashicorp.com/gpg |
sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

Add the HashiCorp repository to your sources list
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

# Update the package list and install Terraform
sudo apt update && sudo apt install terraform –y



Execute the shell script with bash command



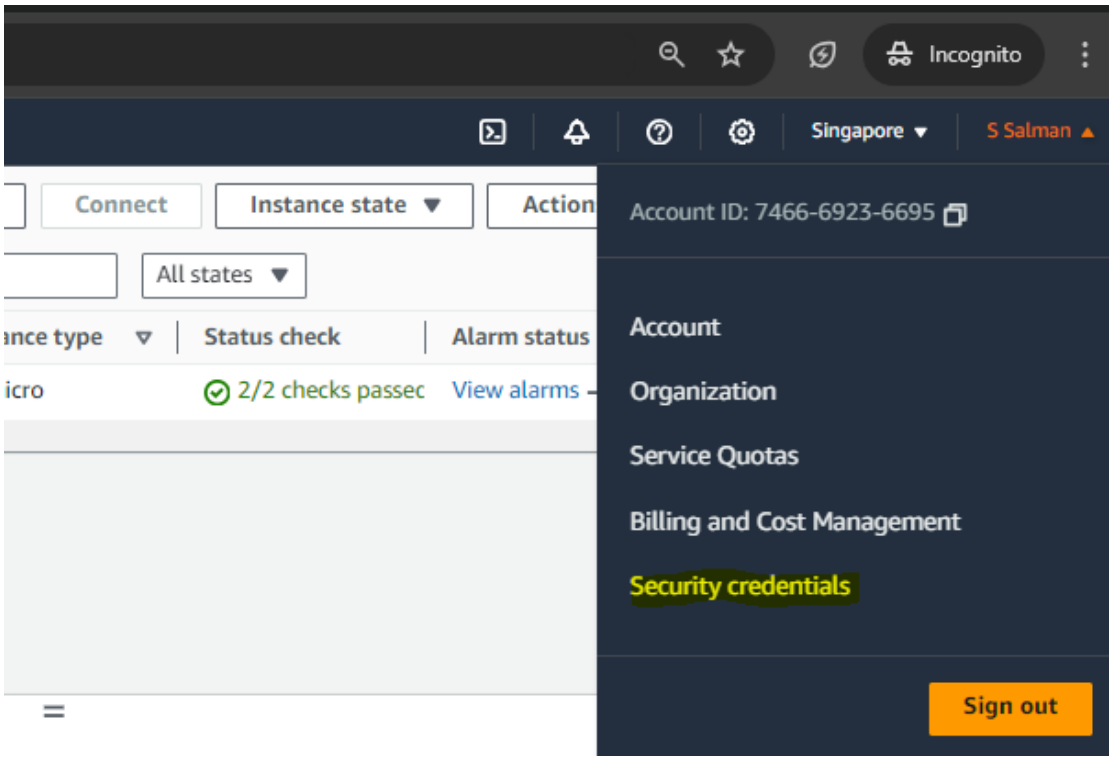
Terraform installed successfully.

```
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-16-86:~$
```

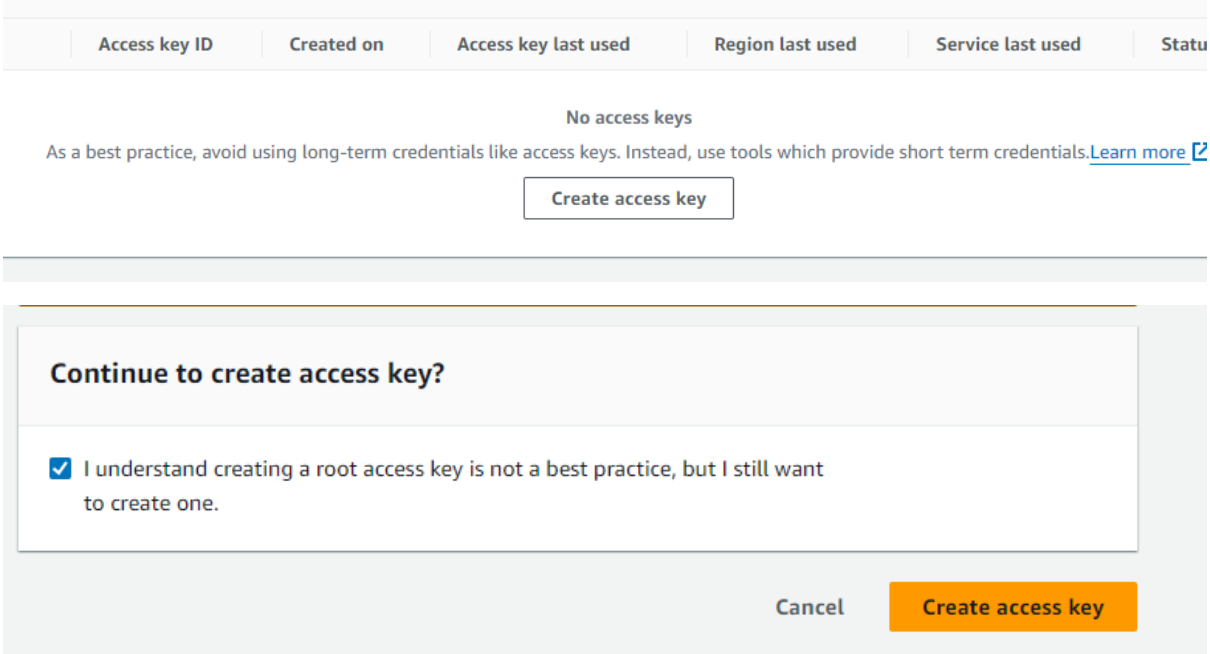
Check the version and confirm terraform installation

```
ubuntu@ip-172-31-16-86:~$ terraform --version
Terraform v1.9.5
on linux_amd64
ubuntu@ip-172-31-16-86:~$
```

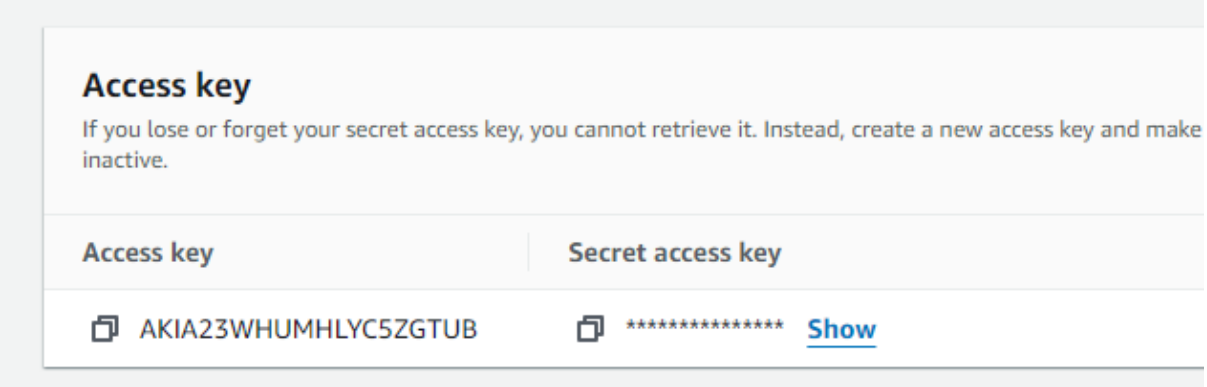
Now to give the aws provider in the terraform script we need to generate access key go to security credentials



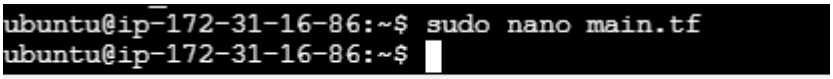
Click on create access key



Access key and secret access key is generated



Create main.tf file for writing script



TERRAFORM SCRIPT:

Give the aws provider

```
provider "aws" {
    region = "ap-southeast-1"
    access_key = "AKIA23WHUMHLYC5ZGTUB"
    secret_key = "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"
}
```

Create a VPC

```
# Create a VPC

resource "aws_vpc" "testvpc" {
  cidr_block = "10.0.0.0/16"

  tags = {
     Name = "testvpc"
  }
}
```

Create a public subnet

```
# Create a Public Subnet

resource "aws_subnet" "testsbnt1" {
   vpc_id = aws_vpc.testvpc.id
   cidr_block = "10.0.1.0/24"
     map_public_ip_on_launch = "true"
     availability_zone = "ap-southeast-1a"

   tags = {
       Name = "testsbnt1"
    }
}
```

Create a private subnet

```
# Create a Private Subnet

resource "aws_subnet" "testsbnt2" {
    vpc_id = aws_vpc.testvpc.id
    cidr_block = "10.0.2.0/24"
      map_public_ip_on_launch = "false"
      availability_zone = "ap-southeast-1b"

     tags = {
       Name = "testsbnt2"
     }
}
```

Create an Internet gateway

```
# Create an Internet Gateway

resource "aws_internet_gateway" "testigw" {
    vpc_id = aws_vpc.testvpc.id
    tags = {
      Name = "testigw"
    }
}
```

Create a route table for public subnet

```
# Create a Route Table for Public Subnet

resource "aws_route_table" "testrtb1" {
      vpc_id = aws_vpc.testvpc.id

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

      route {
        ipv6_cidr_block = "::/0"
        gateway_id = aws_internet_gateway.testigw.id
      }
}
```

Associate public route table with public subnet

```
# Associate Public Route Table with Public Subnet

resource "aws_route_table_association" "testassoc1" {
        subnet_id = aws_subnet.testsbnt1.id
        route_table_id = aws_route_table.testrtb1.id
}
```

Create private RT for subnet 2

```
# Create a Private Route Table for Subnet 2

resource "aws_route_table" "testrtb2" {
     vpc_id = aws_vpc.testvpc.id

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

     tags = {
       Name = "testrtb2"
     }
}
```

Associate RT with Private subnet

```
# Associate Route Table with Private Subnet

resource "aws_route_table_association" "testassoc2" {
      subnet_id = aws_subnet.testsbnt2.id
      route_table_id = aws_route_table.testrtb2.id
}
```

Assign ENI with IP

```
# Assign ENI with IP

resource "aws_network_interface" "testeni1" {
           subnet_id = aws_subnet.testsbnt1.id
           private_ips = ["10.0.1.10"]
           security_groups = [aws_security_group.testsg.id]
}

resource "aws_network_interface" "testeni2" {
           subnet_id = aws_subnet.testsbnt2.id
           private_ips = ["10.0.2.10"]
           security_groups = [aws_security_group.testsg.id]
}
```

Assign Elastic IP to ENI

```
# Assign Elastic IP to ENI

resource "aws_eip" "testeip1" {
        domain = "vpc"
        network_interface = aws_network_interface.testeni1.id
        associate_with_private_ip = "10.0.1.10"
        depends_on= [aws_internet_gateway.testigw, aws_instance.Instance1]
        tags = {
           Name = "testeip1"
        }
}
```

Create an Elastic IP Address for NAT Gateway

```
# Create an Elastic IP Address for NAT Gateway

resource "aws_eip" "testeip2" {
        domain = "vpc"
        associate_with_private_ip = "10.0.2.10"
        depends_on= [aws_internet_gateway.testigw]
        tags = {
           Name = "testeip2"
        }
}
```

Create a NAT Gateway for VPC

```
# Create a NAT Gateway for VPC

resource "aws_nat_gateway" "nat" {
     allocation_id = aws_eip.testeip2.id
     subnet_id = aws_subnet.testsbnt2.id

     tags = {
      Name = "nat"
     }
}
```

Create a security group

```
# Create a Security Group

resource "aws_security_group" "testsg" {
   description = "Allow limited inbound external traffic"
   vpc_id = aws_vpc.testvpc.id
   name = "testsg"

   ingress {
     protocol = "tcp"
     cidr_blocks = ["0.0.0.0/0"]
     from_port = 22
     to_port = 22
   }

   ingress {
     protocol = "tcp"
     cidr_blocks = ["0.0.0.0/0"]
     from_port = 80
     to_port = 80
   }

    ingress {
      protocol = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
      from_port = 443
      to_port = 443
     }
    egress {
      from_port = 0
      to_port = 0
      protocol = "-1"
      cidr_blocks = ["0.0.0.0/0"]

    }
    tags = {
      Name = "testsg"
    }
}
```

Create Linux server & Install/Enable Apache2 (Instance 1)

```
# Create Linux Server & Install/Enable Apache2 (Instance 1)

resource "aws_instance" "Instance1" {
    ami = "ami-0d07675d294f17973"
    instance_type = "t2.medium"
    availability_zone = "ap-southeast-1a"
    key_name = "Ruchii-kp"
    network_interface {
      device_index = 0
      network_interface_id = aws_network_interface.testeni1.id
   }

 user_data = <<-EOF
   #!/bin/bash
   sudo apt update -y
   sudo apt install apache2 -y
   sudo systemctl start apache2
   sudo systemctl enable apache2
   EOF

   tags = {
     Name = "Instance1"
    }
}
```

Create Linux server & Install/Enable Apache2 (Instance 2)

```
# Create Linux Server & Install/Enable Apache2 Here (Instance 2)

resource "aws_instance" "Instance2" {
    ami = "ami-01811d4912b4ccb26"
    instance_type = "t2.medium"
    availability_zone = "ap-southeast-1b"
    key_name = "Ruchii-kp"
    network_interface {
      device_index = 0
      network_interface_id = aws_network_interface.testeni2.id
   }

 user_data = <<-EOF
   #!/bin/bash
   sudo apt update -y
   sudo apt install apache2 -y
   sudo systemctl start apache2
   sudo systemctl enable apache2
   EOF

   tags = {
     Name = "Instance2"
    }
}
```

$ terraform init

```
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.65.0...
- Installed hashicorp/aws v5.65.0 (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.
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

PublicIPs: 54.255.53.1 PrivateIPs: 172.31.16.86

$ terraform plan

```
      + 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" = "testvpc"
        }
      + tags_all                              = {
          + "Name" = "testvpc"
        }
    }

Plan: 16 to add, 0 to change, 0 to destroy.


Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

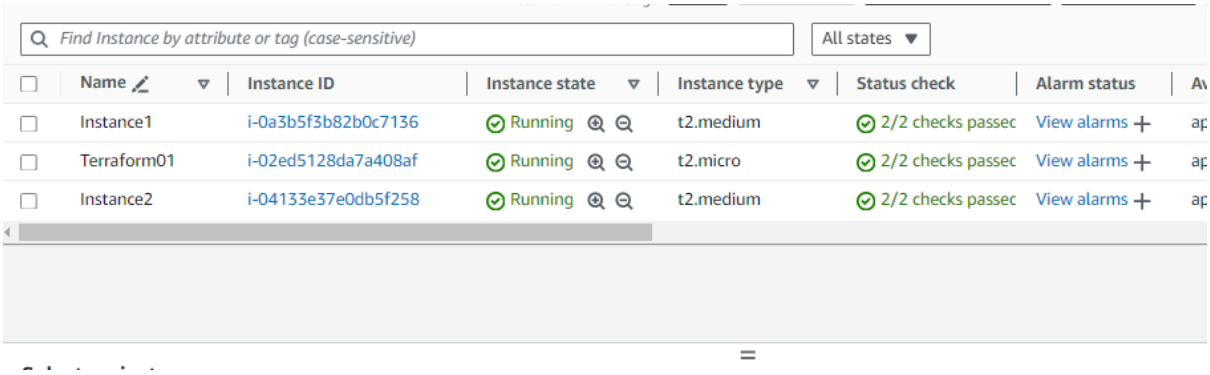
$ terraform apply

```
aws_network_interface.testeni2: Creating...
aws_nat_gateway.nat: Creating...
aws_network_interface.testeni2: Creation complete after 1s [id=eni-01ec06add02403d33]
aws_instance.Instance2: Creating...
aws_nat_gateway.nat: Still creating... [10s elapsed]
aws_instance.Instance2: Still creating... [10s elapsed]
aws_instance.Instance2: Creation complete after 12s [id=i-04133e37e0db5f258]
aws_nat_gateway.nat: Still creating... [20s elapsed]
aws_nat_gateway.nat: Still creating... [30s elapsed]
aws_nat_gateway.nat: Still creating... [40s elapsed]
aws_nat_gateway.nat: Still creating... [50s elapsed]
aws_nat_gateway.nat: Still creating... [1m0s elapsed]
aws_nat_gateway.nat: Still creating... [1m10s elapsed]
aws_nat_gateway.nat: Still creating... [1m20s elapsed]
aws_nat_gateway.nat: Still creating... [1m30s elapsed]
aws_nat_gateway.nat: Still creating... [1m40s elapsed]
aws_nat_gateway.nat: Still creating... [1m50s elapsed]
aws_nat_gateway.nat: Creation complete after 1m54s [id=nat-0a58e3f240a13cfb0]
aws_route_table.testrtb2: Modifying... [id=rtb-002e76786f330f5e0]
aws_route_table.testrtb2: Modifications complete after 1s [id=rtb-002e76786f330f5e0]
aws_route_table_association.testassoc2: Creating...
aws_route_table_association.testassoc2: Creation complete after 0s [id=rtbassoc-0b61de676f1b2e686]

Apply complete! Resources: 5 added, 1 changed, 4 destroyed.
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

PublicIPs: 54.255.53.1 PrivateIPs: 172.31.16.86

Instance 1 and instance 2 are created newly by the script

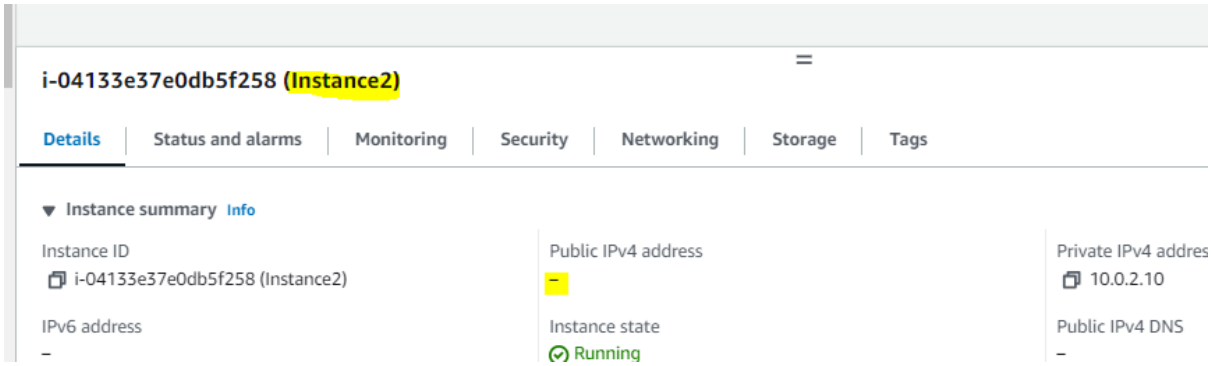


Public IP, private IP and elastic IP are addressed in Instance 1

i-0a3b5f3b82b0c7136 (Instance1)

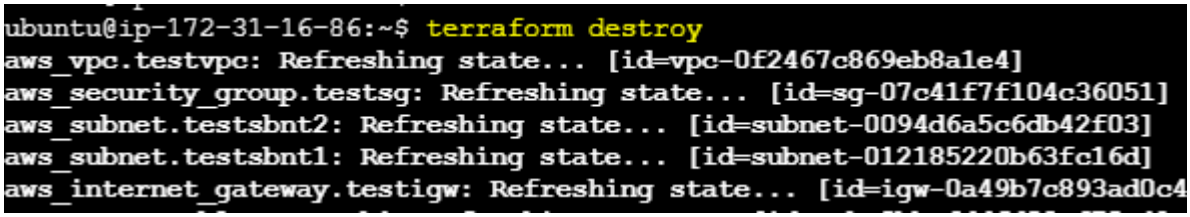
Instance ID i-0a3b5f3b82b0c7136 (Instance1)	Public IPv4 address 175.41.138.206 | open address	Private IPv4 addresses 10.0.1.10
IPv6 address –	Instance state Running	Public IPv4 DNS –
Hostname type IP name: ip-10-0-1-10.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only) ip-10-0-1-10.ap-southeast-1.compute.internal	
Answer private resource DNS name –	Instance type t2.medium	Elastic IP addresses 175.41.138.206 (testeip1) [Public IP]

You can find public IP in instance2, as we have not written script for it



Browse the Public IP of Instance 1 to visit the web page of apache2

At all destroy your terraform script



Successfully destroyed

```
aws_internet_gateway.testigw: Destroying... [id=igw-0a49b7c893ad0c4a9]
aws_internet_gateway.testigw: Destruction complete after 1s
aws_network_interface.testeni1: Destruction complete after 1s
aws_security_group.testsg: Destroying... [id=sg-07c41f7f104c36051]
aws_subnet.testsbnt1: Destroying... [id=subnet-012185220b63fc16d]
aws_subnet.testsbnt1: Destruction complete after 0s
aws_security_group.testsg: Destruction complete after 0s
aws_vpc.testvpc: Destroying... [id=vpc-0f2467c869eb8a1e4]
aws_vpc.testvpc: Destruction complete after 1s

Destroy complete! Resources: 16 destroyed.
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

PublicIPs: 54.255.53.1 PrivateIPs: 172.31.16.86

you can see instances are terminated

Find Instance by attribute or tag (case-sensitive) | All states

Name	Instance ID	Instance state	Instance type	Status check
Terraform01	i-02ed5128da7a408af	Running	t2.micro	2/2 checks
Instance2	i-04133e37e0db5f258	Terminated	t2.medium	–
Instance1	i-0a3b5f3b82b0c7136	Terminated	t2.medium	–