Module-12: Provisioning Infrastructure using Terraform Part-II

Demo Document - 2

DEMO-2: Terraform Project

In this demonstration we will Set up the entire infrastructure using a Terraform Configuration. Following resources need to be deployed:

1. Network Setup
    a. Create a VPC
    b. Create an internet gateway
    c. Create a custom Route Table
    d. Create a Subnet
    e. Associate the Subnet with the Route Table
2. Security Group Setup
    a. Create a new security group
    b. Enable ports 22, 80, 443
3. Network Interface Setup
    a. Create a new network interface with IP in the previously created subnet
    b. Create an elastic IP associated with the network interface
4. Ec2 instance setup
    a. Create a new ubuntu ec2 instance and attach the network interface to it
    b. Install httpd server on it

All Configuration code has been taken from:
https://registry.terraform.io/providers/hashicorp/aws/latest/docs

1. Create a new directory and a new terraform configuration to run in it

   Syntax: mkdir <newDir>
   terraform init
   vi filename.tf

   All the configuration code given below should be kept in a single file only.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      # optional
      version = "~> 3.0"
    }
  }
}

# Configuring provider
provider "aws" {
  region = "us-east-2"
  access_key = "my-access-key"
  secret_key = "my-secret-key"
}
```

Configuration for Network Setup

```
# Creating a VPC
resource "aws_vpc" "proj-vpc" {
  cidr_block = "10.0.0.0/16"
}

# Create an Internet Gateway
resource "aws_internet_gateway" "proj-ig" {
  vpc_id = aws_vpc.proj-vpc.id
  tags = {
    Name = "gateway1"
  }
}

# Setting up the route table
resource "aws_route_table" "proj-rt" {
  vpc_id = aws_vpc.proj-vpc.id

  route {
    # pointing to the internet
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.proj-ig.id
  }

  route {
    ipv6_cidr_block        = "::/0"
    gateway_id = aws_internet_gateway.proj-ig.id
  }

  tags = {
    Name = "rt1"
  }
}
```

```
# Setting up the subnet
resource "aws_subnet" "proj-subnet" {
  vpc_id     = aws_vpc.proj-vpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "us-east-2b"

  tags = {
    Name = "subnet1"
  }
}

# Associating the subnet with the route table
resource "aws_route_table_association" "proj-rt-sub-assoc" {
  subnet_id      = aws_subnet.proj-subnet.id
  route_table_id = aws_route_table.proj-rt.id
}
```

Security Group Configuration

```
# Creating a Security Group
resource "aws_security_group" "proj-sg" {
  name        = "proj-sg"
  description = "Enable web traffic for the project"
  vpc_id      = aws_vpc.proj-vpc.id

  ingress {
    description      = "HTTPS traffic"
    from_port        = 443
    to_port          = 443
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }
  ingress {
    description      = "HTTP traffic"
    from_port        = 80
    to_port          = 80
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  ingress {
    description      = "SSH port"
    from_port        = 22
    to_port          = 22
    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"]
    ipv6_cidr_blocks = ["::/0"]
  }

  tags = {
    Name = "proj-sg1"
  }
}
```

Network Interface setup

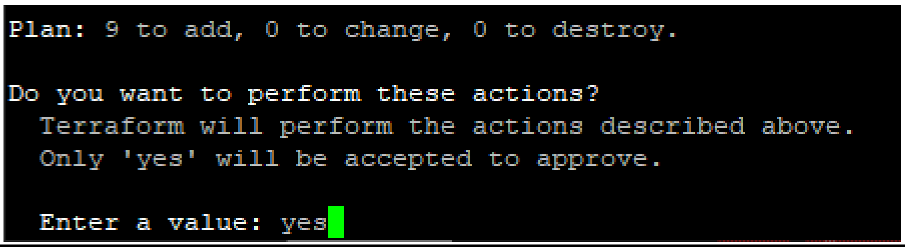
```
# Creating a new network interface
resource "aws_network_interface" "proj-ni" {
  subnet_id       = aws_subnet.proj-subnet.id
  private_ips     = ["10.0.1.10"]
  security_groups = [aws_security_group.proj-sg.id]
}

# Attaching an elastic IP to the network interface
resource "aws_eip" "proj-eip" {
  vpc                       = true
  network_interface         = aws_network_interface.proj-ni.id
  associate_with_private_ip = "10.0.1.10"
}
```

Creating a new EC2 instance

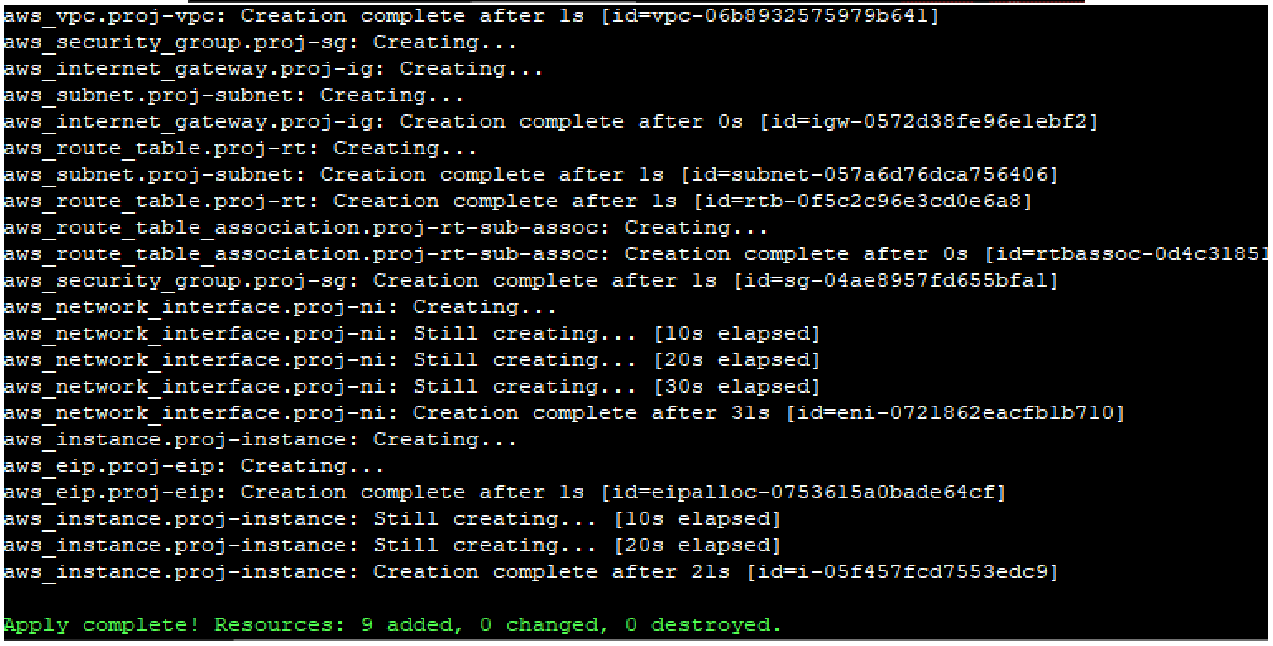
```
# Creating an Ubuntu EC2 instance
resource "aws_instance" "proj-instance" {
  ami           = "ami-00399ec92321828f5"
  instance_type = "t2.micro"
  availability_zone = "us-east-2b"
  key_name = "<your-aws-key>"

  network_interface {
    device_index = 0
    network_interface_id = aws_network_interface.proj-ni.id
  }

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

  tags = {
    Name = "project-instance"
  }
}
```

1. Execute the apply command and provision the infrastructure

   Syntax: terraform apply

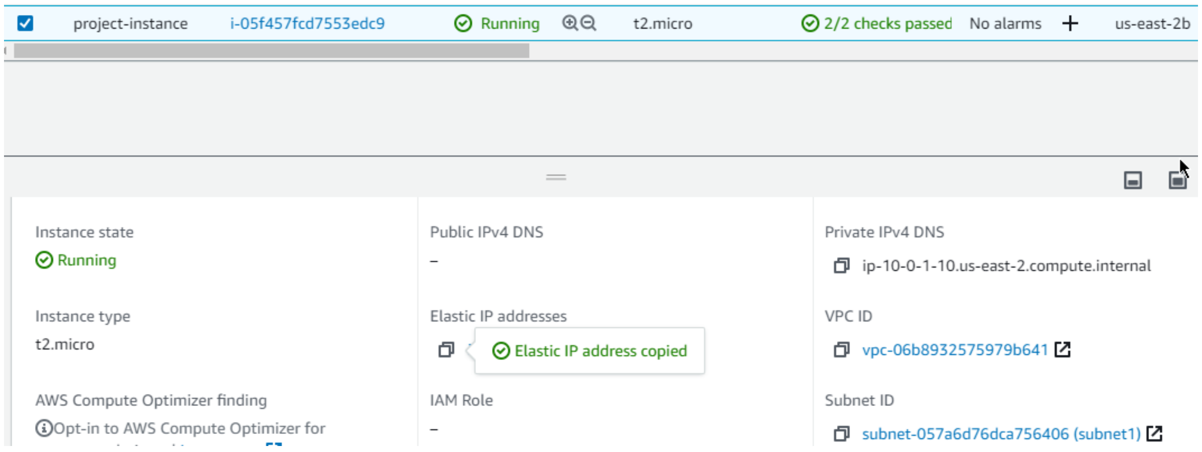


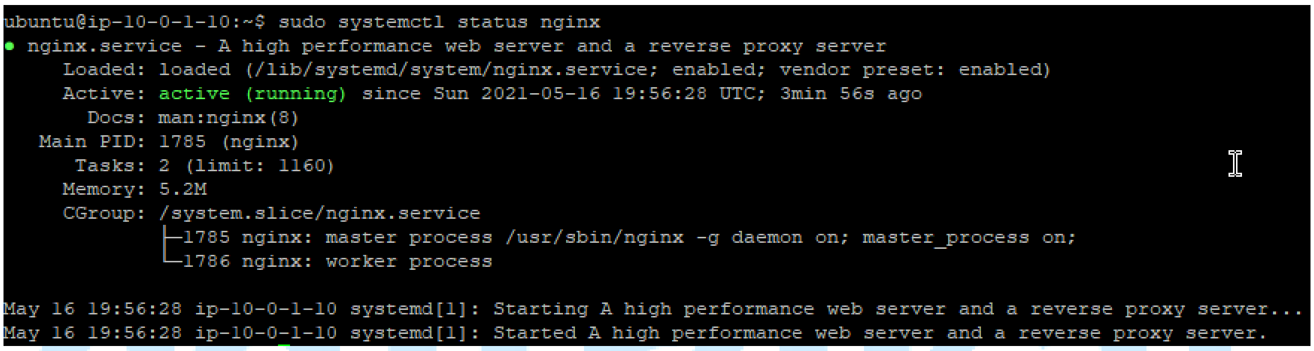
```
Plan: 9 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```



```
aws_vpc.proj-vpc: Creation complete after 1s [id=vpc-06b8932575979b641]
aws_security_group.proj-sg: Creating...
aws_internet_gateway.proj-ig: Creating...
aws_subnet.proj-subnet: Creating...
aws_internet_gateway.proj-ig: Creation complete after 0s [id=igw-0572d38fe96e1ebf2]
aws_route_table.proj-rt: Creating...
aws_subnet.proj-subnet: Creation complete after 1s [id=subnet-057a6d76dca756406]
aws_route_table.proj-rt: Creation complete after 1s [id=rtb-0f5c2c96e3cd0e6a8]
aws_route_table_association.proj-rt-sub-assoc: Creating...
aws_route_table_association.proj-rt-sub-assoc: Creation complete after 0s [id=rtbassoc-0d4c3185
aws_security_group.proj-sg: Creation complete after 1s [id=sg-04ae8957fd655bfa1]
aws_network_interface.proj-ni: Creating...
aws_network_interface.proj-ni: Still creating... [10s elapsed]
aws_network_interface.proj-ni: Still creating... [20s elapsed]
aws_network_interface.proj-ni: Still creating... [30s elapsed]
aws_network_interface.proj-ni: Creation complete after 31s [id=eni-0721862eacfb1b710]
aws_instance.proj-instance: Creating...
aws_eip.proj-eip: Creating...
aws_eip.proj-eip: Creation complete after 1s [id=eipalloc-0753615a0bade64cf]
aws_instance.proj-instance: Still creating... [10s elapsed]
aws_instance.proj-instance: Still creating... [20s elapsed]
aws_instance.proj-instance: Creation complete after 21s [id=i-05f457fcd7553edc9]

Apply complete! Resources: 9 added, 0 changed, 0 destroyed.
```

2. Now we can verify using aws that everything has been deployed like we wanted





```
ubuntu@ip-10-0-1-10:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2021-05-16 19:56:28 UTC; 3min 56s ago
       Docs: man:nginx(8)
   Main PID: 1785 (nginx)
      Tasks: 2 (limit: 1160)
     Memory: 5.2M
     CGroup: /system.slice/nginx.service
             ├─1785 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─1786 nginx: worker process

May 16 19:56:28 ip-10-0-1-10 systemd[1]: Starting A high performance web server and a reverse proxy server...
May 16 19:56:28 ip-10-0-1-10 systemd[1]: Started A high performance web server and a reverse proxy server.
```