Terraform intern Induction

Write Terraform script to create a custom VPC and deploy two EC2 VMs on AWS using Terraform.

- **The code should be broken into three different parts:**
- **Networking (define the VPC and all of its components)**
- **SSH-Key (dynamically create an SSH-key pair for connecting to VMs)**
- **EC2 (deploy a VM in the public subnet, and deploy another VM in a private subnet)**
- **NGINX should be accessed for all the internet**
- **Automate Terraform Deployment with Jenkins Pipelines**

Working directory

```
terraform_project/
├── networking.tf
├── ssh_key.tf
├── ec2.tf
├── Jenkinsfile
```

networking.tf

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "custom_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support   = true
  enable_dns_hostnames = true
}

resource "aws_subnet" "public_subnet" {
  vpc_id                  = aws_vpc.custom_vpc.id
  cidr_block              = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1a"
}
```

```
resource "aws_subnet" "private_subnet" {
  vpc_id            = aws_vpc.custom_vpc.id
  cidr_block        = "10.0.2.0/24"
  availability_zone = "us-east-1a"
}

resource "aws_internet_gateway" "internet_gateway" {
  vpc_id = aws_vpc.custom_vpc.id
}

resource "aws_route_table" "public_route_table" {
  vpc_id = aws_vpc.custom_vpc.id

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

resource "aws_route_table_association" "public_subnet_association" {
  subnet_id      = aws_subnet.public_subnet.id
  route_table_id = aws_route_table.public_route_table.id
}

resource "aws_security_group" "default" {
  name        = "default-sg"
  vpc_id      = aws_vpc.custom_vpc.id

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

  ingress {
    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"]
  }
}
```

ec2.tf

```
resource "aws_instance" "public_vm" {
  ami           = "ami-0c02fb55956c7d316" # Amazon Linux 2 AMI (replace with your region's AMI)
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_subnet.id
  key_name      = aws_key_pair.generated_key_pair.key_name
  security_groups = [
    aws_security_group.default.name,
  ]

  tags = {
    Name = "Public-VM"
  }

  user_data = <<-EOF
              #!/bin/bash
              yum update -y
              amazon-linux-extras enable nginx1
              yum install nginx -y
              systemctl start nginx
  EOF
}

resource "aws_instance" "private_vm" {
  ami           = "ami-0c02fb55956c7d316" # Amazon Linux 2 AMI (replace with your region's AMI)
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.private_subnet.id
  key_name      = aws_key_pair.generated_key_pair.key_name
  security_groups = [
    aws_security_group.default.name,
  ]

  tags = {
    Name = "Private-VM"
  }
}
```

ssh_key.tf

```
resource "tls_private_key" "ssh_key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "aws_key_pair" "generated_key_pair" {
  key_name   = "dynamic-key"
  public_key = tls_private_key.ssh_key.public_key_openssh
}

output "private_key" {
  value     = tls_private_key.ssh_key.private_key_pem
  sensitive = true
}
```

Jenkinsfile

```
pipeline {
    agent any

    environment {
        AWS_ACCESS_KEY_ID = credentials('aws-credentials')
        AWS_SECRET_ACCESS_KEY = credentials('aws-credentials')
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/shashankhl-sigmoid/Terraform_assignment.git'
            }
        }

        stage('Terraform Init') {
            steps {
                sh 'terraform init'
            }
        }

        stage('Terraform Plan') {
            steps {
```

```
            sh 'terraform plan -out=tfplan'
        }
    }

    stage('Terraform Apply') {
        steps {
            sh 'terraform apply -auto-approve tfplan'
        }
    }

    stage('Terraform Destroy') {
        steps {
            sh 'terraform destroy -auto-approve'
        }
    }
  }
}
```

Initialize Terraform:

```
cd terraform_project
terraform init
```

Validate and Plan:

```
terraform validate
```

Generate an execution plan:

```
terraform plan -out=tfplan
```

Deploy the Infrastructure:

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

Access the Public VM

After deployment, get the **Public IP Address** of the public VM:

Run:

```
terraform output
```

Use the output to SSH into the public VM:

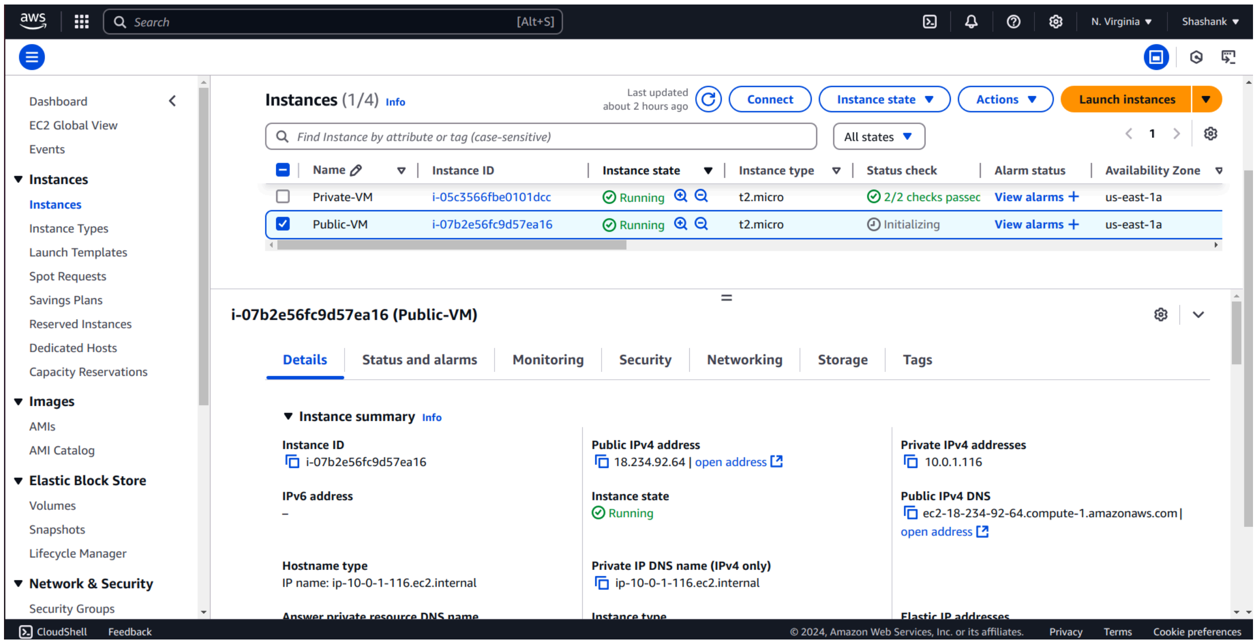
```
ssh -i <path-to-private-key.pem> ec2-user@<public-ip>
```

Replace <path-to-private-key.pem> with the path to your SSH private key.

Access NGINX in your browser using the public VM's IP:
http://<public-ip>

Clean Up Resources

terraform destroy -auto-approve



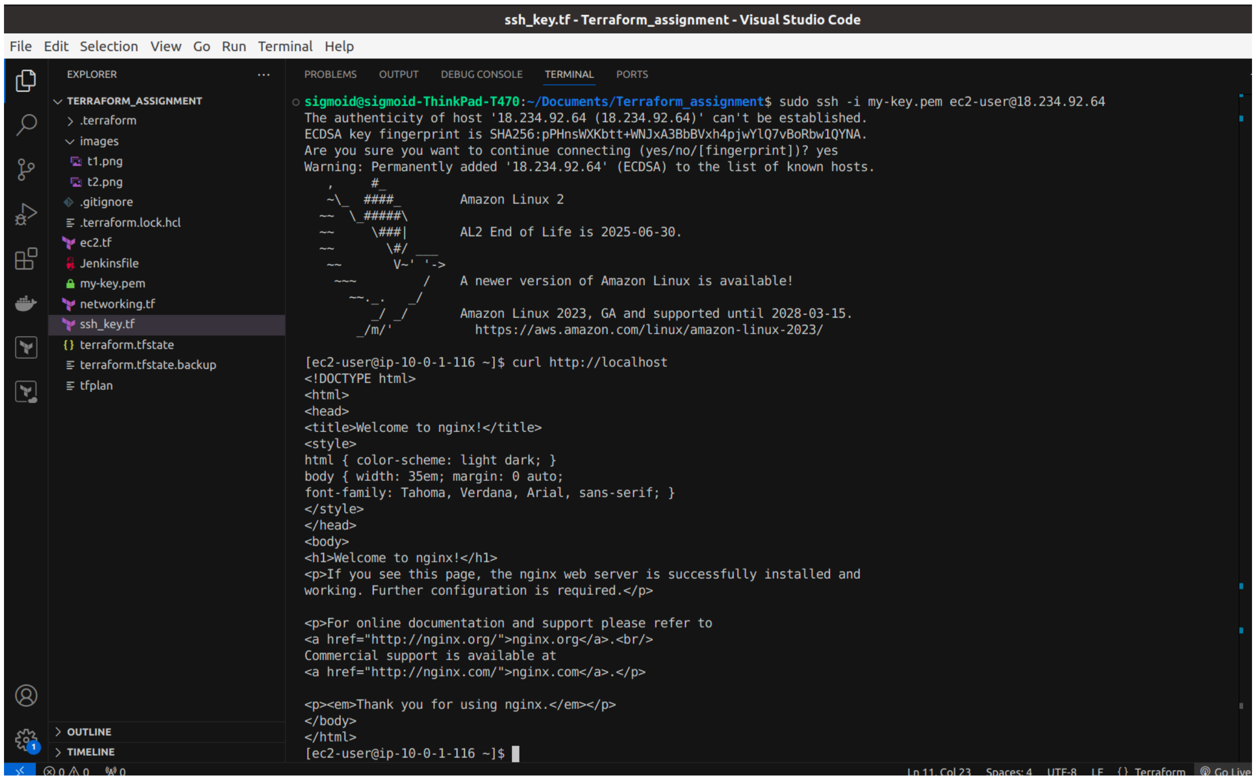
Instances (1/4) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☐	Private-VM	i-05c3566fbe0101dcc	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a
☑	Public-VM	i-07b2e56fc9d57ea16	Running	t2.micro	Initializing	View alarms +	us-east-1a

i-07b2e56fc9d57ea16 (Public-VM)

Instance summary

Instance ID: i-07b2e56fc9d57ea16
Public IPv4 address: 18.234.92.64 | open address
Private IPv4 addresses: 10.0.1.116
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-18-234-92-64.compute-1.amazonaws.com | open address
Hostname type: IP name: ip-10-0-1-116.ec2.internal
Private IP DNS name (IPv4 only): ip-10-0-1-116.ec2.internal

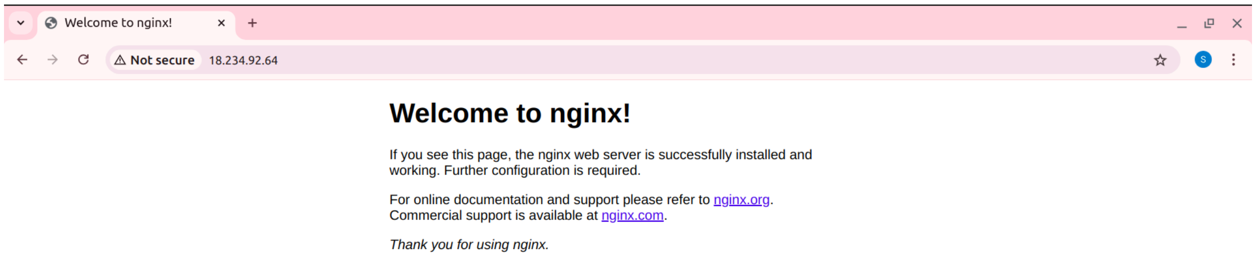


```
sigmoid@sigmoid-ThinkPad-T470:~/Documents/Terraform_assignment$ sudo ssh -i my-key.pem ec2-user@18.234.92.64
The authenticity of host '18.234.92.64 (18.234.92.64)' can't be established.
ECDSA key fingerprint is SHA256:pPHnsWXKbtt+WNJxA3BbBVxh4pjwYlQ7vBoRbw1QYNA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '18.234.92.64' (ECDSA) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-10-0-1-116 ~]$ curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[ec2-user@ip-10-0-1-116 ~]$
```

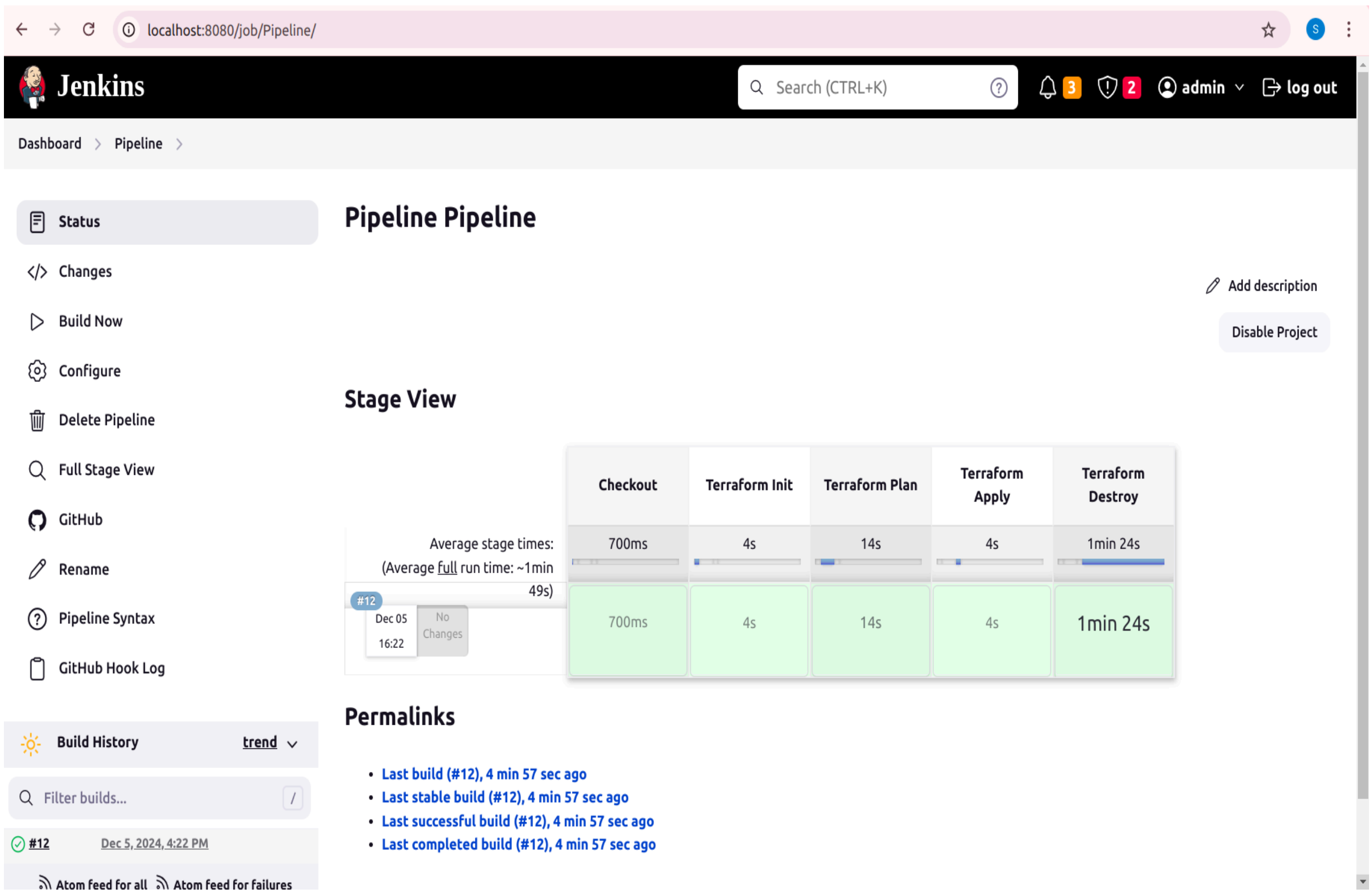


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



Pipeline Pipeline

Stage View

	Checkout	Terraform Init	Terraform Plan	Terraform Apply	Terraform Destroy
Average stage times: (Average full run time: ~1min 49s)	700ms	4s	14s	4s	1min 24s
#12 Dec 05 16:22 No Changes	700ms	4s	14s	4s	1min 24s

Permalinks

- Last build (#12), 4 min 57 sec ago
- Last stable build (#12), 4 min 57 sec ago
- Last successful build (#12), 4 min 57 sec ago
- Last completed build (#12), 4 min 57 sec ago