Assignment NO. 5 (CloudComputing and DevOps)

> Name: Ratnakar patil.
> PRN No: 22110373
> Roll No: 322050.

Write IaC using terraform to create EC2 machine on aws or azure or google cloud. (Compulsory to use Input and output variable files)

Theory:

> Que 1. What is Terraform

1. **Infrastructure as Code (IaC):** Terraform is a tool used for building, changing, and versioning infrastructure safely and efficiently. It allows you to describe your infrastructure using a declarative configuration language (HCL - HashiCorp Configuration Language), enabling you to manage infrastructure as code.
2. **Multi-Cloud Support:** One of Terraform's notable features is its ability to provision and manage infrastructure across various cloud providers like AWS, Azure, Google Cloud Platform, and others. This flexibility enables organizations to adopt a multi-cloud strategy or migrate between cloud providers seamlessly.
3. **State Management:** Terraform maintains a state file that keeps track of the resources it manages. This state file is crucial for Terraform to understand the current state of the infrastructure and to determine what changes need to be made to achieve the desired state.
4. **Dependency Resolution:** Terraform automatically handles dependencies between resources. It determines the order in which resources need to be provisioned or updated based on their dependencies, ensuring consistent and reliable infrastructure deployments.
5. **Community and Ecosystem:** Terraform boasts a vibrant community and a rich ecosystem of modules, plugins, and integrations. Users can leverage pre-built modules to easily provision complex infrastructure components, accelerating development and ensuring best practices are followed. Additionally, Terraform's extensibility allows for integration with other tools and automation workflows.

Task: Terraform script to create Infrastructure on any cloud platform (AWS or Azure or Google)

```
provider "aws" {
  region     = var.region
  access_key = "XXXX5O5NIPRZXXXXXXXXX"
  secret_key = "XXXXXXXClNDyY/LK8MDyPhHTcu6XXXXXX"
}

# Input variable for region
variable "region" {
  description = "AWS region"
}

# Output to display instance creation status
output "instance_status" {
  value = "Instance created successfully."
}

# 1. Create VPC

resource "aws_vpc" "prod-vpc" {
  cidr_block = "10.0.0.0/16"

  tags = {
    Name = "production"
  }
}

# 2. Create Internet Gateway

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.prod-vpc.id
}

# 3. Create Custom route table

resource "aws_route_table" "prod-route-table" {
  vpc_id = aws_vpc.prod-vpc.id

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

  route {
    ipv6_cidr_block = "::/0"
    gateway_id      = aws_internet_gateway.gw.id
  }
  tags = {
    Name = "Prod"
  }
}

# 4. Create a Subnet

resource "aws_subnet" "subnet-1" {
  vpc_id     = aws_vpc.prod-vpc.id
  cidr_block = "10.0.1.0/24"

  availability_zone = "us-east-1a"

  tags = {
    Name = "proud-subnet"
  }
}

# 5. Associate subnet with route table

resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.subnet-1.id
  route_table_id = aws_route_table.prod-route-table.id
}

# 6. Create a security group to allow port 22,80,443

resource "aws_security_group" "allow_web" {
  name        = "allow_web_traffic"
  description = "Allow web inbound traffic"
  vpc_id      = aws_vpc.prod-vpc.id

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

  tags = {
    Name = "allow_web"
  }
}

# 7. Create a network interface with an ip in the subnet that was created in step 4

resource "aws_network_interface" "web-server-nic" {
  subnet_id       = aws_subnet.subnet-1.id
  private_ips     = ["10.0.1.50"]
  security_groups = [aws_security_group.allow_web.id]
}

# 8. Assign an elastic IP to the netwrok interface created in step 7 (public ip)

resource "aws_eip" "eip-nic" {
  domain                    = "vpc"
  network_interface         = aws_network_interface.web-server-nic.id
  associate_with_private_ip = "10.0.1.50"
  depends_on                = [aws_internet_gateway.gw]
}

#  9. Create Ubuntu server

resource "aws_instance" "web-server-instance" {
  ami               = "ami-0f403e3180720dd7e"
  instance_type     = "t2.micro"
  availability_zone = "us-east-1a"
  key_name          = "main-key"

  network_interface {
    device_index         = 0
    network_interface_id = aws_network_interface.web-server-nic.id
  }

  user_data = <<-EOF
            #!/bin/bash
```

```
              sudo apt update -y
              sudo apt install apache2 -y
              sudo systemctl start apache2
              sudo bash -c "echo 'your very first web server' > /var/www/html/index.html"
            EOF


  tags = {
    Name = "web-server"
  }
}
```