Terraform Task: Deploy Two EC2 Instances in Different Regions with Nginx

Overview

This document provides a step-by-step guide to deploying two Amazon EC2 instances in different AWS regions using a single Terraform configuration. Each instance automatically installs and starts the Nginx web server. The configuration dynamically fetches the latest Amazon Linux 2 AMIs using AWS SSM Parameter Store to avoid errors related to outdated AMI IDs.

Prerequisites

- AWS CLI installed and configured (run `aws configure`)
- Terraform installed (https://developer.hashicorp.com/terraform/downloads)
- An AWS account with sufficient permissions to launch EC2 instances

Step-by-Step Procedure

Step 1: Create Working Directory

1. Open a terminal and create a project folder:
   mkdir ec2-multi-region-nginx
   cd ec2-multi-region-nginx
2. Create a file named `main.tf`:
   touch main.tf

Step 2: Add Terraform Configuration

Copy and paste the following content into `main.tf`:

```
provider "aws" {
  alias  = "use1"
  region = "us-east-1"
}
```

```
provider "aws" {
 alias  = "usw2"
 region = "us-west-2"
}

data "aws_ssm_parameter" "use1_ami" {
 provider = aws.use1
 name     = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
}

data "aws_ssm_parameter" "usw2_ami" {
 provider = aws.usw2
 name     = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
}

data "aws_vpc" "use1" {
 provider = aws.use1
 default  = true
}

data "aws_vpc" "usw2" {
 provider = aws.usw2
 default  = true
}

resource "aws_security_group" "use1_sg" {
 provider = aws.use1
 name     = "nginx-sg-use1"
 vpc_id   = data.aws_vpc.use1.id

 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
```

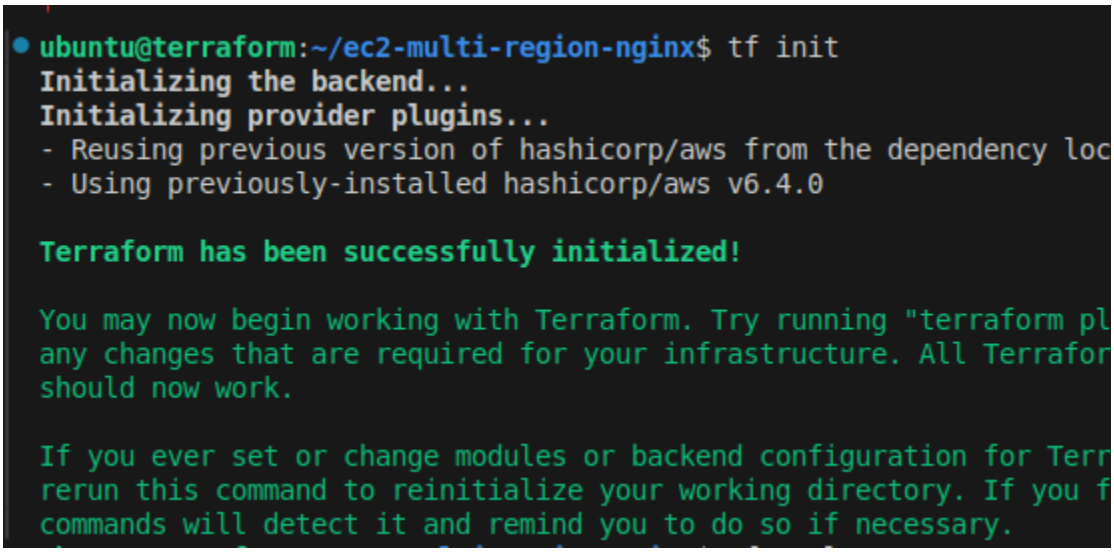
```
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

resource "aws_security_group" "usw2_sg" {
  provider = aws.usw2
  name     = "nginx-sg-usw2"
  vpc_id   = data.aws_vpc.usw2.id

  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"]
  }
}

resource "aws_instance" "use1_nginx" {
  provider       = aws.use1
```

```
  ami           = data.aws_ssm_parameter.use1_ami.value
  instance_type   = "t2.micro"
  security_groups = [aws_security_group.use1_sg.name]

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

  tags = {
    Name = "nginx-use1"
  }
}

resource "aws_instance" "usw2_nginx" {
  provider       = aws.usw2
  ami           = data.aws_ssm_parameter.usw2_ami.value
  instance_type   = "t2.micro"
  security_groups = [aws_security_group.usw2_sg.name]

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

  tags = {
    Name = "nginx-usw2"
  }
}

output "us_east_instance_ip" {
  value = aws_instance.use1_nginx.public_ip
}

output "us_west_instance_ip" {
  value = aws_instance.usw2_nginx.public_ip
```

}

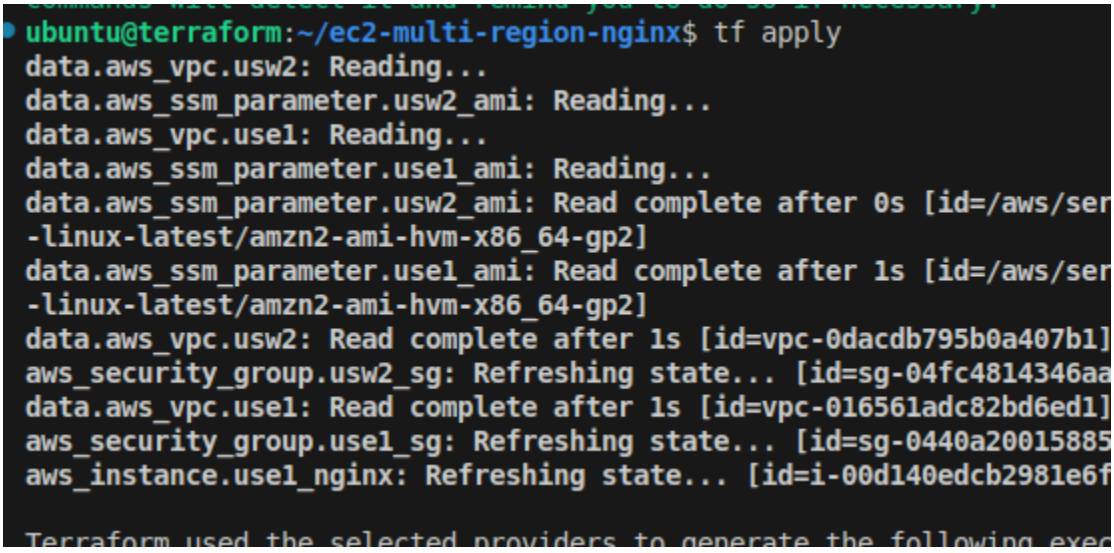
Step 3: Initialize Terraform

Run the following command to initialize Terraform in your project directory:
terraform init



Step 4: Apply the Terraform Configuration

Run the following to deploy the infrastructure:
terraform apply



Type `yes` when prompted.

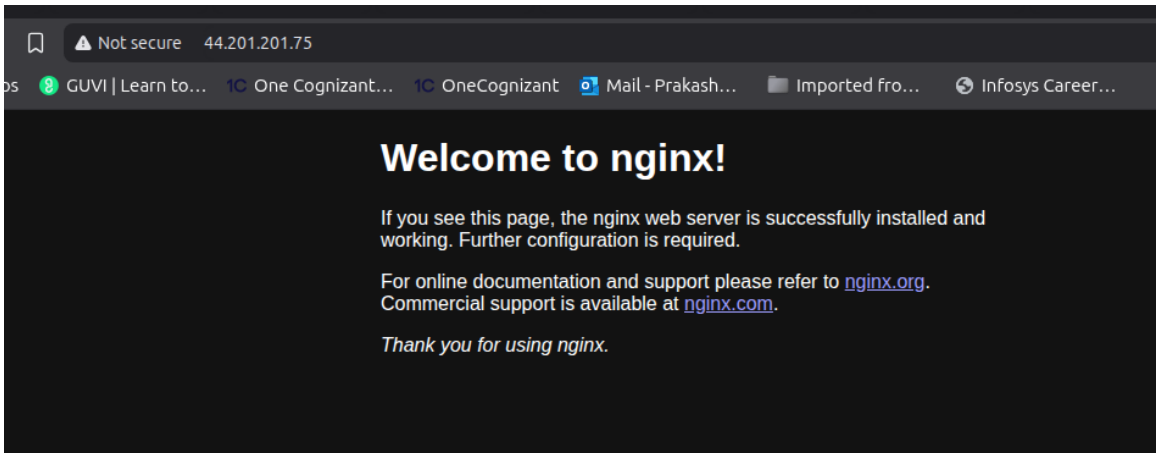
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.usw2_sg: Modifying... [id=sg-04fc4814346aa93d7]
aws_instance.use1_nginx: Destroying... [id=i-00d140edcb2981e6f]
aws_security_group.usw2_sg: Modifications complete after 1s [id=sg-0
7]
aws_instance.usw2_nginx: Creating...
aws_instance.use1_nginx: Still destroying... [id=i-00d140edcb2981e6f
d]
aws_instance.usw2_nginx: Still creating... [00m10s elapsed]
aws_instance.use1_nginx: Still destroying... [id=i-00d140edcb2981e6f
```

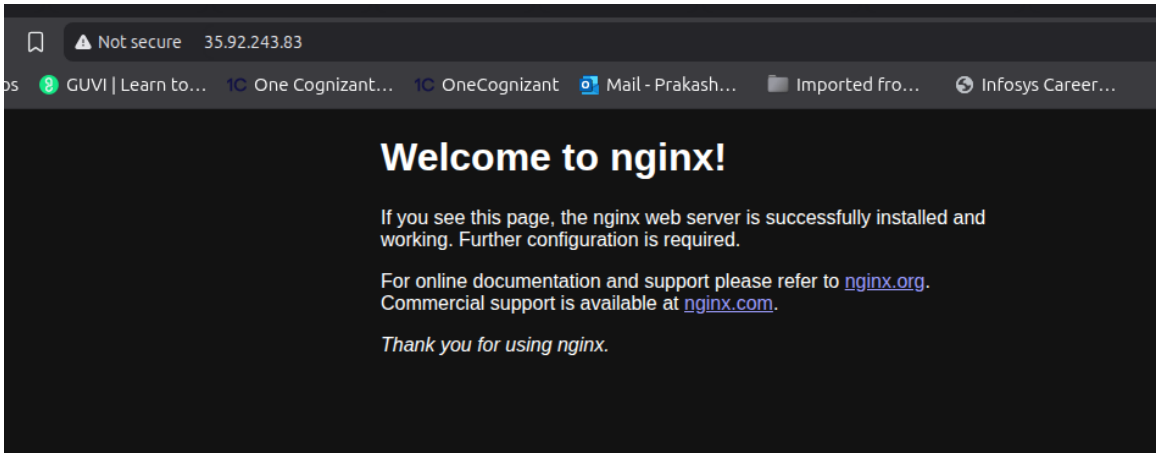
Step 5: Verify Nginx is Running

Terraform will output two public IP addresses. Open each one in a web browser:

```
  Apply complete! Resources: 2 added, 2 changed, 1 destroyed.

  Outputs:

  us_east_instance_ip = "44.201.201.75"
  us_west_instance_ip = "35.92.243.83"
ubuntu@terraform:~/ec2-multi-region-nginx$
                                                    Ln 131, Col 1   Sp
```

http://<us_east_instance_ip>
http://<us_west_instance_ip>
You should see the Nginx welcome page.





Step 6: Destroy the Resources (Optional)

To clean up and avoid charges, run:
terraform destroy

```
aws_instance.use1_nginx: Destruction complete after 31s
aws_security_group.use1_sg: Destroying... [id=sg-0440a20015885114d]
aws_instance.usw2_nginx: Destruction complete after 30s
aws_security_group.usw2_sg: Destroying... [id=sg-04fc4814346aa93d7]
aws_security_group.use1_sg: Destruction complete after 1s
aws_security_group.usw2_sg: Destruction complete after 1s

Destroy complete! Resources: 4 destroyed.
ubuntu@terraform:~/ec2-multi-region-nginx$
```