Terraform Task - 2

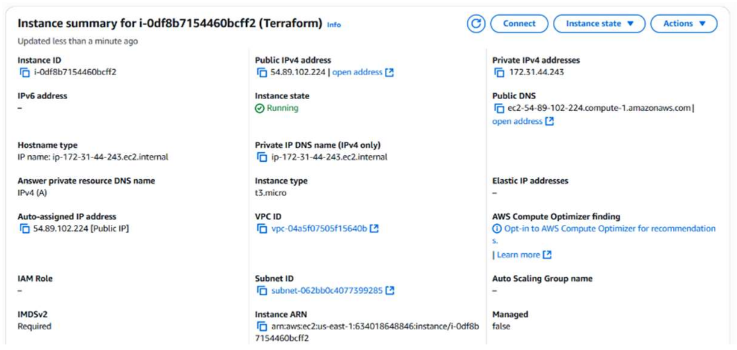
TASKS

Work Flow:

- Create an EC2 instance with the help of AWS Management Console with linux OS of required configuration.
- Now, Connect an EC2 instance with an help of Windows Terminal or Gitbash or Vbox.
- To connect an EC2 instance the command is:
  - ssh -i "**key_file**" ec2-user@"**Public_IP_address**"

Key_file --- Key file of the instance with the extension .pem

Public_IP_address --- Public IP address of the instance.



Instance summary for i-0df8b7154460bcff2 (Terraform)

Updated less than a minute ago

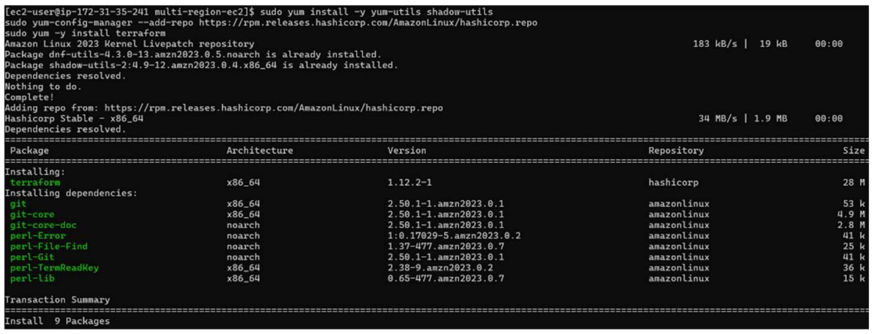
Instance ID i-0df8b7154460bcff2	Public IPv4 address 54.89.102.224 | open address	Private IPv4 addresses 172.31.44.243
IPv6 address –	Instance state Running	Public DNS ec2-54-89-102-224.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-44-243.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-44-243.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t3.micro	Elastic IP addresses –
Auto-assigned IP address 54.89.102.224 [Public IP]	VPC ID vpc-04a5f07505f15640b	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-062bb0c4077399285	Auto Scaling Group name –
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:634018648846:instance/i-0df8b7154460bcff2	Managed false

1. Create 2 EC2 instances on 2 different regions and install nginx using terraform script.

Step 1: Install Terraform in Linux EC2-Instance

Install An Terraform :

- ✓ To install an Terraform in linux machine go to an official website by using below link.
- ✓ Link : https://developer.hashicorp.com/terraform/install#linux
- ✓ Now you can see the commands given in the official page to install an Terraform, follow all the steps to install in your linux machine one by one.



```
[ec2-user@ip-172-31-35-241 multi-region-ec2]$ sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
Amazon Linux 2023 Kernel Livepatch repository                                     183 kB/s |  19 kB     00:00
Package dnf-utils-4.3.0-13.amzn2023.0.5.noarch is already installed.
Package shadow-utils-2:4.9-12.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Hashicorp Stable - x86_64                                                          34 MB/s | 1.9 MB     00:00
Dependencies resolved.
=======================================================================================================================
 Package                  Architecture         Version                             Repository                   Size
=======================================================================================================================
Installing:
 terraform                x86_64               1.12.2-1                            hashicorp                    28 M
Installing dependencies:
 git                      x86_64               2.50.1-1.amzn2023.0.1               amazonlinux                  53 k
 git-core                 x86_64               2.50.1-1.amzn2023.0.1               amazonlinux                 4.9 M
 git-core-doc             noarch               2.50.1-1.amzn2023.0.1               amazonlinux                 2.8 M
 perl-Error               noarch               1:0.17029-5.amzn2023.0.2            amazonlinux                  41 k
 perl-File-Find           noarch               1.37-477.amzn2023.0.7               amazonlinux                  25 k
 perl-Git                 noarch               2.50.1-1.amzn2023.0.1               amazonlinux                  41 k
 perl-TermReadKey         x86_64               2.38-9.amzn2023.0.2                 amazonlinux                  36 k
 perl-lib                 x86_64               0.65-477.amzn2023.0.7               amazonlinux                  15 k

Transaction Summary
=======================================================================================================================
Install  9 Packages
```

Step 2: Create a Working Directory.

- ✓ Create a directory to store your Terraform configuration files
- ✓ To create an directory in your linux, the command is:
    - **mkdir ec2-multiregion-nginx**
- ✓ Now to move into the created directory, the command is:
    - **cd ec2-multiregion-nginx**
- ✓ This folder will hold the Terraform configuration for deploying EC2 instances in two AWS regions.

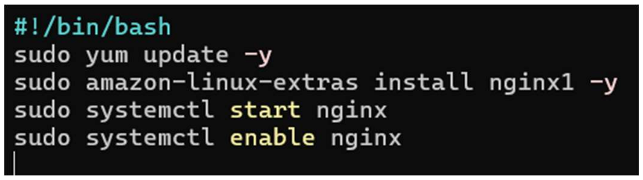
Step 3: Create a Terraform Configuration File.

We'll create the following files:

- **main.tf** – to define providers and resources
- **variables.tf** – to define variables
- **outputs.tf** – to show the public IPs
- **user_data.sh** – to install Nginx

<u>File 1: user_data.sh</u>

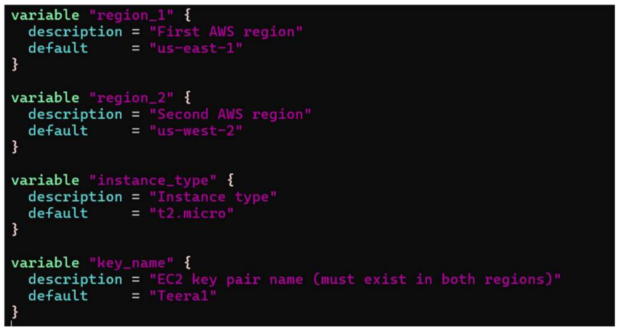
- ✓ Create a shell script that will be executed on instance boot to install and start the Nginx web server.
- ✓ Now save this as user_data.sh in your Terraform directory.



```
#!/bin/bash
sudo yum update -y
sudo amazon-linux-extras install nginx1 -y
sudo systemctl start nginx
sudo systemctl enable nginx
```

<u>File 2: variables.tf</u>

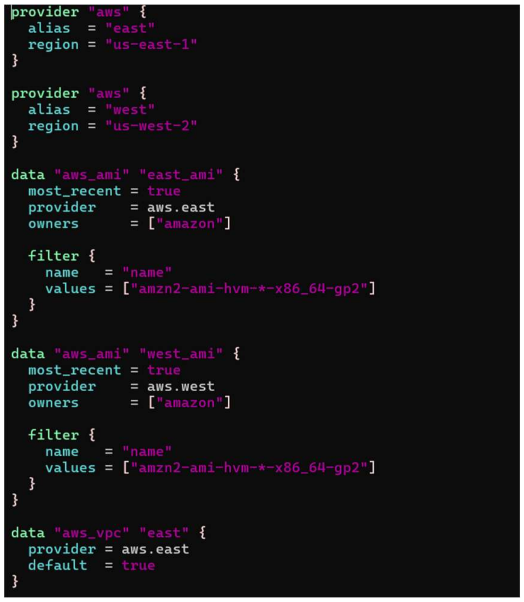
- ✓ In this file defining the required input variables such as AWS AMI ID, instance type, and key pair name in a separate file for better flexibility.



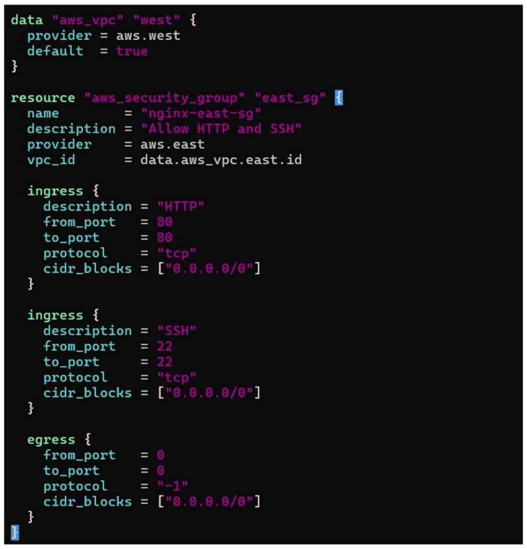
```
variable "region_1" {
  description = "First AWS region"
  default     = "us-east-1"
}

variable "region_2" {
  description = "Second AWS region"
  default     = "us-west-2"
}

variable "instance_type" {
  description = "Instance type"
  default     = "t2.micro"
}

variable "key_name" {
  description = "EC2 key pair name (must exist in both regions)"
  default     = "Teera1"
}
```

<u>File 3: main.tf</u>

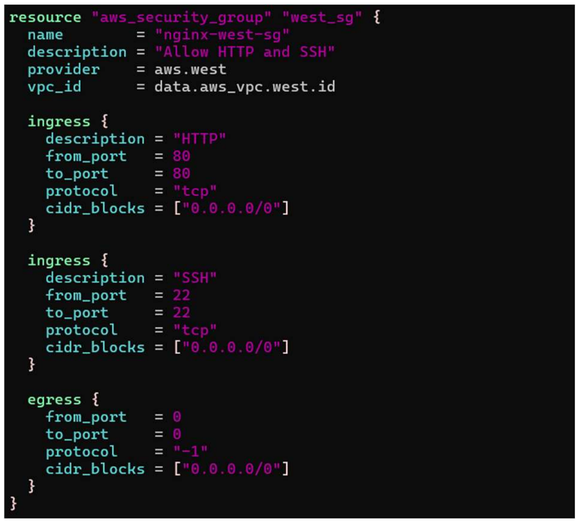
- ✓ In this file defining the AWS provider for each region and create one EC2 instance per region using the appropriate provider.
- ✓ Create a file named main.tf inside the directory ,the extension file of terraform is ".tf "and inside the file we can write an terraform configuration file.
- ✓ To create an new file the command is:
    - **touch main.tf**
- ✓ To open the terraform configuration main.tf file and write terraform configuration, the command is:
    - vi main.tf
- ✓ To change the file into insert mode press "i" and write an terraform configuration and to save an file press "!wq:".
- ✓ The terraform configuration is given below:



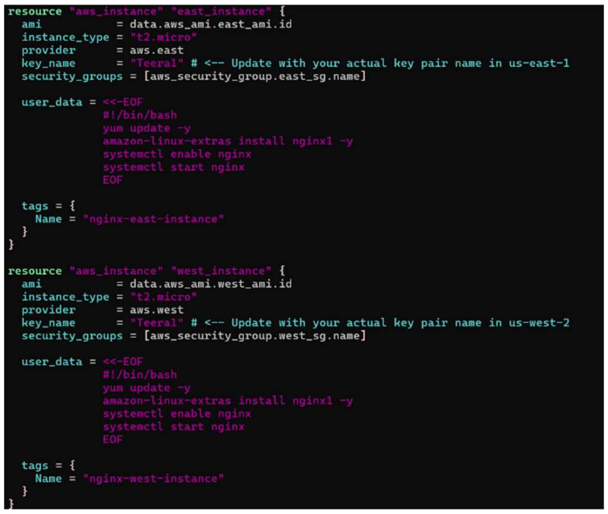
```
provider "aws" {
  alias  = "east"
  region = "us-east-1"
}

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

data "aws_ami" "east_ami" {
  most_recent = true
  provider    = aws.east
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-gp2"]
  }
}

data "aws_ami" "west_ami" {
  most_recent = true
  provider    = aws.west
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-gp2"]
  }
}

data "aws_vpc" "east" {
  provider = aws.east
  default  = true
}
```



```
data "aws_vpc" "west" {
  provider = aws.west
  default  = true
}

resource "aws_security_group" "east_sg" {
  name        = "nginx-east-sg"
  description = "Allow HTTP and SSH"
  provider    = aws.east
  vpc_id      = data.aws_vpc.east.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"]
  }
}
```



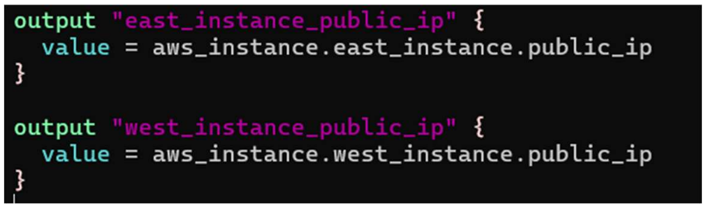
```
resource "aws_security_group" "west_sg" {
  name        = "nginx-west-sg"
  description = "Allow HTTP and SSH"
  provider    = aws.west
  vpc_id      = data.aws_vpc.west.id

  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" "east_instance" {
  ami           = data.aws_ami.east_ami.id
  instance_type = "t2.micro"
  provider      = aws.east
  key_name      = "Teera1" # <-- Update with your actual key pair name in us-east-1
  security_groups = [aws_security_group.east_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-east-instance"
  }
}

resource "aws_instance" "west_instance" {
  ami           = data.aws_ami.west_ami.id
  instance_type = "t2.micro"
  provider      = aws.west
  key_name      = "Teera1" # <-- Update with your actual key pair name in us-west-2
  security_groups = [aws_security_group.west_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-west-instance"
  }
}
```

<u>File 4: outputs.tf</u>

✓ In this file the output of public IPs of both EC2 instances to verify deployment and access the Nginx welcome page.



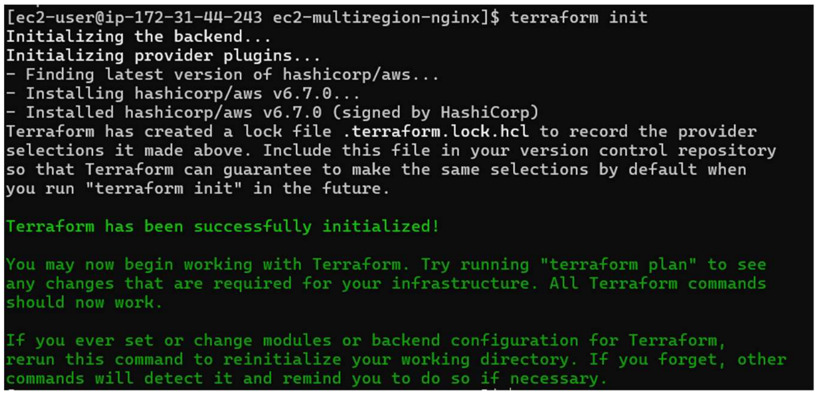
```
output "east_instance_public_ip" {
  value = aws_instance.east_instance.public_ip
}

output "west_instance_public_ip" {
  value = aws_instance.west_instance.public_ip
}
```

Step 4: Initialize Terraform.

✓ To initialize the project directory for Terraform, the command is :

- **terraform init**

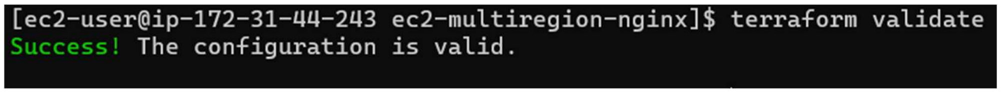
✓ This command downloads the required provider plugins and prepares the working directory.



```
[ec2-user@ip-172-31-44-243 ec2-multiregion-nginx]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.7.0...
- Installed hashicorp/aws v6.7.0 (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.
```

Step 5: Validate the Terraform Configuration

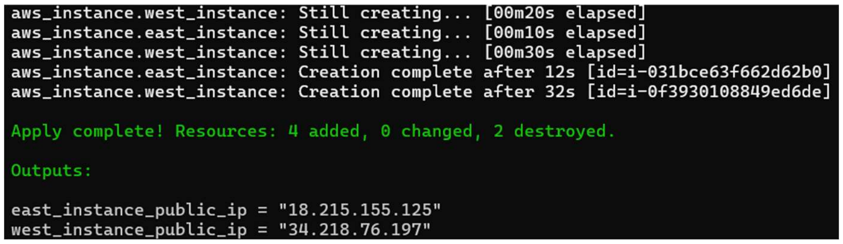
- ✓ To check the configuration is correct or not , the command is:
  - **terraform validate**
- ✓ This helps you to ensure the syntax and structure of the configuration is correct.



```
[ec2-user@ip-172-31-44-243 ec2-multiregion-nginx]$ terraform validate
Success! The configuration is valid.
```

Step 6: Apply the Configuration

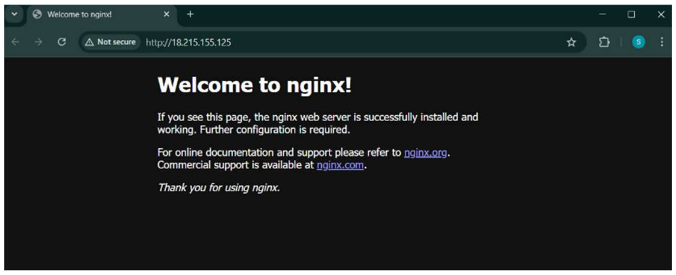
- ✓ To Execute the configuration and create the instances by running , the command is:
  - **terraform apply**
- ✓ Type yes when prompted.
- ✓ Terraform will now create two EC2 instances — one in **us-east-1** and another in **ap-south-1** region.



```
aws_instance.west_instance: Still creating... [00m20s elapsed]
aws_instance.east_instance: Still creating... [00m10s elapsed]
aws_instance.west_instance: Still creating... [00m30s elapsed]
aws_instance.east_instance: Creation complete after 12s [id=i-031bce63f662d62b0]
aws_instance.west_instance: Creation complete after 32s [id=i-0f3930108849ed6de]

Apply complete! Resources: 4 added, 0 changed, 2 destroyed.

Outputs:

east_instance_public_ip = "18.215.155.125"
west_instance_public_ip = "34.218.76.197"
```

Step 7: Verify the EC2 Instances in AWS Console.

- ✓ After the EC2 instances are created, copy the public IPs from the Terraform output and access them via a browser. You should see the Nginx welcome page.
- ✓ You should see one EC2 instance in each specified region.



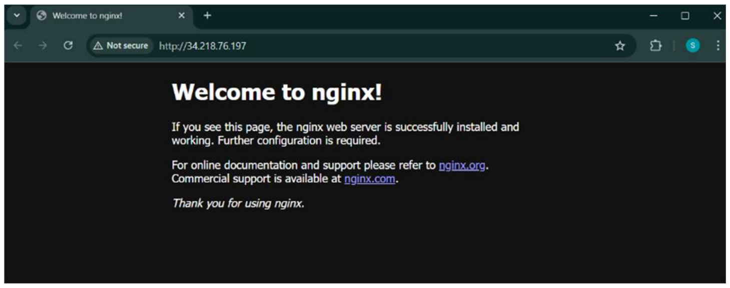
Welcome to nginx!
Not secure http://18.215.155.125

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.



Welcome to nginx!

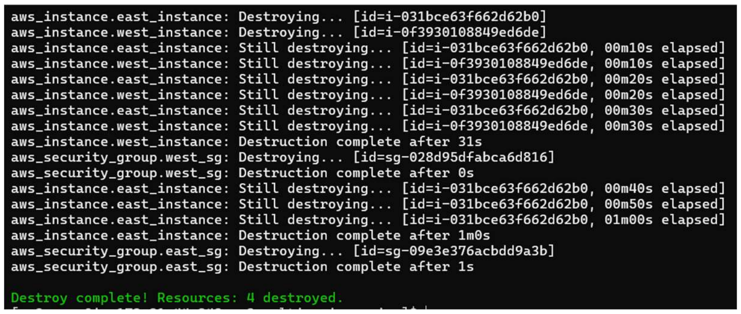
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

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

Thank you for using nginx.

Step 8: Destroy the Infrastructure (Optional).

- ✓ To destroy or clean up the created resources, the command is:
  - **terraform destroy**
- ✓ Type yes to confirm.
- ✓ This command deletes all the EC2 instances created by the Terraform script.



```
aws_instance.east_instance: Destroying... [id=i-031bce63f662d62b0]
aws_instance.west_instance: Destroying... [id=i-0f3930108849ed6de]
aws_instance.east_instance: Still destroying... [id=i-031bce63f662d62b0, 00m10s elapsed]
aws_instance.west_instance: Still destroying... [id=i-0f3930108849ed6de, 00m10s elapsed]
aws_instance.east_instance: Still destroying... [id=i-031bce63f662d62b0, 00m20s elapsed]
aws_instance.west_instance: Still destroying... [id=i-0f3930108849ed6de, 00m20s elapsed]
aws_instance.east_instance: Still destroying... [id=i-031bce63f662d62b0, 00m30s elapsed]
aws_instance.west_instance: Still destroying... [id=i-0f3930108849ed6de, 00m30s elapsed]
aws_instance.west_instance: Destruction complete after 31s
aws_security_group.west_sg: Destroying... [id=sg-028d95dfabca6d816]
aws_security_group.west_sg: Destruction complete after 0s
aws_instance.east_instance: Still destroying... [id=i-031bce63f662d62b0, 00m40s elapsed]
aws_instance.east_instance: Still destroying... [id=i-031bce63f662d62b0, 00m50s elapsed]
aws_instance.east_instance: Still destroying... [id=i-031bce63f662d62b0, 01m00s elapsed]
aws_instance.east_instance: Destruction complete after 1m0s
aws_security_group.east_sg: Destroying... [id=sg-09e3e376acbdd9a3b]
aws_security_group.east_sg: Destruction complete after 1s

Destroy complete! Resources: 4 destroyed.
```

Conclusion

- ✓ Using Terraform, we successfully launched EC2 instances in two different AWS regions and installed Nginx automatically with a single script. This setup showcases the power of multi-region deployment and infrastructure automation.

************** **TASK COMPLETED** **************