Terraform Task

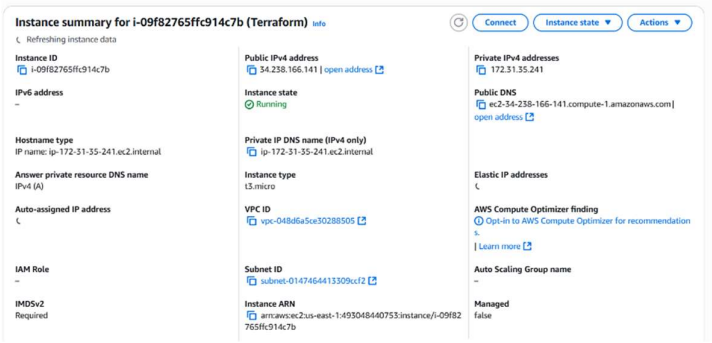
TASKS

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.

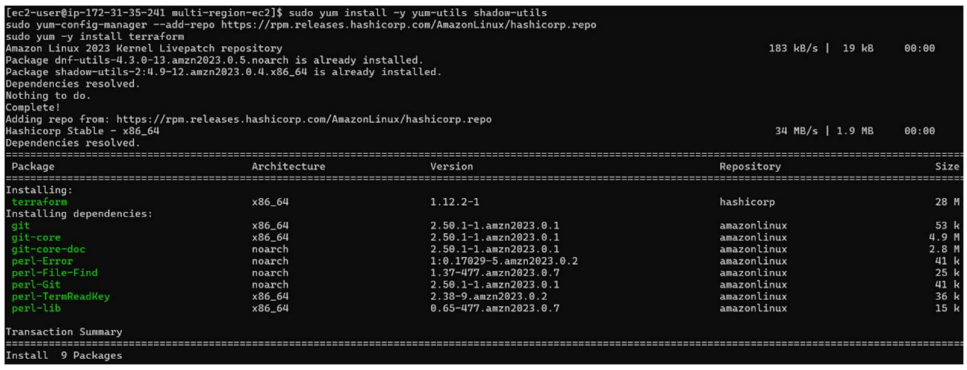


1. Launch Linux EC2 instances in two regions using a single Terraform file.

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.



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 multi-region-ec2**
- ✓ Now to move into the created directory, the command is:
  - **cd multi-region-ec2**
- ✓ This folder will hold the Terraform configuration for deploying EC2 instances in two AWS regions.

```
[ec2-user@ip-172-31-35-241 ~]$ mkdir multi-region-ec2
[ec2-user@ip-172-31-35-241 ~]$ cd multi-region-ec2
[ec2-user@ip-172-31-35-241 multi-region-ec2]$
```

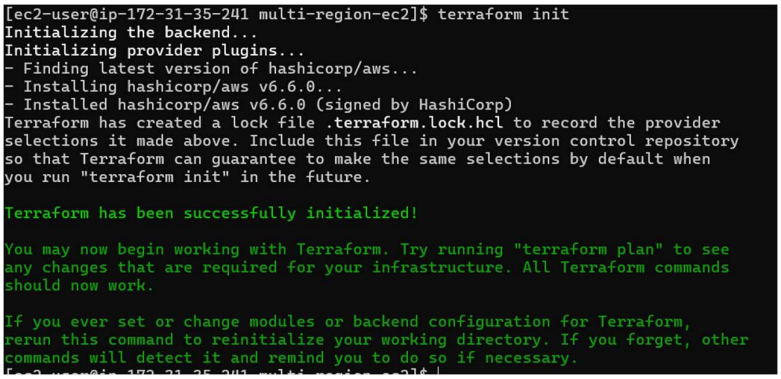
Step 3: Create a Terraform Configuration File.

- ✓ 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:
- ✓ This file defines two AWS providers with different region aliases and launches one EC2 instance in each region using appropriate AMI IDs**.**
- ✓ Terraform Configuration File:

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

provider "aws" {
  alias  = "ap_south"
  region = "ap-south-1"
}

resource "aws_instance" "east_instance" {
  provider      = aws.us_east
  ami           = "ami-0c02fb55956c7d316" # Amazon Linux 2 in us-east-1
  instance_type = "t2.micro"
  tags = {
    Name = "EC2-US-East"
  }
}

resource "aws_instance" "ap_instance" {
  provider      = aws.ap_south
  ami           = "ami-03f4878755434977f" # Amazon Linux 2 in ap-south-1
  instance_type = "t2.micro"
  tags = {
    Name = "EC2-AP-South"
  }
}
```

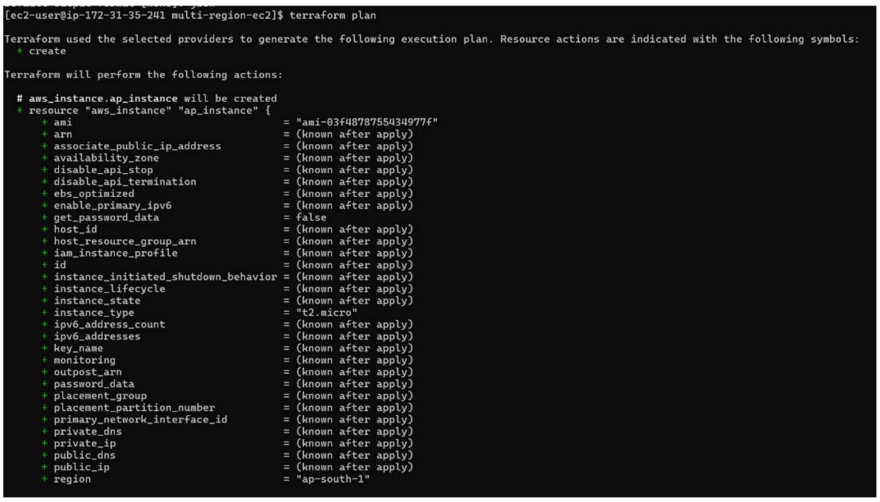
Step 4: Initialize Terraform.

- ✓ To initialize the project directory for Terraform, the command is :
  - **terraform init**
- ✓ This command downloads the required provider plugins and prepares the working directory.



Step 5: Review the Execution Plan.

- ✓ To see the resources that will be created , the command is:
  - **terraform plan**
- ✓ This helps you verify what Terraform intends to do before making any actual changes.



Step 6: Apply the Configuration

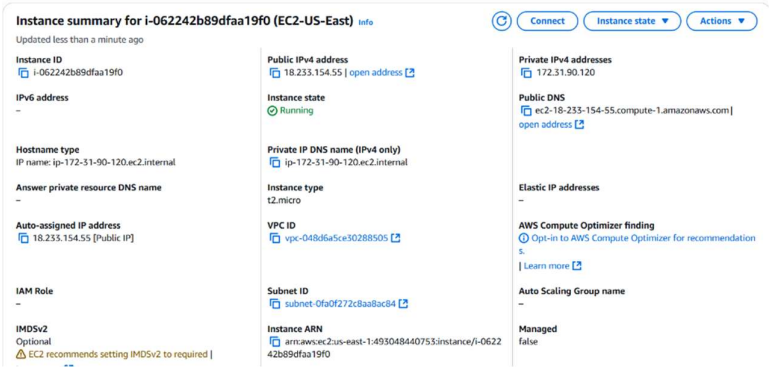
- ✓ To Execute the configuration and create the instances by running , the command is:
  - **terraform apply**
- ✓ Type yes when prompted.
- ✓ Terraform will now create two EC2 instances — one in **us-east-1** and another in **ap-south-1** region.

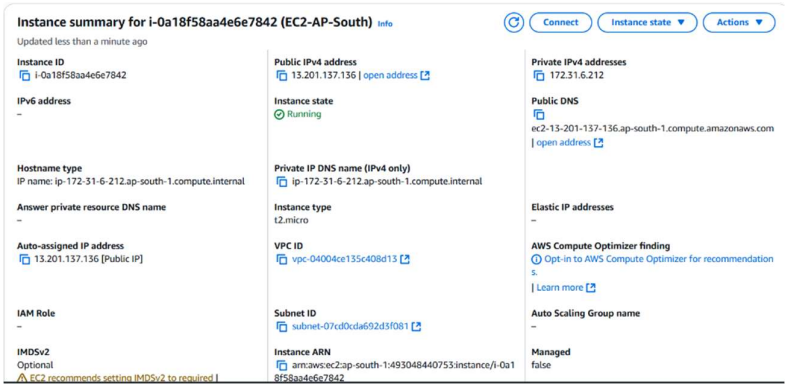
```
[ec2-user@ip-172-31-35-241 multi-region-ec2]$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.ap_instance will be created
  + resource "aws_instance" "ap_instance" {
      + ami                                  = "ami-03f4878755434977f"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
```

```
Plan: 2 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_instance.east_instance: Creating...
aws_instance.ap_instance: Creating...
aws_instance.east_instance: Still creating... [00m10s elapsed]
aws_instance.ap_instance: Still creating... [00m10s elapsed]
aws_instance.east_instance: Still creating... [00m20s elapsed]
aws_instance.ap_instance: Still creating... [00m20s elapsed]
aws_instance.east_instance: Still creating... [00m30s elapsed]
aws_instance.ap_instance: Still creating... [00m30s elapsed]
aws_instance.east_instance: Creation complete after 32s [id=i-062242b89dfaa19f0]
aws_instance.ap_instance: Creation complete after 36s [id=i-0a18f58aa4e6e7842]

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

Step 7: Verify the EC2 Instances in AWS Console.

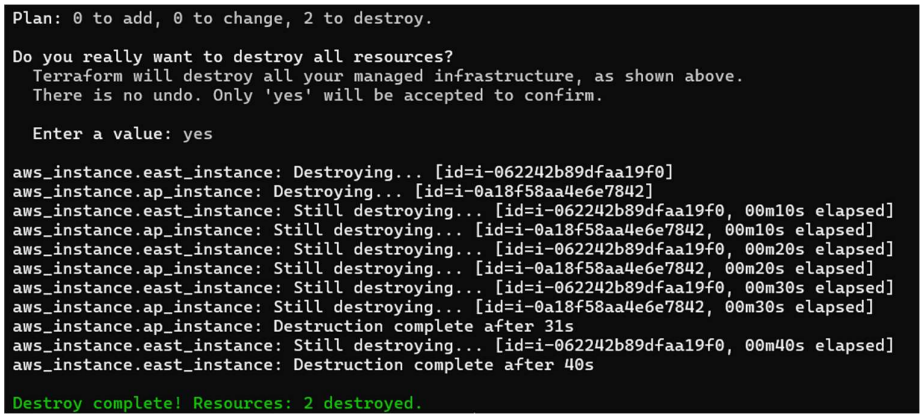
- ✓ Log in to the AWS Management Console and switch between **us-east-1** and **ap-south-1** regions to confirm that the instances are running.
- ✓ You should see one EC2 instance in each specified region.





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.



Conclusion

- ✓ By following this guide, you successfully launched Linux EC2 instances in two different AWS regions using a **single Terraform file**. This approach is useful for managing **multi-region deployments** from a centralized infrastructure codebase.

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