MARCH 20, 2025

# VPC NETWORK CREATION USING THE TERRAFORM SCRIPTT

# TERRAFORM – A AUTOMATION TOOL

TERRAFORM DOCUMENTATION

AYUSH A. SHAHA
CLOUDBLITZ ,KOTHRUDE
Kothrude ,pune

# INDEX

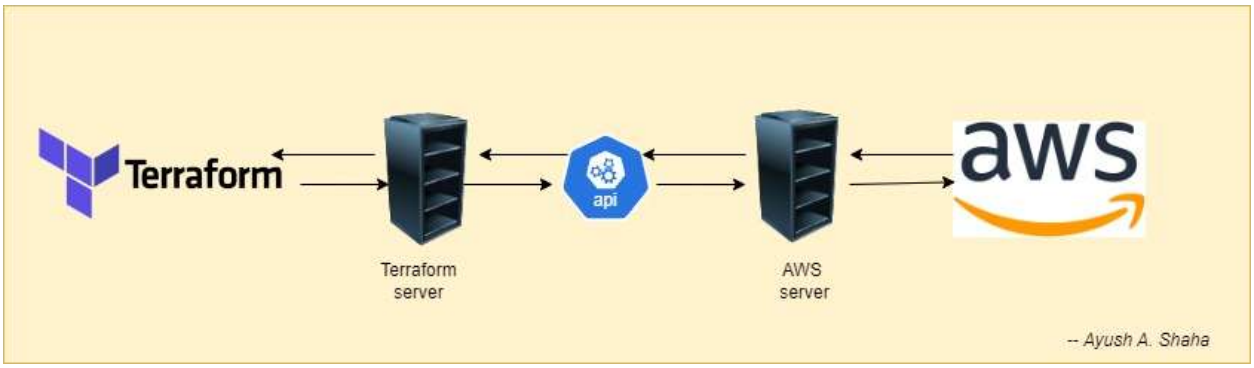
# ➢ WHAT IS TERRAFORM AND WHAT IS ITS HISTORY.

- **What is a Terraform?**

Terraform is an IAC (Infrastructure As Code) tool that contains a human-readable Configuration file for managing infrastructure, whether on-premise or cloud.

- **How does the Terraform work?**

Terraform can integrate with the cloud because of the API, which stands for 'Application Programming Interface.' This is responsible for making the connection between the cloud and Terraform.




- **What is the history of the Terraform?**

Terraform, created by HashiCorp in July 2014, is an open-source Infrastructure as Code (IaC) tool designed to automate cloud resource provisioning. It was invented to provide a declarative approach to managing infrastructure, ensuring consistency, and reducing manual errors. Unlike traditional configuration tools, Terraform introduced state management, multi-cloud support, and immutable infrastructure to prevent configuration drift. It enables scalability and automation, making infrastructure deployment faster and more efficient. Terraform's popularity grew as it became a universal tool for managing cloud and on-prem environments. Today, it is widely used in DevOps for provisioning and maintaining infrastructure efficiently.

# ➢ TERRAFORM INSTALLATION

For the installation of the Terraform, it is recommended that you refer to the authentic documentation of the Hasicorp Terraform itself as frequently changes the process of the installation of brings regular updates for improved security. But for reference here is the installation process for the AWS terminal which is up-to-date at the time.

**STEP 1:-** Install yum-config-manager to manage your repositories.

```
sudo yum install -y yum-utils
```

**STEP 2:-** Use yum-config-manager to add the official HashiCorp Linux repository.

```
sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
```

**STEP 3:-**Install Terraform from the new repository.

```
sudo yum -y install terraform
```

**STEP 4:-**Verify that the installation worked by opening a new terminal session and listing Terraform's available subcommands.

```
terraform -help
```

# ➢ TERRAFORM LANGUAGE AND BLOCKS

Terraform uses HashiCorp Configuration Language (HCL), a declarative language designed to define infrastructure as code (IaC). It is human-readable and supports JSON formatting.

- **Terraform Blocks:-**

### 1. Provider Block

This type of block is used to specify the type of provider for eg. AWS, or GCP. AZURE.

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

### 2. Resource Block

This type of block is used to specify the type of AWS or any other resource from the provider is used.

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

### 3. Variable Block

This type of block allows parameterizing configurations.

```
variable "instance_type" {
  default = "t2.micro"
}
```

### 4. **Output Block**

This type of block is used to show the result after making the configuration.

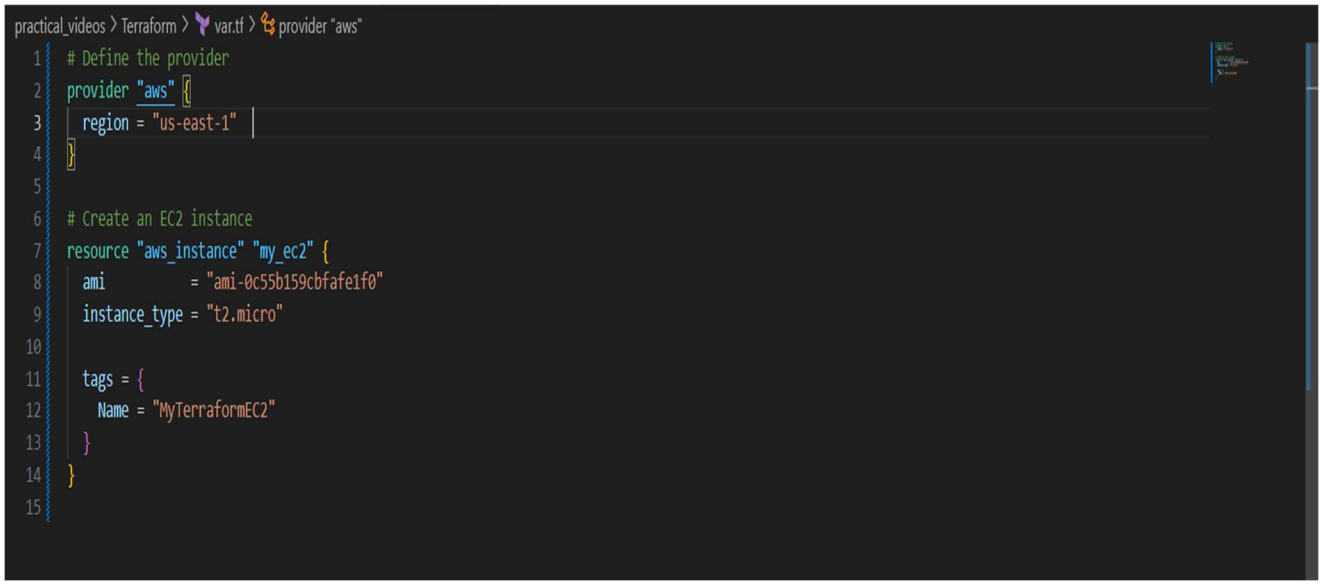
```
output "instance_id" {
  value = aws_instance.example.id
}
```

### 5. **Data Block**

This Block is used for fetching information about the currently available resources.

```
data "aws_ami" "example" {
  most_recent = true
  owners      = ["self"]
```

- EXAMPLE OF THE BLOCKS USED IN THE SCRIPT



# ➢ WHAT IS THE TERRAFORM LIFECYCLE?

In the Terraform Lifecycle basically, there are 5 stages and that are as follows and each step is unique in its own.

### 1. Write

This is the first step of the Terraform lifecycle in this step the user creates a Terraform script with the extension '.tf' and moves to the next step.

### 2. Init

This is the second step of the Terraform lifecycle in this step the user initializes the Terraform with the command " `terraform init` ".

### 3. Plan

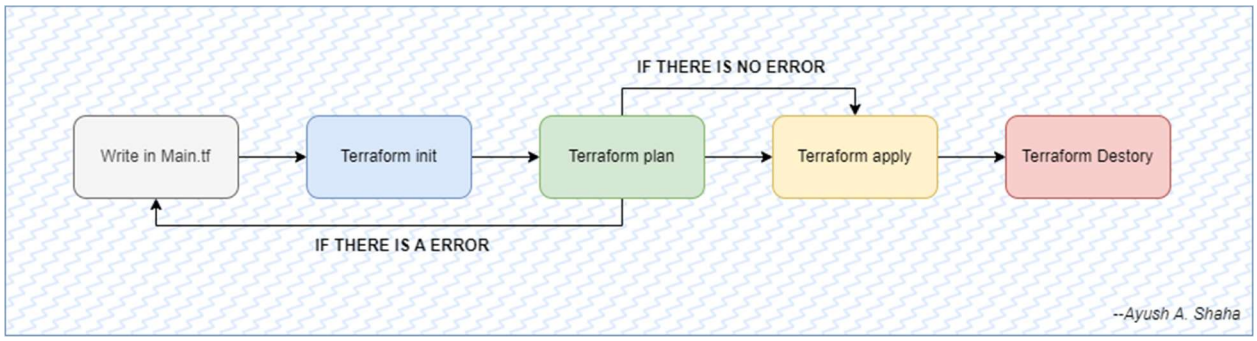
This is the third step of the Terraform lifecycle in this step the Terraform checks for the error in the script.

### 4. Apply

This is the fourth and the most important step in the terraform lifecycle which is responsible for building the infrastructure that is required.

### 5. Destroy

This is the Fifth step which is used to destroy the infrastructure that is built using the Terraform.



# ➢ CREATION OF VPC USING THE TERRAFORM

Here we create a whole network with all the components like vpc, subnet, route table, and other required things. All these things are achieved and can be built using the Terraform script we can also launch the instance in the same network that we have created just we need one thing and that is a Terraform script which is written in HCL language and add this script in the ‘main.tf ’file and follow the life cycle to build the infrastructure

These are the certain steps that are required to build the vpc and the instance using the Terraform script

### 1. Create a main.tf on the terminal or machine

```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Mon Mar 24 18:59:21 2025 from 13.48.4.203
[ec2-user@ip-172-31-23-170 ~]$ ls
[ec2-user@ip-172-31-23-170 ~]$ mkdir main.tf
[ec2-user@ip-172-31-23-170 ~]$ ls
main.tf
```

### 2. Add the script to the ‘Main.tf ‘ and save

```
provider "aws" {
  region = "eu-north-1"
  access_key = "AKIA2NK3YMEB4YALB5OV"
  secret_key = "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"
}
resource "aws_vpc" "Ayush" {
  cidr_block       = "10.0.0.0/16"
  tags = {
  Name = "Ayush"
  }
}
resource "aws_subnet" "main" {
  vpc_id = aws_vpc.Ayush.id
  map_public_ip_on_launch = true
  cidr_block     = "10.0.0.0/24"
}
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.Ayush.id

  tags = {
    Name = "my-igw"
  }
}
resource "aws_route_table" "example" {
  vpc_id = aws_vpc.Ayush.id


  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
}
resource "aws_route_table_association" "a" {
```

### 3. Initilize the terraform

```
[root@ip-172-31-23-170 ~]# tf init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.92.0

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.
[root@ip-172-31-23-170 ~]#
```

## 4. Check the Plan and find the error

```
commands will detect it and remind you to do so if necessary.
[root@ip-172-31-23-170 ~]# tf plan

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.this will be created
  + resource "aws_instance" "this" {
      + ami                                  = "ami-0c2e61fdcb5495691"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (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                        = "t3.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)
```

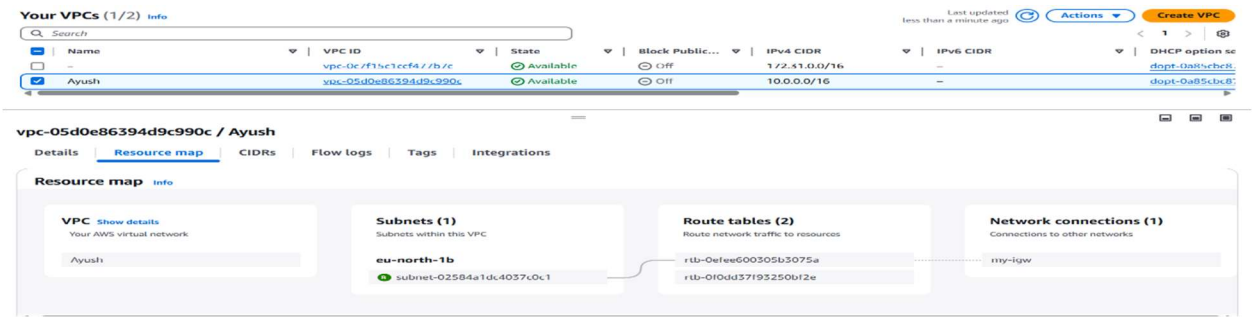
## 5. Built the Planned Infrastructure by apply command

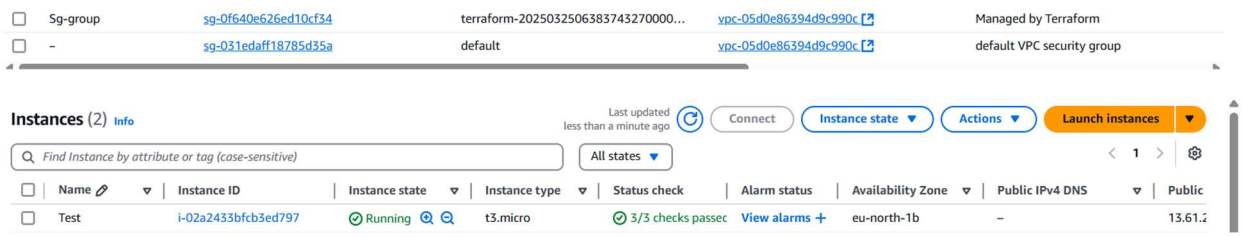
```
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "Ayush"
        }
      + tags_all                             = {
          + "Name" = "Ayush"
        }
    }

Plan: 7 to add, 0 to change, 0 to destroy.
aws_vpc.Ayush: Creating...
aws_vpc.Ayush: Creation complete after 1s [id=vpc-05d0e86394d9c990c]
aws_internet_gateway.igw: Creating...
aws_subnet.main: Creating...
aws_security_group.sg: Creating...
aws_internet_gateway.igw: Creation complete after 1s [id=igw-077e6569198e3848a]
aws_route_table.example: Creating...
aws_route_table.example: Creation complete after 1s [id=rtb-0efee600305b3075a]
aws_security_group.sg: Creation complete after 3s [id=sg-0f640e626ed10cf34]
aws_subnet.main: Still creating... [10s elapsed]
aws_subnet.main: Creation complete after 12s [id=subnet-02584a1dc4037c0c1]
aws_route_table_association.a: Creating...
aws_instance.this: Creating...
aws_route_table_association.a: Creation complete after 0s [id=rtbassoc-0689c160d90466aae]
aws_instance.this: Still creating... [10s elapsed]
aws_instance.this: Creation complete after 12s [id=i-02a2433bfcb3ed797]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
[root@ip-172-31-23-170 ~]#
[root@ip-172-31-23-170 ~]#
```

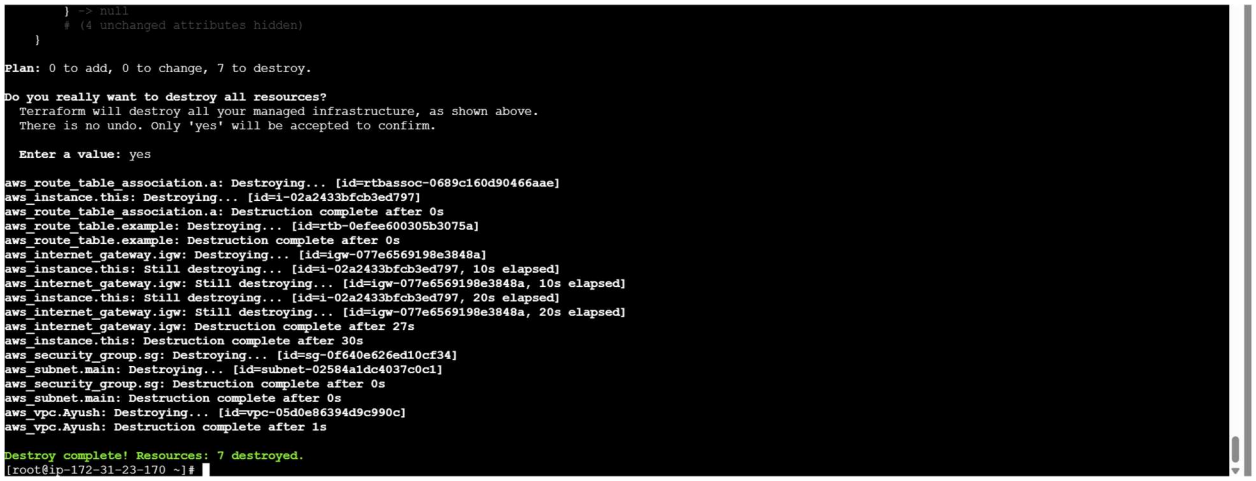
## 6. Check the infra






This is the after screenshot when the terraform apply command was fired

## 7. Destroy the built infrastructure



```
        } -> null
        # (4 unchanged attributes hidden)
    }

Plan: 0 to add, 0 to change, 7 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_route_table_association.a: Destroying... [id=rtbassoc-0689c160d90466aae]
aws_instance.this: Destroying... [id=i-02a2433bfcb3ed797]
aws_route_table_association.a: Destruction complete after 0s
aws_route_table.example: Destroying... [id=rtb-0efee600305b3075a]
aws_route_table.example: Destruction complete after 0s
aws_internet_gateway.igw: Destroying... [id=igw-077e6569198e3848a]
aws_instance.this: Still destroying... [id=i-02a2433bfcb3ed797, 10s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-077e6569198e3848a, 10s elapsed]
aws_instance.this: Still destroying... [id=i-02a2433bfcb3ed797, 20s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-077e6569198e3848a, 20s elapsed]
aws_internet_gateway.igw: Destruction complete after 27s
aws_instance.this: Destruction complete after 30s
aws_security_group.sg: Destroying... [id=sg-0f640e626ed10cf34]
aws_subnet.main: Destroying... [id=subnet-02584a1dc4037c0c1]
aws_security_group.sg: Destruction complete after 0s
aws_subnet.main: Destruction complete after 0s
aws_vpc.Ayush: Destroying... [id=vpc-05d0e86394d9c990c]
aws_vpc.Ayush: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
[root@ip-172-31-23-170 ~]#
```

This firing the ‘Terraform destroy’ command all the built-up infrastructure is destroyed

***…END***