

# Experiment No. 6

**Aim –** To Build, change, and destroy AWS infrastructure Using Terraform.

# Theory –

1. **Introduction to Terraform:** Terraform is an open-source infrastructure-as-code (IaC) tool that allows you to define, provision, and manage cloud infrastructure through code. It enables the creation and modification of AWS resources like EC2 instances, VPCs, and S3 buckets by writing configuration files in a simple, declarative language called HCL (HashiCorp Configuration Language).

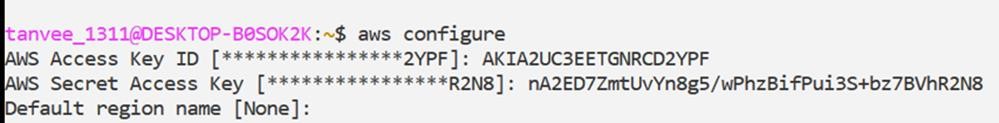
# Terraform Workflow Overview:

* + **Write Configuration:** Define the desired infrastructure in .tf files.
  + **Initialize:** Initialize the working directory where Terraform will store its state and download required provider plugins.
  + **Plan:** Terraform creates an execution plan that shows what changes it will make to match the desired state.
  + **Apply:** Apply the changes to create, modify, or destroy resources in AWS.
  + **Destroy:** Tear down and remove all infrastructure defined in the configuration.

1. **Key Components of Terraform:**
2. **Providers:** Providers are plugins that interact with external services (e.g., AWS, Azure). Terraform's AWS provider allows you to manage AWS resources.
3. **Resources:** Resources are the building blocks of your infrastructure. Each resource block represents a specific piece of infrastructure, like an EC2 instance or an S3 bucket.
4. **State:** Terraform keeps track of the resources it manages using a "state" file. This file acts as a map between the configuration and the real-world resources.
5. **Modules:** Modules are reusable pieces of configuration that can be shared across projects or teams.
   * Your AWS credentials.

# Configure the AWS CLI from your terminal. Follow the prompts to input your AWS Access Key ID and Secret Access Key.

* + $ aws configure



1. **Write configuration:** The set of files used to describe infrastructure in Terraform is known as a Terraform configuration. You will write your first configuration to define a single AWS EC2 instance. Each Terraform configuration must be in its own working

directory.

Create a directory for your configuration.

* $ mkdir learn-terraform-aws-instance Change into the directory.
* $ cd learn-terraform-aws-instance Create a file to define your infrastructure.
* $ sudo nano main.tf



# Open main.tf in your text editor, paste in the configuration below, and save the file. main.tf

terraform{ required\_providers{ aws ={

source = "hashicorp/aws"

version = "~> 3.27"

}

}

required\_version = ">= 0.14.9"

}

provider "aws" { profile = "default" region = "us-west-2"

}

resource "aws\_instance" "app\_server" {

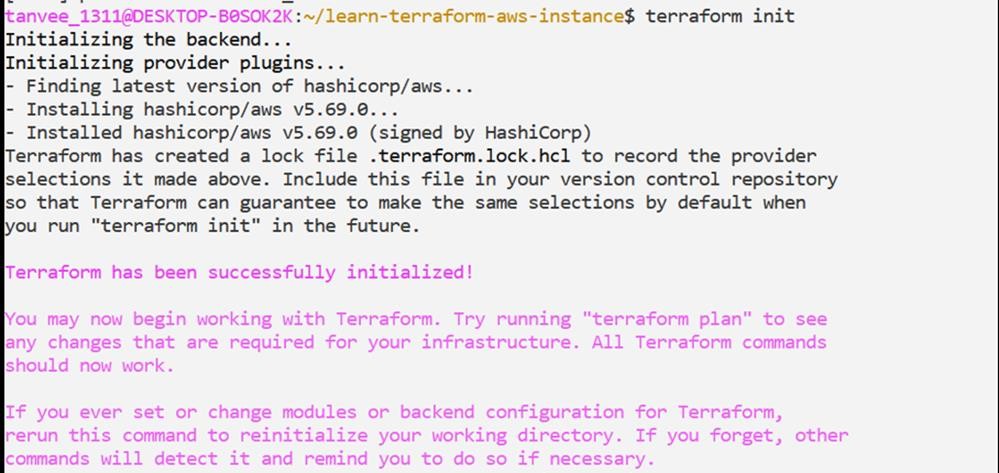
ami = "ami-830c94e3" instance\_type = "t2.micro" tags ={

Name = "ExampleAppServerInsatnce"

}

}

1. **When you create a new configuration —** you need to initialize the directory with terraform init. Initialize the directory.
   * $ terraform init



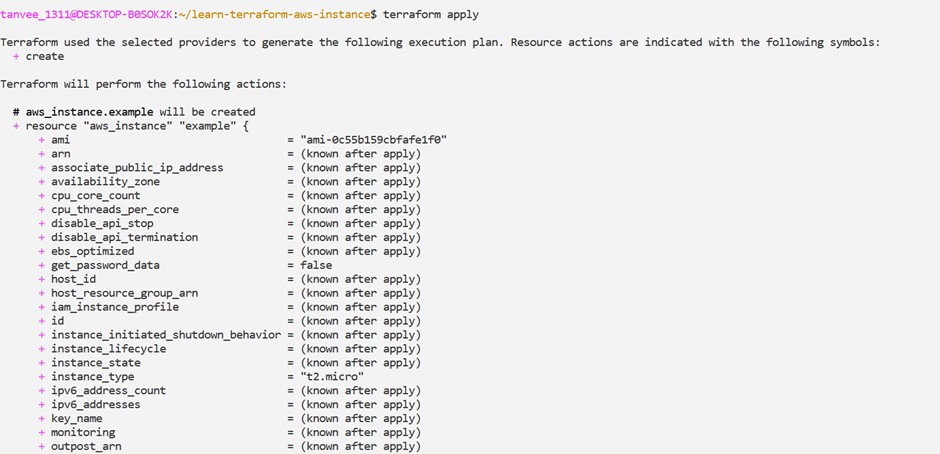
# Validate your configuration.

- $ terraform validate

# Create infrastructure

Apply the configuration now with the terraform apply command. Terraform will print output similar to what is shown below.

* + $ terraform apply



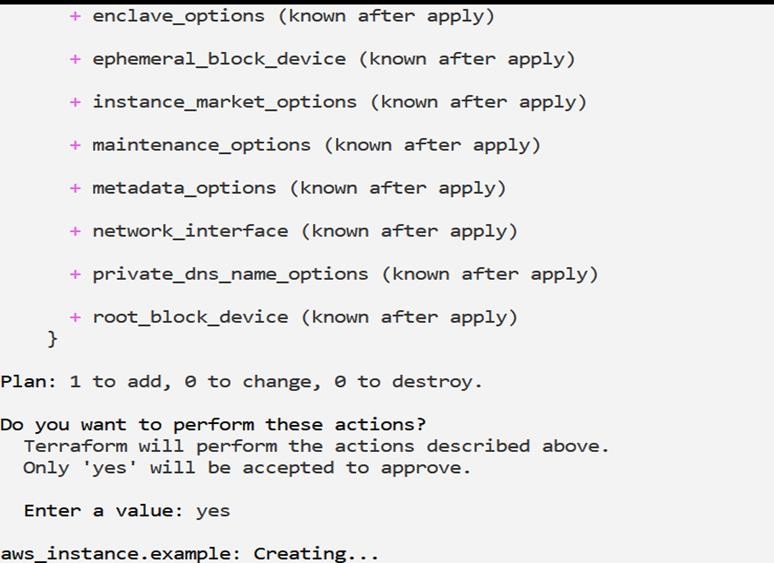
# Inspect state

When you applied your configuration, Terraform wrote data into a file called

terraform.tfstate. Terraform stores the IDs and properties of the resources it manages in this file, so that it can update or destroy those resources going forward.

* + $ terraform show





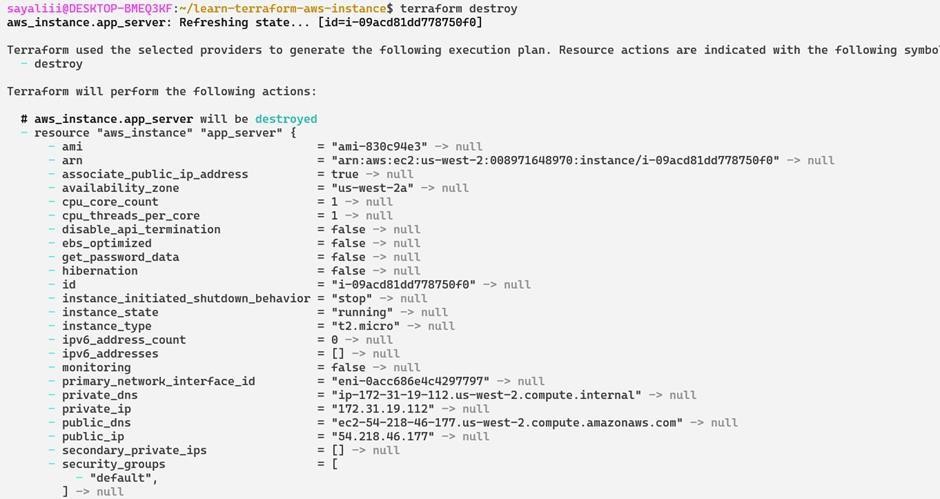




# Destroy

The terraform destroy command terminates resources managed by your Terraform project.

* + $ terraform destroy







# Conclusion –

Therefore, we built, changed, and destroyed AWS infrastructure Using Terraform.