**TERRAFORM**

**Terraform** is a tool that is used to manage and provision infrastructure through code instead of manual process. It uses declarative approach instead of imperative approach to do the same. It was developed by Mitchell Hashimoto from Hashicorp

Terraform uses **Hashicorp Configure Language (HCL)**, it is like JSON, and easily human readable.

To install terraform go to “https://developer.hashicorp.com/terraform/downloads” and choose the desired version and OS for the installation.

Useful terminologies and commands in Terraform:

1. **Provider** – A provider is a plugin responsible for understanding API interactions and exposing resources. To make a provider available on Terraform, we need to run terraform init command.

For example, if you want to use AWS provider create a .tf file like below:

Also, one should have the access key and secret key generated in AWS for the user and they should be provided in the .tf file as shown below

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 4.0"

}

}

}

# Configure the AWS Provider

provider "aws" {

region = "us-west-2"

access\_key = "my-access-key"

secret\_key = "my-secret-key"

}

The terraform {} block contains Terraform settings, including the required providers Terraform will use to provision your infrastructure. Terraform installs providers from the [Terraform Registry](https://registry.terraform.io/) by default

1. **terraform init** – After you create a new configuration — or check out an existing configuration from version control — you need to initialize the directory with terraform init command
2. **terraform plan** - The terraform plan command creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure.
3. **terraform validate**- Validate runs checks that verify whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state. It is thus primarily useful for general verification of reusable modules, including correctness of attribute names and value types.
4. **terraform apply**- The terraform apply command executes the actions proposed in a Terraform plan.
5. **terraform destroy** – This command destroys or removes the applied configuration or infrastructure

To create an EC2 instance