# Automating Infrastructure using Terraform

This is document is about how to implement the project 1 of DevOps certification training: Automating Infrastructure using Terraform. The source code is in this GitHub repository: https://github.com/pratipc/Automating-Infrastructure-using-Terraform--Simplilearn-Project

## Objective

This Terraform configuration that will create an EC2 instance in the AWS account.

## Procedure

To automate infrastructure using Terraform, AWS account credentials and a Keypair are required. Below is the code to launch an EC2 instance using Terraform:

```
provider "aws" {

  region = "us-east-1"

  access_key = "ACCESS_KEY"

  secret_key = "SECRET_KEY"

}
```

```
resource "aws_instance" "terraform_instance" {
  ami = "AMI_ID"
  instance_type = "t2.micro"
  key_name = "KEYPAIR_NAME”
}
```

## Creating EC2 Instance using terraform

- First we make a folder named Terrafrom when we download the zip file.
- Download terraform zip file and install it using:
  - $ wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
  - $ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
  - sudo apt update
  - sudo apt install terraform

- Configure AWS credentials: Before we can create an EC2 instance, we need to configure our AWS credentials in Terraform.
- Create a Terraform configuration file: Next, we need to create two Terraform configuration files that defines the EC2 instance we want to create. These files should include information such as the instance type, AMI, region, key name and key pair.
- Initialize Terraform: Before we can create the EC2 instance, we need to initialize Terraform. This can be done by running the $ “**terraform init**" command.

- Terraform plan: This command creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure. By default, when Terraform creates a plan it:
    - It reads the current state of any already-existing remote objects to make sure that the Terraform state is up-to-date.
    - Compares the current configuration to the prior state and noting any differences.
    - Proposes a set of change actions that should, if applied, make the remote objects match the configuration.
- Terraform apply: The $ “**terraform apply**” command executes the actions proposed in a Terraform plan.
- Verify the EC2 instance: After the EC2 instance is created, we can verify that it was created successfully by logging into the AWS Management Console or logging in using a Secure Shell toolbox.

## Installing Jenkins and Java

- Update the Ubuntu package index: This can be done by running the command "**sudo apt-get update**" in the terminal.
- Install Java: Jenkins requires Java to be installed on the system. To install Java, run the command "**sudo apt install default-jdk**" in the terminal.
- Install Jenkins: we can install Jenkins by running the command "**sudo apt install Jenkins -y** " in the terminal.
- Start Jenkins service: start the Jenkins service by running the command "**sudo systemctl start jenkins**" in the terminal.
- Verify the Jenkins installation: To verify that Jenkins is installed check “**Jenkins  --version**” in the terminal.

## Installing Python

- Update the Ubuntu package index:This can be done by running the command "**sudo apt-get update**" in the terminal.
- Install Python: To install the latest version of Python, run the command "**sudo apt get install python3.8**" in the terminal.
- Verify the Python installation: To verify that Python is installed and accessible, open a terminal and enter "**python3 --version**".

## Conclusion

Terraform is an interesting Infrastructure-as-Code project with many benefits, e.g. Provides the ability to spin up an entire environment in minutes & It reduces time to rollout complex network and storage changes to less than a few minutes. Using design best practices of terraform, enterprises can quickly build and manage infrastructure, which is highly scalable and efficient. Further, this automation can be hooked to a Jenkins pipeline project for automated code pushes for infra changes which can be tied to a standard release and deployment process.