Lab Manual- AWS Automation with Terraform

Prepared for: TechPledge

Date: 18th Nov 2018

Prepared by: Bipin Sinhaa

Document Name: Lab Manual

Document Number DevOpsLab279

Contributor:

Shruti Sinhaa

Table of Contents

1 OBJECTIVE

Terraform is an agnostic cloud-provisioning tool created by Hashicorp. Terraform allows you to create, manage, and update your infrastructure in a safe and efficient manner.

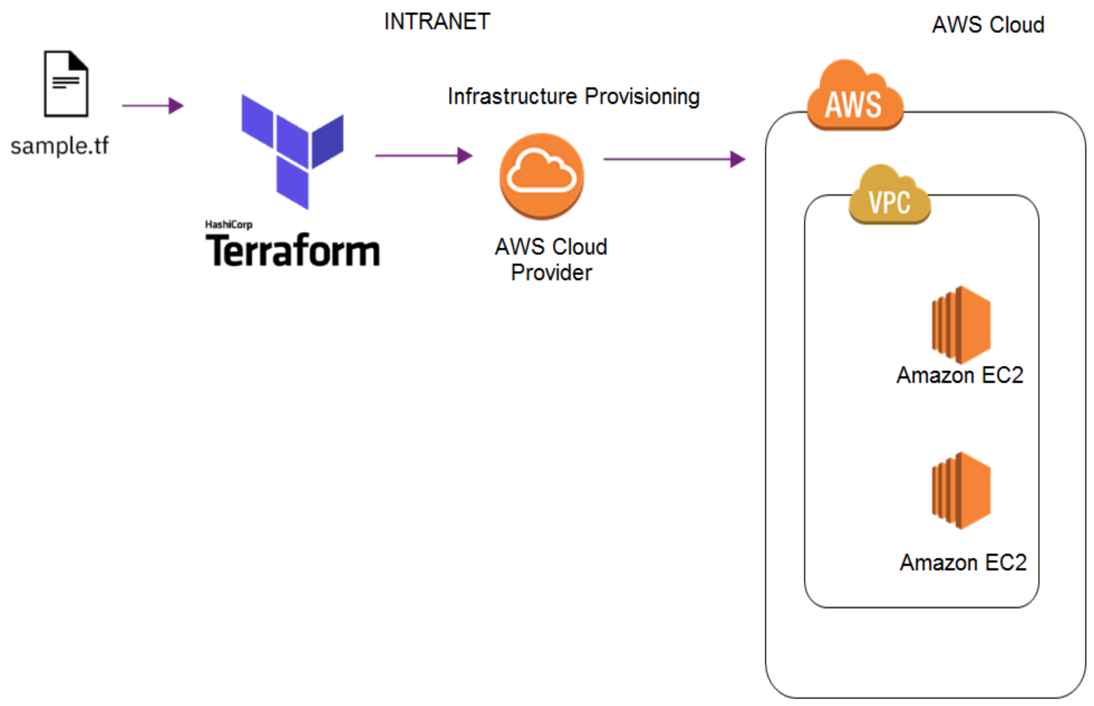
Terraform's configuration is all done using a language called the HashiCorp Configuration Language (HCL).

In This Lab will cover the basics of Installing and configure Terraform on my Windows 10 Platform.

2 PRE-REQUISISTE

- Prior knowledge of Linux
- Accounts in AWS
- A local Computer with 4 CPU, 16 GB RAM, 200 GB disk space

3 Configure Traaform to Provision EC2 Instance

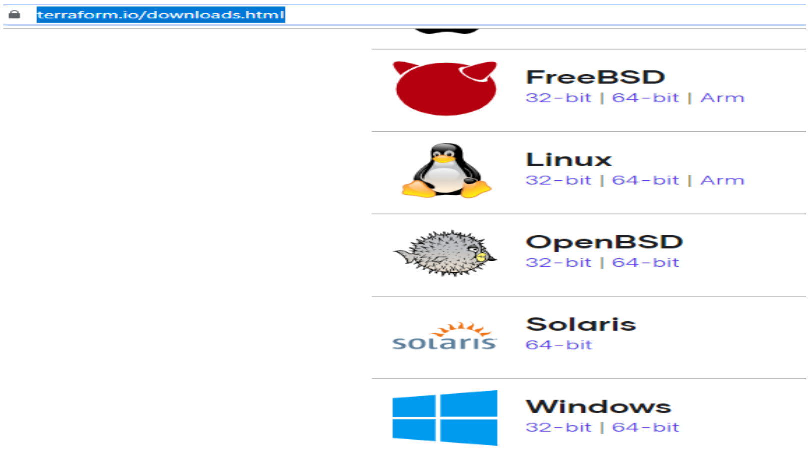


3.1 Install TerraForm on Wndows 10

Steps 1: Download terraform for windows from below URL

https://www.terraform.io/downloads.html

- Note: Terraform is packaged as a zip archive, so after downloading Terraform, unzip the package. Terraform runs as a single binary named terraform. Any other files in the package can be safely removed and Terraform will still function

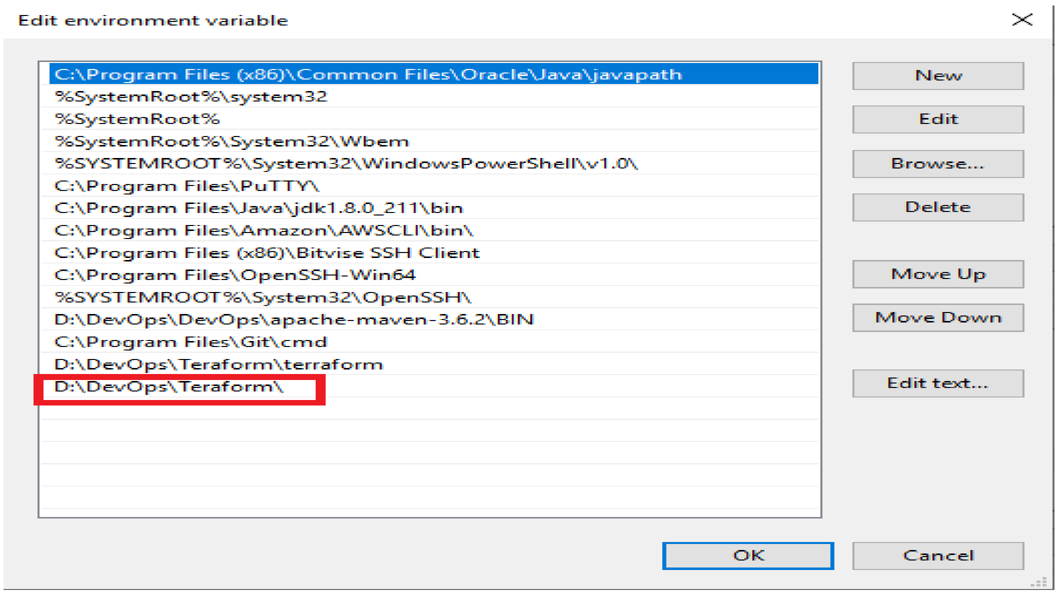


-

Steps 2: Copy files from the zip to **“d:\DevOps\Terraform”** for example. That’s our terraform **PATH.**

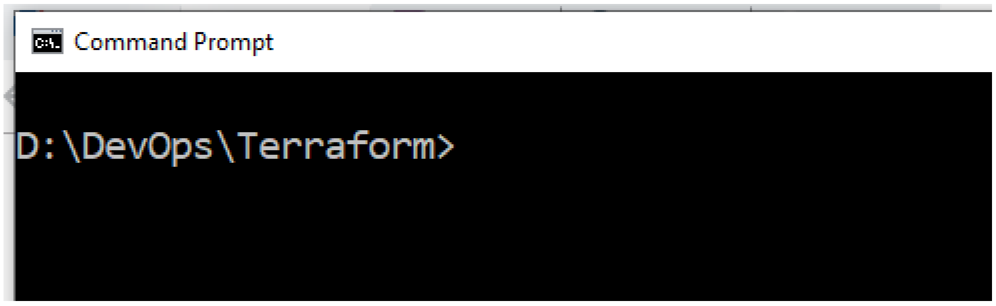
Steps 3: The final step is to make sure that the terraform binary is available on the **PATH**.

Steps 4: Set the Environment Variable



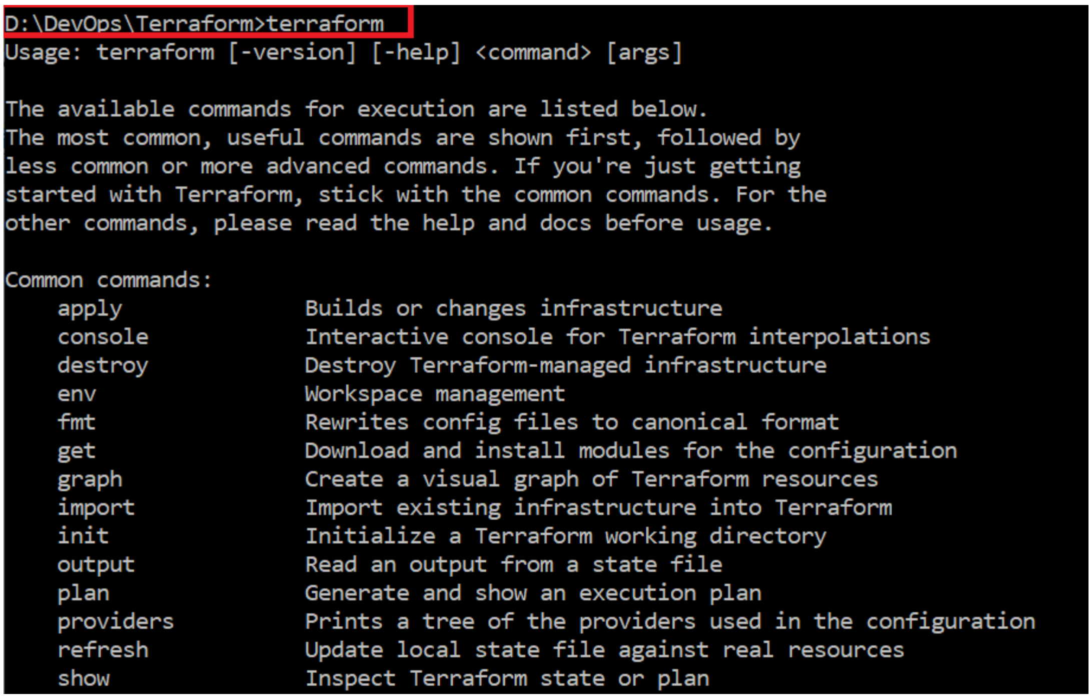
```
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files\PuTTY\
C:\Program Files\Java\jdk1.8.0_211\bin
C:\Program Files\Amazon\AWSCLI\bin\
C:\Program Files (x86)\Bitvise SSH Client
C:\Program Files\OpenSSH-Win64
%SYSTEMROOT%\System32\OpenSSH\
D:\DevOps\DevOps\apache-maven-3.6.2\BIN
C:\Program Files\Git\cmd
D:\DevOps\Teraform\terraform
D:\DevOps\Teraform\
```

Steps 5: To Verify the installation , Open the command prompt and cd to install folder



```
D:\DevOps\Terraform>
```

Steps 6: Now type terraform

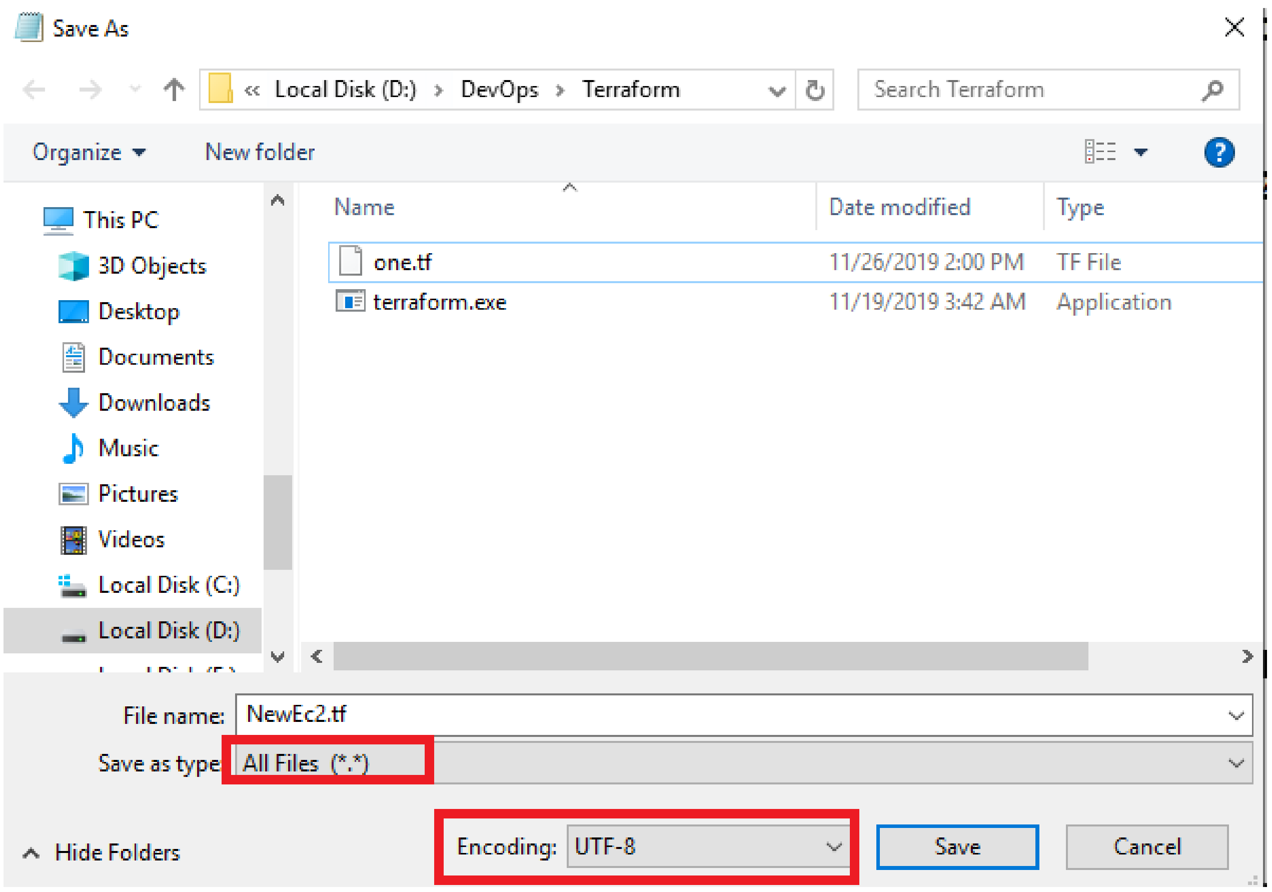


```
D:\DevOps\Terraform>terraform
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a Terraform working directory
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
```

4 Creating your first Terraform infrastructure on AWS

Step1: create a file called **newec2.tf** with the following code and save in **UTF-8** Format in same directory .

```
# Create a new instance of the latest Ubuntu 14.04 on an

resource "aws_instance" "web" {
  ami           = "ami-00eb20669e0990cb4"
  instance_type = "t2.micro"
  availability_zone = "us-east-1c"
}
```

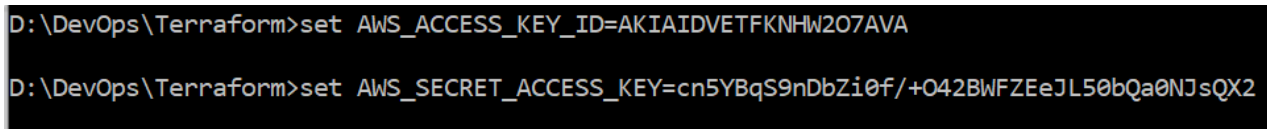


Name	Date modified	Type
one.tf	11/26/2019 2:00 PM	TF File
terraform.exe	11/19/2019 3:42 AM	Application

File name: NewEc2.tf
Save as type: All Files (*.*)
Encoding: UTF-8

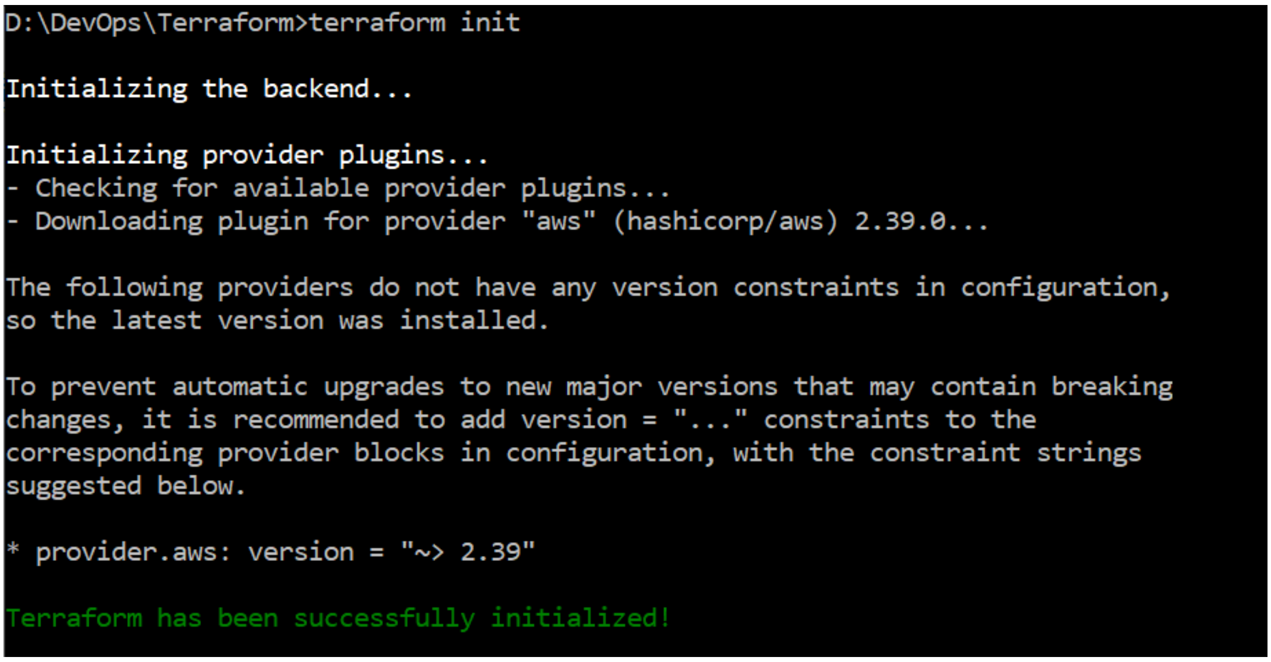
Step2: Now on command prompt use below envio9rnment variable

set AWS_ACCESS_KEY_ID=Your Access Key
set AWS_SECRET_ACCESS_KEY=Your Secret Key



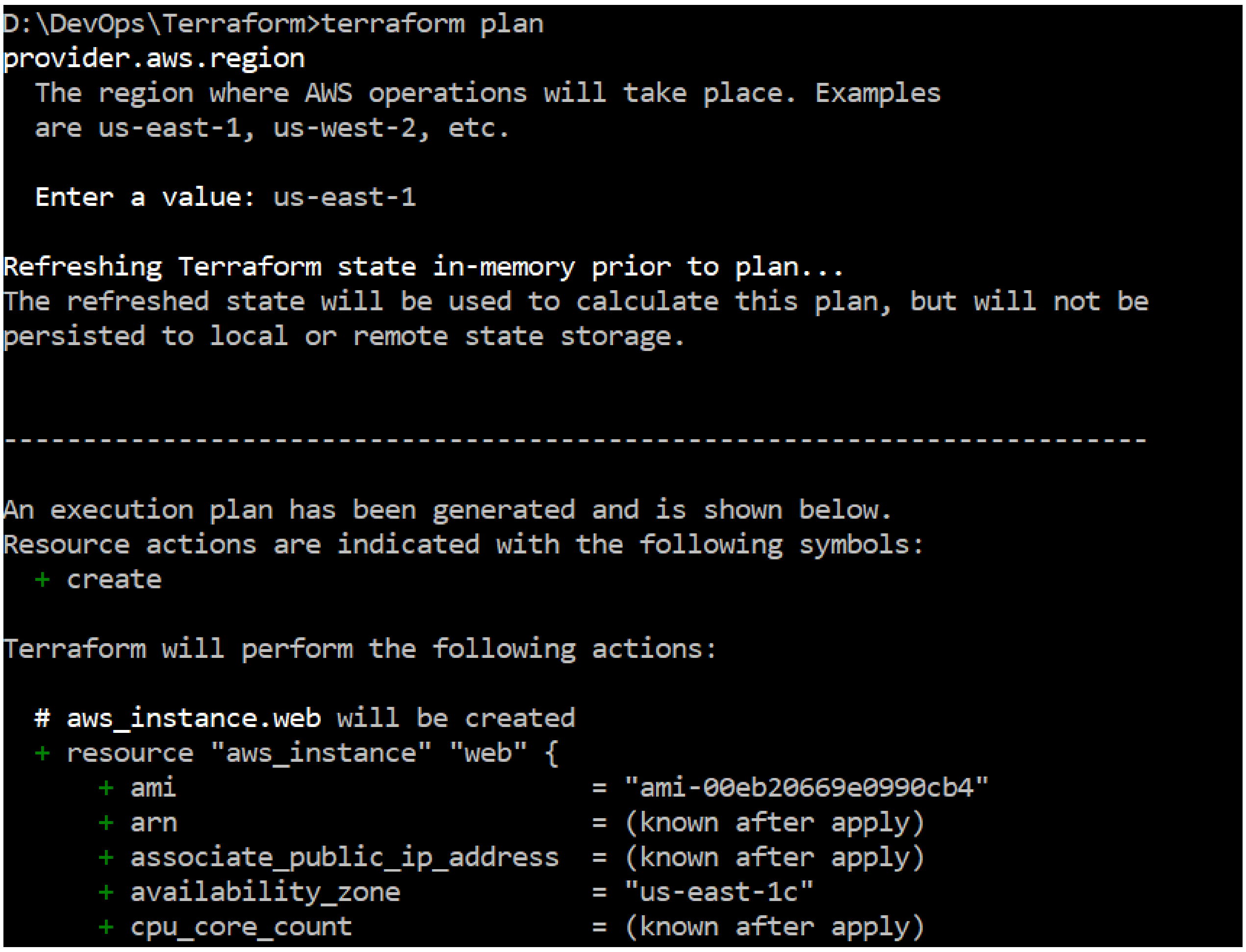
```
D:\DevOps\Terraform>set AWS_ACCESS_KEY_ID=AKIAIDVETFKNHW2O7AVA

D:\DevOps\Terraform>set AWS_SECRET_ACCESS_KEY=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
```

Step3: Now we will run the **"terraform init"** command where we created our instance.tf file to download and initialize the appropriate provider plugins. In this case, we are downloading the **AWS provider plugin** we specified in our **newec2.tf** file.



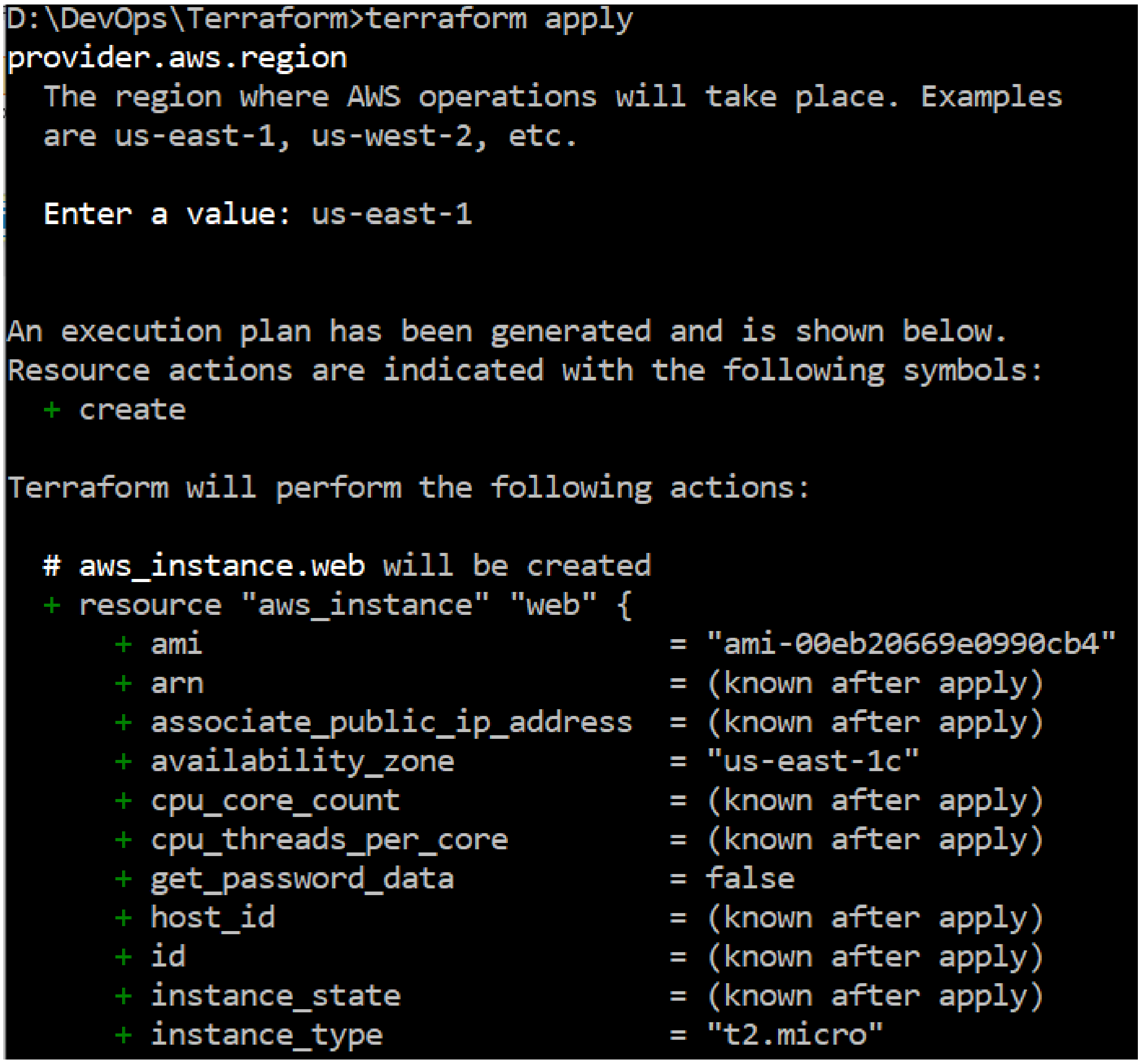
```
D:\DevOps\Terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 2.39.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.39"

Terraform has been successfully initialized!
```

Step3: Now we will run the **"terraform Plan"**



```
D:\DevOps\Terraform>terraform plan
provider.aws.region
  The region where AWS operations will take place. Examples
  are us-east-1, us-west-2, etc.

  Enter a value: us-east-1

Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.web will be created
  + resource "aws_instance" "web" {
      + ami                          = "ami-00eb20669e0990cb4"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)
      + availability_zone            = "us-east-1c"
      + cpu_core_count               = (known after apply)
```

Step4: Now we will run the **"terraform apply"**



```
D:\DevOps\Terraform>terraform apply
provider.aws.region
  The region where AWS operations will take place. Examples
  are us-east-1, us-west-2, etc.

  Enter a value: us-east-1


An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.web will be created
  + resource "aws_instance" "web" {
      + ami                          = "ami-00eb20669e0990cb4"
      + arn                          = (known after apply)
      + associate_public_ip_address  = (known after apply)
      + availability_zone            = "us-east-1c"
      + cpu_core_count               = (known after apply)
      + cpu_threads_per_core         = (known after apply)
      + get_password_data            = false
      + host_id                      = (known after apply)
      + id                           = (known after apply)
      + instance_state               = (known after apply)
      + instance_type                = "t2.micro"
```

Step4: Now you can verify it on aws console

5 Attaching the New Volume to just created intsance

Step1: Open the same file and put below code to add 1 GB HDD to instance just created . and save it.

```
resource "aws_instance" "web" {
  ami          = "ami-00eb20669e0990cb4"
  instance_type = "t2.micro"
  availability_zone = "us-east-1c"
}

resource "aws_volume_attachment" "ebs_att" {
  device_name = "/dev/sdh"
  volume_id   = "${aws_ebs_volume.example.id}"
  instance_id = "${aws_instance.web.id}"
}
resource "aws_ebs_volume" "example" {
  availability_zone = "us-east-1c"
  size            = 1
}
```

Step2: Now Run Below command

Terraform Plan

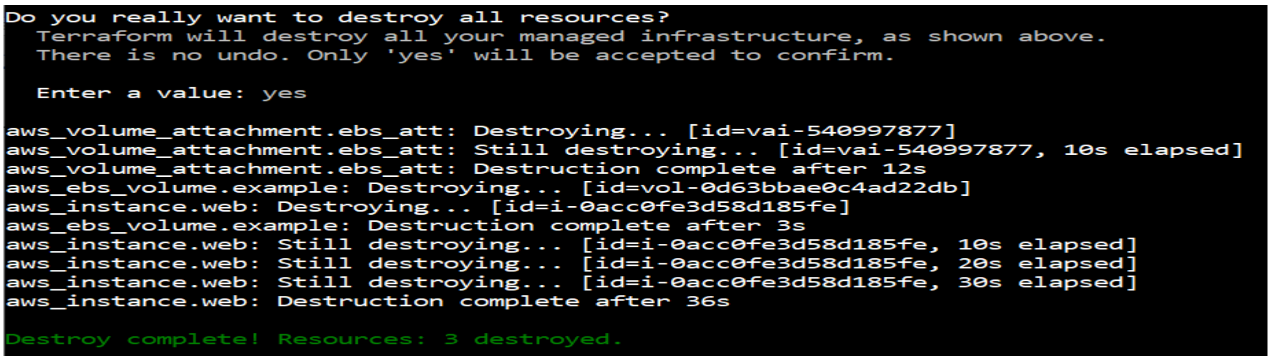
Terraform Apply

Step3: Check on Portal

6 Destroy the Instance

Step1: Run a Below command destroy all the configuration

Terraform 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_volume_attachment.ebs_att: Destroying... [id=vai-540997877]
aws_volume_attachment.ebs_att: Still destroying... [id=vai-540997877, 10s elapsed]
aws_volume_attachment.ebs_att: Destruction complete after 12s
aws_ebs_volume.example: Destroying... [id=vol-0d63bbae0c4ad22db]
aws_instance.web: Destroying... [id=i-0acc0fe3d58d185fe]
aws_ebs_volume.example: Destruction complete after 3s
aws_instance.web: Still destroying... [id=i-0acc0fe3d58d185fe, 10s elapsed]
aws_instance.web: Still destroying... [id=i-0acc0fe3d58d185fe, 20s elapsed]
aws_instance.web: Still destroying... [id=i-0acc0fe3d58d185fe, 30s elapsed]
aws_instance.web: Destruction complete after 36s

Destroy complete! Resources: 3 destroyed.
```