=> Terraform :-

--> Terraform is free and open source tool/sw developed by Hashicorp

--> To Create/provision infrastructure in cloud platform --> Supports all most all the cloud platforms

(Infrastructure as code) IAC

--> Terraform will use HCL --> Hashicorp configuration language

Terraform Installation
==========

https://developer.hashicorp.com/terraform/install?product_intent=terraform

1) Created Linux VM in AWS Cloud (Ami : Amazon Linux)

2) Connect with Linux VM using (MobaXterm/Gitbash)

3)Execute the below commands to setup Terraform in Linux VM

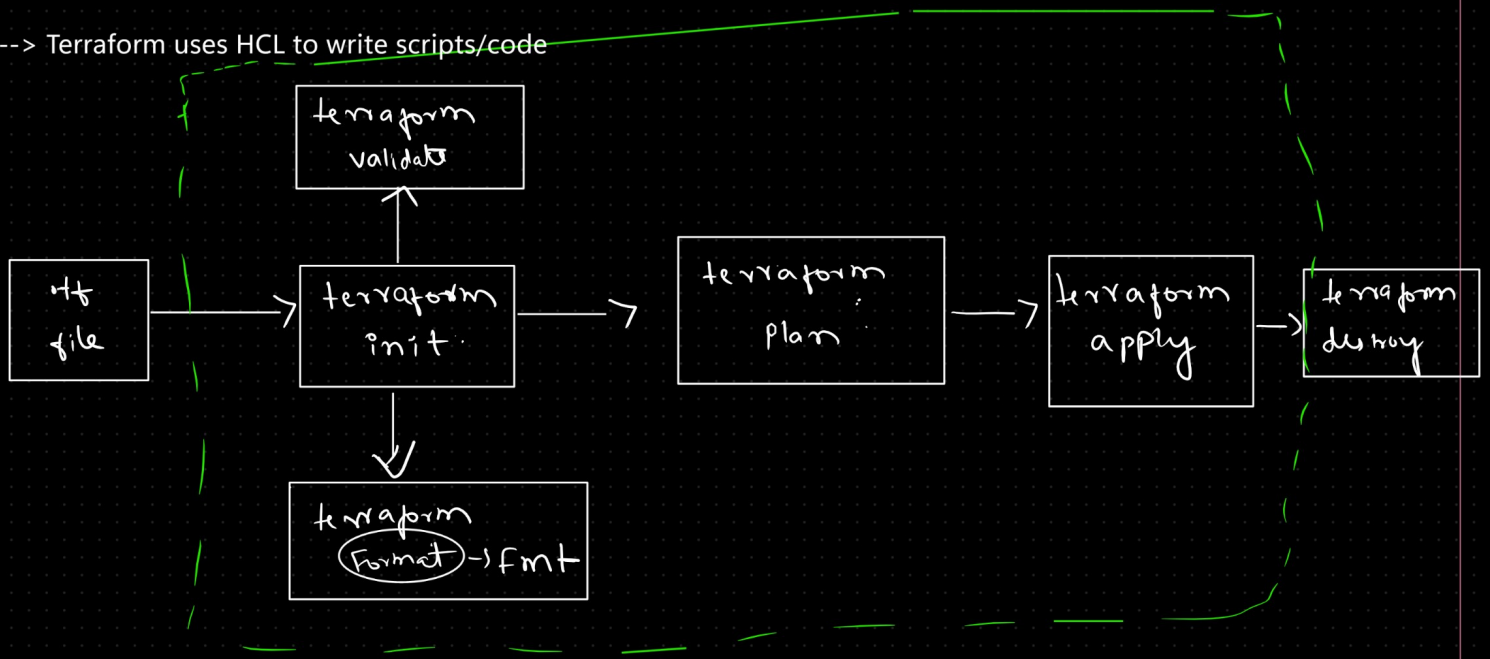
```
sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
```

4) Verify Terraform installation

```
terraform -v
terraform version
```

Terraform Architecture
==========

--> Terraform uses HCL to write scripts/code



https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-build

--> Terraform script we use .tf extension

.tf-> init-->fmt-->validate--plan--apply--destroy

terraform init--> Initialize terraform script(.tf file)

terraform fmt --> Formate terraform script indent spacing(optional)

terraform validate --> Verify terraform script syntax is valid or not (optional)

terraform plan: Create Execution plan for terraform script

terraform apply : Create actual resource in cloud based on given plan

terraform destroy: delete the resources created with our terraform script

==========
Terraform Script To create EC2 Instance
==========

```
provider "aws" {
  region ="ap-south-1"
  access_key = "AKIASDRANAWRNJJ7CZXR"
  secret_key = "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"
}

resource "aws_instance"  "linux-vm" {
          ami = "ami-002f6e91abff6eb96"
          instance_type = "t2.micro"
          key_name = "terraform"
          security_groups = ["default"]
          tags = {
                    Name ="Telusko-Linux_VM"
          }

}
```

Dealing with Access Key and Secret Key
==========
-->Instead of configuring access key and secret in terraform script file we can configure them in environment variables

$export AWS_ACCESS_KEY_ID="AKIASDRANAWRNJJ7CZXR"

$export AWS_SECRET_ACCESS_KEY="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"

$ echo $AWS_ACCESS_KEY_ID

$ echo $AWS_SECRET_ACCESS_KEY

```
provider "aws" {
  region ="ap-south-1"
}

resource "aws_instance"  "linux-vm" {
        ami = "ami-002f6e91abff6eb96"
        instance_type = "t2.micro"
        key_name = "terraform"
        security_groups = ["default"]
        tags = {
                Name ="Telusko-Linux_VM"
        }

} terraform init, terraform fmt, terraform validate, terraform plan, terraform apply --> yes


Creating EC2 VM with User Data
======================

Add access key and secret key in environment

Create  a new dir : mkdir 01-tf-script-userdata

Create a script file

$ vi installHttpd.sh

#! /bin/bash

sudo su

yum install httpd -y
cd /var/www/html
echo "<html><h1> Welcome to Telusko WebServer </h1></html>" > index.html
service httpd start

//provide the execute permission for script file

$ chmod u+x installHttpd.sh

create resource file

provider "aws" {
  region ="ap-south-1"
}

resource "aws_instance"  "linux-vm" {
        ami = "ami-002f6e91abff6eb96"
        instance_type = "t2.micro"
        key_name = "terraform"
        security_groups = ["default"]
        user_data= file("installHttpd.sh")
        tags = {
                Name ="DevOps-Linux_VM"
        }

}  terraform init, terraform fmt, terraform validate, terraform plan, terraform apply --auto-approve  -->
```

Variables in Terraform
==========

==> Variables are used to store data in key-value formate

name =telusko

-->Types of variables

Input Variable --> supply values to terraform script

We can remove hard coded values from resources script

Output variable --> get the values from terraform script after execution

examples : After EC2 VM created --> print ec2-vm public ip .....
After IAM user got created print IAM user info

After S3 bucket got created , print bucket info

so on--------

==> Variables we can maintain in separate tf file

created a new dir ==> mkdir 03-tf-script-vars

cd mkdir 03-tf-script-vars

created separate tf file for vars

```
$ vi vars.tf
variable "ami" {
     description = "Amazon vm image value"
     default = "ami-002f6e91abff6eb96"
}

variable "instance_type" {
      description = "Represents the type of instance"
      default = "t2.micro"
}

$ vi main.tf

resource "aws_instance" "linux-vm" {
  ami             = var.ami
  instance_type   = var.instance_type
  key_name        = "terraform"
  security_groups = ["default"]
  tags = {
    Name = "var-Linux_VM"
  }
}
$ vi provider.tf

provider "aws" {
  region = "ap-south-1"
}
```

terraform init, terraform fmt, terraform validate , terraform plan, terraform apply --auto-approve

terraform destroy

Terraform Script with Input & Output Variables
==========
Create new directory

Cd to new dir

Create provider .tf

$ vi provider.tf

```
provider "aws" {
  region = "ap-south-1"
}
```

Create input-vars.tf

$ vi input-vars.tf

```
variable "ami" {
      description = "Amazon vm image value"
      default = "ami-002f6e91abff6eb96"
}

variable "instance_type" {
       description = "Represents the type of instance"
       default = "t2.micro"
}
```

Create main resource

vi main.tf

```
resource "aws_instance" "linux_vm" {
  ami             = var.ami
  instance_type   = var.instance_type
  key_name        = "terraform"
  security_groups = ["default"]
  tags = {
    Name = "var-Linux_VM"
  }
}
```

Create outfile
 $ vi output.tf

```
output "ev2_vm_info" {
   value= aws_instance.linux_vm
}
output "ev2_vm_public_ip" {
   value= aws_instance.linux_vm.public_i
```

```
}
```

Creating S3 Bucket

==========

```
resource "aws_s3_bucket" "telusko_bucket" {
  bucket = "telusko4343"  # Ensure this is globally unique
  acl    = "private"  # Set the access control list (ACL) for the bucket

versioning {
    enabled = true  # Enable versioning on the bucket
  }

}
```

Creating IAM User

==========

```
# Create an IAM user
resource "aws_iam_user" "example_user" {
  name = "my-example-user"  # Name of the IAM user
}

# Optionally, you can attach a policy to the user
resource "aws_iam_user_policy_attachment" "example_user_policy" {
  user       = aws_iam_user.example_user.name
  policy_arn = "arn:aws:iam::aws:policy/AdministratorAccess"  # Replace with your desired policy
}
```

Create RDS instance and print DB instance endpoint url using Terraform