++++++++++++++++++
Terraform (IAAC s/w)
++++++++++++++++++

-> Terraform is an open source s/w created by HashiCorp and written in Go programming language

-> Terraform is an infrastructure as code (IaaC) software tool,

-> Infrastructure as code is the process of managing infrastructure in a file or files rather than manually configuring resources using user interface (UI)

-> In Terraform resources are nothing but Virtual machines, Elastic IPs, Security Groups, Network interfaces, RDS, LBR etc..

-> Terraform code is written in the HashiCorp Configuration langauge (HCL) in files with the extension .tf

-> Terraform allows users to use HashiCorp Configuration Language (HCL) to create the files containing definitions of the their desired resources

-> Terraform Supports all most all cloud providers (AWS, AZURE, GCP, Openstack etc..)

-> To automate infrastructure creation in cloud platforms we will use Terraform.

++++++++++++++++++++++++++
Terraform vs Cloud Formation
++++++++++++++++++++++++++++

-> Terraform developed by HashiCorp
-> CloudFormation developed by AWS

-> Terraform supports many cloud providers
-> Cloud Formation willl support only in AWS

-> Terraform uses HashiCorp configuration language (HCL) which built by HashiCorp. It is fully compatible with JSON.

-> AWS Cloud Formation utilizes either JSON or YAML. Cloud formation has a limit of 51,000 bytes for the template body itself.

++++++++++++++++++
Terraform Vs Ansible
++++++++++++++++++++
-> Terraform developed by HashiCorp
-> Ansible is also an open source software

-> Terraform is an infrastructure as a Code, which means they are designed to provision the servers themselves.
-> Ansible is a configuration management tool. Which means ansibled designed o install and manage software on existing servers.

-> Terraform is ideal for creating, managing and improving infrastructure.
-> Ansible is ideal for software provisioning, application deployment and configuration management.

Pre-Requisites
+++++++++++++++

1) Cloud Platform Account (AWS, Azure, GCP, Openstack etc..)
2) IAM User account (Secret Key and Access Key)
3) IAM User should have resources Access

################################

```
Terraform Installation
##############################

1) Create EC2 instance (RED HAT Linux)

2) Connect to EC2 VM using Mobaxterm

3) Swith to root user
        $ sudo su -

4) Install unzip software
        $ sudo yum install wget unzip vim -y

5) Download Terraform Software (https://www.terraform.io/downloads)

$ sudo yum install -y yum-utils
$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
$ sudo yum -y install terraform

6) Check Terraform Version

$ terraform -v


########################################
Working with EC2 Instance using Terraform
########################################


1) Create IAM user with Programmatic Access   (IAM user should have EC2FullAccess)

2) Download Secret Key and Access Key

3) Write First Terraform Script

$ mkdir terraformscript
$ cd terraformscripts
$ vi FirstTFScript.tf

provider "aws" {
   region = "ap-south-1"
   access_key = "AKIA4MGQ5UW757KVKECC"
   secret_key = "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"
}

resource "aws_instance" "AWSServer" {
  ami = "ami-08df646e18b182346"
  instance_type = "t2.micro"
  key_name = "linux"
  security_groups = ["ashokit_security_group"]
  tags = {
   Name = "MyEC2-VM"
  }
}

10) Initialize Terraform using init command

        $ terraform init

11) Format your script (indent spaces)

        $ terraform fmt

12) Validate Your Script
```

```
        $ terraform validate

13) Create Execution Plan For Your Script

        $ terraform plan

14) Create Infrastructure

        $ terraform apply

Note: When the script got executed it will store that state in a file. If we execute script again it
will not create. If you delete that state file and execute script again then it will create it.

15) Destory Infrastructure

        $ terraform destroy -auto-approve


-> In first script we kept provider and resources info in single script file. We can keep provider
and resources information in seperate files

Ex : proder.tf & main.tf


###################################
Script to create multiple Ec2 instances
###################################

provider "aws" {
   region = "ap-south-1"
   access_key = "AKIA4MGQ5UW757KVKECC"
   secret_key = "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"
}

resource "aws_instance" "AWSVM_Server" {
  count           = "2"
  ami             = "ami-05c8ca4485f8b138a"
  instance_type   = "t2.micro"
  key_name        = "linux"
  security_groups = ["ashokit_security_group"]
  tags = {
    Name = "REDHAT-EC2-VM"
  }
}


Note: Once it is created, then destory infrastructure using below command

$ terraform destroy -auto-approve

++++++++++++++++++++++++++
Variables in TypeScript
++++++++++++++++++++++

-> We can maintain variables in seperate file

$ vi vars.tf

variable "ami"{
 description="Amazon Machine Image value"
 default = "ami-05c8ca4485f8b138a"
}

variable "instance_type"{
   description="Amazon Instance Type"
```

```
    default = "t2.micro"
}

variable "instances_count"{
        description="Total No.of Instances"
        default = "2"
}
```

-> Create main tf file using variables

$ vi main.tf

```
provider "aws" {
  region = "ap-south-1"
   access_key = "AKIA4MGQ5UW757KVKECC"
   secret_key = "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"
}

resource "aws_instance" "AWSServer" {
  count="${var.instances_count}"
  ami = "${var.ami}"
  instance_type = "${var.instance_type}"
  key_name = "linux"
  security_groups = ["ashokit_security_group"]
  tags = {
   Name = "EC2 VM - ${count.index}"
  }
}
```

Note: We can supply variables in runtime also

-> Remove instances_count variable from var.tf file and pass like below

$ terraform apply -var instances_count="2" -auto-aprove

++++++++++++++++++++++++++++
Comments in Terraform Script
++++++++++++++++++++++++++++

- single line comment

// - single line comment (java style)

/* and */ - Multi line comments

Dealing with Secret Key and Access Key
++++++++++++++++++++++++++++++++++++

-> We have configure secret_key and access_key in terraform script file. Instead of that we can configure them as environment variables.

$ export AWS_ACCESS_KEY_ID="AKIA4MGQ5UW7UPMSQKXK"
$ export AWS_SECRET_ACCESS_KEY="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"

-> To verify environment variables we can use echo command

$ echo $AWS_ACCESS_KEY_ID
$ echo $AWS_SECRET_ACCESS_KEY

-> Now remove credentials from terraform script and execute it.

Note: We are setting provider credentials in terminal so these variables will be available for current session. If we want to set permanently add them in .bashrc file

++++++++++++++++++++++++
Working with User Data
++++++++++++++++++++++

-> It is used to execute script when instance launched for first time.

-> Create Userdata in one file

$ vi installHttpd.sh

```
#!/bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT...!!</h1></html>" > index.html
service httpd start
```

$ chmod u+x installHttpd.sh

-> create main scrit in main.tf file

-> vi main.tf

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

resource "aws_instance" "AWSServer" {
  ami = "ami-05c8ca4485f8b138a"
  instance_type = "t2.micro"
  key_name = "linux"
  security_groups = ["ashokit_security_group"]
  user_data = "${file("installHttpd.sh")}"
  tags = {
   Name = "Web-Server"
  }
}
```

++++++++++++++++++++++++++++++++
Creating S3 bucket using Terraform script
++++++++++++++++++++++++++++++++

-> Add S3 policy for IAM user

-> Execute below terraform script to create s3 bucket in AWS

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

resource "aws_s3_bucket" "s3bucketashokit"{

bucket = "s3bucketashokit"
acl="private"

versioning{
        enabled = true
}
```

```
  tags = {
    Name = "S3 Bucket By Ashok"
  }
}



++++++++++++++++++++++++++++++++++
Create MySQL DB in AWS using Terraform
++++++++++++++++++++++++++++++++++

-> Provider RDS access for IAM user

-> Execute below script to create MySQL DB in AWS cloud

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

resource "aws_db_instance" "default" {
  allocated_storage    = 100
  engine                     = "mysql"
  engine_version        = "5.7"
  instance_class        = "db.t3.micro"
  name                         = "mydb"
  username                = "foo"
  password                = "foobarbaz"
  parameter_group_name = "default.mysql5.7"
  skip_final_snapshot  = true
}
```