Terraform Concepts & Working

Make some folders and place your code.

```
$ mkdir /usr/local/terraform-demo
$ cd /usr/local/terraform-demo
$ mkdir demo1
$ cd demo1
```

Create a tf file within demo1

```
$ vim example1.tf
```

For AMI visit link - https://aws.amazon.com/amazon-linux-ami/

```
# Example 1 - First EC2 instance

provider "aws" {
 region = "ap-south-1"
 access_key = "<Access-Key>"
 secret_key = "<Secret-Key>"
}

resource "aws_instance" "instance01" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
}
```

// Save the above content in file and follow below commands

```
$ terraform init
$ terraform apply
```

How to write comment in terraform

```
# Single line comment

//Single line comment

/*
Block comment
*/
```

Modify existing file and write below example -

# Example 2 - AWS Authentication using shared credentials file

(NOTE: Use aws configure before to add the AWS credentials to .aws/credentials file)

```
provider "aws" {
profile = "ritesh-devops"
region = "ap-south-1"
}
resource "aws_instance" "instance01" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
 tags = {
    "Name"        = "web-server"
    "environment" = "dev"
  }
}
resource "aws_instance" "instance02" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
 tags = {
    "Name"        = "appserver"
    "environment" = "stage"
  }
 }
```

$ terraform plan
$ terraform apply

# Example 3 - Change in the infrastructure

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

resource "aws_instance" "instance01" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
 tags = {
    "Name"        = "web-server"
    "environment" = "dev"
  }
}
resource "aws_eip" "newIP" {
 instance = "${aws_instance.instance01.id}"
 vpc = true
}
```

$ terraform plan

$ terraform apply

# Example 4 - Destroy the infrastructure

```
$ terraform show
$ terraform destroy
```

# Example 5 - Resource Dependency // Implicit & Explicit

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

resource "aws_instance" "instance01" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
 tags = {
    "Name"        = "web-server"
    "environment" = "dev"
 }
depends_on = [aws_ebs_volume.diskSize]
}

resource "aws_ebs_volume" "diskSize" {
 availability_zone = "ap-south-1a"
 size = 10
}

resource "aws_volume_attachment" "ebs_add" {
  device_name = "/dev/xvdf"
  volume_id   = aws_ebs_volume.diskSize.id
  instance_id = aws_instance.instance01.id
}

resource "aws_eip" "newIP" {
 instance = aws_instance.instance01.id
 vpc = true
}
```

$ terraform apply

# Example 6 - Provision local/external

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

resource "aws_instance" "instance01" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
 tags = {
    "Name"        = "web-server"
    "environment" = "dev"
 }
  provisioner "local-exec" {
    command = "echo ${aws_instance.instance01.public_ip} > ip_address.txt"
 }

}
```

$ terraform apply

--

# Example 7 - Defining Variable - Input / Output Variable

```
variable "region" {
  default = "ap-south-1"
}

provider "aws" {
 profile = "ritesh-devops"
 region = var.region
}

resource "aws_instance" "instance01" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
 tags = {
    "Name"        = "web-server"
    "environment" = "dev"
  }
}

output "ip" {
  value = aws_instance.instance01.public_ip

}
```

--

#Example 8 - Splitting the input, output and provider in different files

----

vim example.tf

```
resource "aws_instance" "instance01" {
 ami = "ami-04db49c0fb2215364"
 instance_type = "t2.micro"
 tags = {
    "Name"        = "web-server"
    "environment" = "dev"
 }
}
```

----

vim variables.tf

```
variable "region" {
  default = "ap-south-1"
}
```

----

vim outputs.tf

```
output "ip" {
  value = aws_instance.instance01.public_ip
}
```

----

vim provider.tf

```
provider "aws" {
  profile = "ritesh-devops"
  region = var.region
}
```

--

#Example 9 - Backend configuration

Each Terraform configuration can specify a backend, which defines where state snapshots are stored.

```
terraform {
  backend "s3" {
    bucket = "core-infrastructure-devops-tfstate"
    key    = "devops/terraform.tfstate"
    region = "ap-south-1"
  }
```

}
Module compute
Module VPC
Module security

Name: Ritesh Goyal
Contact: 9960930111
Email-id: ritesh.devopstrainer@gmail.com