Basics of Terraform

main.tf file

```
#define provider and authentication
provider "aws" {
  region     = "us-west-2"
  access_key = "my-access-key"
  secret_key = "my-secret-key"
}
#How to create resources inside a provider
# resource "<provider>_<resource_type>" "name" {
#     config options ...
#     key = value
#     key = value

# }

#create  ec2 instance
resource "aws_instance" "first-ec2instance" {
    ami = "ami-0d75513e7706cf2d9"
    instance_type = "t2.micro"

}
```

terraform running terminal command

1. Now run > terraform init (from terminal)

2.
```
terraform plan (optional to get what you are going to do with this tf config file)
```

3.
```
terraform apply (terraform apply --auto-approve if you dont want to type yes)
```

4.
```
terraform destroy (will destroy the created resources. For single destroy you
need to add parameters in the above command)
Or an alternative could be to remove that resource code and run terraform apply(declarative approach)
```

Your ami value is different for different regions, so fetch ami value after selecting the region you mentioned in the provider dictionary.

.terraform folder

is created by the terraform init command to install the required packages for the mentioned provider in the tf file.

terraform.tfstate

This file is critical for the proper fucntioning as it keeps track of the current state of all the variable associated with our resources.

terraform variables

Definition of a varible inside .tf file

```
variable "nameofvariable"{
description = "repres cidr block" (Optional)
default = "" (Optional)
type = string(optional)
}
```

Variable as list

Assigning value to a varin .tfvars file:

```
subnet = ['10.0.0.1/24','10.0.1.0/24']
```

Access this variable in .tf file:

```
cidr_block = subnet[0]
```

Variable as an object

Assign value to a variable in .tfvars file

```
subnet = [
{
cidr_bloc='10.0.0.0/24',
name = "production"
},
{
cidr_block = value
name = value
}]
```

Access these values

```
cidr_block = var.subnet[0].cidr_block
name = var.subnet[1].name
```

Passing value to a varialbe

3 ways:

1. When we apply(terraform apply), the terraform will ask us to enter the value through command line.

2.
   ```
   terraform apply -var "varname = value"
   ```
3. Best way is to create a separate file for variable assignment. Create a file called "terraform.tfvars". Inside this file give value to all the variable. nameofvariable = value
4. Through default = value in the definition of the variable.

How to reference variable

cidr_block = var.nameofvariable

Further commands

1.
   ```
   terraform state list
   ```

2.
   ```
   terraform state show resourcename
   ```
3. Resource name in the above command will come from the first command. It will display the ocmplete information regarding that resource
4. __> output "this is the print statement"{ value = resource.resourcename.property}
5. Insted of seeing the desired variable values through the second command. We can simply print them out through the 4th command.

6.
   ```
   terraform output
   ```

7.
   ```
   terraform refresh (to get all the states without applying them)
   ```

8.
   ```
   terraform apply target resource.resourcename (for staged
   deployement only this resource will be deployed)
   ```

Complete project

```
# define provider
provider "aws" {
  region = "us-east-1"
  access_key = "yourkey" #create from security credentials
  secret_key = "yourkey"
}
```

```
#How to create resources inside a provider
# resource "<provider>_<resource_type>" "name" {
#     config options ...
#     key = value
#     key = value

# }

# define ec2 instance
# Go to terraform documentation of aws then go to
# ec2 → resources→aws_instance

# resource "aws_instance" "terraformec2" {
#   ami           = "ami-052efd3df9dad4825"   #different regions have different ami
values. ec2 inst ami value
#   instance_type = "t2.micro"
#   tags = {
#     "Name" = "ubuntu"
#   }
# }

#  if you comment out the above resource code, it will destroy the created instance
'terraformec2'
#  on running terraform apply command

# define VPC in aws
# resource "aws_vpc" "vpc-1" {
#   cidr_block = "10.0.0.0/16"
#   tags = {
#     "Name" = "production"
#   }
# }

# define subnet
# resource "aws_subnet" "subnet-1" {
#   vpc_id = aws_vpc.vpc-1.id       #reference other
resource..resource.resourcename.property
#   cidr_block = "10.0.1.0/24"

#   tags = {
#     "Name" = "prod-subnet"
#   }
# }


#Complete tutorial
# create keypair of ec2 instance

#1. Create vpc network
resource "aws_vpc" "vpc-prod" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "production"
  }
}

# 2. create internet gateway
resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.vpc-prod.id

  tags = {
    Name = "production"
  }
}
# 3. Create route table
resource "aws_route_table" "prod-rtable" {
  vpc_id = aws_vpc.vpc-prod.id

  route {
    cidr_block = "0.0.0.0/0"  #send all traffic to gatewway id
    gateway_id = aws_internet_gateway.gw.id
  }

  route {
    ipv6_cidr_block        = "::/0"
    gateway_id = aws_internet_gateway.gw.id
  }

  tags = {
    Name = "production"
  }
}
# 4. Create subnet
resource "aws_subnet" "prod-subnet" {
  vpc_id = aws_vpc.vpc-prod.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "us-east-1a"
  tags = {
    "Name" = "production-subnet"
  }

}

# 5. Associate subnet with route table
resource "aws_route_table_association" "prod-rt-association" {
  subnet_id = aws_subnet.prod-subnet.id
  route_table_id = aws_route_table.prod-rtable.id

}
# 6. create security group
resource "aws_security_group" "allow_webtraffic" {
  name        = "allow_web_traffic"
  description = "Allow Web traffic traffic"
  vpc_id      = aws_vpc.vpc-prod.id

  ingress {
    description      = "HTTPS"
    from_port        = 443
    to_port          = 443
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"] #any ip address we can also give a specific ip
address
  }
  ingress {
    description      = "HTTP"
    from_port        = 80
    to_port          = 80
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"] #any ip address we can also give a specific ip
address
  }
  ingress {
    description      = "SSH"
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"] #any ip address we can also give a specific ip
address
  }
  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1" #any protocl
    cidr_blocks      = ["0.0.0.0/0"]
    ipv6_cidr_blocks = ["::/0"]
  }

  tags = {
    Name = "allow_web"
  }
}
# 7. network interface
resource "aws_network_interface" "prod-network-interface" {
  subnet_id       = aws_subnet.prod-subnet.id
  private_ips     = ["10.0.1.50"] # choose ip for server, we can give a list of ips
  security_groups = [aws_security_group.allow_webtraffic.id]

}
# 8. eip for public access
resource "aws_eip" "prod-eip-one" {
  vpc                       = true
  network_interface         = aws_network_interface.prod-network-interface.id
  associate_with_private_ip = "10.0.1.50"
  depends_on = [aws_internet_gateway.gw]
}

# 9. create ubuntu server and install apache
resource "aws_instance" "web-ubuntu-server" {
  ami = "ami-052efd3df9dad4825"
  instance_type = "t2.micro"
  availability_zone = "us-east-1a"
  key_name = "main-key"
  network_interface {
    device_index = 0
    network_interface_id = aws_network_interface.prod-network-interface.id

  }
  user_data = <<-EOF
              #!/bin/bash
              sudo apt update -y
              sudo apt install apache2 -y
              sudo systemctl start apache2
              sudo bash -c 'echo your first web server >/var/www/html/index.html'
              EOF
  tags = {
    "Name" = "webserver"
  }

}
```