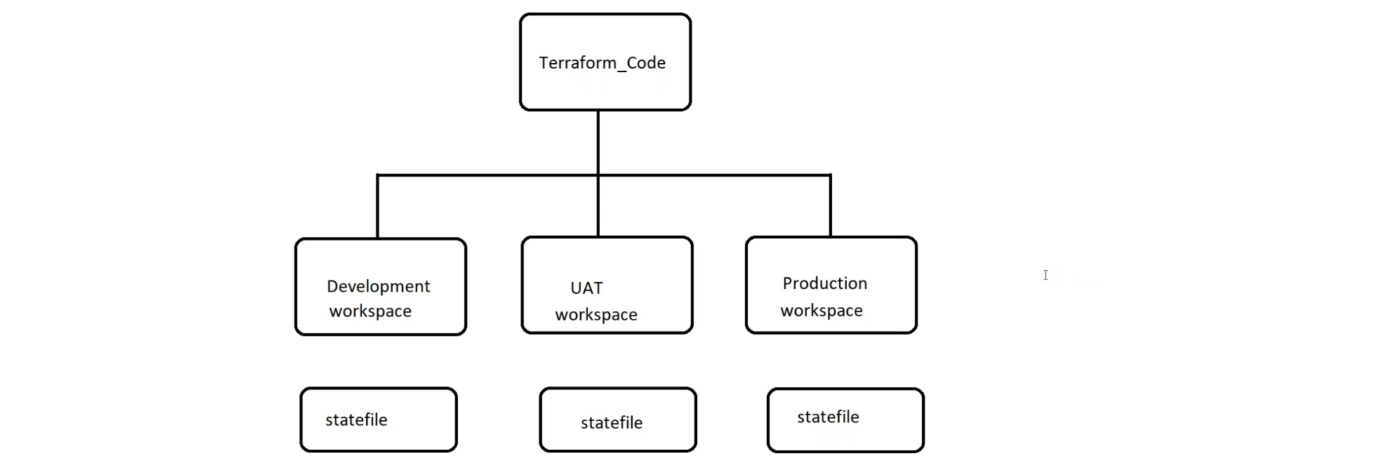
**18.DevOps-B24-Terraform-part-4**

--- **note** – in this session, we will talk about workspace, before talk about this workspace. We need to talk use case. So far, we have created single environment like test environment. We do not want that we have a requirement to create 3 environments like **dev(3-servers)**, **UAT(3-servers)**, **Prod(3-servers)**. For this kind of scenarios, we will use workspaces.

**Terraform workspace**



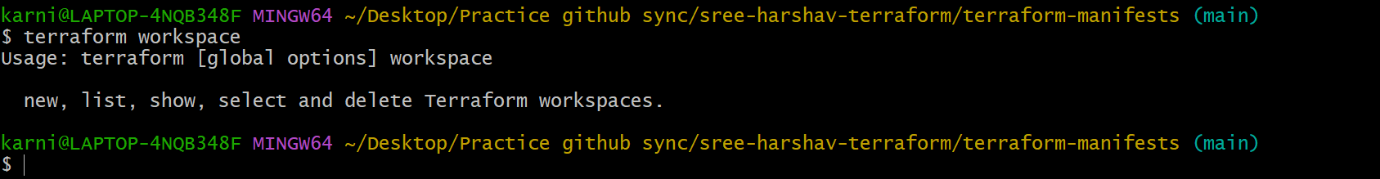
--- **note** – in terraform workspace, each environment has its own state file. Here, iam I will create 3 workspaces like dev, uat, prod. This way I will maintain separate environment for each environment.

**When will we use terraform workspace?**

--- you have a same code for all the environments and needs to give different variables.

--- all the environmental code is identical like, if you have 3 servers in dev and same servers, you should have in uat and prod.

--- **terraform workspace**



--- **note** – when you execute above command, it will give you the options like new, list, show, select and delete.

**# Create new workspace**

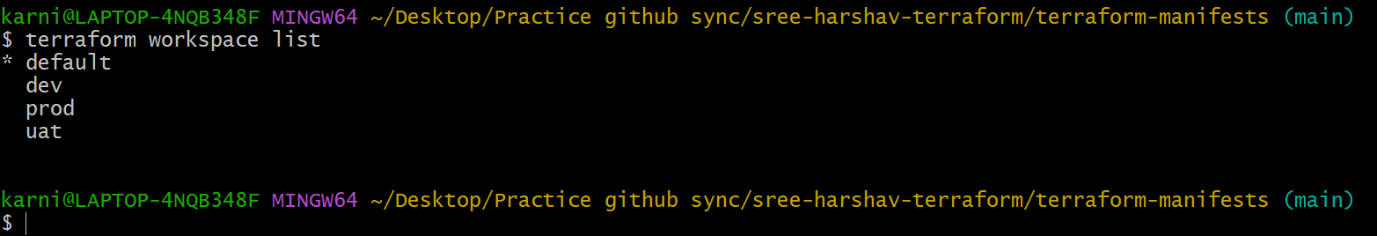
--- terraform workspace new dev

--- terraform workspace new uat

--- terraform workspace new prod

**# List the workspace**

--- terraform workspace list



**# Switch workspace**

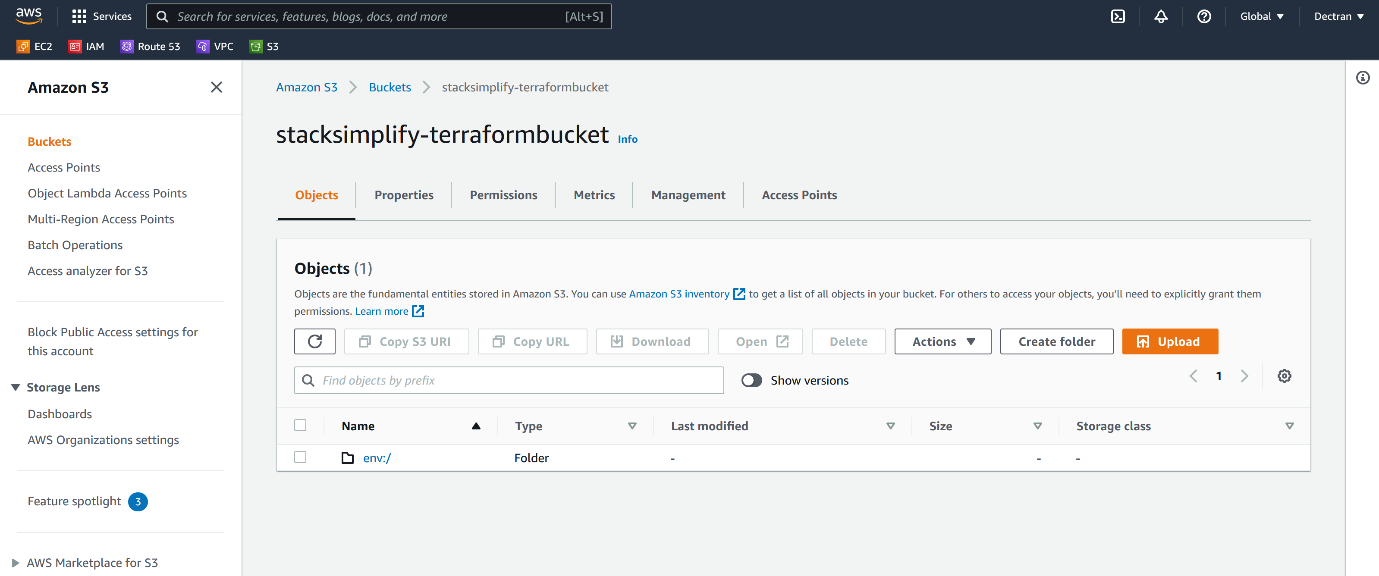
--- terraform workspace select prod

**# Delete workspace**

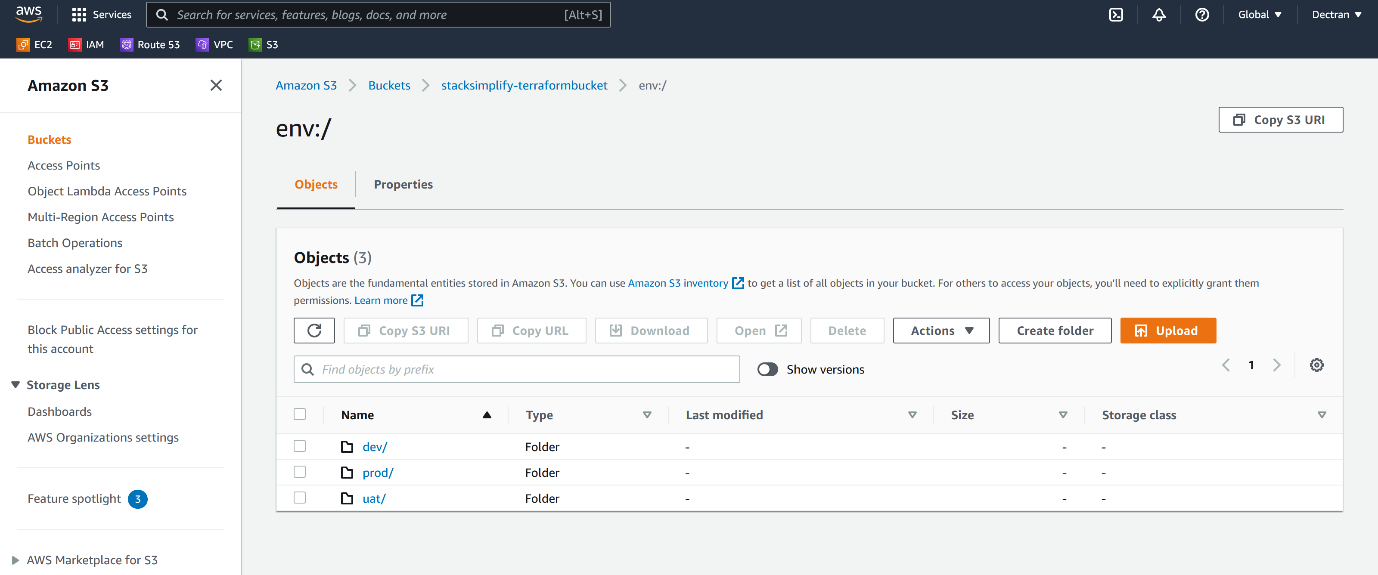
--- terraform workspace delete prod

**What happened in s3bucket**

--- **note** – after creating 3 environments in terraform work space, the terraform will create a separate state file for each environment and store it in each environment.



--- click on env



**Terraform userdata**

--- Reference <https://registry.terraform.io/providers/serverscom/serverscom/latest/docs/guides/user-data>

 resource "aws\_instance" "web-1" {

     ami = "ami-052efd3df9dad4825"

     #ami = "ami-0d857ff0f5fc4e03b"

     #ami = "${data.aws\_ami.my\_ami.id}"

     availability\_zone = "us-east-1a"

     instance\_type = "t2.micro"

     key\_name = "terraform-key"

     subnet\_id = "${aws\_subnet.subnet1-public.id}"

     vpc\_security\_group\_ids = ["${aws\_security\_group.allow\_all.id}"]

     associate\_public\_ip\_address = true

     tags = {

         Name = "${var.environment}-Server-1"

         Env = "${var.environment}"

         Owner = "prabhu"

    CostCenter = "ABCD"

     }

user\_data = <<-EOF

#! /bin/bash

sudo apt-get update

sudo apt-get install -y nginx

echo "<h1>${var.environment}-server-1</h1>" | sudo tee /var/www/html/index.html

EOF

 }

--- we can give user data in ec2 instance like this.

**dev**

--- **note** – go to dev environment and execute below commands.

--- **terraform workspace select dev**

--- **terraform plan --var-file dev.tfvars**

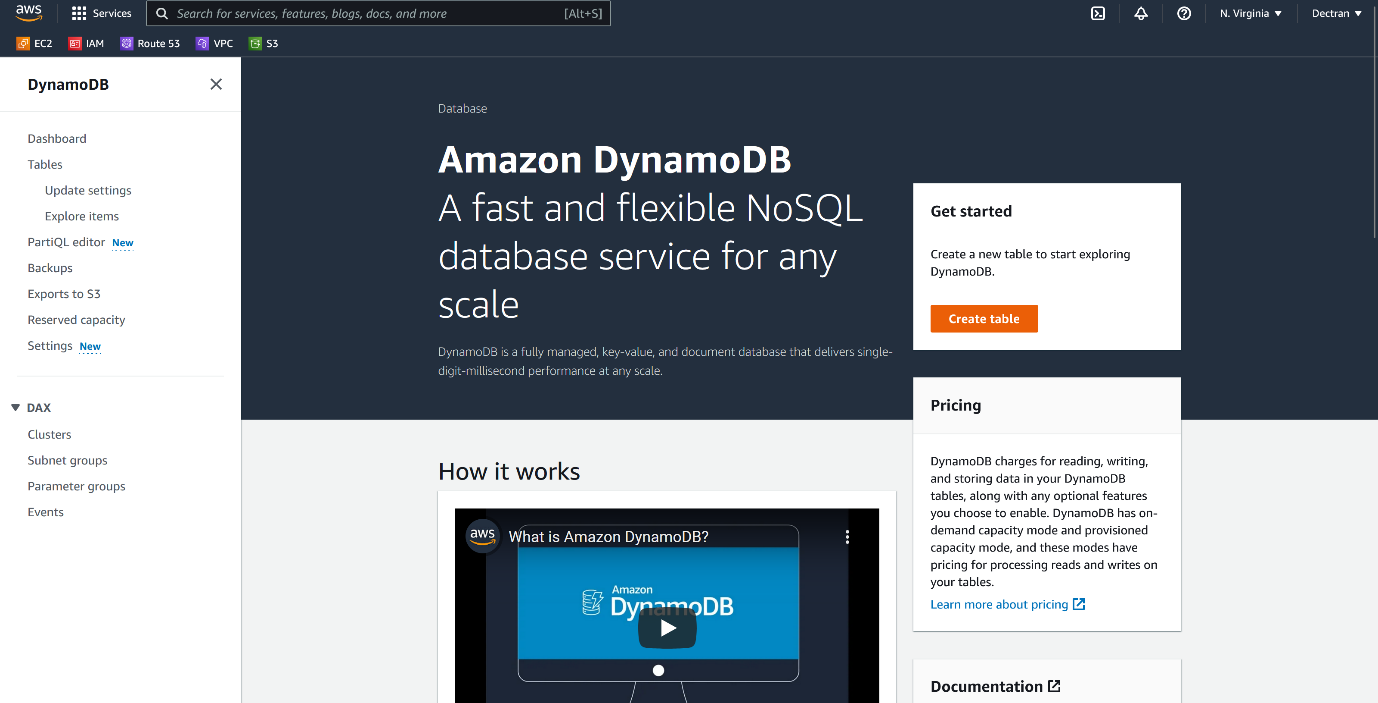
--- **terraform apply --var-file dev.tfvars**

--- **terraform destroy --var-file dev.tfvars -auto-approve**

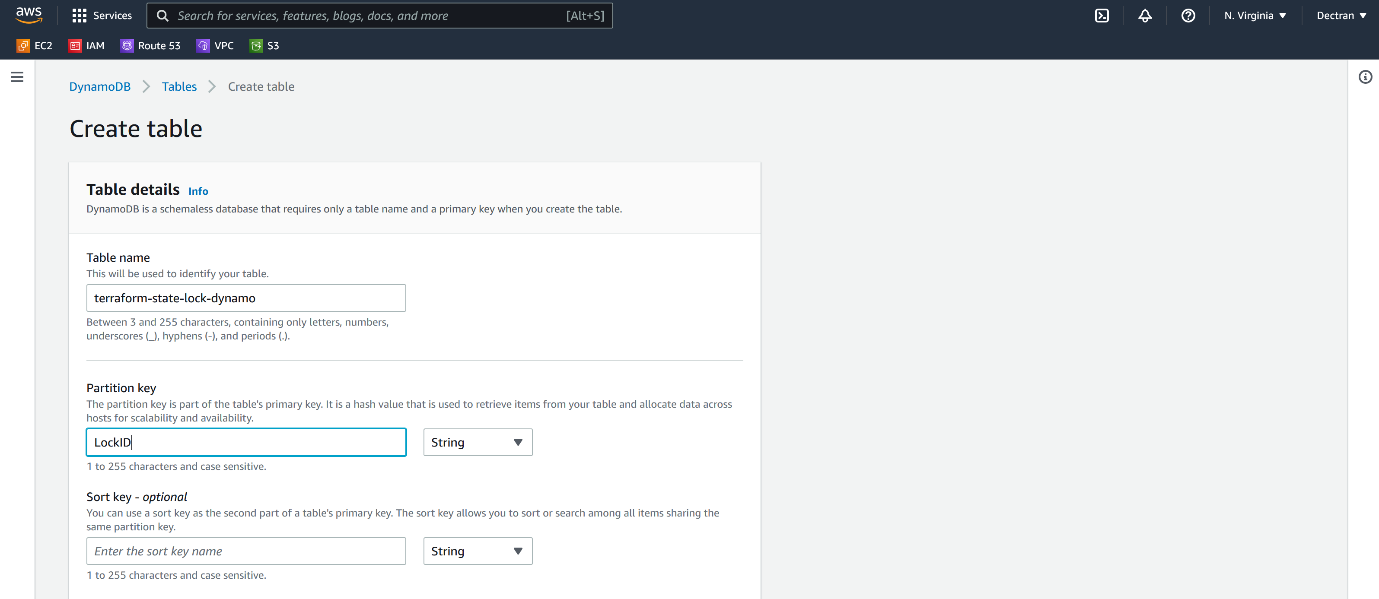
--- **note** – if you want to deploy the infrastructure for dev environment then you can deploy the like this.

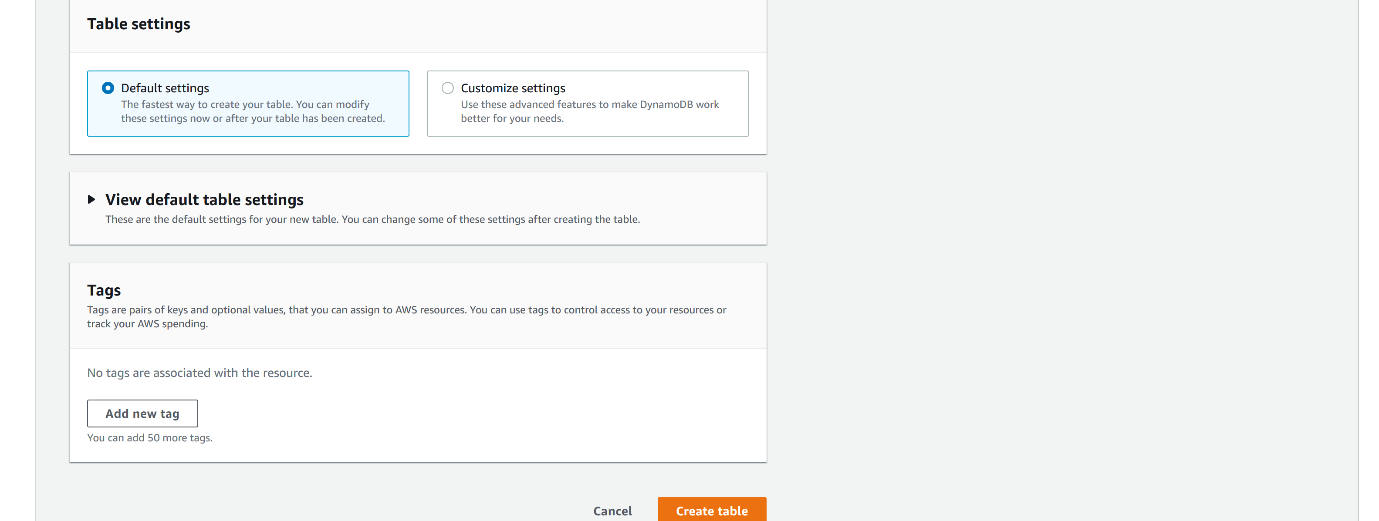
--- **note** – never use other environment tfvars in dev.

**DynamoDB Locking**



--- Click on create





--- click on create.

--- **note** – we don’t need to manually create dynamodb, terraform will create the db.

**Terraform manifest to create dynamodb.**

# create a dynamodb table for locking the state file

resource "aws\_dynamodb\_table" "dynamodb-terraform-state-lock" {

  name = "terraform-state-lock-dynamo"

  hash\_key = "LockID"

  read\_capacity = 20

  write\_capacity = 20

  attribute {

    name = "LockID"

    type = "S"

  }

  tags= {

    Name = "DynamoDB Terraform State Lock Table"

  }

}

--- **note** – after creating dynamodb resource, please add (dynamodb\_table = "terraform-state-lock-dynamo" ) this parameter in s3backend.tf file.

--- s3backend.tf

terraform {

  backend "s3" {

    bucket = "stacksimplify-terraformbucket"

    key    = "workspace.tfstate"

    region = "us-east-1"

    dynamodb\_table = "terraform-state-lock-dynamo"

  }

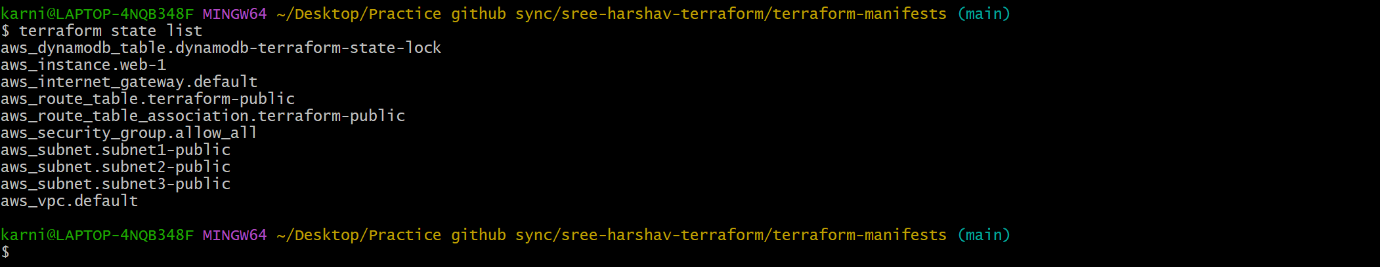
}

--- **Scenario** – if you destroy your infrastructure then the terraform also destroy dynamo db so, you do not want that to happened then you can remove dynamodb from terraform state file.

--- **important** – 1st create dynamodb and then mention the dynamodb name in the s3backend.tf file

**# List the terraform resources**

--- terraform state list



**# Remove dynamodb form state file**

--- terraform state rm aws\_dynamodb\_table.dynamodb-terraform-state-lock

**Terraform manifest for workspace project**

--- c1-0-provider.tf

provider "aws" {

    #access\_key = "${var.aws\_access\_key}"

    #secret\_key = "${var.aws\_secret\_key}"

    region = "${var.aws\_region}"

}

terraform {

  required\_version = "<= 1.2" #Forcing which version of Terraform needs to be used

  required\_providers {

    aws = {

      version = "<= 3.0.0" #Forcing which version of plugin needs to be used.

      source = "hashicorp/aws"

    }

  }

}

--- c1-1-provider-variables.tf

variable "aws\_region" {}

#variable "aws\_access\_key" {}

#variable "aws\_secret\_key" {}

--- c2-0-vpc.tf

resource "aws\_vpc" "default" {

    cidr\_block = "${var.vpc\_cidr}"

    enable\_dns\_hostnames = true

    tags = {

        Name = "${var.vpc\_name}"

    Owner = "karni prabhu"

    environment = "${var.environment}"

    }

}

resource "aws\_internet\_gateway" "default" {

    vpc\_id = "${aws\_vpc.default.id}"

    tags = {

        Name = "${var.IGW\_name}"

    }

}

resource "aws\_subnet" "subnet1-public" {

    vpc\_id = "${aws\_vpc.default.id}"

    cidr\_block = "${var.public\_subnet1\_cidr}"

    availability\_zone = "us-east-1a"

    tags = {

        Name = "${var.public\_subnet1\_name}"

    }

}

resource "aws\_subnet" "subnet2-public" {

    vpc\_id = "${aws\_vpc.default.id}"

    cidr\_block = "${var.public\_subnet2\_cidr}"

    availability\_zone = "us-east-1b"

    tags = {

        Name = "${var.public\_subnet2\_name}"

    }

}

resource "aws\_subnet" "subnet3-public" {

    vpc\_id = "${aws\_vpc.default.id}"

    cidr\_block = "${var.public\_subnet3\_cidr}"

    availability\_zone = "us-east-1c"

    tags = {

        Name = "${var.public\_subnet3\_name}"

    }

}

resource "aws\_route\_table" "terraform-public" {

    vpc\_id = "${aws\_vpc.default.id}"

    route {

        cidr\_block = "0.0.0.0/0"

        gateway\_id = "${aws\_internet\_gateway.default.id}"

    }

    tags = {

        Name = "${var.Main\_Routing\_Table}"

    }

}

resource "aws\_route\_table\_association" "terraform-public" {

    subnet\_id = "${aws\_subnet.subnet1-public.id}"

    route\_table\_id = "${aws\_route\_table.terraform-public.id}"

}

--- c2-1-vpc-variables.tf

variable "vpc\_cidr" {}

variable "vpc\_name" {}

variable "IGW\_name" {}

variable "key\_name" {}

variable "public\_subnet1\_cidr" {}

variable "public\_subnet2\_cidr" {}

variable "public\_subnet3\_cidr" {}

variable "private\_subnet\_cidr" {}

variable "public\_subnet1\_name" {}

variable "public\_subnet2\_name" {}

variable "public\_subnet3\_name" {}

variable "private\_subnet\_name" {}

variable Main\_Routing\_Table {}

variable "azs" {

  description = "Run the EC2 Instances in these Availability Zones"

  default = ["us-east-1a", "us-east-1b", "us-east-1c"]

}

--- c2-3-vpc-output.tf

--- c3-0-security-group.tf

resource "aws\_security\_group" "allow\_all" {

  name        = "allow\_all"

  description = "Allow all inbound traffic"

  vpc\_id      = "${aws\_vpc.default.id}"

  ingress {

    from\_port   = 0

    to\_port     = 0

    protocol    = "-1"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  egress {

    from\_port       = 0

    to\_port         = 0

    protocol        = "-1"

    cidr\_blocks     = ["0.0.0.0/0"]

    }

}

--- c3-1-security-group-variables.tf

--- c3-3-security-group-output.tf

--- c4-0-ec2-instance.tf

 resource "aws\_instance" "web-1" {

     ami = "ami-052efd3df9dad4825"

     #ami = "ami-0d857ff0f5fc4e03b"

     #ami = "${data.aws\_ami.my\_ami.id}"

     availability\_zone = "us-east-1a"

     instance\_type = "t2.micro"

     key\_name = "terraform-key"

     subnet\_id = "${aws\_subnet.subnet1-public.id}"

     vpc\_security\_group\_ids = ["${aws\_security\_group.allow\_all.id}"]

     associate\_public\_ip\_address = true

     tags = {

         Name = "${var.environment}-Server-1"

         Env = "${var.environment}"

         Owner = "prabhu"

    CostCenter = "ABCD"

     }

user\_data = <<-EOF

#! /bin/bash

sudo apt-get update

sudo apt-get install -y nginx

echo "<h1>${var.environment}-server-1</h1>" | sudo tee /var/www/html/index.html

EOF

 }

--- c4-1-ec2-instance-variables.tf

variable "amis" {

    description = "AMIs by region"

    default = {

        us-east-1 = "ami-97785bed" # ubuntu 14.04 LTS

    us-east-2 = "ami-f63b1193" # ubuntu 14.04 LTS

    us-west-1 = "ami-824c4ee2" # ubuntu 14.04 LTS

    us-west-2 = "ami-f2d3638a" # ubuntu 14.04 LTS

    }

}

variable "instance\_type" {

  default = {

    dev = "t2.nano"

    test = "t2.micro"

    prod = "t2.medium"

    }

}

--- c4-3-ec2-instance-outputs.tf

output "web-1\_id" {

 description = "web-1 id"

  value       = ["${aws\_instance.web-1.id}"]

}

output "web-1\_public\_ip" {

  description = "web-1 instance id"

  value       = ["${aws\_instance.web-1.public\_ip}"]

}

--- c5-1-s3backend.tf

terraform {

  backend "s3" {

    bucket = "stacksimplify-terraformbucket"

    key    = "workspace.tfstate"

    region = "us-east-1"

  }

}

--- dev.tfvars

#aws\_access\_key = "xxxxxx"

#aws\_secret\_key = "yyyyyyy"

aws\_region = "us-east-1"

vpc\_cidr = "10.1.0.0/16"

public\_subnet1\_cidr = "10.1.1.0/24"

public\_subnet2\_cidr = "10.1.2.0/24"

public\_subnet3\_cidr = "10.1.3.0/24"

private\_subnet\_cidr = "10.1.20.0/24"

vpc\_name = "terraform-aws-dev"

IGW\_name = "terraform-aws-igw-dev"

public\_subnet1\_name = "Terraform\_Public\_Subnet1-dev"

public\_subnet2\_name = "Terraform\_Public\_Subnet2-dev"

public\_subnet3\_name = "Terraform\_Public\_Subnet3-dev"

private\_subnet\_name = "Terraform\_Private\_Subnet-dev"

Main\_Routing\_Table = "Terraform\_Main\_table-dev"

key\_name = "terraform-key"

environment = "dev"

--- uat.tfvars

#aws\_access\_key = "xxxxxx"

#aws\_secret\_key = "yyyyyyy"

aws\_region = "us-east-1"

vpc\_cidr = "10.1.0.0/16"

public\_subnet1\_cidr = "10.1.1.0/24"

public\_subnet2\_cidr = "10.1.2.0/24"

public\_subnet3\_cidr = "10.1.3.0/24"

private\_subnet\_cidr = "10.1.20.0/24"

vpc\_name = "terraform-aws-uat"

IGW\_name = "terraform-aws-igw-uat"

public\_subnet1\_name = "Terraform\_Public\_Subnet1-uat"

public\_subnet2\_name = "Terraform\_Public\_Subnet2-uat"

public\_subnet3\_name = "Terraform\_Public\_Subnet3-uat"

private\_subnet\_name = "Terraform\_Private\_Subnet-uat"

Main\_Routing\_Table = "Terraform\_Main\_table-uat"

key\_name = "terraform-key"

environment = "uat"

--- prod.tfvars

#aws\_access\_key = "xxxxxx"

#aws\_secret\_key = "yyyyyyy"

aws\_region = "us-east-1"

vpc\_cidr = "10.1.0.0/16"

public\_subnet1\_cidr = "10.1.1.0/24"

public\_subnet2\_cidr = "10.1.2.0/24"

public\_subnet3\_cidr = "10.1.3.0/24"

private\_subnet\_cidr = "10.1.20.0/24"

vpc\_name = "terraform-aws-prod"

IGW\_name = "terraform-aws-igw-prod"

public\_subnet1\_name = "Terraform\_Public\_Subnet1-prod"

public\_subnet2\_name = "Terraform\_Public\_Subnet2-prod"

public\_subnet3\_name = "Terraform\_Public\_Subnet3-prod"

private\_subnet\_name = "Terraform\_Private\_Subnet-prod"

Main\_Routing\_Table = "Terraform\_Main\_table-prod"

key\_name = "terraform-key"

environment = "prod"