TERRAFORM CLASS 3

TERRAFORM DYNAMIC BLOCKS: Dynamic blocks helps to reduce repetitive codes.

```
resource "aws_security_group" "mysec" {
  name = "prodsecgroups"

  dynamic "ingress" {
    for_each = var.ports
    content {
      from_port   = ingress.value
      to_port     = ingress.value
      protocol    = "tcp"
      cidr_blocks = ["10.0.0.0/0"]
    }
  }
}

variable "ports" {
  type    = list(any)
  default = [22, 2049, 80]
}
```

ALIAS & PROVIDERS:

provider "aws" {

region = "us-east-1"

}

resource "aws_instance" "one" {

ami = "ami-0715c1897453cabd1"

instance_type = "t2.micro"

tags = {

Name = "web-server"

}

}

```
provider "aws" {

region = "ap-south-1"

alias = "south"

}

resource "aws_instance" "two" {

provider = "aws.south"

ami = "ami-0607784b46cbe5816"

instance_type = "t2.micro"

tags = {

Name = "web-server"

}

}
```

TERRAFORM OUTPUTS: used to show the properties/metadata of resources

```
provider "aws" {

}

resource "aws_instance" "two" {

    ami = "ami-0715c1897453cabd1"

    instance_type = "t2.micro"

    tags = {

        Name = "web-server"

    }

}

output "abc" {
```

```
    value = [aws_instance.two.public_ip, aws_instance.two.public_dns,
    aws_instance.two.private_ip]

}
```

TERRAFORM WORKSPACE:

In Terraform, a workspace is a way to manage multiple instances of your infrastructure configurations. Workspaces allow you to maintain different sets of infrastructure within the same Terraform configuration files. Each workspace has its own state, variables, and resources, allowing you to manage and deploy distinct environments or configurations.

Default Workspace:

- When you initialize a Terraform configuration without explicitly creating a workspace, you are in the default workspace. The default workspace is often used for the main or production environment.

Create a Workspace:

- You can create additional workspaces using the terraform workspace new

List Workspaces:

- To see a list of available workspaces, you can use: terraform workspace list

Select a Workspace:

- Use the terraform workspace select command to switch between workspaces: terraform workspace select dev

Destroy Specific Workspace:

- You can destroy resources for a specific workspace using: terraform workspace select dev && terraform destroy

EX ON WORKSPACE:

vim main.tf

```
provider "aws" {
  region = var.region
  access_key = ""
  secret_key = ""
}
resource "aws_instance" "example" {
  ami           = var.ami
  instance_type = var.size
  tags = {
    Name = "demo-server"
  }
}

variable "region" {
  default = "us-east-1"
}

variable "ami" {
  default = "ami-00b8917ae86a424c9"
}

variable "size" {
  default = "t2.micro"
}
```

terraform init
terraform plan
terraform apply

This will create instance in default workspace

Create new workspace called **dev**

terraform init
terraform plan
terraform apply

This will create one more instance with same config on **dev** env

Even though there is one instance already up & running on AWS, terraform will create one more instance because we run the terraform commands from different workspace!

Create new workspace called **prod**

terraform init
terraform plan
terraform apply

This will create one more instance with same config on **prod** env

As per the above code, all the instances will get same instances names, if you wish to change it then change the instance name like below code:

```
tags = {
    Name = "example-server-${terraform.workspace}"
  }
```

If we apply the changes, that will creates the instances like below names:

- demo-server-stg
- demo-server-dev
- demo-server-prod

If i want to give different size of instance for different environment also, for that we need to modify our main.tf file as below

```
provider "aws" {
  region = var.region
}
```

```
locals {
 instance_types = {
  dev   = "t2.micro"
  test = "t2.small"
  prod  = "m4.large"
 }
}

resource "aws_instance" "demo_server" {
 ami          = var.ami
 instance_type = local.instance_types[terraform.workspace]
 tags = {
  Name = "example-server-${terraform.workspace}"
 }
}
```

If we execute the above code, that will create different sizes for instances.

IMPORTING EXISITING RESOURCES TO TERRAFORM:

If the infra is already created manually, we can import all the data in terraform.

create **provider.tf** file

```
provider "aws" {

       region = "us-east-1"

      profile = "default"

}
```

now use **terraform init** command

create **main.tf** file:

```
resource "aws_instance" "my-server" {
 ami          = "ami-0578f2b35d0328762"
 instance_type = "t2.micro"
```

```
  tags = {
    Name = "my-server"
  }
}
```

Note: Make sure that we have to mention existing EC2 AMI & Name on **main.tf** file

Now use **terraform import aws_instance.<instancename> <instanceid>** command to import all the data.