sudo yum update -y

update komutu

- Go to the terminal and run `terraform validate`. It validates the Terraform files syntactically correct and internally consistent.

Terraform validate s’ntak hatalarini gösterir terrraformda

terraform fmt

bununla ilgi genel yapi bozulduğunda hatalari gosteriri/ terraformdaki boşluklar hata değildir

terraform console

terraform console komutu ile terraformda ayrica sana bir console veriri ve terraform un içinde istenilen bilgiler cagrilabilir.ctrl c ile console den cikilir consol ile istenilen bilgiler çekilir ,rnegin

terraform console

> aws\_instance.tf-ec2

> aws\_instance.tf-ec2.private\_ip

> min (1,2,3)

> lower("HELLO")

> file("${path.module}/cloud")

> aws\_s3\_bucket.tf-s3

> aws\_s3\_bucket.tf-s3.bucket

> exit or (ctrl+c)

Terraform graph komutu le

- Go to the terminal and run `terraform graph`. It creates a visual graph of Terraform resources. The output of "terraform graph" command is in the DOT format, which can easily be converted to an image by making use of dot provided by GraphViz.

Terraform show var olan kaynaklari gösterir

Terraform graph tum kaynkalri siralar https://dreampuf.github.io/GraphvizOnline

Bu siteye getiridigimiz komutla içerini bununla değiştirir ve kaynaklarin garfik seklini g,rmus oluruz/

Vscode da terraform dan habersiz aws console uzeurnde kayankarda değişiklik yapldiginda TERRAFORMUN HABERI OLMAZ bu nedenle terraformun haberi olmasi ve sistemi terraformun güncellemesi için “terraform apply -refresh-only

Terraform output :kaynaklarin ciktiisini verir

terraform output -json

bu kodla kaynaklarin json formati verilrir

terraform output tf\_example\_public\_ip

burda ne istiyorsak onu yaziyoruz burda tf\_example instace makinenin public ip si istenoyor onu veiri/

terraform state list

terraformda oluşturulan kaynaklari listeler

- The `terraform apply -refresh-only` command is used to update the state file with the real-world infrastructure. This can be used to detect any drift from the last-known state, and to update the state file. First, check the current state of your resources with `terraform state list`. Then go to the AWS console and delete your S3 bucket `sam-tf-test-bucket-addwhateveryouwant`. Display the state list again and refresh the state. Run the following commands.

Bu terreform apply -refresh-only komutu ile aws console uzerindeki degisikleri vscode uzerinde calistirlimis olan terraform a da bu degisileri gorulur ve terraform statle list yada terraform show dediğimizde aws consolde yapilan degisiklikler burdada gösterilmiş olunur/

terraform apply -auto-approve

bu komuytla terraform apply dedidgimizde onay için herzaman yes dememze gerek kalmadan otomatil uygula diyor

terraform apply -refresh=false

bu lomut ille daha j hiizli bir sekilde kayanakl sile r ve bu komtla refresh yapma hizlica sonuca gider direk ciktiyio verir/

terraform destroy

terraform destroy ile tum kayanklar silinir

sam.auto.tfvars

terraform.tfvars

sam.tfvars

yukaridaki 3 tf dosyasinda onceliklerine gore siralndi oncelik “auto.tfvars”sonra “terraform.tfvars” sonra “tfvars” ta bulunan bilgiler çekilir/

- Terraform loads variables in the following order:

  - Any -var and -var-file options on the command line, in the order they are provided.

  - Any \*.auto.tfvars or \*.auto.tfvars.json files, processed in lexical order of their filenames.

  - The terraform.tfvars.json file, if present.

  - The terraform.tfvars file, if present.

  - Environment variables

-komut satırında -var kullanıldıysa önce onu dikkate alır. yoksa auto.tfvars ya da auto.tfvars.json dosyalarını dikkate alır.

onlar yoksa terraform.tfvars.json dikkate alınır. O yoksa terraform.tfvars dikkate alınır. Bunlardan sonra da environment variables dikkate alınır.

- Run terraform apply command.

**TERRAFORM MODULS**

```txt

 terraform-modules

   ├── dev

   │   └── dev-vpc.tf

   ├── modules

   │   ├── main.tf

   │   ├── outputs.tf

   │   └── variables.tf

   └── prod

       └── prod-vpc.tf

```

Terraform modüle rood directory dur . dev ,pro ve modules child modlu alt modüllerdir .modules içinde yazilan kod bloklari hem dev de hemde prod ta kullanilabilr .

mkdir terraform-modules && cd terraform-modules && mkdir dev modules prod && cd dev && touch dev-vpc.tf && cd ../modules && touch main.tf outputs.tf variables.tf && cd ../prod && touch prod-vpc.tf && cd ../modules

bu sekilde birçok kodu ayni anda calistirlabilrinir

**modüles içindeki main.tf dosyasi”**

provider "aws" {

region = "us-east-1"

}

resource "aws\_vpc" "module\_vpc" {

cidr\_block = var.vpc\_cidr\_block

tags = {

Name = "terraform-vpc-${var.environment}"

}

}

resource "aws\_subnet" "public\_subnet" {

cidr\_block = var.public\_subnet\_cidr

vpc\_id = aws\_vpc.module\_vpc.id

tags = {

Name = "terraform-public-subnet-${var.environment}"

}

}

resource "aws\_subnet" "private\_subnet" {

cidr\_block = var.private\_subnet\_cidr

vpc\_id = aws\_vpc.module\_vpc.id

tags = {

Name = "terraform-private-subnet-${var.environment}"

}

}

**VARIABLE DOSYASI ICIN “**

variable "environment" {

default = "Techproed"

}

variable "vpc\_cidr\_block" {

default = "10.0.0.0/16"

description = "this is our vpc cidr block"

}

variable "public\_subnet\_cidr" {

default = "10.0.1.0/24"

description = "this is our public subnet cidr block"

}

variable "private\_subnet\_cidr" {

default = "10.0.2.0/24"

description = "this is our private subnet cidr block"

}

**OUTPUT ICIN”**

output "vpc\_id" {

value = aws\_vpc.module\_vpc.id

}

output "vpc\_cidr" {

value = aws\_vpc.module\_vpc.cidr\_block

}

output "public\_subnet\_cidr" {

value = aws\_subnet.public\_subnet.cidr\_block

}

output "private\_subnet\_cidr" {

value = aws\_subnet.private\_subnet.cidr\_block

}

**Dev içindeki kodlar**

module "tf-vpc" {

source = "../modules"

environment = "DEV"

}

output "vpc-cidr-block" {

value = module.tf-vpc.vpc\_cidr

**prod içindekiler”**

module "tf-vpc" {

source = "../modules"

environment = "PROD"

}

output "vpc-cidr-block" {

value = module.tf-vpc.vpc\_cidr

NOT : Burda “modül “ içinde oluşturulan main.tf ana dosyadır variable.tf dosyasi main.tf dosyasinin refereans aldigi dosyaya varabletf dir output.tf ise bizim goremk istediğimiz ciktilardir/

Moduls yapmamizdaki amaç her oluşturulan farkli projelerde ayni calismalari devam li yapmak yerine sabit bir yerde hazirlayip ordaki kodlari istediğimiz alanda calisma ortaminda kullanmaktır.orek uugulamada bir tane moduls dosyasi bir tane dev dosyasi bir tane proddosyasi hazirlanir .moduls iinde di[er iki ortaminda (dev ve prod)calisabilecegi ana kodlar burda maintf varialetf ve output tf seklinde dosyalar içinde kodlar hazirlanir ,dev ortaminda moduls icindehaziranan bu dosyalar referans alinarak calisma alt yapisi hazrilanmis olur ayni sekilde prod ortaminda moduls içindeki hazir kodlar referans alinarak alt yapi hazirlanmis olnunur ,bu calisma zaman kaynini onlemek daha az hatalarin olamasini onlemek vb birck alanda kolaylık saglamis olur,

Terraform tfstate: bu dosya maint.tf dosyasini calistirmak için kullandigimiz “terraform init “ kodunu girdiğimizde terraform tfstate dosyasi oluşur.burda kaynaklarin çizelgesi alinir.”terraform apply “ dedikten sonra “terraform tfstate”calisir.Bu apply komutu ile çizelge uygulanir .

Terraform.tfstate.backup: bu dosya içinde eski calistiirlna tum kaynaklarin backuplari bulur.

TERRAFORM IMPORT

Terraform dusinda oluşturulmuş kaynakalrin terraforma import ile getirilip yönetilmesidir.ornek uygulama :

AWS consolde 2 adet ec2 instance yaga kaldiririiz ve bunlarin securty group olustuurp baglariz/vscode terraformun maintf dosyasini calistirmadan klasör dissinda bir tane daha “learn-terraform-import” adli klasör olusuturuz ve bunun altinda main.tf klasör olu;turuz

Main.tf dosyasi “

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.23.1"

}

}

}

provider "aws" {

region = "us-east-1"

}

variable "tf-ami" {

type = list(string)

default = ["ami-005f9685cb30f234b", "ami-0557a15b87f6559cf", "ami-0c9978668f8d55984"]

}

variable "tf-tags" {

type = list(string)

default = ["aws-linux-2", "ubuntu-22.04", "red-hat-linux-8"]

}

resource "aws\_instance" "tf-instances" {

ami = element(var.tf-ami, count.index )

instance\_type = "t2.micro"

count = 3

key\_name = "selma\_key1" // change here

security\_groups = ["tf-import-sg"]

tags = {

Name = element(var.tf-tags, count.index )

}

}

resource "aws\_security\_group" "tf-sg" {

name = "tf-import-sg-1"

description = "terraform import security group"

tags = {

Name = "tf-import-sg-1"

}

ingress {

from\_port = 80

protocol = "tcp"

to\_port = 80

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 22

protocol = "tcp"

to\_port = 22

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

protocol = -1

to\_port = 0

cidr\_blocks = ["0.0.0.0/0"]

}

}

Bu main.tf dosyasi hazirlandiktan sonra terraform disinda aws console uzerinden olusturlan iki tane ec2 instance makineyi buraya tasimak için aws consola gidin o iki makinenin tek tek id adresleri alip altaki kodun sonunaki id ile degistilir.

terraform import "aws\_instance.tf-instances[0]" *i-013bbbe1123ce832e*

burdaki kod satriinin sonuna aws deki mainein id si ile değiştirilip enter ile kod calistirilir

vscodeda terminalde terraform state list dediğimizde terraform daki tum kaynaklari gösterir buraya yeni aws den makinen buraya geldirgini goruuruz ayni sekilde aws deki ikinci makinenin de id kodu buraya

terraform import "aws\_instance.tf-instances[1]" *i-013bbbe1123ce832e*

sondaki id nol ile değiştirilip terraform a bu iki makine eklenmiş olur.[] koseli parantez içindeki 0.1.2.3 vb numaralar makinelerin numraanip siraya konuğunu gösterir.

Terraform destroy ile terraformda oluşturulan tum makineler gbi tum kaynaklar silinmiş olur .