# Infraestructura

## Networking.tf

Creamos la VPC

module "vpc" {

source = "terraform-aws-modules/vpc/aws"

name = "${local.project\_name}-${terraform.workspace}"

cidr = "${local.vpc\_cidr}"

azs = ["${local.aws\_region}a", "${local.aws\_region}b", "${local.aws\_region}c"]

private\_subnets = ["${local.private\_subnet\_1}", "${local.private\_subnet\_2}", "${local.private\_subnet\_3}"]

public\_subnets = ["${local.public\_subnet\_1}", "${local.public\_subnet\_2}", "${local.public\_subnet\_3}"]

create\_vpc = true

create\_igw = true

enable\_nat\_gateway = true

single\_nat\_gateway = false

reuse\_nat\_ips = true

external\_nat\_ip\_ids = "${aws\_eip.nat.\*.id}"

}

Definimos el CIDR, región y nombre del proyecto para utilizar en los tags. Creamos 3 subredes públicas y 3 subredes privadas

locals {

cluster\_name = "${var.project\_name}-${terraform.workspace}"

#Private Subnets

private\_subnet\_1 = cidrsubnet("${local.vpc\_cidr}", 8, 1)

private\_subnet\_2 = cidrsubnet("${local.vpc\_cidr}", 8, 2)

private\_subnet\_3 = cidrsubnet("${local.vpc\_cidr}", 8, 3)

#Public Subnets

public\_subnet\_1 = cidrsubnet("${local.vpc\_cidr}", 8, 11)

public\_subnet\_2 = cidrsubnet("${local.vpc\_cidr}", 8, 12)

public\_subnet\_3 = cidrsubnet("${local.vpc\_cidr}", 8, 13)

project\_name = "obligatorio"

aws\_region = "us-east-1"

vpc\_cidr = "10.0.0.0/16"

}

Se asignan 3 elastic ips , las cuales se utilizaran en los nat-gateways. Esto proporciona redundancia en caso de que una zona se caiga.

count = 3

vpc = true

}

## Main.tf

Defininos los providers:

Hashicorp/aws - para todo la interacción con AWS

Hashicorp/Helm - para desplegar algún helmchart dentro de eks.

Gavinbunney/kubectl - para ejecutar comandos dentro de nuestro cluster.

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 4.18"

}

helm = {

source = "hashicorp/helm"

version = "~> 2.4"

}

kubectl = {

source = "gavinbunney/kubectl"

version = "~> 1.14"

}

}

Defininos el bucket de S3 que se creó a mano donde vamos a guardar el estado de la infra manejado por terraform.

backend "s3" {

bucket = "obligatorio-abdm-terraform"

key = "obligatorio.tfstate"

region = "us-east-1"

encrypt = true

}

aws s3api create-bucket --bucket obligatorio-abdm-terraform --region us-east-1

## Iam.tf

Se definan reglas de entropía de password para los usuarios de la cuenta.

module "iam\_account" {

source = "terraform-aws-modules/iam/aws//modules/iam-account"

version = "~> 4.3"

account\_alias = var.company\_name

minimum\_password\_length = 12

max\_password\_age = 30

password\_reuse\_prevention = 5

require\_lowercase\_characters = true

require\_uppercase\_characters = true

require\_symbols = true

require\_numbers = true

}

Se crean políticas y roles para los distintos componentes que corren desde eks.

(Por ser muy largo solo se pone un ejemplo)

resource "aws\_iam\_role" "route53-externaldns-controller" {

name = "route53-externaldns-controller"

assume\_role\_policy = data.aws\_iam\_policy\_document.external\_dns.json

}

## Route53.tf

Se define la zona pública de DNS.

resource "aws\_route53\_zone" "primary" {

name = var.route53\_domain\_name

}

Se crea un registro de DNS tipo A con el valor del balanceador que se crea más adelante.

resource "aws\_route53\_record" "obligatorio" {

zone\_id = aws\_route53\_zone.primary.zone\_id

name = "obligatorio"

type = "A"

alias {

name = aws\_lb.obligatorio.dns\_name

zone\_id = aws\_lb.obligatorio.zone\_id

evaluate\_target\_health = false

}

}

## Lb.tf

Se crea un balanceador externo y se vincula con el security group creado en sg.tf

resource "aws\_lb" "obligatorio" {

name = var.project\_name

internal = false

load\_balancer\_type = "application"

security\_groups = [module.sg\_external\_alb.security\_group\_id]

subnets = "${module.vpc.public\_subnets}"

enable\_deletion\_protection = true

}

Se crea un listener para el puerto 80 y se lo agrega al balanceador previamente creado.

resource "aws\_lb\_listener" "obligatorio" {

load\_balancer\_arn = aws\_lb.obligatorio.arn

port = "80"

protocol = "HTTP"

default\_action {

type = "forward"

target\_group\_arn = aws\_lb\_target\_group.obligatorio.arn

}

}

Se crea un target group que va a escuchar en el puerto 31234.

resource "aws\_lb\_target\_group" "obligatorio" {

name = "obligatorio-tg"

port = 31234

protocol = "HTTP"

vpc\_id = module.vpc.vpc\_id

}

Se listan todas las instancias de EC2 que tengan como tag Name = initial.

data "aws\_instances" "obligatorio" {

instance\_tags = {

Name = "initial"

}

instance\_state\_names = ["running", "stopped"]

}

Se agregan las instancias previamente listadas al target group.

resource "aws\_lb\_target\_group\_attachment" "obligatorio" {

target\_group\_arn = aws\_lb\_target\_group.obligatorio.id

count = length(data.aws\_instances.obligatorio.ids)

target\_id = data.aws\_instances.obligatorio.ids[count.index]

port = 31234

}

## Sg.tf

Se define el security group que se agrega al balanceador que recibe todo el tráfico externo.

module "sg\_external\_alb" {

source = "terraform-aws-modules/security-group/aws"

name = "external\_alb"

description = "Security group for external connections"

vpc\_id = module.vpc.vpc\_id

egress\_rules = ["all-all"]

ingress\_with\_cidr\_blocks = [

{

from\_port = 80

to\_port = 80

protocol = "tcp"

description = "external to LB"

cidr\_blocks = "0.0.0.0/0"

}

]

}

## Eks.tf

Se crea el cluster de k8s.

module "eks" {

source = "terraform-aws-modules/eks/aws"

version = "~> 18.0"

cluster\_name = "${var.project\_name}-${terraform.workspace}"

cluster\_version = "1.22"

cluster\_endpoint\_private\_access = true

cluster\_endpoint\_public\_access = true

enable\_irsa = true

cluster\_addons = {

coredns = {

resolve\_conflicts = "OVERWRITE"

}

kube-proxy = {}

vpc-cni = {

resolve\_conflicts = "OVERWRITE"

}

}

Se crean instancias auto manejadas tipo spot para los workers.

eks\_managed\_node\_groups = {

initial = {

min\_size = 1

max\_size = 1

desired\_size = 1

instance\_types = ["t3.medium"]

capacity\_type = "SPOT" # ON\_DEMAND or SPOT

}

}

Se asignan permisos al cluster para un grupo de usuario aws/iam llamado 2soAdmin.

# aws-auth configmap

manage\_aws\_auth\_configmap = true

aws\_auth\_roles = [

{

rolearn = "arn:aws:iam::813224394680:group/2soAdmin"

usergroup = "2soAdmin"

groups = ["system:masters"]

}

]

Graphical user interface, application

Description automatically generated