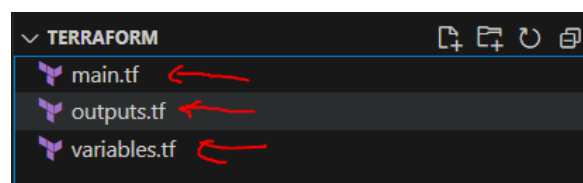


Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

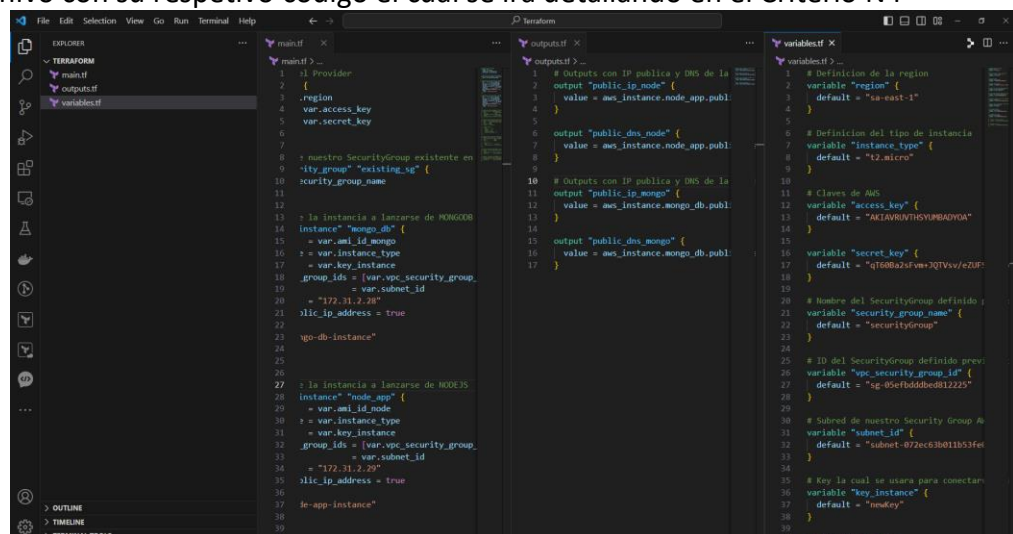
## Actividad Abierta Laboratorio: despliegue de MEAN multicapa mediante Terraform

### Criterio 1: Se entrega un proyecto de Terraform.

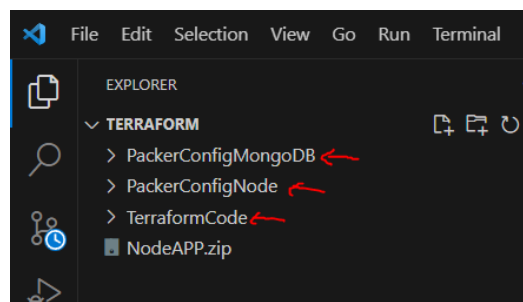
Se ha desarrollado el proyecto completo cumpliendo la arquitectura de Terraform donde se tienen los archivos de “variables.tf”, “outputs.tf” como también el principal que es “main.tf”



Cada Archivo con su respectivo código el cual se ira detallando en el Criterio N4



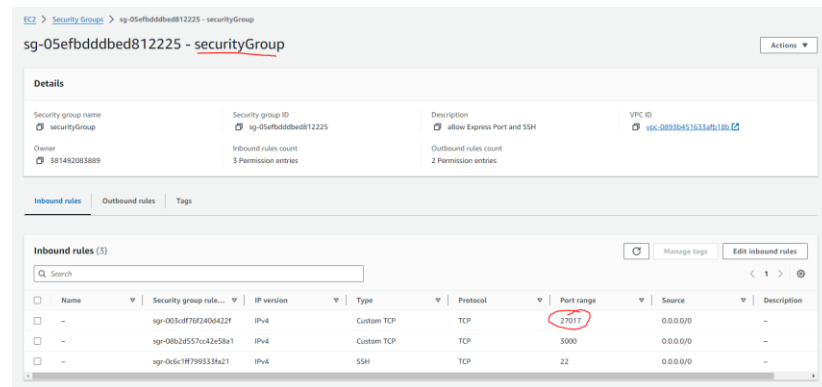
En el entregable comprimido esta Una carpeta con el código de terrform y tambien La configuración de las imágenes de cada una de las instancias como la aplicación como tal en node comprimida



Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

## Criterio 2: Security Groups.

Para el apartado de Security Group se uso el mismo que se uso para la actividad de Packer que ya esta definido en AWS donde ya teníamos las reglas definidas y únicamente se añadió la regla para la conexión a mongodb que es el puerto 27017.



El cual también hacemos referencia directamente desde nuestro archivo “variables.tf” de terraform

```

variables.tf
variables.tf > ...
1 # Definicion de la region
2 variable "region" {
3   default = "sa-east-1"
4 }
5
6 # Definicion del tipo de instancia
7 variable "instance_type" {
8   default = "t2.micro"
9 }
10
11 # Claves de AWS
12 variable "access_key" {
13   default = "AKIAVRUVTHSYUMBADYOA"
14 }
15
16 variable "secret_key" {
17   default = "qT60Ba2sFvm+JQIVsv/eZUFSjlp4WDkCdEHjBxr6"
18 }
19
20 # Nombre del SecurityGroup definido previamente en AWS
21 variable "security_group_name" {
22   default = "securityGroup"
23 }

```

Para posteriormente asociarlo a las instancias a deployarse en el archivo “main.tf”

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

```

main.tf x
main.tf > ...
1  # Definicion del Provider
2  provider "aws" {
3      region = var.region
4      access_key = var.access_key
5      secret_key = var.secret_key
6  }
7
8  # Aclaracion de nuestro SecurityGroup existente en AWS
9  data "aws_security_group" "existing_sg" {
10     name = var.security_group_name
11 }
12
13 # Definicion de la instancia a lanzarse de MONGODB
14 resource "aws_instance" "mongo_db" {
15     ami           = var.ami_id_mongo
16     instance_type = var.instance_type
17     key_name       = var.key_instance
18     vpc_security_group_ids = [var.vpc_security_group_id]
19     subnet_id      = var.subnet_id
20     private_ip     = "172.31.2.28"
21     associate_public_ip_address = true
22     tags = {
23         Name = "mongo-db-instance"
24     }
25 }
26
27 # Definicion de la instancia a lanzarse de NODEJS
28 resource "aws_instance" "node_app" {
29     ami           = var.ami_id_node
30     instance_type = var.instance_type
31     key_name       = var.key_instance
32     vpc_security_group_ids = [var.vpc_security_group_id]
33     subnet_id      = var.subnet_id
34     private_ip     = "172.31.2.29"
35     associate_public_ip_address = true
36     tags = {
37         Name = "node-app-instance"
38     }
39 }
40

```

### Criterio 3: El Provider de Terraform es correcto.

El provider seleccionado es AWS por que tiene mayor documentación y además que ya teníamos pa imagen de node de la anterior actividad la cual solo se tuvo que ajustar para poder realizar la conexión a MongoDB. Dicho provider es configurado en Terraform junto a las credenciales de AWS

```

main.tf x
main.tf > ...
1  # Definicion del Provider
2  provider "aws" {
3      region = var.region
4      access_key = var.access_key
5      secret_key = var.secret_key
6  }
7

```

### Criterio 4: Existen dos instancias y se demuestra su despliegue en el PDF.

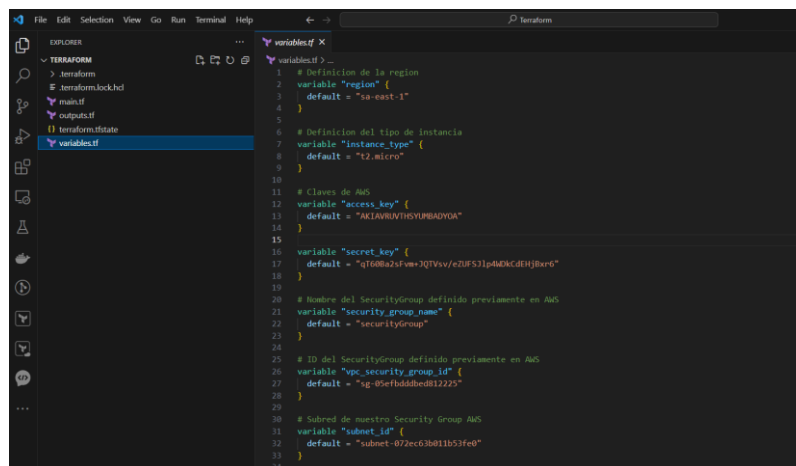
La definición de las instancias de las realiza en el archivo "main.tf" con todas las opciones que se deben configurar cuando lo hacemos manualmente con direcciones IP privadas estáticas para ambas instancias.

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

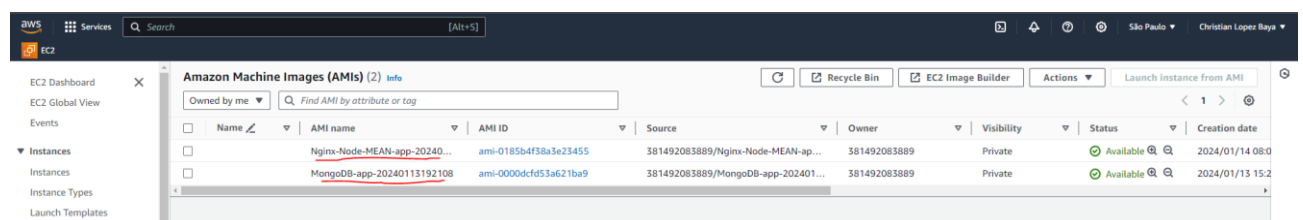
```
# Definicion de la instancia a lanzarse de MONGODB
resource "aws_instance" "mongo_db" {
  ami           = var.ami_id_mongo
  instance_type = var.instance_type
  key_name      = var.key_instance
  vpc_security_group_ids = [var.vpc_security_group_id]
  subnet_id    = var.subnet_id
  private_ip   = "172.31.2.28"
  associate_public_ip_address = true
  tags = {
    Name = "mongo-db-instance"
  }
}

# Definicion de la instancia a lanzarse de NODEJS
resource "aws_instance" "node_app" {
  ami           = var.ami_id_node
  instance_type = var.instance_type
  key_name      = var.key_instance
  vpc_security_group_ids = [var.vpc_security_group_id]
  subnet_id    = var.subnet_id
  private_ip   = "172.31.2.29"
  associate_public_ip_address = true
  tags = {
    Name = "node-app-instance"
  }
}
```

Cabe resaltar que las variables fueron leídas del archivo “variables.tf” configurado



Posteriormente seleccionamos la imagen a deployar desde AWS, las cuales fueron creadas ambas con packer.



Cabe aclarar que teníamos previamente creada la imagen de NodeJS con NGINX pero realice algunos ajustes para la conexión a MongoDB para mostrar la conexión y posteriormente volví a generar la imagen con PACKER, cabe resaltar que la ip esta definida privada y estática por que al momento de lanzar la instancia estoy creando ips privadas estáticas para cada instancia para tener mayor control de la misma

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

```

1 const express = require('express')
2 const { MongoClient } = require('mongodb')
3
4 const username = 'christian-unir'
5 const password = 'unir12345'
6 const host = '172.31.2.28'
7 const port = '2701'
8 const database = 'test'
9
10 // Cadena de conexión a MongoDB (URI)
11 const uri = `mongodb://${username}:${password}@${host}:${port}/${database}`
12 let stringMongoConnection
13
14 // Verificación conexión a MongoDB
15 async function checkMongoDBConnection() {
16   try {
17     const client = new MongoClient(uri)
18     await client.connect()
19     console.log('Conexión exitosa a MongoDB')
20     stringMongoConnection = `Mongo Conectado correctamente a: ${uri}`
21     await client.close()
22   } catch (error) {
23     console.error('Error al conectar a MongoDB:', error)
24     stringMongoConnection = `Error al conectar a MongoDB: ${error}`
25   }
26 }
27 checkMongoDBConnection()
28
29 // Crea una instancia de la aplicación Express
30 const app = express()
31
32 // Configura una ruta básica
33 app.get('/', (req, res) => {

```

Para la creación de la imagen de Mongo se realizó con Packer y se definió sobre una imagen de Amazon Linux la instalación limpia de MONGODB con la creación de un nuevo usuario y DB la cual será la que se usó para la conexión desde NODE

```

1 #!/bin/bash
2 sudo yum update -y
3 # Instalar MongoDB en Rocky Linux
4 sudo tee /etc/yum.repos.d/mongodb-org-7.0.repo << EOF
5 [mongodb-org-7.0]
6 name=MongoDB Repository
7 baseurl=https://repo.mongodb.org/yum/amazon/2023/mongodb-org/7.0/x86_64/
8 gpgcheck=1
9 enabled=1
10 gpgkey=https://www.mongodb.org/static/pgp/server-7.0.asc
11 EOF
12
13 # Instalar MongoDB en Rocky Linux
14 sudo yum install -y mongodb-org
15
16 # Reiniciar el servicio MongoDB para aplicar la configuración
17 sudo systemctl restart mongod
18 sudo systemctl start mongod
19 sudo systemctl enable mongod
20
21 # Crear un usuario y una base de datos con los datos proporcionados
22 mongosh --eval 'db.createUser({user: "christian-unir", pwd: "unir12345", roles: [{role: "readwrite", db: "test"}]})'
23
24 # Configurar MongoDB para aceptar conexiones remotas
25 sudo sed -i 's/bindip: 127.0.0.1/bindip: 0.0.0.0/' /etc/mongod.conf

```

## Criterio 5: Las dos instancias están configuradas y se muestra el Connection String a MongoDB.

Al momento de correr el comando “terraform init” este inicia con la instalación del plugin del provider seleccionado

```

1 # Definición del Provider
2 provider "aws" {
3   region = var.region
4   access_key = var.access_key
5   secret_key = var.secret_key
6 }
7
8 # Aclaración de nuestro SecurityGroup existente en AWS
9 data "aws_security_group" "existing_sg" {
10   name = var.security_group_name
11 }
12
13 # Definición de la instancia a lanzarse de MONGODB
14 resource "aws_instance" "mongo_db" {
15   ami           = var.ami_id_mongo
16   instance_type = var.instance_type
17 }

```

```

PS D:\UNIR DEVOPS\Herramientas DevOps\Terraform-MEANStack\Terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selection by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS D:\UNIR DEVOPS\Herramientas DevOps\Terraform-MEANStack\Terraform>

```

Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

Posteriormente al correr el comando “terraform apply” este empieza con el deployment de las instancias configuradas en “main.tf” y al finalizar nos muestra el ip publica como también el dns de la ip publica de cada instancia lanzada

```

1 # Definición del Provider
2 provider "aws" {
3   region = var.region
4   access_key = var.access_key
5   secret_key = var.secret_key
6 }
7
8 # Aclaración de nuestro SecurityGroup existente en AWS
9 data "aws_security_group" "existing_sg" {
10   name = var.security_group_name
11 }
12
13 # Definición de la instancia a lanzarse de MONGODB
14 resource "aws_instance" "mongo_db" {
15   ami           = var.ami_id_mongo
16   instance_type = "t2.micro"
17   security_groups = [aws_security_group.existing_sg.id]
18   tags = {
19     Name = "mongo-db"
20   }
21 }
22
23 # Definición de la instancia a lanzarse de NODEJS
24 resource "aws_instance" "node_app" {
25   ami           = var.ami_id_nodejs
26   instance_type = "t2.micro"
27   security_groups = [aws_security_group.existing_sg.id]
28   tags = {
29     Name = "node-app"
30   }
31 }
32
33 # Outputs
34 output "mongo_db_public_ip" {
35   value = aws_instance.mongo_db.public_ip
36 }
37
38 output "mongo_db_public_dns" {
39   value = aws_instance.mongo_db.public_dns
40 }
41
42 output "node_app_public_ip" {
43   value = aws_instance.node_app.public_ip
44 }
45
46 output "node_app_public_dns" {
47   value = aws_instance.node_app.public_dns
48 }
49
50 # Outputs con IP publica y DNS de la instancia de MONGO
51 output "mongo_ip_mongo" {
52   value = aws_instance.mongo_db.public_ip
53 }
54
55 output "mongo_dns_mongo" {
56   value = aws_instance.mongo_db.public_dns
57 }
58
59 # Outputs con IP publica y DNS de la instancia de NODE
60 output "node_ip_node" {
61   value = aws_instance.node_app.public_ip
62 }
63
64 output "node_dns_node" {
65   value = aws_instance.node_app.public_dns
66 }
67
68 # Outputs con IP publica y DNS de la instancia de MONGO y NODE
69 output "mongo_ip_node" {
70   value = aws_instance.mongo_db.public_ip
71 }
72
73 output "mongo_dns_node" {
74   value = aws_instance.mongo_db.public_dns
75 }
76
77 output "node_ip_mongo" {
78   value = aws_instance.node_app.public_ip
79 }
80
81 output "node_dns_mongo" {
82   value = aws_instance.node_app.public_dns
83 }
84
85 # Outputs con IP publica y DNS de la instancia de MONGO y NODE
86 output "mongo_ip_mongo_node" {
87   value = aws_instance.mongo_db.public_ip
88 }
89
90 output "mongo_dns_mongo_node" {
91   value = aws_instance.mongo_db.public_dns
92 }
93
94 output "node_ip_mongo_db" {
95   value = aws_instance.node_app.public_ip
96 }
97
98 output "node_dns_mongo_db" {
99   value = aws_instance.node_app.public_dns
100 }

```

Los outputs fueron previamente configurados para que nos muestre en el archivo “outputs.tf”

```

1 # Outputs con IP publica y DNS de la instancia de NODE
2 output "public_ip_node" {
3   value = aws_instance.node_app.public_ip
4 }
5
6 output "public_dns_node" {
7   value = aws_instance.node_app.public_dns
8 }
9
10 # Outputs con IP publica y DNS de la instancia de MONGO
11 output "public_ip_mongo" {
12   value = aws_instance.mongo_db.public_ip
13 }
14
15 output "public_dns_mongo" {
16   value = aws_instance.mongo_db.public_dns
17 }

```

Al momento de ir a la consola de AWS se puede ver claramente que se lanzaron sin problemas las Instancias de MONGODB y NODJS

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
mongo-db-inst...	i-0fe17aba3c73e3b1	Running	t2.micro	2/2 checks passed	No alarms	sa-east-1a	ec2-54-94-114-57-sa-e...	54.94.114.57	-
node-app-inst...	i-00b4bee541e67fd3b	Running	t2.micro	2/2 checks passed	No alarms	sa-east-1a	ec2-15-228-191-21-sa...	15.228.191.21	-

Si ingresamos a la IP publica de NODEJS con el puerto 3000 claramente se puede ver que si se levanto el servidor express y también se conectó correctamente a MONGODB

Instance ID	Instance state	Instance type	Public IPv4 address	Private IPv4 addresses
i-00b4bee541e67fd3b (node-app-instance)	Running	t2.micro	15.228.191.21 (open address)	172.31.2.29



Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

```

40 # ID Imagen de Node
41 variable "ami_id_node" {
42   default = "ami-0185b4f38a3e23455"
43 }
44
45 # ID Imagen de Mongo
46 variable "ami_id_mongo" {
47   default = "ami-0000dcfd53a621ba9"
48 }
49
50 # variable "subnet_id_node" {
51 #   default = "subnet-0a81d25bb0e324be7"
52 # }
53
54 # variable "vpc_id" {
55 #   default = "vpc-0893b451633afb18b"
56 # }

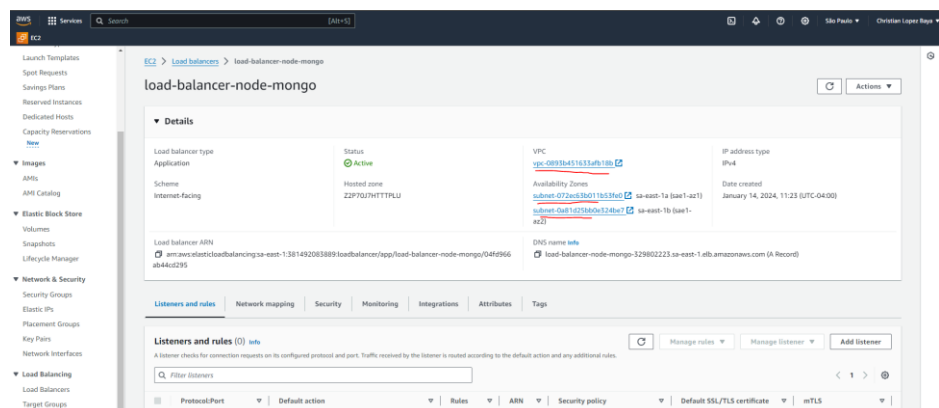
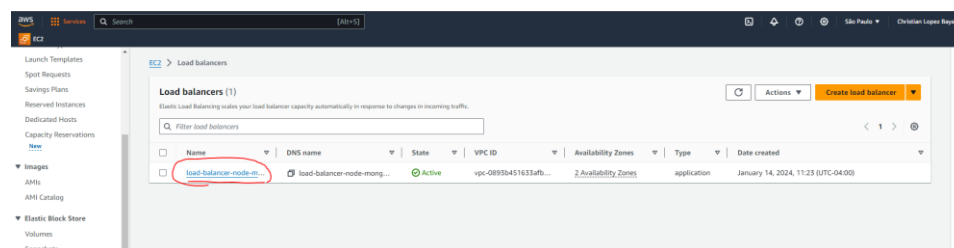
```

El Load Balancer se creaba en AWS apuntando ambas subredes a la misma VPC pero daba errores en procesamiento de tiempo ya que tardaba mucho

```

41 # Definición del balancador de carga
42 resource "aws_lb" "loadbalancing" {
43   name               = "load-balancer-node-mongo"
44   internal            = false
45   load_balancer_type = "application"
46   subnets            = [var.subnet_id, var.subnet_id_node]
47   enable_deletion_protection = false
48 }
49
50 # Definición de grupo de destinos para instancias de Node.js
51 resource "aws_lb_target_group" "node_app" {
52   name     = "node-app-target-group"
53   port     = 3000
54   protocol = "HTTP"
55   vpc_id   = var.vpc_id
56
57   health_check {
58     path       = "/"
59     interval   = 30
60     timeout    = 5
61     healthy_threshold = 2
62     unhealthy_threshold = 2
63   }
64 }

```





Asignatura	Datos del alumno	Fecha
Herramientas DevOps	Apellidos: Lopez Baya	
	Nombre: Christian	

### Criterio 7: Explicación del vídeo completa.

El video completo de la explicación se encuentra en el siguiente link:

[bfz-womh-ssx \(2024-01-14 12\\_13 GMT-4\).mp4](#)