



Herramientas DevOps - MEX DEVOPS

“Despliegue de MEAN multicapa
mediante Terraform

Instructor: MIGUEL ANGEL MUÑOZ ALVARADO

Actividad: MADEO05_ACT3

Equipo 2D

Enero 2025

Introducción

En el ámbito del desarrollo moderno, la infraestructura como código (IaC) se ha convertido en una práctica esencial para gestionar entornos escalables, repetibles y eficientes. Terraform, como herramienta declarativa de IaC, permite automatizar el despliegue de infraestructura en la nube de manera modular y controlada.

En este trabajo se realizará la implementación de un stack MEAN (MongoDB, Express.js, Angular y Node.js) sobre AWS, utilizando Terraform para la gestión de la infraestructura y Packer para la creación de imágenes preconfiguradas (AMIs). La solución sigue principios de infraestructura inmutable, asegurando que cada componente sea desplegado de manera consistente y reproducible.

El stack se estructurará en una arquitectura multicapa, donde los servicios de frontend y backend se ejecutarán en una instancia con Nginx y Node.js, mientras que la base de datos MongoDB se alojará en una instancia separada. Además, se implementarán prácticas recomendadas como la modularización en Terraform, la configuración de grupos de seguridad, el uso de un balanceador de carga y la generación de outputs para la gestión eficiente de los recursos desplegados.

Objetivo

El objetivo de este trabajo es automatizar el despliegue de un stack MEAN en AWS utilizando Terraform y siguiendo un enfoque de infraestructura inmutable. Para ello, se crearán imágenes de máquina (AMIs) preconfiguradas con Packer para el servidor de aplicaciones y la base de datos, permitiendo un despliegue consistente y reproducible. La infraestructura se diseñará de manera modular, separando los componentes en distintos módulos de Terraform para facilitar su gestión y reutilización. Además, se configurarán grupos de seguridad para garantizar un acceso seguro a los servicios, y se implementará un balanceador de carga para optimizar la disponibilidad y escalabilidad de la aplicación. Finalmente, se generará un archivo de outputs en Terraform que proporcionará información relevante sobre los recursos desplegados, como direcciones IP y DNS, asegurando un control eficiente de la infraestructura en la nube.

Índice

Contenido

Introducción	2
Objetivo	2
Índice	3
Creación de usuario de AWS para Terraform	4
Creación de AMIs	5
Creación de Templates de Terraform	5
Creación de los módulos	7
Ejecución de Terraform	8
Outputs de Terraform	8
Demostración de la infraestructura	9
Instrucciones para crear la infraestructura con Packer, Terraform y AWS	12
Conclusiones	14
Bibliografía	15

Creación de usuario de AWS para Terraform

Para iniciar con la creación de las AMIs y la infraestructura en AWS, es necesario contar con una cuenta de servicio con los privilegios adecuados para interactuar con servicios como EC2 y VPC.

Se creó una cuenta de servicio denominada **terraform-user**, la cual posee los permisos necesarios para construir AMIs con Packer y aprovisionar la infraestructura definida en Terraform. Para permitir la interacción con la CLI de AWS desde el dispositivo de aprovisionamiento, se generaron claves de acceso para esta cuenta de servicio.

Adicionalmente, se creó una key pair para las instancias de EC2 con el fin de facilitar el acceso a las mismas.

Users (1) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Q Search

<input type="checkbox"/>	User name	▲	Path	▼	Group:	▼	Last activity	▼
<input type="checkbox"/>	terraform-user		/		0		-	

Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
<div><div></div></div>	<div><div></div> Show</div>

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key p

Key pair name - required

packer

Creación de AMIs

Configuración y creación de AMIs con Packer

AMI con Nginx y Node.js

Se implementó un CRUD sencillo utilizando Angular y Express.js. Para ello, se tomó una aplicación del repositorio de MongoDB con el stack MEAN, utilizando Ubuntu como sistema base. Se descargaron todas las dependencias necesarias para compilar y desplegar tanto el frontend como el backend dentro de la AMI generada para AWS.

AMI con MongoDB

Se configuró e implementó una base de datos MongoDB para el CRUD. Fue necesario instalar y configurar MongoDB, crear un usuario y una contraseña, y compartir las credenciales junto con la cadena de conexión con la AMI de la aplicación MEAN. Esta imagen es responsable de inicializar la base de datos y los documentos BSON al iniciarse.

Creación de Templates de Terraform

Estructura Modular de Terraform

- Módulo de Red: VPC, subnets e Internet Gateway.
- Módulo de Seguridad: Grupos de seguridad para la aplicación y la base de datos.
- Módulo de Instancias: Implementación de las AMIs generadas con Packer.
- Módulo de Balanceador de Carga: Configuración del ALB y target groups.

Estructura del Proyecto Terraform

mean-terraform-deployment-aws-unir-devop/

└─ main.tf	# Template principal que llama a los módulos
└─ variables.tf	# Variables globales del proyecto
└─ outputs.tf	# Salidas globales del proyecto
└─ terraform.tfvars	# Valores de las variables (si es necesario)
└─ modules/	# Carpeta donde van todos los módulos
└─ network/	# Módulo de red (VPC, subnets, Internet Gateway)

```

| | ├── main.tf      # Configuración principal del módulo de red
| | ├── variables.tf  # Variables específicas del módulo de red
| | ├── outputs.tf    # Salidas del módulo de red (vpc_id, subnets, etc.)
| ├── security/      # Módulo de seguridad (grupos de seguridad)
| | ├── main.tf      # Configuración principal del módulo de seguridad
| | ├── variables.tf  # Variables específicas del módulo de seguridad
| | ├── outputs.tf    # Salidas del módulo de seguridad (sg_ids)
| ├── instances/     # Módulo de instancias (AMIs generadas con Packer)
| | ├── main.tf      # Configuración principal del módulo de instancias
| | ├── variables.tf  # Variables específicas del módulo de instancias
| | ├── outputs.tf    # Salidas del módulo de instancias (instance_ids)
| ├── load_balancer/ # Módulo de balanceador de carga
| | ├── main.tf      # Configuración principal del módulo de balanceador de carga
| | ├── variables.tf  # Variables específicas del módulo de balanceador de carga
| | ├── outputs.tf    # Salidas del módulo de balanceador de carga (ALB, listeners)

```

Para garantizar una infraestructura organizada y reutilizable, la implementación se dividirá en distintos módulos de Terraform, cada uno encargado de gestionar un componente específico del despliegue.

Cada módulo se ejecuta en un orden específico debido a sus dependencias. La red se configura primero, seguida de la seguridad, las instancias y finalmente el balanceador de carga.

El módulo de red establecerá la infraestructura base, incluyendo la creación de una VPC, subnets, tabla de enrutamiento y un Internet Gateway para permitir la conectividad externa.

El módulo de seguridad definirá los grupos de seguridad necesarios para la aplicación y la base de datos, asegurando que solo los servicios autorizados puedan comunicarse entre sí y con el exterior, mediante los puertos adecuados.

El módulo de instancias gestionará la creación de las máquinas virtuales utilizando las AMIs previamente generadas con Packer, garantizando la coherencia y estabilidad del entorno.

Finalmente, el módulo de balanceador de carga permitirá distribuir el tráfico entrante de manera eficiente, mejorando la disponibilidad y escalabilidad de la aplicación. Esta modularización facilitará la gestión y mantenimiento de la infraestructura, permitiendo futuras modificaciones y expansiones de manera controlada.

Creación de los módulos

Se crea el template principal para la ejecución de los módulos en el orden adecuado y los módulos necesarios para crear la infraestructura en AWS.

Los módulos se ejecutan desde un archivo MAIN principal en la raíz del proyecto y los ejecuta en un orden específico debido a que cada módulo tiene dependencia de la información que se genera de los módulos anteriores. Se usan los archivos main, variables, terraform.tfvars y output para la raíz del proyecto y para cada módulo se incluyen los mismos archivos con excepción de terraform.tfvars.

El primer módulo es de la red debido a que primero debemos crear una VPC donde estarán gran parte de nuestros recursos, así como 3 subredes, una tabla de enrutamiento, su configuración y un internet Gateway.

A continuación, se ejecuta el módulo de seguridad para definir los grupos de seguridad para las máquinas virtuales que estarán en subredes públicas y privadas. El siguiente módulo es el de las instancias y necesita algunos datos ya generados en módulos anteriores y recuperados con las variables de output como el id de la VPC, el id de las subredes, el id de los grupos de seguridad para crear las dos instancias mínimas necesarias para el proyecto.

Finalmente se ejecuta el módulo del balanceador de carga y se configura el balanceador de carga y el target group para la instancia MEAN.

Al final de la ejecución veremos los datos que declaramos en el output de la raíz del proyecto con la información general de nuestra infraestructura y podremos interactuar con la infraestructura aprovisionada y correctamente configurada en AWS.

Ejecución de Terraform

```
$ terraform init
```

```
$ terraform plan
```

```
# Para guardar el log
```

```
$ terraform apply | tee terraform_output.log
```

```
# Al final destruimos la infraestructura si ya no la necesitamos
```

```
$ terraform destroy
```

Outputs de Terraform

Fichero output.tf que contenga:

- IP públicas de cada nodo
- IP privadas de cada nodo
- DNS del balanceador
- IP pública instance MongoDB

Se creo el archivo terraform_output.log para guardar el log de ejecución de Terraform así como su log final. Este es un ejemplo del resultado del Output final.

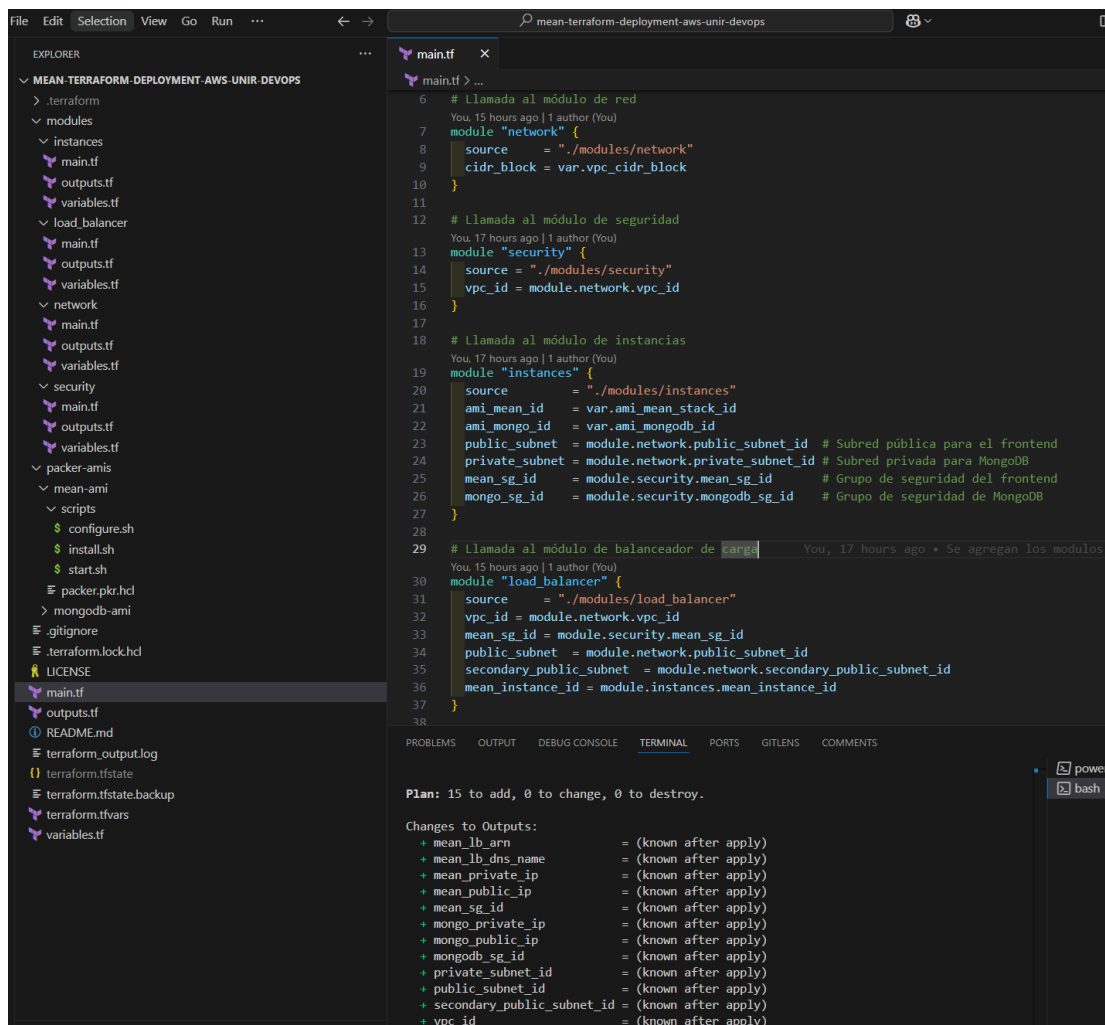
```
mean_lb_arn = "arn:aws:elasticloadbalancing:us-east-1:755944857852:loadbalancer/app/mean-lb/2381fd9f13d1ef02"
mean_lb_dns_name = "mean-lb-2137615682.us-east-1.elb.amazonaws.com"
mean_private_ip = "10.0.1.246"
mean_public_ip = "3.83.254.55"
mean_sg_id = "sg-07d5f638c841c80ab"
mongo_private_ip = "10.0.2.65"
mongo_public_ip = ""
mongodb_sg_id = "sg-0e62eed69bf39ac0f"
private_subnet_id = "subnet-074582fd0da1d62a3"
public_subnet_id = "subnet-04f4129c41f98e4e5"
secondary_public_subnet_id = "subnet-0995022b660919b14"
vpc_id = "vpc-01ffe6c025cb568d2"
```


Enlace al archivo completo de log de Terraform:

https://github.com/Lordbear117/mean-terraform-deployment-aws-unir-devops/blob/main/terraform_output.log

Demostración de la infraestructura

Se ejecutaron los templates de Terraform desde una PC con Windows 11 Pro.



```
File Edit Selection View Go Run ... mean-terraform-deployment-aws-unir-devops
EXPLORER
MEAN-TERRAFORM-DEPLOYMENT-AWS-UNIR-DEVOPS
  .terraform
  modules
  instances
    main.tf
    outputs.tf
    variables.tf
  load_balancer
    main.tf
    outputs.tf
    variables.tf
  network
    main.tf
    outputs.tf
    variables.tf
  security
    main.tf
    outputs.tf
    variables.tf
  packer-amis
  mean-ami
  scripts
    $ configure.sh
    $ install.sh
    $ start.sh
  packer.pkr.hcl
  > mongodb-ami
  .gitignore
  .terraform.lock.hcl
  LICENSE
  main.tf
  outputs.tf
  README.md
  terraform_output.log
  terraform.tfstate
  terraform.tfstate.backup
  terraform.tfvars
  variables.tf

main.tf
6 # Llamada al módulo de red
7 You, 15 hours ago | 1 author (You)
8 module "network" {
9   source = "../modules/network"
10   cidr_block = var.vpc_cidr_block
11 }
12 # Llamada al módulo de seguridad
13 You, 17 hours ago | 1 author (You)
14 module "security" {
15   source = "../modules/security"
16   vpc_id = module.network.vpc_id
17 }
18 # Llamada al módulo de instancias
19 You, 17 hours ago | 1 author (You)
20 module "instances" {
21   source = "../modules/instances"
22   ami_mean_id = var.ami_mean_stack_id
23   ami_mongo_id = var.ami_mongodb_id
24   public_subnet = module.network.public_subnet_id # Subred pública para el frontend
25   private_subnet = module.network.private_subnet_id # Subred privada para MongoDB
26   mean_sg_id = module.security.mean_sg_id # Grupo de seguridad del frontend
27   mongo_sg_id = module.security.mongodb_sg_id # Grupo de seguridad de MongoDB
28 }
29 # Llamada al módulo de balanceador de carga
30 You, 15 hours ago | 1 author (You)
31 module "load_balancer" {
32   source = "../modules/load_balancer"
33   vpc_id = module.network.vpc_id
34   mean_sg_id = module.security.mean_sg_id
35   public_subnet = module.network.public_subnet_id
36   secondary_public_subnet = module.network.secondary_public_subnet_id
37   mean_instance_id = module.instances.mean_instance_id
38 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS
Plan: 15 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ mean_lb_arn = (known after apply)
+ mean_lb_dns_name = (known after apply)
+ mean_private_ip = (known after apply)
+ mean_public_ip = (known after apply)
+ mean_sg_id = (known after apply)
+ mongo_private_ip = (known after apply)
+ mongo_public_ip = (known after apply)
+ mongodb_sg_id = (known after apply)
+ private_subnet_id = (known after apply)
+ public_subnet_id = (known after apply)
+ secondary_public_subnet_id = (known after apply)
+ vpc_id = (known after apply)
```

Se crearon dos AMIs para las instancias EC2 de la aplicación y la base de datos.

Amazon Machine Images (AMIs) (2) [Info](#)

Owned by me ▼		Find AMI by attribute or tag	
<input type="checkbox"/>	Name ✎	AMI name ▼	AMI ID
<input type="checkbox"/>		mean-stack-ami	ami-0668ad107f101ba2f
<input type="checkbox"/>		mongodb-ami	ami-0094f7f158e63c722

Se implementaron las instancias EC2 necesarias.

Instances (2) Info					Last updated ⌚ less than a minute ago
Find Instance by attribute or tag (case-sensitive)					Running ▼
<input type="checkbox"/>	Name ✎	Instance ID	Instance state	Instance type	
<input type="checkbox"/>	MongoInstance	i-00776630e93503b2e	Running 🔍 🔍	t2.micro	
<input type="checkbox"/>	MeanInstance	i-046b640767f288028	Running 🔍 🔍	t2.medium	

Se configuraron los grupos de seguridad para las instancias en subredes públicas y privadas.

Security Groups (7) [Info](#)

Find resources by attribute or tag			
<input type="checkbox"/>	Name ▼	Security group ID ▼	Security group name ▼
<input type="checkbox"/>	MongoDBSecurityGr...	sg-0e62eed69bf39ac0f	terraform-2025020307132862800000...
<input type="checkbox"/>	MeanSecurityGroup	sg-07d5f638c841c80ab	terraform-2025020307132426390000...

Se creó un balanceador de carga y un target group.

mean-lb

▼ Details			
Load balancer type Application	Status Active	VPC vpc-01ffe6c025cb568d2 🔍	Load balance IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0995022b660919b14 🔍 us-east-1c (use1-az6) subnet-04f4129c41f98e4e5 🔍 us-east-1a (use1-az2)	Date created February 3, 2025
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:755944857852:loadbalancer/app/mean-lb/2381fd9f13d1ef02		DNS name Info mean-lb-2137615682.us-east-1.elb.amazonaws.com (A Record)	

mean-target-group

<div>Details</div> <div>arn:aws:elasticloadbalancing:us-east-1:755944857852:targetgroup/mean-target-group/b92e1c889ef50608</div>					
<div>Target type</div> <div>Instance</div>		<div>Protocol : Port</div> <div>HTTP: 80</div>	<div>Protocol version</div> <div>HTTP1</div>		<div>VPC</div> <div>vpc-01ffe6c025cb568d2</div>
<div>IP address type</div> <div>IPv4</div>		<div>Load balancer</div> <div>mean-lb</div>			
<div>1</div> <div>Total targets</div>	<div><div>✔ 1</div><div>Healthy</div><div>0 Anomalous</div></div>	<div><div>✘ 0</div><div>Unhealthy</div></div>	<div><div>⊖ 0</div><div>Unused</div></div>	<div><div>⌚ 0</div><div>Initial</div></div>	<div><div>⌚ 0</div><div>Draining</div></div>

Se implementó una VPC con sus respectivas subredes.

Your VPCs (2) Info

<input type="checkbox"/>	Name	VPC ID	State	Block Public...	IPv4 CIDR
<input type="checkbox"/>	MyVPC	vpc-02d145498ae55089c	✔ Available	⊖ Off	10.0.0.0/16

Se crean 2 subredes públicas para el ALB y una privada para la base de datos.

Subnets (9) Info

<input type="checkbox"/>	Name ▼	Subnet ID ▼	State ▼	VPC ▼	Bloc... ▼	IPv4 CIDR
<input type="checkbox"/>	SecondaryPublicS...	subnet-0995022b660919b...	Available	vpc-01ffe6c025c...	Off	10.0.3.0/24
<input type="checkbox"/>	PublicSubnet	subnet-04f4129c41f98e4e5	Available	vpc-01ffe6c025c...	Off	10.0.1.0/24
<input type="checkbox"/>	PrivateSubnet	subnet-074582fd0da1d62a3	Available	vpc-01ffe6c025c...	Off	10.0.2.0/24

Route tables (3) Info

☐

Name

▼

Route table ID

▼

Explicit subnet associations

☐

PublicRouteTable

[rtb-01df54ef561171770](#)

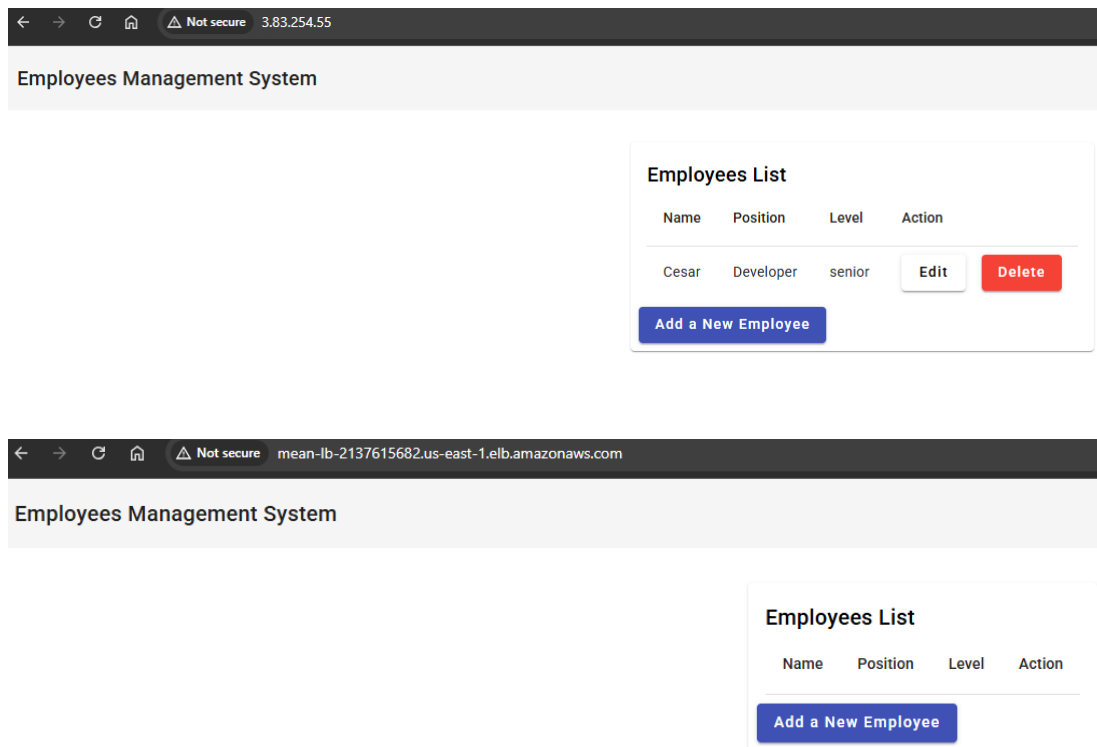
[subnet-04f4129c41f98e4e5](#) / [PublicSubnet](#)

Internet gateways (2) Info

Search

<input type="checkbox"/>	Name	Internet gateway ID	State	VPC ID
<input type="checkbox"/>	MyInternetGateway	igw-0418996867d47947f	✔ Attached	vpc-01ffe6c025cb568d2 MyVPC

Se verificó la comunicación de la aplicación con la base de datos mediante HTTP y balanceo de carga.



Instrucciones para crear la infraestructura con Packer, Terraform y AWS

1. Crear un usuario de AWS con los permisos para EC2 y VPC, crear un key pair para las instancias, se puede llamar **"Packer"** para no modificar nada.
2. Ejecutar **aws configure** para introducir la contraseñas del usuario de servicio de AWS.
3. Generar las AMIs con Packer. Para ello se ejecuta el comando:

```
packer build packer.pkr.hcl
```

Este se ejecuta en la raíz de cada una de las amis. Existe la carpeta llamada mean-ami y mongodb-ami.
4. Se debe guardar el id de las amis en el archivo terraform.tfvars para sustituir el id existente por el nuevo que se generó.
5. Ejecutamos terraform init para descargar las dependencias, después terraform plan para revisar el código de los templates y la

infraestructura que se aprovisionara, finalmente, si todo está correcto, se ejecuta terraform apply para aprovisionar la infraestructura en AWS.

```
black@Bear-Laptop MINGW64 ~/mean-terraform-deployment-aws-unir-devops/packer-amis/mean-ami (main)  
$ packer build packer.pkr.hcl
```

```
black@Bear-Laptop MINGW64 ~/mean-terraform-deployment-aws-unir-devops/packer-amis/mongodb-ami (main)  
$ packer build packer.pkr.hcl
```

```
black@Bear-Laptop MINGW64 ~/mean-terraform-deployment-aws-unir-devops (main)  
$ terraform apply
```

Conclusiones

Este proyecto demostró la viabilidad y eficiencia de la automatización del aprovisionamiento de infraestructura en AWS utilizando Packer y Terraform. Se logró una implementación modular que facilita la escalabilidad y el mantenimiento de los recursos desplegados. Además, la configuración adecuada de los módulos permitió un despliegue estructurado y seguro, asegurando que cada componente dependiera de la información generada por los anteriores.

El uso de AMIs personalizadas optimizó el tiempo de despliegue, ya que las instancias se iniciaron con las configuraciones necesarias predefinidas. La implementación del balanceador de carga garantizó la disponibilidad y distribución eficiente del tráfico, lo que resulta fundamental en arquitecturas escalables.

En futuras iteraciones, podría mejorarse la configuración de seguridad mediante la implementación de VPNs, el uso de IAM roles más refinados para cada servicio y el uso de un manejador de secretos para las credenciales sensibles como la contraseña de la base de datos. También sería recomendable automatizar el monitoreo de la infraestructura con herramientas como Prometheus o CloudWatch para una supervisión más efectiva.

Finalmente, este proyecto proporciona una base sólida para futuras mejoras y ampliaciones en entornos productivos, asegurando un modelo de infraestructura como código eficiente, repetible y confiable.

Bibliografía

Para desarrollar este proyecto, se consultaron las siguientes referencias:

1. Mongodb-Developer. (s. f.). GitHub - mongodb-developer/mean-stack-example: Sample CRUD application built with the MEAN stack. GitHub.
<https://github.com/mongodb-developer/mean-stack-example>
2. Lordbear. (s. f.). GitHub - Lordbear117/mean-stack-example: Sample CRUD application built with the MEAN stack. GitHub.
<https://github.com/Lordbear117/mean-stack-example>
3. Terraform by HashiCorp. (s. f.). Terraform By HashiCorp.
<https://www.terraform.io/>
4. Packer by HashiCorp. (2024, 22 abril). Packer By HashiCorp.
<https://www.packer.io/>
5. ¿Qué es la infraestructura como código? - Explicación de IaC - AWS. (s. f.). Amazon Web Services, Inc. <https://aws.amazon.com/es/what-is/iac/>
6. Module creation - recommended pattern | Terraform | HashiCorp Developer. (s. f.). Module Creation - Recommended Pattern | Terraform | HashiCorp Developer.
<https://developer.hashicorp.com/terraform/tutorials/modules/pattern-module-creation>
7. Team, M. D. (s. f.). Install MongoDB Community Edition on Ubuntu. MongoDB Manual v8.0.
<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/>
8. NGINX Reverse Proxy. (s. f.). NGINX Documentation.
<https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>