

# mundosE

PIN Final

devops 2403

**Profesor:**

Guazzardo, Marcelo

**Grupo 1:**

- Cabral, Damian Esteban
- Ferreira, Alexander
- Gonzalez, Claudio
- Huataquispe Poma, Arnold
- Rico, Cristian

<b>devops 2403</b>	<b>Grupo 1</b>  <b>PIN Final</b>	<b>mundosE</b>
--------------------	--	----------------

<b>INTRODUCCIÓN</b>	<b>4</b>
<b>GITHUB</b>	<b>6</b>
COMMITS	6
REPOSITORY SECRETS	7
<b>AWS</b>	<b>8</b>
Configuraciones generales	8
IAM	8
Billing and Cost Management	10
Estructura de directorios	10
EC2 (Bastion)	12
KEY PAIR	12
OTROS ARCHIVOS	13
user_data.sh	13
ec2-admin.json	15
.gitignore	19
TERRAFORM	19
main.tf	19
backend.tf	21
Bucket en S3	21
iam.tf	22
variables.tf	23
outputs.tf	23
providers.tf	23
terraform.tfvars	24
vpc y subnet id	24
GITHUB ACTIONS	26
Workflow	26
Ejecución workflow	27
AWS (EC2)	29
Validación de creación de instancia	29
Permisos	30
Acceso a instancia	31
Ubuntu Version	32
Paquetes	32
Elastic IP	32
EKS	34
<b>DESPLIEGUE</b>	<b>34</b>
Script crear_cluster.sh	34
Preparación	37

<b>devops 2403</b>	<b>Grupo 1</b>  <b>PIN Final</b>	<b>mundosE</b>
--------------------	--	----------------

Ejecución de script y verificaciones	37
Permisos	40
configMap	40
configmap.sh	41
Test IAM User access	46
NGINX	50
Despliegue	50
Comprobaciones:	54
EBS CSI Driver	55
Crear rol	55
Instalación	56
Validar funcionamiento	57
MONITOREO	62
PROMETHEUS	62
Instalación	62
Configurar NodePort (opcional)	65
Validación:	66
Troubleshooting pod alertmanager	67
Port forward access:	69
Node port access:	72
GRAFANA	74
Instalación	74
Verificación y configs:	79
EKS	79
Web	80
Acceso	80
Datasource:	81
Métricas de Prometheus	82
Dashboards	82
CLEAN	86
REVISIÓN	88
Conclusiones	88
Topología general	89
Consumo	90

<b>devops 2403</b>	<b>Grupo 1</b>  <b>PIN Final</b>	<b>mundosE</b>
--------------------	--	----------------

# INTRODUCCIÓN

Este proyecto, denominado PIN FINAL, ha sido diseñado para desplegar y gestionar un clúster de Kubernetes en AWS utilizando Elastic Kubernetes Service (EKS).

Su propósito es integrar diferentes herramientas vistas durante la diplomatura de DevOps de MundosE de la clase 2403.

Utilizamos AWS para crear recursos como EC2, EKS, IAM, CloudFormation, S3, Load Balancer, etc... y herramientas de monitoreo opensource para crear una infraestructura optimizada con enfoque DevOps.

El proyecto se compone de los siguientes aspectos principales:

- Aprovisionamiento de infraestructura:
  - Se crea una instancia EC2 en AWS con Github Actions mediante Terraform. El mismo funcionará como Bastion Host para la administración del entorno.
- Creación y configuración de un clúster Kubernetes (EKS):
  - Se utiliza eksctl en un script para desplegar un clúster de Kubernetes administrado con tres nodos.
- Gestión de accesos y permisos:
  - Se configura IAM y el configmap/aws-auth para administrar usuarios y accesos al clúster.
- Monitoreo y visualización de métricas: Se despliega Prometheus para recolectar métricas del clúster o sus pods, y Grafana para visualizarlas a través de dashboards personalizables

## Herramientas utilizadas

- Terraform: para despliegue de EC2 en aws
- Visual Code: Para armar estructura de directorios, scripts y archivos de configuración
- EKS: Servicio de Kubernetes en AWS para el despliegue de la infraestructura requerida.
- EC2: Se utilizará una instancia como Bastión Host donde se instalará y utilizarán herramientas de gestión como AWS CLI, kubectl, eksctl, Docker, Helm.
- S3: Para crear un bucket y colocar el archivo de tfstate de terraform
- GitHub: Repositorio de código y config files para versionado y despliegue mediante workflows con Github Actions.

<b>devops 2403</b>	<b>Grupo 1</b> <b>PIN Final</b>	<b>mundosE</b>
--------------------	------------------------------------	----------------

- Nginx: Despliegue de un servicio de nginx de prueba mediante un deployment de RollingUpdate y un Service LoadBalancer.
- Prometheus: Herramienta para la recolección de métricas del cluster de Kubernetes.  
 URL Interna: <http://prometheus.monitoreo.svc.cluster.local:8080>  
 URL Externa: http://nodeip:32000
- Grafana: Plataforma para la visualización de las métricas recolectadas por Prometheus.  
 Importación de Dashboard ID: 3119  
 URL Externa: <http://aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com/>

<b>devops 2403</b>	<b>Grupo 1</b> <b>PIN Final</b>	<b>mundosE</b>
--------------------	------------------------------------	----------------

## GITHUB

Creamos un repositorio en GitHub para manejar el control de versiones, mantener un ambiente colaborativo y poder manejar el CI/CD con ayuda de GitHub Actions.

Repositorio utilizado: <https://github.com/dec-wil/mundose.pinfinal.grupo1>

Inicializamos el repositorio y creamos la rama dev en nuestro repositorio

```
# Creacion de Rama principal
git checkout -b main #Crea una nueva rama llamada main y te posiciona en ella.
git add . #Añade todos los cambios y archivos al area de staging.
git commit -m "Main initial commit" #Realiza un commit con el mensaje indicado.
git push -u origin main #Envía la rama main al repositorio remoto y la marca como rama por
defecto para futuros push.
```

```
# Creacion de Rama de desarrollo
git checkout -b dev #Crea y cambia a una nueva rama llamada dev basada en la rama actual
(main).
```



# AWS

## Configuraciones generales

### IAM

Se crean usuarios **nominales** para el acceso a la consola para auditoría. También se permite crear ak/sk a los usuarios para que puedan utilizar la línea de comandos. Los mismos también fueron agregados a un grupo de usuarios admin-users con políticas. Asimismo, se crea el usuario terraform sin acceso a la consola y con claves ak/sk

User name	MFA	Password age	Console access
aferreira	Virtual	7 days	Enabled
ahuataspique	Virtual	7 days	Enabled
cgonzalez	Virtual	7 days	Enabled
crico	Virtual	7 days	Enabled
dcabral	Virtual	7 days	Enabled
terraform	-	-	Disabled

Se crea una AK/SK sobre el usuario terraform (sin acceso a la consola)

También creamos un grupo con algunas políticas

Group name	Users	Permissions
admin-users	5	Defined



**admin-users** Info

### Summary

User group name	Creation time
admin-users	February 25, 2025, 16:25 (UTC-03:00)

**Users** (5)    **Permissions**    **Access Advisor**

#### Permissions policies (2) Info

You can attach up to 10 managed policies.

**Filter by:** All type

<input type="checkbox"/> Policy name	Type
<input type="checkbox"/> <a href="#">AdministratorAccess</a>	AWS managed - job function
<input type="checkbox"/> <a href="#">AssumeRoleCustomPolicy</a>	Customer inline

**AssumeRoleCustomPolicy**

```

1 [{}]
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "sts:AssumeRole",
7       "Resource": "arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole"
8     }
9   ]
10 ]

```

Como se ve en la imagen se agrega la siguiente custom inline policy. Se utilizará para que los usuarios IAM puedan acceder al cluster del EKS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole"
    }
  ]
}
```

devops 2403	Grupo 1 PIN Final	mundosE
-------------	----------------------	---------

## Billing and Cost Management

The screenshot shows the AWS Billing and Cost Management console. In the top navigation bar, 'Billing and Cost Management' is selected. Below it, the 'Budgets' section is shown with the title 'Budgets (1)'. A table lists one budget entry:

Name	Thresholds	Budget
<a href="#">My Monthly Cost Budget</a>	<span>OK</span>	\$100.00

## Estructura de directorios

Estructura de directorios en github y archivos para el despliegue de EC2, EKS y PODs de NGINX, Prometheus y Grafana.

El EC2 se desplegará mediante terraform, mientras que el EKS y sus pods mediante scripts, línea de comando y config files.

```

mundose.pinfinal.grupo1/
├── .github/
│   └── workflows/
│       └── deploy.yml
└── aws/
    └── terraform/
        └── ec2/
            ├── policies/
            │   └── ec2-admin.json
            ├── scripts/
            │   └── user_data.sh
            ├── main.tf      # Configuración de EC2
            ├── variables.tf # Definición de variables
            ├── outputs.tf   # Valores de salida
            ├── providers.tf # Configuración de AWS
            ├── terraform.tfvars # Valores específicos
            └── scripts/

```

```
|           └── user_data.sh # Script con las herramientas
|   └── eks/
|       ├── 00-eks
|       |   ├── 00-crear_cluster.sh
|       |   ├── 01-configmap.sh
|       |   ├── 02-change-to-iamuser.sh
|       |   ├── 03-eks-nodes-scale-config.sh
|       |   └── EKSIAMAdminAccessRole.yaml
|       ├── 01-nginx/
|       |   ├── nginx.sh
|       |   ├── nginx-service.yaml
|       |   └── nginx-pod.yaml
|       ├── 02-ebs-driver-tests/
|       |   ├── 01-test-create-volume.sh
|       |   ├── 02-storageclass.yaml
|       |   ├── 03-pvc.yaml
|       |   ├── 04-pod-ebs-test.yaml
|       |   └── 05-verifycreation.sh
|       ├── 03-prometheus/
|       |   └── prometheus_install.sh
|       └── 04-grafana/
|           ├── grafana.yaml
|           └── grafana_install.sh
└── keys/          # Se ignora con .gitignore (NO SE SUBE A GIT)
└── .gitignore      # Ignora claves y datos sensibles
└── README.md      # Documentación del proyecto
```

devops 2403	Grupo 1 PIN Final	mundosE
-------------	----------------------	---------

## EC2 (Bastion)

Amazon EC2 (Elastic Compute Cloud) es un servicio web de AWS que proporciona capacidad de cómputo escalable en la nube. En otras palabras, permite lanzar y administrar servidores virtuales (llamados "instancias") bajo demanda, facilitando:

- **Flexibilidad:** Puedes elegir entre diferentes tipos de instancias, tamaños, sistemas operativos y configuraciones para satisfacer necesidades específicas.
- **Escalabilidad:** Permite aumentar o reducir la capacidad de cómputo de forma rápida según la demanda de la aplicación.
- **Pago por uso:** Solo pagas por el tiempo y la capacidad que utilizas.
- **Integración:** Se integra con otros servicios de AWS para construir soluciones completas y seguras.

Para el despliegue de esta instancia de EC2 utilizaremos Terraform y Github Actions, tambien utilizamos un bucket de S3 para guardar el archivo de estado

## KEY PAIR

Primero creamos un par de claves pública / privada para acceder de forma segura a nuestra instancia:

```
PS E:\Cursos\MUNDOSE\DevOps\PIN FINAL> ssh-keygen -t rsa -b 4096 -m PEM -f pin.pem -N ""
Generating public/private rsa key pair.
Your identification has been saved in pin.pem
Your public key has been saved in pin.pem.pub
The key fingerprint is:
SHA256:B5oW4hx1cm0t803b/Canj+fjsH354eB5s6l0cJaNXeU dec@dec-prd
The key's randomart image is:
+--[RSA 4096]----+
|   o +o .   .|
|   . + .+..   ..|
|   o . .....   E|
|   o o + . . = .|
|   o + S ...= o |
|       . .++   |
|       .+o..   |
|       o OBB   |
|       .**#@    |
+---[SHA256]-----+
```

## OTROS ARCHIVOS

### user\_data.sh

Archivo utilizado para la instalación de paquetes durante el proceso de despliegue y primer inicio de la instancia. Útil para automatizar la instalación de herramientas de gestión.

```
#!/bin/bash
# Habilita el modo "exit on error": si algún comando falla, el script se detiene inmediatamente.
set -e

# =====
# Actualizar paquetes del sistema
# =====
# Se actualizan los índices de paquetes y se actualizan los paquetes instalados a sus versiones más
recientes.
apt-get update -y && apt-get upgrade -y

# =====
# Instalar AWS CLI
# =====
# AWS CLI es la herramienta de línea de comandos para interactuar con los servicios de Amazon Web
Services.
apt install -y awscli

# =====
# Instalar Docker
# =====
# Docker es una plataforma para desarrollar, enviar y ejecutar aplicaciones en contenedores.
apt install -y docker.io
# Inicia el servicio de Docker.
systemctl start docker
# Habilita Docker para que se inicie automáticamente al arrancar el sistema.
systemctl enable docker
# Agrega el usuario 'ubuntu' al grupo 'docker' para poder ejecutar comandos Docker sin utilizar sudo.
usermod -aG docker ubuntu

# =====
# Instalar kubectl
# =====
# kubectl es la herramienta de línea de comandos para interactuar con clústeres de Kubernetes.
# Se descarga la última versión estable, se le da permisos de ejecución y se mueve a /usr/local/bin
para que esté en el PATH.
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
mv kubectl /usr/local/bin/
```

```
# =====
# Instalar Helm
# =====
# Helm es el gestor de paquetes para Kubernetes, que facilita la instalación y gestión de aplicaciones en clústeres.
# Se descarga y ejecuta el script oficial de instalación de Helm 3.
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# =====
# Instalar eksctl
# =====
# eksctl es la herramienta de línea de comandos para crear y gestionar clústeres en Amazon EKS (Elastic Kubernetes Service).
# Se descarga la última versión, se extrae el binario y se mueve a /usr/local/bin para que esté disponible en el PATH.
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
mv /tmp/eksctl /usr/local/bin
# Muestra la versión instalada de eksctl para confirmar que la instalación fue exitosa.
eksctl version

# =====
# Instalar Docker Compose
# =====
# Docker Compose es una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores.
# Se descarga la última versión desde GitHub, se asignan permisos de ejecución y se verifica la instalación.
curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
docker-compose --version

# =====
# Instalar Terraform
# =====
# Terraform es una herramienta de infraestructura como código, que permite definir, provisionar y gestionar infraestructura de forma declarativa.
# Se instalan dependencias necesarias, se agrega la llave GPG oficial de HashiCorp, se añade el repositorio oficial de HashiCorp,
# se actualizan los índices de paquetes y se instala Terraform.
apt-get install -y gnupg software-properties-common
curl -fsSL https://apt.releases.hashicorp.com/gpg | apt-key add -
apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
apt-get update -y && apt-get install -y terraform
# Muestra la versión instalada de Terraform para confirmar que la instalación fue exitosa.
```

```
terraform version

# =====
# Instalar aws cli v2
# =====

sudo apt install unzip -y
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip" && unzip awscliv2.zip && sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update
aws --version
```

### ec2-admin.json

Permisos del rol creado ec2-admin basado en el principio de seguridad “Least privileges”.

Creando este permiso permitirá que desde nuestra instancia se pueda crear y gestionar el cluster de EKS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation>CreateStack",
                "cloudformation>DeleteStack",
                "cloudformation>UpdateStack",
                "cloudformation>DescribeStacks",
                "cloudformation>ListStacks",
                "cloudformation>DescribeStackEvents",
                "cloudformation>ContinueUpdateRollback",
                "cloudformation>GetTemplate",
                "cloudformation>DescribeStackResources",
                "cloudformation>CreateChangeSet",
                "cloudformation>DescribeChangeSet",
                "cloudformation>ExecuteChangeSet",
                "cloudformation>GetTemplateSummary"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks>CreateNodegroup",
                "eks>UpdateNodegroupConfig",
                "eks>DeleteNodegroup",
                "eks>ListNodegroups",
                "eks>DescribeNodegroup",
                "eks>ListFargateProfiles",
                "eks>ListContainerImages"
            ]
        }
    ]
}
```

```
"eks:DescribeFargateProfile",
"eks>CreateAddon",
"eks:UpdateAddon",
"eks>DeleteAddon",
"eks:DescribeAddon",
"eks:DescribeAddonVersions",
"eks>ListAddons",
"eks:DescribeAddonConfiguration",
"eks>ListUpdates",
"eks:DescribeUpdate",
"eks>CreateCluster",
"eks>DeleteCluster",
"eks:UpdateClusterVersion",
"eks:UpdateClusterConfig",
"eks>ListClusters",
"eks:DescribeCluster",
"eks:DescribeClusterVersions",
"eks:AssociateEncryptionConfig",
"eks:DescribeEncryptionConfig",
"eks:AssociateIdentityProviderConfig",
"eks:DescribeIdentityProviderConfig",
"eks:DisassociateIdentityProviderConfig",
"eks>ListIdentityProviderConfigs",
"eks:TagResource",
"eks:UntagResource",
"eks:AccessKubernetesApi"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"iam>CreateRole",
"iam:AttachRolePolicy",
"iam:DetachRolePolicy",
"iam>ListAttachedRolePolicies",
"iam>DeleteRole",
"iam:TagRole",
"iam:PassRole",
"iam:GetRole",
"iam:GetRolePolicy",
"iam>DeleteRolePolicy",
"iam>ListRolePolicies",
"iam>CreateServiceLinkedRole",
"iam>DeleteServiceLinkedRole",
"iam>ListRoles",
"iam>ListPolicies",
"iam:UpdateAssumeRolePolicy",
"iam:PutRolePolicy",
"iam:UpdateRoleDescription",
"iam:TagOpenIDConnectProvider",
"iam:UntagOpenIDConnectProvider",
"iam:GetOpenIDConnectProvider",
```

```
"iam>CreateOpenIDConnectProvider",
"iam>DeleteOpenIDConnectProvider",
"iam>ListOpenIDConnectProviders",
"iam>UpdateOpenIDConnectProviderThumbprint",
"iam>AddClientIDToOpenIDConnectProvider",
"iam>RemoveClientIDFromOpenIDConnectProvider"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"ec2:DescribeInstances",
"ec2:DescribeInstanceStatus",
"ec2:DescribeSecurityGroups",
"ec2:DescribeKeyPairs",
"ec2:StartInstances",
"ec2:StopInstances",
"ec2:RebootInstances",
"ec2:DescribeInstanceTypeOfferings",
"ec2:CreateTags",
"ec2:DeleteTags",
"ec2:DescribeVolumes",
"ec2:DeleteVolume",
"ec2:CreateVolume",
"ec2>CreateInternetGateway",
"ec2:AttachInternetGateway",
"ec2:DescribeInternetGateways",
"ec2:DetachInternetGateway",
"ec2:DeleteInternetGateway",
"ec2:AllocateAddress",
"ec2:ReleaseAddress",
"ec2:DescribeAddresses",
"ec2:CreateVpc",
"ec2:DeleteVpc",
"ec2:ModifyVpcAttribute",
"ec2:DescribeVpcs",
"ec2:CreateSubnet",
"ec2:DescribeSubnets",
"ec2:DeleteSubnet",
"ec2:ModifySubnetAttribute",
"ec2:DeleteSubnet",
"ec2:CreateRouteTable",
"ec2:DescribeRouteTables",
"ec2:AssociateRouteTable",
"ec2:DisassociateRouteTable",
"ec2:DeleteRouteTable",
"ec2:CreateRoute",
"ec2:ReplaceRoute",
"ec2:DeleteRoute",
"ec2:CreateSecurityGroup",
"ec2:DeleteSecurityGroup",
"ec2:DescribeNetworkInterfaces",
```

```
"ec2:CreateNatGateway",
"ec2:DeleteNatGateway",
"ec2:DescribeNatGateways",
"ec2:DetachInternetGateway",
"ec2:DeleteInternetGateway",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",
"ec2:CreateNetworkInterface",
"ec2:DeleteNetworkInterface",
"ec2:ModifyNetworkInterfaceAttribute",
"ec2:CreateLaunchTemplate",
"ec2:CreateLaunchTemplateVersion",
"ec2:DescribeLaunchTemplates",
"ec2:DescribeLaunchTemplateVersions",
"ec2:DeleteLaunchTemplate",
"ec2:RunInstances"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"elasticloadbalancing>CreateLoadBalancer",
"elasticloadbalancing>DeleteLoadBalancer",
"elasticloadbalancing>DescribeLoadBalancers",
"elasticloadbalancing>AddTags",
"elasticloadbalancing>RemoveTags"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"autoscaling>CreateAutoScalingGroup",
"autoscaling>UpdateAutoScalingGroup",
"autoscaling>DeleteAutoScalingGroup",
"autoscaling>DescribeAutoScalingGroups",
"autoscaling>DescribeScalingActivities",
"autoscaling>SetDesiredCapacity",
"autoscaling>TerminateInstanceInAutoScalingGroup",
"autoscaling>AttachLoadBalancerTargetGroups",
"autoscaling>DetachLoadBalancerTargetGroups"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"s3>ListBucket",
"s3>GetObject",
"s3>PutObject"
]
```

```
],
"Resource": [
    "arn:aws:s3::::terraform-state-bucket-g1-2403",
    "arn:aws:s3::::terraform-state-bucket-g1-2403/*"
]
},
{
    "Effect": "Allow",
    "Action": [
        "ssm:DescribeInstanceInformation",
        "ssm:SendCommand",
        "ssm:GetCommandInvocation",
        "ssm:StartSession",
        "ec2messages:GetMessages",
        "ec2messages:AcknowledgeMessage",
        "ec2messages:SendReply"
    ],
    "Resource": "*"
}
]
```

### .gitignore

El archivo **.gitignore** es un mecanismo que utiliza Git para determinar qué archivos o directorios deben ser ignorados y no ser rastreados o enviados (push) al repositorio remoto. Es especialmente útil en un trabajo práctico (TP) para evitar subir archivos que:

- Contienen datos sensibles (como claves, contraseñas, configuraciones privadas).
- Son generados automáticamente (archivos temporales, compilados, logs).
- No aportan valor al código fuente o la documentación del TP.

```
# Ignorar claves privadas y archivos sensibles
keys/
*.pem
*.swp
terraform.tfstate
terraform.tfstate.backup
.terraform/
.vscode
```

## TERRAFORM

Crearemos los archivos de terraform detallados a continuación para que al finalizar realicemos el commit y merge (comandos descriptos anteriormente) en main para que se ejecute el workflow de github actions:

main.tf

```
resource "aws_key_pair" "pin" {
    key_name   = var.key_name
    public_key = var.public_ssh_key
}

resource "aws_security_group" "bastion_sg" {
    name        = "bastion-sg"
    description = "Security Group for Bastion Host"
    vpc_id      = var.vpc_id

    ingress {
        description = "Allow SSH Access"
        from_port   = 22
        to_port     = 22
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    egress {
        description = "Allow all outbound traffic"
        from_port   = 0
        to_port     = 0
        protocol    = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
}

resource "aws_instance" "bastion" {
    ami           = var.ami_id
    instance_type = var.instance_type
    key_name      = aws_key_pair.pin.key_name
    vpc_security_group_ids = [aws_security_group.bastion_sg.id]
    subnet_id     = var.subnet_id

    iam_instance_profile = aws_iam_instance_profile.ec2_admin_profile.name

    user_data = file("${path.module}/scripts/install_tools.sh")

    tags = {
        Name = "bastion-host"
    }
}
```



backend.tf

Bucket en S3

Porcedemos a ingresar al servicio de S3 y presionamos sobre el boton create bucket

The screenshot shows the AWS S3 console with the 'General purpose buckets' tab selected. At the top right, there is a yellow 'Create bucket' button. Below it, a list of buckets is shown, with one specific bucket named 'terraform-state-bucket-g1-2403' highlighted by a red box.

Colocamos el **nombre** **terraform-state-bucket-g1-2403**, como general purpose y creamos el bucket.

The screenshot shows the 'Create bucket' wizard. In the 'General configuration' step, the 'Bucket type' dropdown is set to 'General purpose'. The 'Bucket name' field contains 'terraform-state-bucket-g1-2403'. In the 'Object Ownership' step, the 'ACLs disabled (recommended)' option is selected.

En el archivo backend.tf seteamos que el archivo .tfstate se guarde en el bucket de s3

```
terraform {
  backend "s3" {
    bucket  = "terraform-state-bucket-g1-2403"      # Nombre del bucket S3 donde se almacenará el estado
    key     = "ec2/statefile.tfstate"                 # Ruta y nombre del archivo de estado dentro del bucket
    region  = "us-east-1"
    encrypt = true

    # La siguiente línea se utiliza para habilitar el bloqueo del estado usando una tabla DynamoDB.
    # El bloqueo evita que múltiples procesos modifiquen el estado simultáneamente.
    # Como no es un ambiente productivo se deshabilita el bloqueo comentando dicha línea .
    # dynamodb_table = "terraform-lock-table"
  }
}
```

### iam.tf

Crea el rol ec2-admin con los permisos definidos en el archivo ec2-admin.json mostrado anteriormente

```
resource "aws_iam_role" "ec2_admin_role" {
  name = "ec2-admin"

  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
}

resource "aws_iam_instance_profile" "ec2_admin_profile" {
  name = "ec2-admin"
  role = aws_iam_role.ec2_admin_role.name
}

resource "aws_iam_policy" "ec2_admin_policy" {
  name         = "ec2-admin-policy"
  description = "Permisos para administrar EC2 y EKS desde el bastion"
  policy      = file("${path.module}/policies/ec2-admin.json")
}

resource "aws_iam_role_policy_attachment" "attach_bastion_eks_policy" {
  role      = aws_iam_role.ec2_admin_role.name
  policy_arn = aws_iam_policy.ec2_admin_policy.arn
}
```

## variables.tf

Aquí se definen los nombres de las variables

```
variable "vpc_id" {
  description = "ID de la VPC"
  type        = string
}

variable "vpc_id" {
  description = "ID de la SUBNET"
  type        = string
}

variable "ami_id" {
  description = "AMI para Ubuntu 22.04 LTS"
  default     = "ami-0e1bed4f06a3b463d"
}

variable "instance_type" {
  description = "Tipo de instancia EC2"
  default     = "t2.micro"
}

variable "key_name" {
  description = "Nombre del par de claves SSH"
  default     = "pin"
}

variable "public_ssh_key" {
  description = "Clave pública para SSH"
  type        = string
}
```

## outputs.tf

```
output "bastion_public_ip" {
  description = "IP pública del bastion"
  value       = aws_instance.bastion.public_ip
}
```

## providers.tf

```
provider "aws" {
  region = "us-east-1"
}
```



terraform.tfvars

vpc y subnet id

Copiamos el VPC ID de la región us-east-1 desde la consola o por linea de comandos:

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Virtual private cloud', 'Your VPCs' is selected. In the main area, 'Your VPCs (1/1)' is listed with one item: 'default'. The 'VPC ID' column for this item is highlighted with a red box and contains the value 'vpc-04adbff65ec30ad98'. Below this, the 'Resource map' tab is selected in the navigation bar. The 'Subnets (6)' section shows two subnets: 'us-east-1a' (subnet-068cae90d5c2d69f6) and 'us-east-1b' (subnet-02b78307463dc5893). A route table 'rtb-0e2a3712at' is also shown.

Por línea de comandos podemos consultararlo de la siguiente forma:

```
aws ec2 describe-subnets \
--region us-east-1 \
--query "Subnets[*].{SubnetId:SubnetId, VpcId:VpcId, Name:Tags[?Key=='Name'][0].Value, CIDR:CidrBlock}" \
--output table
```

devops 2403	<b>Grupo 1</b> <b>PIN Final</b>	mundosE
-------------	------------------------------------	---------

```
ubuntu@ip-172-31-90-146:~$ aws ec2 describe-subnets \
--region us-east-1 \
--query "Subnets[*].{SubnetId:SubnetId, VpcId:VpcId, Name:Tags[?Key=='Name'][0].Value, CIDR:CidrBlock}" \
--output table

|                               DescribeSubnets |
+-----+-----+-----+-----+
|   CIDR    | Name | SubnetId | VpcId |
+-----+-----+-----+-----+
| 172.31.0.0/20 | None | subnet-08ff6d1e09f52ce23 | vpc-04adbff65ec30ad98 |
| 172.31.16.0/20 | None | subnet-068cae90d5c2d69f6 | vpc-04adbff65ec30ad98 |
| 172.31.64.0/20 | None | subnet-0d81ad806b20fd77e | vpc-04adbff65ec30ad98 |
| 172.31.32.0/20 | None | subnet-02b78307463dc5893 | vpc-04adbff65ec30ad98 |
| 172.31.48.0/20 | None | subnet-0636e0571600ea5b9 | vpc-04adbff65ec30ad98 |
| 172.31.80.0/20 | None | subnet-09b05791e36ae2f33 | vpc-04adbff65ec30ad98 |
+-----+-----+-----+-----+
```

Asignamos los valores a las variables definidas en el archivo variables.tf, conforme los datos que extraemos de nuestro entorno de AWS:

```
vpc_id      = "vpc-04adbff65ec30ad98"
subnet_id   = "subnet-09b05791e36ae2f33"
ami_id      = "ami-0e1bed4f06a3b463d"
key_name    = "bastion-key"
public_ssh_key = "ssh-rsa AAAAB3NzaC1....."
```

devops 2403	<b>Grupo 1</b> <b>PIN Final</b>	mundosE
-------------	------------------------------------	---------

## GITHUB

### REPOSITORY SECRETS

Por seguridad se configura el **Access y Secret Key** del usuario de servicio terraform dentro de Repository Secrets

The screenshot shows the GitHub repository settings for 'dec-wil / mundose.pinfinal.grupo1'. The 'Actions secrets and variables' section is open, showing two environment secrets: 'AWS\_ACCESS\_KEY\_ID' and 'AWS\_SECRET\_ACCESS\_KEY'. There are also two repository secrets listed: 'AWS\_ACCESS\_KEY\_ID' and 'AWS\_SECRET\_ACCESS\_KEY'. A green button at the top right says 'New repository secret'.

Name	Last updated
AWS_ACCESS_KEY_ID	now
AWS_SECRET_ACCESS_KEY	now

### GITHUB ACTIONS

Workflow deploy.yaml

.github/workflows/deploy.yml

Este archivo es un workflow de GitHub Actions que automatiza el despliegue de infraestructura en AWS utilizando Terraform. Se ejecuta cada vez que se realiza un push a la rama main y consta de dos jobs principales:

plan:

- Revisa el código del repositorio.
- Utiliza las credenciales de AWS mediante la funcionalidad de Repository Secret de GitHub.
- Instala Terraform (versión 1.5.0).
- Inicializa Terraform en el directorio correspondiente y ejecuta terraform plan para mostrar qué cambios se realizarán sin aplicarlos.

apply:

- Depende de la ejecución del job plan.

- Realiza básicamente los mismos pasos de checkout, configuración de credenciales e instalación de Terraform.
- Inicializa Terraform y ejecuta terraform apply con -auto-approve para aplicar los cambios automáticamente.

```
name: Terraform Deploy to AWS
on:
  push:
    branches:
      - main
jobs:
  plan:
    plan:
      runs-on: ubuntu-latest
      steps:
        - name: Checkout código del repositorio
          uses: actions/checkout@v2

        - name: Configurar credenciales AWS desde GitHub Secrets
          uses: aws-actions/configure-aws-credentials@v2
          with:
            aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
            aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
            aws-region: us-east-1

        - name: Instalar Terraform
          uses: hashicorp/setup-terraform@v2
          with:
            terraform_version: 1.5.0

        - name: Inicializar Terraform
          run: cd aws/terraform/ec2 && terraform init

        - name: Ejecutar `terraform plan`
          run: cd aws/terraform/ec2 && terraform plan -lock=false

  apply:
    needs: plan
    plan:
      runs-on: ubuntu-latest
      steps:
        - name: Checkout código del repositorio
          uses: actions/checkout@v2

        - name: Configurar credenciales AWS desde GitHub Secrets
          uses: aws-actions/configure-aws-credentials@v2
          with:
            aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
            aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
            aws-region: us-east-1

        - name: Instalar Terraform
          uses: hashicorp/setup-terraform@v2
          with:
```

```
terraform_version: 1.5.0
- name: Inicializar Terraform
  run: cd aws/terraform/ec2 && terraform init
- name: Aplicar cambios con Terraform
  run: cd aws/terraform/ec2 && terraform apply -auto-approve -lock=false
```

## COMMITS

Cuando tengamos creados todos los archivos de terraform y el archivo de workflow, debemos ejecutar los siguientes comandos, para actualizar, fusionar e inicializar el workflow en github actions para que despliegue nuestra instancia de EC2.

Primero deberíamos ejecutar el push de los archivos modificados a dev para su revisión, validación y aprobación

```
# Actualizar rama dev
git status #Mostramos los archivos modificados y el estado actual de la rama
git checkout dev #Nos aseguramos de estar en la rama de desarrollo.
git add . #Agrega todos los archivos modificados.
git commit -m "Update files" #El mensaje de commit
git push origin dev #Enviamos los archivos de la rama dev al repositorio remoto
```

Una vez revisado, validado y aprobado realizamos el merge a la rama main.

```
# Merge de rama dev a main
git checkout main #Cambiamos a la rama main
git merge dev #Ingresamos los cambios de la rama dev a main
git push origin main #Enviamos los archivos de la rama main al repositorio remoto
```

En caso de no querer ejecutar el workflow de github actions, dentro del mensaje del commit antes del merge, agregar al final el texto “[skip ci]”, con esto se evitará la ejecución del workflow.

Ejemplo: **git commit -m "Update files [skip ci]"**



## Ejecución workflow

Luego de realizar el push a dev y el merge a main se ejecuta el workflow en github actions.

dec-wil / mundose.pinfinal.grupo1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

← Terraform Deploy to AWS

**Agregando file backend.tf #12**

**Summary**

Triggered via push 1 minute ago Status In progress

dec-wil pushed -o- 70fec46 main

Total duration — Artifacts —

Jobs

plan (✓) apply (●)

Run details

Usage Workflow file

**deploy.yml**  
on: push

plan (✓) 15s → apply (●) 1m 11s

Se observa que el workflow se ejecuta sin errores.

dec-wil / mundose.pinfinal.grupo1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

← Terraform Deploy to AWS

**Agregando file backend.tf para guardado de statefile #19**

**Summary**

Triggered via push 1 minute ago Status Success

dec-wil pushed -o- c15cb53 main

Total duration 1m 13s Artifacts —

Jobs

plan (✓) apply (✓)

Run details

Usage Workflow file

**deploy.yml**  
on: push

plan (✓) 17s → apply (✓) 38s

devops 2403

Grupo 1

PIN Final

mundosE

Verificación y extracción de ip pública de la instancia:

```
apply
succeeded 10 minutes ago in 38s

  ✓ Aplicar cambios con Terraform (sin bloqueo)

      ]
  + name          = "bastion-sg"
  + name_prefix   = (known after apply)
  + owner_id      = (known after apply)
  + revoke_rules_on_delete = false
  + tags_all      = (known after apply)
  + vpc_id        = "vpc-04adbfff65ec30ad98"

326 }
327
328 Plan: 7 to add, 0 to change, 0 to destroy.
329
330 Changes to Outputs:
331 ~ bastion_public_ip = "100.27.19.49" -> (known after apply)
332 aws_key_pair.pin: Creating...
333 aws_iam_policy.ec2_admin_policy: Creating...
334 aws_iam_role.ec2_admin_role: Creating...
335 aws_security_group.bastion_sg: Creating...
336 aws_key_pair.pin: Creation complete after 0s [id=pin]
337 aws_iam_policy.ec2_admin_policy: Creation complete after 0s [id=arn:aws:iam::123456789012:policy/...
338 aws_iam_role.ec2_admin_role: Creation complete after 0s [id=ec2-admin...
339 aws_iam_role_policy_attachment.attach_bastion_eks_policy: Creating...
340 aws_iam_instance_profile.ec2_admin_profile: Creating...
341 aws_iam_role_policy_attachment.attach_bastion_eks_policy: Creation co...
342 aws_security_group.bastion_sg: Creation complete after 2s [id=sg-0e4a...
343 aws_iam_instance_profile.ec2_admin_profile: Creation complete after 6...
344 aws_instance.bastion: Creating...
345 aws_instance.bastion: Still creating... [10s elapsed]
346 aws_instance.bastion: Creation complete after 15s [id=i-0f1b495c8364f...
347
348 Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
349
350 Outputs:
351
352 bastion_public_ip = "54.224.53.78"
```

El archivo tfstate se guarda y actualiza en un bucket de S3, gracias al archivo backend.tf

Amazon S3 > Buckets > [terraform-state-bucket-g1-2403](#) > ec2/

azón S3 <

General purpose buckets

Temporary buckets

Access Grants

Access Points

Direct Lambda Access Points

Multi-Region Access Points

Cloud Operations

Access Analyzer for S3

Public Access settings for this bucket

Range Lens

Objects (1)

Copy S3 URI Copy URL Download Open Actions

Create folder Upload

Find objects by prefix

Name	Type	Last modified
<a href="#">statefile.tfstate</a>	tfstate	March 4, 2025, 20:28:24 (UTC-03:00)



## AWS (EC2)

### Validación de creación de instancia

Validamos que la instancia se haya creado correctamente, con sus paquetes instalados, permisos asignados, etc.

The screenshot shows the AWS EC2 Instances page. The sidebar on the left has 'Instances' selected. The main area displays a table with one row for the instance 'bastion-host'. The table columns include Name, Instance ID, Instance state, Instance type, Status check, and Availability Zone. The instance is listed as 'Running' with a green checkmark and the ID i-037357b73672e7fd2.

### Permisos

Vemos que el rol ec2-admin se encuentra asignado

The screenshot shows the AWS EC2 Instance summary page for the instance i-037357b73672e7fd2. The sidebar on the left has 'Instances' selected. The main area shows the instance summary for 'bastion-host'. The 'Security' tab is active. Under the 'IAM Role' section, the role 'ec2-admin' is listed and highlighted with a red box. Other tabs include Details, Status and alarms, Monitoring, Security, and Network.

## Acceso a instancia

Como en el archivo main.tf dejamos el puerto 22 abierto en el Security Group probamos acceder a la instancia mediante utilizando nuestra llave privada.

```
ssh -i keys/pin.pem ubuntu@x.x.x.x
```

```
PS E:\Cursos\MUNDOSE\DevOps\PIN FINAL> ssh -i keys/pin.pem ubuntu@3.95.154.227
The authenticity of host '3.95.154.227 (3.95.154.227)' can't be established.
ED25519 key fingerprint is SHA256:E7ju9w5hz1vzi9R13nh60AYCGAkGxTA2f840YVhnXV8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.95.154.227' (ED25519) to the list of known hosts.

Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1021-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Sat Mar  1 22:22:02 UTC 2025

System load:  0.13           Processes:          111
Usage of /:   35.8% of 7.57GB  Users logged in:    0
Memory usage: 29%           IPv4 address for eth0: 172.31.90.146
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
9 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

10 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

## Ubuntu Version

```
ubuntu@ip-172-31-90-146:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.5 LTS
Release:        22.04
Codename:       jammy
ubuntu@ip-172-31-90-146:~$
```

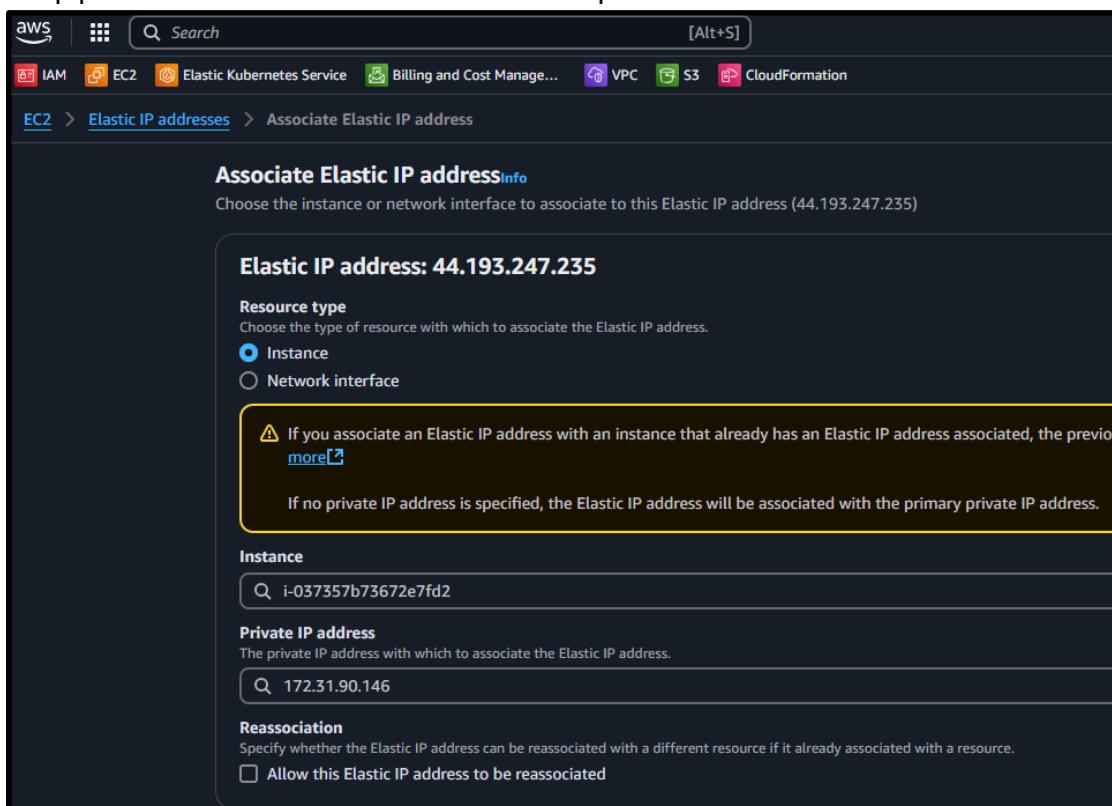
## Paquetes

Observamos los paquetes instalados mediante el script user\_data.sh

```
ubuntu@ip-172-31-90-146:~$ aws --version
aws-cli/1.22.34 Python/3.10.12 Linux/6.8.0-1021-aws botocore/1.23.34
ubuntu@ip-172-31-90-146:~$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~22.04.1
ubuntu@ip-172-31-90-146:~$ kubectl version --client
Client Version: v1.32.2
Kustomize Version: v5.5.0
ubuntu@ip-172-31-90-146:~$ helm version
version.BuildInfo{Version:"v3.17.1", GitCommit:"980d8ac1939e39138101364400756af2bdee1da5", GitTreeState:"clean", GoVersion:"go1.23.5"}
ubuntu@ip-172-31-90-146:~$ eksctl version
0.205.0
```

## Elastic IP

Para que la instancia no pierda la ip pública luego de un reinicio, reservaremos y asignaremos una dirección ip pública mediante Elastic IP a la interfaz privada de nuestra instancia de EC2.



devops 2403

Grupo 1

PIN Final

mundosE

Observamos que la ip fue asignada correctamente.

The screenshot shows the AWS EC2 Instance Details page for instance **i-037357b73672e7fd2 (bastion-host)**. The left sidebar shows navigation options like Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, and Network & Security. The main content area displays various instance details:

- Public IPv4 address:** 44.193.247.235 (with a link to open address)
- Instance state:** Running
- Private IP4 addresses:** 172.31.90.146
- Public IPv4 DNS:** ec2-44-193-247-235.compute-1.amazonaws.com (with a link to open address)
- Private IP DNS name (IPv4 only):** ip-172-31-90-146.ec2.internal
- Instance type:** t2.micro
- VPC ID:** vpc-04adbff65ec30ad98
- Subnet ID:** subnet-09b05791e36ae2f33
- Instance ARN:** arn:aws:ec2:us-east-1:194722402815:instance/i-037357b73672e7fd2
- IAM Role:** ec2-admin
- IMDSv2:** Optional (with a note: EC2 recommends setting IMDSv2 to require it and a link to Learn more)
- Elastic IP addresses:** 44.193.247.235 [Public IP]
- AWS Compute Optimizer finding:** Opt-in to AWS Compute Optimizer for recommendations. (with a link to Learn more)
- Auto Scaling Group name:** -
- Managed:** false

devops 2403	Grupo 1  PIN Final	mundosE
-------------	--------------------------	---------

## EKS

**Amazon EKS (Elastic Kubernetes Service)** es el servicio administrado de Kubernetes de AWS. Permite desplegar, administrar y escalar aplicaciones en contenedores sin la complejidad de operar un clúster de Kubernetes de forma manual. Entre sus características destacan:

- **Gestión simplificada:** AWS se encarga del aprovisionamiento y mantenimiento del plano de control (control plane) de Kubernetes.
- **Integración con otros servicios de AWS:** Se integra con servicios como IAM, VPC, CloudWatch, y otros para ofrecer seguridad, networking y monitoreo.
- **Alta disponibilidad y escalabilidad:** Permite configurar clústeres escalables y distribuidos en múltiples zonas de disponibilidad.
- **Actualizaciones y parches:** AWS administra actualizaciones y parches para el plano de control, lo que facilita mantener el clúster actualizado y seguro.

## DESPLIEGUE

Script crear\_cluster.sh

Script de instalación de EKS.

```
#!/bin/bash
# Configura el script para que se detenga ante errores (-e), variables no definidas (-u)
# y que los errores en pipelines se propaguen (-o pipefail).
set -euo pipefail

usage() {
    echo "Uso: $0 [-d]"
    echo " -d  Modo delete: elimina el clúster '$CLUSTER_NAME' en la región '$AWS_REGION' ."
    exit 1
}

# =====
# Procesar parámetros
# =====
DELETE_MODE=false
while getopts ":d" opt; do
    case ${opt} in
        d )
            DELETE_MODE=true
            ;;
        \? )
            ;;
    esac
done
```

```
usage
;;
esac
done

# =====
# Configuración de variables
# =====
# Define el nombre del clúster que se creará o eliminará.
CLUSTER_NAME="2403-g1-pin-final"
# Define la región de AWS donde se creará o eliminará el clúster.
AWS_REGION="us-east-1"
# Define el nombre de la clave SSH que se usará para acceder a los nodos. Asegúrate de que exista en tu cuenta.
SSH_KEY="pin"
# Define las zonas de disponibilidad en las que se desplegarán los nodos.
ZONES="us-east-1a,us-east-1b,us-east-1c"
# Define el número de nodos (workers) que tendrá el clúster.
NODE_COUNT=3
# Define el tipo de instancia para los nodos.
NODE_TYPE="t2.small"

# =====
# Funciones de utilidad
# =====
# Función para comprobar si un comando existe en el sistema.
command_exists() {
    command -v "$1" >/dev/null 2>&1
}

# =====
# Verificaciones previas
# =====
if ! command_exists aws; then
    echo "Error: aws CLI no está instalado. Por favor, instálalo antes de continuar." >&2
    exit 1
fi

if ! command_exists eksctl; then
    echo "Error: eksctl no está instalado. Por favor, instálalo antes de continuar." >&2
    exit 1
fi

if ! aws sts get-caller-identity >/dev/null 2>&1; then
    echo "Por favor, ejecuta 'aws configure' para establecer credenciales válidas." >&2
    exit 1
fi
```

```
# =====
# Operación: Crear o eliminar el clúster
# =====
if [ "$DELETE_MODE" = true ]; then
    echo "Modo delete activado: eliminando el clúster '$CLUSTER_NAME' en la región '$AWS_REGION'..."
    # Eliminar el clúster con eksctl
    eksctl delete cluster --name "$CLUSTER_NAME" --region "$AWS_REGION"
    # Verificar la eliminación consultando el stack de CloudFormation (opcional)
    echo "Verificando la eliminación del clúster..."
    aws cloudformation describe-stacks --stack-name "eksctl-$CLUSTER_NAME" --region "$AWS_REGION" || \
        echo "El clúster '$CLUSTER_NAME' ha sido eliminado exitosamente."
    exit 0
fi
echo "Credenciales verificadas. Procediendo con la creación del clúster '$CLUSTER_NAME' en la región '$AWS_REGION'."

# =====
# Creación del clúster con eksctl
# =====
# Se utiliza 'eksctl create cluster' con varios parámetros:
#   --name: asigna el nombre del clúster.
#   --region: define la región de AWS.
#   --nodes: establece la cantidad de nodos que se desplegarán.
#   --node-type: define el tipo de instancia para los nodos.
#   --with-oidc: habilita la integración con OIDC.
#   --ssh-access: habilita el acceso SSH a los nodos.
#   --ssh-public-key: especifica la clave SSH a utilizar.
#   --managed: indica que los nodos serán administrados (managed node groups).
#   --full-ecri-access: otorga acceso completo a ECR (Elastic Container Registry).
#   --zones: define las zonas de disponibilidad a utilizar.

if eksctl create cluster \
    --name "$CLUSTER_NAME" \
    --region "$AWS_REGION" \
    --nodes "$NODE_COUNT" \
    --node-type "$NODE_TYPE" \
    --version "1.32" \
    --with-oidc \
    --ssh-access \
    --ssh-public-key "$SSH_KEY" \
    --managed \
    --full-ecri-access \
    --zones "$ZONES"; then
    echo "Configuración del clúster completada con éxito mediante eksctl."
else
    echo "La configuración del clúster falló durante la ejecución de eksctl." >&2
    exit 1

```

fi

## Preparación

Puedo subir el script al Bastion Host mediante scp o bien si hicimos un git clone del repositorio. nos dirigimos al directorio donde se encuentra el script. Asignamos permisos de ejecución al script y ejecutamos.

```
scp -i .\keys\pin.pem .\aws\eks\scripts\crear_cluster.sh ubuntu@3.95.154.227:/home/ubuntu  
PS E:\Cursos\MUNDOSE\DevOps\PIN FINAL> scp -i .\keys\pin.pem .\aws\eks\scripts\crear_cluster.sh ubuntu@3.95.154.227:/home/ubuntu  
crear_cluster.sh  
PS E:\Cursos\MUNDOSE\DevOps\PIN FINAL> ssh -i keys/pin.pem ubuntu@3.95.154.227  
100% 3696 22.3KB/s 00:00  
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1021-aws x86_64)
```

```
git clone https://github.com/dec-wil/mundose.pinfinal.grupo1
```

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1$ git clone https://github.com/dec-wil/mundose.pinfinal.grupo1  
Cloning into 'mundose.pinfinal.grupo1'...  
remote: Enumerating objects: 293, done.  
remote: Counting objects: 100% (293/293), done.  
remote: Compressing objects: 100% (129/129), done.  
remote: Total 293 (delta 104), reused 274 (delta 85), pack-reused 0 (from 0)  
Receiving objects: 100% (293/293), 16.09 MiB | 44.77 MiB/s, done.  
Resolving deltas: 100% (104/104), done.
```

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/00-eks$ chmod +x 00-crear_cluster.sh && ls -la  
total 28  
drwxrwxr-x 2 ubuntu ubuntu 4096 Mar 13 21:12 .  
drwxrwxr-x 7 ubuntu ubuntu 4096 Mar 13 21:12 ..  
-rwxrwxr-x 1 ubuntu ubuntu 4002 Mar 13 21:12 00-crear_cluster.sh  
-rw-rw-r-- 1 ubuntu ubuntu 3486 Mar 13 21:12 01-configmap.sh  
-rw-rw-r-- 1 ubuntu ubuntu 2735 Mar 13 21:12 02-change-to-iamuser.sh  
-rw-rw-r-- 1 ubuntu ubuntu 2993 Mar 13 21:12 03-eks-nodes-scale-config.sh  
-rw-rw-r-- 1 ubuntu ubuntu 803 Mar 13 21:12 EKSIAAMAdminAccessRole.yaml
```

## Ejecución de script y verificaciones

### Logs resultantes del script de despliegue del cluster

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/00-eks$ ./00-crear_cluster.sh  
Credenciales verificadas. Procediendo con la creación del clúster '2403-g1-pin-final' en la región 'us-east-1'.  
2025-03-13 20:54:00 [i] eksctl version 0.205.0  
2025-03-13 20:54:00 [i] using region us-east-1  
2025-03-13 20:54:00 [i] subnets for us-east-1a - public:192.168.0.0/19 private:192.168.96.0/19  
2025-03-13 20:54:00 [i] subnets for us-east-1b - public:192.168.32.0/19 private:192.168.128.0/19  
2025-03-13 20:54:00 [i] subnets for us-east-1c - public:192.168.64.0/19 private:192.168.160.0/19  
2025-03-13 20:54:01 [i] nodegroup "ng-0e91b496" will use "" [AmazonLinux2/1.32]  
2025-03-13 20:54:01 [i] using EC2 key pair "pin"  
2025-03-13 20:54:01 [i] using Kubernetes version 1.32
```

```
2025-03-13 20:54:01 [i]  creating EKS cluster "2403-g1-pin-final" in "us-east-1" region with managed nodes
2025-03-13 20:54:01 [i]  will create 2 separate CloudFormation stacks for cluster itself and the initial
managed nodegroup
2025-03-13 20:54:01 [i]  if you encounter any issues, check CloudFormation console or try 'eksctl utils
describe-stacks --region=us-east-1 --cluster=2403-g1-pin-final'
2025-03-13 20:54:01 [i]  Kubernetes API endpoint access will use default of {publicAccess=true,
privateAccess=false} for cluster "2403-g1-pin-final" in "us-east-1"
2025-03-13 20:54:01 [i]  CloudWatch logging will not be enabled for cluster "2403-g1-pin-final" in "us-
east-1"
2025-03-13 20:54:01 [i]  you can enable it with 'eksctl utils update-cluster-logging --enable-
types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=2403-g1-pin-final'
2025-03-13 20:54:01 [i]  default addons vpc-cni, kube-proxy, coredns, metrics-server were not specified,
will install them as EKS addons
2025-03-13 20:54:01 [i]
2 sequential tasks: { create cluster control plane "2403-g1-pin-final",
  2 sequential sub-tasks: {
    5 sequential sub-tasks: {
      1 task: { create addons },
      wait for control plane to become ready,
      associate IAM OIDC provider,
      no tasks,
      update VPC CNI to use IRSA if required,
    },
    create managed nodegroup "ng-0e91b496",
  }
}
2025-03-13 20:54:01 [i]  building cluster stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 20:54:01 [i]  deploying stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 20:54:31 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 20:55:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 20:56:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 20:57:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 20:58:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 20:59:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 21:00:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 21:01:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 21:02:01 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-13 21:02:03 [!]  recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on
the cluster, eksctl cannot configure the requested permissions; the recommended way to provide IAM
permissions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add
all recommended policies to the config file, under `addon.PodIdentityAssociations`, and run `eksctl update
addon`
2025-03-13 21:02:03 [i]  creating addon: vpc-cni
2025-03-13 21:02:03 [i]  successfully created addon: vpc-cni
2025-03-13 21:02:03 [i]  creating addon: kube-proxy
2025-03-13 21:02:03 [i]  successfully created addon: kube-proxy
2025-03-13 21:02:04 [i]  creating addon: coredns
2025-03-13 21:02:04 [i]  successfully created addon: coredns
2025-03-13 21:02:05 [i]  creating addon: metrics-server
2025-03-13 21:02:05 [i]  successfully created addon: metrics-server
2025-03-13 21:04:06 [i]  addon "vpc-cni" active
2025-03-13 21:04:07 [i]  deploying stack "eksctl-2403-g1-pin-final-addon-vpc-cni"
2025-03-13 21:04:07 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-vpc-cni"
2025-03-13 21:04:37 [i]  waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-vpc-cni"
2025-03-13 21:04:37 [i]  updating addon
2025-03-13 21:04:47 [i]  addon "vpc-cni" active
2025-03-13 21:04:47 [i]  building managed nodegroup stack "eksctl-2403-g1-pin-final-nodegroup-ng-0e91b496"
```

devops 2403

Grupo 1

PIN Final

mundosE

```
2025-03-13 21:04:48 [i] deploying stack "eksctl-2403-g1-pin-final-nodegroup-ng-0e91b496"
2025-03-13 21:04:48 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-0e91b496"
2025-03-13 21:05:18 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-0e91b496"
2025-03-13 21:06:02 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-0e91b496"
2025-03-13 21:06:57 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-0e91b496"
2025-03-13 21:06:57 [i] waiting for the control plane to become ready
2025-03-13 21:06:58 [✓] saved kubeconfig as "/home/ubuntu/.kube/config"
2025-03-13 21:06:58 [i] no tasks
2025-03-13 21:06:58 [✓] all EKS cluster resources for "2403-g1-pin-final" have been created
2025-03-13 21:06:58 [i] nodegroup "ng-0e91b496" has 3 node(s)
2025-03-13 21:06:58 [i] node "ip-192-168-12-6.ec2.internal" is ready
2025-03-13 21:06:58 [i] node "ip-192-168-51-193.ec2.internal" is ready
2025-03-13 21:06:58 [i] node "ip-192-168-73-160.ec2.internal" is ready
2025-03-13 21:06:58 [i] waiting for at least 3 node(s) to become ready in "ng-0e91b496"
2025-03-13 21:06:58 [i] nodegroup "ng-0e91b496" has 3 node(s)
2025-03-13 21:06:58 [i] node "ip-192-168-12-6.ec2.internal" is ready
2025-03-13 21:06:58 [i] node "ip-192-168-51-193.ec2.internal" is ready
2025-03-13 21:06:58 [i] node "ip-192-168-73-160.ec2.internal" is ready
2025-03-13 21:06:58 [✓] created 1 managed nodegroup(s) in cluster "2403-g1-pin-final"
2025-03-13 21:06:59 [i] kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-03-13 21:06:59 [✓] EKS cluster "2403-g1-pin-final" in "us-east-1" region is ready
Configuración del clúster completada con éxito mediante eksctl.
ubuntu@ip-172-31-90-146:~$
```

## Nodos creados

```
ubuntu@ip-172-31-90-146:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
ip-192-168-12-6.ec2.internal   Ready    <none>   18m   v1.32.1-eks-5d632ec
ip-192-168-51-193.ec2.internal   Ready    <none>   18m   v1.32.1-eks-5d632ec
ip-192-168-73-160.ec2.internal   Ready    <none>   18m   v1.32.1-eks-5d632ec
ubuntu@ip-172-31-90-146:~$
```

## Pods del namespace kube-system

```
ubuntu@ip-172-31-90-146:~$ kubectl get pods -n kube-system
NAME          READY   STATUS    RESTARTS   AGE
aws-node-l7vxv   2/2    Running   0          18m
aws-node-wj27x   2/2    Running   0          18m
aws-node-wtp56   2/2    Running   0          18m
coredns-6b9575c64c-75sgv   1/1    Running   0          23m
coredns-6b9575c64c-dn2kp   1/1    Running   0          23m
kube-proxy-fc256   1/1    Running   0          18m
kube-proxy-rqxsq   1/1    Running   0          18m
kube-proxy-xjwzh   1/1    Running   0          18m
metrics-server-57b774cc8d-jjt9t   1/1    Running   0          22m
metrics-server-57b774cc8d-mt877   1/1    Running   0          22m
ubuntu@ip-172-31-90-146:~$
```

## Verificando OIDC

A raíz del warning que muestra el output del script verifíco si esta funcionando

```
ubuntu@ip-172-31-90-146:~$ aws eks describe-cluster \
--name 2403-g1-pin-final \
--region us-east-1 \
--query "cluster.identity.oidc.issuer" \
--output text
https://oidc.eks.us-east-1.amazonaws.com/id/C4435C9D6FDD7A9C7853F4D42473612E
ubuntu@ip-172-31-90-146:~$
```



## EC2

Observamos las instancias creadas por el cluster de EKS

Instances (4) <a href="#">Info</a>	
<input type="text"/> Find Instance by attribute or tag (case-sensitive)	
<a href="#">All states</a>	
Instance state = running	<a href="#">Clear filters</a>
<input type="checkbox"/> Name <a href="#">🔗</a>	Instance ID
<input type="checkbox"/> 2403-g1-pin-final-nginx-0e91b496-N...	i-0ccf4354eb454c47b
<input type="checkbox"/> 2403-g1-pin-final-nginx-0e91b496-N...	i-0da0bc4567c02d9ed
<input type="checkbox"/> 2403-g1-pin-final-nginx-0e91b496-N...	i-03db3c3c29fce6fb
<input type="checkbox"/> bastion-host	i-037357b73672e7fd2
	Instance state
	<a href="#">Running</a> <a href="#">🔗</a> <a href="#">🔍</a>
	t2.small
	<a href="#">Running</a> <a href="#">🔗</a> <a href="#">🔍</a>
	t2.small
	<a href="#">Running</a> <a href="#">🔗</a> <a href="#">🔍</a>
	t2.small
	<a href="#">Running</a> <a href="#">🔗</a> <a href="#">🔍</a>
	t2.micro

## Permisos

Cuando se despliega un nuevo clúster EKS, los usuarios de IAM por defecto no tienen permisos para administrar el clúster. Al ingresar desde el portal nos lo informa mediante un recuadro.

ⓘ Your current IAM principal doesn't have access to Kubernetes objects on this cluster.  
This might be due to the current principal not having an IAM access entry with permissions to access the cluster.

**2403-g1-pin-final** [C](#)

ⓘ Your current IAM principal doesn't have access to Kubernetes objects on this cluster.  
This may be due to the current user or role not having Kubernetes RBAC permissions to describe cluster resources or not having an entry in the cluster's auth config map. [Learn more](#) [🔗](#)

Cluster info <a href="#">Info</a>			
Status <span style="color: green;">Active</span>	Kubernetes version <a href="#">Info</a> 1.32	Support period <a href="#">Standard support until March 21, 2026</a>	Provider EKS
Cluster health issues <span style="color: green;">0</span>	Upgrade insights <span style="color: green;">4</span>	Node health issues <span style="color: green;">0</span>	

Por ello, siguiendo las buenas prácticas de seguridad, es necesario configurar un rol específico en AWS IAM, configurar los arn de los usuarios al rol y asignarlo en el bloque **mapRoles** del ConfigMap. También se podría configurar los usuarios directamente en la sección **mapUsers**.

El ConfigMap **aws-auth** en EKS es fundamental para integrar las identidades de AWS con el control de acceso en el clúster de Kubernetes. De esta forma, al asumir este rol, se otorgan los permisos administrativos adecuados en el clúster, garantizando un control de acceso centralizado y minimizando los riesgos.

## configMap

En un clúster EKS existen varios ConfigMaps, cada uno con funciones específicas que facilitan la configuración y operación del clúster. A continuación se detalla una lista que incluye al **aws-auth** junto con otros ConfigMaps comunes:

devops 2403	Grupo 1  PIN Final	mundosE
-------------	--------------------------	---------

- **aws-auth:**  
Este es el ConfigMap crítico para la integración de AWS IAM con Kubernetes. Permite mapear roles (mapRoles) y usuarios (mapUsers) de AWS a identidades y grupos en Kubernetes, gestionando la autenticación y autorización de los usuarios y nodos en el clúster.
- **coredns:**  
Este ConfigMap gestiona la configuración del servicio DNS interno del clúster, que es fundamental para la resolución de nombres y el descubrimiento de servicios dentro de Kubernetes.
- **aws-node:**  
Utilizado por el complemento de red de Amazon VPC CNI, este ConfigMap configura parámetros específicos relacionados con la red, como la asignación de direcciones IP a los pods y otros ajustes que optimizan la conectividad de red del clúster.
- **kube-proxy (según la configuración):**  
En algunos clústeres, se utiliza un ConfigMap para kube-proxy, el componente que maneja el enrutamiento del tráfico y la comunicación entre pods. La configuración de kube-proxy puede variar según la implementación y necesidades del clúster.
- **ConfigMaps personalizados:**  
Además de los ConfigMaps gestionados por el sistema, los usuarios pueden crear ConfigMaps personalizados para almacenar configuraciones específicas de aplicaciones, parámetros de entorno, o cualquier otra información que se desee inyectar en los pods sin requerir secretos.

Cada uno de estos ConfigMaps juega un rol esencial en la administración, operación y configuración de un clúster EKS, permitiendo gestionar desde la autenticación de usuarios hasta la configuración de la red interna y de los servicios de la aplicación.

#### configmap.sh

El siguiente script crea un rol mediante cloudformation, realiza un respaldo del configmap aws-auth, muestra el configmap actual, los parámetros que debemos agregar y permite ingresar/editar el configmap aws-auth

```
#!/bin/bash
set -euo pipefail
# Función de ayuda
usage() {
  echo "Uso: $0 -u <IAM_user>"
  exit 1
}
# =====
# Procesar parámetros
# =====
```

```
while getopts ":u:" opt; do
    case ${opt} in
        u )
            USER_NAME="$OPTARG"
            ;;
        \? )
            echo "Opción inválida: -$OPTARG" >&2
            usage
            ;;
        : )
            echo "La opción -$OPTARG requiere un argumento." >&2
            usage
            ;;
    esac
done
# Si no se proporcionó el parámetro -u, usar valor por defecto "dcabral"
if [ -z "${USER_NAME:-}" ]; then
    USER_NAME="dcabral"
fi
# =====
# Variables de configuración
# =====
NAMESPACE="kube-system"
CONFIGMAP_NAME="aws-auth"
BACKUP_FILE="aws-auth-backup.yaml"
TEMP_CURRENT_CONFIG="aws-auth-current.yaml"
# Variables para el rol
ROLE_TEMPLATE="EKSIAMAdminAccessRole.yaml"
PROCESSED_ROLE_TEMPLATE="EKSIAMAdminAccessRole-processed.yaml"
STACK_NAME="EKSIAMAdminAccessRole-stack"
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
ROLE_NAME="EKSIAMAdminAccessRole"

# Variable para el usuario (con el valor de -u)
USER_ARN="arn:aws:iam::${ACCOUNT_ID}:user/${USER_NAME}"

# =====
# 0. Procesar la plantilla del rol reemplazando el marcador de ACCOUNT_ID
# =====
echo "Procesando la plantilla del rol para reemplazar __ACCOUNT_ID__ con ${ACCOUNT_ID}..."
sed "s/__ACCOUNT_ID__/${ACCOUNT_ID}/g" "$ROLE_TEMPLATE" > "$PROCESSED_ROLE_TEMPLATE"
# =====
# 1. Crear/Actualizar el rol desde la plantilla procesada (CloudFormation)
# =====
echo "Creando/actualizando el rol '$ROLE_NAME' desde la plantilla '$PROCESSED_ROLE_TEMPLATE'..."
aws cloudformation deploy \
    --template-file "$PROCESSED_ROLE_TEMPLATE" \
```

```
--stack-name "${STACK_NAME}" \
--capabilities CAPABILITY_NAMED_IAM

echo "Rol '$ROLE_NAME' creado/actualizado correctamente."
echo
# =====
# 2. Respaldar la configuración actual de aws-auth
# =====
echo "Respaldando el ConfigMap '${CONFIGMAP_NAME}'..."
kubectl get configmap ${CONFIGMAP_NAME} -n ${NAMESPACE} -o yaml > ${BACKUP_FILE}
echo "Respaldo guardado en '${BACKUP_FILE}'"
echo
# =====
# 3. Obtener la configuración actual en un archivo temporal
# =====
kubectl get configmap ${CONFIGMAP_NAME} -n ${NAMESPACE} -o yaml > ${TEMP_CURRENT_CONFIG}
echo "Configuración actual del ConfigMap ${CONFIGMAP_NAME}:"
cat ${TEMP_CURRENT_CONFIG}
echo
# =====
# 4. Mostrar las líneas que se deben agregar
# =====
echo "Debes agregar las siguientes líneas en la sección 'mapRoles:' dentro del ConfigMap
'${CONFIGMAP_NAME}':"
echo
cat <<EOF
  - rolearn: arn:aws:iam::${ACCOUNT_ID}:role/${ROLE_NAME}
    username: admin
    groups:
      - system:masters
EOF
echo
echo "Y en la sección 'mapUsers:' puedes agregar lo siguiente para el usuario '${USER_NAME}':"
echo
cat <<EOF
  - userarn: ${USER_ARN}
    username: ${USER_NAME}
    groups:
      - system:masters
EOF
echo
# =====
# 5. Invitar al usuario a editar el ConfigMap manualmente
# =====
read -rp "Presiona Enter para editar el ConfigMap ahora (o CTRL+C para cancelar)... " _
kubectl edit -n ${NAMESPACE} configmap/${CONFIGMAP_NAME}
# =====
```

```
# Limpieza del archivo temporal
# =====
rm -f ${TEMP_CURRENT_CONFIG}
echo "Proceso finalizado."
```

Output:

Ejecutamos el script y mostrará lo siguiente

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/00-eks$ ./01-configmap.sh -u dcabral
Procesando la plantilla del rol para reemplazar __ACCOUNT_ID__ con 194722402815...
Creando/actualizando el rol 'EKSIAMAdminAccessRole' desde la plantilla 'EKSIAMAdminAccessRole-processed.yaml' ...

Waiting for changeset to be created..

No changes to deploy. Stack EKSIAMAdminAccessRole-stack is up to date
Rol 'EKSIAMAdminAccessRole' creado/actualizado correctamente.

Respaldando el ConfigMap 'aws-auth'...
Respaldo guardado en 'aws-auth-backup.yaml'

Configuración actual del ConfigMap aws-auth:
apiVersion: v1
data:
  mapRoles: |
    - groups:
        - system:bootstrappers
        - system:nodes
      rolearn: arn:aws:iam::194722402815:role/eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-0ztZAoPbRLZ6
      username: system:node:{EC2PrivateDNSName}
  kind: ConfigMap
metadata:
  creationTimestamp: "2025-03-13T21:05:49Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "1447"
  uid: 880f8811-f002-4547-a8d7-29f20cfac4e0

Debes agregar las siguientes líneas en la sección 'mapRoles:' dentro del ConfigMap 'aws-auth':
- rolearn: arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole
  username: admin
  groups:
    - system:masters

Y en la sección 'mapUsers:' puedes agregar lo siguiente para el usuario 'dcabral':
- userarn: arn:aws:iam::194722402815:user/dcabral
  username: dcabral
  groups:
    - system:masters

Presiona Enter para editar el ConfigMap ahora (o CTRL+C para cancelar)...
```

Copiamos el código que está en los recuadros rojos y presionamos enter para editar el configmap aws-auth dentro de mapRoles y mapUsers según corresponda. A continuación se detallan algunos de los parámetros:

#### **rolearn:**

Se indica el ARN completo del rol en AWS. Esta sección especifica que este rol de IAM es el que se utilizará para mapear a un usuario administrador dentro del clúster.

#### **username:**

Al mapear el rol, se le asigna el nombre de usuario "admin" en Kubernetes. Esto significa que cualquier entidad que asuma este rol será reconocida como "admin" dentro del clúster.

devops 2403	Grupo 1  PIN Final	mundosE
-------------	--------------------------	---------

**groups:**

Además, se asigna al rol dos grupos:

- **system:masters:**

Este es el grupo de Kubernetes con permisos administrativos completos, lo que permite realizar cualquier acción en el clúster.

Guardamos con :wq y los cambios se impactan de forma inmediata.

Verificamos los cambios con el siguiente comando

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/00-eks$ kubectl get configmap aws-auth -n kube-system -o yaml
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::194722402815:role/EKSIAccessRole
      username: admin
      groups:
        - system:masters
    - rolearn: arn:aws:iam::194722402815:role/eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-0ztZAoPbRLZ6
      username: system:node:{[EC2PrivateDNSName]}
  mapUsers: |
    - userarn: arn:aws:iam::194722402815:user/dcabral
      username: dcabral
      groups:
        - system:masters
kind: ConfigMap
metadata:
  creationTimestamp: "2025-03-13T21:05:49Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "7869"
  uid: 880f8811-f002-4547-a8d7-29f20cfcae4e0
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/00-eks$ 
```

Verificamos el rol creado:

devops 2403	<b>Grupo 1</b> <b>PIN Final</b>	mundosE
-------------	------------------------------------	---------

```

1- {
2-     "Version": "2012-10-17",
3-     "Statement": [
4-         {
5-             "Effect": "Allow",
6-             "Principal": [
7-                 "arn:aws:iam::194722402815:user/crico",
8-                 "arn:aws:iam::194722402815:user/dcabral",
9-                 "arn:aws:iam::194722402815:user/ahustaspique",
10                "arn:aws:iam::194722402815:user/afanetela",
11                "arn:aws:iam::194722402815:user/cgonzalez"
12            ],
13        },
14        {
15            "Action": "sts:AssumeRole"
16        }
17    ]
18 }

```

### Test IAM User access

Una vez aplicada la configuración anterior, verificamos su correcto funcionamiento. Para ello, ejecutaremos un script (01-iamuser-assumerole.sh) el cual realizará las siguientes acciones:

- Permite especificar el perfil (usuario iam) mediante la opción -u.
- Verifica si el perfil existe (usando aws configure list-profiles).
- Si no existe, ejecuta aws configure --profile <usuario> para que se configure (previamente debemos tener el AK/SK creada para ese usuario).
- Actualiza el kubeconfig para el cluster EKS asumiendo el rol indicado, y crea un contexto con un alias.

Este script es utilizado para crear el profile de nuestro usuario iam en el CLI.

### 02-change-to-iamuser.sh

```

#!/bin/bash
set -euo pipefail

#####
# - Permite especificar el perfil (usuario iam) mediante la opción -u.
# - Verifica si el perfil existe (usando aws configure list-profiles).
# - Si no existe, ejecuta aws configure --profile <usuario> para que se configure.
# - Actualiza el kubeconfig para el cluster EKS asumiendo el rol indicado, y crea un contexto con un alias.

# Función para mostrar el uso del script

```

```
usage() {  
    echo "Uso: $0 -u <usuario> [-c <cluster_name>] [-r <region>] [-a <alias>] [-R <role_arn>]"  
    echo ""  
    echo "  -u  Nombre del usuario IAM (AWS CLI profile) a utilizar (obligatorio)."   
    echo "  -c  Nombre del cluster EKS (por defecto: 2403-g1-pin-final)."   
    echo "  -r  Región AWS (por defecto: us-east-1)."   
    echo "  -a  Alias para el contexto kubeconfig (por defecto: <usuario>-eks)."   
    echo "  -R  ARN del rol a asumir (por defecto:  
arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole)."   
    exit 1  
}  
  
# Valores por defecto  
CLUSTER_NAME="2403-g1-pin-final"  
REGION="us-east-1"  
ROLE_ARN="arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole"  
  
# Procesar opciones  
while getopts ":u:c:r:a:R:" opt; do  
    case ${opt} in  
        u )  
            PROFILE="$OPTARG"  
            ;;  
        c )  
            CLUSTER_NAME="$OPTARG"  
            ;;  
        r )  
            REGION="$OPTARG"  
            ;;  
        a )  
            ALIAS="$OPTARG"  
            ;;  
        R )  
            ROLE_ARN="$OPTARG"  
            ;;  
        \? )  
            echo "Opción inválida: -$OPTARG" >&2  
            usage  
            ;;  
        : )  
            echo "La opción -$OPTARG requiere un argumento." >&2  
            usage  
            ;;  
    esac  
done  
# Verificar que el parámetro obligatorio -u (usuario) se haya proporcionado  
if [ -z "${PROFILE:-}" ]; then
```

```
echo "El parámetro -u <usuario> es obligatorio."
usage
fi
# Si no se especificó alias, usar "<usuario>-eks"
if [ -z "${ALIAS:-}" ]; then
    ALIAS="${PROFILE}-eks"
fi
# Verificar si el perfil existe
if ! aws configure list-profiles | grep -q "^${PROFILE}$"; then
    echo "El perfil '${PROFILE}' no existe. Ejecutando 'aws configure --profile ${PROFILE}...."
    aws configure --profile "${PROFILE}"
fi
echo "Actualizando kubeconfig para el cluster '${CLUSTER_NAME}' en la región '${REGION}'...."
echo "Usando el perfil '${PROFILE}' y asumiendo el rol '${ROLE_ARN}'"

aws eks update-kubeconfig \
    --name "${CLUSTER_NAME}" \
    --region "${REGION}" \
    --profile "${PROFILE}" \
    --role-arn "${ROLE_ARN}" \
    --alias "$ALIAS"

echo "Verificando los contextos existentes"
kubectl config get-contexts  #kubectl config current-context

echo "Cambiando al contexto '${ALIAS}'...."
kubectl config use-context "$ALIAS"

echo "Verificando acceso al cluster con 'kubectl get nodes'...."
kubectl get nodes

echo "El kubeconfig se ha actualizado y el rol ha sido asumido correctamente en el contexto '${ALIAS}'.."
```

Output:

En la salida vemos que el contexto se creó y asignó correctamente. También vemos que al ejecutar *kubectl get nodes* muestra los nodos del cluster.

devops 2403	<b>Grupo 1</b> <b>PIN Final</b>	mundosE
-------------	------------------------------------	---------

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/00-eks$ ./02-change-to-iamuser.sh -u dcabral
El perfil 'dcabral' no existe. Ejecutando 'aws configure --profile dcabral'...
AWS Access Key ID [None]: Al .T
AWS Secret Access Key [None]: l1 Z
Default region name [None]: us-east-1
Default output format [None]:
Actualizando kubeconfig para el cluster '2403-g1-pin-final' en la región 'us-east-1'...
Usando el perfil 'dcabral' y asumiendo el rol 'arn:aws:iam::194722402815:role/EKSIAccessRole'
Updated context dcabral-eks in /home/ubuntu/.kube/config
Verificando los contextos existentes
CURRENT      NAME                                     CLUSTER
          arn:aws:eks:us-east-1:194722402815:cluster/2403-g1-pin-final
*   dcabral-eks                                     arn:aws:eks:us-east-1:194722402815:cluster/2403-g1-pin-final
          i-037357b73672e7fd2@2403-g1-pin-final.us-east-1.eksctl.io  2403-g1-pin-final.us-east-1.eksctl.io
Cambiando al contexto 'dcabral-eks'...
Switched to context "dcabral-eks".
Verificando acceso al cluster con 'kubectl get nodes'...
NAME           STATUS  ROLES   AGE  VERSION
ip-192-168-12-6.ec2.internal  Ready    <none>  43m  v1.32.1-eks-5d632ec
ip-192-168-51-193.ec2.internal  Ready    <none>  43m  v1.32.1-eks-5d632ec
ip-192-168-73-160.ec2.internal  Ready    <none>  43m  v1.32.1-eks-5d632ec
El kubeconfig se ha actualizado y el rol ha sido asumido correctamente en el contexto 'dcabral-eks'.
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/00-eks$ 
```

## Consola Web

También vemos en la consola que luego de actualizar la sección de mapUsers ya podemos visualizar los nodos:



ice > Clusters > 2403-g1-pin-final

## 2403-g1-pin-final

[Edit](#) [Delete cluster](#) [View dashboard](#)

**Cluster info** [Info](#)

Status <a href="#">Active</a>	Kubernetes version <a href="#">Info</a> 1.32	Support period <a href="#">Standard support until March 21, 2026</a>	Provider EKS
Cluster health issues <a href="#">0</a>	Upgrade insights <a href="#">4</a>	Node health issues <a href="#">0</a>	

[Overview](#) [Resources](#) [Compute](#) [Networking](#) [Add-ons 1](#) [Access](#) [Observability](#)

### Nodes (3) [Info](#)

Filter Nodes by property or value

Node name	Instance type	Compute	Managed by	Created	Status
ip-192-168-25-154.ec2.internal	t2.small	Node group	ng-c0ddb93c	Created 7 hours ago	<a href="#">Ready</a>
ip-192-168-52-201.ec2.internal	t2.small	Node group	ng-c0ddb93c	Created 7 hours ago	<a href="#">Ready</a>
ip-192-168-91-252.ec2.internal	t2.small	Node group	ng-c0ddb93c	Created 7 hours ago	<a href="#">Ready</a>

<b>devops 2403</b>	<b>Grupo 1</b> <b>PIN Final</b>	
--------------------	------------------------------------	--

## NGINX

### Despliegue

Se despliega un nginx mediante un script en bash utilizando los archivos de manifiesto para el deployment y servicio. El pod con nginx estará expuesto públicamente a internet mediante un servicio de tipo LoadBalancer (creando un ELB en AWS):

nginx.sh

```
#!/bin/bash
set -euo pipefail

# Función de ayuda
usage() {
    echo "Uso: $0 [-d]"
    echo " -d  Modo delete: elimina el Deployment y el Service de Nginx en el namespace 'nginx' en"
    echo "lugar de crearlos."
    exit 1
}

# Procesar parámetros
DELETE_MODE=false
while getopts ":d" opt; do
    case ${opt} in
        d )
            DELETE_MODE=true
            ;;
        \? )
            usage
            ;;
    esac
done

NAMESPACE="nginx"
POD_NAME="nginx-deployment"
SERVICE_NAME="nginx-service"

# Archivos YAML originales y temporales (para corregir el namespace)
POD_YAML_ORIG="nginx-deployment.yaml"
SERVICE_YAML_ORIG="nginx-service.yaml"
POD_YAML_TEMP="nginx-deployment-temp.yaml"
SERVICE_YAML_TEMP="nginx-service-temp.yaml"

# Crear el namespace si no existe
if ! kubectl get ns "$NAMESPACE" >/dev/null 2>&1; then
    echo "El namespace '$NAMESPACE' no existe. Creándolo..."
fi
```

```
kubectl create namespace "$NAMESPACE"
fi

# Modificar los YAML para que usen el namespace "nginx"
# Se reemplaza "namespace: default" por "namespace: nginx" si está definido en el YAML
if grep -q "namespace:" "$POD_YAML_ORIG"; then
    sed "s/namespace:[[:space:]]*__namespace__/_namespace: ${NAMESPACE}/g" "$POD_YAML_ORIG" >
"$POD_YAML_TEMP"
else
    cp "$POD_YAML_ORIG" "$POD_YAML_TEMP"
fi

if grep -q "namespace:" "$SERVICE_YAML_ORIG"; then
    sed "s/namespace:[[:space:]]*__namespace__/_namespace: ${NAMESPACE}/g" "$SERVICE_YAML_ORIG" >
"$SERVICE_YAML_TEMP"
else
    cp "$SERVICE_YAML_ORIG" "$SERVICE_YAML_TEMP"
fi

if [ "$DELETE_MODE" = true ]; then
    echo "Modo delete activado: eliminando el Pod y el Service de Nginx en el namespace '$NAMESPACE'..."
    kubectl delete -f "$POD_YAML_TEMP" --namespace "$NAMESPACE" --ignore-not-found
    kubectl delete -f "$SERVICE_YAML_TEMP" --namespace "$NAMESPACE" --ignore-not-found
    echo "Recursos eliminados."
    # Eliminar archivos temporales
    rm -f "$POD_YAML_TEMP" "$SERVICE_YAML_TEMP"
    exit 0
fi

# Despliegue y validación
echo "Aplicando manifiesto para el Pod de Nginx en el namespace '$NAMESPACE'..."
kubectl apply -f "$POD_YAML_TEMP" --namespace "$NAMESPACE"

echo "Aplicando manifiesto para el Service de Nginx en el namespace '$NAMESPACE'..."
kubectl apply -f "$SERVICE_YAML_TEMP" --namespace "$NAMESPACE"

echo "Esperando a que al menos un Pod con la etiqueta 'app=nginx' alcance el estado 'Running' en el
namespace '$NAMESPACE'..."
while true; do
    POD_STATUS=$(kubectl get pods -n "$NAMESPACE" -l app=nginx --output
jsonpath='{.items[0].status.phase}' 2>/dev/null || echo "NotFound")
    if [ "$POD_STATUS" = "Running" ]; then
        echo "Al menos un Pod con la etiqueta 'app=nginx' está en estado Running."
        break
    elif [ "$POD_STATUS" = "NotFound" ]; then
        echo "Ningún Pod con la etiqueta 'app=nginx' encontrado. Esperando 5 segundos..."
    else
```

```
echo "Estado actual del primer Pod: $POD_STATUS. Esperando 5 segundos..."  
fi  
sleep 5  
done  
  
echo "Esperando que se asigne una IP/hostname público al Service '$SERVICE_NAME' en el namespace  
'$NAMESPACE'..."  
# Espera hasta que se asigne una IP/hostname al Service  
while true; do  
    EXTERNAL_IP=$(kubectl get svc "$SERVICE_NAME" -n "$NAMESPACE" --output  
jsonpath='{.status.loadBalancer.ingress[0].hostname}' 2>/dev/null || echo "")  
    if [ -z "$EXTERNAL_IP" ]; then  
        EXTERNAL_IP=$(kubectl get svc "$SERVICE_NAME" -n "$NAMESPACE" --output  
jsonpath='{.status.loadBalancer.ingress[0].ip}' 2>/dev/null || echo "")  
    fi  
    if [ -n "$EXTERNAL_IP" ]; then  
        echo "El Service '$SERVICE_NAME' está disponible en: $EXTERNAL_IP"  
        break  
    fi  
    echo "Esperando 10 segundos para que se asigne la IP/hostname..."  
    sleep 10  
done  
  
echo "Despliegue completado en el namespace '$NAMESPACE'. Puedes acceder a Nginx desde Internet  
utilizando el hostname/IP asignado."  
  
# Limpieza de archivos temporales  
rm -f "$POD_YAML_TEMP" "$SERVICE_YAML_TEMP"
```

### nginx-deployment.yaml

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
  # El namespace será definido por el script que ejecuta este manifiesto  
  namespace: __namespace__  
  labels:  
    app: nginx  
spec:  
  replicas: 3  
  selector:  
    matchLabels:  
      app: nginx  
  strategy:  
    type: RollingUpdate  
    rollingUpdate:
```

**devops 2403**

**Grupo 1**

**PIN Final**

**mundosE**

```
# Permite crear 1 pod adicional durante la actualización
maxSurge: 1
# No se permite que ningún pod esté inactivo durante la actualización
maxUnavailable: 0
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
      - name: nginx
        image: nginx:latest
        ports:
          - containerPort: 80
        resources:
          requests:
            cpu: "50m"
            memory: "64Mi"
          limits:
            cpu: "100m"
            memory: "128Mi"
```

**nginx-service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: __namespace__
spec:
  type: LoadBalancer
  ports:
    - port: 80          # Puerto que se expondrá públicamente
      targetPort: 80   # Puerto del contenedor
  selector:
    app: nginx       # Selecciona los pods con la etiqueta "app: nginx"
```

**Output:**

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/01-nginx$ ./nginx.sh
El namespace 'nginx' no existe. Creándolo...
namespace/nginx created
Aplicando manifiesto para el Pod de Nginx en el namespace 'nginx'...
deployment.apps/nginx-deployment created
Aplicando manifiesto para el Service de Nginx en el namespace 'nginx'...
service/nginx-service created
Esperando a que al menos un Pod con la etiqueta 'app=nginx' alcance el estado 'Running' en el namespace 'nginx'...
Estado actual del primer Pod: Pending. Esperando 5 segundos...
Al menos un Pod con la etiqueta 'app=nginx' está en estado Running.
Esperando que se asigne una IP/hostname público al Service 'nginx-service' en el namespace 'nginx'...
El Service 'nginx-service' está disponible en: a9c1c644a73e54b16af985b2f758e276-1298888930.us-east-1.elb.amazonaws.com
Despliegue completado en el namespace 'nginx'. Puedes acceder a Nginx desde Internet utilizando el hostname/IP asignado.
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/01-nginx$
```

## Comprobaciones:

Verificamos que el pod y el servicio hayan iniciado correctamente

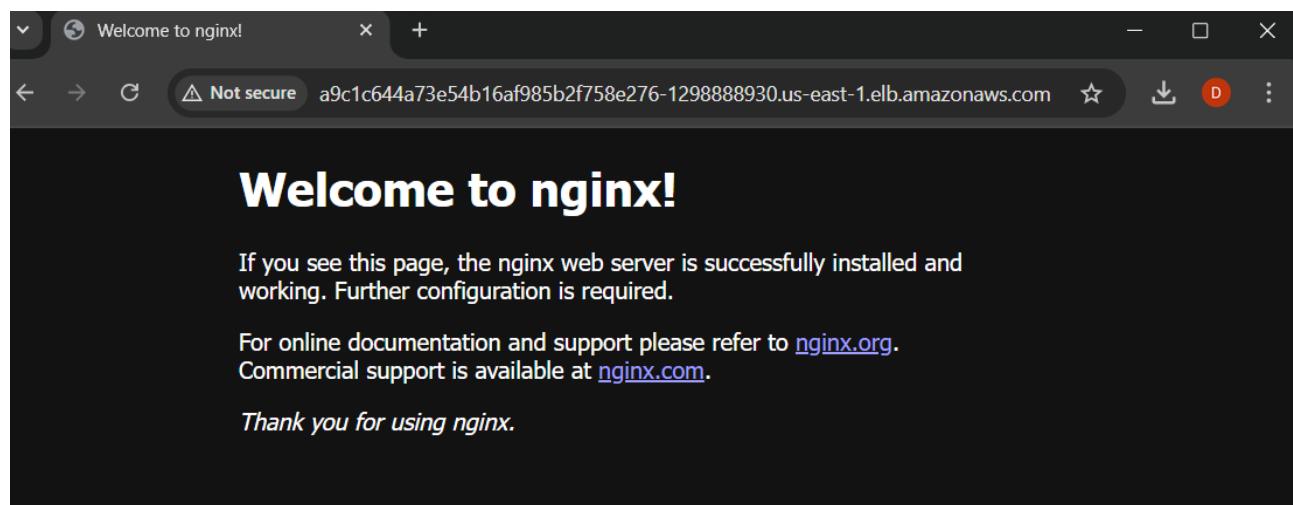
```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/01-nginx$ kubectl get all -n nginx
NAME                         READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-f7c784678-c8hcr   1/1     Running   0          39s
pod/nginx-deployment-f7c784678-hcd6s   1/1     Running   0          39s
pod/nginx-deployment-f7c784678-xz2ns   1/1     Running   0          39s

NAME                TYPE      CLUSTER-IP   EXTERNAL-IP           PORT(S)        AGE
service/nginx-service   LoadBalancer   10.100.9.46   a9c1c644a73e54b16af985b2f758e276-1298888930.us-east-1.elb.amazonaws.com   80:32278/TCP   38s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3           3           39s

NAME                           DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-f7c784678   3        3        3       39s
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/01-nginx$
```

Validamos el acceso al nginx mediante la url del elb sobre el puerto 80 ( En este ejemplo es <http://a9c1c644a73e54b16af985b2f758e276-1298888930.us-east-1.elb.amazonaws.com>)



devops 2403	<b>Grupo 1</b> <b>PIN Final</b>	mundosE
-------------	------------------------------------	---------

## EBS CSI Driver

El **EBS CSI Driver** es un plugin basado en la interfaz de almacenamiento de contenedores (Container Storage Interface, CSI) que permite a Kubernetes aprovisionar, administrar y eliminar volúmenes de almacenamiento de Amazon EBS (Elastic Block Store) de manera dinámica. Utilizado para:

- **Aprovisionamiento dinámico de volúmenes:** Permite crear volúmenes EBS de forma automática cuando se solicita un PersistentVolumeClaim (PVC) en Kubernetes.
- **Gestión de almacenamiento persistente:** Facilita el uso de volúmenes EBS como almacenamiento persistente para aplicaciones que se ejecutan en pods, garantizando que los datos se mantengan incluso si el pod se elimina o reinicia.
- **Integración con Kubernetes:** Funciona conforme al estándar CSI, lo que permite una integración nativa con la gestión de volúmenes de Kubernetes.
- **Operaciones de administración:** Soporta funciones como redimensionamiento (resizing), snapshots y eliminación de volúmenes, todo gestionado a través de las API de Kubernetes.

### Crear rol

Se crea el rol IAM llamado **AmazonEKS\_EBS\_CSI\_DriverRole** adjuntando la política **AmazonEBSCSIDriverPolicy**, y configura la relación de confianza necesaria para que el ServiceAccount (llamado "ebs-csi-controller-sa" y ubicado en el namespace "kube-system") pueda asumir ese rol. Debido a la opción --role-only, solo se crea el rol IAM y su configuración de confianza; el objeto ServiceAccount en Kubernetes deberá crearse por separado o ya existir, y deberá estar anotado con el ARN de este rol para que EKS permita que el controlador del driver EBS CSI utilice esos permisos.

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster 2403-g1-pin-final \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
    --approve \
    --role-only \
    --role-name AmazonEKS_EBS_CSI_DriverRole
```

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/02-ebs-csi-driver-tests$ eksctl create iamserviceaccount \
--name ebs-csi-controller-sa \
--namespace kube-system \
--cluster 2403-g1-pin-final \
--attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
--approve \
--role-only \
--role-name AmazonEKS_EBS_CSI_DriverRole
2025-03-13 22:02:50 [i] 1 iamserviceaccount (kube-system/ebs-csi-controller-sa) was included (based on the include/exclude rules)
2025-03-13 22:02:50 [!] serviceaccounts in Kubernetes will not be created or modified, since the option --role-only is used
2025-03-13 22:02:50 [i] 1 task: { create IAM role for serviceaccount "kube-system/ebs-csi-controller-sa" }
2025-03-13 22:02:50 [i] building iamserviceaccount stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-13 22:02:51 [i] deploying stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-13 22:02:51 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-13 22:03:21 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
```

El rol se crea para otorgar a los pods (o al controlador) que usan la service account los permisos necesarios para interactuar con los recursos de AWS de manera segura. En este caso, el rol (con la política AmazonEBSCSIDriverPolicy) permite al controlador del driver EBS CSI gestionar volúmenes EBS (crear, adjuntar, eliminar, etc.) en el cluster EKS. Al asociar este rol a la service account, se implementa el principio de "IAM Roles for Service Accounts" (IRSA), lo que garantiza que los pods obtengan únicamente los permisos específicos que necesitan.

## Instalación

Para la instalación del Driver se puede utilizar eksctl, helm o kubectl cada uno con sus pro y contras. Nosotros lo instalaremos mediante eksctl.

### eksctl

```
eksctl create addon \
--name aws-ebs-csi-driver \
--cluster 2403-g1-pin-final \
--service-account-role-arn arn:aws:iam::$(aws sts get-caller-identity --query "Account" --output text):role/AmazonEKS_EBS_CSI_DriverRole \
--force
```

**Pros:** Integración nativa con EKS, automatización de IRSA, gestión centralizada del addon.

**Contras:** Menos flexibilidad para personalizaciones muy específicas en los manifiestos.

### helm

```
helm repo add aws-ebs-csi-driver https://kubernetes-sigs.github.io/aws-ebs-csi-driver
helm repo update
helm upgrade --install aws-ebs-csi-driver \
--namespace kube-system \
aws-ebs-csi-driver/aws-ebs-csi-driver
```

**Pros:** Charts versionados, fácil actualización y rollback, y personalización a través de valores.

**Contras:** Puede requerir pasos adicionales para la configuración completa de IRSA y gestión de IAM.

### kubectl

```
kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-1.40"
```

**Pros:** Mayor control y flexibilidad al aplicar directamente manifiestos con Kustomize.

**Contras:** Requiere configuración manual de IRSA y ajustes adicionales, no se integra directamente en la consola de EKS como un addon.

Output:

En la siguiente imagen vemos que el driver se instaló correctamente mediante eksctl.

```
ubuntu@ip-172-31-90-146:~$ eksctl create addon \
--name aws-ebs-csi-driver \
--cluster 2403-g1-pin-final \
--service-account-role-arn arn:aws:iam::$(aws sts get-caller-identity --query "Account" --output text):role/AmazonEKS_EBS_CSI_DriverRole \
--force
2025-03-07 21:20:55 [i] Kubernetes version "1.32" in use by cluster "2403-g1-pin-final"
2025-03-07 21:20:56 [i] IRSA is set for "aws-ebs-csi-driver" addon; will use this to configure IAM permissions
2025-03-07 21:20:56 [!] the recommended way to provide IAM permissions for "aws-ebs-csi-driver" addon is via pod identity associations; after addon creation is completed, run `eksctl utils migrate-to-pod-identity`
2025-03-07 21:20:56 [i] using provided ServiceAccountRoleARN "arn:aws:iam::194722402815:role/AmazonEKS_EBS_CSI_DriverRole"
2025-03-07 21:20:56 [i] creating addon: aws-ebs-csi-driver
ubuntu@ip-172-31-90-146:~$
```

Validar funcionamiento

Para la validación crearemos un storageclass y pvc y lo asignaremos a un pod de prueba.

Una vez se crea el PersistentVolumeClaim (PVC) con ese StorageClass, el EBS driver se encargará de crear automáticamente un PV que se vincule a tu PVC.

01-test-create-volume.sh

Con este script crearemos el pod con un punto de montaje en un volumen. También permite el parámetro -d para proceder con la eliminación de los recursos creados.

```
#!/bin/bash
set -euo pipefail

usage() {
    echo "Uso: $0 [-d]"
    echo " -d Modo delete: elimina los recursos (StorageClass, PVC y Pod) creados para la prueba."
    exit 1
}

# Procesar parámetros
DELETE_MODE=false
while getopts ":d" opt; do
    case ${opt} in
        d )
            DELETE_MODE=true
    esac
done
```

```
DELETE_MODE=true
;;
\? )
usage
;;
esac
done

# Variables de configuración
NAMESPACE="default"
STORAGECLASS_YAML="02-storageclass.yaml"
PVC_YAML="03-pvc.yaml"
POD_YAML="04-pod-ebs-test.yaml"

PVC_NAME="test-ebs-pvc"
POD_NAME="test-ebs-pod"
SC_NAME="ebs-sc-gp3"

if [ "$DELETE_MODE" = true ]; then
    echo "Modo delete activado: eliminando StorageClass, PVC y Pod de la prueba..."

    # Eliminar recursos en este orden (el Pod y PVC dependen del StorageClass)
    kubectl delete -f "$POD_YAML" --namespace "$NAMESPACE" --ignore-not-found
    kubectl delete -f "$PVC_YAML" --namespace "$NAMESPACE" --ignore-not-found
    kubectl delete -f "$STORAGECLASS_YAML" --namespace "$NAMESPACE" --ignore-not-found

    echo "Recursos eliminados."
    exit 0
fi

# Despliegue de recursos
echo "Aplicando manifiesto para el StorageClass..."
kubectl apply -f "$STORAGECLASS_YAML"

echo "Aplicando manifiesto para el PersistentVolumeClaim..."
kubectl apply -f "$PVC_YAML" --namespace "$NAMESPACE"

echo "Aplicando manifiesto para el Pod de prueba..."
kubectl apply -f "$POD_YAML" --namespace "$NAMESPACE"

# Validación del PVC
echo "Validando que el PVC '$PVC_NAME' esté en estado 'Bound'..."
while true; do
    PVC_STATUS=$(kubectl get pvc "$PVC_NAME" -n "$NAMESPACE" --output jsonpath='{.status.phase}' 2>/dev/null || echo "NotFound")
    if [ "$PVC_STATUS" = "Bound" ]; then
        echo "El PVC '$PVC_NAME' se encuentra en estado 'Bound'."
        break
    fi
done
```

```
break
fi
echo "Estado actual del PVC: $PVC_STATUS. Esperando 5 segundos..."
sleep 5
done

# Validación del Pod
echo "Validando que el Pod '$POD_NAME' esté en estado 'Running'..."
while true; do
    POD_STATUS=$(kubectl get pod "$POD_NAME" -n "$NAMESPACE" --output jsonpath='{.status.phase}' 2>/dev/null || echo "NotFound")
    if [ "$POD_STATUS" = "Running" ]; then
        echo "El Pod '$POD_NAME' se encuentra en estado 'Running'."
        break
    fi
    echo "Estado actual del Pod: $POD_STATUS. Esperando 5 segundos..."
    sleep 5
done

# Mostrar recursos creados
echo "Mostrando los recursos creados en el namespace '$NAMESPACE':"
echo "StorageClass:"
kubectl get sc "$SC_NAME" -o wide || echo "No se encontró StorageClass '$SC_NAME'"
echo
echo "PersistentVolumeClaim:"
kubectl get pvc "$PVC_NAME" -n "$NAMESPACE" -o wide
echo
echo "Pod:"
kubectl get pod "$POD_NAME" -n "$NAMESPACE" -o wide

echo "Despliegue completado. Puedes acceder a Nginx utilizando el Service (si está expuesto) o verificar el contenido en el Pod."
```

## 02-storageclass.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc-gp3
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
```

## 03-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test-ebs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: ebs-sc-gp3
  resources:
    requests:
      storage: 1Gi
```

#### 04-pod-ebs-test.yaml

Este manifiesto crea un Pod llamado "test-ebs-pod" en el namespace "default", que ejecuta un contenedor basado en la imagen de Nginx. El contenedor monta un volumen (llamado "ebs-volume") en la ruta `/usr/share/nginx/html`, y este punto de montaje se conecta a través de un PersistentVolumeClaim llamado "test-ebs-pvc". Esto permite que el contenido de Nginx se almacene en un volumen persistente gestionado por Kubernetes (y, en este contexto, aprovisionado dinámicamente a través del driver EBS CSI).

```
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs-pod
  namespace: default
spec:
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
        - name: ebs-volume
          mountPath: /usr/share/nginx/html
  volumes:
    - name: ebs-volume
      persistentVolumeClaim:
        claimName: test-ebs-pvc
```

devops 2403	Grupo 1 PIN Final	mundosE
-------------	----------------------	---------

Output:

Luego de ejecutar 01-test-create-volume.sh

Se observa que el StorageClass está creado, el pvc se encuentra en estado Bound y el pod está ejecutándose.

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/02-ebs-csi-driver-tests$ ./01-test-create-volume.sh
Aplicando manifiesto para el StorageClass...
storageclass.storage.k8s.io/ebs-sc-gp3 created
Aplicando manifiesto para el PersistentVolumeClaim...
persistentvolumeclaim/test-ebs-pvc created
Aplicando manifiesto para el Pod de prueba...
pod/test-ebs-pod created
Validando que el PVC 'test-ebs-pvc' esté en estado 'Bound'...
Estado actual del PVC: Pending. Esperando 5 segundos...
El PVC 'test-ebs-pvc' se encuentra en estado 'Bound'.
Validando que el Pod 'test-ebs-pod' esté en estado 'Running'...
El Pod 'test-ebs-pod' se encuentra en estado 'Running'.
Mostrando los recursos creados en el namespace 'default':
StorageClass:
NAME      PROVISIONER      RECLAIMPOLICY      VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION      AGE
ebs-sc-gp3    ebs.csi.aws.com   Delete           WaitForFirstConsumer    false                   11s

PersistentVolumeClaim:
NAME      STATUS      VOLUME      CAPACITY      ACCESS MODES      STORAGECLASS      VOLUMEATTRIBUTESCLASS      AGE      VOLUMEMODE
test-ebs-pvc  Bound      pvc-39a63388-684e-41ab-b079-94ce05dc712a  1Gi          RWO      ebs-sc-gp3      <unset>      11s      Filesystem

Pod:
NAME      READY      STATUS      RESTARTS      AGE      IP          NODE      NOMINATED NODE      READINESS GATES
test-ebs-pod  1/1      Running     0          10s      192.168.80.38  ip-192-168-73-160.ec2.internal  <none>      <none>
Despliegue completado. Puedes acceder a Nginx utilizando el Service (si está expuesto) o verificar el contenido en el Pod.
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/02-ebs-csi-driver-tests$
```

Eliminamos los recursos de prueba

```
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/02-ebs-csi-driver-tests$ ./01-test-create-volume.sh -d
Modo delete activado: eliminando StorageClass, PVC y Pod de la prueba...
pod "test-ebs-pod" deleted
persistentvolumeclaim "test-ebs-pvc" deleted
Warning: deleting cluster-scoped resources, not scoped to the provided namespace
storageclass.storage.k8s.io "ebs-sc-gp3" deleted
Recursos eliminados.
ubuntu@ip-172-31-90-146:~/mundose.pinfinal.grupo1/aws/eks/02-ebs-csi-driver-tests$
```

devops 2403	Grupo 1 PIN Final	mundosE
-------------	----------------------	---------

## MONITOREO

### PROMETHEUS

Prometheus es un sistema de monitoreo y alerta de código abierto diseñado para recopilar y almacenar métricas en series de tiempo. Se utiliza ampliamente en entornos de microservicios y Kubernetes para:

- **Recopilar métricas:** Extrae datos de aplicaciones y servicios en intervalos regulares.
- **Almacenamiento de series de tiempo:** Guarda estos datos de manera eficiente para consultas históricas y en tiempo real.
- **Lenguaje de consulta PromQL:** Permite realizar consultas avanzadas para analizar y visualizar las métricas.
- **Alertas:** Integra reglas de alerta que pueden disparar notificaciones cuando se cumplen condiciones específicas.

#### Instalación

Realizaremos la instalación mediante un script que ejecuta helm.

`prometheus_install.sh`

#### Modo Delete:

Si se ejecuta con la opción -d, desinstala la release de Helm y elimina el namespace "prometheus", borrando todos los recursos relacionados.

#### Instalación:

- Agrega el repositorio de Helm de Prometheus Community y lo actualiza.
- Crea el namespace "prometheus" si aún no existe.
- Instala Prometheus usando un chart de Helm, configurando el almacenamiento persistente (usando la clase "gp2") y exponiendo el servicio del servidor mediante NodePort (en el puerto 32000).

#### Validación:

- Espera hasta que todos los Pods en el namespace "prometheus" estén en estado "Running".
- Muestra el estado de todos los recursos creados (Pods, Services, etc.) en el namespace.
- Lista los nodos del cluster junto con sus IPs y muestra la información del servicio NodePort.

#### Recordatorio:

Advierte que, para acceder al servicio desde el exterior, es posible que sea necesario

actualizar el grupo de seguridad de los nodos para permitir el tráfico entrante en el puerto 32000.

```
#!/bin/bash
set -euo pipefail

usage() {
    echo "Uso: $0 [-d]"
    echo " -d  Modo delete: elimina la instalación de Prometheus y el namespace 'prometheus'."
    exit 1
}

# Procesar parámetros
DELETE_MODE=false
while getopts ":d" opt; do
    case ${opt} in
        d )
            DELETE_MODE=true
            ;;
        \? )
            usage
            ;;
    esac
done

NAMESPACE="prometheus"
RELEASE_NAME="prometheus"

if [ "$DELETE_MODE" = true ]; then
    echo "Modo delete activado: eliminando la instalación de Prometheus..."
    helm uninstall "$RELEASE_NAME" --namespace "$NAMESPACE" || echo "La release '$RELEASE_NAME' no se encontró."
    kubectl delete namespace "$NAMESPACE" --ignore-not-found
    echo "El entorno de Prometheus ha sido eliminado."
    exit 0
fi

# Agregar el repositorio de Helm de Prometheus Community y actualizarlo
echo "Agregando el repositorio de Prometheus Community..."
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update

# Crear el namespace 'prometheus' si no existe
echo "Creando el namespace '$NAMESPACE'..."
kubectl create namespace "$NAMESPACE" || echo "El namespace '$NAMESPACE' ya existe."
```

```
# Instalar Prometheus usando el chart de prometheus-community
echo "Instalando Prometheus en el namespace '$NAMESPACE'..."
helm install "$RELEASE_NAME" prometheus-community/prometheus \
--namespace "$NAMESPACE" \
--set alertmanager.persistentVolume.storageClass="gp2" \
--set server.persistentVolume.storageClass="gp2" \
--set server.service.type="NodePort" \
--set server.service.nodePort=32000

# Esperar a que los Pods se encuentren en estado Running, con timeout de 60 segundos
TIMEOUT=60
WAIT_INTERVAL=10
elapsed=0
echo "Esperando a que todos los Pods en el namespace '$NAMESPACE' estén en estado 'Running'..."
while true; do
    NOT_RUNNING=$(kubectl get pods -n "$NAMESPACE" --field-selector=status.phase!=Running --no-headers | wc -l)
    if [ "$NOT_RUNNING" -eq 0 ]; then
        echo "Todos los Pods están en estado 'Running' ."
        break
    fi
    if [ "$elapsed" -ge "$TIMEOUT" ]; then
        echo "Timeout: No se pudieron levantar todos los Pods en $TIMEOUT segundos. Revisar manualmente."
        kubectl get all -n prometheus
        exit 1
    fi
    echo "Algunos Pods no están Running. Esperando $WAIT_INTERVAL segundos..."
    sleep $WAIT_INTERVAL
    elapsed=$((elapsed + WAIT_INTERVAL))
done

# Mostrar el estado de los recursos creados en el namespace
echo "Mostrando el estado de los recursos creados en el namespace '$NAMESPACE':"
kubectl get all -n "$NAMESPACE"

# Mostrar información de los nodos y sus IPs
echo "Mostrando la lista de nodos y sus IPs:"
kubectl get nodes -o wide

# Mostrar información del servicio para verificar el NodePort
echo "Mostrando la información del servicio del Prometheus Server:"
kubectl get svc -n "$NAMESPACE" | grep -i nodeport

echo "Prometheus se ha instalado correctamente y todos los recursos están en estado saludable."
echo "Recuerda que, para acceder al servicio desde fuera, es posible que necesites actualizar el grupo de seguridad de los nodos para permitir el tráfico en el puerto 32000."
```

devops 2403	Grupo 1  PIN Final	mundosE
-------------	--------------------------	---------

Output:

```
ubuntu@ip-172-31-90-146:~$ ./prometheus_install.sh
Agregando el repositorio de Prometheus Community...
"prometheus-community" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "grafana" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helming!*
Creando el namespace 'prometheus'...
namespace/prometheus created
Instalando Prometheus en el namespace 'prometheus'...
NAME: prometheus
LAST DEPLOYED: Fri Mar 7 22:41:27 2025
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local

Get the Prometheus server URL by running these commands in the same shell:
export NODE_PORT=$(kubectl get --namespace prometheus -o jsonpath=".spec.ports[0].nodePort" services prometheus-server)
export NODE_IP=$(kubectl get nodes --namespace prometheus -o jsonpath=".items[0].status.addresses[0].address")
echo http://$NODE_IP:$NODE_PORT
Esperando a que todos los Pods en el namespace prometheus esten en estado running ...
Algunos Pods no están Running. Esperando 10 segundos...
Timeout: No se pudieron levantar todos los Pods en 60 segundos.
ubuntu@ip-172-31-90-146:~$
```

**Server URL:** prometheus-server.prometheus.svc.cluster.local

**Alertmanager URL:** prometheus-alertmanager.prometheus.svc.cluster.local

**Pushgateway URL:** prometheus-prometheus-pushgateway.prometheus.svc.cluster.local

Configurar NodePort (opcional)

En caso que el servicio este en ClusterIP, podemos actualizar la instalación de Prometheus a fin de modificar el tipo de Service que expone el servidor de Prometheus, de **ClusterIP** a **NodePort**, permitiendo el acceso externo a través de un puerto fijo en cada nodo del clúster. Asimismo reutilizamos la configuración ya realizada anteriormente mediante el parámetro “`--reuse-values`” y solo modificando aquellos que estén en “`--set`”

```
helm upgrade prometheus prometheus-community/prometheus \
--namespace prometheus \
--reuse-values \
--set server.service.type="NodePort" \
--set server.service.nodePort=32000
```

**devops 2403**

**Grupo 1**

**PIN Final**

**mundosE**

```
ubuntu@ip-172-31-90-146:~$ kubectl get nodes -o wide
NAME           STATUS  ROLES   AGE    VERSION
ip-192-168-21-236.ec2.internal  Ready   <none>  26h   v1.32.1-eks-5d632ec  192.168.21.236  184.72.74.132
ip-192-168-59-68.ec2.internal  Ready   <none>  26h   v1.32.1-eks-5d632ec  192.168.59.68   18.212.133.56
ip-192-168-74-238.ec2.internal  Ready   <none>  26h   v1.32.1-eks-5d632ec  192.168.74.238  35.169.107.6
```

Output:

```
ubuntu@ip-172-31-90-146:~$ helm upgrade prometheus prometheus-community/prometheus \
--namespace prometheus \
--reuse-values \
--set server.service.type="NodePort" \
--set server.service.nodePort=32000
Release "prometheus" has been upgraded. Happy Helming!
NAME: prometheus
LAST DEPLOYED: Tue Mar  4 23:45:18 2025
NAMESPACE: prometheus
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local
```

Get the Prometheus server URL by running these commands in the same shell:

```
export NODE_PORT=$(kubectl get --namespace prometheus -o jsonpath=".spec.ports[0].nodePort" services prometheus-server)
export NODE_IP=$(kubectl get nodes --namespace prometheus -o jsonpath=".items[0].status.addresses[0].address")
echo http://$NODE_IP:$NODE_PORT
```

Validación:

Al ejecutar el comando “*kubectl get all -n prometheus*”, observamos que el pod *prometheus-alertmanager-0* se encuentra en estado pendiente.

```
NAME                                         READY  STATUS    RESTARTS  AGE
pod/prometheus-alertmanager-0                0/1    Pending   0          46m
pod/prometheus-kube-state-metrics-5bd466f7f6-8ndjd  1/1    Running  0          46m
pod/prometheus-prometheus-node-exporter-msgq2  1/1    Running  0          46m
pod/prometheus-prometheus-node-exporter-nt98w   1/1    Running  0          46m
pod/prometheus-prometheus-node-exporter-rhtmg   1/1    Running  0          46m
pod/prometheus-prometheus-pushgateway-544579d549-w7ftm  1/1    Running  0          46m
pod/prometheus-server-596945876b-j4csn        2/2    Running  0          46m

NAME                           TYPE            CLUSTER-IP      EXTERNAL-IP    PORT(S)         AGE
service/prometheus-alertmanager  ClusterIP       10.100.243.103  <none>        9093/TCP       46m
service/prometheus-alertmanager-headless  ClusterIP       None           <none>        9093/TCP       46m
service/prometheus-kube-state-metrics  ClusterIP       10.100.1.127   <none>        8080/TCP       46m
service/prometheus-prometheus-node-exporter  ClusterIP       10.100.203.106  <none>        9100/TCP       46m
service/prometheus-prometheus-pushgateway  ClusterIP       10.100.137.149  <none>        9091/TCP       46m
service/prometheus-server           ClusterIP       10.100.218.243  <none>        80/TCP         46m

NAME                                DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
daemonset.apps/prometheus-prometheus-node-exporter  3        3        3      3           3          kubernetes.io/os=linux  46m

NAME                                READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/prometheus-kube-state-metrics  1/1    1          1          46m
deployment.apps/prometheus-prometheus-pushgateway  1/1    1          1          46m
deployment.apps/prometheus-server             1/1    1          1          46m

NAME                                DESIRED  CURRENT  READY  AGE
replicaset.apps/prometheus-kube-state-metrics-5bd466f7f6  1        1        1      46m
replicaset.apps/prometheus-prometheus-pushgateway-544579d549  1        1        1      46m
replicaset.apps/prometheus-server-596945876b                 1        1        1      46m
```

devops 2403

Grupo 1

PIN Final

mundosE

Troubleshooting pod alertmanager

Procedemos a analizar porque no inicia.

Comandos para evaluar el estado de pods

```
kubectl get pod -n prometheus #Para identificar el nombre del pod con problemas
```

```
kubectl describe pod prometheus-alertmanager-0 -n prometheus #Para ver los detalles y eventos del pod
```

Observamos en la sección de eventos que no pudo adjuntar los PVC a los nodos del pod no pudiendo así iniciar el mismo.

```
Events:
Type    Reason     Age           From            Message
----    ----     --            --             -----
Warning FailedScheduling 57s (x17 over 73m) default-scheduler 0/3 nodes are available: pod has unbound immediate PersistentVolumeClaims. preemption: 0/3 nodes are available: 3 Preemption is not helpful for scheduling.
```

Comandos para validar el estado de los PVC

```
#Identificar los volumenes y estados que tienen en el namespace prometheus
kubectl get pvc -n prometheus
# Mostrar en detalle el estado del pvc afectado
kubectl describe pvc storage-prometheus-alertmanager-0 -n prometheus
```

```
ubuntu@ip-172-31-90-146:~$ kubectl get pvc -n prometheus
NAME          STATUS    VOLUME          CAPACITY   ACCESS MODES  STORAGECLASS  VOLUME ATTRIBUTESCLASS AGE
prometheus-server   Bound    pvc-9ea2186f-e3d7-4c5a-9c2d-39034b7f73dd  8Gi        RWO          gp2          <unset>      <unset>          79m
storage-prometheus-alertmanager-0 Pending
ubuntu@ip-172-31-90-146:~$ kubectl describe pvc storage-prometheus-alertmanager-0 -n prometheus
Name:          storage-prometheus-alertmanager-0
Namespace:    prometheus
StorageClass: [redacted]
Status:       Pending
Volume:
Labels:       app.kubernetes.io/instance=prometheus
              app.kubernetes.io/name=alertmanager
Annotations: <none>
Finalizers:  [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:  Filesystem
Used By:     prometheus-alertmanager-0
Events:
Type    Reason     Age           From            Message
----    ----     --            --             -----
Normal  FailedBinding  3m35s (x322 over 83m) persistentvolume-controller [redacted] no persistent volumes available for this claim and no storage class is set
```

Vemos que no tiene un StorageClass asignado.

Como el PVC ya está creado procedemos a especificar el StorageClass en el PVC mediante el siguiente comando

```
kubectl patch pvc storage-prometheus-alertmanager-0 -n prometheus -p '{"spec":{"storageClassName": "gp2"}}'
```

Después de unos minutos (ya que demora un tiempo en crear el volumen) volvemos a validar el estado del pod y pvc mediante los siguientes comandos

```
kubectl get pods -n prometheus
kubectl get pvc -n prometheus
kubectl describe pvc storage-prometheus-alertmanager-0 -n prometheus
```

Observamos que el problema se solucionó y ahora el volumen se encuentra asignado

```
ubuntu@ip-172-31-90-146:~$ kubectl describe pvc storage-prometheus-alertmanager-0 -n prometheus
Name:           storage-prometheus-alertmanager-0
Namespace:      prometheus
StorageClass:   gp2
Status:         Bound
Volume:         pvc-90c2836d-b3e4-4908-a728-1878f342f2a5
Labels:         app.kubernetes.io/instance=prometheus
                app.kubernetes.io/name=alertmanager
Annotations:   pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner: ebs.csi.aws.com
                volume.kubernetes.io/selected-node: ip-192-168-91-252.ec2.internal
                volume.kubernetes.io/storage-provisioner: ebs.csi.aws.com
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:       2Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Used By:       prometheus-alertmanager-0
Events:
  Type  Reason          Age             From            Message
  ----  ----          --             --            -----
  Normal FailedBinding  12m (x26 over 18m)  persistentvolume-controller  no persistentvolume found
  Normal WaitForPodScheduled  8m15s (x4 over 8m53s)  persistentvolume-controller  waiting for pod to be scheduled
  Normal Provisioning    3m25s          ebs.csi.aws.com_ebs-csi-controller-5f65cc986-xh798_61c9e5bc-f663-4c33-8334-ec5154bb9ec4  External provisioner is provisioning volume...
  Normal ProvisioningSucceeded  3m23s          ebs.csi.aws.com_ebs-csi-controller-5f65cc986-xh798_61c9e5bc-f663-4c33-8334-ec5154bb9ec4  Successfully provisioned volume v...

```

devops 2403	<b>Grupo 1</b> <b>PIN Final</b>	mundosE
-------------	------------------------------------	---------

Por último vemos que los pvc y los pods del prometheus están ejecutándose OK.

```
ubuntu@ip-172-31-90-146:~$ kubectl get pvc -n prometheus
NAME                      STATUS   VOLUME                                     CAPACITY   ACCESS MODES  STORAGECLASS   VOLUME ATTRIBUTESCLASS   AGE
prometheus-server          Bound    pvc-c05da9a7-e317-4638-b044-ce2c8e2ef483   8Gi       RWO          gp2            <unset>        <unset>           21m
storage-prometheus-alertmanager-0   Bound    pvc-90c2836d-b3e4-4908-a728-1878f342f2a5   2Gi       RWO          gp2            <unset>        <unset>           21m

ubuntu@ip-172-31-90-146:~$ kubectl get all -n prometheus
NAME                                         READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                1/1     Running   0          22m
pod/prometheus-kube-state-metrics-5bd466f7f6-dq24c   1/1     Running   0          22m
pod/prometheus-prometheus-node-exporter-dxht6      1/1     Running   0          22m
pod/prometheus-prometheus-node-exporter-krvnn      1/1     Running   0          22m
pod/prometheus-prometheus-node-exporter-mwjqs      1/1     Running   0          22m
pod/prometheus-prometheus-pushgateway-544579d549-wnwhj   1/1     Running   0          22m
pod/prometheus-server-596945876b-1kl162          2/2     Running   0          22m

NAME                           TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)   AGE
service/prometheus-alertmanager   ClusterIP  10.100.18.105  <none>      9093/TCP  22m
service/prometheus-alertmanager-headless   ClusterIP  None          <none>      9093/TCP  22m
service/prometheus-kube-state-metrics   ClusterIP  10.100.93.181  <none>      8080/TCP  22m
service/prometheus-prometheus-node-exporter   ClusterIP  10.100.240.198  <none>      9100/TCP  22m
service/prometheus-prometheus-pushgateway   ClusterIP  10.100.91.17   <none>      9091/TCP  22m
service/prometheus-server          NodePort    10.100.173.205  <none>      80:32000/TCP  22m

NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter   3         3         3         3             3           kubernetes.io/os=linux  22m

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics   1/1     1           1           22m
deployment.apps/prometheus-prometheus-pushgateway   1/1     1           1           22m
deployment.apps/prometheus-server                1/1     1           1           22m

NAME                           DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-5bd466f7f6   1         1         1         22m
replicaset.apps/prometheus-prometheus-pushgateway-544579d549   1         1         1         22m
replicaset.apps/prometheus-server-596945876b          1         1         1         22m

NAME   READY   AGE
statefulset.apps/prometheus-alertmanager   1/1     22m
ubuntu@ip-172-31-90-146:~$
```

Port forward access:

En nuestro bastion host ejecutamos el siguiente comando

```
kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090 --address 0.0.0.0
```

```
ubuntu@ip-172-31-90-146:~$ kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090 --address 0.0.0.0
Forwarding from 0.0.0.0:8080 -> 9090
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
[]
```

También podemos ejecutarlo en background enviando el output a un archivo de logs mediante el siguiente comando

```
sudo mkdir -p /var/Log/eks/ && sudo chown $(whoami) /var/Log/eks && kubectl port-forward -n prometheus
deploy/prometheus-server 8080:9090 --address 0.0.0.0 > /var/Log/eks/prometheus-port-forward.log 2>&1 &
```



Habilitamos el tráfico entrante del Security Group “bastion-sg” para acceder al Prometheus (en este ejemplo 8080).

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sg-0e86ae78e5bb3d7c3	SSH	TCP	22	Cu... 0.0.0.0/0	Allow SSH Access Delete
-	Custom TCP	TCP	8080	An... 0.0.0.0/0	Prometheus Delete

**Add rule**

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

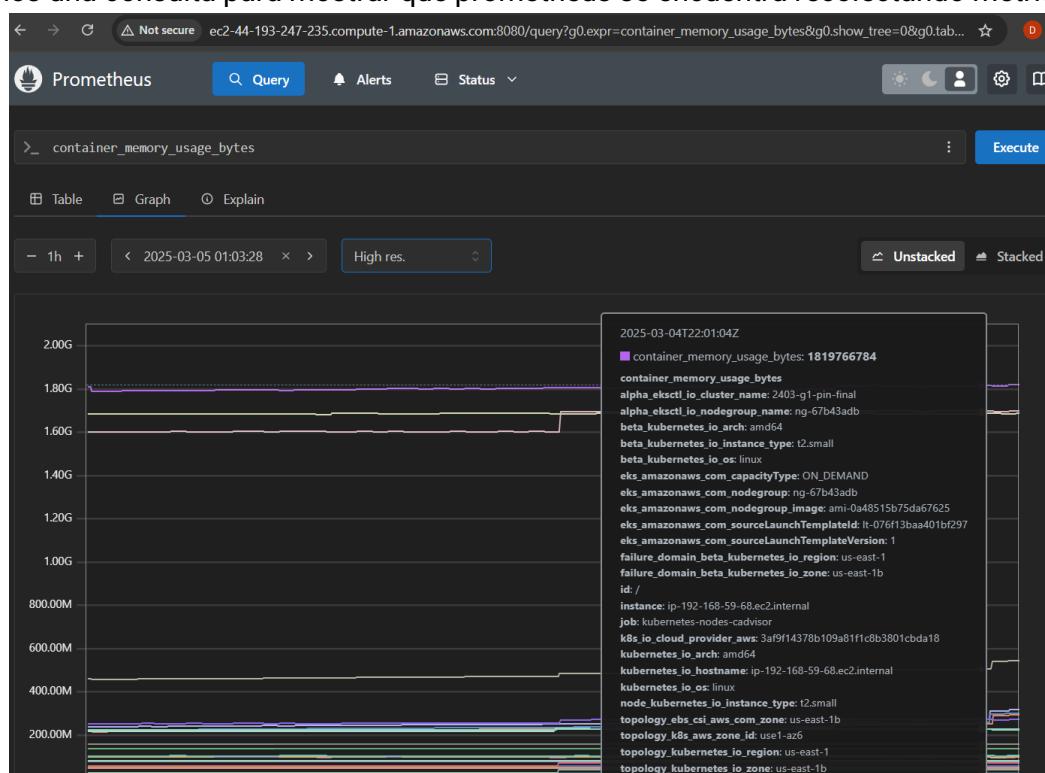
**Cancel** **Preview changes** **Save rules**

En una nueva pestaña del navegador ponemos la url <http://ec2-44-193-247-235.compute-1.amazonaws.com:8080/>



Nos dirigimos a “Status -> Target health” donde vemos el estado de los diferentes endpoints de nuestro cluster de EKS.

Ejecutamos una consulta para mostrar que prometheus se encuentra recolectando métricas.





Node port access:

Como habíamos configurado prometheus con un servicio de tipo NodePort podemos acceder de forma pública desde las ips públicas de los nodos. Recordar habilitar los puertos correspondientes en el security groups del nodegroup.

Security Group de los nodos:

Agregamos el puerto 32000 para permitir el acceso a la web de prometheus

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0c1af0d557480aca7	SSH	TCP	22	Cu... ::/0	Allow SSH access to managed worker nodes in group ng-c0ddb93c
sgr-0fe979657631d8ec9	SSH	TCP	22	Cu... 0.0.0.0/0	Allow SSH access to managed worker nodes in group ng-c0ddb93c
-	Custom TCP	TCP	32000	An... 0.0.0.0/0	Allow access to promett

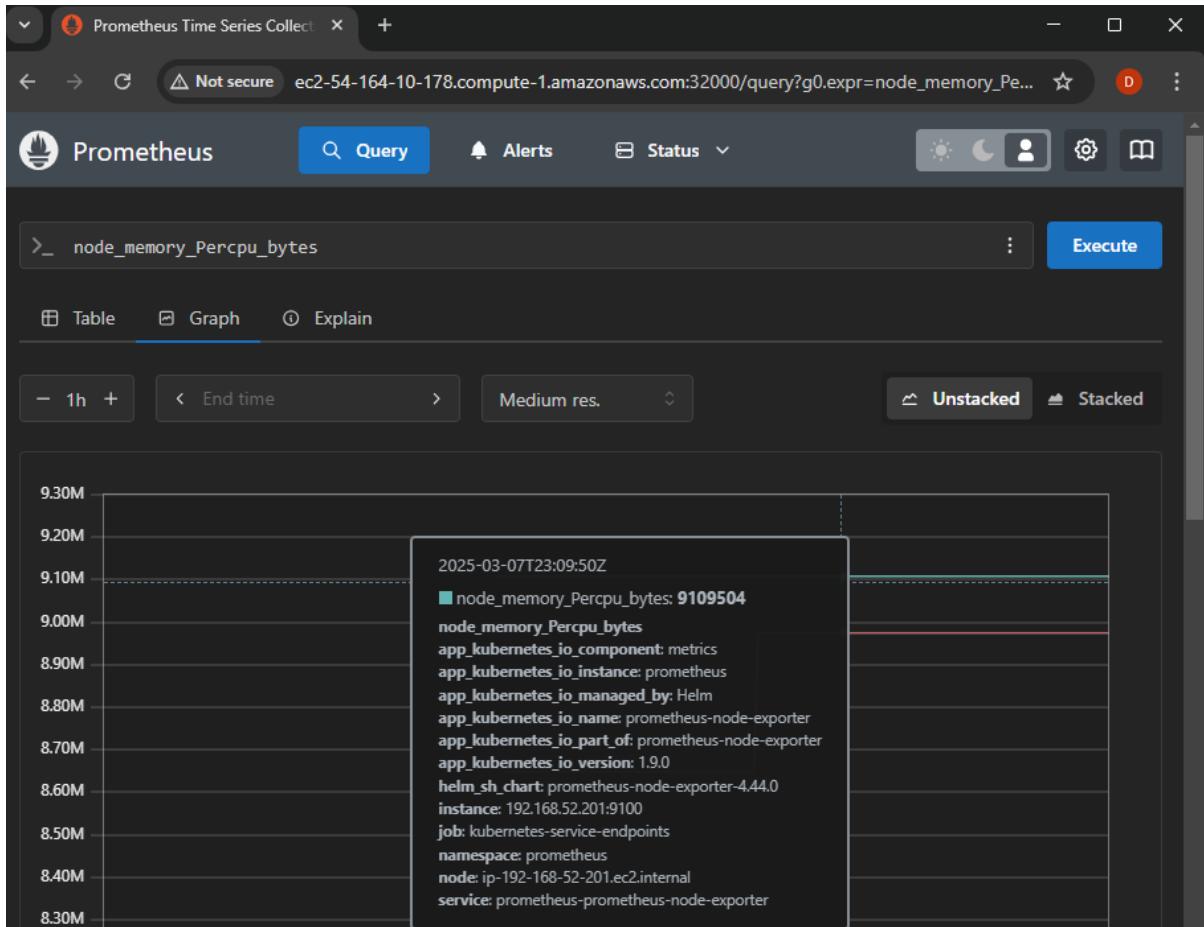
**Add rule**

**Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.**

**Cancel** **Preview changes** **Save rules**



http://<eks-node-public-ip-url>:32000



## GRAFANA

### Instalación

La instalación se realiza mediante un script que ejecuta helm y un archivo de manifiesto yaml.

Creamos en nuestro el directorio y creamos el archivo grafana.yaml

```
mkdir -p ${HOME}/environment/grafana  
cd ${HOME}/environment/grafana  
vim grafana.yaml
```

```
ubuntu@ip-172-31-90-146:~$ mkdir -p ${HOME}/environment/grafana  
ubuntu@ip-172-31-90-146:~$ cd ${HOME}/environment/grafana  
ubuntu@ip-172-31-90-146:~/environment/grafana$ pwd  
/home/ubuntu/environment/grafana  
ubuntu@ip-172-31-90-146:~/environment/grafana$ vim grafana.yaml  
  
grafana.yaml  
# grafana.yaml  
replicaCount: 1  
  
grafana:  
  adminUser: admin  
  adminPassword: "" # Este valor se sobreescribe con el flag --set  
  # Ejemplo de configuración de datasource (opcional)  
  datasources:  
    datasources.yaml:  
      apiVersion: 1  
      datasources:  
        - name: Prometheus  
          type: prometheus  
          access: proxy  
          url: http://prometheus-server.prometheus.svc.cluster.local  
          isDefault: true  
  
    persistence:  
      enabled: true  
      storageClassName: gp2  
      accessModes:  
        - ReadWriteOnce  
      size: 10Gi  
  
    service:  
      type: LoadBalancer  
      port: 80
```

<b>devops 2403</b>	<b>Grupo 1</b> <b>PIN Final</b>	<b>mundosE</b>
--------------------	------------------------------------	----------------

devops 2403	Grupo 1 PIN Final	mundosE
-------------	----------------------	---------

grafana\_install.sh

#### Modo Delete (-d):

- Si se pasa la opción -d, el script desinstala la release de Helm y elimina el namespace grafana, eliminando todo el entorno.

#### Gestión de la contraseña:

- La contraseña de Grafana se suministra a través de la variable de entorno GRAFANA\_ADMIN\_PASSWORD. Esto evita que la contraseña esté codificada en el script y pueda ser subida a GitHub. Además, el archivo grafana.yaml ubicado en \${HOME}/environment/grafana se utiliza para almacenar la configuración personalizada. Se recomienda agregar ese directorio o archivo a .gitignore para proteger la información sensible.

#### Instalación de Grafana:

El script agrega el repositorio de Helm oficial de Grafana, crea el namespace (si no existe) e instala Grafana con los parámetros configurados (persistencia con gp2, servicio de tipo LoadBalancer, etc.).

#### Verificación:

Muestra los recursos creados en el namespace para confirmar que la instalación fue exitosa.

```
#!/bin/bash
set -euo pipefail

usage() {
    echo "Uso: $0 [-d]"
    echo "  -d  Modo delete: elimina la instalación de Grafana y el namespace 'grafana'."
    exit 1
}

# Procesar parámetros
DELETE_MODE=false
while getopts ":d" opt; do
    case ${opt} in
        d )
            DELETE_MODE=true
            ;;
        \? )
            usage
            ;;
    esac
done
```

```
NAMESPACE="grafana"
RELEASE_NAME="grafana"
VALUES_FILE="${HOME}/environment/grafana/grafana.yaml"

# La contraseña de admin se toma de la variable de entorno GRAFANA_ADMIN_PASSWORD;
# si no se define, se usa un valor por defecto.
ADMIN_PASSWORD=${GRAFANA_ADMIN_PASSWORD:-"MSE!pinfinalG1."}

if [ "$DELETE_MODE" = true ]; then
    echo "Modo delete activado: eliminando la instalación de Grafana..."
    helm uninstall "$RELEASE_NAME" --namespace "$NAMESPACE" || echo "La release '$RELEASE_NAME' no se encontró."
    kubectl delete namespace "$NAMESPACE" --ignore-not-found
    echo "El entorno de Grafana ha sido eliminado."
    exit 0
fi

# Crear el namespace 'grafana' si no existe
echo "Creando el namespace '$NAMESPACE'..."
kubectl create namespace "$NAMESPACE" || echo "El namespace '$NAMESPACE' ya existe."

# Agregar el repositorio de Helm de Grafana y actualizar la caché
echo "Agregando el repositorio de Helm de Grafana..."
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update

# Instalar Grafana usando Helm
# --namespace grafana: Se instala en el namespace "grafana".
# --set persistence.storageClassName="gp2": Se establece "gp2" como StorageClass para la persistencia.
# --set persistence.enabled=true: Se habilita la persistencia de datos.
# --set adminPassword='xxxxxxxx': Se define la contraseña de administrador de Grafana.
# --values ${HOME}/environment/grafana/grafana.yaml: Se cargan valores adicionales desde un archivo YAML.
# --set service.type=LoadBalancer: Se configura el servicio para que sea de tipo LoadBalancer.

echo "Instalando Grafana en el namespace '$NAMESPACE'..."
helm install "$RELEASE_NAME" grafana/grafana \
--namespace "$NAMESPACE" \
--set persistence.storageClassName="gp2" \
--set persistence.enabled=true \
--set adminPassword="$ADMIN_PASSWORD" \
--values "$VALUES_FILE" \
--set service.type="LoadBalancer"

# Esperar a que los Pods de Grafana estén en estado Running, con timeout de 60 segundos
TIMEOUT=60
INTERVAL=10
```

```
ELAPSED=0
echo "Esperando a que todos los Pods en el namespace '$NAMESPACE' estén en estado 'Running' (timeout ${TIMEOUT} segundos)..."
while true; do
    NOT_RUNNING=$(kubectl get pods -n "$NAMESPACE" --field-selector=status.phase!=Running --no-headers | wc -l)
    if [ "$NOT_RUNNING" -eq 0 ]; then
        echo "Todos los Pods están en estado 'Running'."
        break
    fi
    if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
        echo "Timeout: No se pudieron levantar todos los Pods en $TIMEOUT segundos."
        exit 1
    fi
    echo "Algunos Pods no están Running. Esperando ${INTERVAL} segundos..."
    sleep $INTERVAL
    ELAPSED=$((ELAPSED + INTERVAL))
done

# Mostrar los recursos creados en el namespace
echo "Mostrando el estado de los recursos creados en el namespace '$NAMESPACE':"
kubectl get all -n "$NAMESPACE"

# Mostrar la información de los nodos y sus IPs
echo "Mostrando la lista de nodos y sus IPs:"
kubectl get nodes -o wide

# Mostrar la información del servicio para verificar el NodePort
echo "Mostrando la información del servicio de Grafana:"
kubectl get svc -n "$NAMESPACE" | grep -i nodeport

echo "La instalación de Grafana se ha completado correctamente en el namespace '$NAMESPACE'." 
echo "Recuerda que, para acceder al servicio desde fuera, es posible que necesites actualizar el grupo de seguridad de los nodos para permitir el tráfico en el puerto configurado."
```

devops 2403

Grupo 1

PIN Final

mundosE

Output:

```
ubuntu@ip-172-31-90-146:~$ ./grafana_install.sh
Creando el namespace 'grafana'...
namespace/grafana created
Agregando el repositorio de Helm de Grafana...
"grafana" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "grafana" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helm-ing!*
Instalando Grafana en el namespace 'grafana'...
NAME: grafana
LAST DEPLOYED: Fri Mar  7 23:49:16 2025
NAMESPACE: grafana
STATUS: deployed
REVISION: 1
NOTES:
1. Get your 'admin' user password by running:

  kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

2. The Grafana server can be accessed via port 80 on the following DNS name from within your cluster:

  grafana.grafana.svc.cluster.local

  Get the Grafana URL to visit by running these commands in the same shell:
  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
  You can watch the status of by running 'kubectl get svc --namespace grafana -w grafana'
  export SERVICE_IP=$(kubectl get svc --namespace grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
  http://$SERVICE_IP:80

3. Login with the password from step 1 and the username: admin
Esperando a que todos los Pods en el namespace 'grafana' estén en estado 'Running' (timeout 60 segundos)...
Algunos Pods no están Running. Esperando 10 segundos...
No resources found in grafana namespace.
Todos los Pods están en estado 'Running'.
Mostrando el estado de los recursos creados en el namespace 'grafana':
NAME          READY   STATUS    RESTARTS   AGE
pod/grafana-5966b67df7-jxnpp   0/1     Running   0          12s

NAME           TYPE      CLUSTER-IP        EXTERNAL-IP          PORT(S)         AGE
service/grafana   LoadBalancer   10.100.242.92   a813e86dcc01f4a69bac3b62751a197e-550556558.us-east-1.elb.amazonaws.com   80:31271/TCP   13s

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana  0/1       1            0           13s

NAME          DESIRED  CURRENT   READY   AGE
replicaset.apps/grafana-5966b67df7  1         1         0        13s
Mostrando la lista de nodos y sus IPs:
NAME          STATUS   ROLES   AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE   KERNEL-VERS
ip-192-168-25-154.ec2.internal   Ready   <none>  11h  v1.32.1-eks-5d632ec  192.168.25.154  54.164.10.178  Amazon Linux 2  5.10.234-22
ip-192-168-52-201.ec2.internal   Ready   <none>  11h  v1.32.1-eks-5d632ec  192.168.52.201  52.207.104.215  Amazon Linux 2  5.10.234-22
ip-192-168-91-252.ec2.internal   Ready   <none>  11h  v1.32.1-eks-5d632ec  192.168.91.252  44.203.247.58   Amazon Linux 2  5.10.234-22
```

Verificación y configs:

EKS

```
ubuntu@ip-172-31-90-146:~$ kubectl get all -n grafana
NAME                         READY   STATUS    RESTARTS   AGE
pod/grafana-5966b67df7-jxnpp   1/1     Running   0          5m38s

NAME            TYPE      CLUSTER-IP        EXTERNAL-IP
service/grafana  LoadBalancer  10.100.242.92   a813e86dcc01f4a69bac3b62751a197e-550556558.us-east-1.elb.amazonaws.com

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana  1/1       1           1          5m39s

NAME           DESIRED  CURRENT   READY   AGE
replicaset.apps/grafana-5966b67df7  1         1         1        5m39s
ubuntu@ip-172-31-90-146:~$
```

Busco la url del LoadBalancer Ingress:

```
ubuntu@ip-172-31-90-146:~$ kubectl describe svc grafana -n grafana
Name:           grafana
Namespace:      grafana
Labels:         app.kubernetes.io/instance=grafana
                app.kubernetes.io/managed-by=Helm
                app.kubernetes.io/name=grafana
                app.kubernetes.io/version=11.5.2
                helm.sh/chart=grafana-8.10.1
Annotations:    meta.helm.sh/release-name: grafana
                meta.helm.sh/release-namespace: grafana
Selector:       app.kubernetes.io/instance=grafana,app.kubernetes.io/name=grafana
Type:          LoadBalancer
IP Family Policy: SingleStack
IP Families:   IPv4
IP:             10.100.242.92
IPs:            10.100.242.92
LoadBalancer Ingress: a813e86dcc01f4a69bac3b62751a197e-550556558.us-east-1.elb.amazonaws.com
Port:          service  80/TCP
TargetPort:     3000/TCP
NodePort:       service  31271/TCP
Endpoints:     192.168.0.61:3000
Session Affinity: None
External Traffic Policy: Cluster
Internal Traffic Policy: Cluster
Events:
  Type  Reason          Age   From           Message
  ----  ----          ----  ----           -----
  Normal  EnsuringLoadBalancer  4m46s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer  4m42s  service-controller  Ensured load balancer
ubuntu@ip-172-31-90-146:~$
```

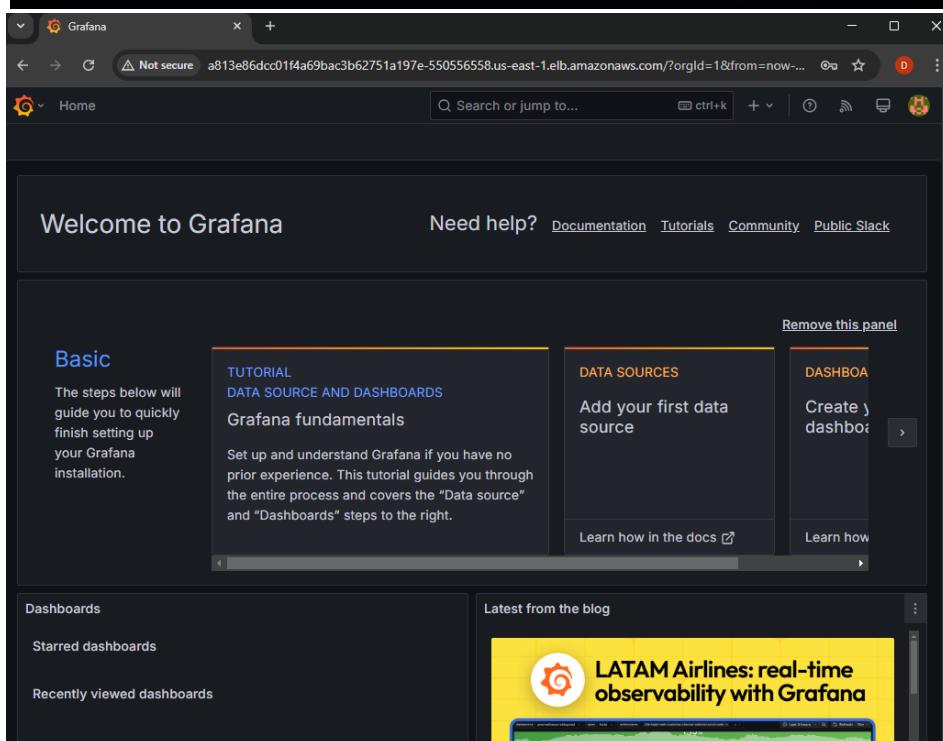
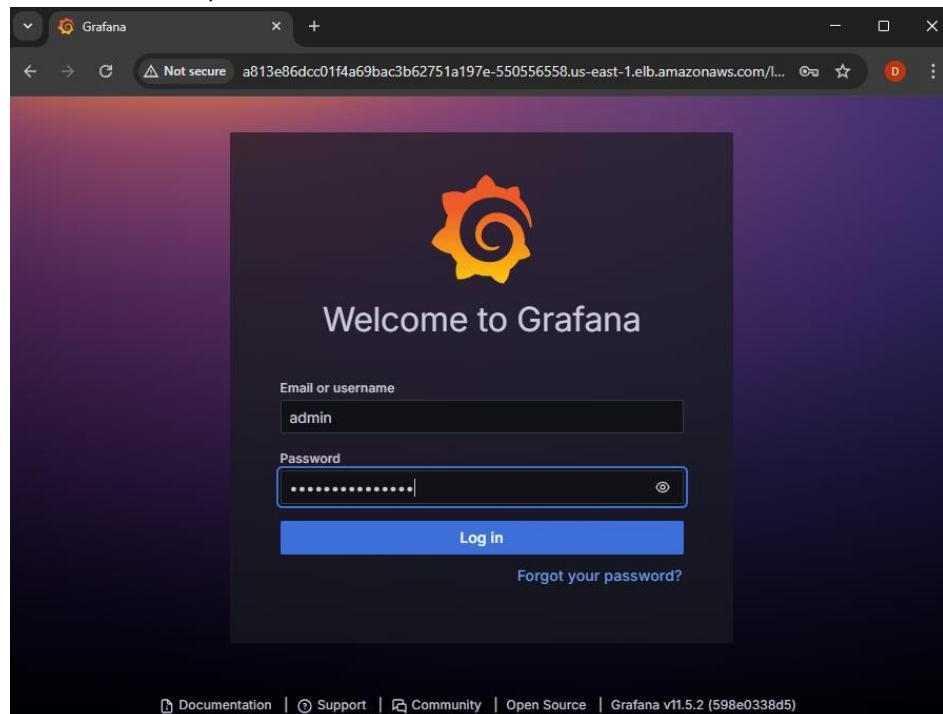
```
ubuntu@ip-172-31-90-146:~$ kubectl get svc --namespace grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0]}'
>{"hostname": "aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com"}ubuntu@ip-172-31-90-146:~$
```



Web

Acceso

Accedemos a la url del ELB Ingress (<http://a813e86dcc01f4a69bac3b62751a197e-550556558.us-east-1.elb.amazonaws.com/>)





Datasource:

Verificamos que el datasource de prometheus esté correctamente configurado y funcionando

The screenshot shows the Grafana interface with the 'Data sources' tab selected in the sidebar. In the main panel, there is a configuration card for a Prometheus data source named 'prometheus'. The URL field contains 'http://prometheus-server.prometheus.svc.cluster.local'. Both the 'Name' field and the URL field are highlighted with red boxes.

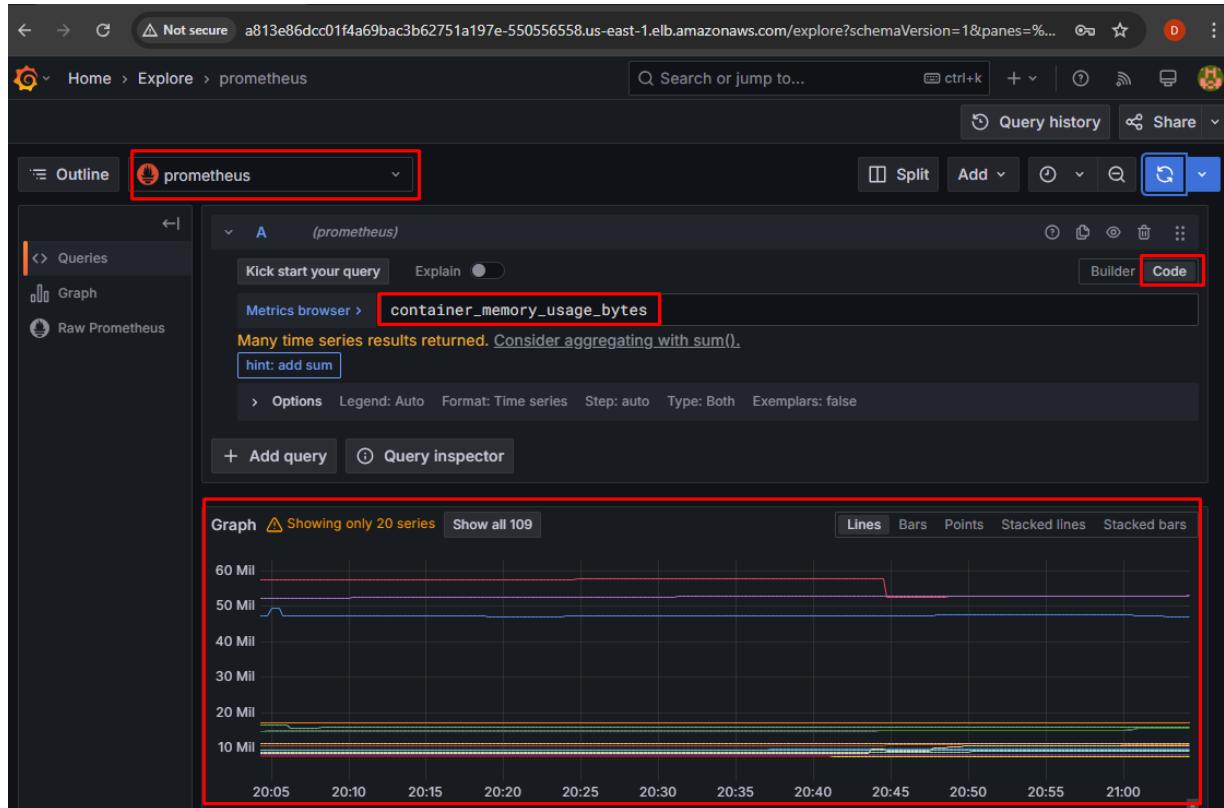
Al presionar el botón "Save & test" vemos que pasa las validaciones y puede consumir correctamente la API de Prometheus

The screenshot shows the Grafana interface after saving and testing the Prometheus data source. A green success message box is displayed, stating 'Successfully queried the Prometheus API.' Below the message, there is a note: 'Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#)'. At the bottom of the screen, there are two buttons: 'Delete' and 'Save & test', with 'Save & test' being the active button.

Validamos que podemos visualizar métricas de prometheus desde la sección "Explore".

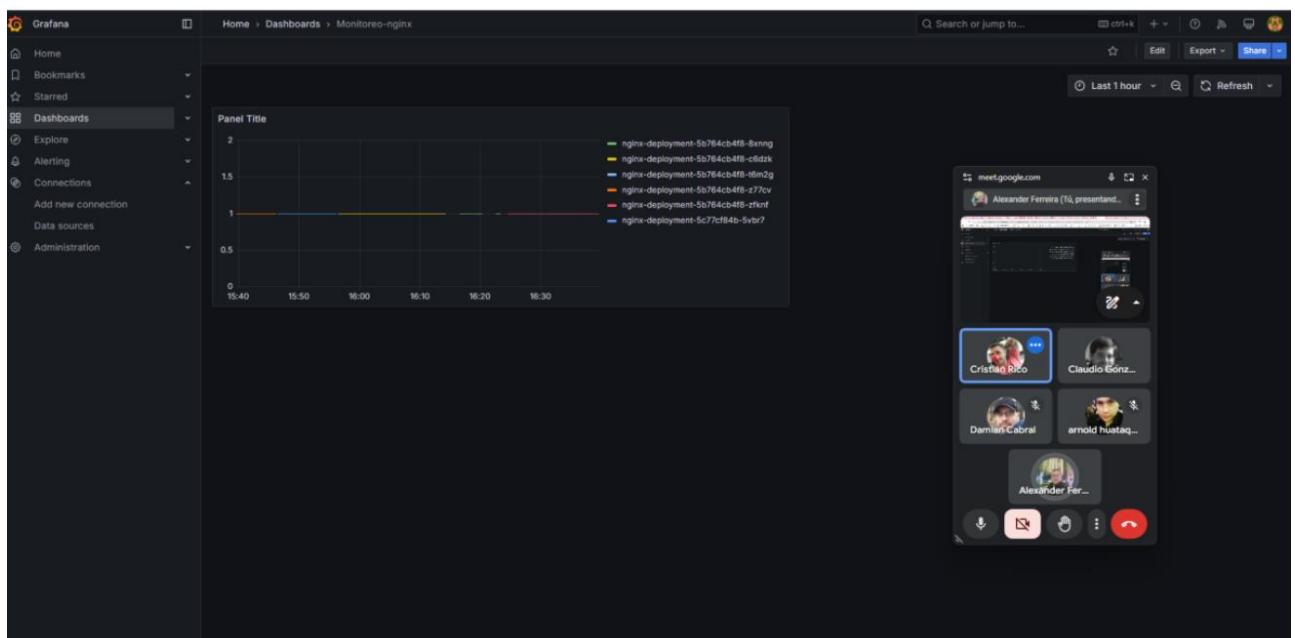


## Métricas de Prometheus



## Dashboards

**Creación de dashboard:** Se verifica que se pueden crear dashboards creando una métrica para el nginx desplegado anteriormente.





**Importar dashboard:** Vamos a Dashboards -> New -> Import

The screenshot shows the Grafana interface. On the left, there's a sidebar with links like Home, Bookmarks, Starred, and Dashboards (which is selected and highlighted with a red box). In the main area, it says 'Dashboards' and 'Create and manage dashboards to visualize your data'. There's a search bar, a 'Filter by tag' dropdown, and a 'Starred' checkbox. On the right, there are buttons for 'New', 'New dashboard', 'New folder', and 'Import' (which is also highlighted with a red box).

Colocamos el id del dashboard (3119)

The screenshot shows the 'Import dashboard' page. The left sidebar includes 'Dashboards' (selected), 'Playlists', 'Snapshots', 'Library panels', 'Shared dashboards', 'Explore', 'Alerting', 'Connections', and 'Administration'. The main area is titled 'Import dashboard' and says 'Import dashboard from file or Grafana.com'. It features a dashed box for 'Upload dashboard JSON file' with the text 'Drag and drop here or click to browse' and 'Accepted file types: .json, .txt'. Below this is a text input field containing '3119' (which is highlighted with a red box) and a 'Load' button. At the bottom, there's another section for 'Import via dashboard JSON model' with a text input field containing '{'.

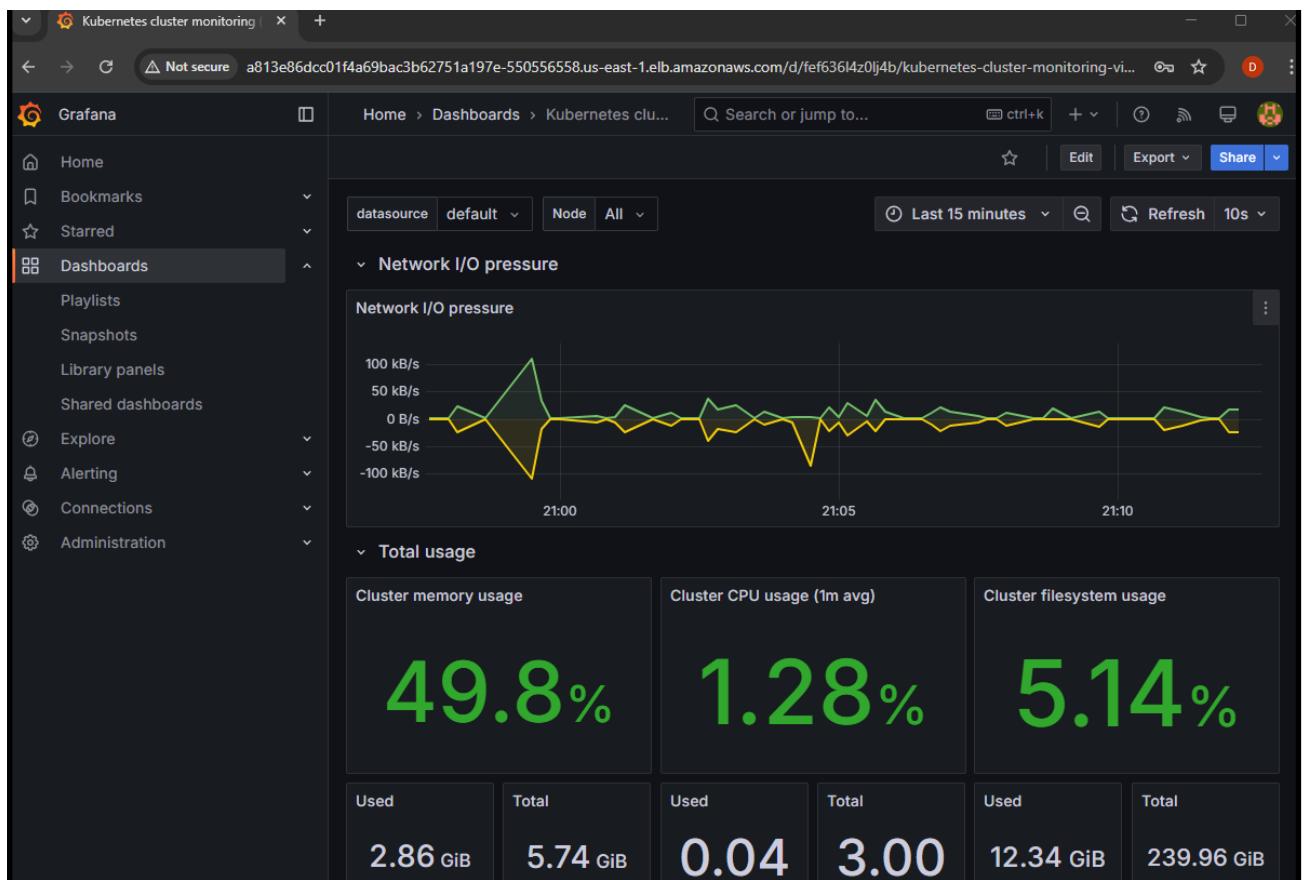


Seleccionamos el datasource del prometheus

The screenshot shows the Grafana interface for importing a dashboard. The left sidebar has 'Dashboards' selected. The main area displays a dashboard from 'Grafana.com' with the title 'Import dashboard'. It shows details like 'Published by' (Jjo Org) and 'Updated on' (2017-09-08 12:22:08). Under 'Options', the 'Name' field is set to 'Kubernetes cluster monitoring (via Prometheus)' and the 'Folder' is set to 'Dashboards'. A dropdown for 'Unique identifier (UID)' contains 'prometheus' and its icon. At the bottom, there are 'Import' and 'Cancel' buttons, with 'Import' being highlighted with a red box.



Observamos que el dashboard se importó correctamente y muestra métricas del cluster correctamente.



## CLEAN

Para eliminar los recursos debemos realizarlo de forma jerárquica ejecutando los diferentes scripts creados en el siguiente orden:

```
./grafana_install.sh -d  
./prometheus_install.sh -d  
./nginx.sh -d  
./00-crear_cluster -d
```

Eliminar entorno grafana

```
ubuntu@ip-172-31-90-146:~$ ./grafana_install.sh -d  
Modo delete activado: eliminando la instalación de Grafana...  
release "grafana" uninstalled  
release "grafana" uninstalled  
namespace "grafana" deleted  
namespace "grafana" deleted  
El entorno de Grafana ha sido eliminado.
```

Eliminar entorno prometheus

```
ubuntu@ip-172-31-90-146:~$ ./prometheus_install.sh -d  
Modo delete activado: eliminando la instalación de Prometheus...  
Modo delete activado: eliminando la instalación de Prometheus...  
release "prometheus" uninstalled  
namespace "prometheus" deleted  
namespace "prometheus" deleted  
El entorno de Prometheus ha sido eliminado.
```

Eliminar entorno de nginx

```
ubuntu@ip-172-31-90-146:~$ ./nginx.sh -d  
Modo delete activado: eliminando el Pod y el Service de Nginx en el namespace 'nginx'...  
deployment.apps "nginx-deployment" deleted  
service "nginx-service" deleted  
namespace "nginx" deleted  
Recursos eliminados.
```

## Eliminar cluster de EKS

```
ubuntu@ip-172-31-90-146:~$ ./crear_cluster.sh -d
Modo delete activado: eliminando el clúster '2403-g1-pin-final' en la región 'us-east-1'...
Modo delete activado: eliminando el clúster '2403-g1-pin-final' en la región 'us-east-1'...
2025-03-08 00:25:27 [i] deleting EKS cluster "2403-g1-pin-final"
2025-03-08 00:25:27 [i] will drain 0 unmanaged nodegroup(s) in cluster "2403-g1-pin-final"
2025-03-08 00:25:27 [i] starting parallel draining, max in-flight of 1
2025-03-08 00:25:27 [i] deleted 0 Fargate profile(s)
2025-03-08 00:25:28 [✓] kubeconfig has been updated
2025-03-08 00:25:28 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2025-03-08 00:26:04 [i]
4 sequential tasks: { delete nodegroup "ng-c0ddb93c",
  2 sequential sub-tasks: {
    2 sequential sub-tasks: {
      delete IAM role for serviceaccount "kube-system/ebs-csi-controller-sa",
      delete serviceaccount "kube-system/ebs-csi-controller-sa",
    },
    delete IAM OIDC provider,
  }, delete addon IAM "eksctl-2403-g1-pin-final-addon-vpc-cni", delete cluster control plane "2403-g1-pin-final" [async]
}
2025-03-08 00:26:05 [i] will delete stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:26:05 [i] waiting for stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c" to get deleted
2025-03-08 00:26:05 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:26:35 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:27:06 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:28:01 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:29:41 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:30:19 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:31:37 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:32:43 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:33:32 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:34:47 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-c0ddb93c"
2025-03-08 00:34:47 [i] will delete stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-08 00:34:47 [i] waiting for stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa" to get deleted
2025-03-08 00:34:47 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-08 00:35:17 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-08 00:35:17 [i] serviceaccount "kube-system/ebs-csi-controller-sa" was not created by eksctl; will not be deleted
2025-03-08 00:35:17 [i] will delete stack "eksctl-2403-g1-pin-final-addon-vpc-cni"
2025-03-08 00:35:18 [i] will delete stack "eksctl-2403-g1-pin-final-cluster"
2025-03-08 00:35:18 [✓] all cluster resources were deleted
Verificando la eliminación del clúster...

An error occurred (ValidationError) when calling the DescribeStacks operation: Stack with id eksctl-2403-g1-pin-final does not exist
El clúster '2403-g1-pin-final' ha sido eliminado exitosamente.
ubuntu@ip-172-31-90-146:~$ []
```

<b>devops 2403</b>	<b>Grupo 1</b> <b>PIN Final</b>	<b>mundosE</b>
--------------------	------------------------------------	----------------

# REVISIÓN

## Conclusiones

Se logró desplegar completamente la infraestructura en AWS con terraform, eksctl, kubectl y helm, optimizando despliegues y reduciendo errores manuales. La configuración correcta de permisos, volúmenes y puertos fue clave para garantizar el funcionamiento estable de los servicios.

El monitoreo en tiempo real con Prometheus y Grafana permitió la observabilidad del clúster, configurando Exporters y Service Monitors para recolectar métricas críticas. Como también la importación de dashboards desde <https://grafana.com/grafana/dashboards/>

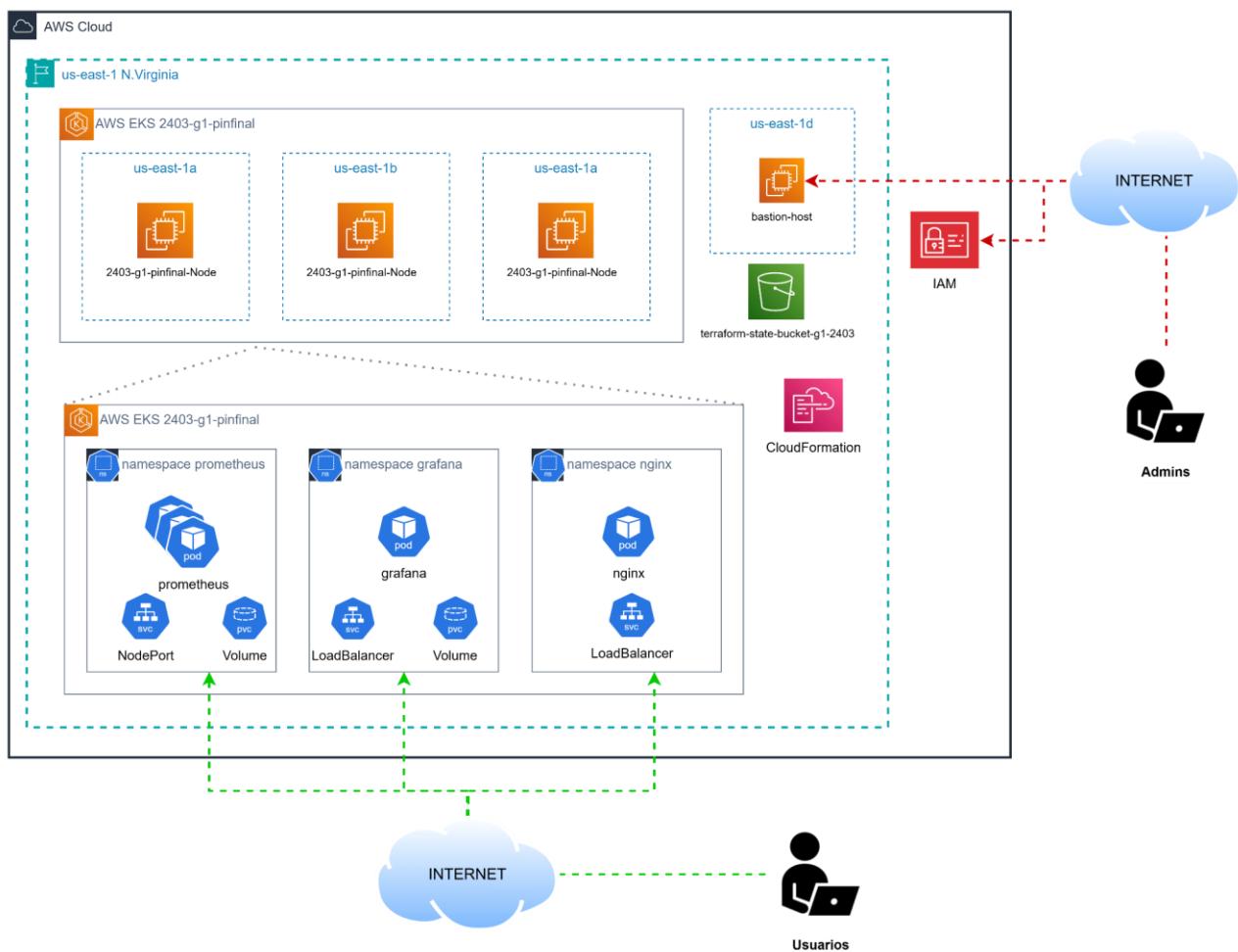
Se utilizaron servicios de tipo ClusterIP, NodePort y LoadBalancer para exponer servicios privados y públicos, como es el caso de Nginx, asegurando accesibilidad y escalabilidad.

Se resolvieron desafíos en almacenamiento, compatibilidad de versiones y permisos mediante validaciones de storageClass e instalación y actualización de addons.

Se implementó una estrategia ordenada para la eliminación de recursos, evitando bloqueos en AWS.

La modularidad del proyecto permite escalar y agregar nuevos servicios sin afectar la estabilidad. En conclusión, esta infraestructura automatizada y monitoreada con Kubernetes y AWS demuestra buenas prácticas de DevOps, garantizando eficiencia, seguridad y escalabilidad.

## Topología general



devops 2403

Grupo 1

PIN Final

mundosE

## Consumo

Verificación de los costos consumidos.

The screenshot shows the AWS Billing and Cost Management home page. On the left, there's a sidebar with navigation links for Home, Billing and Payments, Cost and Usage Analysis, and Cost Organization. The main content area displays the Cost summary and Cost monitor sections. In the Cost summary section, it shows a month-to-date cost of \$20.13 compared to last month for the same period. Total forecasted cost for the current month is listed as Data unavailable. Last month's total cost is \$0.00. The Cost monitor section shows budgets status as OK with 1 active budget(s), and cost anomalies status (MTD) as None detected with 1 monitor(s) active.

Billing and Cost Management | us-east-1.console.aws.amazon.com/costmanagement/home?region=us-east-1#/home

Billing and Cost Management home

Provide feedback Contact AWS support Reset layout

Cost summary

Month-to-date cost  
\$20.13

- compared to last month for same period

Last month's cost for same time period  
\$0.00

Feb 1 – 10

Total forecasted cost for current month  
Data unavailable

Last month's total cost  
\$0.00

View bill

Cost monitor

Budgets status  
OK

1 active budget(s)

Cost anomalies status (MTD)  
None detected

1 monitor(s) active