

# 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>mundosE</b>
	<b>PIN Final</b>	

<b>INTRODUCCIÓN</b>	<b>4</b>
<b>GITHUB</b>	<b>6</b>
COMMITTS	6
REPOSITORY SECRETS	7
<b>AWS</b>	<b>8</b>
Configuraciones generales	8
IAM	8
Billing and Cost Management	9
Estructura de directorios	9
EC2 (Bastion)	11
KEY PAIR	11
OTROS ARCHIVOS	12
user_data.sh	12
ec2-admin.json	14
.gitignore	18
TERRAFORM	19
main.tf	19
backend.tf	19
iam.tf	20
variables.tf	20
outputs.tf	21
providers.tf	21
terraform.tfvars	21
GITHUB ACTIONS	22
Workflow	22
Ejecución workflow	23
AWS (EC2)	25
Validación de creación de instancia	25
Rol ec2-admin asignado	26
Acceso a instancia	27
Ubuntu Version	28
Paquetes	28
Elastic IP	28
EKS	30
DESPLIEGUE	30
Script crear_cluster.sh	30
Preparación	32
Ejecución de script y verificaciones	33
Permisos	36

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

Consola Web	36
configMap	37
NGINX	41
Despliegue	41
Comprobaciones:	43
EBS CSI Driver	44
Instalación	44
Crear addon EBS	45
Validar funcionamiento	47
Error de permisos al crear el pvc:	49
ServiceAccount del EBS CSI Driver:	51
Verificación	54
MONITOREO	56
PROMETHEUS	56
Instalación	56
Configurar NodePort	58
Validación:	58
Troubleshooting pod alertmanager	60
Port forward access:	62
Node port access:	65
GRAFANA	66
Instalación	66
Verificación:	68
EKS	68
Web	69
Dashboard	72
CLEAN	75
REVISIÓN	78
Conclusiones	78
Topología general	79
Consumo	80

<b>devops 2403</b>	<b>Grupo 1</b>	<b>mundosE</b>
	<b>PIN Final</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 Bastion Host donde se instalará y utilizarán herramientas de gestión como AWS CLI, kubect, eksctl, Docker, Helm.
- GitHub: Repositorio de código y config files para versionado y despliegue mediante workflows con Github Actions.
- Prometheus: Herramienta para la recolección de métricas del cluster de Kubernetes.  
URL Interna: <http://prometheus.monitoreo.svc.cluster.local:8080>

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

URL Externa: <http://18.212.133.56:32000>

- Grafana: Plataforma para la visualización de las métricas recolectadas por Prometheus.  
Importacion de Dashboard ID: 3119  
URL Externa: <http://aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com/>

devops 2403	Grupo 1	mundosE
	PIN Final	

## GITHUB

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

```
# Creacion de Rama principal
git checkout -b main
git add .
git commit -m "Inicial commit en main"
git push -u origin main
```

```
# Creacion de Rama de desarrollo
git checkout -b dev
```

## COMMITTS

```
# Actualizar rama dev
git status
git checkout dev
git add .
git commit -m "Update [skip ci]"
git push origin dev
```

```
# Merge de rama dev a main
git checkout main
git merge dev
git push origin main
```

En caso de no querer ejecutar el workflow de github actions, dentro del mensaje del commit agregar al final el texto "[skip ci]"

devops 2403	Grupo 1	mundosE
	PIN Final	

## REPOSITORY SECRETS

Se configura el Access Key y Secret Key del usuario de servicio terraform como Repository Secrets

Repository secret added.

**Actions secrets and variables**

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

**Environment secrets**

This environment has no secrets.

[Manage environment secrets](#)

**Repository secrets** [New repository secret](#)

Name ↕	Last updated
AWS_ACCESS_KEY_ID	now
AWS_SECRET_ACCESS_KEY	now

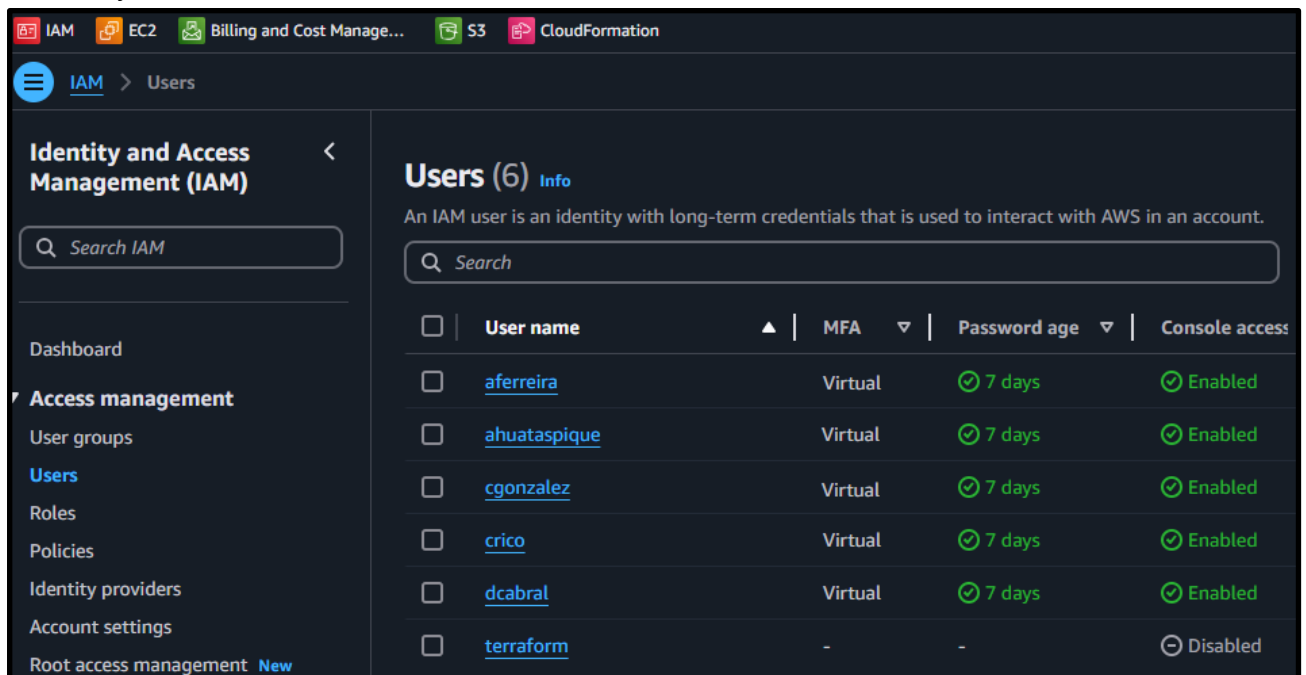
devops 2403	Grupo 1	mundosE
	PIN Final	

## AWS

### Configuraciones generales

#### IAM

Se crean usuarios nominales para el acceso a la consola para auditoría. Los mismos también fueron agregados a un grupo de usuarios. Asimismo, se crea el usuario terraform sin acceso a la consola y con claves ak/sk

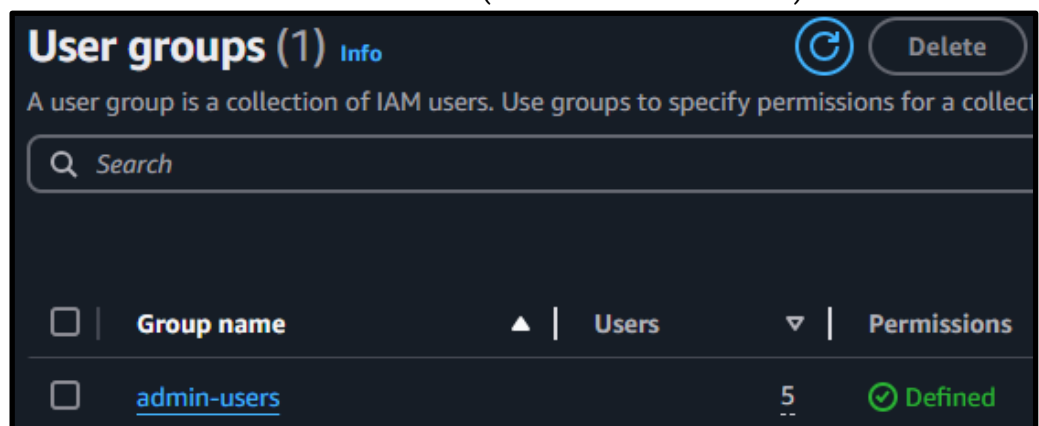


**Users (6)** Info

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

<input type="checkbox"/>	User name	MFA	Password age	Console access
<input type="checkbox"/>	<a href="#">aferreira</a>	Virtual	✓ 7 days	✓ Enabled
<input type="checkbox"/>	<a href="#">ahuataspique</a>	Virtual	✓ 7 days	✓ Enabled
<input type="checkbox"/>	<a href="#">cgonzalez</a>	Virtual	✓ 7 days	✓ Enabled
<input type="checkbox"/>	<a href="#">crico</a>	Virtual	✓ 7 days	✓ Enabled
<input type="checkbox"/>	<a href="#">dcabral</a>	Virtual	✓ 7 days	✓ Enabled
<input type="checkbox"/>	<a href="#">terraform</a>	-	-	⊘ Disabled

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



**User groups (1)** Info

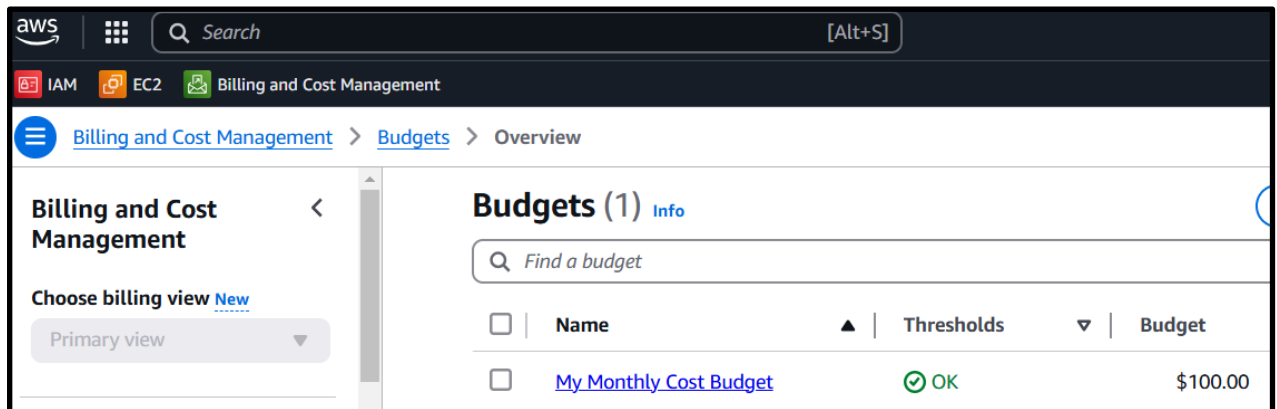
A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.

<input type="checkbox"/>	Group name	Users	Permissions
<input type="checkbox"/>	<a href="#">admin-users</a>	5	✓ Defined



devops 2403	Grupo 1	mundosE
	PIN Final	

## Billing and Cost Management



## 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, linea 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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

|           |─ terraform.tfvars  # Valores específicos
|           |─ scripts/
|           |   └─ user_data.sh # Script con las herramientas
|   └─ eks/
|       └─ 01-nginx/
|           └─ nginx_install.sh
|           └─ nginx-pod.yaml
|       └─ 02-ebs-driver-tests/
|           └─ 01storageclass.yaml
|           └─ 02pvc.yaml
|           └─ 03ebs-pod.yaml
|           └─ 04ebs-observation.sh
|           └─ 05authorization-in-ebs-csi-pod-to-nodegroup.sh
|           └─ 06authorization-in-ebs-sa.sh
|       └─ 03-prometheus/
|           └─ prometheus_install.sh
|       └─ 04-grafana/
|           └─ grafana.yaml
|           └─ grafana_install.sh
|       └─ crear_cluster.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	mundosE
	PIN Final	

## 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:B5olw4hx1cm0t803b/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]-----+
```

devops 2403	Grupo 1	mundosE
	PIN Final	

## OTROS ARCHIVOS

user\_data.sh

Archivo utilizado para la instalación de paquetes durante el proceso de despliegue e inicialización de la instancia.

```
#!/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/
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
# =====
# 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
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
# =====
# 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 la vm 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:CreateChangeSet"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:CreateNodegroup",
        "eks:UpdateNodegroupConfig",
        "eks>DeleteNodegroup",
        "eks:ListNodegroups",
        "eks:DescribeNodegroup",

        "eks:ListFargateProfiles",
        "eks:DescribeFargateProfile",

        "eks:CreateAddon",
        "eks:UpdateAddon",
        "eks>DeleteAddon",
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

    "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",

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

    "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",

```



devops 2403	Grupo 1	mundosE
	PIN Final	

```

    "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",

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

        "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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
terraform.tfstate
terraform.tfstate.backup
.terraform/
.vscode
```

## TERRAFORM

main.tf

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

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]

  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

Guardamos el .tfstate en un bucket de s3

devops 2403	Grupo 1	mundosE
	PIN Final	

```

terraform
  backend
    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

```

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 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

```

variable "vpc_id" {
  description = "ID de la VPC"
}

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

    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_id      = "vpc-04adbff65ec30ad98"
ami_id      = "ami-0e1bed4f06a3b463d"
key_name    = "bastion-key"
public_ssh_key = "ssh-rsa AAAAB3NzaC1....."

```

devops 2403	Grupo 1	mundosE
	PIN Final	

## GITHUB ACTIONS

### Workflow

.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:
    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
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

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

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

```

## Ejecución workflow

The screenshot shows the GitHub Actions interface for a workflow named 'mundose.pinfinal.grupo1'. The workflow is titled 'Terraform Deploy to AWS' and is currently running a job named 'Agregando file backend.tf #12'. The job is triggered by a push to the 'main' branch by user 'dec-wil' at 7:07 PM on February 4th. The job status is 'In progress'.

The workflow file 'deploy.yml' is shown, with the trigger 'on: push'. The workflow consists of two jobs: 'plan' (15s) and 'apply' (1m 11s).

The left sidebar shows the workflow file 'Workflow file' and the job 'plan' (15s) and 'apply' (1m 11s).

devops 2403	Grupo 1	mundosE
	PIN Final	

Se observa que el workflow se ejecuta sin errores.

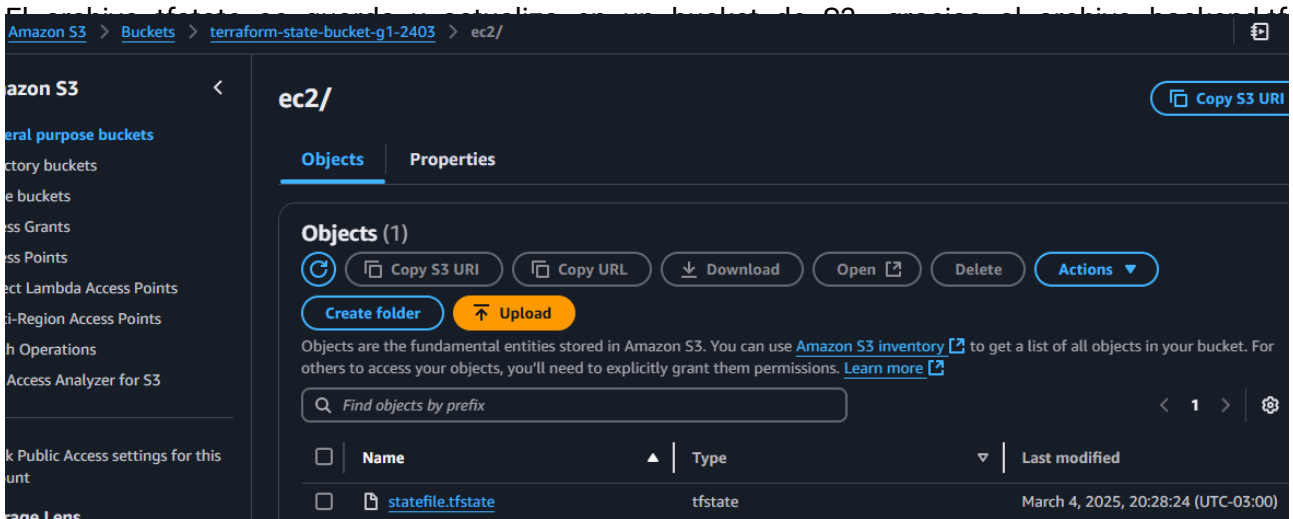
The screenshot shows the GitHub Actions interface for the workflow 'Terraform Deploy to AWS' in the repository 'dec-wil / mundose.pinfinal.grupo1'. The workflow is titled 'Agregando file backend.tf para guardado de statefile #19' and has a green checkmark indicating success. It was triggered via push 1 minute ago by user 'dec-wil' on the 'main' branch. The status is 'Success' with a total duration of '1m 13s'. The workflow consists of two jobs: 'plan' (17s) and 'apply' (38s), both marked as successful. The 'Summary' tab is selected, showing the workflow file 'deploy.yml' with the trigger 'on: push'.

Verificación y extracción de ip pública de la instancia:

The screenshot shows the 'apply' job output for the workflow 'Terraform Deploy to AWS'. The job is titled 'Aplicar cambios con Terraform (sin bloqueo)' and succeeded 10 minutes ago in 38s. The output shows the Terraform plan and apply process. The plan indicates 7 resources to be added, 0 to be changed, and 0 to be destroyed. The apply process shows the creation of various AWS resources, including 'aws\_key\_pair.pin', 'aws\_iam\_policy.ec2\_admin\_policy', 'aws\_iam\_role.ec2\_admin\_role', 'aws\_security\_group.bastion\_sg', 'aws\_iam\_instance\_profile.ec2\_admin\_profile', and 'aws\_instance.bastion'. The final output shows the public IP address of the instance: 'bastion\_public\_ip = "54.224.53.78"'. The workflow file 'deploy.yml' is also visible in the left sidebar.



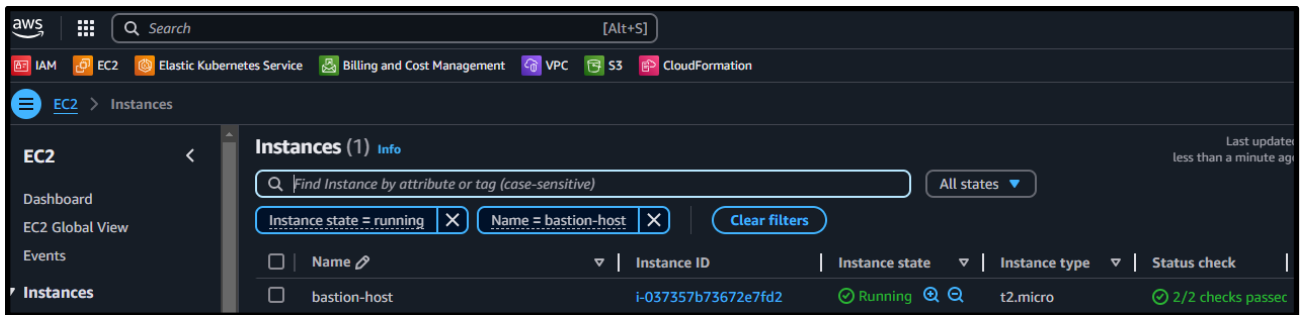
devops 2403	Grupo 1	mundosE
	PIN Final	



## AWS (EC2)

### Validación de creación de instancia

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



devops 2403	Grupo 1	mundosE
	PIN Final	

Rol ec2-admin asignado

The screenshot displays the AWS Management Console interface for an EC2 instance. The left sidebar shows the navigation menu with categories like EC2, Images, Elastic Block Store, and Network & Security. The main content area shows the 'Instance summary' for the instance ID 'i-037357b73672e7fd2' (named 'bastion-host'). The instance is in a 'Running' state. Key details include the Public IPv4 address (3.95.154.227), Private IPv4 address (172.31.90.146), Instance type (t2.micro), and IAM Role (ec2-admin). A warning message indicates that EC2 recommends setting IMDSv2 to required.

Instance summary for i-037357b73672e7fd2 (bastion-host)		
<b>Instance ID</b> i-037357b73672e7fd2	<b>Public IPv4 address</b> 3.95.154.227   <a href="#">open address</a>	<b>Private IPv4 addresses</b> 172.31.90.146
<b>IPv6 address</b> -	<b>Instance state</b> Running	<b>Public IPv4 DNS</b> ec2-3-95-154-227.compute-1.amazonaws.com   <a href="#">open address</a>
<b>Hostname type</b> IP name: ip-172-31-90-146.ec2.internal	<b>Private IP DNS name (IPv4 only)</b> ip-172-31-90-146.ec2.internal	<b>Elastic IP addresses</b> -
<b>Answer private resource DNS name</b> -	<b>Instance type</b> t2.micro	<b>AWS Compute Optimizer finding</b> Opt-in to AWS Compute Optimizer for recommendations.   <a href="#">Learn more</a>
<b>Auto-assigned IP address</b> 3.95.154.227 [Public IP]	<b>VPC ID</b> vpc-04adbff65ec30ad98	<b>Auto Scaling Group name</b> -
<b>IAM Role</b> ec2-admin	<b>Subnet ID</b> subnet-09b05791e36ae2f33	<b>Managed</b> false
<b>IMDSv2</b> Optional ⚠ EC2 recommends setting IMDSv2 to required   <a href="#">Learn more</a>	<b>Instance ARN</b> arn:aws:ec2:us-east-1:194722402815:instance/i-037357b73672e7fd2	
<b>Operator</b> -		

devops 2403	Grupo 1	mundosE
	PIN Final	

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@3.95.154.227
```

```
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.
```

devops 2403	Grupo 1	mundosE
	PIN Final	

## 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.

The screenshot shows the AWS Management Console interface for associating an Elastic IP address. The breadcrumb navigation indicates the path: EC2 > Elastic IP addresses > Associate Elastic IP address. The main heading is 'Associate Elastic IP address' with an information icon. Below the heading, a message states: 'Choose the instance or network interface to associate to this Elastic IP address (44.193.247.235)'. The 'Elastic IP address: 44.193.247.235' is displayed. Under 'Resource type', the 'Instance' radio button is selected. A warning box notes: 'If you associate an Elastic IP address with an instance that already has an Elastic IP address associated, the previous Elastic IP address will be disassociated. If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.' The 'Instance' field contains the ID 'i-037357b73672e7fd2'. The 'Private IP address' field contains '172.31.90.146'. At the bottom, the 'Reassociation' section has an unchecked checkbox for 'Allow this Elastic IP address to be reassociated'.

Observamos que la ip fue asignada correctamente.

devops 2403	Grupo 1	mundosE
	PIN Final	

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-037357b73672e7fd2

Search [Alt+S] United States (N. Virginia) dcabral @ 1947-2240-2

EC2 > Instances > i-037357b73672e7fd2

### EC2

- Dashboard
- EC2 Global View
- Events
- Instances
  - Instances
  - Instance Types
  - Launch Templates
  - Spot Requests
  - Savings Plans
  - Reserved Instances
  - Dedicated Hosts
  - Capacity Reservations
- Images
  - AMIs
  - AMI Catalog
- Elastic Block Store
  - Volumes
  - Snapshots
  - Lifecycle Manager
- Network & Security
  - Security Groups

### Instance summary for i-037357b73672e7fd2 (bastion-host)

Updated less than a minute ago

**Instance ID**  
i-037357b73672e7fd2

**Public IPv4 address**  
44.193.247.235 | [open address](#)

**Private IPv4 addresses**  
172.31.90.146

**Public IPv4 DNS**  
ec2-44-193-247-235.compute-1.amazonaws.com | [open address](#)

**Instance state**  
Running

**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

**Hostname type**  
IP name: ip-172-31-90-146.ec2.internal

**Answer private resource DNS name**  
-

**Auto-assigned IP address**  
-

**IAM Role**  
ec2-admin

**IMDSv2**  
Optional  
⚠ EC2 recommends setting IMDSv2 to required | [Learn more](#)

**Elastic IP addresses**  
44.193.247.235 [Public IP]

**AWS Compute Optimizer finding**  
ⓘ Opt-in to AWS Compute Optimizer for recommendations. | [Learn more](#)

**Auto Scaling Group name**  
-

**Managed**  
false

devops 2403	Grupo 1	mundosE
	PIN Final	

## 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

# =====
# Configuración de variables
# =====
# Define el nombre del clúster que se creará.
CLUSTER_NAME="2403-g1-pin-final"
# Define la región de AWS donde se creará 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"
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
#eksctl delete cluster --region=us-east-1 --name=2403-g1-pin-final
#aws cloudformation delete-stack --stack-name eksctl-2403-g1-pin-final --region us-east-1
#aws cloudformation describe-stacks --stack-name eksctl-2403-g1-pin-final --region us-east-1

# =====
# Funciones de utilidad
# =====
# Función para comprobar si un comando existe en el sistema.
# Se utiliza 'command -v' para buscar la ruta del comando; si no se encuentra, retorna un error.
command_exists() {
    command -v "$1" >/dev/null 2>&1
}

# =====
# Verificaciones previas
# =====
# Verificar si el AWS CLI está instalado. Si no, muestra un mensaje de error y termina el script.
if ! command_exists aws; then
    echo "Error: aws CLI no está instalado. Por favor, instálalo antes de continuar." >&2
    exit 1
fi

# Verificar si eksctl está instalado. Si no, muestra un mensaje de error y termina el script.
if ! command_exists eksctl; then
    echo "Error: eksctl no está instalado. Por favor, instálalo antes de continuar." >&2
    exit 1
fi

# Verificar que las credenciales de AWS estén configuradas correctamente usando 'aws sts get-caller-identity'.
# Si la comprobación falla, se solicita al usuario que ejecute 'aws configure'.
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

# Mensaje informativo para indicar que las credenciales han sido verificadas y se procederá a la creación del clúster.
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.
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
# --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-ecr-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 "$SSH_KEY" \
    --managed \
    --full-ecr-access \
    --zones "$ZONES"; then
    # Si el comando se ejecuta correctamente, se muestra un mensaje de éxito.
    echo "Configuración del clúster completada con éxito mediante eksctl."
else
    # Si ocurre algún error durante la creación, se muestra un mensaje de error y se termina el script.
    echo "La configuración del clúster falló durante la ejecución de eksctl." >&2
    exit 1
fi
```

## Preparación

Subo el script al Bastion Host mediante scp, accedemos al mismo y asignamos permisos de ejecución al file.

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



devops 2403	Grupo 1	mundosE
	PIN Final	

```
ubuntu@ip-172-31-90-146:~$ ls -lah
total 36K
drwxr-x--- 4 ubuntu ubuntu 4.0K Mar  2 13:47 .
drwxr-xr-x 3 root    root    4.0K Mar  1 22:18 ..
-rw----- 1 ubuntu ubuntu 174 Mar  2 13:44 .bash_history
-rw-r--r-- 1 ubuntu ubuntu 220 Jan  6 2022 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3.7K Jan  6 2022 .bashrc
drwx----- 2 ubuntu ubuntu 4.0K Mar  1 22:22 .cache
-rw-r--r-- 1 ubuntu ubuntu 807 Jan  6 2022 .profile
drwx----- 2 ubuntu ubuntu 4.0K Mar  1 22:18 .ssh
-rw-rw-r-- 1 ubuntu ubuntu 3.7K Mar  2 13:48 crear_cluster.sh
ubuntu@ip-172-31-90-146:~$ chmod +x crear_cluster.sh && ls -lah crear_cluster.sh
-rwxrwxr-x 1 ubuntu ubuntu 3.7K Mar  2 13:48 crear_cluster.sh
ubuntu@ip-172-31-90-146:~$
```

Ejecución de script y verificaciones

Logs resultantes del script de despliegue del cluster

```
ubuntu@ip-172-31-90-146:~$ ./crear_cluster.sh
Credenciales verificadas. Procediendo con la creación del clúster '2403-g1-pin-final' en la región 'us-east-1'.
2025-03-03 17:39:04 [i] eksctl version 0.205.0
2025-03-03 17:39:04 [i] using region us-east-1
2025-03-03 17:39:04 [i] subnets for us-east-1a - public:192.168.0.0/19 private:192.168.96.0/19
2025-03-03 17:39:04 [i] subnets for us-east-1b - public:192.168.32.0/19 private:192.168.128.0/19
2025-03-03 17:39:04 [i] subnets for us-east-1c - public:192.168.64.0/19 private:192.168.160.0/19
2025-03-03 17:39:04 [i] nodegroup "ng-67b43adb" will use "" [AmazonLinux2/1.32]
2025-03-03 17:39:04 [i] using EC2 key pair "pin"
2025-03-03 17:39:04 [i] using Kubernetes version 1.32
2025-03-03 17:39:04 [i] creating EKS cluster "2403-g1-pin-final" in "us-east-1" region with managed nodes
2025-03-03 17:39:04 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-03-03 17:39:04 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=2403-g1-pin-final'
2025-03-03 17:39:04 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "2403-g1-pin-final" in "us-east-1"
2025-03-03 17:39:04 [i] CloudWatch logging will not be enabled for cluster "2403-g1-pin-final" in "us-east-1"
2025-03-03 17:39:04 [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'
2025-03-03 17:39:04 [i] default addons coredns, metrics-server, vpc-cni, kube-proxy were not specified, will install them as EKS addons
2025-03-03 17:39:04 [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 update VPC CNI to use IRSA if required,
},
create managed nodegroup "ng-67b43adb",
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

}
}
2025-03-03 17:39:04 [i] building cluster stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:39:04 [i] deploying stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:39:34 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:40:04 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:41:04 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:42:04 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:43:04 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:44:05 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:45:05 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:46:05 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:47:05 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-cluster"
2025-03-03 17:47:06 [i] creating addon: coredns
2025-03-03 17:47:06 [i] successfully created addon: coredns
2025-03-03 17:47:06 [i] creating addon: metrics-server
2025-03-03 17:47:07 [i] successfully created addon: metrics-server
2025-03-03 17:47:07 [!] 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`
2025-03-03 17:47:07 [i] creating addon: vpc-cni
2025-03-03 17:47:08 [i] successfully created addon: vpc-cni
2025-03-03 17:47:08 [i] creating addon: kube-proxy
2025-03-03 17:47:08 [i] successfully created addon: kube-proxy
2025-03-03 17:49:09 [i] addon "vpc-cni" active
2025-03-03 17:49:09 [i] deploying stack "eksctl-2403-g1-pin-final-addon-vpc-cni"
2025-03-03 17:49:10 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-vpc-cni"
2025-03-03 17:49:40 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-vpc-cni"
2025-03-03 17:49:40 [i] updating addon
2025-03-03 17:49:50 [i] addon "vpc-cni" active
2025-03-03 17:49:50 [i] building managed nodegroup stack "eksctl-2403-g1-pin-final-nodegroup-ng-67b43adb"
2025-03-03 17:49:50 [i] deploying stack "eksctl-2403-g1-pin-final-nodegroup-ng-67b43adb"
2025-03-03 17:49:51 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-67b43adb"
2025-03-03 17:50:21 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-67b43adb"
2025-03-03 17:51:16 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-67b43adb"
2025-03-03 17:52:15 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-nodegroup-ng-67b43adb"
2025-03-03 17:52:15 [i] waiting for the control plane to become ready
2025-03-03 17:52:16 [✓] saved kubeconfig as "/home/ubuntu/.kube/config"
2025-03-03 17:52:16 [✓] saved kubeconfig as "/home/ubuntu/.kube/config"
2025-03-03 17:52:16 [i] no tasks
2025-03-03 17:52:16 [✓] all EKS cluster resources for "2403-g1-pin-final" have been created
2025-03-03 17:52:16 [i] nodegroup "ng-67b43adb" has 3 node(s)
2025-03-03 17:52:16 [i] node "ip-192-168-16-253.ec2.internal" is ready
2025-03-03 17:52:16 [i] node "ip-192-168-39-20.ec2.internal" is ready
2025-03-03 17:52:16 [i] node "ip-192-168-95-119.ec2.internal" is ready
2025-03-03 17:52:16 [i] no tasks
2025-03-03 17:52:16 [✓] all EKS cluster resources for "2403-g1-pin-final" have been created
2025-03-03 17:52:16 [i] nodegroup "ng-67b43adb" has 3 node(s)
2025-03-03 17:52:16 [i] node "ip-192-168-16-253.ec2.internal" is ready
2025-03-03 17:52:16 [i] node "ip-192-168-39-20.ec2.internal" is ready
2025-03-03 17:52:16 [i] node "ip-192-168-95-119.ec2.internal" is ready
2025-03-03 17:52:16 [i] waiting for at least 3 node(s) to become ready in "ng-67b43adb"
2025-03-03 17:52:16 [i] nodegroup "ng-67b43adb" has 3 node(s)
2025-03-03 17:52:16 [i] node "ip-192-168-16-253.ec2.internal" is ready
2025-03-03 17:52:16 [i] waiting for at least 3 node(s) to become ready in "ng-67b43adb"
2025-03-03 17:52:16 [i] nodegroup "ng-67b43adb" has 3 node(s)

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

2025-03-03 17:52:16 [i] node "ip-192-168-16-253.ec2.internal" is ready
2025-03-03 17:52:16 [i] node "ip-192-168-39-20.ec2.internal" is ready
2025-03-03 17:52:16 [i] node "ip-192-168-39-20.ec2.internal" is ready
2025-03-03 17:52:16 [i] node "ip-192-168-95-119.ec2.internal" is ready
2025-03-03 17:52:16 [✓] created 1 managed nodegroup(s) in cluster "2403-g1-pin-final"
2025-03-03 17:52:16 [i] node "ip-192-168-95-119.ec2.internal" is ready
2025-03-03 17:52:16 [✓] created 1 managed nodegroup(s) in cluster "2403-g1-pin-final"
2025-03-03 17:52:17 [i] kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-03-03 17:52:17 [i] kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-03-03 17:52:17 [✓] 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-192-168-16-253.ec2.internal      Ready    <none>    90m   v1.32.1-eks-5d632ec
ip-192-168-39-20.ec2.internal       Ready    <none>    90m   v1.32.1-eks-5d632ec
ip-192-168-95-119.ec2.internal      Ready    <none>    90m   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-ctcf8                      2/2      Running   0           111m
aws-node-ldlgz                      2/2      Running   0           111m
aws-node-rmk5t                      2/2      Running   0           111m
coredns-6b9575c64c-hh9j9           1/1      Running   0           115m
coredns-6b9575c64c-ltzl9           1/1      Running   0           115m
kube-proxy-bcx8n                    1/1      Running   0           111m
kube-proxy-w9rgt                    1/1      Running   0           111m
kube-proxy-wf67w                    1/1      Running   0           111m
metrics-server-57b774cc8d-n58mw     1/1      Running   0           115m
metrics-server-57b774cc8d-rgzfm     1/1      Running   0           115m

```

Verificando OIDC

A raíz del warning que muestra el output del script verifico 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/89CE59E8EB211D52275F5FD0E36229F4
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" value="Find Instance by attribute or tag (case-sensitive)"/>				All states ▾
<input type="checkbox"/>	Name <a href="#">↗</a>	Instance ID	Instance state <a href="#">↕</a>	Instance type
<input type="checkbox"/>	2403-g1-pin-final-ng-67b43adb-Node	i-083437b753804dcf7	Running <a href="#">🔍</a> <a href="#">🔍</a>	t2.small
<input type="checkbox"/>	2403-g1-pin-final-ng-67b43adb-Node	i-0e62e6b54a022dde1	Running <a href="#">🔍</a> <a href="#">🔍</a>	t2.small
<input type="checkbox"/>	2403-g1-pin-final-ng-67b43adb-Node	i-01e3b193e2a991cf9	Running <a href="#">🔍</a> <a href="#">🔍</a>	t2.small
<input type="checkbox"/>	bastion-host	i-037357b73672e7fd2	Running <a href="#">🔍</a> <a href="#">🔍</a>	t2.micro

devops 2403	Grupo 1	mundosE
	PIN Final	

## Permisos

Cuando se despliega un nuevo clúster EKS, los usuarios de IAM por defecto no tienen permisos para administrar el clúster. 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.

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.

## Consola Web

Podemos también realizarlo desde la consola web donde, luego de desplegar el cluster, nos muestra una alerta en pantalla informando de esta situación.

A continuación se procede a asignar un usuario a modo de ejemplo (esto no sigue las buenas practicas de gestión)

**2403-g1-pin-final**

**Cluster info**

<b>Status</b> Active	<b>Kubernetes version</b> 1.32	<b>Support period</b> Standard support until March 21, 2026	<b>Provider</b> EKS
<b>Cluster health issues</b> 0	<b>Upgrade insights</b> 4	<b>Node health issues</b> 0	

### Configure IAM access entry

**IAM principal**

The IAM principal that you want to grant access to Kubernetes objects on your cluster.

**IAM principal ARN**

arn:aws:iam::194722402815:user/dcabral

**Type**

By selecting an option other than Standard, EKS automatically associates the permissions required for nodes using the IAM role for this access entry to

**Type**

Select the type of IAM access entry

Standard

devops 2403	Grupo 1	mundosE
	PIN Final	

**Add access policy - optional**

**Access policies** [Info](#)  
Select an access policy to associate to the access entry and the scope of the access policy.

**Policy name**  
Policy to associate

**Access scope**  
Type of access scope  
☒ Cluster  
☐ Kubernetes namespace

**Add policy**

**Added policies**

Policy name	Kubernetes namespaces
AmazonEKSAAdminPolicy	-

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:

- aws-auth:**  
 Este es el ConfigMap crítico para la integración de AWS IAM con Kubernetes. Permite mapear roles y usuarios de AWS a identidades y grupos en Kubernetes, gestionando la autenticación y autorización de los usuarios y nodos en el clúster. Es especialmente importante porque, al desplegar un nuevo clúster, por defecto los usuarios de IAM no tienen permisos para administrarlo, por lo que se debe configurar un rol específico y asignarlo en el bloque **mapRoles** para otorgar permisos administrativos.
- 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

devops 2403	Grupo 1	mundosE
	PIN Final	

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.

Ejecutamos el siguiente comando para visualizar y realizar un backup de la configuración actual del configMap aws-auth

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

```
ubuntu@ip-172-31-90-146:~$ kubectl get configmap aws-auth -n kube-system -o yaml
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::194722402815:role/eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqIhuZob5D
      username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2025-03-03T17:50:52Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "1489"
  uid: 6013ceb7-ab06-4604-b353-d17ba9e1a048
ubuntu@ip-172-31-90-146:~$ kubectl get configmap aws-auth -n kube-system -o yaml > aws-auth-backup.yaml
```

devops 2403	Grupo 1	mundosE
	PIN Final	

Procedemos a crear un rol, y la dejamos lo más restringida posible a solo los arn de los usuarios administradores del cluster (este paso no es necesario, podríamos poner solo uno) para luego realizar el mapeo del rol en la sección de mapRoles del configmap.

**Identity and Access Management (IAM)**

**Role EKSAdminAccessRole created.**

**EKSAdminAccessRole** Info

Rol para la asignación de permisos de administración sobre el cluster de Kubernetes mediante IAM a través del configMap/Auth

**Summary**

**Creation date**  
March 05, 2025, 17:00 (UTC-03:00)

**ARN**  
arn:aws:iam::194722402815:role/EKSAdminAccessRole

**Last activity**  
-

**Maximum session duration**  
1 hour

**Permissions** **Trust relationships** **Tags** **Last Accessed** **Revoke sessions**

**Trusted entities**

Entities that can assume this role under specified conditions.

```

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

Ingresamos a editar el configmap aws-auth mediante el siguiente comando

```
kubectl edit -n kube-system configmap/aws-auth
```

y agregamos debajo de mapRoles las siguientes 5 líneas:

```

mapRoles: |
  - rolearn: arn:aws:iam::194722402815:role/EKSAdminAccessRole
    username: admin
    groups:
      - system:masters
      - admin-users
```

devops 2403	Grupo 1	mundosE
	PIN Final	

Donde cada uno de los parámetros son:

**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 administrativo 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.

**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.

- **admin-users:**

Es un grupo adicional que puede ser usado para identificar o agrupar a administradores, lo que permite una mayor flexibilidad en la gestión de permisos y auditorías.

```
ubuntu@ip-172-31-90-146:~$ kubectl edit -n kube-system configmap/aws-auth
configmap/aws-auth edited
ubuntu@ip-172-31-90-146:~$
```

Guardamos y los cambios se impactan de forma inmediata.



devops 2403	Grupo 1	mundosE
	PIN Final	

## NGINX

### Despliegue

Se despliega un nginx expuesto públicamente a internet mediante un Servicio de tipo LoadBalancer (creando un ELB en AWS) mediante el uso de archivos manifiestos YAML:

#### nginx-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: default
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:latest
      ports:
        - containerPort: 80
      resources:
        requests:
          cpu: "250m"      # Solicita 250 milicores de CPU
          memory: "256Mi" # Solicita 256 MiB de RAM
        limits:
          cpu: "500m"      # Límite de 500 milicores de CPU
          memory: "512Mi" # Límite de 512 MiB de RAM
```

#### nginx-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: default
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"
```

devops 2403	Grupo 1	mundosE
	PIN Final	

nginx\_install.sh

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

echo "Aplicando manifiesto para el Pod de Nginx..."
kubectl apply -f nginx-pod.yaml

echo "Aplicando manifiesto para el Service de Nginx..."
kubectl apply -f nginx-service.yaml

echo "Esperando que se asigne una IP pública al Service (esto puede tardar unos minutos)..."
# Espera hasta que se asigne una IP externa
while true; do
    EXTERNAL_IP=$(kubectl get svc nginx-service -n default --output
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
    if [ -n "$EXTERNAL_IP" ]; then
        echo "El Service está disponible en: $EXTERNAL_IP (o su IP asociada)"
        break
    fi
    echo "Esperando 10 segundos para que se asigne la IP..."
    sleep 10
done

echo "Despliegue completado. Puedes acceder a Nginx desde Internet utilizando el hostname/IP asignado."
```

Output:

```
ubuntu@ip-172-31-90-146:~$ ./nginx_install.sh
Aplicando manifiesto para el Pod de Nginx...
pod/nginx-pod created
Aplicando manifiesto para el Service de Nginx...
service/nginx-service created
Esperando que se asigne una IP pública al Service (esto puede tardar unos minutos)...
Esperando 10 segundos para que se asigne la IP...
El Service está disponible en: ad13fb71070c84e8ea6f1e5acc0df0f7-2005330311.us-east-1.elb.amazonaws.com (o su IP asociada)
Despliegue completado. Puedes acceder a Nginx desde Internet utilizando el hostname/IP asignado.
ubuntu@ip-172-31-90-146:~$
```

devops 2403	Grupo 1	mundosE
	PIN Final	

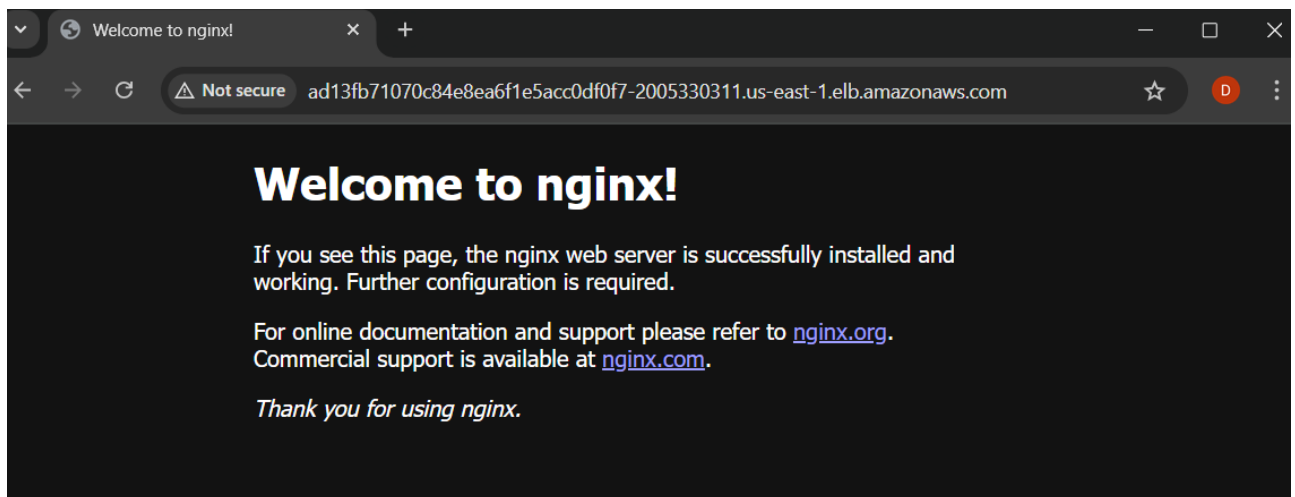
Comprobaciones:

Verificamos que el pod y el servicio hayan iniciado correctamente

```
ubuntu@ip-172-31-90-146:~$ kubectl get pods -n default
NAME      READY   STATUS    RESTARTS   AGE
nginx-pod  1/1     Running   0           63s
ubuntu@ip-172-31-90-146:~$ kubectl get svc -n default
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes  ClusterIP   10.100.0.1    <none>         443/TCP           25h
nginx-service  LoadBalancer 10.100.131.49 ad13fb71070c84e8ea6f1e5acc0df0f7-2005330311.us-east-1.elb.amazonaws.com 80:32483/TCP 69s
```

Validamos el acceso al nginx mediante la url del elb sobre el puerto 80

(<http://ad13fb71070c84e8ea6f1e5acc0df0f7-2005330311.us-east-1.elb.amazonaws.com>)



devops 2403	Grupo 1	mundosE
	PIN Final	

## 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.

## Instalación

Para la instalación del Driver se puede utilizar kubectl o helm

### kubectl

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

### 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
```

devops 2403	Grupo 1	mundosE
	PIN Final	

## Crear addon EBS

```
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:~$ 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
2025-03-04 20:23:19 [i] 1 iamserviceaccount (kube-system/ebs-csi-controller-sa) was included (based on the include/exclude rules)
2025-03-04 20:23:19 [!] serviceaccounts in Kubernetes will not be created or modified, since the option --role-only is used
2025-03-04 20:23:19 [i] 1 task: { create IAM role for serviceaccount "kube-system/ebs-csi-controller-sa" }
2025-03-04 20:23:19 [i] building iamserviceaccount stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-04 20:23:20 [i] deploying stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-04 20:23:20 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
2025-03-04 20:23:50 [i] waiting for CloudFormation stack "eksctl-2403-g1-pin-final-addon-iamserviceaccount-kube-system-ebs-csi-controller-sa"
```

```
eksctl create addon \
  --name aws-ebs-csi-driver \
  --cluster $(aws eks list-clusters --query "clusters[0]" --output text) \
  --service-account-role-arn arn:aws:iam::$(aws sts get-caller-identity --query "Account" --output text):role/AmazonEKS_EBS_CSI_DriverRole \
  --force
```

```
ubuntu@ip-172-31-90-146:~$ eksctl create addon --name aws-ebs-csi-driver --cluster $(aws eks list-clusters --query "clusters[0]" --output text) --service-account-role-arn arn:aws:iam::$(aws sts get-caller-identity --query "Account" --output text):role/AmazonEKS_EBS_CSI_DriverRole --force
2025-03-04 19:43:05 [i] Kubernetes version "1.32" in use by cluster "2403-g1-pin-final"
2025-03-04 19:43:06 [i] IRSA is set for "aws-ebs-csi-driver" addon; will use this to configure IAM permissions
2025-03-04 19:43:06 [!] 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`
```

devops 2403	Grupo 1	mundosE
	PIN Final	

devops 2403	Grupo 1	mundosE
	PIN Final	

En la siguiente imagen vemos que el driver se instaló correctamente y los pods iniciaron ok.

```
ubuntu@ip-172-31-90-146:~$ kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-1.40"
serviceaccount/ebs-csi-controller-sa created
serviceaccount/ebs-csi-node-sa created
role.rbac.authorization.k8s.io/ebs-csi-leases-role created
clusterrole.rbac.authorization.k8s.io/ebs-csi-node-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-attacher-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-resizer-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-snapshotter-role created
rolebinding.rbac.authorization.k8s.io/ebs-csi-leases-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-attacher-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-node-getter-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-provisioner-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-resizer-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-snapshotter-binding created
deployment.apps/ebs-csi-controller created
poddisruptionbudget.policy/ebs-csi-controller created
daemonset.apps/ebs-csi-node created
csidriver.storage.k8s.io/ebs.csi.aws.com created
ubuntu@ip-172-31-90-146:~$ kubectl get pods -n kube-system -l app=ebs-csi-controller
```

NAME	READY	STATUS	RESTARTS	AGE
ebs-csi-controller-bdfb955b6-96j2b	5/6	Running	0	11s
ebs-csi-controller-bdfb955b6-99nhj	5/6	Running	0	11s

```
ubuntu@ip-172-31-90-146:~$ kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
aws-node-ctcf8	2/2	Running	0	136m
aws-node-ldlgz	2/2	Running	0	136m
aws-node-rmk5t	2/2	Running	0	136m
coredns-6b9575c64c-hh9j9	1/1	Running	0	140m
coredns-6b9575c64c-ltzl9	1/1	Running	0	140m
ebs-csi-controller-bdfb955b6-96j2b	6/6	Running	0	4m13s
ebs-csi-controller-bdfb955b6-99nhj	6/6	Running	0	4m13s
ebs-csi-node-dkv57	3/3	Running	0	4m13s
ebs-csi-node-p4hzt	3/3	Running	0	4m13s

## Validar funcionamiento

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

### storageclass.yaml

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

### pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test-ebs-pvc
spec:
  accessModes:
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
- ReadWriteOnce
storageClassName: ebs-sc
resources:
  requests:
    storage: 1Gi
```

test-ebs-pod.yaml

Creamos un pod y le asignamos el pvc creado anteriormente

```
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs-pod
  namespace: default
spec:
  containers:
    - name: test-container
      image: nginx
      resources:
        requests:
          cpu: "100m"
          memory: "128Mi"
        limits:
          cpu: "500m"
          memory: "256Mi"
      volumeMounts:
        - name: ebs-volume
          mountPath: /usr/share/nginx/html
  volumes:
    - name: ebs-volume
      persistentVolumeClaim:
        claimName: test-ebs-pvc
```

Aplicamos las configuraciones mediante los siguientes comandos

kubectl apply -f storageclass.yaml

kubectl apply -f pvc.yaml

kubectl apply -f test-ebs-pod.yaml

```
ubuntu@ip-172-31-90-146:~$ kubectl apply -f storageclass.yaml
storageclass.storage.k8s.io/ebs-sc created
ubuntu@ip-172-31-90-146:~$
ubuntu@ip-172-31-90-146:~$ kubectl apply -f pvc.yaml
persistentvolumeclaim/test-ebs-pvc created
```



devops 2403	Grupo 1	mundosE
	PIN Final	

```
ubuntu@ip-172-31-90-146:~$ kubectl apply -f test-ebs-pod.yaml
pod/test-ebs-pod created
```

Error de permisos al crear el pvc:

Verificamos los logs para validar el correcto despliegue del pod de prueba con su pvc  
Observamos que luego de 30 minutos el pvc y el pod no levantan

```
ubuntu@ip-172-31-90-146:~$ kubectl get pvc test-ebs-pvc
NAME          STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
test-ebs-pvc  Pending                                     ebs-sc          <unset>       32m

ubuntu@ip-172-31-90-146:~$ kubectl get pod test-ebs-pod
NAME          READY   STATUS    RESTARTS   AGE
test-ebs-pod  0/1     Pending   0           28m
```

Ejecutamos “*kubectl describe pvc test-ebs-pvc*” y observamos un warning “ProvisioningFailed” donde el mensaje detalla que:

“api error UnauthorizedOperation: You are not authorized to perform this operation. User: arn:aws:sts::194722402815:assumed-role/eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqlhuZob5D/i-0da101a607833db30 is not authorized to perform: ec2:CreateVolume on resource: arn:aws:ec2:us-east-1:194722402815:volume/\* because no identity-based policy allows the ec2:CreateVolume action.”

```
ubuntu@ip-172-31-90-146:~$ kubectl describe pvc test-ebs-pvc
Name:          test-ebs-pvc
Namespace:     default
StorageClass:  ebs-sc
Status:        Pending
Volume:
Labels:        <none>
Annotations:   volume.beta.kubernetes.io/storage-provisioner: ebs.csi.aws.com
               volume.kubernetes.io/selected-node: ip-192-168-21-236.ec2.internal
               volume.kubernetes.io/storage-provisioner: ebs.csi.aws.com
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:  Filesystem
VolumeMode:
Used By:       test-ebs-pod
Events:
  Type      Reason      Age      From      Message
  ----      -
Normal     WaitForFirstConsumer  28m (x17 over 32m)  persistentvolume-controller  waiting for first consumer to be created before binding
Warning    ProvisioningFailed    28m                ebs.csi.aws.com/ebs-csi-controller-bdfb95b6-mqzqp_7ia3a777-8597-45fc-9ac0-9857be2470e0  failed to provision volume with StorageClass "ebs-sc": rpc error: code = Internal desc = Could not create volume "pvc-30ba5fac-5114-4298-9416-88d22915040c": could not create volume in EC2: operation error EC2: CreateVolume, https response error StatusCode: 403, RequestID: a2b10436-3001-4834-b087-b956315ad39, api error UnauthorizedOperation: You are not authorized to perform this operation. User: arn:aws:sts::194722402815:assumed-role/eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqlhuZob5D/i-0da101a607833db30 is not authorized to perform: ec2:CreateVolume on resource: arn:aws:ec2:us-east-1:194722402815:volume/* because no identity-based policy allows the ec2:CreateVolume action. Encoded authorization failure message: anQVxvFhyb3tbyYk4bAIdoaljrSaky_P3-1c0xzaJJ-1iZv6JLw4I14xQw912kvd8p67ZK0EpDCj1Yj8iaITwCj3KE14myAULAShcYnULDX641VZX0367rRQ_9IKecymIGLH_2wF7tFyWtZk0NgBj_f325UDn6ygmP6tFmW7hY4sBWPVtC1w0rCtUp7zAMJW4Q0qV7ZlRKCQ21a-0R91qtcs5MwqV5i9NwAB3VZ0tp3e4zQ08YgDa7X6S8mZcBgmPukHHw6t4_OTDF5IP8Q4nJaBYCQ14m8oJ755JQRtaPycVKVuyMGfauwPF5UIr28drfdn9I-QfQ2T1W0J1I_RALN8K2m2FLc7V_-49FE27xrnWjlyK8E0V5Tbn37Mk4Ww_07dAM4z21p4p4dmsK9oJapDpsx000x8Ycduytsd4700pymislPp2030TuxyjrpxthRk8-Y8_upA8uHjyx0lpJbx_X9b2Umk3TtMeg-d57m1A-zcvtb1AAMBypACfo6itLUUhp17g0epoKzmDD0T1NSQ-UirFsvicsgrevBc_B070tF9G7J5Qyb2R_FV3oJNgCQ0HmFY76pu-19NT7bexAheXWYb2ez47J270TZhGqRknlVkrprk8_xv27PpauB_GqH4e1A
```

El error que se observa indica que el rol de instancia del Node Group no tiene el permiso para ejecutar la acción ec2:CreateVolume. En otras palabras, cuando Kubernetes (a través del EBS CSI Driver) intenta crear un volumen EBS, el rol asumido por la instancia de EC2 ( eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqlhuZob5D) no tiene permiso para hacerlo.

devops 2403	Grupo 1	mundosE
	PIN Final	

Verificamos las políticas que tiene el rol eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-xxx

```
ubuntu@ip-172-31-90-146:~$ aws iam list-attached-role-policies --role-name eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqIhuZob5D
{
  "AttachedPolicies": [
    {
      "PolicyName": "AmazonEC2ContainerRegistryPowerUser",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPowerUser"
    },
    {
      "PolicyName": "AmazonSSMManagedInstanceCore",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
    },
    {
      "PolicyName": "AmazonEC2ContainerRegistryReadOnly",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
    },
    {
      "PolicyName": "AmazonEKSWorkerNodePolicy",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
    }
  ]
}
```

Procedemos a crear una política y asignarla al rol del node group del EKS, por medio del siguiente script:

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

# Verificar que se haya pasado el nombre del rol como argumento
if [ "$#" -ne 1 ]; then
    echo "Uso: $0 <NODE_GROUP_ROLE>"
    exit 1
fi

NODE_GROUP_ROLE="$1"
# ARN de la política administrada de Amazon EBS CSI Driver (en la partición estándar)
POLICY_ARN="arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"

echo "Adjuntando la política $POLICY_ARN al rol $NODE_GROUP_ROLE..."

# Adjuntar la política al rol
aws iam attach-role-policy --role-name "$NODE_GROUP_ROLE" --policy-arn "$POLICY_ARN"

echo "Verificando que la política se adjuntó correctamente..."
aws iam list-attached-role-policies --role-name "$NODE_GROUP_ROLE" \
    --query "AttachedPolicies[?PolicyArn=='$POLICY_ARN']" --output table

echo "La política AmazonEBSCSIDriverPolicy se ha adjuntado exitosamente al rol $NODE_GROUP_ROLE."
```

devops 2403	Grupo 1	mundosE
	PIN Final	

Ejecución y salida del script:

```
ubuntu@ip-172-31-90-146:~$ ./attach-ebs-csi-policy-nodegroup.sh eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqIhuZob5D
Adjuntando la política arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy al rol eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqIhuZob5D...
Verificando que la política se adjuntó correctamente...
-----
|                               ListAttachedRolePolicies                               |
|-----|-----|
|                               PolicyArn                               | PolicyName |
|-----|-----|
| arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy | AmazonEBSCSIDriverPolicy |
|-----|-----|
La política AmazonEBSCSIDriverPolicy se ha adjuntado exitosamente al rol eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqIhuZob5D.
```

Verificamos que se haya asignado la nueva política

```
ubuntu@ip-172-31-90-146:~$ aws iam list-attached-role-policies --role-name eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-1dmqIhuZob5D
{
  "AttachedPolicies": [
    {
      "PolicyName": "AmazonEC2ContainerRegistryPowerUser",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPowerUser"
    },
    {
      "PolicyName": "AmazonSSMManagedInstanceCore",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
    },
    {
      "PolicyName": "AmazonEC2ContainerRegistryReadOnly",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
    },
    {
      "PolicyName": "AmazonEKSWorkerNodePolicy",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
    },
    {
      "PolicyName": "AmazonEBSCSIDriverPolicy",
      "PolicyArn": "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
    }
  ]
}
```

ServiceAccount del EBS CSI Driver:

Verificamos si la service account del EBS tienen las políticas necesarias para que el controlador tenga permisos para crear, eliminar y gestionar volúmenes EBS. Para ello primero buscamos las service accounts que está utilizando el ebs-csi-controller:

```
kubectl get serviceaccount -n kube-system
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
ubuntu@ip-172-31-90-146:~$ kubectl get serviceaccount -n kube-system
NAME                                SECRETS  AGE
attachdetach-controller            0        25h
aws-cloud-provider                 0        25h
aws-node                           0        25h
certificate-controller             0        25h
clusterrole-aggregation-controller 0        25h
coredns                            0        25h
cronjob-controller                 0        25h
daemon-set-controller              0        25h
default                            0        25h
deployment-controller              0        25h
disruption-controller              0        25h
ebs-csi-controller-sa              0        10m
ebs-csi-node-sa                    0        10m
endpoint-controller                0        25h
```

Realizamos un describe de la service account con el siguiente comando

*kubectl describe serviceaccount ebs-csi-controller-sa -n kube-system*

```
ubuntu@ip-172-31-90-146:~$ kubectl describe serviceaccount ebs-csi-controller-sa -n kube-system
Name:                ebs-csi-controller-sa
Namespace:            kube-system
Labels:               app.kubernetes.io/name=aws-ebs-csi-driver
Annotations:          <none>
Image pull secrets:   <none>
Mountable secrets:    <none>
Tokens:               <none>
Events:               <none>
ubuntu@ip-172-31-90-146:~$
```

En annotations no aparece el rol por lo que no lo tiene asignado, verificamos cual es su ARN

```
ubuntu@ip-172-31-90-146:~$ aws iam list-policies --query "Policies[?PolicyName=='AmazonEBSCSIDriverPolicy'].Arn" --output text
arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
ubuntu@ip-172-31-90-146:~$
```

Mediante el siguiente script creamos un rol, le asignamos la política AmazonEBSCSIDriverPolicy y asignamos el rol a la service account.

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

# Variables (ajusta estos valores según tu entorno)
REGION="us-east-1"
CLUSTER_NAME="2403-g1-pin-final"
SERVICE_ACCOUNT="ebs-csi-controller-sa"
NAMESPACE="kube-system"
ROLE_NAME="AmazonEKS_EBS_CSI_DriverRole"

# Obtener el ID de cuenta
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
# Obtener el proveedor OIDC del clúster (se quita el prefijo "https://")
OIDC_PROVIDER=$(aws eks describe-cluster --name "$CLUSTER_NAME" --region "$REGION" --query
"cluster.identity.oidc.issuer" --output text | sed 's/https:\/\/\\\\/\\\\/')
```

```
if [ -z "$OIDC_PROVIDER" ]; then
    echo "Error: No se pudo obtener el proveedor OIDC. Asegúrate de que el clúster esté creado con --with-oidc."
    exit 1
fi

echo "Cuenta: $ACCOUNT_ID"
echo "OIDC Provider: $OIDC_PROVIDER"

# Verificar si el rol ya existe
if aws iam get-role --role-name "${ROLE_NAME}" >/dev/null 2>&1; then
    echo "El rol ${ROLE_NAME} ya existe. Se utilizará este rol."
else
    echo "Creando el rol IAM ${ROLE_NAME}..."
    # Crear el archivo de política de confianza temporalmente
    cat > trust-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::${ACCOUNT_ID}:oidc-provider/${OIDC_PROVIDER}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "${OIDC_PROVIDER}:sub": "system:serviceaccount:${NAMESPACE}:${SERVICE_ACCOUNT}"
                }
            }
        }
    ]
}
EOF

aws iam create-role --role-name "${ROLE_NAME}" --assume-role-policy-document file:///trust-policy.json

echo "Adjuntando la política AmazonEBSCSIDriverPolicy al rol ${ROLE_NAME}..."
aws iam attach-role-policy --role-name "${ROLE_NAME}" --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy

# Limpieza: eliminar el archivo temporal de política de confianza
rm trust-policy.json
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

fi

# Obtener el ARN del rol (ya existente o recién creado)
ROLE_ARN=$(aws iam get-role --role-name "${ROLE_NAME}" --query "Role.Arn" --output text)
echo "Usando el rol con ARN: ${ROLE_ARN}"

# Aplicar el patch al ServiceAccount para asociarlo con el rol
echo "Actualizando el ServiceAccount ${SERVICE_ACCOUNT} en el namespace ${NAMESPACE}..."
kubectl patch serviceaccount "${SERVICE_ACCOUNT}" -n "${NAMESPACE}" \
  -p "{\"metadata\":{\"annotations\":{\"eks.amazonaws.com/role-arn\": \"${ROLE_ARN}\"}}}"

echo "Verificando el ServiceAccount actualizado:"
kubectl get serviceaccount "${SERVICE_ACCOUNT}" -n "${NAMESPACE}" -o yaml

```

```

ubuntu@ip-172-31-90-146:~$ ./authorize-ebs-sa.sh
Cuenta: 194722402815
OIDC Provider: oidc.eks.us-east-1.amazonaws.com/id/89CE59E8EB211D52275F5FD0E36229F4
Creando el rol IAM AmazonEKS_EBS_CSI_DriverRole...
Rol creado con ARN: arn:aws:iam::194722402815:role/AmazonEKS_EBS_CSI_DriverRole
Actualizando el ServiceAccount ebs-csi-controller-sa en el namespace kube-system...
serviceaccount/ebs-csi-controller-sa patched
Verificando el ServiceAccount actualizado:
apiVersion: v1
automountServiceAccountToken: true
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::194722402815:role/AmazonEKS_EBS_CSI_DriverRole
    kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","automountServiceAccountToken":true,"kind":"ServiceAccount","metadata":{"annotations":{"eks.amazonaws.com/role-arn":"arn:aws:iam::194722402815:role/AmazonEKS_EBS_CSI_DriverRole"},"labels":{"app.kubernetes.io/name":"aws-ebs-csi-driver"},"name":"ebs-csi-controller-sa","namespace":"kube-system"}}
  creationTimestamp: "2025-03-03T20:03:45Z"
  labels:
    app.kubernetes.io/name: aws-ebs-csi-driver
  name: ebs-csi-controller-sa
  namespace: kube-system
  resourceVersion: "316552"
  uid: 2243a0d8-2239-4efc-bd86-e942ffb712ff
ubuntu@ip-172-31-90-146:~$

```

Volvemos a realizar un describe y también un get sobre la service account ebs-csi-controller-sa y vemos que se ha asignado el rol

```

ubuntu@ip-172-31-90-146:~$ kubectl describe serviceaccount ebs-csi-controller-sa -n kube-system
Name: ebs-csi-controller-sa
Namespace: kube-system
Labels: app.kubernetes.io/name=aws-ebs-csi-driver
Annotations: eks.amazonaws.com/role-arn: arn:aws:iam::194722402815:role/AmazonEKS_EBS_CSI_DriverRole
Image pull secrets: <none>
Mountable secrets: <none>
Tokens: <none>
Events: <none>
ubuntu@ip-172-31-90-146:~$

```

devops 2403	Grupo 1	mundosE
	PIN Final	

## Verificación

Eliminamos los recursos de prueba para volver a crearlos y verificar que no hayan errores.

```
kubectl delete pod test-ebs-pod -n default
```

```
kubectl delete pvc test-ebs-pvc -n default
```

```
kubectl delete sc ebs-sc
```

```
aws ec2 describe-volumes --filters "Name=status,Values=error"
```

```
ubuntu@ip-172-31-90-146:~$ kubectl delete pod test-ebs-pod -n default
pod "test-ebs-pod" deleted
ubuntu@ip-172-31-90-146:~$ kubectl delete pvc test-ebs-pvc -n default
persistentvolumeclaim "test-ebs-pvc" deleted
ubuntu@ip-172-31-90-146:~$ kubectl delete sc ebs-sc
storageclass.storage.k8s.io "ebs-sc" deleted
ubuntu@ip-172-31-90-146:~$ aws ec2 describe-volumes --filters "Name=status,Values=error"
{
  "Volumes": []
}
```

Una vez re-creado el pvc verificamos que se aprovisionó sin errores

```
ubuntu@ip-172-31-90-146:~$ kubectl describe pvc test-ebs-pvc
Name:          test-ebs-pvc
Namespace:     default
StorageClass:  ebs-sc
Status:        Bound
Volume:        pvc-0352f256-cec0-44ec-bbf2-03cc21408e7d
Labels:        <none>
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-192-168-21-236.ec2.internal
               volume.kubernetes.io/storage-provisioner: ebs.csi.aws.com
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Used By:       test-ebs-pod
Events:
  Type     Reason              Age   From                                     Message
  ----     -
  Normal   WaitForFirstConsumer 48s   persistentvolume-controller            waiting for first consumer to be created before binding
  Normal   ExternalProvisioning 46s   persistentvolume-controller            waiting for a volume to be created either by the external provisioner 'ebs.csi.aws.com' or manually by the system administrator. If volume creation is delayed, please verify that the provisioner is running and correctly registered.
  Normal   Provisioning         46s   ebs.csi.aws.com_ebs-csi-controller-bdfb955b6-mqzqp_71a3a777-0597-45fc-9ac0-9857be2470e0 External provisioner is provisioning volume for claim "default/test-ebs-pvc"
  Normal   ProvisioningSucceeded 44s   ebs.csi.aws.com_ebs-csi-controller-bdfb955b6-mqzqp_71a3a777-0597-45fc-9ac0-9857be2470e0 Successfully provisioned volume pvc-0352f256-cec0-44ec-bbf2-03cc21408e7d
```

Por último eliminamos el entorno de prueba creado (storageclass, pvc y pod)

devops 2403	Grupo 1	mundosE
	PIN Final	

## 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

prometheus\_install.sh

```
#!/bin/bash
set -euo pipefail
# 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'
echo "Creando el namespace 'prometheus' en nuestro cluster de eks"
kubectl create namespace prometheus || echo "El namespace 'prometheus' ya existe."
# Instalar Prometheus usando el chart de prometheus-community dentro del namespace prometheus
# Las opciones --set alertmanager.persistentVolume.storageClass="gp2" y --set
# server.persistentVolume.storageClass="gp2" configuran el StorageClass para los volúmenes persistentes
# tanto de Alertmanager como del servidor de Prometheus, utilizando "gp2".

echo "Instalando Prometheus en el namespace 'prometheus' "
helm install prometheus prometheus-community/prometheus \
  --namespace prometheus \
  --set alertmanager.persistentVolume.storageClass="gp2" \
  --set server.persistentVolume.storageClass="gp2"

echo "Prometheus se ha instalado correctamente."
```



devops 2403	Grupo 1	mundosE
	PIN Final	

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 "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: Tue Mar  4 20:01:35 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
```

Notas

```
Get the Prometheus server URL by running these commands in the same shell:
export POD_NAME=$(kubectl get pods --namespace prometheus -l
"app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prometheus" -o
jsonpath="{.items[0].metadata.name}")
kubectl --namespace prometheus port-forward $POD_NAME 9090

The Prometheus alertmanager can be accessed via port 9093 on the following DNS name from within your
cluster:
prometheus-alertmanager.prometheus.svc.cluster.local

Get the Alertmanager URL by running these commands in the same shell:
export POD_NAME=$(kubectl get pods --namespace prometheus -l
"app.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=prometheus" -o
jsonpath="{.items[0].metadata.name}")
kubectl --namespace prometheus port-forward $POD_NAME 9093
#####
##### WARNING: Pod Security Policy has been disabled by default since #####
##### it deprecated after k8s 1.25+. use #####
##### (index .Values "prometheus-node-exporter" "rbac" #####
##### "pspEnabled") with (index .Values #####
##### "prometheus-node-exporter" "rbac" "pspAnnotations") #####
##### in case you still need it. #####
#####

The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your
cluster:
prometheus-prometheus-pushgateway.prometheus.svc.cluster.local

Get the PushGateway URL by running these commands in the same shell:
export POD_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus-
pushgateway,component=pushgateway" -o jsonpath="{.items[0].metadata.name}")
kubectl --namespace prometheus port-forward $POD_NAME 9091
```

devops 2403	Grupo 1	mundosE
	PIN Final	

### Configurar NodePort

Actualizamos la instalación de Prometheus para cambiar 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
```

```
ubuntu@ip-172-31-90-146:~$ kubectl get nodes -o wide
NAME                                STATUS    ROLES    AGE   VERSION            INTERNAL-IP    EXTERNAL-IP
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.

devops 2403	Grupo 1	mundosE
	PIN Final	

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	mundosE
	PIN Final	

Troubleshooting pod alertmanager

Procedemos a analizar porque no inicia.

Comandos para evaluar el estado de pods

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

kubectll describe pod prometheus-alertmanager-0 -n prometheus
```

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. 0/3 nodes are available: 3 Preemption is not helpful for scheduling.
```

Comandos para validar el estado de los PVC

```
#Identificar los volúmenes y estados que tienen en el namespace prometheus
kubectll get pvc -n prometheus

# Mostrar en detalle el estado del pvc afectado
kubectll describe pvc storage-prometheus-alertmanager-0 -n prometheus
```

```
ubuntu@ip-172-31-90-146:~$ kubectll get pvc -n prometheus
NAME                                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
prometheus-server                   Bound    pvc-9ea2186f-e3d7-4c5a-9c2d-39034b7f73dd   8Gi        RWO            gp2            <unset>                  79m
storage-prometheus-alertmanager-0   Pending
ubuntu@ip-172-31-90-146:~$ kubectll describe pvc storage-prometheus-alertmanager-0 -n prometheus
Name:                               storage-prometheus-alertmanager-0
Namespace:                          prometheus
StorageClass:
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  no persistent volumes available for this claim and no storage class is set
```

Vemos que no tiene un StorageClass asignado

devops 2403	Grupo 1	mundosE
	PIN Final	

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"}}'
```

Validamos nuevamente el pvc

```
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-10d38bcf-1bc4-4af8-b7f2-ae7adbbd59f6
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-192-168-74-238.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        113s (x362 over 92m)  persistentvolume-controller  no persistent volumes available for this claim and no storage class is set
Normal     Provisioning         17s                ebs.csi.aws.com_ebs-csi-controller-ffdb6fc5f-xnv5g_6990f131-229b-439f-b21d-bb0478b7e8f5  External provisioner is provisioning volume for claim "prometheus/storage-prometheus-alertmanager-0"
Normal     ProvisioningSucceeded 15s                ebs.csi.aws.com_ebs-csi-controller-ffdb6fc5f-xnv5g_6990f131-229b-439f-b21d-bb0478b7e8f5  Successfully provisioned volume pvc-10d38bcf-1bc4-4af8-b7f2-ae7adbbd59f6
```

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

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
prometheus-server                    Bound   pvc-9ea2186f-e3d7-4c5a-9c2d-39034b7f73dd  8Gi       RWO           gp2
storage-prometheus-alertmanager-0   Bound   pvc-10d38bcf-1bc4-4af8-b7f2-ae7adbbd59f6  2Gi       RWO           gp2
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
ubuntu@ip-172-31-90-146:~$ kubectl get all -n prometheus
```

NAME	READY	STATUS	RESTARTS	AGE
pod/prometheus-alertmanager-0	1/1	Running	0	95m
pod/prometheus-kube-state-metrics-5bd466f7f6-8ndjd	1/1	Running	0	95m
pod/prometheus-prometheus-node-exporter-msgq2	1/1	Running	0	95m
pod/prometheus-prometheus-node-exporter-nt98w	1/1	Running	0	95m
pod/prometheus-prometheus-node-exporter-rhtmg	1/1	Running	0	95m
pod/prometheus-prometheus-pushgateway-544579d549-w7ftm	1/1	Running	0	95m
pod/prometheus-server-596945876b-j4csn	2/2	Running	0	95m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/prometheus-alertmanager	ClusterIP	10.100.243.103	<none>	9093/TCP	95m
service/prometheus-alertmanager-headless	ClusterIP	None	<none>	9093/TCP	95m
service/prometheus-kube-state-metrics	ClusterIP	10.100.1.127	<none>	8080/TCP	95m
service/prometheus-prometheus-node-exporter	ClusterIP	10.100.203.106	<none>	9100/TCP	95m
service/prometheus-prometheus-pushgateway	ClusterIP	10.100.137.149	<none>	9091/TCP	95m
service/prometheus-server	ClusterIP	10.100.218.243	<none>	80/TCP	95m

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	95m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/prometheus-kube-state-metrics	1/1	1	1	95m
deployment.apps/prometheus-prometheus-pushgateway	1/1	1	1	95m
deployment.apps/prometheus-server	1/1	1	1	95m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/prometheus-kube-state-metrics-5bd466f7f6	1	1	1	95m
replicaset.apps/prometheus-prometheus-pushgateway-544579d549	1	1	1	95m
replicaset.apps/prometheus-server-596945876b	1	1	1	95m

NAME	READY	AGE
statefulset.apps/prometheus-alertmanager	1/1	95m

Port forward access:

```
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 en las reglas de tráfico entrante del Security Group “bastion-sg” los puertos necesarios para publicar Prometheus (en este ejemplo 8080).

devops 2403	Grupo 1	mundosE
	PIN Final	

**Edit inbound rules** Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

**Inbound rules** Info

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	
sgr-0f7d8e7fc83786931	Custom TCP	TCP	8080	0.0.0.0/0	web	Delete
sgr-06ecf2b99e431c7e7	HTTPS	TCP	443	0.0.0.0/0	https	Delete
sgr-0e86ae78e5bb3d7c3	SSH	TCP	22	0.0.0.0/0	Allow SSH Access	Delete
sgr-0af439d9535cd44ed	Custom TCP	TCP	6417	0.0.0.0/0	grafana	Delete
sgr-0433798ff94ee3633	HTTP	TCP	80	0.0.0.0/0	Prometheus	Delete
sgr-0aacacf9c300df30	Custom TCP	TCP	3119	0.0.0.0/0	grafana	Delete

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

Prometheus Time Series Collect

Not secure ec2-3-95-154-227.compute-1.amazonaws.com:8080/query

Prometheus Query Alerts Status

Enter expression (press Shift+Enter for newlines) Execute

Table Graph Explain

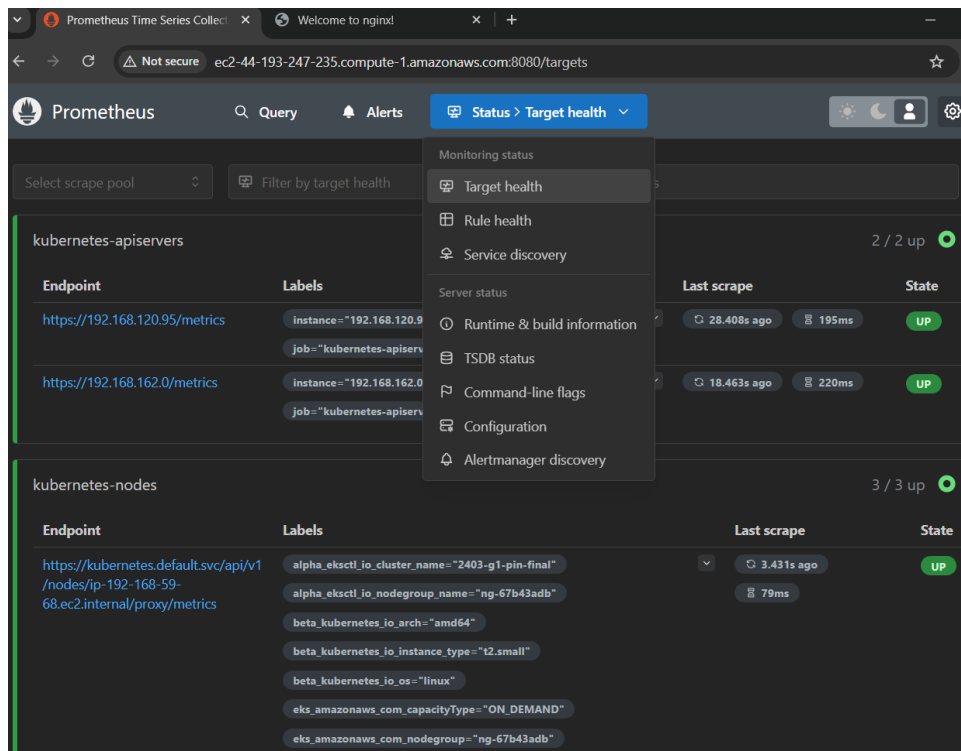
Evaluation time

No data queried yet

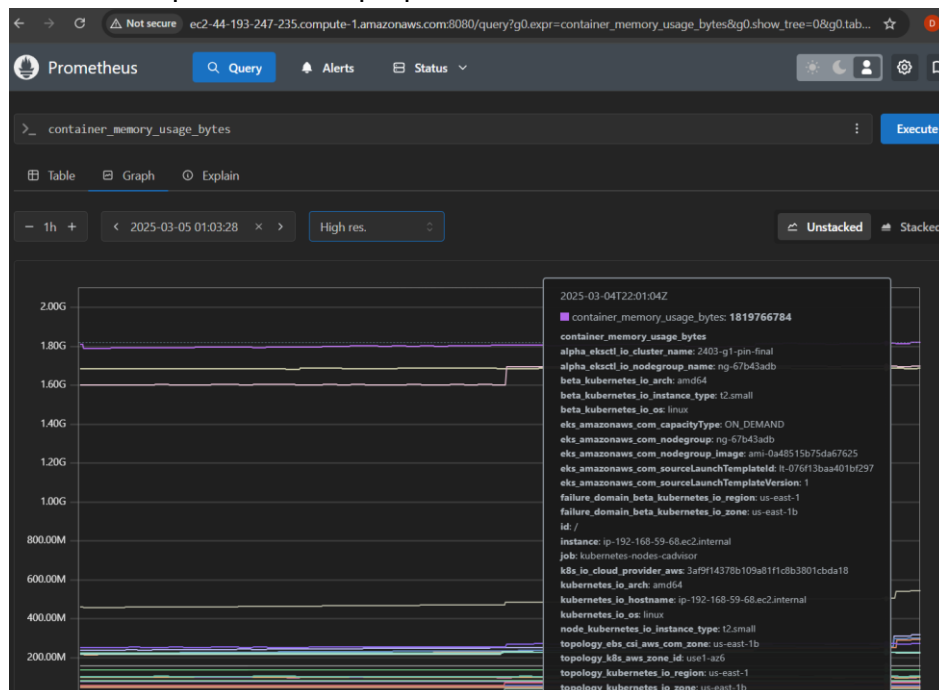
+ Add query

devops 2403	Grupo 1	mundosE
	PIN Final	

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.





devops 2403	Grupo 1	mundosE
	PIN Final	

Node port access:

<http://18.212.133.56:32000>

The screenshot shows the Prometheus web interface at the URL `18.212.133.56:32000/targets`. The page displays the 'Target health' status for two scrape pools: 'kubernetes-apiservers' and 'kubernetes-nodes'.

**kubernetes-apiservers** (2 / 2 up):

Endpoint	Labels	Last scrape	State
<a href="https://192.168.120.95/metrics">https://192.168.120.95/metrics</a>	instance="192.168.120.95:443" job="kubernetes-apiservers"	35.042s ago 267ms	UP
<a href="https://192.168.162.0/metrics">https://192.168.162.0/metrics</a>	instance="192.168.162.0:443" job="kubernetes-apiservers"	25.097s ago 238ms	UP

**kubernetes-nodes** (3 / 3 up):

Endpoint	Labels	Last scrape	State
<a href="https://kubernetes.default.svc/api/v1/nodes/ip-192-168-21-100.us-east-1.elb.amazonaws.com">https://kubernetes.default.svc/api/v1/nodes/ip-192-168-21-100.us-east-1.elb.amazonaws.com</a>	alpha_eksctl_io_cluster_name="2403-g1-pin-final"	48.578s ago 47ms	UP

devops 2403	Grupo 1	mundosE
	PIN Final	

## GRAFANA

### Instalación

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

Creamos en nuestro el directorio `${HOME}/environment/grafana` y creamos el archivo `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: "MSE!pinfinalG1." # Este valor se puede sobrescribir 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
```

devops 2403	Grupo 1	mundosE
	PIN Final	

## grafana\_install.sh

```
#!/bin/bash
# Configura el script para que se detenga si ocurre algún error,
# se usen variables no definidas o haya errores en pipelines.
set -euo pipefail

# =====
# Crear el namespace para Grafana
# =====
# Este comando crea el namespace "grafana" en Kubernetes.
kubectl create namespace grafana

# =====
# Agregar el repositorio de Helm de Grafana y actualizar la caché
# =====
# Agrega el repositorio de charts oficial de Grafana.
helm repo add grafana https://grafana.github.io/helm-charts
# Actualiza la caché de repositorios para tener la última versión del chart.
helm repo update

# =====
# Instalar Grafana usando Helm
# =====
# El siguiente comando instala Grafana en el namespace "grafana" utilizando el chart
# del repositorio "grafana/grafana". Se configuran los siguientes parámetros:
#
# --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 ${HOME}/environment/grafana/grafana.yaml: Se cargan valores adicionales desde un archivo
# YAML.
# --set service.type=LoadBalancer: Se configura el servicio para que sea de tipo LoadBalancer.
helm install grafana grafana/grafana \
  --namespace grafana \
  --set persistence.storageClassName="gp2" \
  --set persistence.enabled=true \
  --set adminPassword='MSE!pinfinalG1.' \
  --values "${HOME}/environment/grafana/grafana.yaml" \
  --set service.type=LoadBalancer

echo "Instalación de Grafana completada en el namespace 'grafana'."
```

devops 2403	Grupo 1	mundosE
	PIN Final	

Output:

```
ubuntu@ip-172-31-90-146:~$ ./grafana_install.sh
namespace/grafana created
"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 Helming!✌️
NAME: grafana
LAST DEPLOYED: Tue Mar  4 22:28:26 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
Instalación de Grafana completada en el namespace 'grafana'.
ubuntu@ip-172-31-90-146:~$
```

Verificación:

EKS

```
ubuntu@ip-172-31-90-146:~$ kubectl get all -n grafana
NAME                                READY   STATUS    RESTARTS   AGE
pod/grafana-5966b67df7-64dp2        1/1     Running   0           2m51s

NAME                                TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/grafana                     LoadBalancer  10.100.196.32    aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com  80:30388/TCP     2m51s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana             1/1     1             1           2m51s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/grafana-5966b67df7  1         1         1       2m51s
ubuntu@ip-172-31-90-146:~$
```

Busco la url del LoadBalancer Ingress:

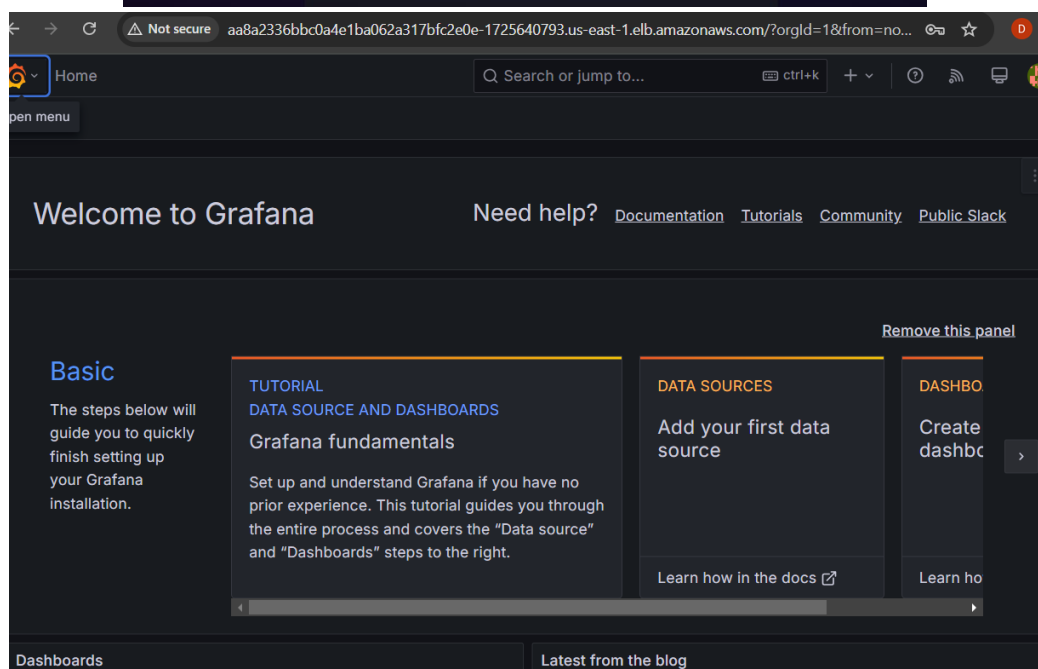
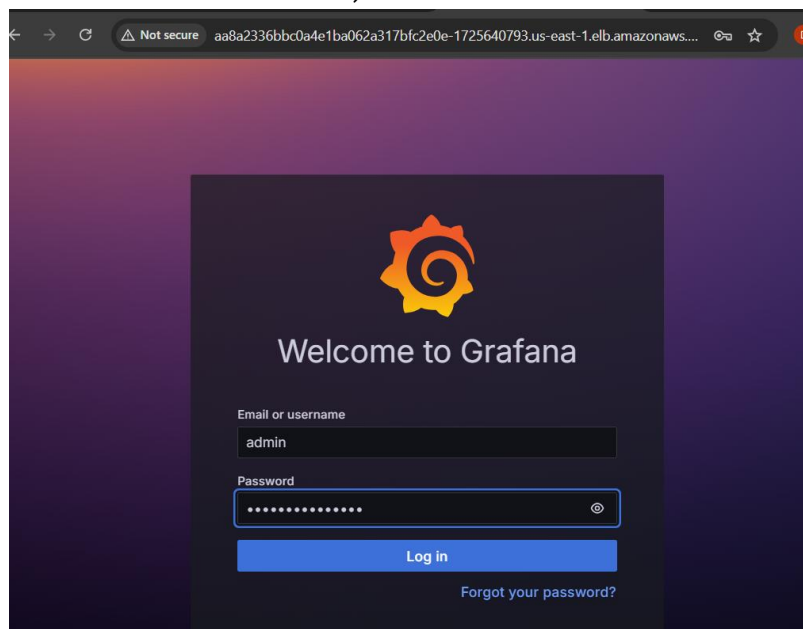
```
ubuntu@ip-172-31-90-146:~$ kubectl describe svc grafana -n grafana
Name: grafana
Namespace: grafana
Labels:
  app.kubernetes.io/instance=grafana
  app.kubernetes.io/managed-by=Helm
  app.kubernetes.io/name=grafana
  app.kubernetes.io/version=11.5.2
  helm.sh/chart=grafana-8.10.1
Annotations:
  meta.helm.sh/release-name: 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: 10.100.196.32
IPs: 10.100.196.32
LoadBalancer Ingress: aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com
Port: service 80/TCP
TargetPort: 3000/TCP
NodePort: service 30388/TCP
Endpoints: 192.168.79.161:3000
Session Affinity: None
External Traffic Policy: Cluster
Internal Traffic Policy: Cluster
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
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:~$
```

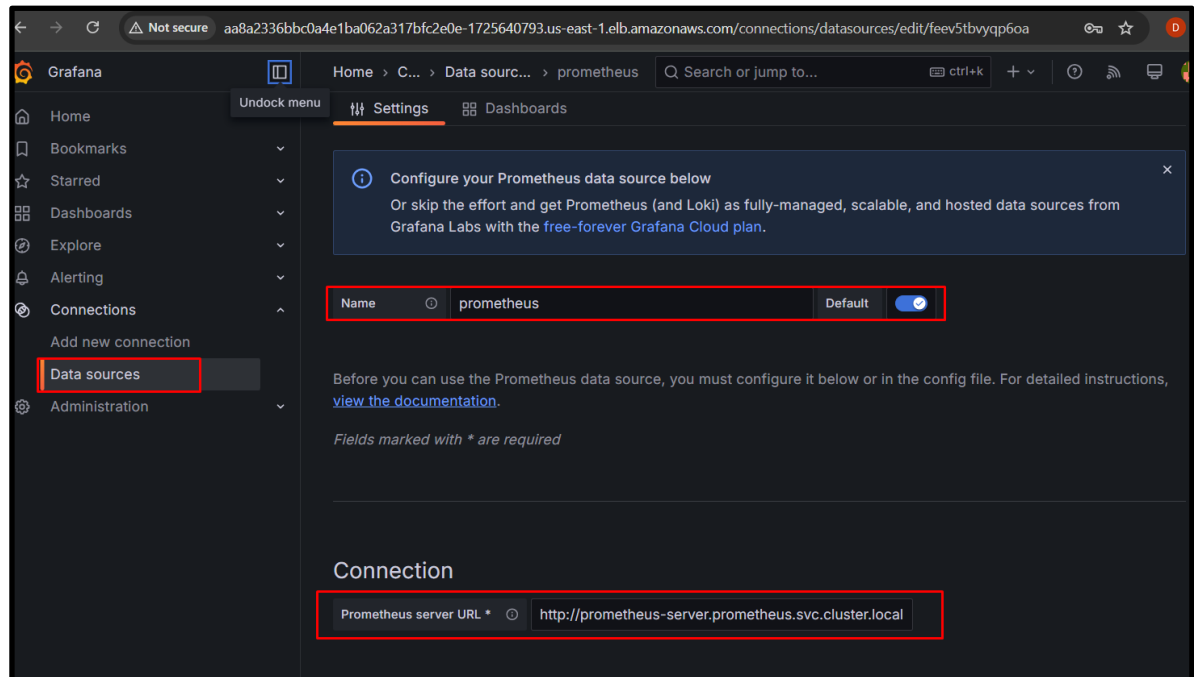
Web

Accedemos a la url del ELB Ingress (<http://aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com/>)

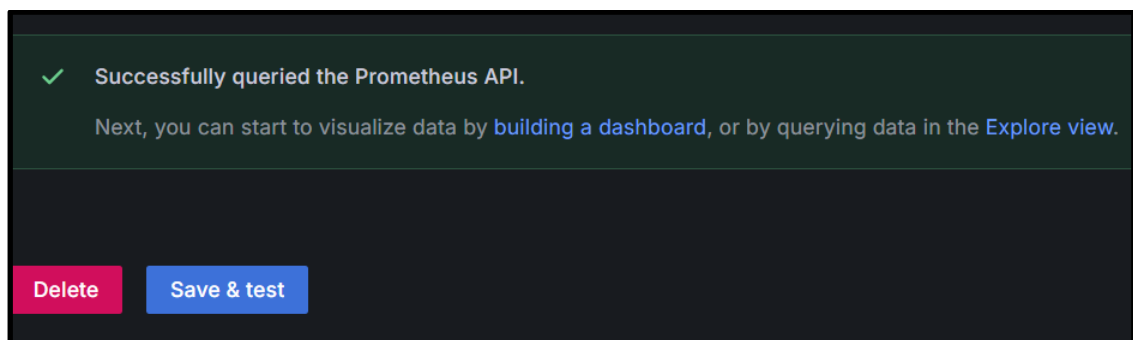


devops 2403	Grupo 1	mundosE
	PIN Final	

Verificamos que el datasource de prometheus este correctamente configurado y funcionando

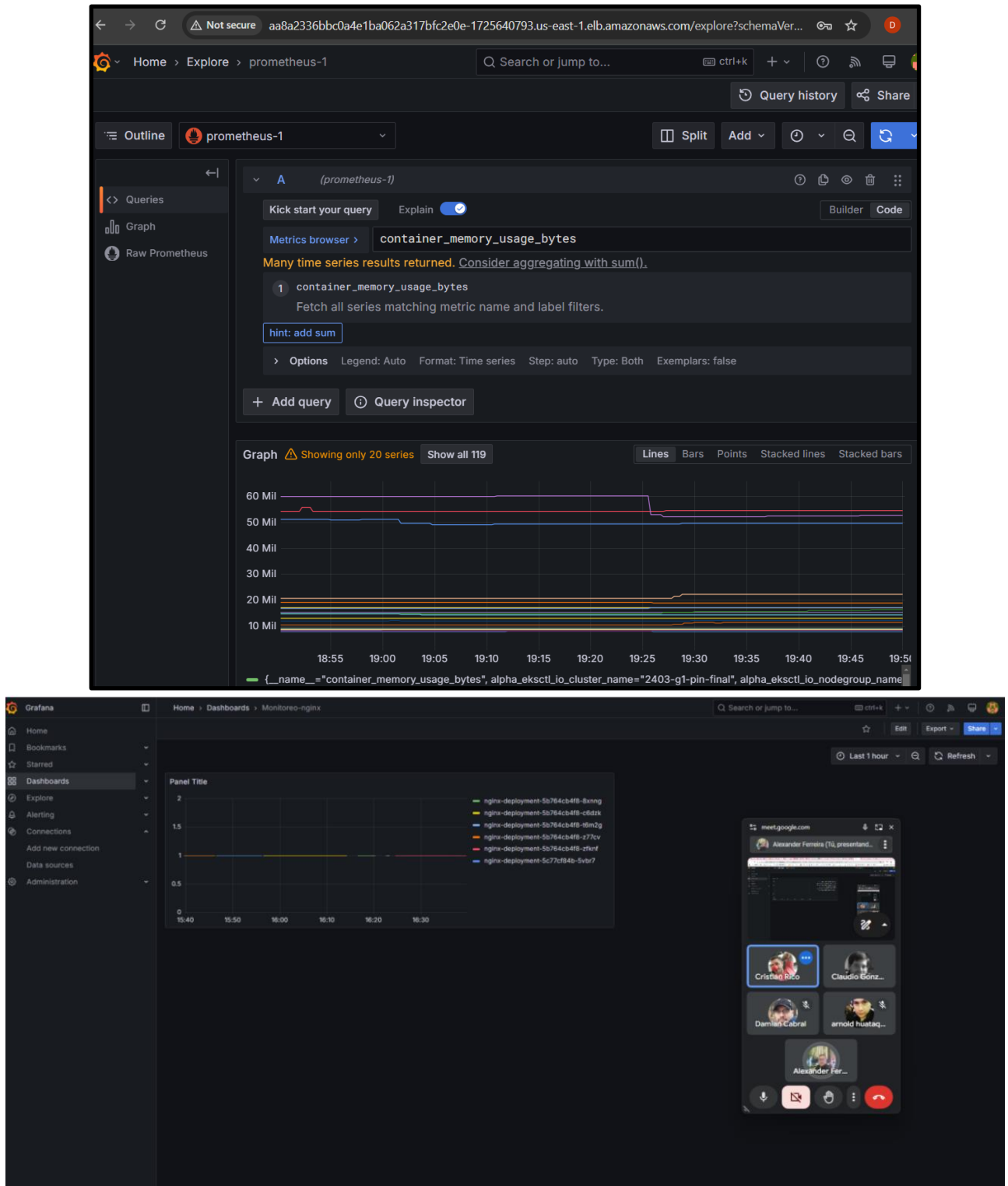


Al presionar el boton “Save & test” vemos que pasa las validaciones y puede consumir correctamente la API de Prometheus



devops 2403	Grupo 1	mundosE
	PIN Final	

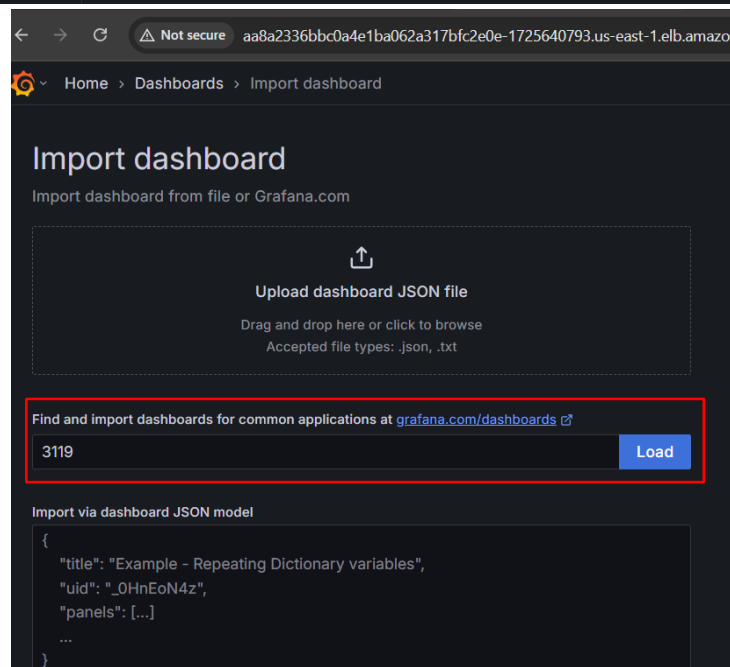
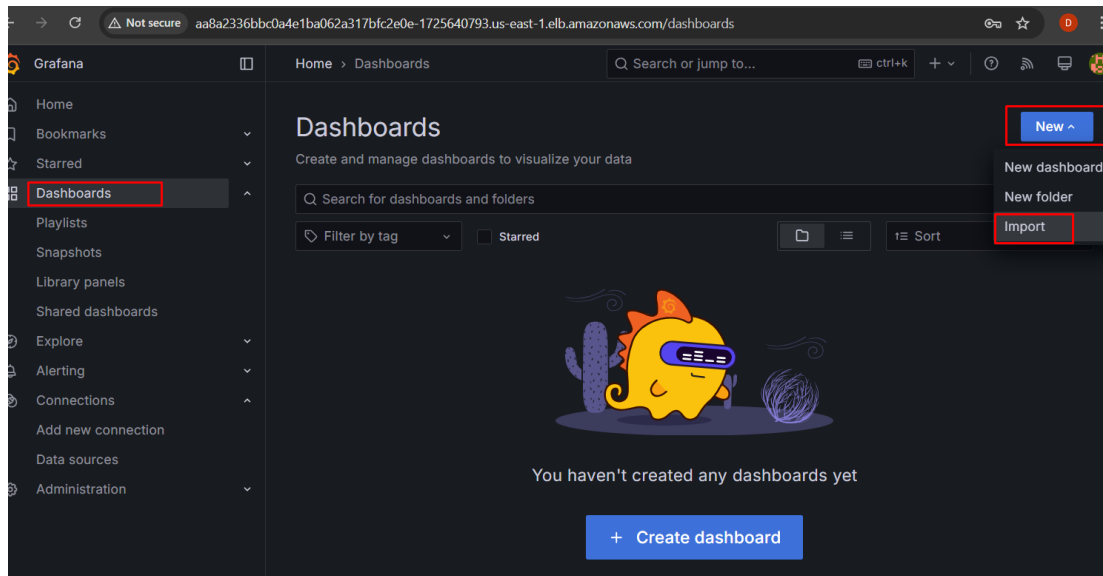
Validamos que podemos visualizar métricas de prometheus desde la sección “Explore”.



devops 2403	Grupo 1	mundosE
	PIN Final	

## Dashboard

Se importa dashboard número 3119





devops 2403	Grupo 1	mundosE
	PIN Final	

← → ↻ Not secure aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com

Home > Dashboards > Import dashboard

## Import dashboard

Import dashboard from file or Grafana.com

### Importing dashboard from Grafana.com

Published by Jjo Org

Updated on 2017-09-08 12:22:08

### Options

Name  
Kubernetes cluster monitoring (via Prometheus)

Folder  
Dashboards

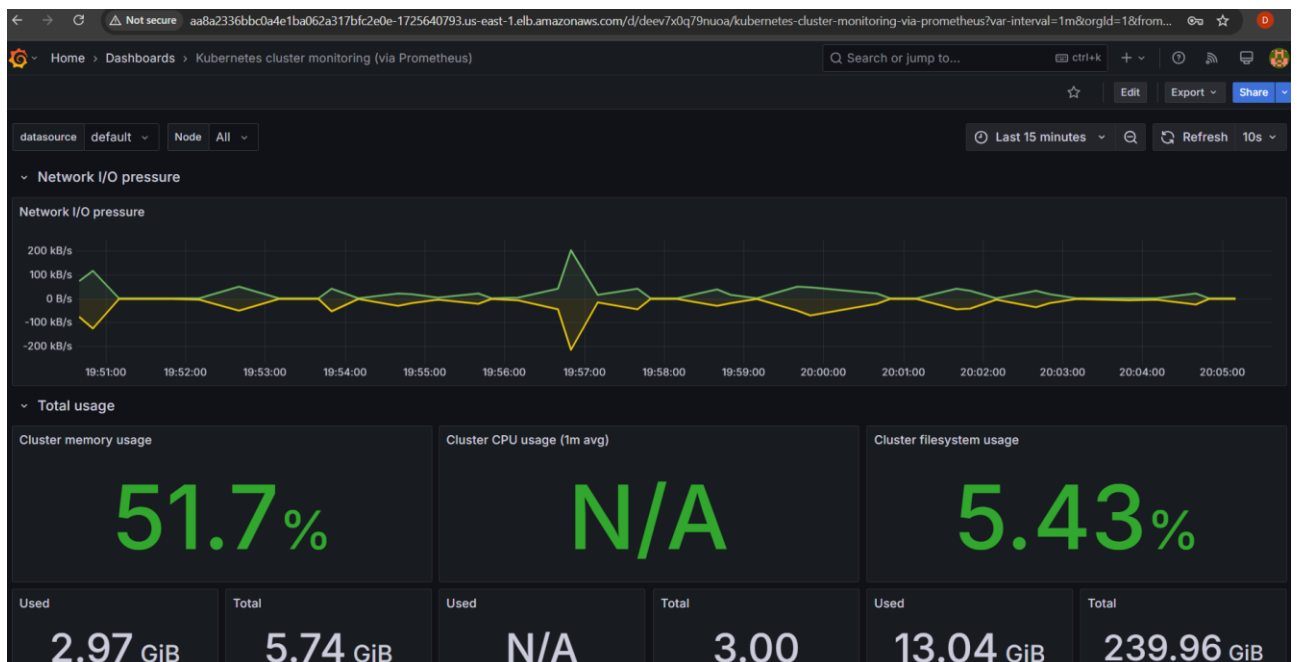
Unique identifier (UID)  
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

Change uid

prometheus

prometheus

Import Cancel



devops 2403	Grupo 1	mundosE
	PIN Final	



devops 2403	Grupo 1	mundosE
	PIN Final	

## CLEAN

Para reducir los costos de esta maqueta se creó un script para disminuir los nodos del cluster de eks a demanda.

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

# Valores por defecto
CLUSTER_NAME="2403-g1-pin-final"
NODEGROUP_NAME="ng-67b43adb" # Puedes obtenerlo con: aws eks list-nodegroups --cluster-name 2403-g1-pin-final --query "nodegroups" --output text
MIN_NODES=0
MAX_NODES=3

# Función para mostrar el uso del script
usage() {
    echo "Uso: $0 -d <nodos-deseados> [-c <nombre-del-clúster>] [-n <nombre-del-nodegroup>] [-m <mínimo-de-nodos>] [-x <máximo-de-nodos>]"
    echo ""
    echo "Parámetros:"
    echo "  -d    Cantidad de nodos deseados (obligatorio)"
    echo "  -c    Nombre del clúster (opcional, por defecto: ${CLUSTER_NAME})"
    echo "  -n    Nombre del nodegroup (opcional, por defecto: ${NODEGROUP_NAME})"
    echo "  -m    Mínimo de nodos (opcional, por defecto: ${MIN_NODES})"
    echo "  -x    Máximo de nodos (opcional, por defecto: ${MAX_NODES})"
    exit 1
}

# Variable para la cantidad de nodos deseados
DESIRED_NODES=""

# Procesar parámetros con getopt
while getopt "c:n:d:m:x:" opt; do
    case ${opt} in
        c)
            CLUSTER_NAME="${OPTARG}"
            ;;
        n)
            NODEGROUP_NAME="${OPTARG}"
            ;;
        d)
            DESIRED_NODES="${OPTARG}"
            ;;
        m)
            MIN_NODES="${OPTARG}"
            ;;
    esac
done
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

;;
x)
    MAX_NODES="$OPTARG"
    ;;
*)
    usage
    ;;
esac
done

# Verificar que se haya pasado el parámetro obligatorio para la cantidad de nodos
if [ -z "${DESIRED_NODES}" ]; then
    echo "Error: Debes especificar la cantidad de nodos deseados con -d."
    usage
fi

# Mostrar los parámetros que se usarán
echo "Parámetros:"
echo "  Clúster:          ${CLUSTER_NAME}"
echo "  Nodegroup:        ${NODEGROUP_NAME}"
echo "  Nodos deseados:    ${DESIRED_NODES}"
echo "  Mínimo de nodos:  ${MIN_NODES}"
echo "  Máximo de nodos:  ${MAX_NODES}"
echo "!!! Recordar si el numero de nodos actuales es mayor al de Nodos deseados seteado, el scale down
puede demorar hasta 10minutos, para no demorar reducir el nro maximo de nodos"

# Construir el comando de escalado
SCALE_CMD="eksctl scale nodegroup --cluster ${CLUSTER_NAME} --name ${NODEGROUP_NAME} --nodes
${DESIRED_NODES}"

# Agregar opciones opcionales
if [ -n "${MIN_NODES}" ]; then
    SCALE_CMD+=" --nodes-min ${MIN_NODES}"
fi

if [ -n "${MAX_NODES}" ]; then
    SCALE_CMD+=" --nodes-max ${MAX_NODES}"
fi

echo "Ejecutando: ${SCALE_CMD}"
${SCALE_CMD}

# Monitorear el estado del escalado en vivo
echo "Esperando a que se complete el escalado..."
while true; do
    # Obtenemos el tamaño actual del nodegroup en formato JSON y extraemos el campo CurrentSize

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

CURRENT_SIZE=$(eksctl get nodegroup --cluster "${CLUSTER_NAME}" --name "${NODEGROUP_NAME}" -o json |
jq '.[0].CurrentSize')
echo "Tamaño actual del nodegroup: ${CURRENT_SIZE} nodos."

# Si el tamaño actual coincide con el deseado, salimos del bucle
if [ "${CURRENT_SIZE}" -eq "${DESIRED_NODES}" ]; then
    echo "El escalado ha finalizado."
    break
fi

# Esperamos 10 segundos antes de volver a consultar
sleep 10
done

echo "La operación de escalado se ha completado."

```

## Output

```

ubuntu@ip-172-31-90-146:~$ ./eks-nodes-scale-config.sh
Error: Debes especificar la cantidad de nodos deseados con -d.
Uso: ./eks-nodes-scale-config.sh -d <nodos-deseados> [-c <nombre-del-clúster>] [-n <nombre-del-nodegroup>] [-m <mínimo-de-nodos>] [-x <máximo-de-nodos>]

Parámetros:
  -d  Cantidad de nodos deseados (obligatorio)
  -c  Nombre del clúster (opcional, por defecto: 2403-g1-pin-final)
  -n  Nombre del nodegroup (opcional, por defecto: ng-67b43adb)
  -m  Mínimo de nodos (opcional, por defecto: 0)
  -x  Máximo de nodos (opcional, por defecto: 3)
ubuntu@ip-172-31-90-146:~$ ./eks-nodes-scale-config.sh -d 1 -x 1
Parámetros:
  Clúster:      2403-g1-pin-final
  Nodegroup:    ng-67b43adb
  Nodos deseados: 1
  Mínimo de nodos: 0
  Máximo de nodos: 1
Ejecutando: eksctl scale nodegroup --cluster 2403-g1-pin-final --name ng-67b43adb --nodes 1 --nodes-min 0 --nodes-max 1
2025-03-05 21:38:02 [i] scaling nodegroup "ng-67b43adb" in cluster 2403-g1-pin-final
2025-03-05 21:38:02 [i] initiated scaling of nodegroup
2025-03-05 21:38:02 [i] to see the status of the scaling run `eksctl get nodegroup --cluster 2403-g1-pin-final --region us-east-1 --name ng-67b43adb`
La operación de escalado se ha completado.
ubuntu@ip-172-31-90-146:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-192-168-86-238.ec2.internal      Ready    <none>    19h    v1.32.1-eks-5d632ec

```

Para eliminar los recursos de forma jerarquica debemos ejecutar los siguientes comandos en orden:

```

helm uninstall prometheus --namespace prometheus
kubectl delete ns prometheus
helm uninstall grafana --namespace grafana
kubectl delete ns grafana
rm -rf ${HOME}/environment/grafana
eksctl delete cluster -name 2403-g1-pin-final

```

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

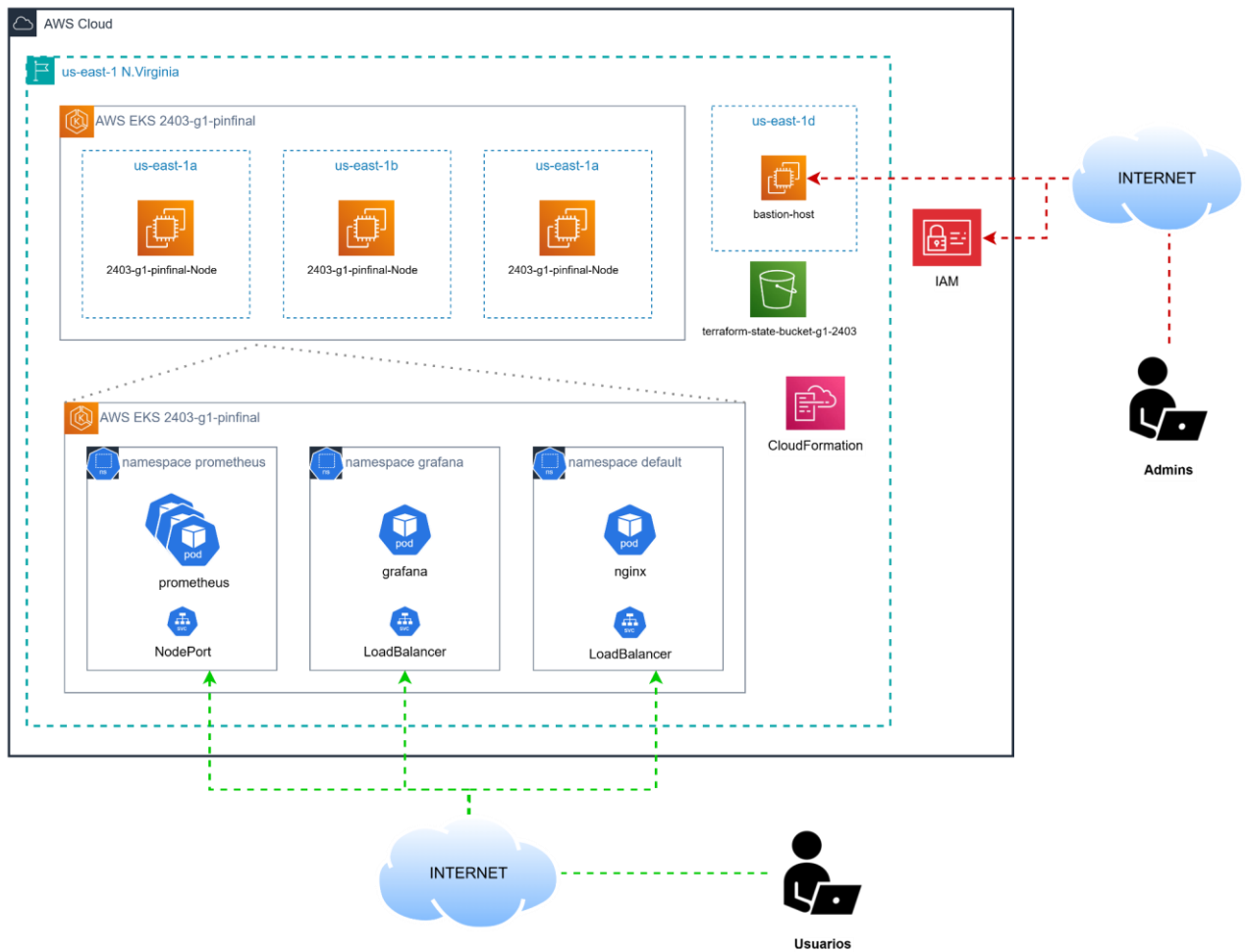
Se resolvieron desafíos en almacenamiento, compatibilidad de versiones y permisos mediante validaciones de storageClass e instalación y actualización de addons.

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.

devops 2403	Grupo 1	mundosE
	PIN Final	

## Topología general



## Observamos todos los servicios corriendo

```
ubuntu@ip-172-31-90-146:~$ kubectl get svc -A
```

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
default	kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP
default	nginx-service	LoadBalancer	10.100.131.49	ad13fb71070c84e8ea6f1e5acc0df0f7-2005330311.us-east-1.elb.amazonaws.com	80:32483/TCP
grafana	grafana	LoadBalancer	10.100.196.32	aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com	80:30388/TCP
kube-system	eks-extension-metrics-api	ClusterIP	10.100.248.253	<none>	443/TCP
kube-system	kube-dns	ClusterIP	10.100.0.10	<none>	53/UDP,53/TCP,9153/TCP
kube-system	metrics-server	ClusterIP	10.100.145.33	<none>	443/TCP
prometheus	prometheus-alertmanager	ClusterIP	10.100.243.103	<none>	9093/TCP
prometheus	prometheus-alertmanager-headless	ClusterIP	None	<none>	9093/TCP
prometheus	prometheus-kube-state-metrics	ClusterIP	10.100.1.127	<none>	8080/TCP
prometheus	prometheus-prometheus-node-exporter	ClusterIP	10.100.203.106	<none>	9100/TCP
prometheus	prometheus-prometheus-pushgateway	ClusterIP	10.100.137.149	<none>	9091/TCP
prometheus	prometheus-server	NodePort	10.100.218.243	<none>	80:32000/TCP

devops 2403	Grupo 1	mundosE
	PIN Final	

## Consumo

The screenshot displays the AWS Billing and Cost Management console. The left sidebar shows navigation options: Home, Getting Started, Billing and Payments, Bills, Payments, Credits, and Purchase Orders. The main content area is titled 'Billing and Cost Management home' and includes buttons for 'Provide feedback', 'Contact AWS support', and 'Reset layout'.

**Cost summary**

Metric	Value	Period
Month-to-date cost	\$4.13	~ compared to last month for same period
Last month's cost for same time period	\$0.00	Feb 1 - 4
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