

mundosE

PIN Final

devops 2403

Profesor:

Guazzardo, Marcelo

Grupo 1:

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

devops 2403	Grupo 1	mundosE
	PIN Final	

INTRODUCCIÓN	4
GITHUB	6
COMMITTS	6
REPOSITORY SECRETS	7
AWS	8
Configuraciones generales	8
IAM	8
Billing and Cost Management	10
Estructura de directorios	10
EC2 (Bastion)	12
KEY PAIR	12
OTROS ARCHIVOS	13
user_data.sh	13
ec2-admin.json	15
.gitignore	19
TERRAFORM	20
main.tf	20
backend.tf	20
iam.tf	21
variables.tf	21
outputs.tf	22
providers.tf	22
terraform.tfvars	22
GITHUB ACTIONS	23
Workflow	23
Ejecución workflow	24
AWS (EC2)	26
Validación de creación de instancia	26
Permisos	27
Acceso a instancia	28
Ubuntu Version	29
Paquetes	29
Elastic IP	29
EKS	31
DESPLIEGUE	31
Script crear_cluster.sh	31
Preparación	34
Ejecución de script y verificaciones	34
Permisos	37

devops 2403	Grupo 1	mundosE
	PIN Final	

configMap	37
configmap.sh	38
Test IAM User access	43
NGINX	48
Despliegue	48
Comprobaciones:	51
EBS CSI Driver	53
Crear rol	53
Instalación	54
Validar funcionamiento	55
MONITOREO	60
PROMETHEUS	60
Instalación	60
Configurar NodePort (opcional)	63
Validación:	65
Troubleshooting pod alertmanager	65
Port forward access:	68
Node port access:	70
GRAFANA	72
Instalación	72
Verificación y configs:	77
EKS	77
Web	78
Acceso	78
Datasource:	79
Métricas de Prometheus	80
Dashboard	80
CLEAN	84
REVISIÓN	85
Conclusiones	85
Topología general	86
Consumo	87

devops 2403	Grupo 1	mundosE
	PIN Final	

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, kubectl, 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>

devops 2403	Grupo 1	mundosE
	PIN Final	

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

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

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

También creamos un grupo con algunas políticas

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

devops 2403	Grupo 1	mundosE
	PIN Final	

admin-users Info

Summary

User group name: admin-users

Creation time: February 25, 2025, 16:25 (UTC-03:00)

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

Permissions policies (2) Info

You can attach up to 10 managed policies.

Search: Filter by: All types

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

AssumeRoleCustomPolicy

```

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

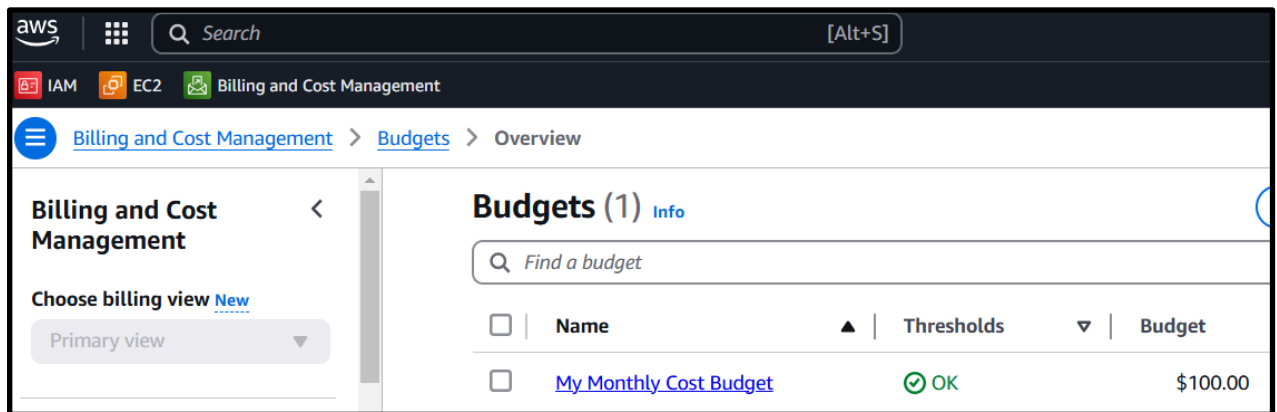
Como se ve en la imagen se agrega la siguiente custom inline policy. Se utilizará para que los usuarios IAM puedan acceder al cluster del EKS

```

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

devops 2403	Grupo 1	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
│           ├── terraform.tfvars # Valores específicos
│           └── scripts/

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

```

devops 2403	Grupo 1	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.
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
terraform version

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

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

ec2-admin.json

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

Creando este permiso permitirá que desde 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",
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

    "eks:DeleteAddon",
    "eks:DescribeAddon",
    "eks:DescribeAddonVersions",
    "eks:ListAddons",
    "eks:DescribeAddonConfiguration",

    "eks:ListUpdates",
    "eks:DescribeUpdate",

    "eks:CreateCluster",
    "eks:DeleteCluster",
    "eks:UpdateClusterVersion",
    "eks:UpdateClusterConfig",
    "eks:ListClusters",
    "eks:DescribeCluster",
    "eks:DescribeClusterVersions",

    "eks:AssociateEncryptionConfig",
    "eks:DescribeEncryptionConfig",

    "eks:AssociateIdentityProviderConfig",
    "eks:DescribeIdentityProviderConfig",
    "eks:DisassociateIdentityProviderConfig",
    "eks:ListIdentityProviderConfigs",

    "eks:TagResource",
    "eks:UntagResource",

    "eks:AccessKubernetesApi"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "iam:CreateRole",
    "iam:AttachRolePolicy",
    "iam:DetachRolePolicy",
    "iam:ListAttachedRolePolicies",
    "iam:DeleteRole",
    "iam:TagRole",
    "iam:PassRole",
    "iam:GetRole",
    "iam:GetRolePolicy",
    "iam:DeleteRolePolicy",
    "iam:ListRolePolicies",
    "iam:CreateServiceLinkedRole",
    "iam:DeleteServiceLinkedRole",
    "iam:ListRoles",
    "iam:ListPolicies",
    "iam:UpdateAssumeRolePolicy",
    "iam:PutRolePolicy",
    "iam:UpdateRoleDescription",

```


devops 2403	Grupo 1	mundosE
	PIN Final	

```

    "iam:TagOpenIDConnectProvider",
    "iam:UntagOpenIDConnectProvider",
    "iam:GetOpenIDConnectProvider",
    "iam:CreateOpenIDConnectProvider",
    "iam>DeleteOpenIDConnectProvider",
    "iam:ListOpenIDConnectProviders",
    "iam:UpdateOpenIDConnectProviderThumbprint",
    "iam:AddClientIDToOpenIDConnectProvider",
    "iam:RemoveClientIDFromOpenIDConnectProvider"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "ec2:DescribeInstances",
    "ec2:DescribeInstanceStatus",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeKeyPairs",
    "ec2:StartInstances",
    "ec2:StopInstances",
    "ec2:RebootInstances",
    "ec2:DescribeInstanceTypeOfferings",
    "ec2:CreateTags",
    "ec2>DeleteTags",

    "ec2:DescribeVolumes",
    "ec2>DeleteVolume",
    "ec2:CreateVolume",

    "ec2:CreateInternetGateway",
    "ec2:AttachInternetGateway",
    "ec2:DescribeInternetGateways",
    "ec2:DetachInternetGateway",
    "ec2>DeleteInternetGateway",

    "ec2:AllocateAddress",
    "ec2:ReleaseAddress",
    "ec2:DescribeAddresses",

    "ec2:CreateVpc",
    "ec2>DeleteVpc",
    "ec2:ModifyVpcAttribute",
    "ec2:DescribeVpcs",

    "ec2:CreateSubnet",
    "ec2:DescribeSubnets",
    "ec2>DeleteSubnet",
    "ec2:ModifySubnetAttribute",
    "ec2>DeleteSubnet",

    "ec2:CreateRouteTable",
    "ec2:DescribeRouteTables",

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

"ec2:AssociateRouteTable",
"ec2:DisassociateRouteTable",
"ec2>DeleteRouteTable",
"ec2:CreateRoute",
"ec2:ReplaceRoute",
"ec2>DeleteRoute",

"ec2:CreateSecurityGroup",
"ec2>DeleteSecurityGroup",

"ec2:DescribeNetworkInterfaces",
"ec2:CreateNatGateway",
"ec2>DeleteNatGateway",
"ec2:DescribeNatGateways",
"ec2:DetachInternetGateway",
"ec2>DeleteInternetGateway",

"ec2:AuthorizeSecurityGroupIngress",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",

"ec2:CreateNetworkInterface",
"ec2>DeleteNetworkInterface",
"ec2:ModifyNetworkInterfaceAttribute",

"ec2:CreateLaunchTemplate",
"ec2:CreateLaunchTemplateVersion",
"ec2:DescribeLaunchTemplates",
"ec2:DescribeLaunchTemplateVersions",
"ec2>DeleteLaunchTemplate",
"ec2:RunInstances"
],
"Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "elasticloadbalancing:CreateLoadBalancer",
    "elasticloadbalancing>DeleteLoadBalancer",
    "elasticloadbalancing:DescribeLoadBalancers",
    "elasticloadbalancing:AddTags",
    "elasticloadbalancing:RemoveTags"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "autoscaling:CreateAutoScalingGroup",
    "autoscaling:UpdateAutoScalingGroup",
    "autoscaling>DeleteAutoScalingGroup",
    "autoscaling:DescribeAutoScalingGroups",

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

    "autoscaling:DescribeScalingActivities",
    "autoscaling:SetDesiredCapacity",
    "autoscaling:TerminateInstanceInAutoScalingGroup",
    "autoscaling:AttachLoadBalancerTargetGroups",
    "autoscaling:DetachLoadBalancerTargetGroups"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject",
    "s3:PutObject"
  ],
  "Resource": [
    "arn:aws:s3:::terraform-state-bucket-g1-2403",
    "arn:aws:s3:::terraform-state-bucket-g1-2403/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "ssm:DescribeInstanceInformation",
    "ssm:SendCommand",
    "ssm:GetCommandInvocation",
    "ssm:StartSession",
    "ec2messages:GetMessages",
    "ec2messages:AcknowledgeMessage",
    "ec2messages:SendReply"
  ],
  "Resource": "*"
}
]
}

```

.gitignore

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

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

```

# Ignorar claves privadas y archivos sensibles
keys/
*.pem
*.swp

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

```

TERRAFORM

main.tf

```

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

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

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

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

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

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

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

```

iam.tf

```

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

      - name: Inicializar Terraform
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

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

The screenshot shows the GitHub Actions interface for the repository 'dec-wil / mundose.pinfinal.grupo1'. The 'Actions' tab is selected, displaying a workflow named 'Terraform Deploy to AWS'. A specific run is highlighted with the job name 'Agregando file backend.tf #12'. The run was triggered by a push to the 'main' branch by user 'dec-wil' 1 minute ago. The status is 'In progress'. The workflow consists of two jobs: 'plan' (status: success, duration: 15s) and 'apply' (status: in progress, duration: 1m 11s). The 'deploy.yml' file is shown with the trigger 'on: push'.

devops 2403	Grupo 1	mundosE
	PIN Final	

Se observa que el workflow se ejecuta sin errores.

The screenshot shows the GitHub Actions interface for a workflow named 'Terraform Deploy to AWS'. The workflow is triggered by a push to the 'main' branch. The run is successful, with a status of 'Success' and a total duration of 1m 13s. The workflow consists of two jobs: 'plan' (17s) and 'apply' (38s). The 'plan' job is shown as a green box with a checkmark, and the 'apply' job is also shown as a green box with a checkmark. The workflow is triggered via push 1 minute ago by user 'dec-wil' pushed to branch 'c15cb53' on the 'main' branch.

← Terraform Deploy to AWS

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

Summary

Jobs

- ✓ plan
- ✓ apply

Run details

- Usage
- Workflow file

Triggered via push 1 minute ago

dec-wil pushed c15cb53 main

Status: Success

Total duration: 1m 13s

Artifacts: —

deploy.yml

on: push

plan 17s

apply 38s

Verificación y extracción de ip pública de la instancia:

devops 2403	Grupo 1	mundosE
	PIN Final	

```

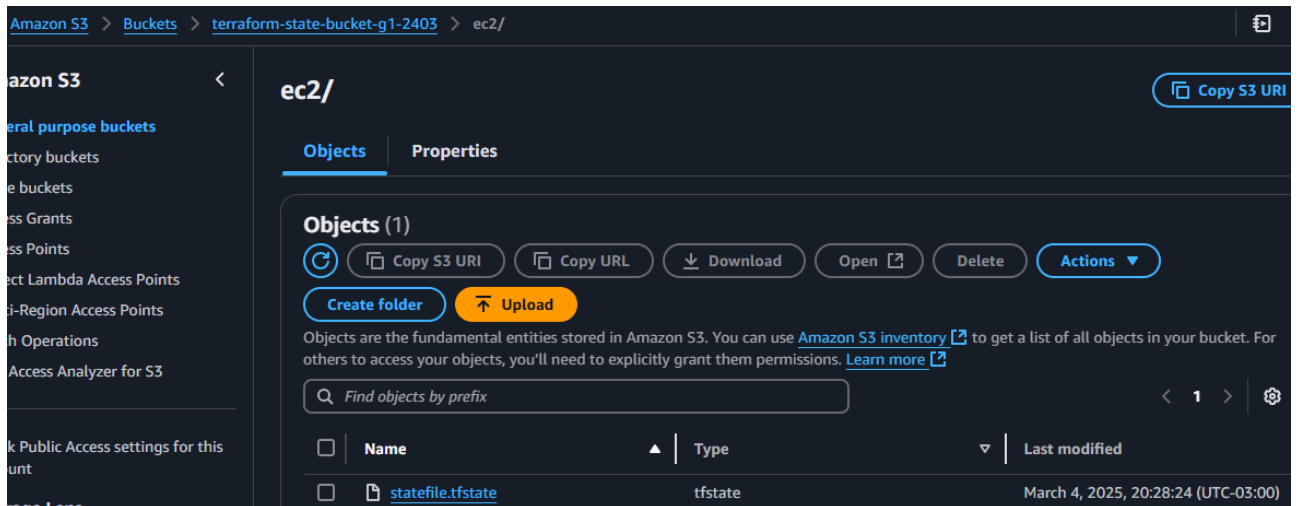
Summary
Jobs
  plan
  apply
Run details
  Usage
  Workflow file

apply
succeeded 10 minutes ago in 38s

  ✓ Aplicar cambios con Terraform (sin bloqueo)
319     ]
320   + name           = "bastion-sg"
321   + name_prefix    = (known after apply)
322   + owner_id       = (known after apply)
323   + revoke_rules_on_delete = false
324   + tags_all       = (known after apply)
325   + vpc_id         = "vpc-04adbff65ec30ad98"
326   }
327
328 Plan: 7 to add, 0 to change, 0 to destroy.
329
330 Changes to Outputs:
331   ~ bastion_public_ip = "100.27.19.49" -> (known after apply)
332 aws_key_pair.pin: Creating...
333 aws_iam_policy.ec2_admin_policy: Creating...
334 aws_iam_role.ec2_admin_role: Creating...
335 aws_security_group.bastion_sg: Creating...
336 aws_key_pair.pin: Creation complete after 0s [id=pin]
337 aws_iam_policy.ec2_admin_policy: Creation complete after 0s [id=arn:a
338 aws_iam_role.ec2_admin_role: Creation complete after 0s [id=ec2-admin
339 aws_iam_role_policy_attachment.attach_bastion_eks_policy: Creating...
340 aws_iam_instance_profile.ec2_admin_profile: Creating...
341 aws_iam_role_policy_attachment.attach_bastion_eks_policy: Creation co
342 aws_security_group.bastion_sg: Creation complete after 2s [id=sg-0e4a
343 aws_iam_instance_profile.ec2_admin_profile: Creation complete after 6
344 aws_instance.bastion: Creating...
345 aws_instance.bastion: Still creating... [10s elapsed]
346 aws_instance.bastion: Creation complete after 15s [id=i-0f1b495c8364f
347
348 Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
349
350 Outputs:
351
352 bastion_public_ip = "54.224.53.78"

```

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

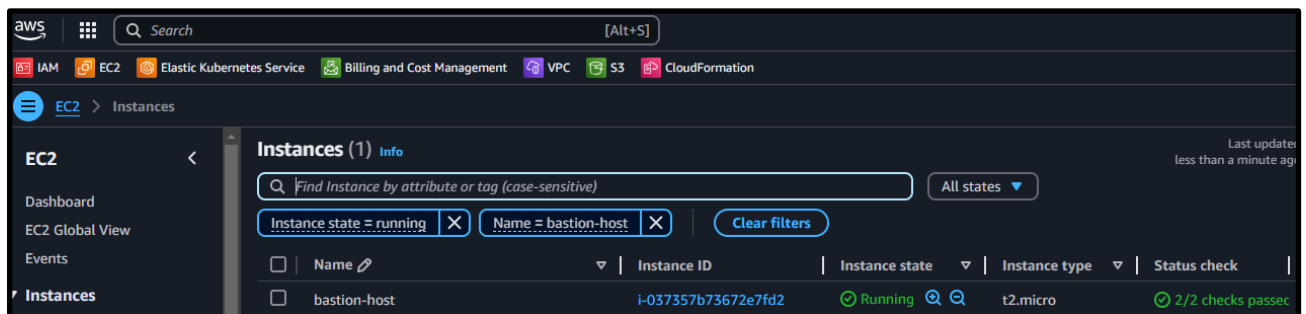


devops 2403	Grupo 1	mundosE
	PIN Final	

AWS (EC2)

Validación de creación de instancia

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



Permisos

Vemos que el rol ec2-admin se encuentra asignado

devops 2403	Grupo 1	mundosE
	PIN Final	

aws [Alt+S] [Icons: IAM, EC2, Elastic Kubernetes Service, Billing and Cost Management, VPC, S3, CloudFormation] 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
 - Elastic IPs

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

[Refresh] **Connect** **Instance state** ▾ **Actions** ▾ Info

Updated 2 minutes ago

Instance ID i-037357b73672e7fd2	Public IPv4 address 3.95.154.227 open address	Private IPv4 addresses 172.31.90.146
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-95-154-227.compute-1.amazonaws.com open address
Hostname type IP name: ip-172-31-90-146.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-90-146.ec2.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address 3.95.154.227 [Public IP]	VPC ID vpc-04adbff65ec30ad98	Auto Scaling Group name -
IAM Role ec2-admin	Subnet ID subnet-09b05791e36ae2f33	Managed false
IMDSv2 Optional EC2 recommends setting IMDSv2 to required Learn more	Instance ARN arn:aws:ec2:us-east-1:1947224028:15:instance/i-037357b73672e7fd2	
Operator -		

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.

devops 2403	Grupo 1	mundosE
	PIN Final	

aws Search [Alt+S]

IAM EC2 Elastic Kubernetes Service Billing and Cost Manage... VPC S3 CloudFormation

EC2 > Elastic IP addresses > Associate Elastic IP address

Associate Elastic IP address info

Choose the instance or network interface to associate to this Elastic IP address (44.193.247.235)

Elastic IP address: 44.193.247.235

Resource type
Choose the type of resource with which to associate the Elastic IP address.

☒ Instance
☐ Network interface

Warning 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. [more](#)

If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.

Instance
i-037357b73672e7fd2

Private IP address
The private IP address with which to associate the Elastic IP address.
172.31.90.146

Reassociation
Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.
☐ Allow this Elastic IP address to be reassociated

Observamos que la ip fue asignada correctamente.

devops 2403	Grupo 1	mundosE
	PIN Final	

The screenshot displays the AWS Management Console 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 is titled 'Instance summary for i-037357b73672e7fd2 (bastion-host)'. It includes buttons for 'Connect', 'Instance state', and 'Actions'. The instance is in a 'Running' state. Key details are listed in a grid:

- Instance ID:** i-037357b73672e7fd2
- Public IPv4 address:** 44.193.247.235 (highlighted with a red box and an 'open address' link)
- Private IPv4 addresses:** 172.31.90.146
- Public IPv4 DNS:** ec2-44-193-247-235.compute-1.amazonaws.com (highlighted with a red box and an 'open address' link)
- 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
- IAM Role:** ec2-admin
- IMDSv2:** Optional (with a warning icon and text: 'EC2 recommends setting IMDSv2 to required')
- Elastic IP addresses:** 44.193.247.235 [Public IP]
- AWS Compute Optimizer finding:** Opt-in to AWS Compute Optimizer for recommendations. (with a 'Learn more' link)
- 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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

```

usage
;;
esac
done

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

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

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

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

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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

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

echo "Credenciales verificadas. Procediendo con la creación del clúster '$CLUSTER_NAME' en la región '$AWS_REGION'."

# =====
# Creación del clúster con eksctl
# =====
# Se utiliza 'eksctl create cluster' con varios parámetros:
# --name: asigna el nombre del clúster.
# --region: define la región de AWS.
# --nodes: establece la cantidad de nodos que se desplegarán.
# --node-type: define el tipo de instancia para los nodos.
# --with-oidc: habilita la integración con OIDC.
# --ssh-access: habilita el acceso SSH a los nodos.
# --ssh-public-key: especifica la clave SSH a utilizar.
# --managed: indica que los nodos serán administrados (managed node groups).
# --full-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
    echo "Configuración del clúster completada con éxito mediante eksctl."
else
    echo "La configuración del clúster falló durante la ejecución de eksctl." >&2
    exit 1
fi
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
fi
```

Preparación

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

```
scp -i .\keys\pin.pem .\aws\eks\scripts\crear_cluster.sh ubuntu@3.95.154.227:/home/ubuntu
```

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

```
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
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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 tasks,
      update VPC CNI to use IRSA if required,
    },
    create managed nodegroup "ng-67b43adb",
  }
}
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"

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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)
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

devops 2403	Grupo 1	mundosE
	PIN Final	

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

Name	Instance ID	Instance state	Instance type
2403-g1-pin-final-ng-67b43adb-Node	i-083437b753804dcf7	Running	t2.small
2403-g1-pin-final-ng-67b43adb-Node	i-0e62e6b54a022dde1	Running	t2.small
2403-g1-pin-final-ng-67b43adb-Node	i-01e3b193e2a991cf9	Running	t2.small
bastion-host	i-037357b73672e7fd2	Running	t2.micro

Permisos

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

2403-g1-pin-final

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

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

Cluster info

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

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

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

configMap

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

devops 2403	Grupo 1	mundosE
	PIN Final	

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

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

configmap.sh

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

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


devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

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

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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

--stack-name "$STACK_NAME" \
--capabilities CAPABILITY_NAMED_IAM

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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

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

Output:

Ejecutamos el script y mostrará lo siguiente

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

Waiting for changeset to be created..
Waiting for stack create/update to complete
Successfully created/updated stack - EKSIAMAdminAccessRole-stack
Rol 'EKSIAMAdminAccessRole' creado/actualizado correctamente.

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

Configuración actual del ConfigMap aws-auth:
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::194722402815:role/eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-2XVwEgsKgzw2
      username: system:node:{{EC2PrivateDNSName}}
  mapUsers: ""
kind: ConfigMap
metadata:
  creationTimestamp: "2025-03-07T12:47:03Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "73079"
  uid: 1cbabe1d-7d38-43e9-acd6-93e6988b39a8

Debes agregar las siguientes líneas en la sección 'mapRoles:' dentro del ConfigMap 'aws-auth':

- rolearn: arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole
  username: admin
  groups:
    - system:masters

Y en la sección 'mapUsers:' puedes agregar lo siguiente para el usuario 'dcabral':

- userarn: arn:aws:iam::194722402815:user/dcabral
  username: dcabral
  groups:
    - system:masters

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

```

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

rolearn:

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

username:

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

groups:

Además, se asigna al rol dos grupos:

devops 2403	Grupo 1	mundosE
	PIN Final	

- **system:masters:**

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

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

Verificamos los cambios con el siguiente comando

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

```
ubuntu@ip-172-31-90-146:~$ kubectl get configmap aws-auth -n kube-system -o yaml
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::194722402815:role/EKSIAAdminAccessRole
      username: admin
      groups:
        - system:masters
    - groups:
        - system:bootstrappers
        - system:nodes
      rolearn: arn:aws:iam::194722402815:role/eksctl-2403-g1-pin-final-nodegroup-NodeInstanceRole-2XVwEgsKgzw2
      username: system:node:{{EC2PrivateDNSName}}
  mapUsers: |
    - userarn: arn:aws:iam::194722402815:user/dcabral
      username: dcabral
      groups:
        - system:masters
kind: ConfigMap
metadata:
  creationTimestamp: "2025-03-07T12:47:03Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "68034"
  uid: 1cbabe1d-7d38-43e9-acd6-93e6988b39a8
ubuntu@ip-172-31-90-146:~$
```

Verificamos el rol creado:

EKSIAAdminAccessRole info

 Rol para la asignación de permisos de administración sobre el clúster 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/EKSIAAdminAccessRole

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/dcabral",
10           "arn:aws:iam::194722402815:user/ahuataspique",
11           "arn:aws:iam::194722402815:user/aferrreira",
12           "arn:aws:iam::194722402815:user/egonzalez"
13         ]
14       },
15       "Action": "sts:AssumeRole"
16     ]
17   }
18 }
```

devops 2403	Grupo 1	mundosE
	PIN Final	

Test IAM User access

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

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

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

02-change-to-iamuser.sh

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

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

# Función para mostrar el uso del script
usage() {
    echo "Uso: $0 -u <usuario> [-c <cluster_name>] [-r <region>] [-a <alias>] [-R <role_arn>]"
    echo ""
    echo "  -u  Nombre del usuario IAM (AWS CLI profile) a utilizar (obligatorio)."
    echo "  -c  Nombre del cluster EKS (por defecto: 2403-g1-pin-final)."
    echo "  -r  Región AWS (por defecto: us-east-1)."
    echo "  -a  Alias para el contexto kubeconfig (por defecto: <usuario>-eks)."
    echo "  -R  ARN del rol a asumir (por defecto:
arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole)."
    exit 1
}

# Valores por defecto
CLUSTER_NAME="2403-g1-pin-final"
REGION="us-east-1"
ROLE_ARN="arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole"

# Procesar opciones
while getopts ":u:c:r:a:R:" opt; do
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

case ${opt} in
    u )
        PROFILE="$OPTARG"
        ;;
    c )
        CLUSTER_NAME="$OPTARG"
        ;;
    r )
        REGION="$OPTARG"
        ;;
    a )
        ALIAS="$OPTARG"
        ;;
    R )
        ROLE_ARN="$OPTARG"
        ;;
    \? )
        echo "Opción inválida: -$OPTARG" >&2
        usage
        ;;
    : )
        echo "La opción -$OPTARG requiere un argumento." >&2
        usage
        ;;
esac
done
# Verificar que el parámetro obligatorio -u (usuario) se haya proporcionado
if [ -z "${PROFILE:-}" ]; then
    echo "El parámetro -u <usuario> es obligatorio."
    usage
fi
# Si no se especificó alias, usar "<usuario>-eks"
if [ -z "${ALIAS:-}" ]; then
    ALIAS="${PROFILE}-eks"
fi
# Verificar si el perfil existe
if ! aws configure list-profiles | grep -q "^${PROFILE}$"; then
    echo "El perfil '${PROFILE}' no existe. Ejecutando 'aws configure --profile ${PROFILE}'..."
    aws configure --profile "${PROFILE}"
fi
echo "Actualizando kubeconfig para el cluster '${CLUSTER_NAME}' en la región '${REGION}'..."
echo "Usando el perfil '${PROFILE}' y asumiendo el rol '${ROLE_ARN}'"

aws eks update-kubeconfig \
    --name "$CLUSTER_NAME" \
    --region "$REGION" \
    --profile "$PROFILE" \

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
--role-arn "$ROLE_ARN" \
--alias "$ALIAS"

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

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

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

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

Output:

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

```
ubuntu@ip-172-31-90-146:~$ ./02-change-to-iamuser.sh -u dcabral
El perfil 'dcabral' no existe. Ejecutando 'aws configure --profile dcabral'...
AWS Access Key ID [None]: A
AWS Secret Access Key [None]: 1
Default region name [None]: us-east-1
Default output format [None]:
Actualizando kubeconfig para el cluster '2403-g1-pin-final' en la región 'us-east-1'...
Usando el perfil 'dcabral' y asumiendo el rol 'arn:aws:iam::194722402815:role/EKSIAMAdminAccessRole'
Updated context dcabral-eks in /home/ubuntu/.kube/config
Verificando los contextos existentes
CURRENT  NAME
*         arn:aws:eks:us-east-1:194722402815:cluster/2403-g1-pin-final
          dcabral-eks
          i-037357b73672e7fd2@2403-g1-pin-final.us-east-1.eksctl.io
CLUSTER
          arn:aws:eks:us-east-1:194722402815:cluster/2403-g1-pin-final
          arn:aws:eks:us-east-1:194722402815:cluster/2403-g1-pin-final
          2403-g1-pin-final.us-east-1.eksctl.io
Cambiando al contexto 'dcabral-eks'...
Switched to context "dcabral-eks".
Verificando acceso al cluster con 'kubectl get nodes'...
NAME                                STATUS  ROLES    AGE  VERSION
ip-192-168-25-154.ec2.internal      Ready   <none>   7h15m v1.32.1-eks-5d632ec
ip-192-168-52-201.ec2.internal      Ready   <none>   7h15m v1.32.1-eks-5d632ec
ip-192-168-91-252.ec2.internal      Ready   <none>   7h15m v1.32.1-eks-5d632ec
El kubeconfig se ha actualizado y el rol ha sido asumido correctamente en el contexto 'dcabral-eks'.
ubuntu@ip-172-31-90-146:~$
```

Consola Web

devops 2403	Grupo 1	mundosE
	PIN Final	

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

The screenshot shows the AWS EKS console for the cluster **2403-g1-pin-final**. The **Compute** tab is active, showing a table of 3 nodes. The nodes are all in a **Ready** status.

Node name	Instance type	Compute	Managed by	Created	Status
ip-192-168-25-154.ec2.internal	t2.small	Node group	ng-c0ddb93c	Created 7 hours ago	Ready
ip-192-168-52-201.ec2.internal	t2.small	Node group	ng-c0ddb93c	Created 7 hours ago	Ready
ip-192-168-91-252.ec2.internal	t2.small	Node group	ng-c0ddb93c	Created 7 hours ago	Ready

devops 2403	Grupo 1	mundosE
	PIN Final	

NGINX

Despliegue

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

nginx.sh

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

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

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

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

```

    kubectl create namespace "$NAMESPACE"
fi

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

if grep -q "namespace:" "$SERVICE_YAML_ORIG"; then
    sed "s/namespace: default/namespace: ${NAMESPACE}/g" "$SERVICE_YAML_ORIG" > "$SERVICE_YAML_TEMP"
else
    cp "$SERVICE_YAML_ORIG" "$SERVICE_YAML_TEMP"
fi

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

# Despliegue y validación

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

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

echo "Esperando a que el Pod '$POD_NAME' alcance el estado 'Running'..."
# Espera hasta que el Pod esté en estado Running
while true; do
    POD_STATUS=$(kubectl get pod "$POD_NAME" -n "$NAMESPACE" --output jsonpath='{.status.phase}'
2>/dev/null || echo "NotFound")
    if [ "$POD_STATUS" = "Running" ]; then
        echo "El Pod '$POD_NAME' está en estado Running."
        break
    elif [ "$POD_STATUS" = "NotFound" ]; then
        echo "El Pod '$POD_NAME' aún no existe. Esperando 5 segundos..."
    else
        echo "Estado actual del Pod: $POD_STATUS. Esperando 5 segundos..."
    fi
done

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

fi
sleep 5
done

echo "Esperando que se asigne una IP/hostname público al Service '$SERVICE_NAME' en el namespace '$NAMESPACE'..."
# Espera hasta que se asigne una IP/hostname al Service
while true; do
    EXTERNAL_IP=$(kubectl get svc "$SERVICE_NAME" -n "$NAMESPACE" --output
jsonpath='{.status.loadBalancer.ingress[0].hostname}' 2>/dev/null || echo "")
    if [ -z "$EXTERNAL_IP" ]; then
        EXTERNAL_IP=$(kubectl get svc "$SERVICE_NAME" -n "$NAMESPACE" --output
jsonpath='{.status.loadBalancer.ingress[0].ip}' 2>/dev/null || echo "")
    fi
    if [ -n "$EXTERNAL_IP" ]; then
        echo "El Service '$SERVICE_NAME' está disponible en: $EXTERNAL_IP"
        break
    fi
    echo "Esperando 10 segundos para que se asigne la IP/hostname..."
    sleep 10
done

echo "Despliegue completado en el namespace '$NAMESPACE'. Puedes acceder a Nginx desde Internet
utilizando el hostname/IP asignado."

# Limpieza de archivos temporales
rm -f "$POD_YAML_TEMP" "$SERVICE_YAML_TEMP"

```

nginx-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: "50m"      # Solicita 50 milicores de CPU

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
memory: "64Mi" # Solicita 64 MiB de RAM
limits:
  cpu: "100m"    # Límite de 100 milicores de CPU
  memory: "128Mi" # Límite de 128 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"
```

Output:

```
ubuntu@ip-172-31-90-146:~$ ./nginx.sh
Aplicando manifiesto para el Pod de Nginx en el namespace 'nginx'...
pod/nginx-pod created
Aplicando manifiesto para el Service de Nginx en el namespace 'nginx'...
service/nginx-service created
Esperando a que el Pod 'nginx-pod' alcance el estado 'Running'...
El Pod 'nginx-pod' está en estado Running.
Esperando que se asigne una IP/hostname público al Service 'nginx-service' en el namespace 'nginx'...
Esperando 10 segundos para que se asigne la IP/hostname...
El Service 'nginx-service' está disponible en: adb25ed9f667a4b85a3cd351a0735ef8-444609040.us-east-1.elb.amazonaws.com
Despliegue completado en el namespace 'nginx'. Puedes acceder a Nginx desde Internet utilizando el hostname/IP asignado.
ubuntu@ip-172-31-90-146:~$
```

Comprobaciones:

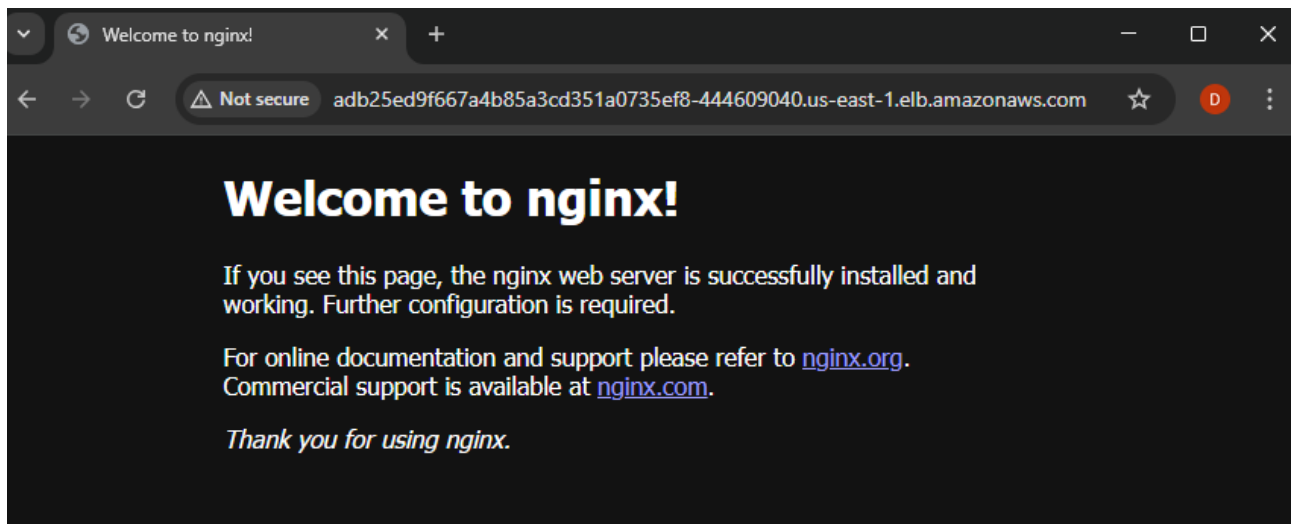
Verificamos que el pod y el servicio haya iniciado correctamente

```
ubuntu@ip-172-31-90-146:~$ kubectl get all -n nginx
NAME          READY   STATUS    RESTARTS   AGE
pod/nginx-pod  1/1     Running   0           3m35s

NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/nginx-service  LoadBalancer  10.100.211.67  adb25ed9f667a4b85a3cd351a0735ef8-444609040.us-east-1.elb.amazonaws.com  80:32501/TCP    3m34s
ubuntu@ip-172-31-90-146:~$
```

Validamos el acceso al nginx mediante la url del elb sobre el puerto 80
(<http://adb25ed9f667a4b85a3cd351a0735ef8-444609040.us-east-1.elb.amazonaws.com>)

devops 2403	Grupo 1	mundosE
	PIN Final	



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.

Crear rol

Consultamos el nombre de la service account de ebs

```
ubuntu@ip-172-31-90-146:~$ kubectl get serviceaccount -n kube-system | grep ebs
ebs-csi-controller-sa          0          7m33s
ebs-csi-node-sa                0          7m33s
ubuntu@ip-172-31-90-146:~$
```

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

```
eksctl create iamserviceaccount \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster 2403-g1-pin-final \
  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --approve \
  --role-only \
  #parametro para crear el rol sin la serviceaccount
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
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"
```

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

Instalación

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

eksctl

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

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

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

helm

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

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

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

kubectl

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


devops 2403	Grupo 1	mundosE
	PIN Final	

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

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

Output:

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

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

Validar funcionamiento

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

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

00-test-create-volume.sh

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

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

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

# Procesar parámetros
DELETE_MODE=false
while getopts ":d" opt; do
  case ${opt} in

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

# Variables de configuración
NAMESPACE="default"
STORAGECLASS_YAML="01-storageclass.yaml"
PVC_YAML="02-pvc.yaml"
POD_YAML="03-pod-ebs-test.yaml"

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

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

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

    echo "Recursos eliminados."
    exit 0
fi

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

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

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

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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

    echo "El PVC '$PVC_NAME' se encuentra en estado 'Bound'."
    break
fi
echo "Estado actual del PVC: $PVC_STATUS. Esperando 5 segundos..."
sleep 5
done

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

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

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

```

01-storageclass.yaml

```

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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

pvc.yaml

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

02-pod-ebs-test.yaml

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

Output:

Luego de ejecutar 00-test-create-volume.sh

```
ubuntu@ip-172-31-90-146:~$ ./00-test-pvc.sh
Aplicando manifiesto para el StorageClass...
storageclass.storage.k8s.io/ebs-sc created
Aplicando manifiesto para el PersistentVolumeClaim...
persistentvolumeclaim/test-ebs-pvc created
Aplicando manifiesto para el Pod de prueba...
pod/test-ebs-pod created
Validando que el PVC 'test-ebs-pvc' esté en estado 'Bound'...
Estado actual del PVC: Pending. Esperando 5 segundos...
El PVC 'test-ebs-pvc' se encuentra en estado 'Bound'.
Validando que el Pod 'test-ebs-pod' esté en estado 'Running'...
Estado actual del Pod: Pending. Esperando 5 segundos...
El Pod 'test-ebs-pod' se encuentra en estado 'Running'.
Mostrando los recursos creados en el namespace 'default':
StorageClass:
NAME      PROVISIONER      RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ebs-sc    ebs.csi.aws.com   Delete          WaitForFirstConsumer false                   17s

PersistentVolumeClaim:
NAME      STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE     VOLUME MODE
test-ebs-pvc Bound     pvc-78750e61-50ea-4663-8216-2ce45d214cda   1Gi        RWO            ebs-sc         <unset>                  17s     Filesystem

Pod:
NAME      READY   STATUS    RESTARTS   AGE   IP              NODE                                     NOMINATED NODE   READINESS GATES
test-ebs-pod 1/1     Running   0          16s   192.168.49.157  ip-192-168-52-201.ec2.internal         <none>           <none>
Despliegue completado. Puedes acceder a Nginx utilizando el Service (si está expuesto) o verificar el contenido en el Pod.
```

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

Eliminamos los recursos de prueba

```
ubuntu@ip-172-31-90-146:~$ ./00-test-pvc.sh -d
Modo delete activado: eliminando StorageClass, PVC y Pod de la prueba...
pod "test-ebs-pod" deleted
persistentvolumeclaim "test-ebs-pvc" deleted
Warning: deleting cluster-scoped resources, not scoped to the provided namespace
storageclass.storage.k8s.io "ebs-sc" deleted
Recursos eliminados.
ubuntu@ip-172-31-90-146:~$
```

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

Realizaremos la instalación mediante un script que ejecuta helm.

`prometheus_install.sh`

Modo Delete:

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

Instalación:

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

Validación:

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

Recordatorio:

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

devops 2403	Grupo 1	mundosE
	PIN Final	

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

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

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

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

NAMESPACE="prometheus"
RELEASE_NAME="prometheus"

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

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

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

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

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

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

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


devops 2403	Grupo 1	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 "grafana" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ✌Happy Helming!✌
Creando el namespace 'prometheus'...
namespace/prometheus created
Instalando Prometheus en el namespace 'prometheus'...
NAME: prometheus
LAST DEPLOYED: Fri Mar 7 22:41:27 2025
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local

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

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

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

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

Configurar NodePort (opcional)

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

```
helm upgrade prometheus prometheus-community/prometheus \
  --namespace prometheus \
  --reuse-values \
```

devops 2403	Grupo 1	mundosE
	PIN Final	

```
--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
```

devops 2403	Grupo 1	mundosE
	PIN Final	

Validación:

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

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

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

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

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

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

Troubleshooting pod alertmanager

Procedemos a analizar porque no inicia.

Comandos para evaluar el estado de pods

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

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

Comandos para validar el estado de los PVC

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

```
ubuntu@ip-172-31-90-146:~$ kubectl get pvc -n prometheus
NAME                                STATUS  VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  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:~$ kubectl describe pvc storage-prometheus-alertmanager-0 -n prometheus
Name:                               storage-prometheus-alertmanager-0
Namespace:                           prometheus
StorageClass:                         gp2
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.

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

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

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

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

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   VOLUMEATTRIBUTESCLASS   AGE
prometheus-server                   Bound    pvc-c05da9a7-e317-4638-b044-ce2c8e2ef483   8Gi        RWO            gp2            <unset>                  21m
storage-prometheus-alertmanager-0   Bound    pvc-90c2836d-b3e4-4908-a728-1878f342f2a5   2Gi        RWO            gp2            <unset>                  21m

ubuntu@ip-172-31-90-146:~$ kubectl get all -n prometheus
NAME                                READY    STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0       1/1     Running   0           22m
pod/prometheus-kube-state-metrics-5bd466f7f6-dq24c  1/1     Running   0           22m
pod/prometheus-prometheus-node-exporter-dxht6       1/1     Running   0           22m
pod/prometheus-prometheus-node-exporter-krvnn       1/1     Running   0           22m
pod/prometheus-prometheus-node-exporter-mwjqs       1/1     Running   0           22m
pod/prometheus-prometheus-pushgateway-544579d549-wnwhj 1/1     Running   0           22m
pod/prometheus-server-596945876b-lk162              2/2     Running   0           22m

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

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

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

Port forward access:

En nuestro bastion host ejecutamos el siguiente comando

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

```
ubuntu@ip-172-31-90-146:~$ kubectll 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 && kubectll port-forward -n prometheus
deploy/prometheus-server 8080:9090 --address 0.0.0.0 > /var/log/eks/prometheus-port-forward.log 2>&1 &
```

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

Edit inbound rules Info

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

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

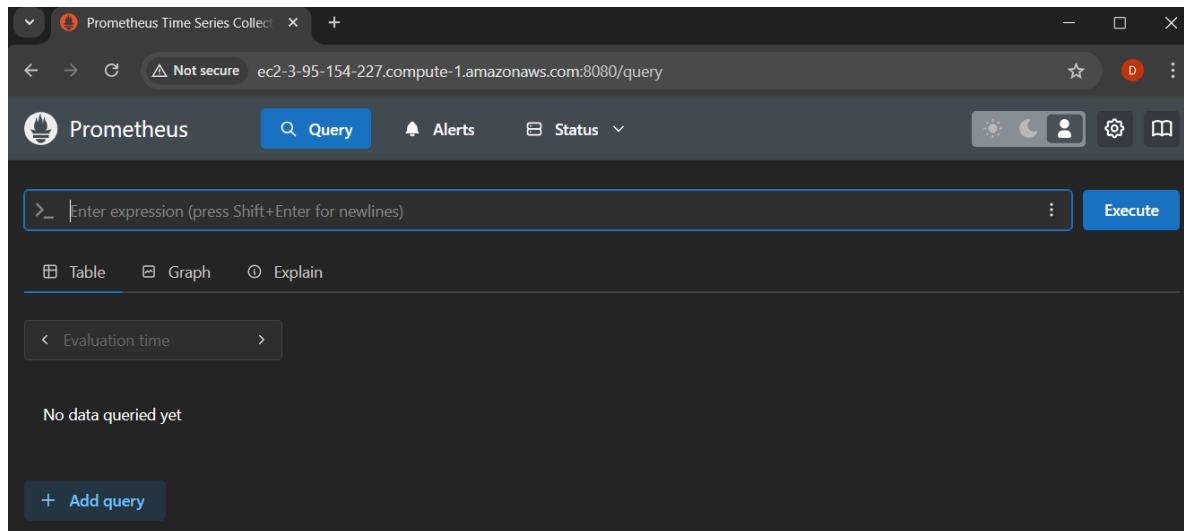
[Add rule](#)

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

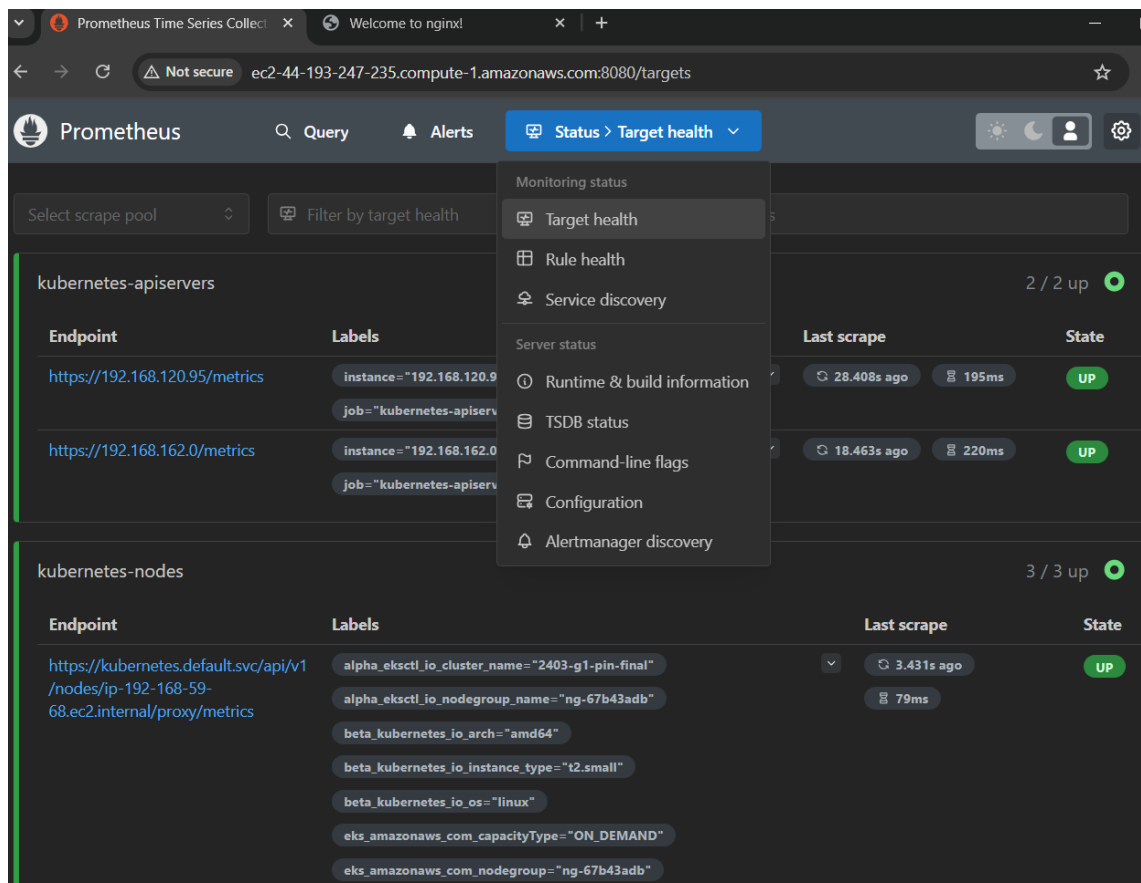
[Cancel](#) [Preview changes](#) [Save rules](#)

devops 2403	Grupo 1	mundosE
	PIN Final	

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

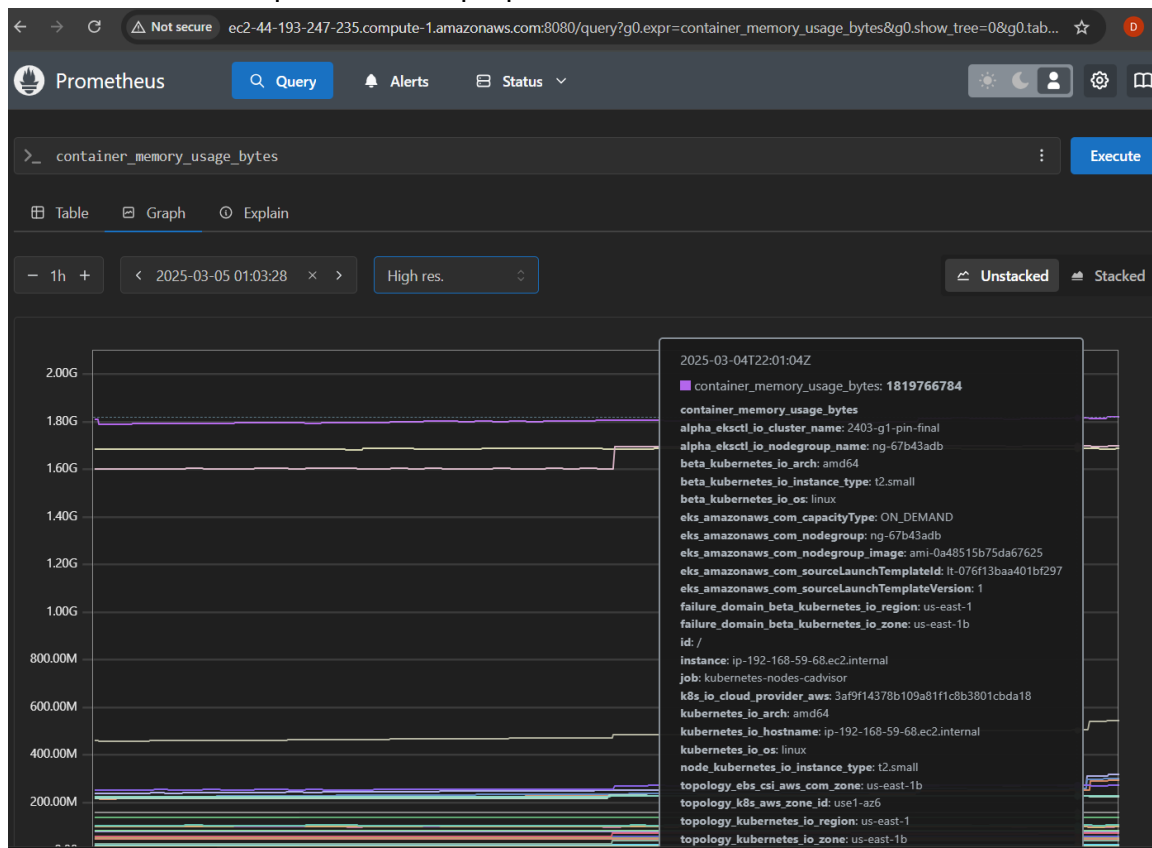


Nos dirigimos a “Status -> Target health” donde vemos el estado de los diferentes endpoints de nuestro cluster de EKS.



devops 2403	Grupo 1	mundosE
	PIN Final	

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



Node port access:

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

devops 2403	Grupo 1	mundosE
	PIN Final	

Security Group de los nodos:

Edit inbound rules Info

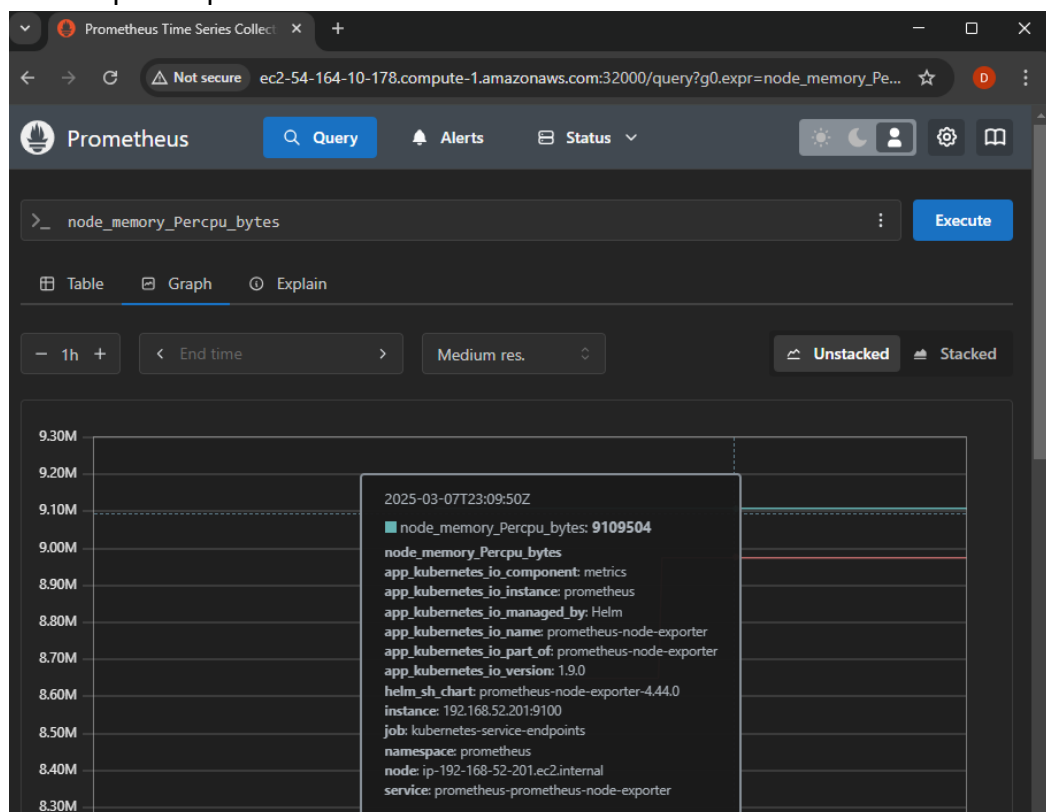
Inbound rules control the incoming traffic that's allowed to reach the instance.

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

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

Cancel Preview changes Save rules

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



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 y creamos el archivo grafana.yaml

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

```
ubuntu@ip-172-31-90-146:~$ mkdir -p ${HOME}/environment/grafana
ubuntu@ip-172-31-90-146:~$ cd ${HOME}/environment/grafana
ubuntu@ip-172-31-90-146:~/environment/grafana$ pwd
/home/ubuntu/environment/grafana
ubuntu@ip-172-31-90-146:~/environment/grafana$ vim grafana.yaml
```

grafana.yaml

```
# grafana.yaml
replicaCount: 1

grafana:
  adminUser: admin
  adminPassword: "" # Este valor se sobrescribe 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	

devops 2403	Grupo 1	mundosE
	PIN Final	

grafana_install.sh

Modo Delete (-d):

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

Gestión de la contraseña:

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

Instalación de Grafana:

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

Verificación:

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

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

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

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

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

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

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

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

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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

```

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

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

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

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

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

```

devops 2403	Grupo 1	mundosE
	PIN Final	

Output:

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

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

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

    grafana.grafana.svc.cluster.local

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

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

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

Verificación y configs:

EKS

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

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

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

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

Busco la url del LoadBalancer Ingress:

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

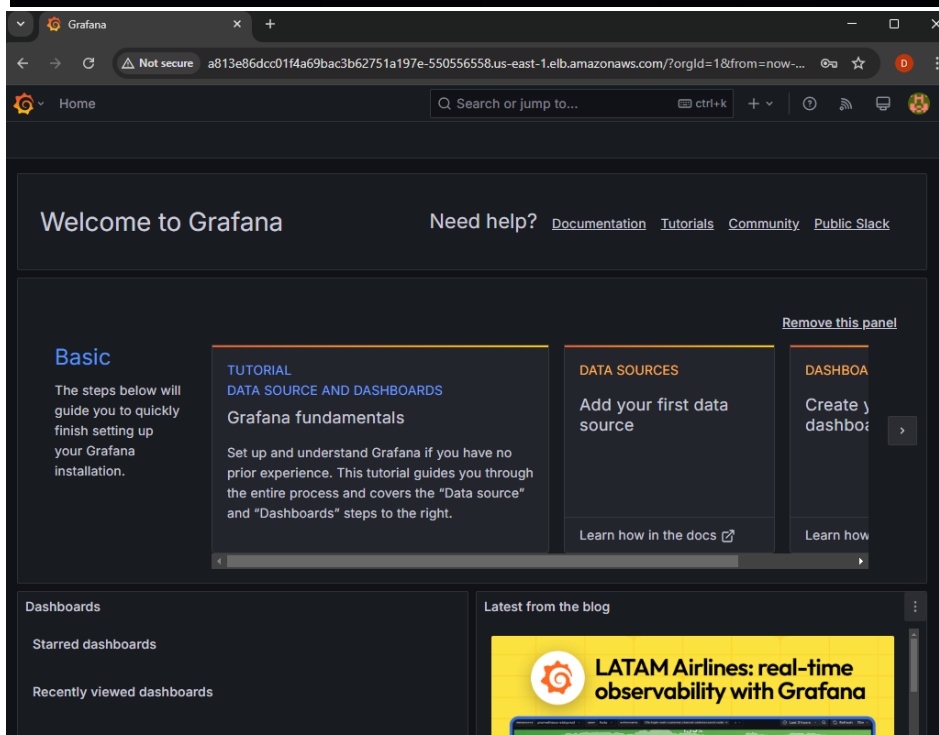
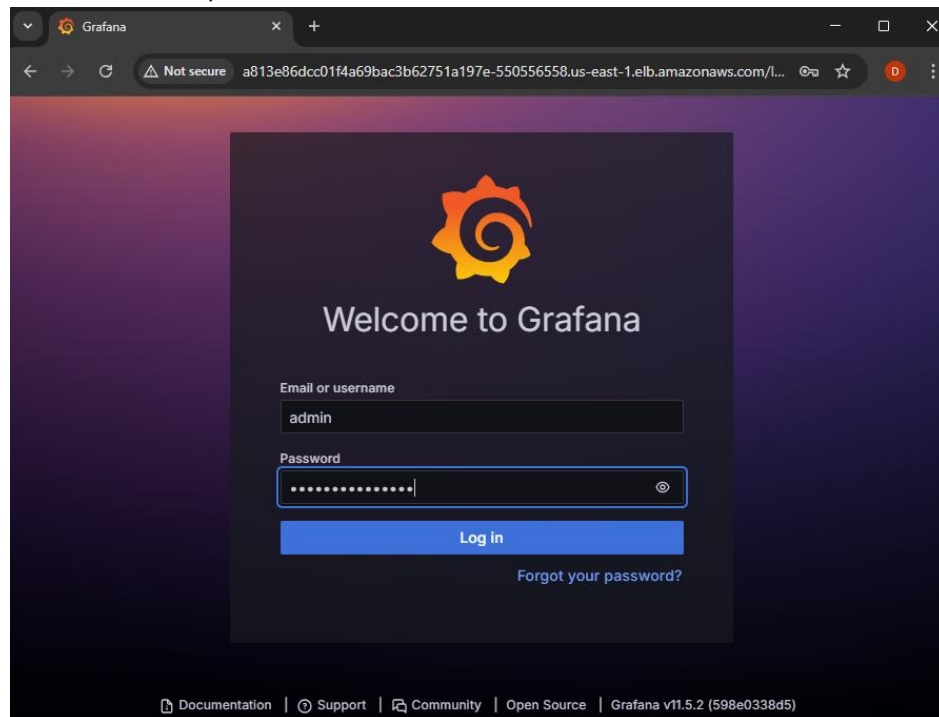
```
ubuntu@ip-172-31-90-146:~$ kubectl get svc --namespace grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0]}'
{"hostname": "aa8a2336bbc0a4e1ba062a317bfc2e0e-1725640793.us-east-1.elb.amazonaws.com"}ubuntu@ip-172-31-90-146:~$
ubuntu@ip-172-31-90-146:~$
```


devops 2403	Grupo 1	mundosE
	PIN Final	

Web

Acceso

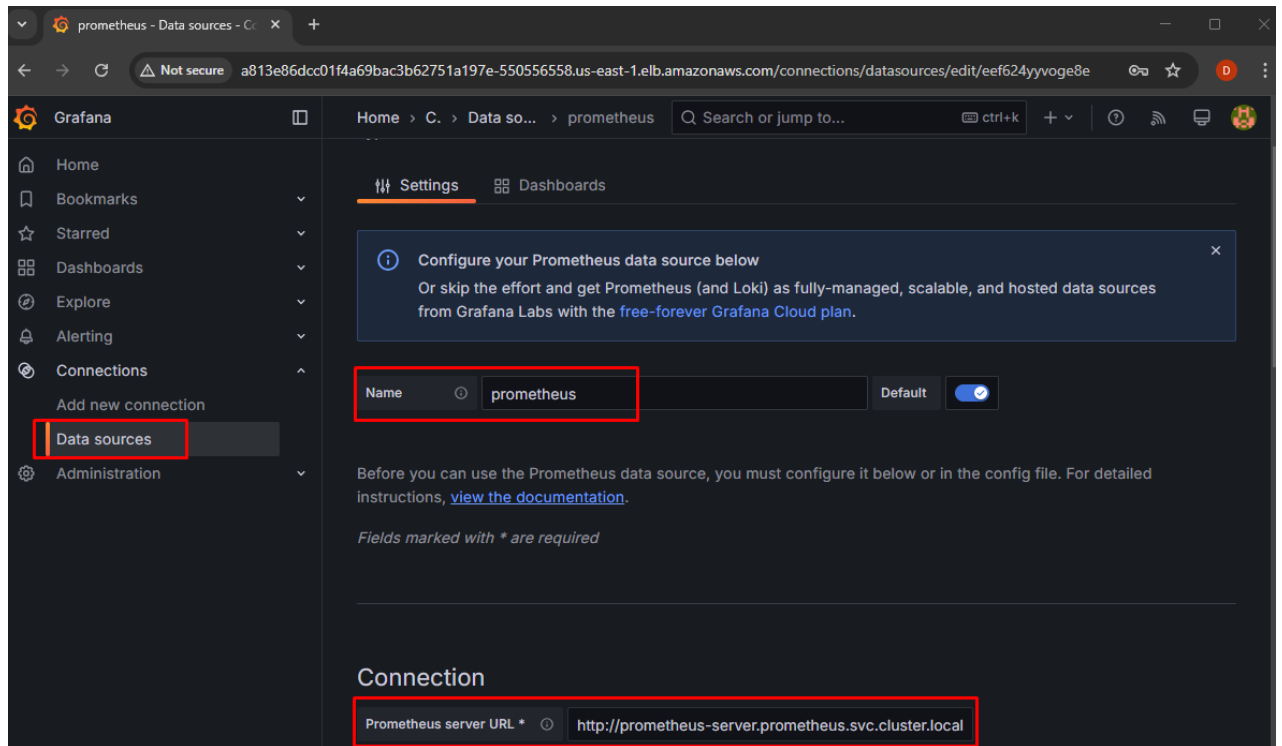
Accedemos a la url del ELB Ingress (<http://a813e86dcc01f4a69bac3b62751a197e-550556558.us-east-1.elb.amazonaws.com/>)



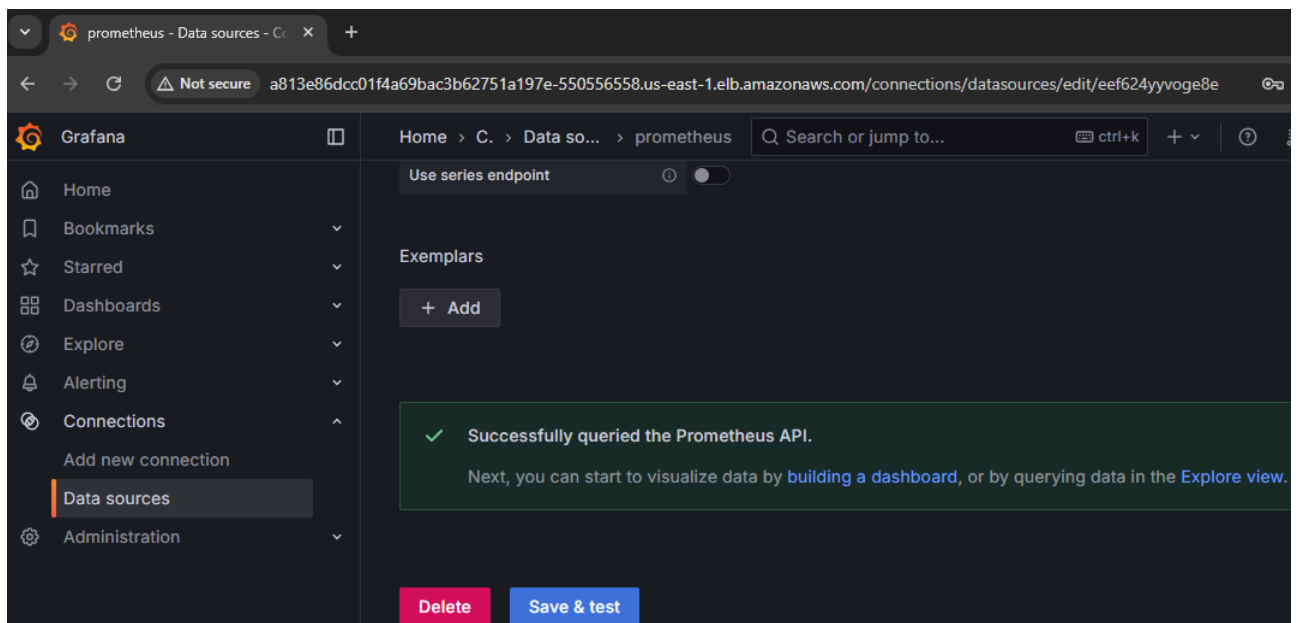
devops 2403	Grupo 1	mundosE
	PIN Final	

Datasource:

Verificamos que el datasource de prometheus esté correctamente configurado y funcionando



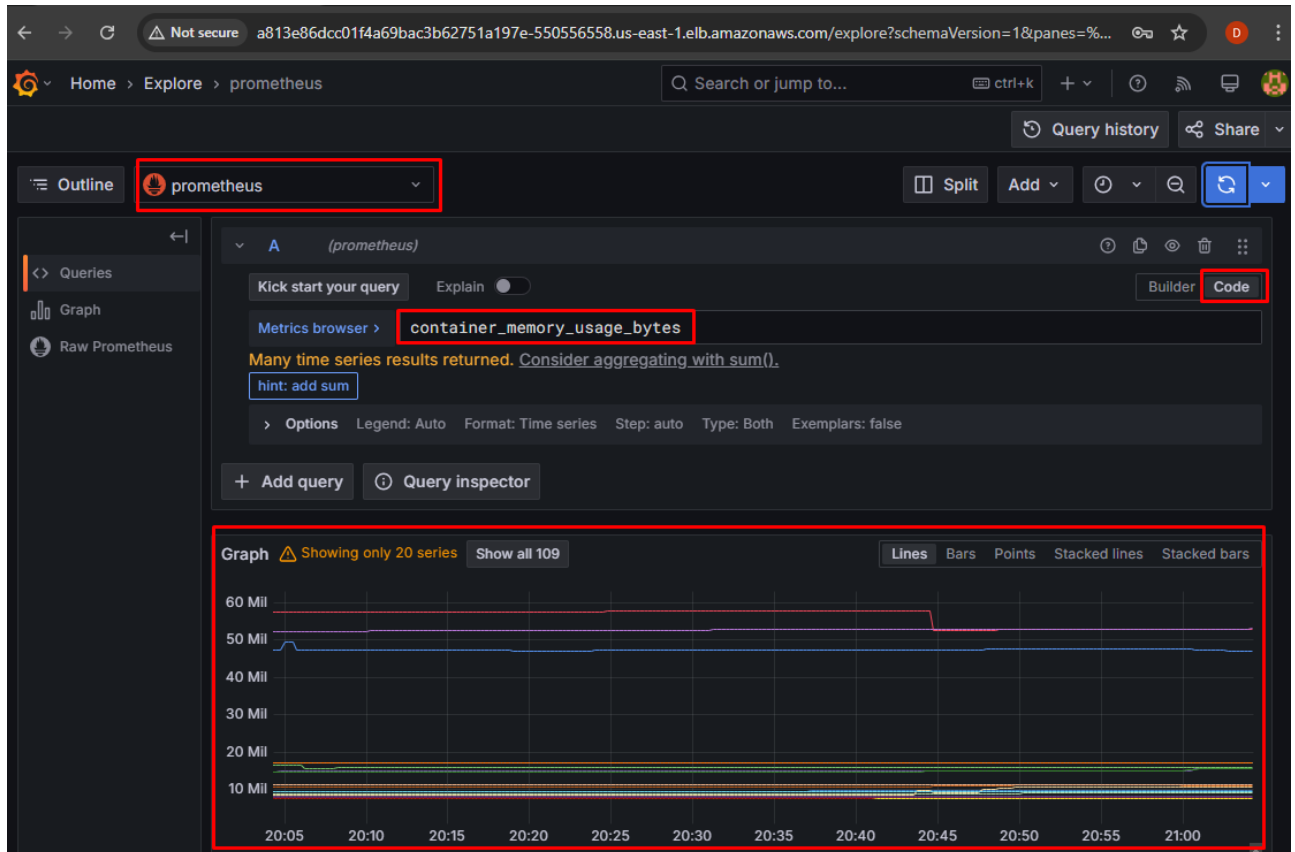
Al presionar el boton “Save & test” vemos que pasa las validaciones y puede consumir correctamente la API de Prometheus



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

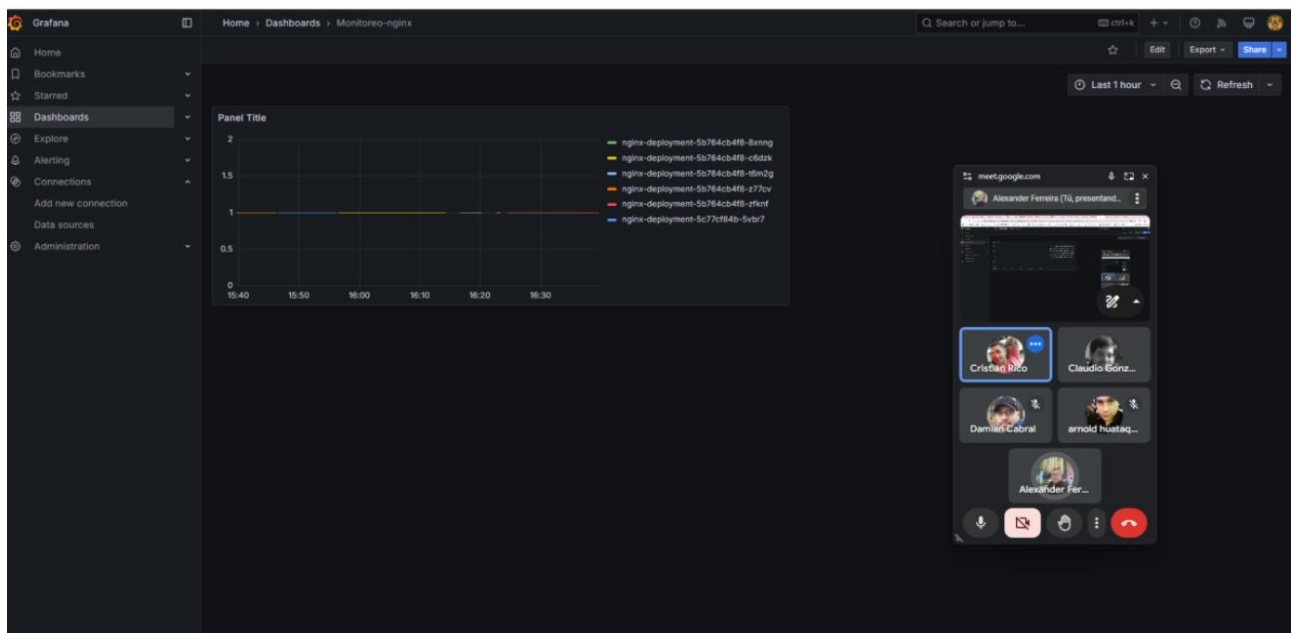
devops 2403	Grupo 1	mundosE
	PIN Final	

Métricas de Prometheus



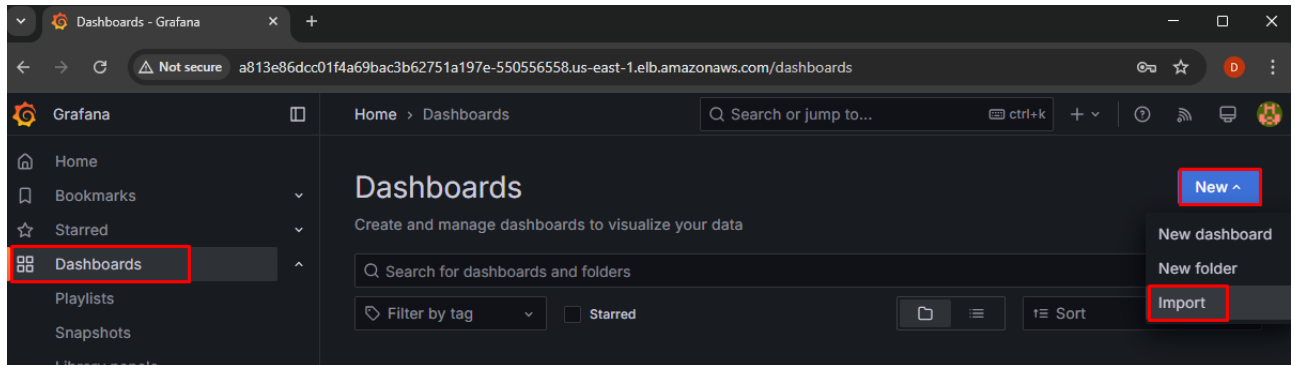
Dashboard

Se verifica que se pueden crear dashboards

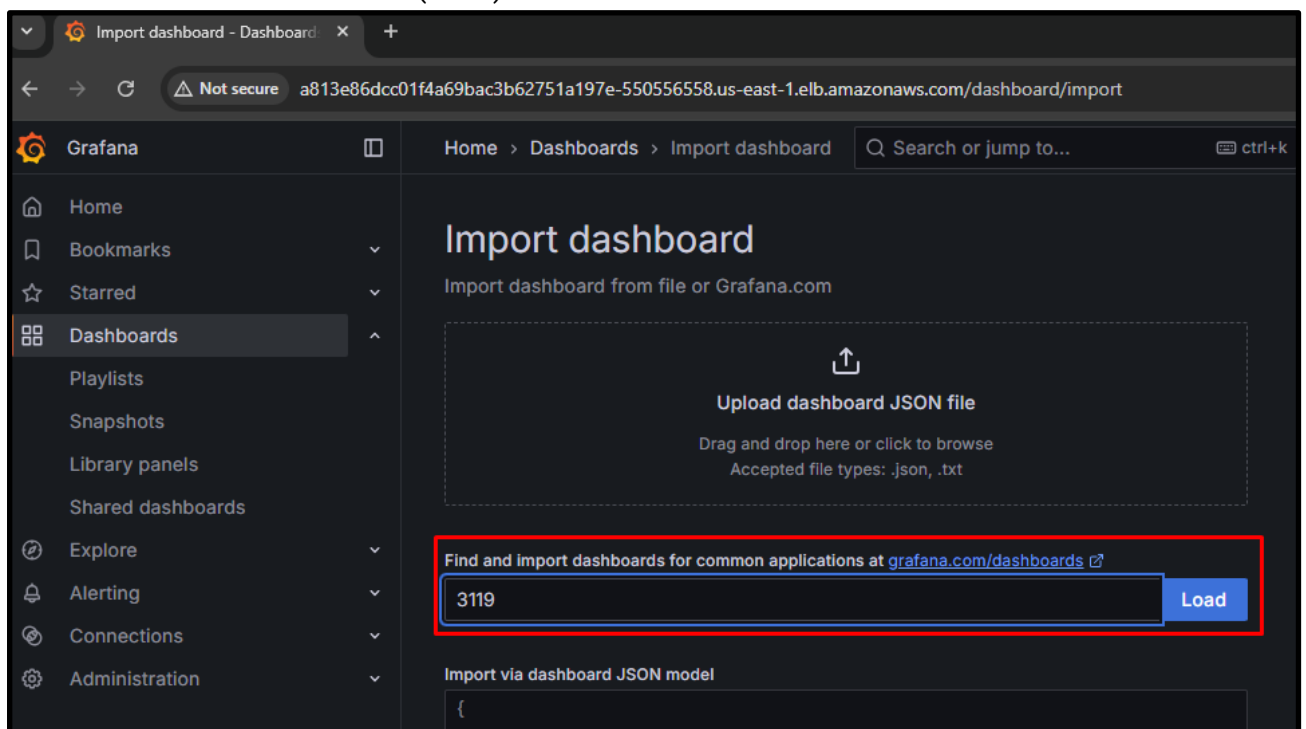


devops 2403	Grupo 1	mundosE
	PIN Final	

Vamos a Dashboards -> New -> Import



Colocamos el id del dashboard (3119)



devops 2403	Grupo 1	mundosE
	PIN Final	

Seleccionamos el datasource del prometheus

Import dashboard - Dashboard: x

Not secure a813e86dcc01f4a69bac3b62751a197e-550556558.us-east-1.elb.amazonaws.com/dashboard/import

Grafana

Home > Dashboards > Import dashboard

Search or jump to... ctrl+k

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.

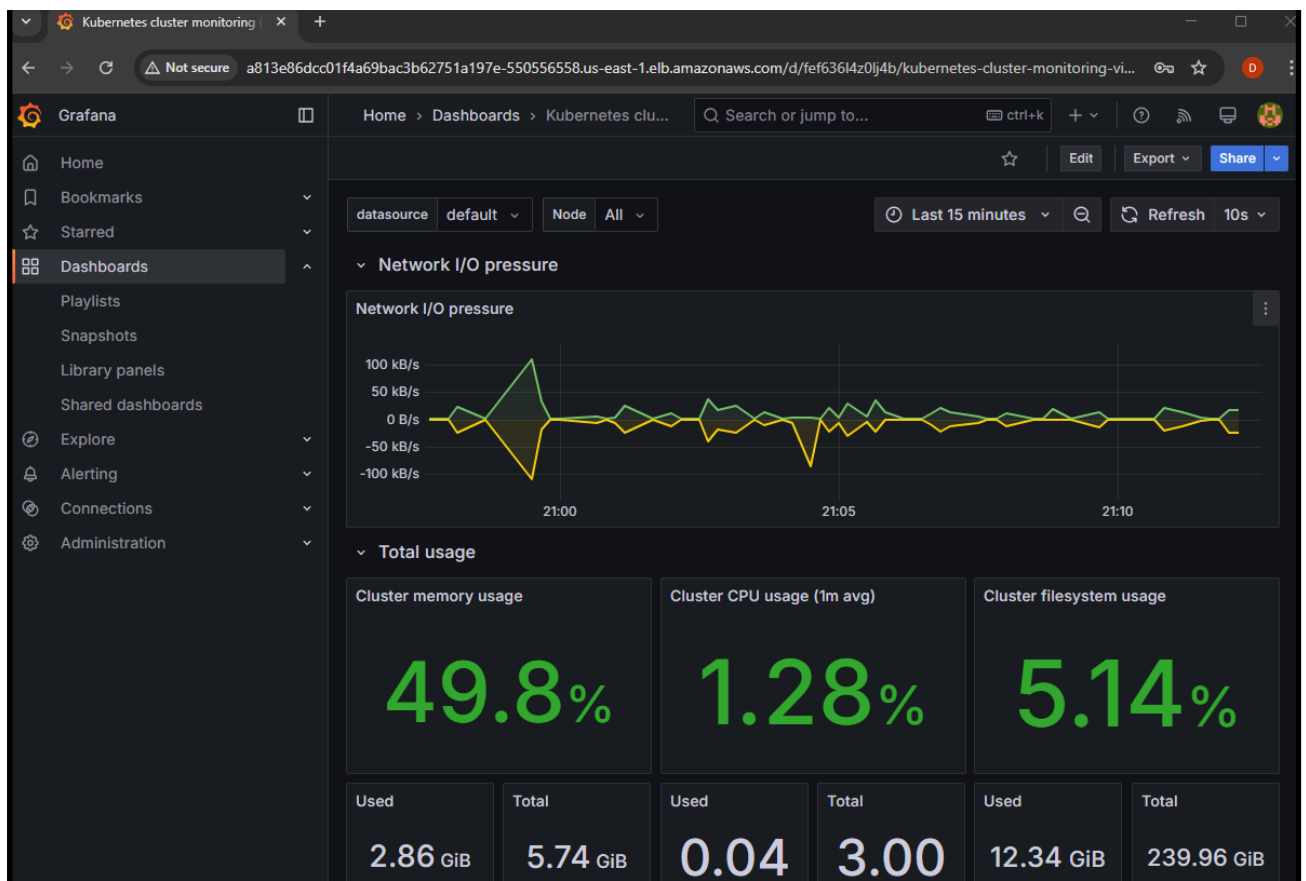
Change uid

prometheus

prometheus

Import Cancel

devops 2403	Grupo 1	mundosE
	PIN Final	



devops 2403	Grupo 1	mundosE
	PIN Final	

CLEAN

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

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

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

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

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

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

devops 2403	Grupo 1	mundosE
	PIN Final	

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.

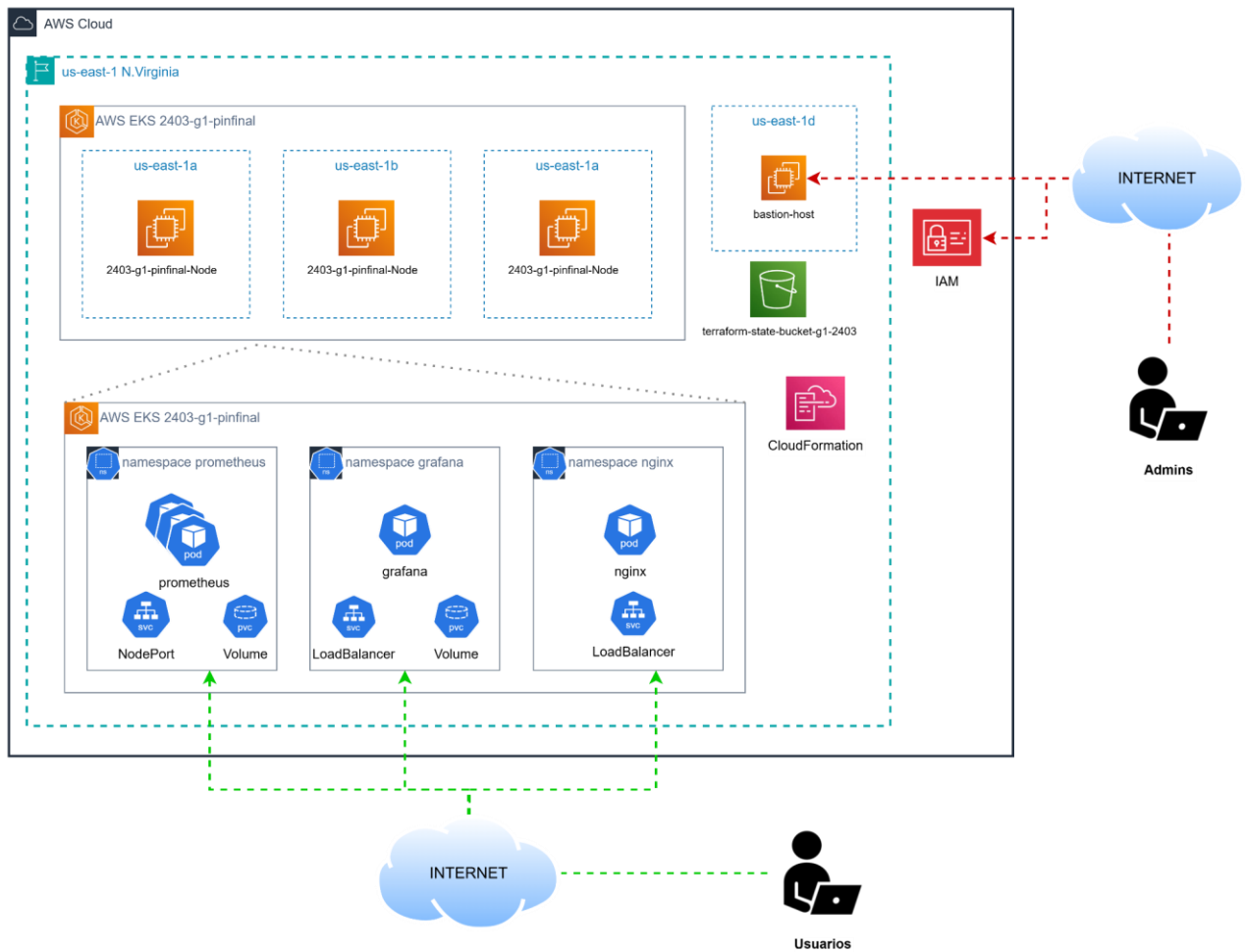
Se resolvieron desafíos en almacenamiento, compatibilidad de versiones y permisos mediante validaciones de storageClass e instalación y actualización de addons.

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	

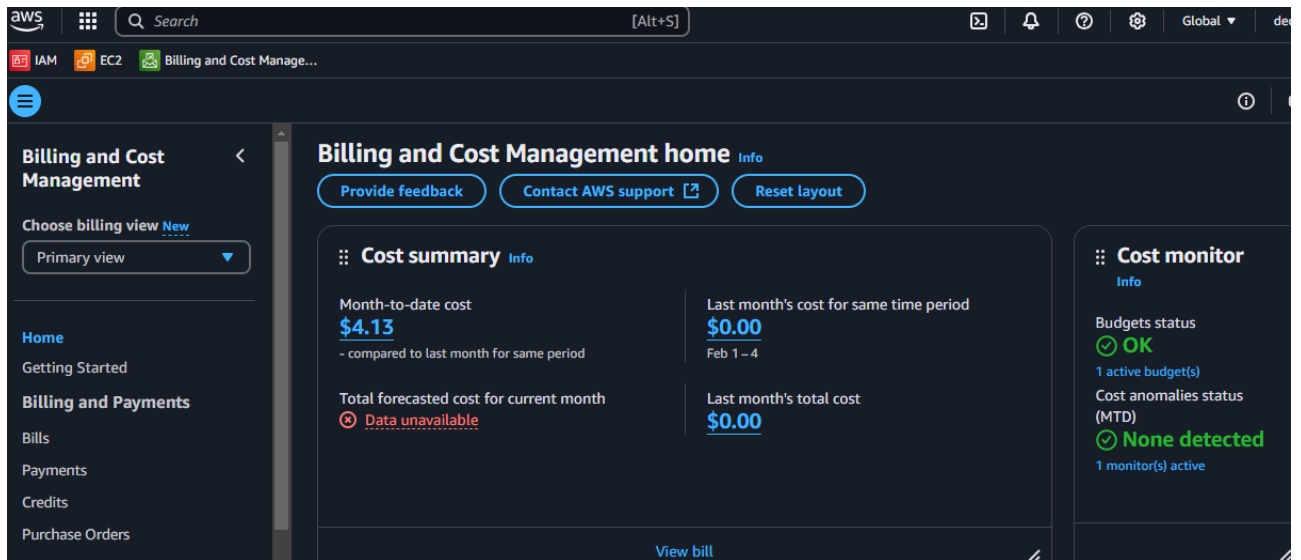
Topología general



devops 2403	Grupo 1	mundosE
	PIN Final	

Consumo

Verificación de los costos consumidos.



The screenshot displays the AWS Billing and Cost Management console. The left sidebar shows the navigation menu with options like 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 a 'Cost summary' section with the following data:

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	

Below the cost summary, there is a 'Cost monitor' section showing the status of budgets and anomalies. It indicates that 1 active budget(s) are present and that no anomalies were detected (None detected). The console also features buttons for 'Provide feedback', 'Contact AWS support', and 'Reset layout'.