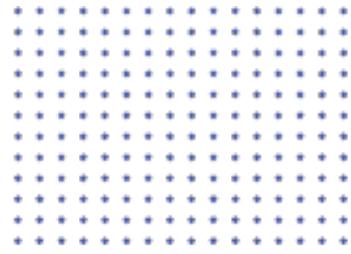


INFORME PIN2

Raúl Benítez Netto / Gabriela Ratti / 2.05.2024



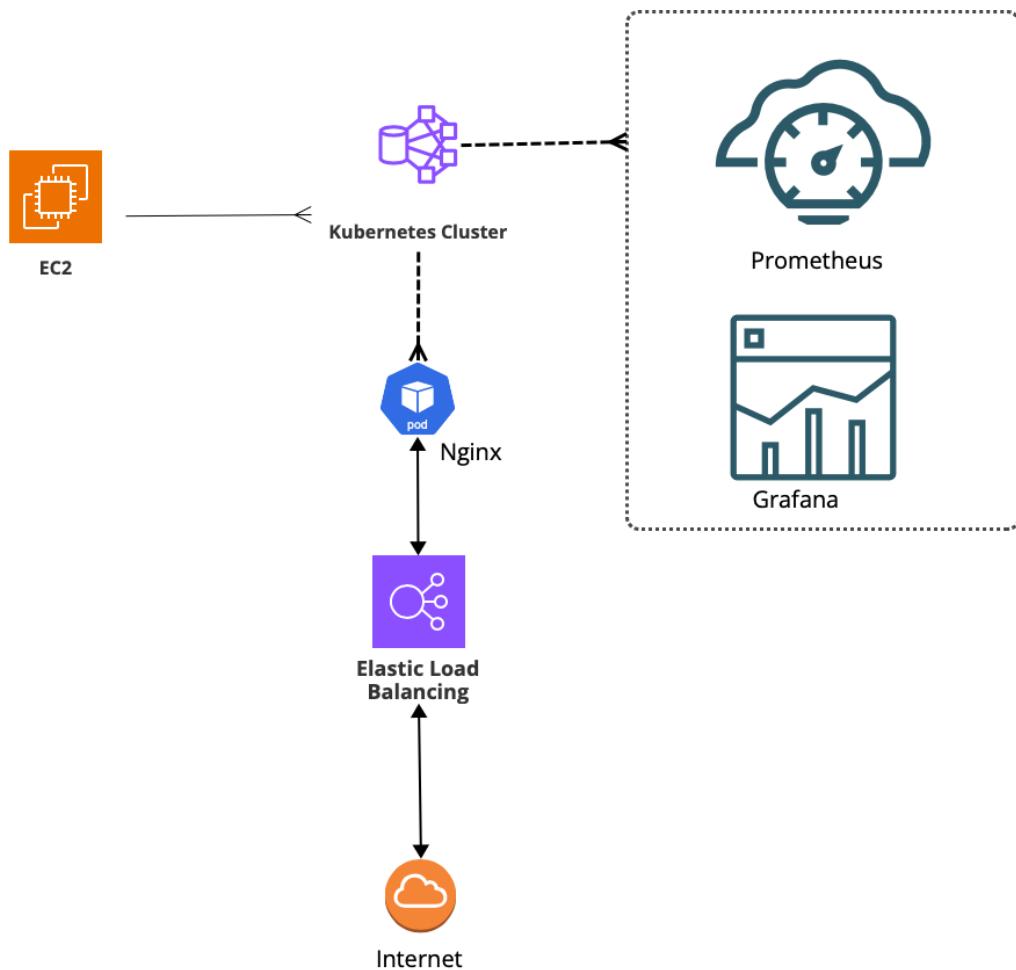
INTRODUCCIÓN



Este proyecto tuvo como idea principal el aprendizaje sobre distintos temas y la puesta en práctica mediante un laboratorio que permita integrar diferentes herramientas y tecnologías. Durante la primera parte nos centramos en la creación de una instancia de EC2 en AWS para poder desde allí realizar todas las tareas necesarias, como montar un cluster Kubernetes, preparar un balanceador de Carga e instalar Nginx además de diversos Pods para monitoreo del cluster con el stack de Prometheus y Grafana.

Para acceder al código y demás documentación dirigirse al siguiente enlace: <https://github.com/Piuliss/mundoes-pin2>

Diagrama diseñado



DESARROLLO

El trabajo de PIN2 (Proyecto Integrador 2) consistió en armar un cluster de Kubernetes en AWS, levantar un pod de NGINX detrás de un balanceador de carga y monitorear los pods con Prometheus y Grafana.

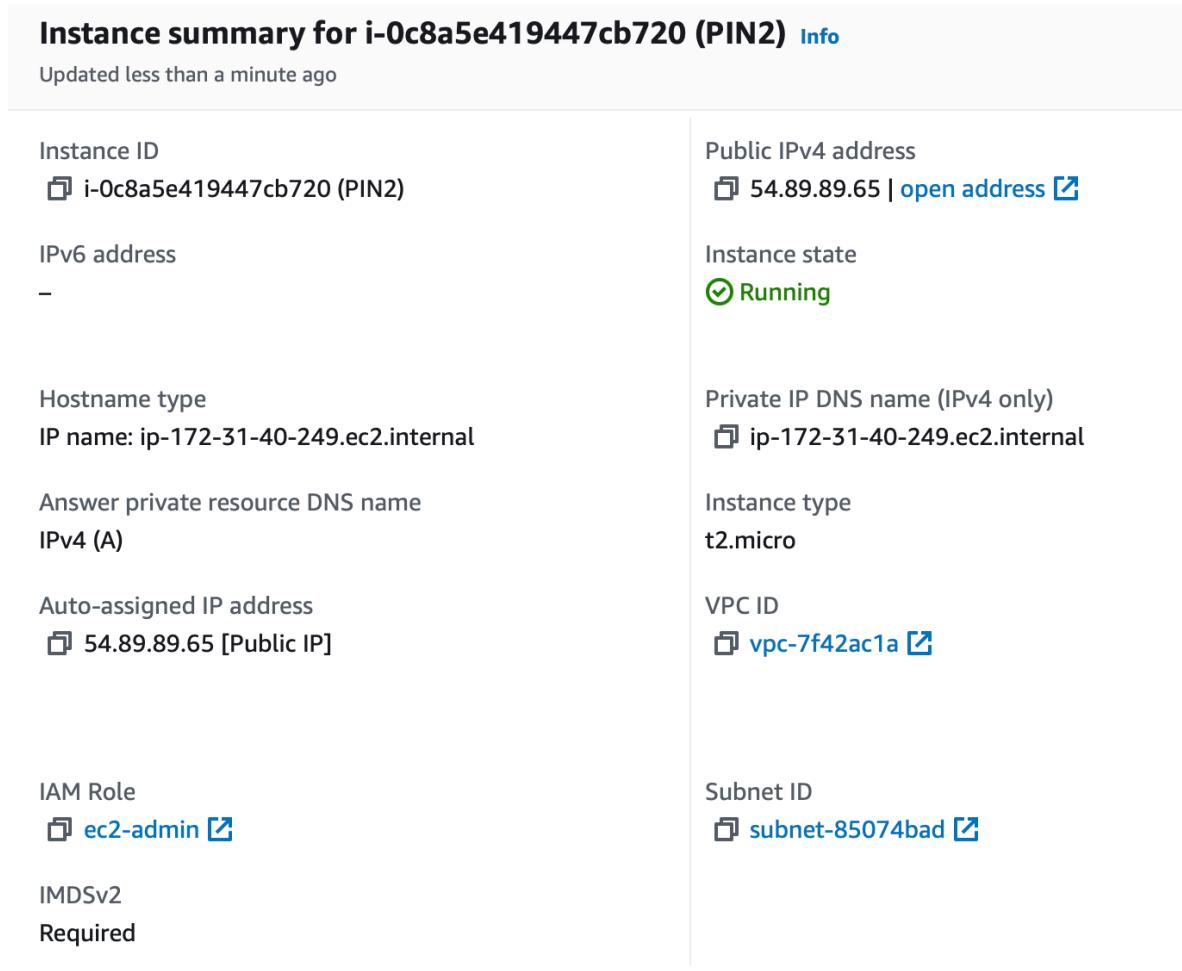
Inicialmente se procedió a crear una instancia EC2 que utilizaremos como Master Node, con la IP pública 54.89.89.65 y con el nombre PIN2

The screenshot shows the AWS EC2 Instances page. On the left, the navigation pane is visible with sections like EC2 Dashboard, EC2 Global View, Events, Console-to-Code, Instances, Images, Elastic Block Store, Network & Security, and more. The main area displays a table of instances. A search bar at the top allows filtering by attribute or tag. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. One instance, 'PIN2' (Instance ID i-0c8a5e419447cb720), is selected and highlighted with a blue border. Below the table, a detailed view for 'i-0c8a5e419447cb720 (PIN2)' is shown. This view includes tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. Under the Details tab, the 'Instance summary' section provides information such as Public IPv4 address (54.89.89.65), Instance state (Running), and Private IP DNS name (ip-172-31-40-249.ec2.internal). The 'Instance details' section lists various configuration parameters like Platform (Ubuntu), AMI ID (ami-080e1f13689e07408), Launch time (Thu Apr 25 2024 11:20:45 GMT-0400), and so on. The 'Monitoring' section indicates that monitoring is disabled.

En AWS, creamos un rol llamado **ec2-admin** con todos los permisos administrador asociados a él.

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with navigation links like Dashboard, Access management, Roles, and Access reports. The main area shows the 'ec2-admin' role details. The 'Summary' tab is selected, displaying creation date (April 25, 2024), ARN (arn:aws:iam::278229051818:role/ec2-admin), last activity (51 minutes ago), and maximum session duration (1 hour). Below the summary, there are tabs for Permissions, Trust relationships, Tags (3), Access Advisor, and Revoke sessions. The 'Permissions' tab is active, showing one policy attached: 'AdministratorAccess' (AWS managed - job function). There are buttons for Simulate, Remove, and Add permissions.

Asociamos dicho rol a la instancia EC2 creada PIN2:



The screenshot shows the 'Instance summary for i-0c8a5e419447cb720 (PIN2)' page. It provides detailed information about the instance, including its ID, IP addresses, IAM role, and VPC details. Key details include:

- Instance ID:** i-0c8a5e419447cb720 (PIN2)
- Public IPv4 address:** 54.89.89.65 | [open address](#)
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-172-31-40-249.ec2.internal
- Instance type:** t2.micro
- VPC ID:** vpc-7f42ac1a
- IAM Role:** ec2-admin
- Subnet ID:** subnet-85074bad
- IMDSv2:** Required

Una vez lista la instancia, nos conectamos a ella por SSH.



PIN2 | 5



```
→ PIN2 ssh -i "raulbeni-aws.pem" ubuntu@ec2-54-89-89-65.compute-1.amazonaws.com
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1018-aws x86_64)

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

System information as of Tue Apr 30 23:16:19 UTC 2024

System load: 0.0          Processes:           105
Usage of /: 16.2% of 28.89GB Users logged in:   1
Memory usage: 30%          IPv4 address for docker0: 172.17.0.1
Swap usage:  0%          IPv4 address for eth0:   172.31.40.249

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Apr 30 22:37:26 2024 from 186.158.200.126
ubuntu@ip-172-31-40-249:~$
```

Dentro de la EC2 escalamos privilegios a root. Procedemos a instalar todas las herramientas necesarias con el siguiente script:

- AWS CLI
- Kubectl
- Ekctl
- Docker
- HELM
- Terraform

```

1 #!/bin/bash
2
3 echo "Installing AWS CLI"
4 apt install unzip -y
5 curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
6 unzip awscliv2.zip
7 sudo ./aws/install
8
9 echo "Installing kubectl"
10 curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.2/2023-03-17/bin/linux/amd64/kubectl
11 chmod +x ./kubectl
12 mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
13 echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
14 kubectl version --client
15
16 echo "Installing eksctl"
17 # Download EKS CLI https://github.com/weaveworks/eksctl
18 curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_a
19 sudo mv /tmp/eksctl /usr/local/bin
20 export PATH=$PATH:/usr/local/bin
21 echo 'export PATH=$PATH:/usr/local/bin' >> ~/.bashrc
22 eksctl version
23
24 echo "Installing docker"
25 sudo yum install -y docker
26 sudo usermod -a -G docker ec2-user
27 newgrp docker
28 wget https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)
29 sudo mv docker-compose-$(uname -s)-$(uname -m) /usr/local/bin/docker-compose
30 sudo chmod -v +x /usr/local/bin/docker-compose
31 sudo systemctl enable docker.service
32 sudo systemctl start docker.service
33
34 echo "Installing Helm"
35 curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
36 chmod 700 get_helm.sh
37 ./get_helm.sh
38
39 echo "Installing terraform "
40 sudo yum install -y yum-utils
41 sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
42 sudo yum install -y terraform
43

```

Una vez las herramientas listas y verificadas, procedemos a crear el cluster Kubernetes.

```

eksctl create cluster \
--name mundose-eks \
--region us-east-1 \
--node-type t2.small \
--with-oidc \
--ssh-access \
--ssh-public-key pin \
--managed \
--full-ecr-access \
--zones us-east-1a,us-east-1b,us-east-1c

```

Verificamos si el cluster se creo correcxtamente en el Amazon Elastic Kubernetes Services

Amazon Elastic Kubernetes Service

Clusters [New](#)

Amazon EKS Anywhere

Enterprise Subscriptions [New](#)

Related services

Amazon ECR

AWS Batch

Documentation [View](#)

Submit feedback

① Extended support for Kubernetes versions pricing
New prices for extended support will start in the April billing cycle. For more information, see the blog post [\[?\]](#).

Notifications [0](#) [0](#) [0](#) [3](#) [0](#) [▼](#)

EKS > Clusters

① New Kubernetes versions are available for 1 cluster.

Clusters (1) [Info](#)

[C](#)

[Delete](#)

[Add cluster ▾](#)

[Filter clusters](#)

< 1 >

Cluster name	Status	Kubernetes version	Support period	Provider
mundose-eks-wPKpsWPv	Active	1.26 Upgrade now	Standard support until June 11, 2024	EKS

EKS > Clusters > mundose-eks-wPKpsWPv

mundose-eks-wPKpsWPv

[C](#)

[Delete cluster](#)

[Upgrade version](#)

⚠ Your cluster's Kubernetes version (1.26) will reach the end of standard support on June 11, 2024. On that date, your cluster will enter the extended support period with additional fees. For more information, see the [pricing page \[?\]](#). [Learn more \[?\]](#)

[Upgrade now](#)

① New AMI release versions are available for 1 node group. [Learn more \[?\]](#)

▼ Cluster info [Info](#)

Status

[Active](#)

Kubernetes version [Info](#)

1.26

Support period

[Standard support until June 11, 2024](#)

Provider

EKS

Volumes (6) Info						
Create volume						
Search						
Name	Volume ID	Type	Size	IOPS	Throughput	
worker-group-1	vol-0a33ae25786802ec9	gp2	20 GiB	100	-	Actions ▾
mundose-eks-wPKpsWPv-dynamic-pvc-6a2ac5d1-5823-47c4-ba73-6c2d697f1a50	vol-0e20532eda0b8ec31	gp2	2 GiB	100	-	Actions ▾
mundose-eks-wPKpsWPv-dynamic-pvc-c513484c-bee0-4c2b-92e1-b9f488de52d1	vol-0d899fab6754ededa	gp2	8 GiB	100	-	Actions ▾
mundose-eks-wPKpsWPv-dynamic-pvc-be025e86-0351-4abf-97ba-263c1a0e496b	vol-0724689ba54ee2d8e	gp2	10 GiB	100	-	Actions ▾
-	vol-033ffe8c170ac56b3	gp2	30 GiB	100	-	Actions ▾
worker-group-1	vol-07e39bb38ac58ec33	gp2	20 GiB	100	-	Actions ▾

Nginx

Una vez el cluster esta listo procedemos a instalar un pod de Nginx usando el siguiente yaml:

```
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3 # tells deployment to run 1 pods matching the template
  template: # create pods using pod definition in this template
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  externalTrafficPolicy: Local
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

También se generará un LoadBalancer asociado al pod del nginx.

The screenshot shows the AWS CloudShell interface with the AWS Management Console open. The user is navigating through the AWS Services menu, specifically under Load Balancing, to view a Network Load Balancer named 'a008572ce8c5144a5ab93d980d79ab61'. The screenshot shows the load balancer's configuration details, including its type (Classic), scheme (Internet-facing), and associated VPC, subnets, and availability zones.

Load balancer type	Status	VPC	Date created
Classic	2 of 2 instances in service	vpc-046e92de4420fc63c	April 25, 2024, 19:45 (UTC-04:00)
Scheme	Hosted zone	Availability Zones	
Internet-facing	Z35SXDOTRQ7X7K	subnet-05a0232b6ede08d0 us-east-1a (use1-az6) subnet-0660636f649bf053b us-east-1b (use1-az1) subnet-079a21467b9b11536 us-east-1c (use1-az2)	

A screenshot of a web browser window. The address bar shows a shield icon and the URL "Not Secure — 3c5144a5ab93d980d79ab61-2116631509.us-east-1.elb.amazonaws.com". Below the address bar is a navigation bar with various links: "To READ", "HomeLab", "BlogsTec", "Translators", "Online Tools", "Projects", "Papers", "UNI", "projects", "TRAVEL", "CERT-PY", and "GripPad". Underneath the navigation bar are several icons: a red 'M', a blue square, a green square, a blue gear, a blue cloud, a blue gear with a plus sign, a blue gear with a minus sign, a blue gear with a question mark, a blue gear with a checkmark, a blue gear with a refresh symbol, and a blue gear with a gear inside. The main content area features a large white "Welcome to nginx!" title. Below it is a message: "If you see this page, the nginx web server is successfully installed and working. Further configuration is required." Another message follows: "For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com". A final message at the bottom says: "Thank you for using nginx."

Prometheus

Luego procedemos a crear el pod de prometheus en el nodo master usando helm:

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
kubectl create namespace prometheus
helm install prometheus prometheus-community/Prometheus \
--namespace Prometheus \
--set alertmanager.persistentVolume.storageClass="gp2" \
--set server.persistentVolume.storageClass="gp2"
```

Pero el mismo no logrará funcionar correctamente, no todas las replicas se levantan de forma correcta, procedemos a realizar unos cambios en el **AWS EBS**:

1. Creamos un IAM Service Account asociado al cluster

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

2. Creamos un addon

```
eksctl create addon --name aws-ebs-csi-driver \
--cluster mundose-eks-wPKpsWPv \
--service-account-role-arn \
arn:aws:iam::278229051818:role/AmazonEKS_EBS_CSI_DriverRole -force
```

Verificamos que todas las replicas funcionen correctamente:

```
***** PIN2 | II *****
```

```

root@ip-172-31-40-249:/home/ubuntu/PIN2# kubectl get all -n prometheus
NAME                                         READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                1/1    Running   0          4d23h
pod/prometheus-kube-state-metrics-59bfcd7b65-mc2lw 1/1    Running   0          4d23h
pod/prometheus-prometheus-node-exporter-74tgh  1/1    Running   0          4d23h
pod/prometheus-prometheus-node-exporter-p6v84  1/1    Running   0          4d23h
pod/prometheus-prometheus-pushgateway-546f8b488f-clhmb 1/1    Running   0          4d23h
pod/prometheus-server-6c5d4f7cdf-r4fmx        2/2    Running   0          4d23h

NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/prometheus-alertmanager       ClusterIP   172.20.109.171 <none>           9093/TCP     4d23h
service/prometheus-alertmanager-headless ClusterIP   None            <none>           9093/TCP     4d23h
service/prometheus-kube-state-metrics  ClusterIP   172.20.115.81  <none>           8080/TCP     4d23h
service/prometheus-prometheus-node-exporter ClusterIP   172.20.19.233 <none>           9100/TCP     4d23h
service/prometheus-prometheus-pushgateway ClusterIP   172.20.77.98  <none>           9091/TCP     4d23h
service/prometheus-server             ClusterIP   172.20.98.188 <none>           80/TCP        4d23h

NAME              DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
daemonset.apps/prometheus-node-exporter  2         2        2      2           2          kubernetes.io/os=linux  4d23h

NAME          READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/prometheus-kube-state-metrics  1/1      1          1          4d23h
deployment.apps/prometheus-prometheus-pushgateway 1/1      1          1          4d23h
deployment.apps/prometheus-server               1/1      1          1          4d23h

NAME          DESIRED  CURRENT  READY  AGE
replicaset.apps/prometheus-kube-state-metrics-59bfcd7b65 1         1          1          4d23h
replicaset.apps/prometheus-prometheus-pushgateway-546f8b488f 1         1          1          4d23h
replicaset.apps/prometheus-server-6c5d4f7cdf            1         1          1          4d23h

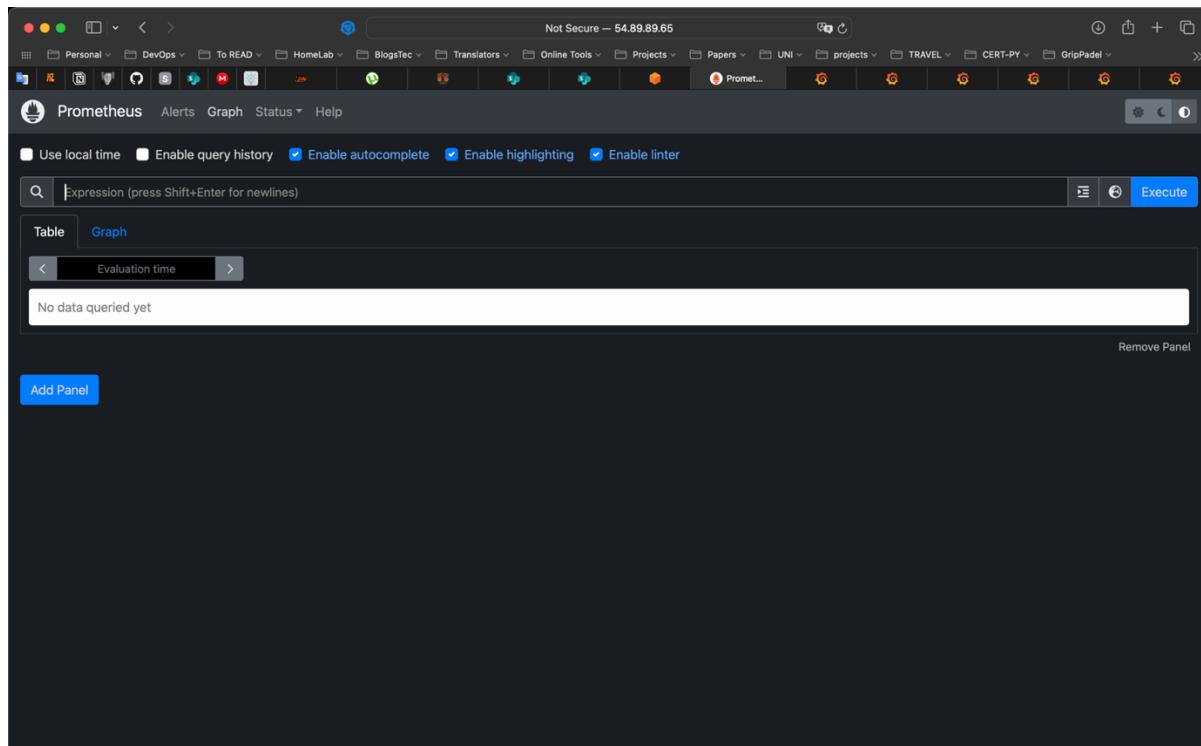
NAME          READY  AGE
statefulset.apps/prometheus-alertmanager  1/1   4d23h
root@ip-172-31-40-249:/home/ubuntu/PIN2#

```

hacemos un port forward del Prometheus

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

Habilitamos el Puerto 8080 en nuestro Group Security del AWS y verificamos la URL <http://54.89.89.65:8080>:



PIN2 | 12

Grafana

Posteriormente configuramos un Grafana,

```
kubectl create namespace grafana

helm install mygrafana grafana/grafana -n grafana

--set persistence.storageClassName="gp2" \
--set persistence.enabled=true \
--set adminPassword='Uhxte32TLMa7zIDW' \
--values grafana.yaml \
--set services.type=LoadBalancer
```

Verificamos que el pod este funcionando correctamente:

```
root@ip-172-31-40-249:/home/ubuntu/PIN2# kubectl get all -n grafana
NAME                           READY   STATUS    RESTARTS   AGE
pod/mygrafana-64c7df7446-6p85b   1/1     Running   0          72m

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/mygrafana   ClusterIP   172.20.116.13  <none>        80/TCP     72m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mygrafana   1/1     1           1           72m

NAME                           DESIRED  CURRENT  READY   AGE
replicaset.apps/mygrafana-64c7df7446  1        1        1       72m
root@ip-172-31-40-249:/home/ubuntu/PIN2#
```

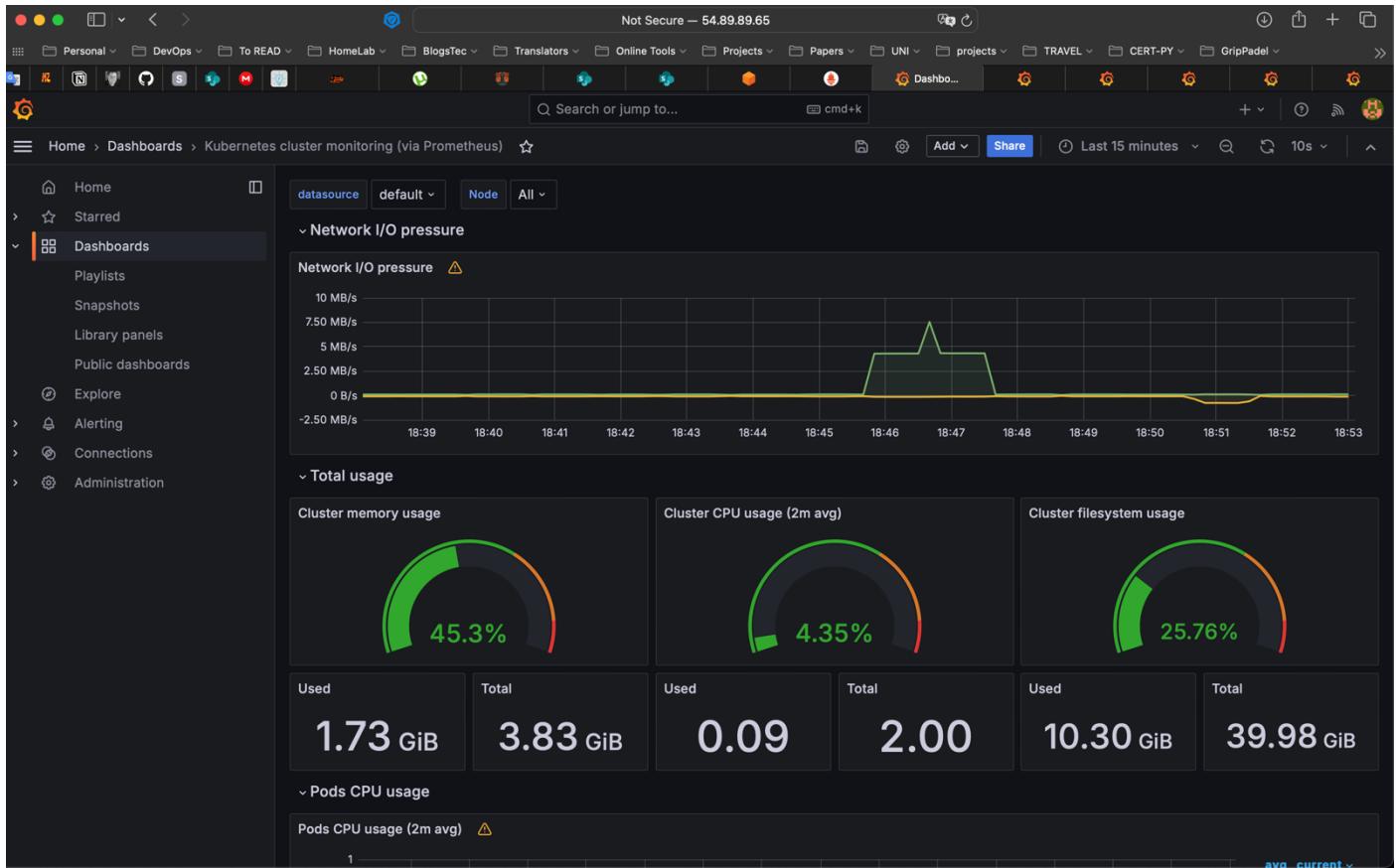
Extraemos la contraseña

```
root@ip-172-31-40-249:/home/ubuntu/PIN2# kubectl get secret --namespace grafana mygrafana -o jsonpath=".data.admin-password" | base64 --decode ; echo
Uhxte32TLMa7zIDW
root@ip-172-31-40-249:/home/ubuntu/PIN2#
```

y luego hacemos un port-forward para ver la instancia de Grafana

```
root@ip-172-31-40-249:/home/ubuntu/PIN2# kubectl get svc -n grafana mygrafana -o jsonpath=".status.loadBalancer.ingress[0].ip"
{}root@ip-172-31-40-249:/home/ubuntu/PIN2# export POD_NAME=$(kubectl get pods --namespace grafana -l "app.kubernetes.io/name=grafana,app.kubernetes.io/namespace=grafana" -o yaml | grep status.phase:Running | awk '{print $1}')
root@ip-172-31-40-249:/home/ubuntu/PIN2# kubectl port-forward -n grafana $POD_NAME 3000:3000 --address 0.0.0.0
Forwarding from 0.0.0.0:3000 -> 3000
root@ip-172-31-40-249:/home/ubuntu/PIN2#
```

Habilitamos el puerto 3000 para el protocolo TCP en el grupo de seguridad de AWS y nos dirigimos a la ip <http://54.89.89.65:3000> y configuramos el dashboard 3119



Clean up

Procedimos a limpiar todo

```
root@ip-172-31-40-249:/home/ubuntu# helm uninstall prometheus --namespace prometheus
release "prometheus" uninstalled
root@ip-172-31-40-249:/home/ubuntu# kubectl delete ns prometheus
namespace "prometheus" deleted
root@ip-172-31-40-249:/home/ubuntu# helm uninstall grafana --namespace grafana
Error: uninstall: Release not loaded: grafana: release: not found
root@ip-172-31-40-249:/home/ubuntu# helm uninstall mygrafana --namespace grafana
release "mygrafana" uninstalled
root@ip-172-31-40-249:/home/ubuntu# kubectl delete ns grafana
namespace "grafana" deleted
root@ip-172-31-40-249:/home/ubuntu# rm -rf ${HOME}/environment/grafana
root@ip-172-31-40-249:/home/ubuntu#
```

Borramos el cluster con el comando

```
eksctl delete cluster --name mundose-eks-wPKpsWPv
```

ANEXO

Para completar este trabajo utilizamos el siguiente security-group

Details					
Security group name launch-wizard-6	Security group ID sg-032ec78b0a403ddf1	Description launch-wizard-6 created 2024-04-25T15:17:48.634Z	VPC ID vpc-7f42ac1a		
Owner 278229051818	Inbound rules count 5 Permission entries	Outbound rules count 1 Permission entry			

[Inbound rules](#) [Outbound rules](#) [Tags](#)

Inbound rules (5)					
<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol
<input type="checkbox"/>	-	sgr-0e70ccc0d7e667c74	IPv4	Custom TCP	TCP
<input type="checkbox"/>	-	sgr-0308d6f9a1e461d73	IPv4	SSH	TCP
<input type="checkbox"/>	-	sgr-0e91729b83fb093...	IPv4	HTTPS	TCP
<input type="checkbox"/>	-	sgr-0eb49ebe852e300...	IPv4	Custom TCP	TCP
<input type="checkbox"/>	-	sgr-0f961d748cae180a4	IPv4	HTTP	TCP

Tambien para ello necesitamos generar una clave pin.pem para construir el cluster

[Key pairs \(3\)](#) [Info](#)

[Create key](#)

Key pairs				
<input type="checkbox"/>	Name	Type	Created	Fingerprint
<input type="checkbox"/>	pin	rsa	2024/04/25 15:50 GMT-4	45:c5:62:91:98:7c:7b:98:2e:7a:f3:32:... ID key-038056f5a8e49f7cc

PIN2 | 15
