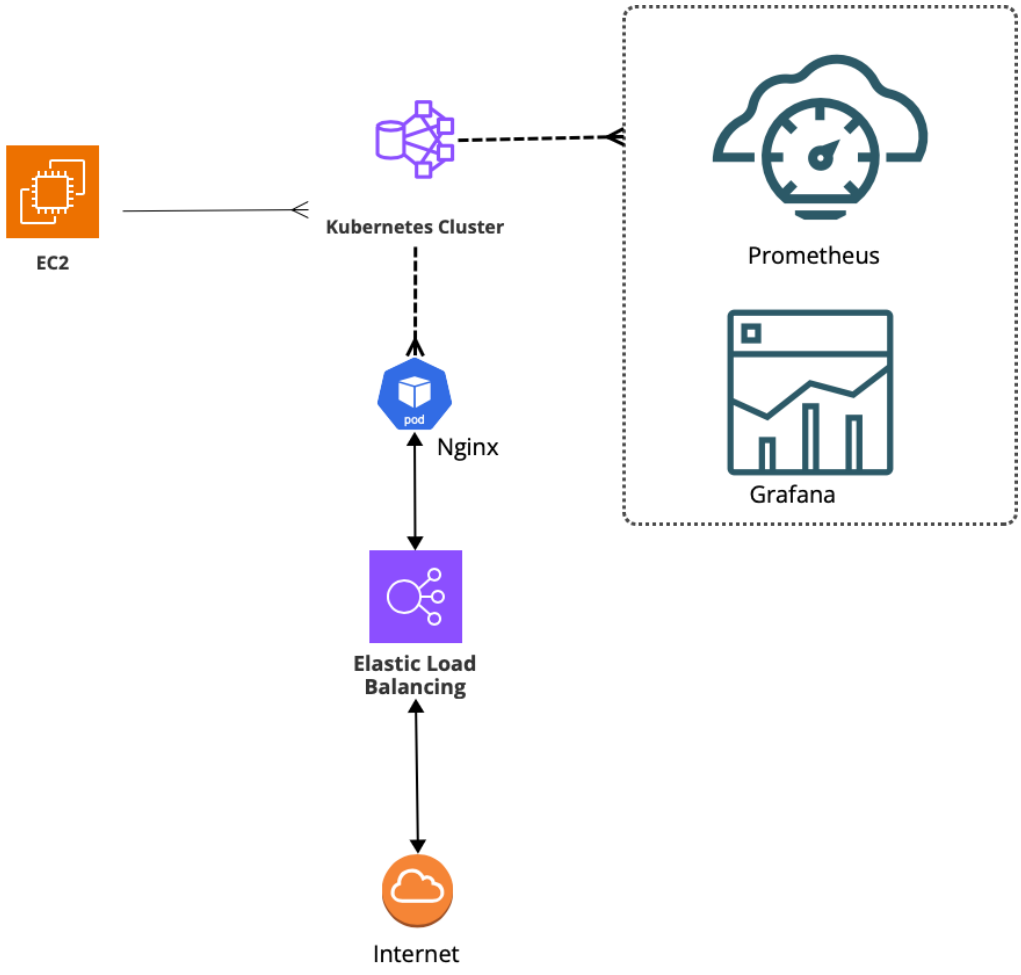
INFORME PIN2

Raúl Benitez 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 demas documentación dirigirse al siguiente enlace: https://github.com/Piuliss/mundoes-pin2

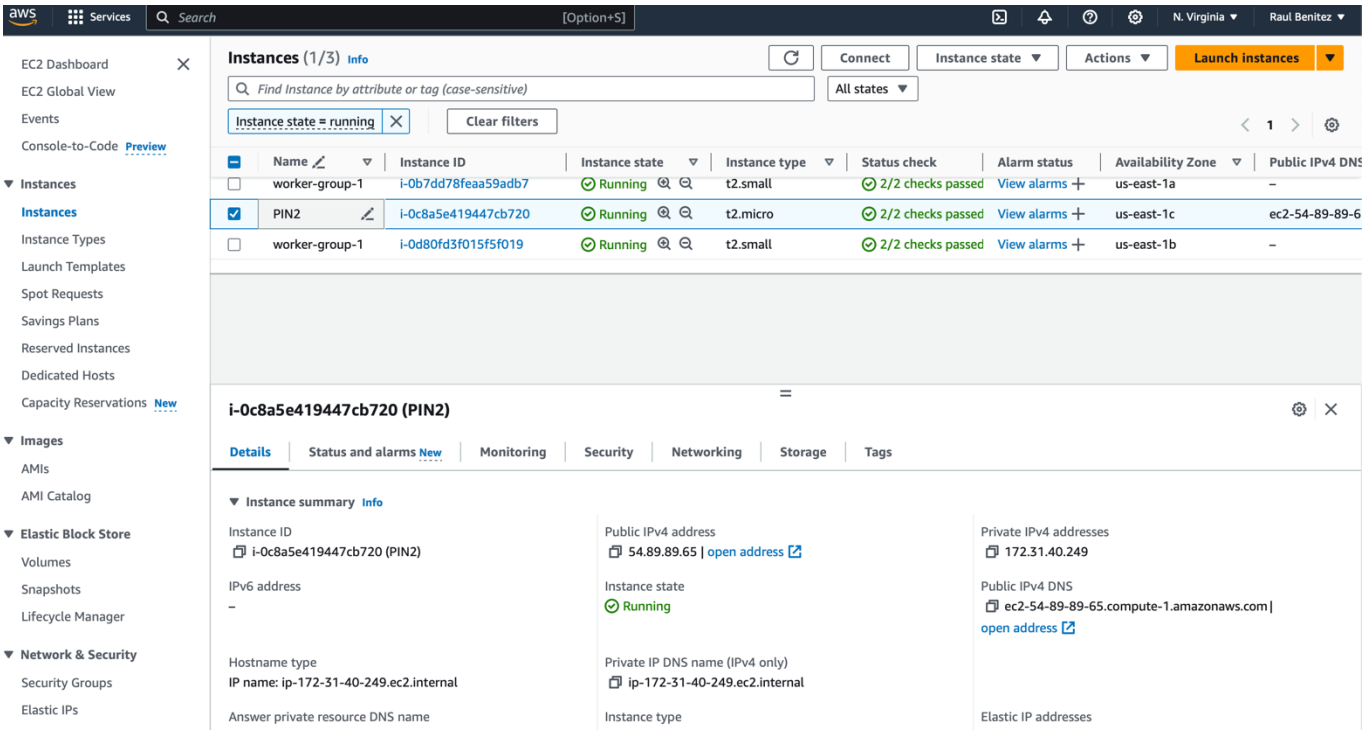
Diagrama diseñado



DESARROLLO

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

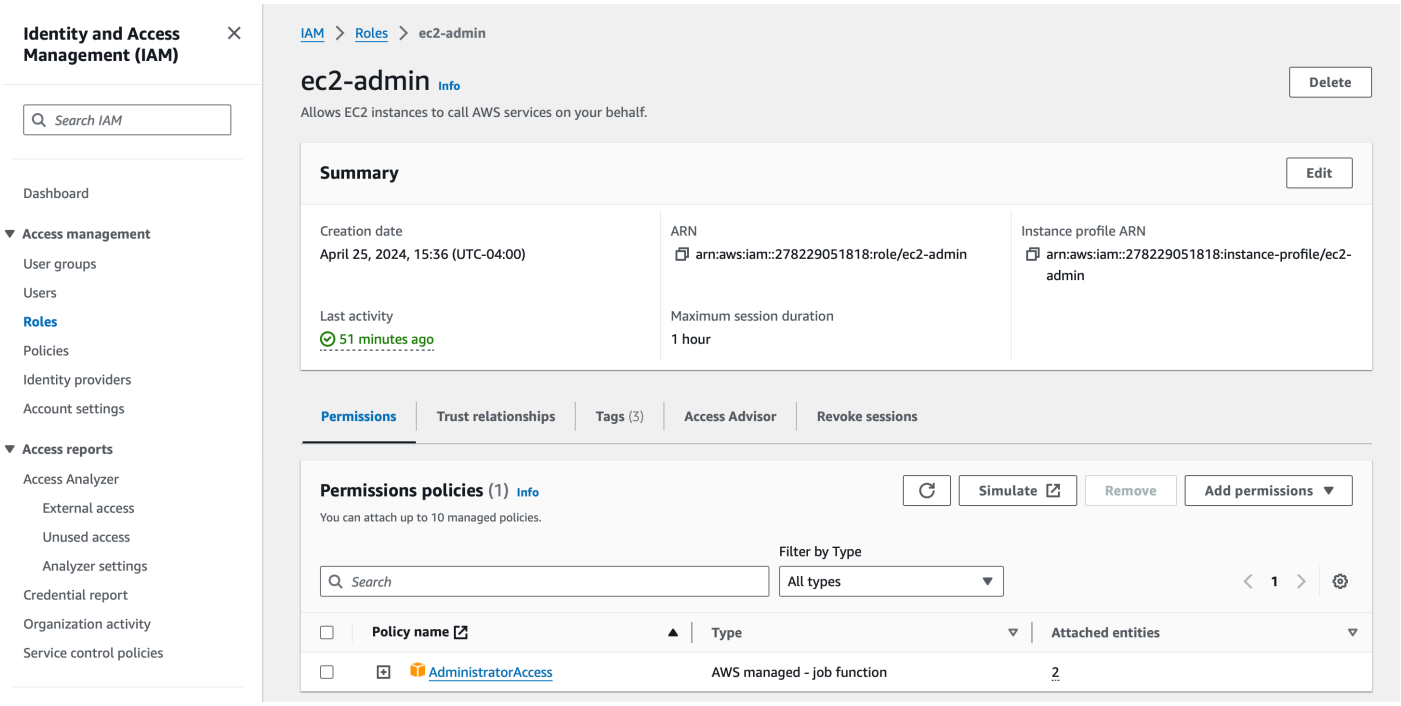
Inicialmente se procedió a crear una instancia EC2 que utilizaremos como Master Node, con la IP publica 54.89.89.65 y con el nombre PIN2



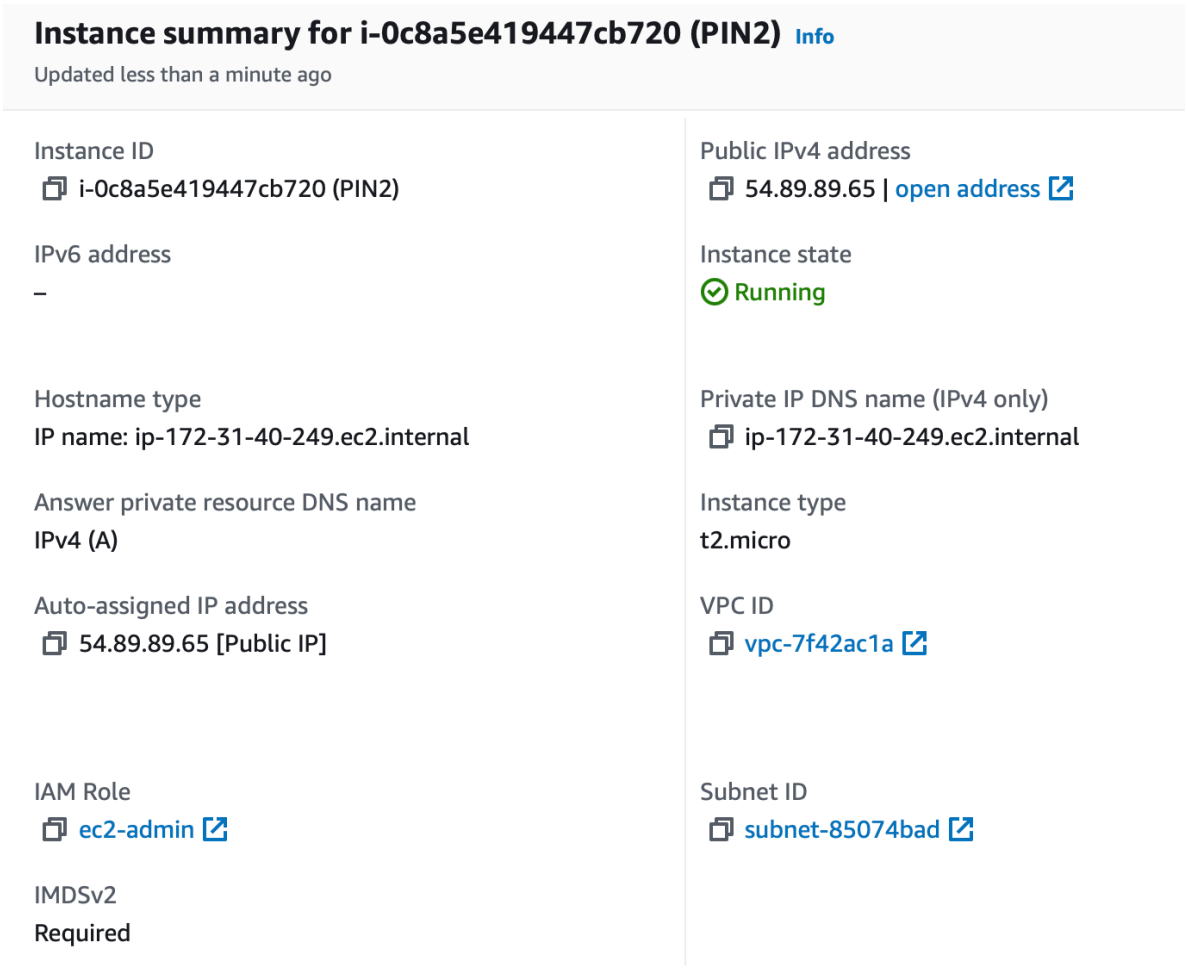
Instance details Info

Platform Ubuntu (Inferred)	AMI ID ami-080e1f13689e07408	Monitoring disabled
Platform details Linux/UNIX	AMI name ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20240301	Termination protection Disabled
Stop protection Disabled	Launch time Thu Apr 25 2024 11:20:45 GMT-0400 (Paraguay Standard Time) (5 days)	AMI location amazon/ubuntu/images/hvm-amd64-server-20240301
Instance auto-recovery Default	Lifecycle normal	Stop-hibernate behavior Disabled
AMI Launch index 0	Key pair assigned at launch raulbeni-aws	State transition reason –
Credit specification standard	Kernel ID –	State transition message –
Usage operation RunInstances	RAM disk ID –	Owner 278229051818
Enclaves Support –	Boot mode uefi-preferred	Current instance boot mode legacy-bios

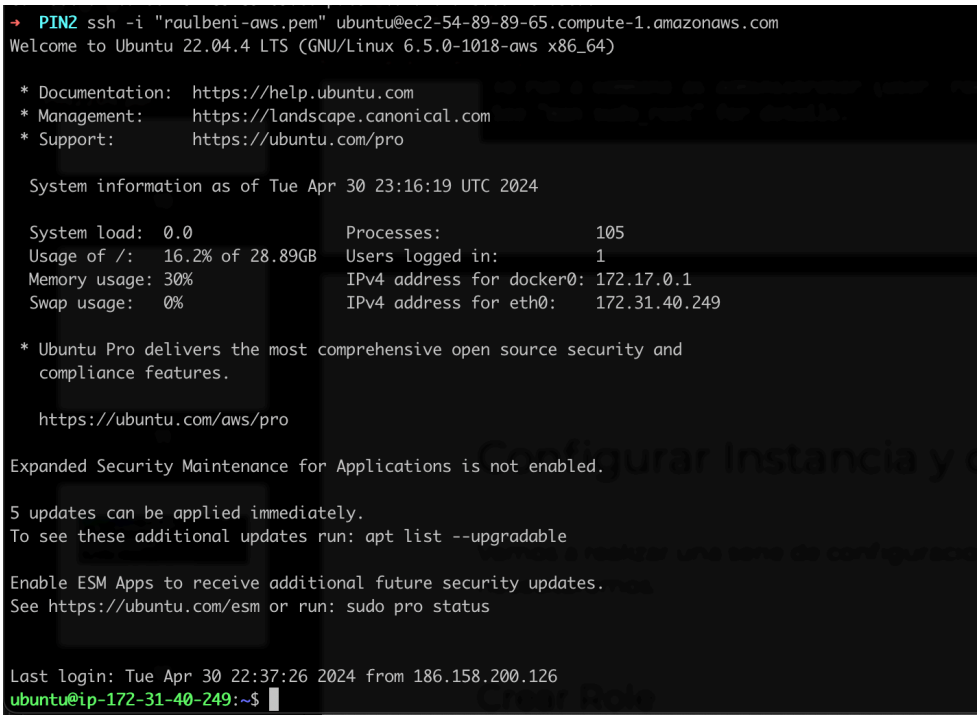
En AWS, creamos un rol llamado **ec2-admin** con todos los permisos administrador asociados a él.



Asociamos dicho rol a la instancia EC2 creada PIN2:



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



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

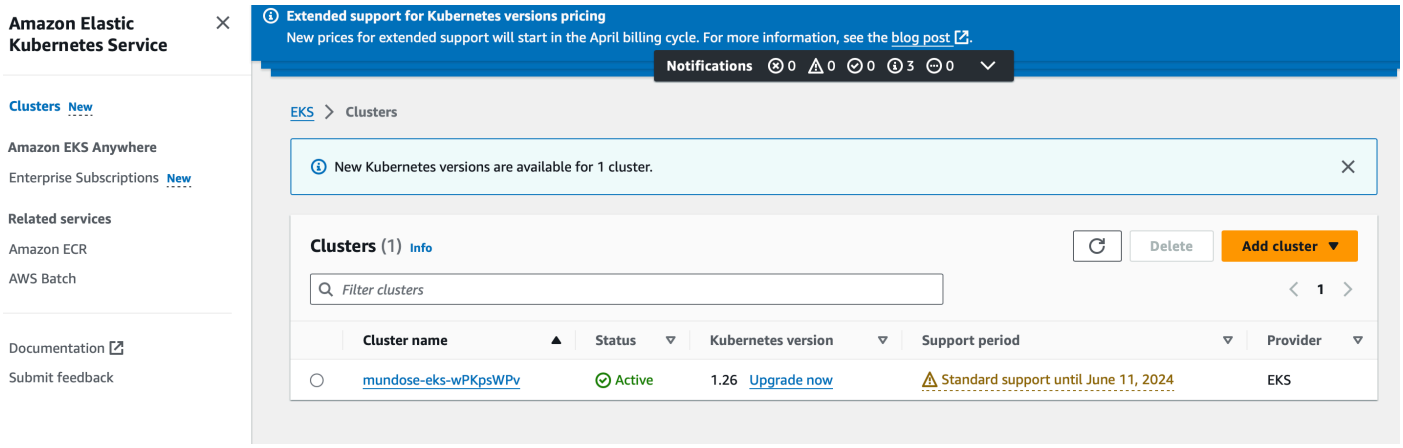
- AWS CLI
- Kubectl
- Ekctl
- Docker
- HELM
- Terraform

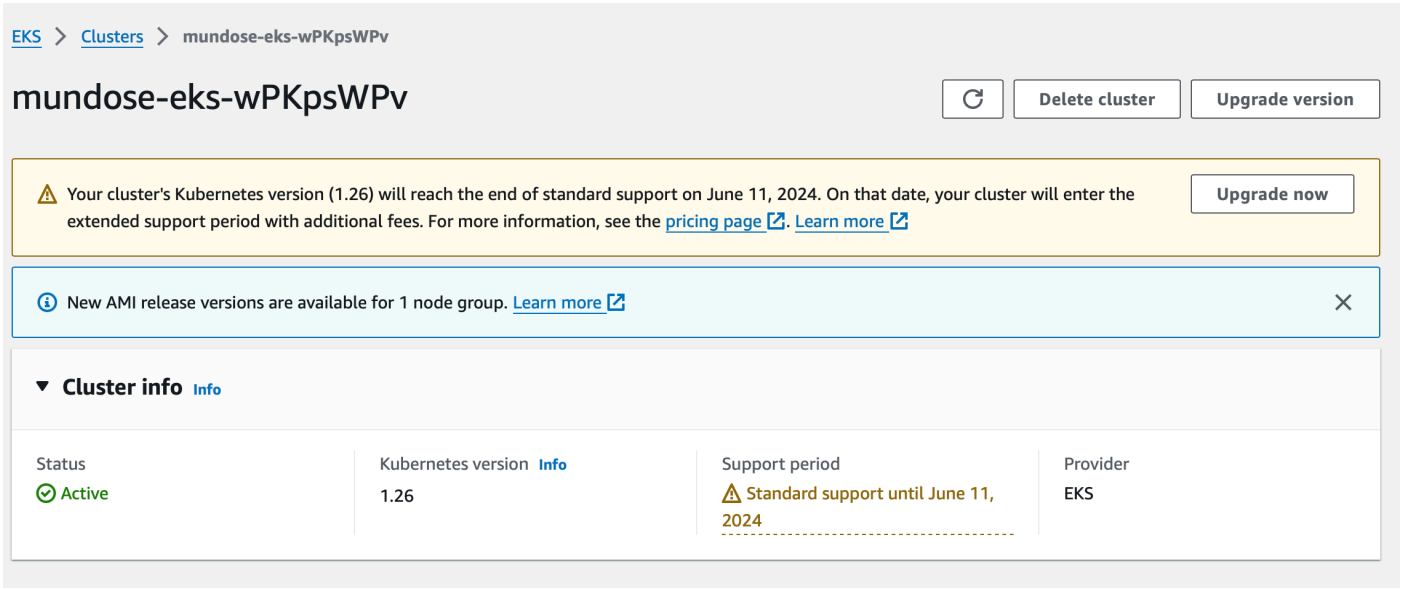
```bash
#!/bin/bash

echo "Installing AWS CLI"
apt install unzip -y
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install

echo "Installing kubectl"
curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.2/2023-03-17/bin/linux/amd64/kubectl
chmod +x ./kubectl
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
kubectl version --client

echo "Installing ekctl"
# Download EKS CLI https://github.com/weaveworks/eksctl
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_a
sudo mv /tmp/eksctl /usr/local/bin
export PATH=$PATH:/usr/local/bin
echo 'export PATH=$PATH:/usr/local/bin' >> ~/.bashrc
eksctl version

echo "Installing docker"
sudo yum install -y docker
sudo usermod -a -G docker ec2-user
newgrp docker
wget https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)
sudo mv docker-compose-$(uname -s)-$(uname -m) /usr/local/bin/docker-compose
sudo chmod -v +x /usr/local/bin/docker-compose
sudo systemctl enable docker.service
sudo systemctl start docker.service

echo "Installing Helm"
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh

echo "Installing terraform "
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum install -y terraform
```

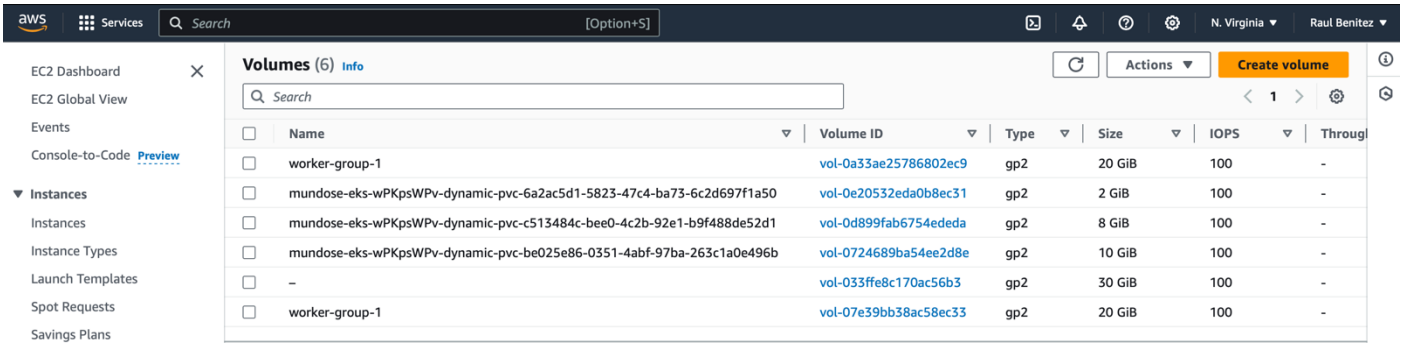
Una vez las herramientas listas y verificadas, procedemos a crear el cluster Kubernetes.

```
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
Submit feedback
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
New Kubernetes versions are available for 1 cluster.
Clusters (1) Info
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
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
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

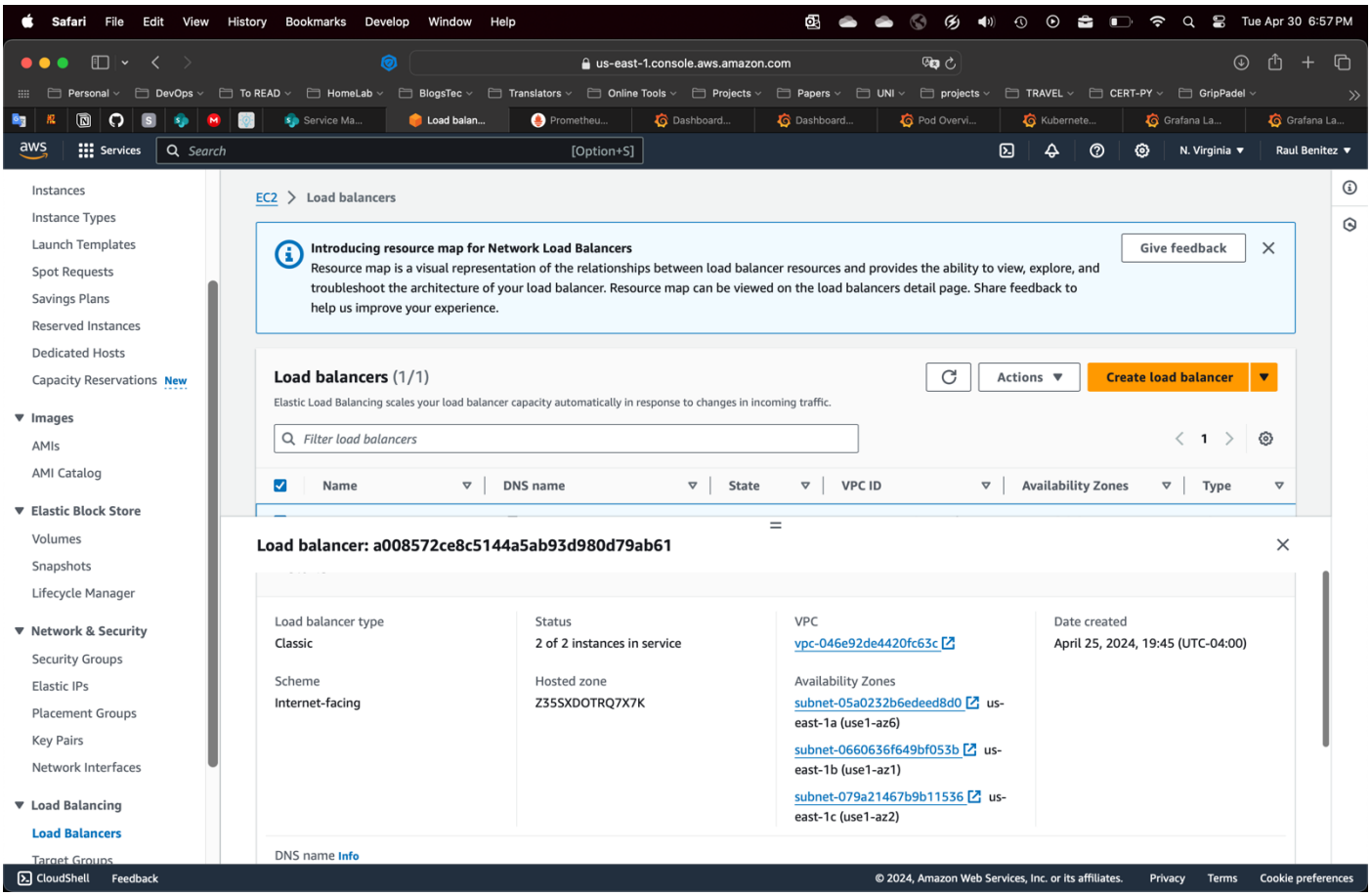

aws
Services
Search
[Option+S]
N. Virginia
Raul Benitez
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Volumes (6) Info
Actions
Create volume
Search
1
Name
Volume ID
Type
Size
IOPS
worker-group-1 vol-0a33ae25786802ec9 gp2 20 GiB 100 -
mundose-eks-wPKpsWPv-dynamic-pvc-6a2ac5d1-5823-47c4-ba73-6c2d697f1a50 vol-0e20532eda0b8ec31 gp2 2 GiB 100 -
mundose-eks-wPKpsWPv-dynamic-pvc-c513484c-bee0-4c2b-92e1-b9f488de52d1 vol-0d899fab6754ededa gp2 8 GiB 100 -
mundose-eks-wPKpsWPv-dynamic-pvc-be025e86-0351-4abf-97ba-263c1a0e496b vol-0724689ba54ee2d8e gp2 10 GiB 100 -
– vol-033ffe8c170ac56b3 gp2 30 GiB 100 -
worker-group-1 vol-07e39bb38ac58ec33 gp2 20 GiB 100 -

Nginx

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

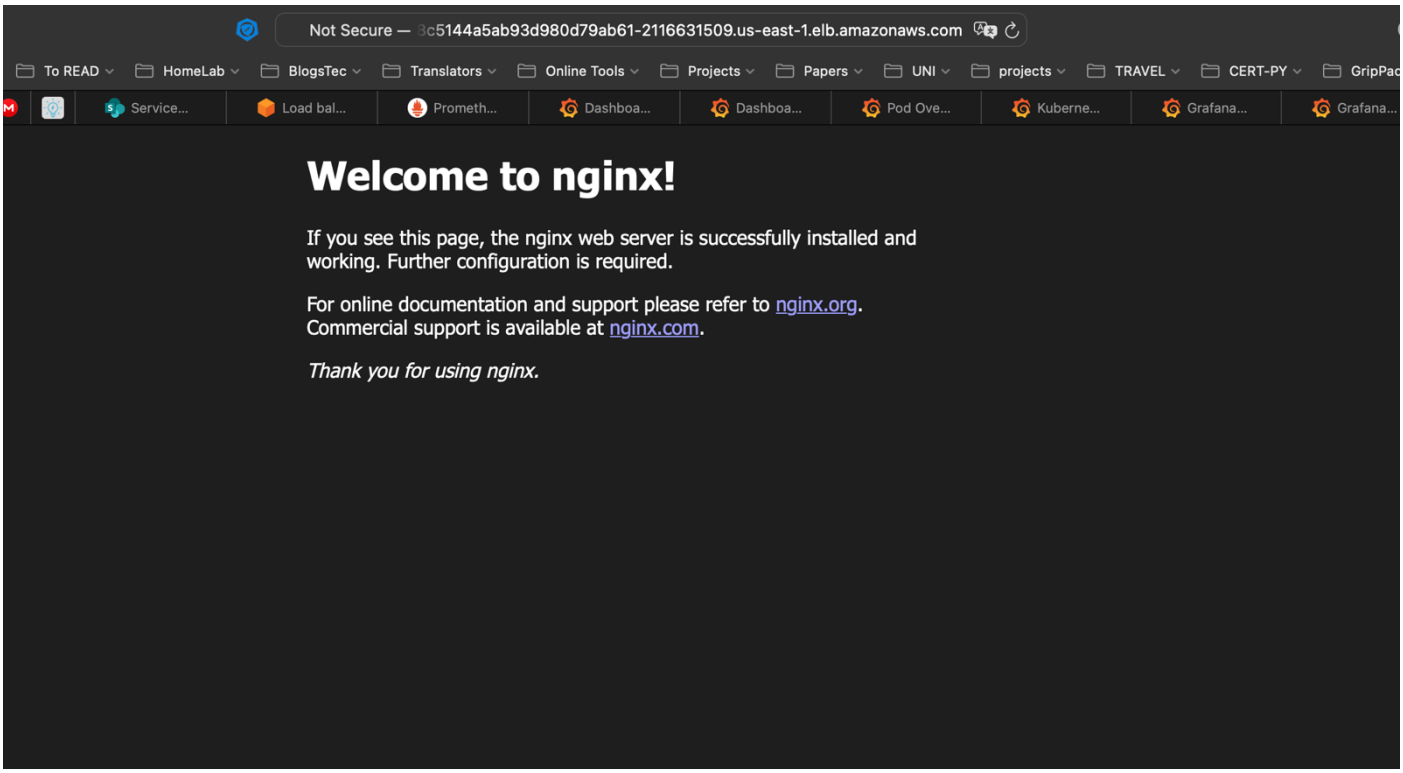
```yaml
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3 # tells deployment to run 1 pods matching the template
  template: # create pods 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 generar un LoadBalancer asociado al pod del nginx.



Verificamos la URL generada por el Load Balancer

http://a008572ce8c5144a5ab93d980d79ab61-2116631509.us-east-1.elb.amazonaws.com



Promotheus

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 lográ 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
```

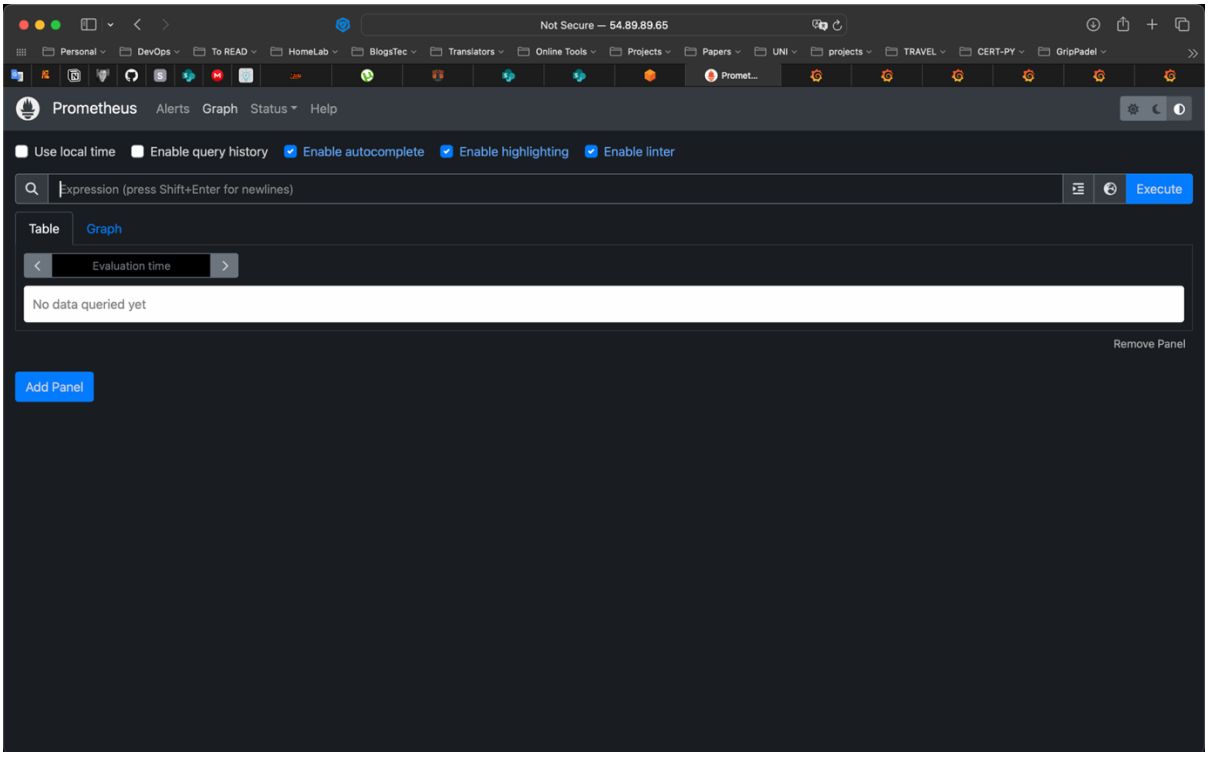
Verificamos que todas las replicas funcionen correctamente:

```
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-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**:



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#
```

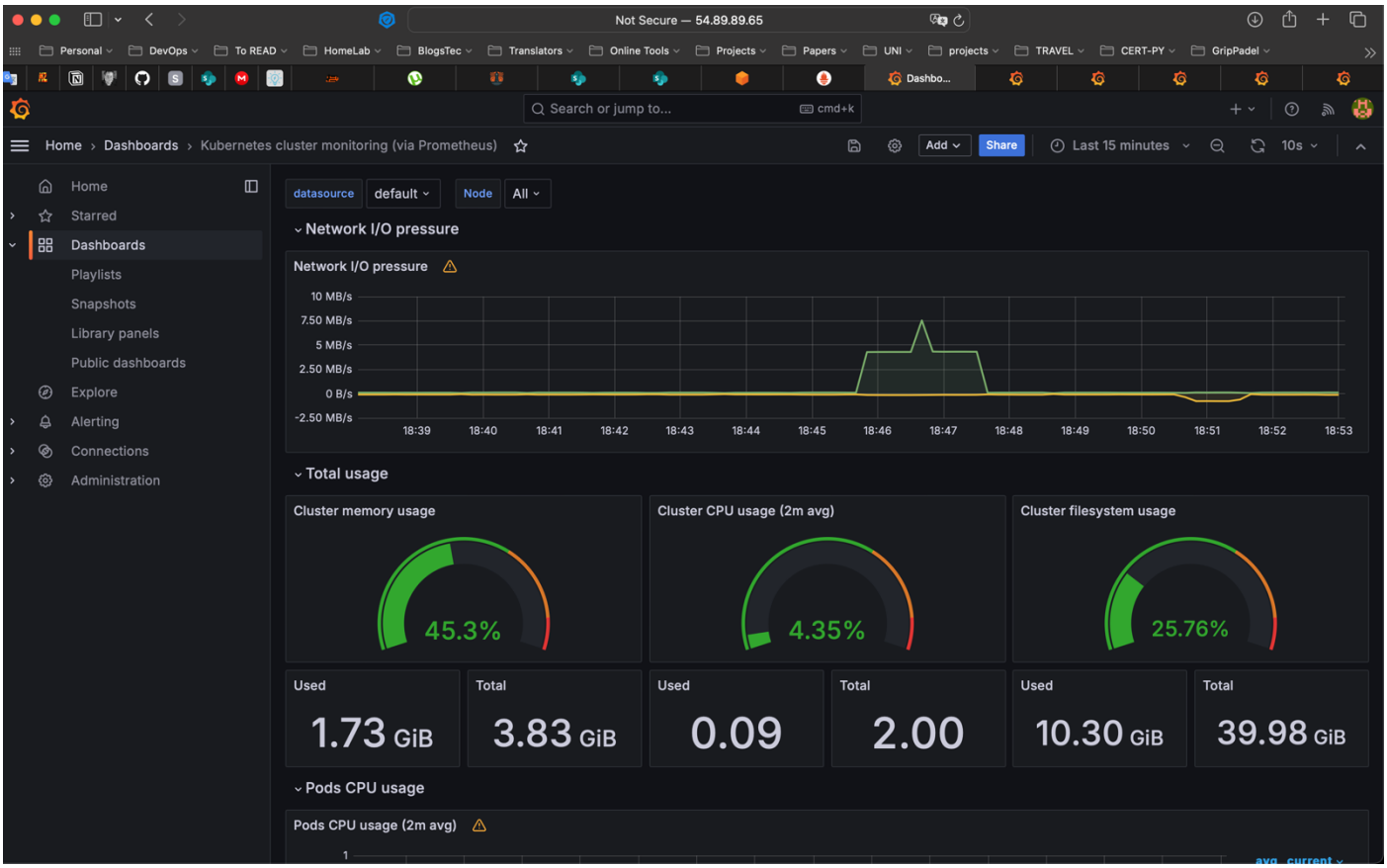
Extraemos la contraseña

```
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/PIN2export POD_NAME=$(kubectl get pods --namespace grafana -l "app.kubernetes.io/name=grafana,app.kubernet
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
```

Habilitamos el puerto 3000 para el protocolo TCP en el grupo de seguridad de AWS y nos dirijimos 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	Security group ID	Description	VPC ID
launch-wizard-6	sg-032ec78b0a403ddf1	launch-wizard-6 created 2024-04-25T15:17:48.634Z	vpc-7f42ac1a

Owner	Inbound rules count	Outbound rules count
278229051818	5 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

Inbound rules (5)

Search

Manage tags | Edit inbound rules

1

Name	Security group rule...	IP version	Type	Protocol	Port range
–	sgr-0e70ccc0d7e667c74	IPv4	Custom TCP	TCP	8080
–	sgr-0308d6f9a1e461d73	IPv4	SSH	TCP	22
–	sgr-0e91729b83fb093...	IPv4	HTTPS	TCP	443
–	sgr-0eb49ebe852e300...	IPv4	Custom TCP	TCP	3000
–	sgr-0f961d748cae180a4	IPv4	HTTP	TCP	80

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

Key pairs (3) Info

Key pairs

Find Key Pair by attribute or tag

Actions | Create key

1

Name	Type	Created	Fingerprint	ID
pin	rsa	2024/04/25 15:50 GMT-4	45:c5:62:91:98:7c:7b:98:2e:7a:f3:32:...	key-038056f5a8e49f7cc