CERTIFICACIÓN UNIVERSITARIA

UNC | FCEFyN

Devops

Proyecto integrador

mundosE

Proyecto integrador Devops

Introducción

Este proyecto tiene como idea principal el aprendizaje sobre distintos temas y la puesta enpráctica 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 AWSpara poder desde allí realizar todas las tareas necesarias.

Por último, configuramos la parte de monitoreo de pods con el stack de Prometheus y Grafana.

https://github.com/EducacionMundose/mundoes

Crear instancia EC2

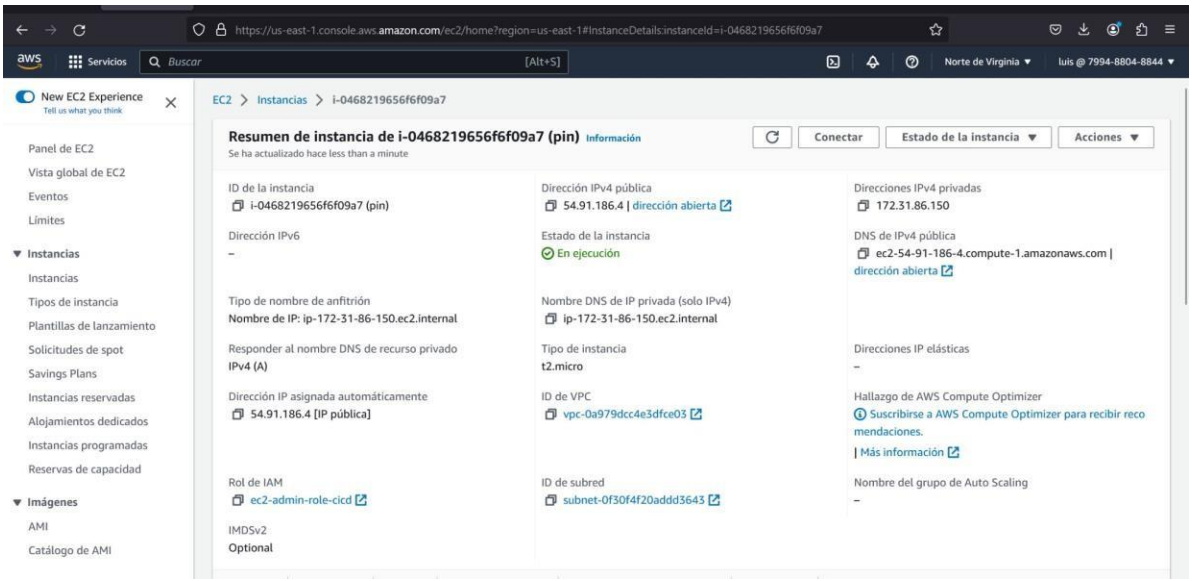
Crear instancia siguiendo ej. del PIN:

Region: us-east-1
Sistema Operativo : Ubuntu Server 22.04
Family (Tipo): t2.micro

En la sección de “user data” se proceden a cargar todos los scripts para instalar las herramientas necesarias, como AWS CLI, KUBECTL, Docker, Helm, etc. y que listamos enel archivo: Repositorio.

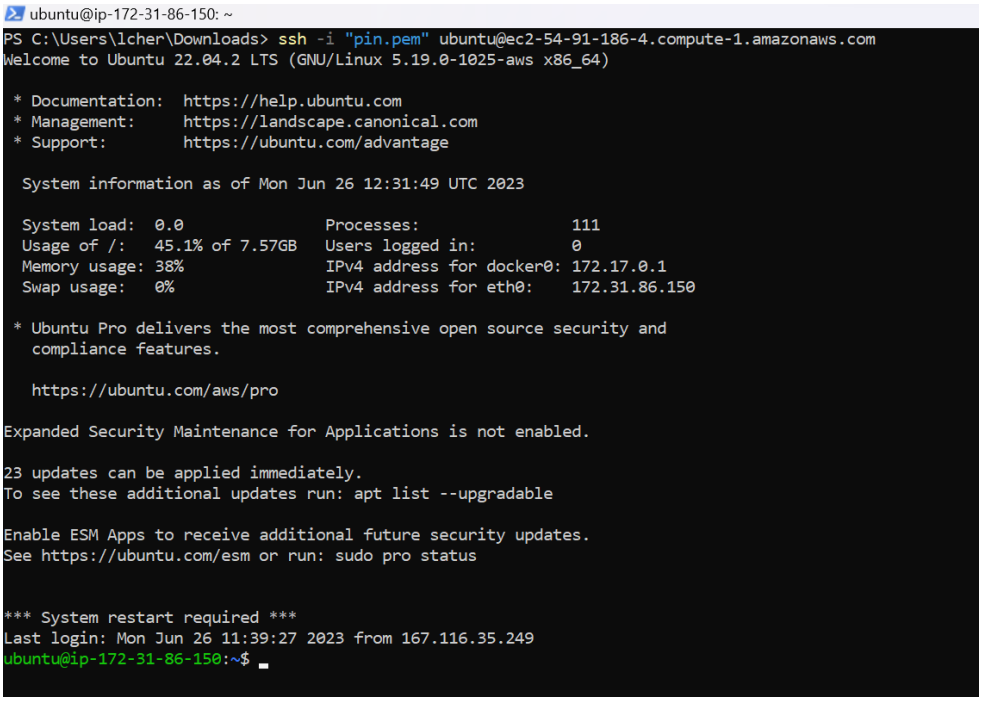
Se crea un par de claves para poder conectarse, llamadas “pin” en formato pem

Data de Instancia:



Se le agrega a la instancia el rol ec2-admin, previamente creado

Conectar con instancia por SSH:



Crear cluster con eksctl

```
eksctl create cluster \
--name eks-mundos-e \
--region us-east-1 \
--node-type t3.small \
–nodes 3 \
--with-oidc \
--ssh-access \
--ssh-public-key pin \
--managed \
--full-ecr-access \
--zones us-east-1a,us-east-1b,us-east-1c
```

El objetivo en esta instancia, es desplegar un pod de nginx, utilizando cualquier método válido,hasta la misma consola de aws.

Verificar NGINX en hostname:

Instalar herramientas de monitoreo de pods

Instalación del driver EBS [acorde a este documento oficial de AWS]

Deploy driver

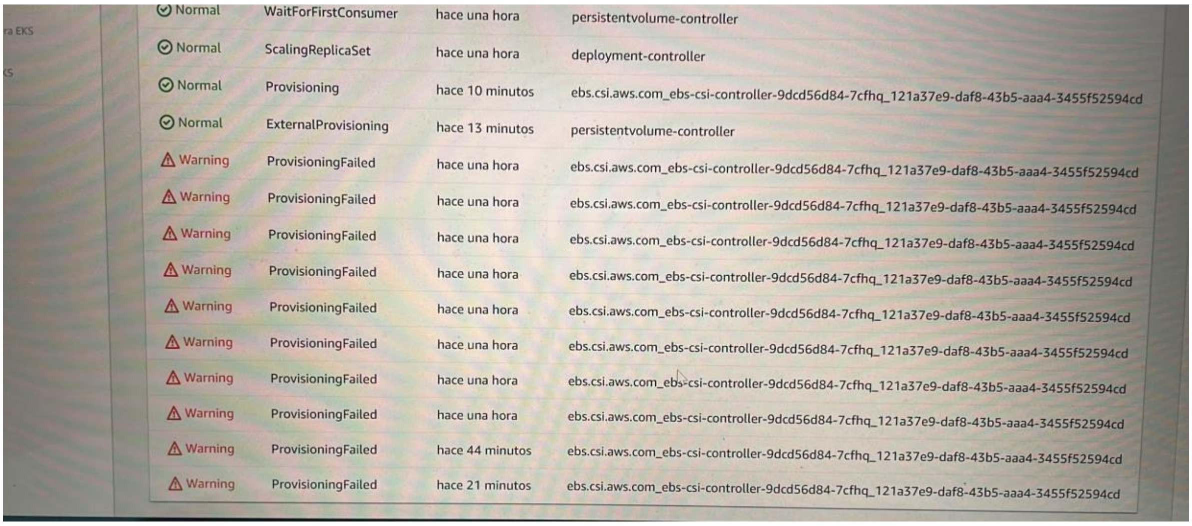
You may deploy the EBS CSI driver via Kustomize, Helm, or as an Amazon EKS managed add-on.

Kustomize

```
kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-1.20"
```

y con el siguiente comando:
kubectl apply -k
"github.com/kubernetes-sigs/aws-ebs-csi-
driver/deploy/kubernetes/overlays/stable/?ref=rele ase-1.20"

Hecho eso, se verifica que no inician los servicios afines a AWS EBS y se revisa en el dashboard de AWS la posible causa:



Normal	WaitForFirstConsumer	hace una hora	persistentvolume-controller
Normal	ScalingReplicaSet	hace una hora	deployment-controller
Normal	Provisioning	hace 10 minutos	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Normal	ExternalProvisioning	hace 13 minutos	persistentvolume-controller
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace una hora	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace 44 minutos	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd
Warning	ProvisioningFailed	hace 21 minutos	ebs.csi.aws.com_ebs-csi-controller-9dcd56d84-7cfhq_121a37e9-daf8-43b5-aaa4-3455f52594cd

Y se corrobora que hay un problema de aprovisionamiento.

Procedemos a hacer troubleshooting del error, para lo cual investigando encontramos que debemos decodificar el mensaje que AWS comparte a fin de identificar el componente quegenera el issue. Hallamos que la causa, es que esos recursos no pueden iniciar por falta de almacenamiento.
Por ende, identificamos el nodegroup que debió generarlos, y le asociamos la política deadministración de almacenamiento (EBS) para poder administrar volúmenes. Esoresuelve el incidente, y nos permite avanzar a lo siguiente.

Instalación Prometheus siguiendo la secuencia siguiente:

helm repo add prometheus-community https://prometheus-community.github.io/helm-chartshelm repo update
kubectl create namespace prometheus
helm install prometheus prometheus-community/prometheus --namespace prometheus --setalertmanager.persistentVolume.storageClass="gp2" --set server.persistentVolume.storageClass="gp2

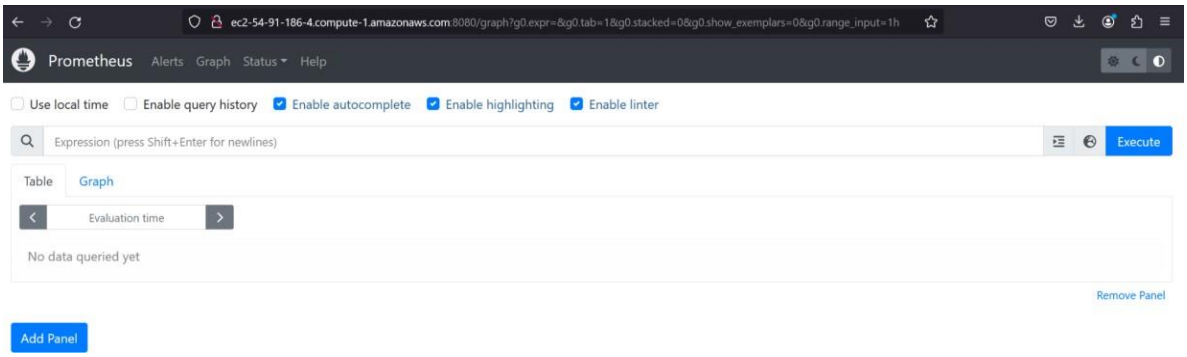
Prometheus corriendo:

```
EKS!sAWSome
ubuntu@ip-172-31-86-150:~/environment/grafana$ kubectl get all -n prometheus
NAME                                                READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                       1/1     Running   0          93m
pod/prometheus-kube-state-metrics-5fb6fbbf78-jxhmd  1/1     Running   0          93m
pod/prometheus-prometheus-node-exporter-5nnmx       1/1     Running   0          93m
pod/prometheus-prometheus-node-exporter-cqfgr       1/1     Running   0          93m
pod/prometheus-prometheus-node-exporter-zpqn2       1/1     Running   0          93m
pod/prometheus-prometheus-pushgateway-7d55869d46-nwbgs   1/1     Running   0          93m
pod/prometheus-server-78c8b85bf7-hcm8b              2/2     Running   0          93m

NAME                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/prometheus-alertmanager               ClusterIP   10.100.227.31    <none>        9093/TCP   93m
service/prometheus-alertmanager-headless      ClusterIP   None             <none>        9093/TCP   93m
service/prometheus-kube-state-metrics         ClusterIP   10.100.200.195   <none>        8080/TCP   93m
service/prometheus-prometheus-node-exporter   ClusterIP   10.100.115.67    <none>        9100/TCP   93m
service/prometheus-prometheus-pushgateway     ClusterIP   10.100.135.108   <none>        9091/TCP   93m
service/prometheus-server                     ClusterIP   10.100.166.163   <none>        80/TCP     93m

NAME                                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter   3         3         3       3            3           <none>          93m

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics   1/1     1            1           93m
deployment.apps/prometheus-prometheus-pushgateway   1/1     1            1           93m
deployment.apps/prometheus-server               1/1     1            1           93m

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-5fb6fbbf78  1         1         1       93m
replicaset.apps/prometheus-prometheus-pushgateway-7d55869d46   1         1         1       93m
replicaset.apps/prometheus-server-78c8b85bf7              1         1         1       93m
```

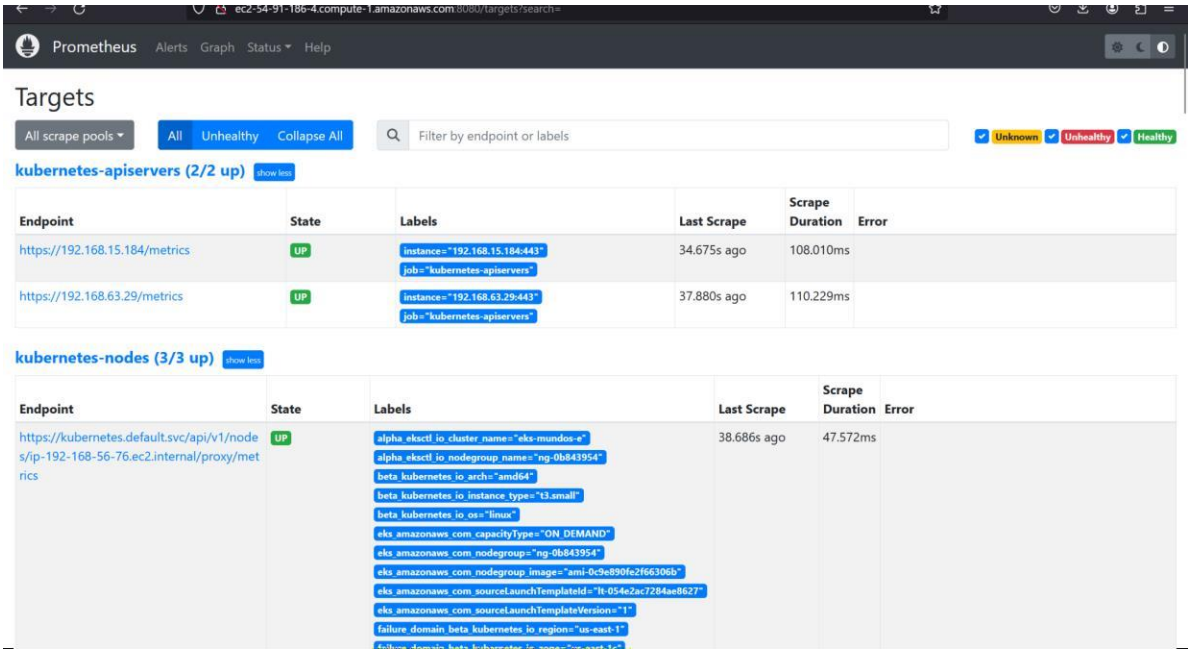
Port forward:

```
statefulset.apps/prometheus-alertmanager   1/1     93m
ubuntu@ip-172-31-86-150:~/environment/grafana$ kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090 --address 0.0.0.0
Forwarding from 0.0.0.0:8080 -> 9090
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

Visita en el público



Ir a targets:



Desplegar Grafana

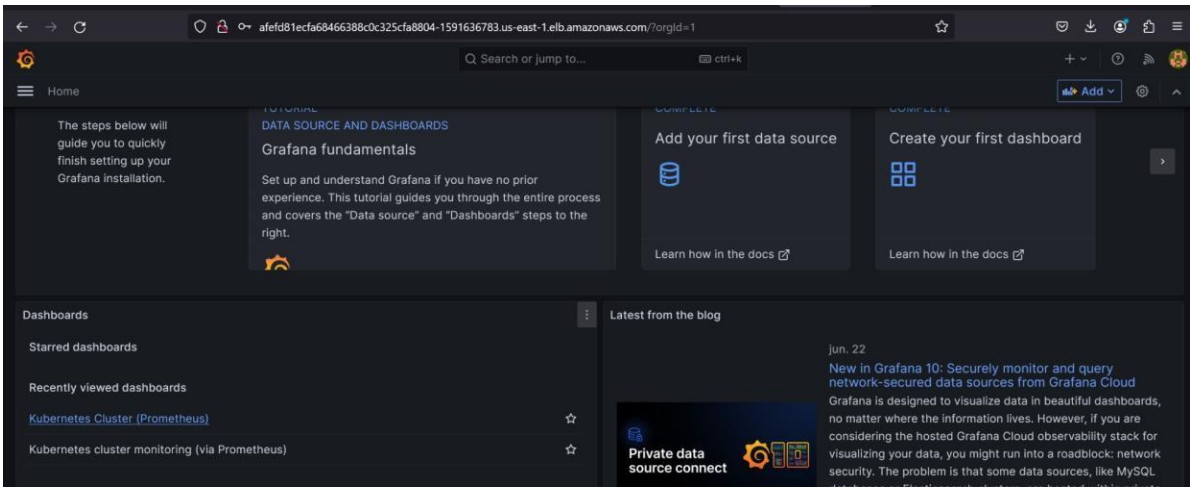
kubectl create namespace grafana

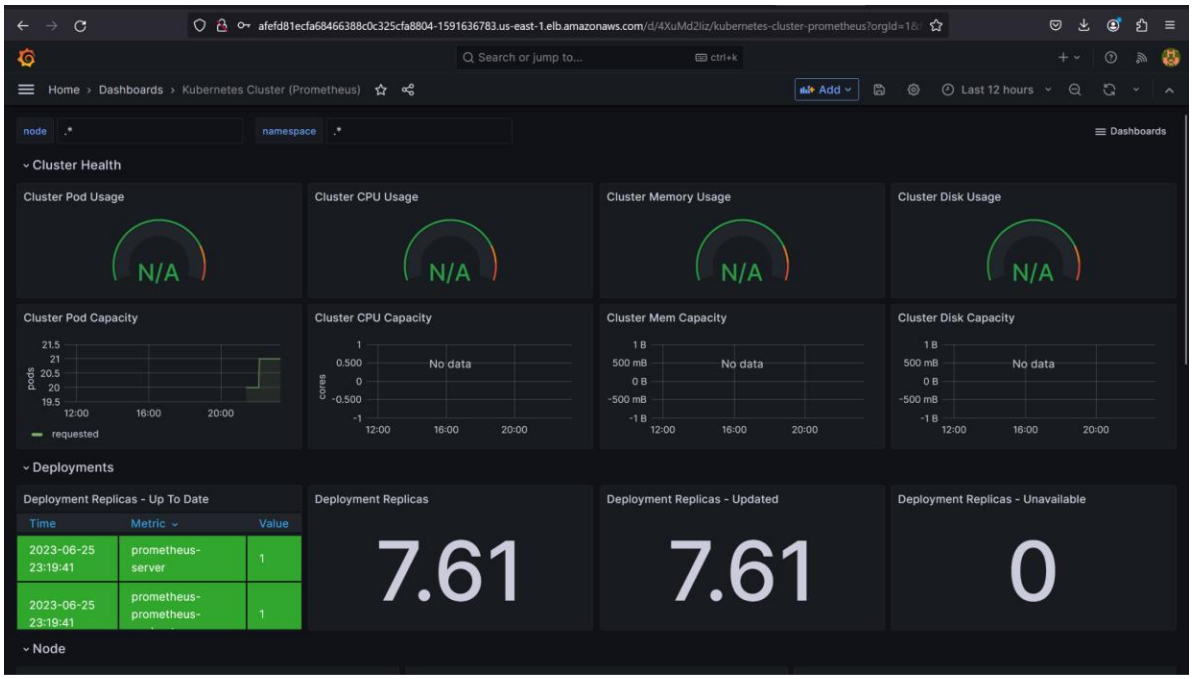
Crear el archivo yaml siguiendo ejemplo del PIN, en la ruta

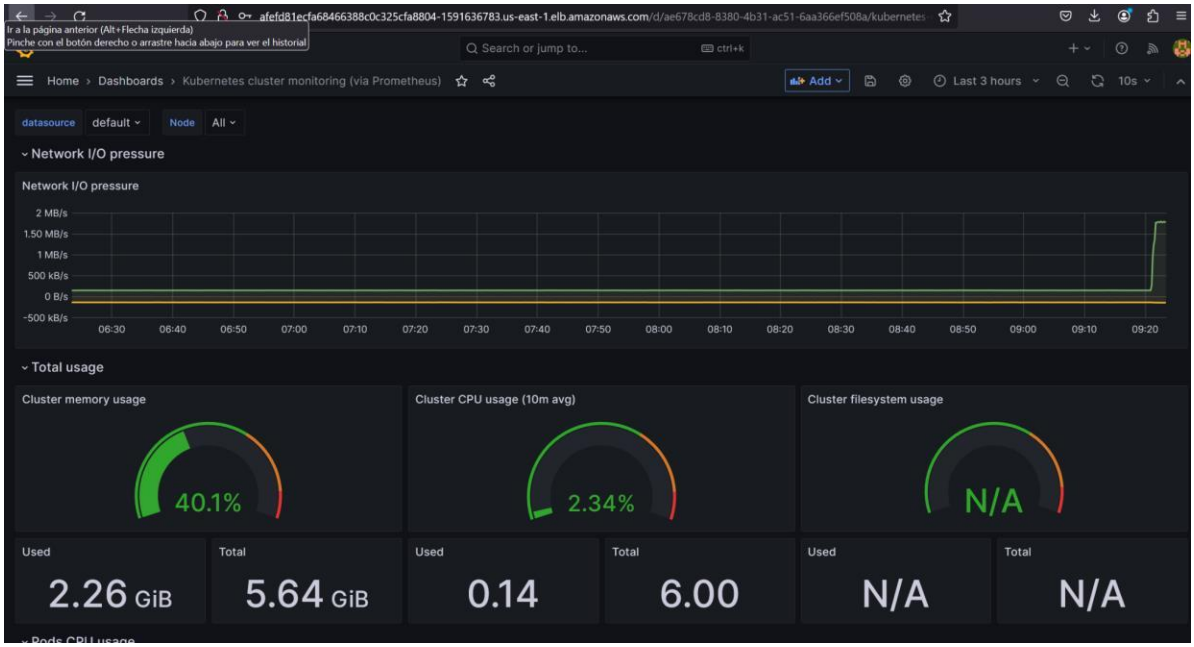
*sugeridahelm install grafana grafana/grafana *
*--namespace grafana *
*--set persistence.storageClassName="gp2" *
*--set persistence.enabled=true *
*--set adminPassword='EKS!sAWSome' *
*--values ${HOME}/environment/grafana/grafana.yaml *
--set service.type=LoadBalancer

```
ubuntu@ip-172-31-86-150:~/environment/grafana$ kubectl get all -n grafana
NAME                           READY   STATUS    RESTARTS   AGE
pod/grafana-574676d57f-vf9dz   1/1     Running   0          70m

NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)        AGE
service/grafana   LoadBalancer   10.100.183.93   afefd81ecfa68466388c0c325cfa8804-1591636783.us-east-1.elb.amazonaws.com   80:30671/TCP   70m

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana   1/1     1            1           70m

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/grafana-574676d57f   1         1         1       70m
ubuntu@ip-172-31-86-150:~/environment/grafana$
```

Acceso y adición de Dashboards (3119 y 6417):







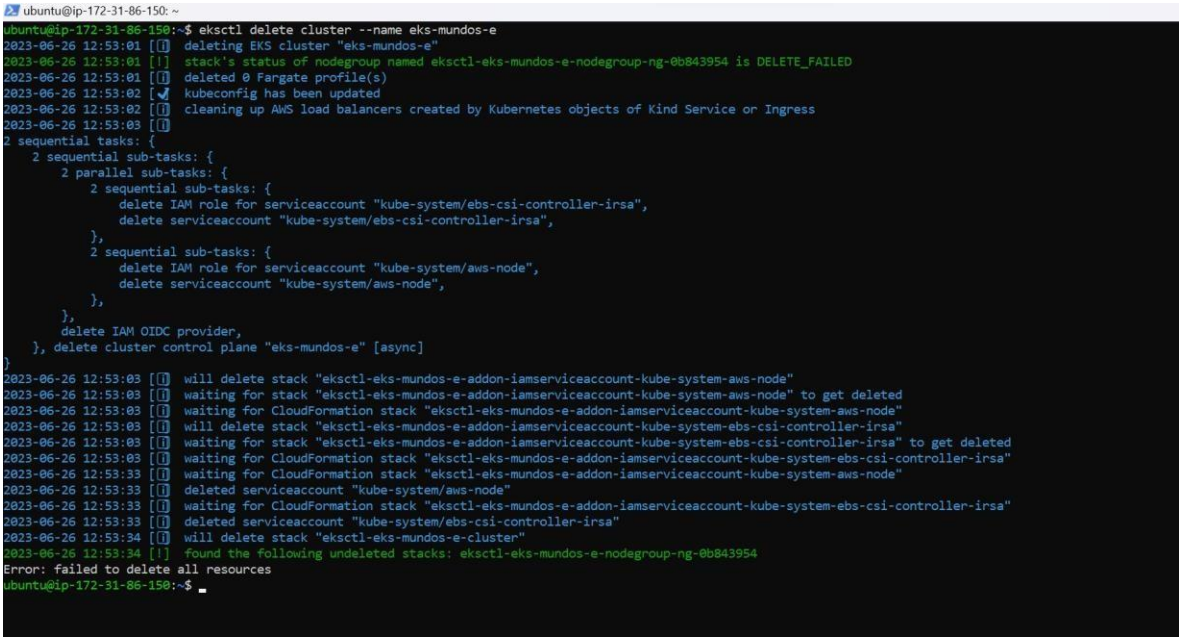
Cleanup de recursos

helm uninstall prometheus --namespace prometheuskubectl delete ns prometheus
helm uninstall grafana --namespace grafanakubectl delete ns grafana
rm -rf ${HOME}/environment/grafana

```
ubuntu@ip-172-31-86-150:~$ kubectl delete ns prometheus
namespace "prometheus" deleted
ubuntu@ip-172-31-86-150:~$ helm uninstall grafana --namespace grafana
release "grafana" uninstalled
ubuntu@ip-172-31-86-150:~$ kubectl delete ns grafana
namespace "grafana" deleted
ubuntu@ip-172-31-86-150:~$ rm -rf ${HOME}/environment/grafana
ubuntu@ip-172-31-86-150:~$
```

Borrar Cluster EKS

eksctl delete cluster --name eks-mun



Anexo

Solución al EKS CSI

Visualizing EKS Cluster metrics With Prometheus And Grafana

Prometheus is used as a data source for monitoring Kubernetes clusters running on EKS. It collects metrics from various sources, such as Kubernetes nodes and containers, and provides a flexible query language for analyzing and visualizing the data. Prometheus can also trigger alerts based on predefined rules, allowing operators to take action when issues arise in the cluster.

Prometheus is often used in conjunction with Grafana, a popular open-source tool for visualizing and analyzing time-series data. Together, Prometheus and Grafana provide a powerful monitoring and alerting solution for EKS clusters.

Installation:

1. Create A EKS Cluster:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: test
  region: ap-south-1
iam:
  withOIDC: true
managedNodeGroups:
  - name: testv3
    instanceType: t2.large
    volumeSize: 20
    ssh:
      allow: false
    iam:
      withAddonPolicies:
        autoScaler: true
    minSize: 1
    maxSize: 3
    desiredCapacity: 1
```

apply the file,

```
eksctl create cluster -f ./path/to/file.yml
```

2. Install Prometheus:

```
kubectl create namespace prometheus

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.stora
geClass="gp2"
```

verify if everything is good:

```
kubectl get pods -n prometheus
```

output:

```
akif@akif-Lenovo-Legion-5-15ARH05:~/Desktop$ k get pod
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-0                        0/1     Pending   0          15s
prometheus-kube-state-metrics-6fcf5978bf-2mzjx   1/1     Running   0          16s
prometheus-prometheus-node-exporter-59298        1/1     Running   0          16s
prometheus-prometheus-node-exporter-t76nx        1/1     Running   0          16s
prometheus-prometheus-pushgateway-fdb75d75f-vlpmn   1/1     Running   0          15s
prometheus-server-744c555674-st7wz               0/2     Pending   0          15ss
```

you will see alertmanager and server pod is in pending step!

this is happening as prometheus server wants to use ebs driver but there is no ebs csi driver installed!

lets install it!

installing ebs csi driver:
create a iam role:

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

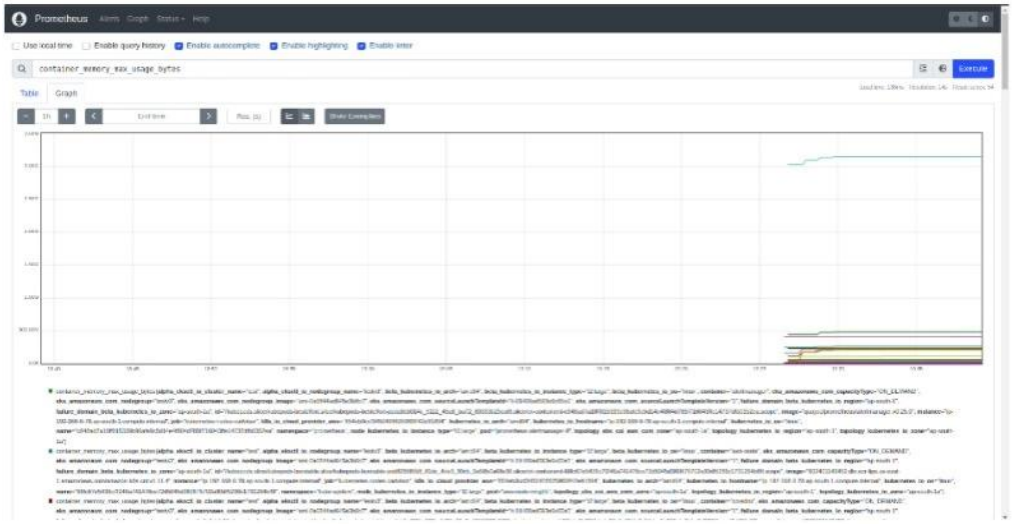
create addon:(replace account id and cluster name)

```
eksctl create addon --name aws-ebs-csi-driver --cluster <cluster    name> --service-acco
unt-role-arn arn:aws:iam::<account     _id>:role/AmazonEKS_EBS_CSI_DriverRole --force
```

now see those pods are also running!
visit the prometheus server at localhost:909

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

now we can see raw controlplane metrics and its graph:



but this data is not very comfy for visualization, so we use grafana on the top of prometheus .

Grafana:

1.make a file grafana.yml and populate it,

```
datasources:
  datasources.yaml:
    apiVersion: 1
    datasources:
    - name: Prometheus
      type: prometheus
      url: http://prometheus-server.prometheus.svc.cluster.local
      access: proxy
      isDefault: true
```

2. Grab Grafana Helm charts:

```
kubectl create namespace grafana
```

```
helm install grafana grafana/grafana \
  --namespace grafana \
  --set persistence.storageClassName="gp2" \
  --set persistence.enabled=true \
  --set adminPassword=' y o u r  p a ss w o r d' \
  --values grafana.yaml \
  --set service.type=LoadBalancer
```

4. Check if Grafana is deployed properly

```
kubectl get all -n grafana
```

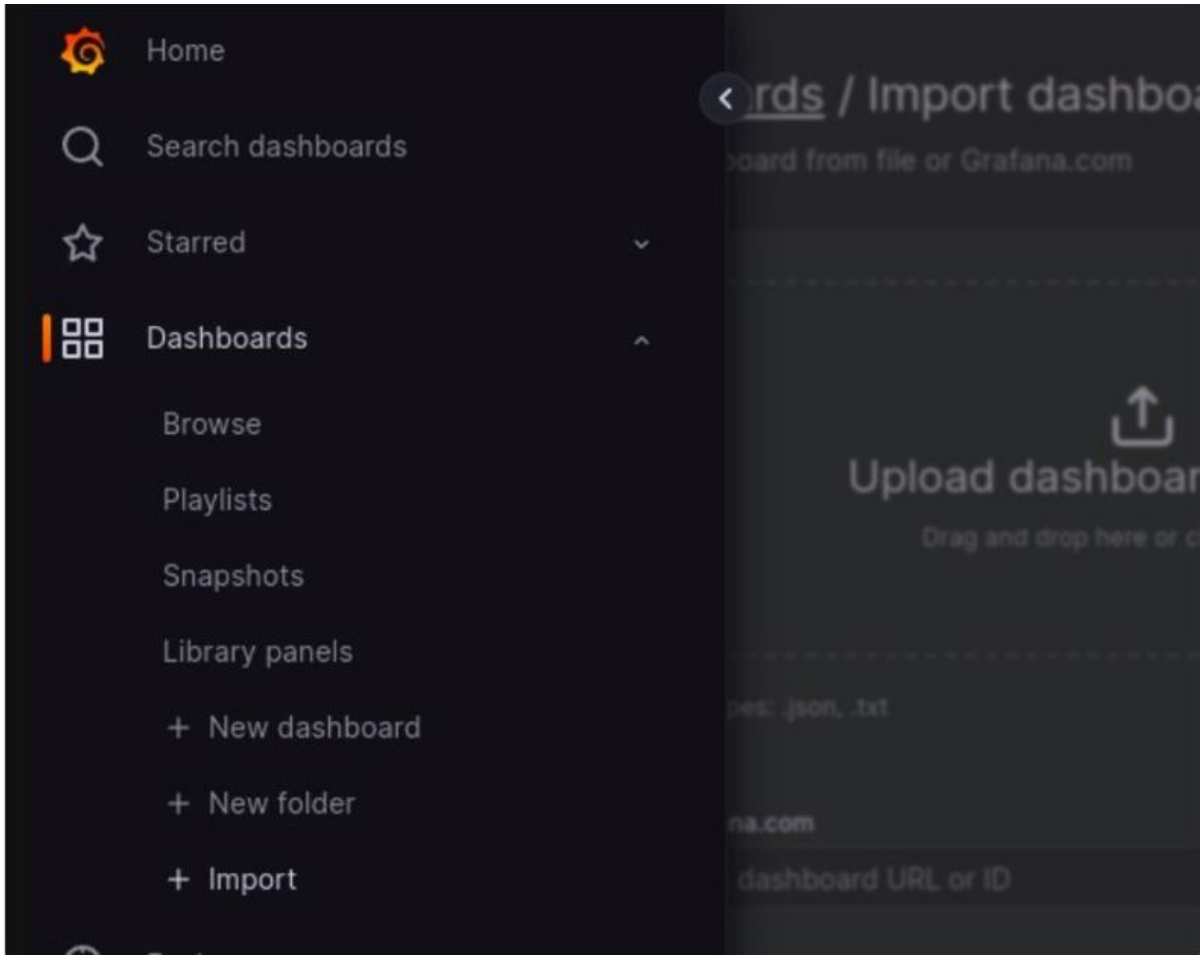
5. Get Grafana loadbalancer url

```
export ELB=$(kubectl get svc -n grafana grafana -o jsonpath='{.status.loadBalancer.ingre
ss[0].hostname}')
echo "http://$ELB"
```

6. Use username "admin" and get password by running the following:

```
kubectl get secret --namespace grafana grafana -o jsonpath="{.data.admin-password}" | ba
se64 --decode ; echo
```

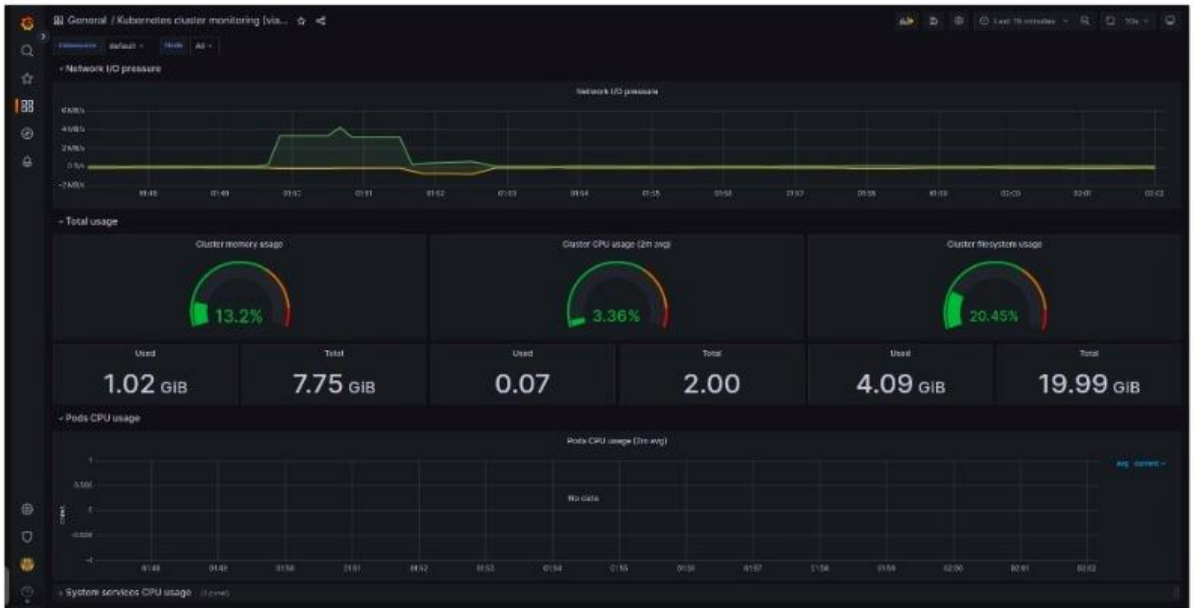
7. visit the url and login and import the dashboard:



import dashboard number 3119

and select the data source as prometheus

lets visualize:



now we can visualize the metrics simply by adding id of pre configured grafana dashboard!

you can add and learn more dashboard from here:

-

```
https://grafana.com/grafana/dashboards/
```