

CERTIFICACIÓN UNIVERSITARIA



Devops

Proyecto
integrador



WE WANT
SKILLS!

mundosE

Proyecto integrador Devops

Introducción

Este proyecto tiene 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.

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 en el archivo: [Repositorio](#).

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

Data de Instancia:

The screenshot displays the AWS Management Console interface for an EC2 instance. The left sidebar contains navigation options like 'Panel de EC2', 'Vista global de EC2', 'Eventos', 'Límites', 'Instancias', 'Tipos de instancia', 'Plantillas de lanzamiento', 'Solicitudes de spot', 'Savings Plans', 'Instancias reservadas', 'Alojamientos dedicados', 'Instancias programadas', 'Reservas de capacidad', 'Imágenes', 'AMI', and 'Catálogo de AMI'. The main content area is titled 'Resumen de instancia de i-0468219656f6f09a7 (pin)' and shows various instance details. The instance is in the 'En ejecución' state. Key details include: ID de la instancia: i-0468219656f6f09a7 (pin); Dirección IPv4 pública: 54.91.186.4; Dirección IPv4 privada: 172.31.86.150; DNS de IPv4 pública: ec2-54-91-186-4.compute-1.amazonaws.com; Tipo de instancia: t2.micro; ID de VPC: vpc-0a979dce4e3dfce03; ID de subred: subnet-0f30f4f20addd3643; Rol de IAM: ec2-admin-role-cicd; IMDSv2: Optional.

| Resumen de instancia de i-0468219656f6f09a7 (pin) | | |
|---|--------------------------------------|---|
| ID de la instancia | Dirección IPv4 pública | Direcciones IPv4 privadas |
| i-0468219656f6f09a7 (pin) | 54.91.186.4 dirección abierta | 172.31.86.150 |
| Dirección IPv6 | Estado de la instancia | DNS de IPv4 pública |
| - | En ejecución | ec2-54-91-186-4.compute-1.amazonaws.com dirección abierta |
| Tipo de nombre de anfitrión | Nombre DNS de IP privada (solo IPv4) | Direcciones IP elásticas |
| Nombre de IP: ip-172-31-86-150.ec2.internal | ip-172-31-86-150.ec2.internal | - |
| Responder al nombre DNS de recurso privado IPv4 (A) | Tipo de instancia | Hallazgo de AWS Compute Optimizer |
| 54.91.186.4 [IP pública] | t2.micro | Suscribirse a AWS Compute Optimizer para recibir recomendaciones. |
| Dirección IP asignada automáticamente | ID de VPC | Más información |
| 54.91.186.4 [IP pública] | vpc-0a979dce4e3dfce03 | |
| Rol de IAM | ID de subred | Nombre del grupo de Auto Scaling |
| ec2-admin-role-cicd | subnet-0f30f4f20addd3643 | - |
| IMDSv2 | | |
| Optional | | |

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

Conectar con instancia por SSH:

```
ubuntu@ip-172-31-86-150: ~
PS C:\Users\lcher\Downloads> ssh -i "pin.pem" ubuntu@ec2-54-91-186-4.compute-1.amazonaws.com
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

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

System information as of Mon Jun 26 12:31:49 UTC 2023

System load:  0.0               Processes:           111
Usage of /:   45.1% of 7.57GB   Users logged in:    0
Memory usage: 38%              IPv4 address for docker0: 172.17.0.1
Swap usage:   0%               IPv4 address for eth0:  172.31.86.150

 * 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.

23 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

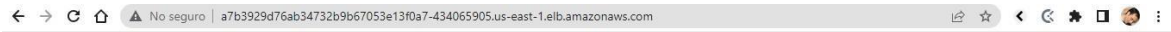
*** System restart required ***
Last login: Mon Jun 26 11:39:27 2023 from 167.116.35.249
ubuntu@ip-172-31-86-150:~$
```

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=release-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:

| | | | | |
|--------|---------|----------------------|-----------------|---|
| ra EKS | Normal | WaitForFirstConsumer | hace una hora | persistentvolume-controller |
| CS | 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 que genera 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 de administración de almacenamiento (EBS)** para poder administrar volúmenes. Eso resuelve 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-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"
```

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

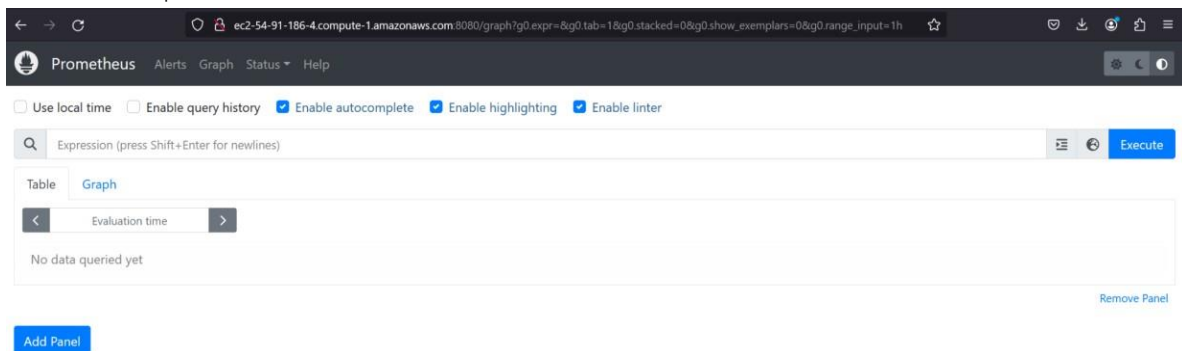
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/prometheus-kube-state-metrics      1/1      1             1           93m
deployment.apps/prometheus-prometheus-pushgateway  1/1      1             1           93m
deployment.apps/prometheus-server           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

The screenshot shows the Prometheus web interface. The 'Targets' tab is selected. There are two target groups listed:

- kubernetes-apiservers (2/2 up)**: This group contains two targets, both in a 'UP' state. The first target is at `https://192.168.15.184/metrics` with a last scrape 34.675s ago. The second target is at `https://192.168.63.29/metrics` with a last scrape 37.880s ago. Both have labels for 'instance' and 'job'.
- kubernetes-nodes (3/3 up)**: This group contains three targets, all in a 'UP' state. The first target is at `https://kubernetes.default.svc/api/v1/nodes/ip-192-168-56-76.ec2.internal/proxy/metrics` with a last scrape 38.686s ago. The other two targets are not visible in the screenshot. The labels for the first target include 'alpha.eksctl.io', 'beta.kubernetes.io', and various AWS-related labels.

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`

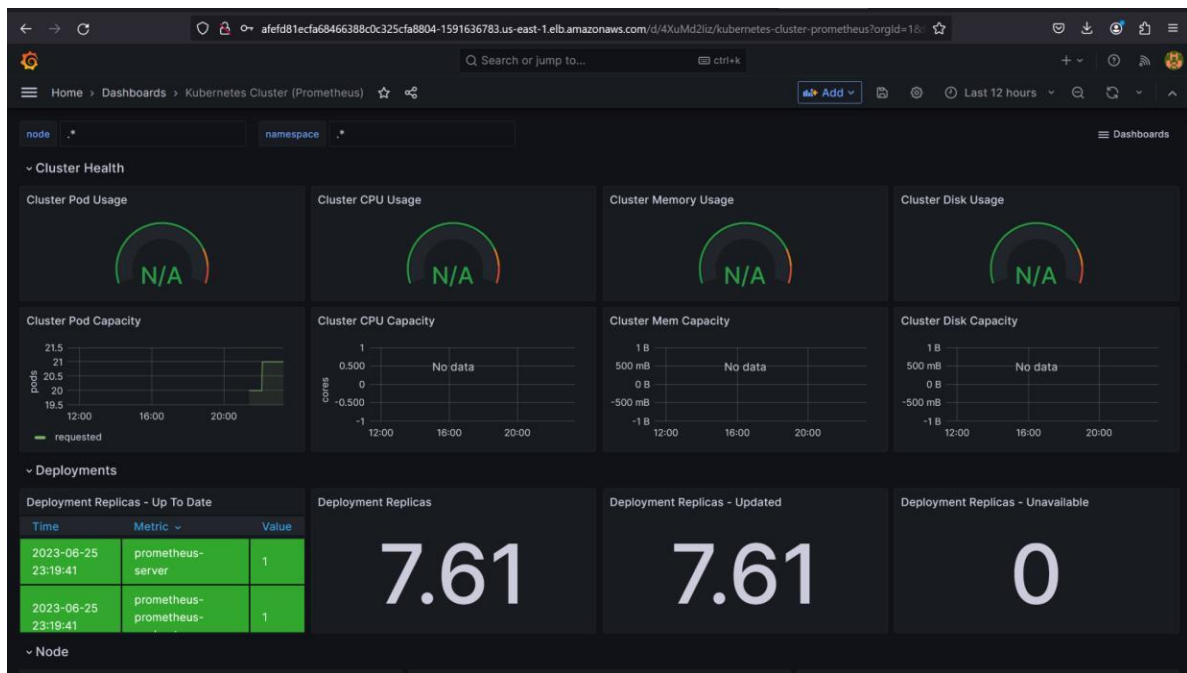
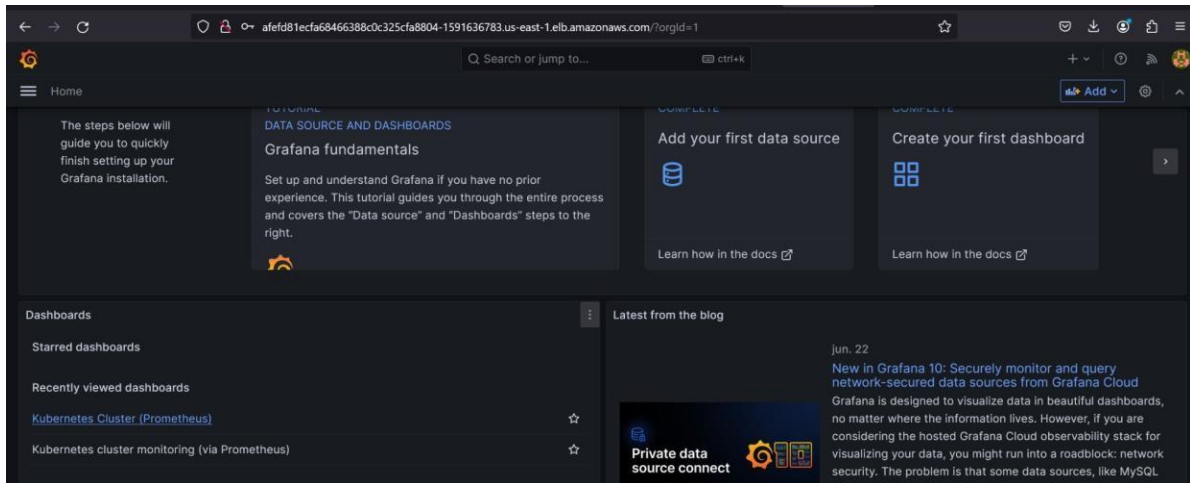
```
ubuntu@ip-172-31-86-150:~/environment/grafana$ kubectl get all -n grafana
NAME                                READY    STATUS    RESTARTS   AGE
pod/grafana-574676d57f-vf9dz        1/1      Running   0           70m

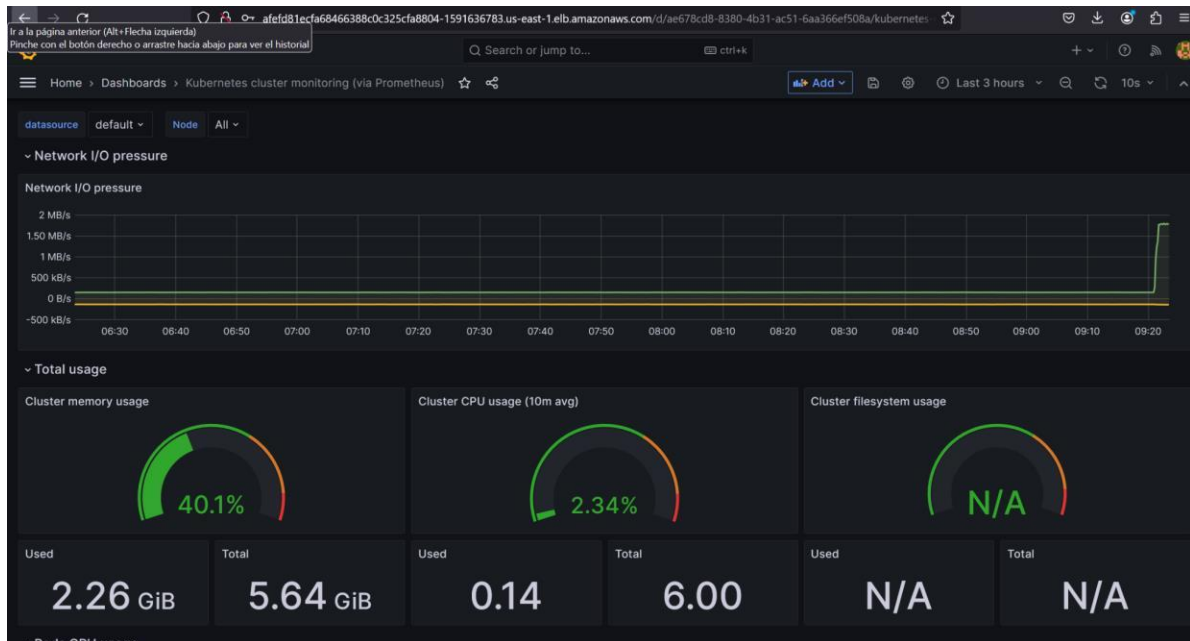
NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/grafana                      LoadBalancer  10.100.183.93    afefd81ecfa68466388c0c325cfa8804-1591636783.us-east-1.elb.amazonaws.com  80:30671/TCP     70m

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: ~

```
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

```
ubuntu@ip-172-31-86-150: ~  
ubuntu@ip-172-31-86-150:~$ eksctl delete cluster --name eks-mundos-e  
2023-06-26 12:53:01 [I] deleting EKS cluster "eks-mundos-e"  
2023-06-26 12:53:01 [I] stack's status of nodegroup named eksctl-eks-mundos-e-nodegroup-ng-0b843954 is DELETE_FAILED  
2023-06-26 12:53:01 [I] deleted 0 Fargate profile(s)  
2023-06-26 12:53:02 [✓] kubeconfig has been updated  
2023-06-26 12:53:02 [I] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress  
2023-06-26 12:53:03 [I]  
2 sequential tasks: {  
  2 sequential sub-tasks: {  
    2 parallel sub-tasks: {  
      2 sequential sub-tasks: {  
        delete IAM role for serviceaccount "kube-system/ebs-csi-controller-irsa",  
        delete serviceaccount "kube-system/ebs-csi-controller-irsa",  
      },  
      2 sequential sub-tasks: {  
        delete IAM role for serviceaccount "kube-system/aws-node",  
        delete serviceaccount "kube-system/aws-node",  
      },  
    },  
    delete IAM OIDC provider,  
  }, delete cluster control plane "eks-mundos-e" [async]  
}  
2023-06-26 12:53:03 [I] will delete stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-aws-node"  
2023-06-26 12:53:03 [I] waiting for stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-aws-node" to get deleted  
2023-06-26 12:53:03 [I] waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-aws-node"  
2023-06-26 12:53:03 [I] will delete stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-irsa"  
2023-06-26 12:53:03 [I] waiting for stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-irsa" to get deleted  
2023-06-26 12:53:03 [I] waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-irsa"  
2023-06-26 12:53:03 [I] waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-aws-node"  
2023-06-26 12:53:03 [I] deleted serviceaccount "kube-system/aws-node"  
2023-06-26 12:53:03 [I] waiting for CloudFormation stack "eksctl-eks-mundos-e-addon-iamserviceaccount-kube-system-ebs-csi-controller-irsa"  
2023-06-26 12:53:03 [I] deleted serviceaccount "kube-system/ebs-csi-controller-irsa"  
2023-06-26 12:53:03 [I] will delete stack "eksctl-eks-mundos-e-cluster"  
2023-06-26 12:53:04 [I] found the following undeleted stacks: eksctl-eks-mundos-e-nodegroup-ng-0b843954  
Error: failed to delete all resources  
ubuntu@ip-172-31-86-150:~$
```

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.storageClass="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)

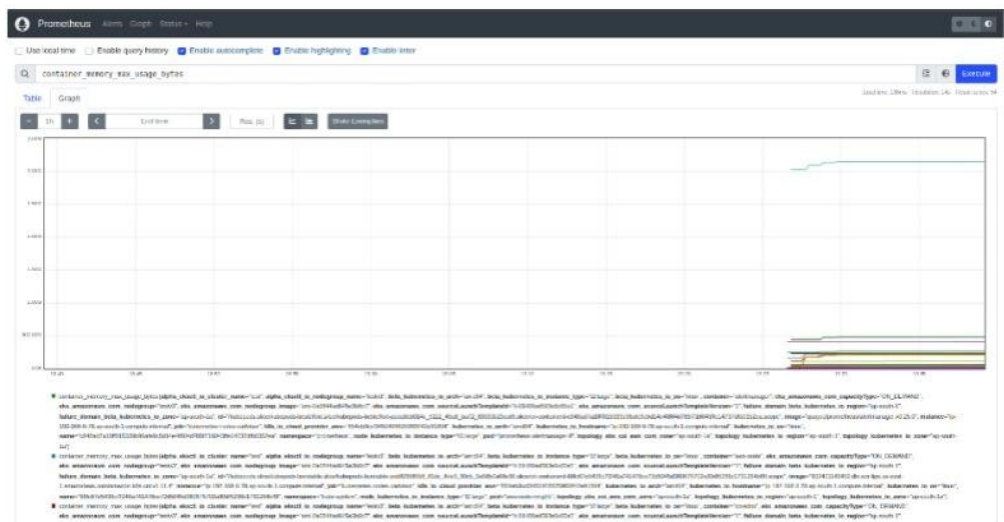
```
eksctl create addon --name aws-ebs-csi-driver --cluster <cluster name> --service-account-role-arn arn:aws:iam::<account id>:role/AmazonEKS_EBS_CSI_DriverRole --force
```

now see those pods are also running!

visit the prometheus server at localhost:9090

```
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 s s w o r d' \

--values grafana.yaml \

--set service.type=LoadBalancer
```

4. Check if Grafana is deployed properly

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

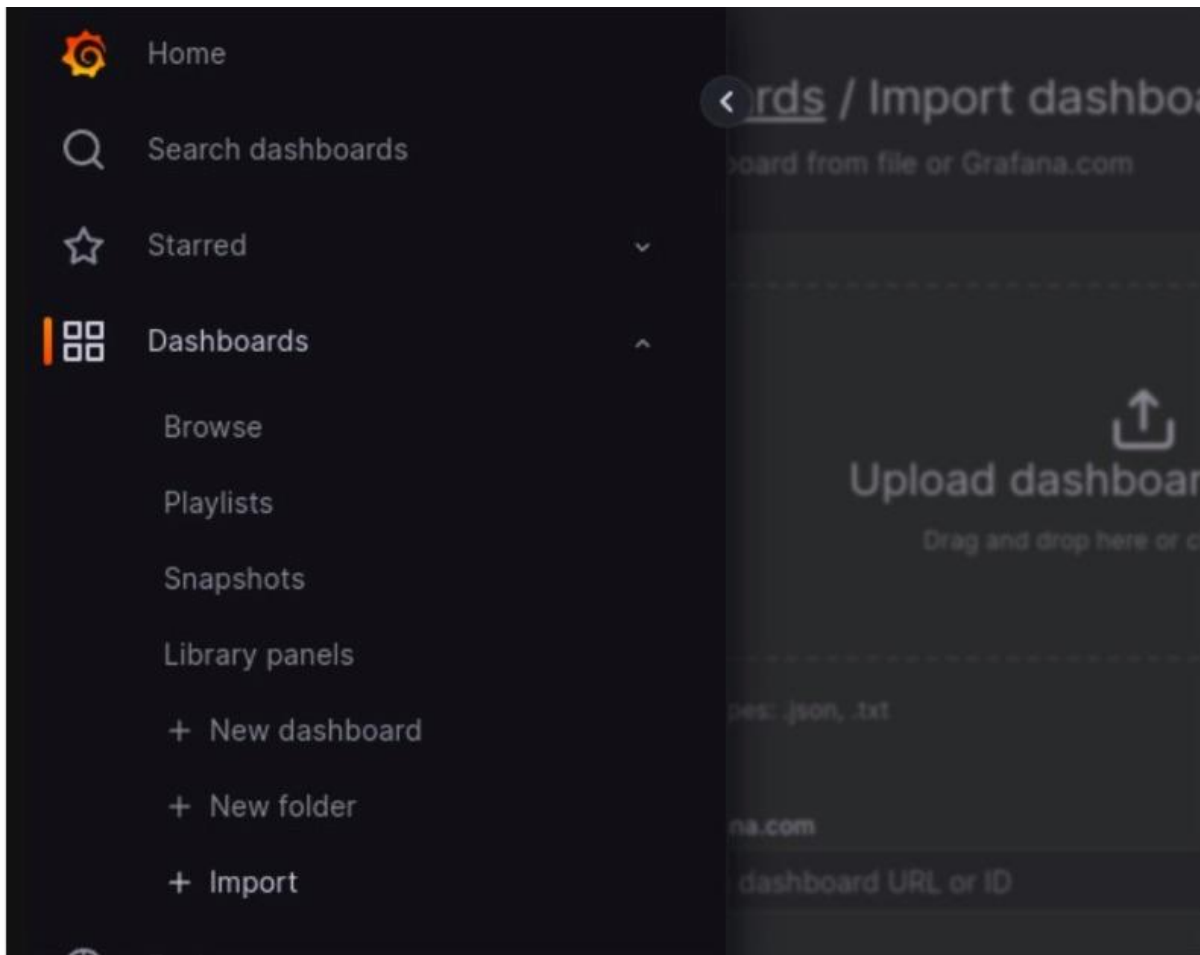
5. Get Grafana loadbalancer url

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

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

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

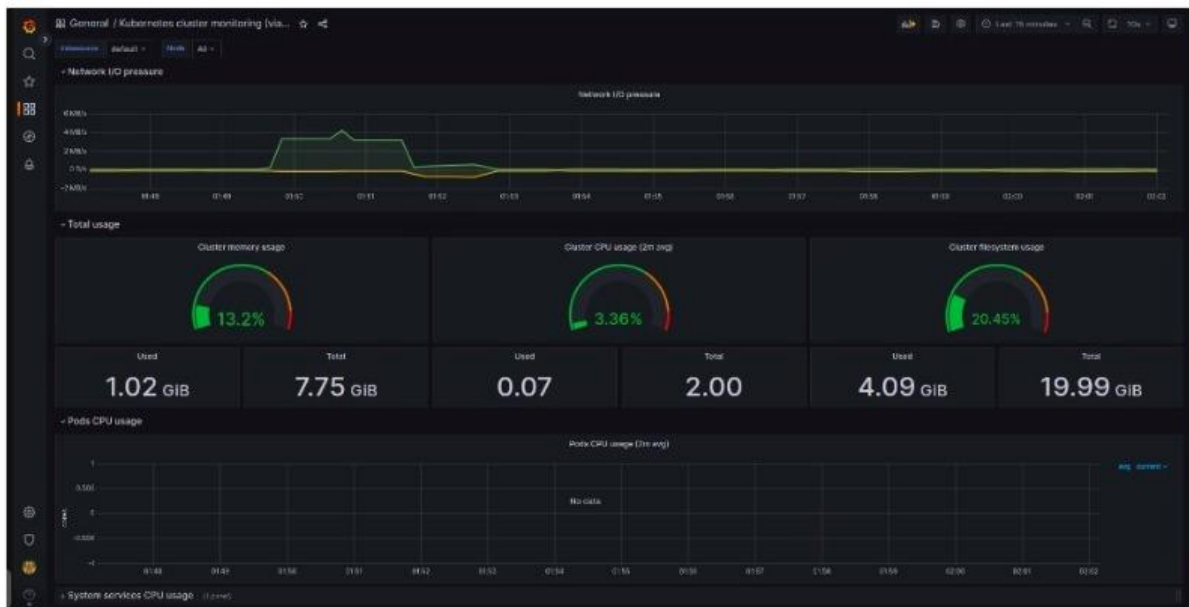
7. visit the url and login and import the dashboard:



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/>