



Proyecto Integrador DevOps

Grupo 4

Integrantes:

- Gabriel García
- Paolo Sinti
- Lucas Laborde
- Julio Vilaboa
- Juan C. Addamo

Índice

| | |
|---|----|
| Configuración de la instancia EC2 | 5 |
| Creación del clúster de Kubernetes en AWS con Eksctl..... | 10 |
| Crear instancia EC2 | 3 |
| Crear y configurar un clúster de Kubernetes en AWS utilizando el servicio Amazon EKS (Elastic Kubernetes Service). | 9 |
| Crear y configurar una instancia en EC2 (Elastic Compute Cloud). | 3 |
| Cuenta de usuario con el perfil para EKS | 9 |
| Desplegar dos pods con Nginx y configurar el servicio Load Balancer. | 13 |
| Despliegue de los pods de Nginx | 13 |
| Despliegue del servicio LoadBalancer..... | 14 |
| Eliminar el clúster..... | 23 |
| Instalación de AWS CLI v2 | 5 |
| Instalación de Eksctl..... | 7 |
| Instalación de Helm en la instancia EC2 | 8 |
| Instalación de Kubectl..... | 6 |
| Instalar el driver EBS CSI | 17 |
| Instalar la herramienta Prometheus en el clúster de Kubernetes..... | 16 |
| Instalar y configurar la herramienta Grafana | 19 |
| Instalar y configurar las herramientas de monitoreo en el clúster de Kubernetes. | 16 |
| Introducción..... | 3 |

Introducción

Este proyecto tiene como objetivo principal poner en práctica lo aprendido durante el curso y para ello se implementó un laboratorio que permitirá integrar las diferentes herramientas y tecnologías que fueron explicadas en las clases.}

El proyecto esta dividido en las siguientes etapas:

- Crear y configurar una instancia en EC2 (Elastic Compute Cloud).
- Crear y configurar un clúster de Kubernetes en AWS utilizando el servicio Amazon EKS (Elastic Kubernetes Service).
- Desplegar dos pods con Nginx y configurar el servicio Load Balancer.
- Instalar y configurar las herramientas de monitoreo en el clúster de Kubernetes.
- Eliminar el clúster.

Crear y configurar una instancia en EC2 (Elastic Compute Cloud).

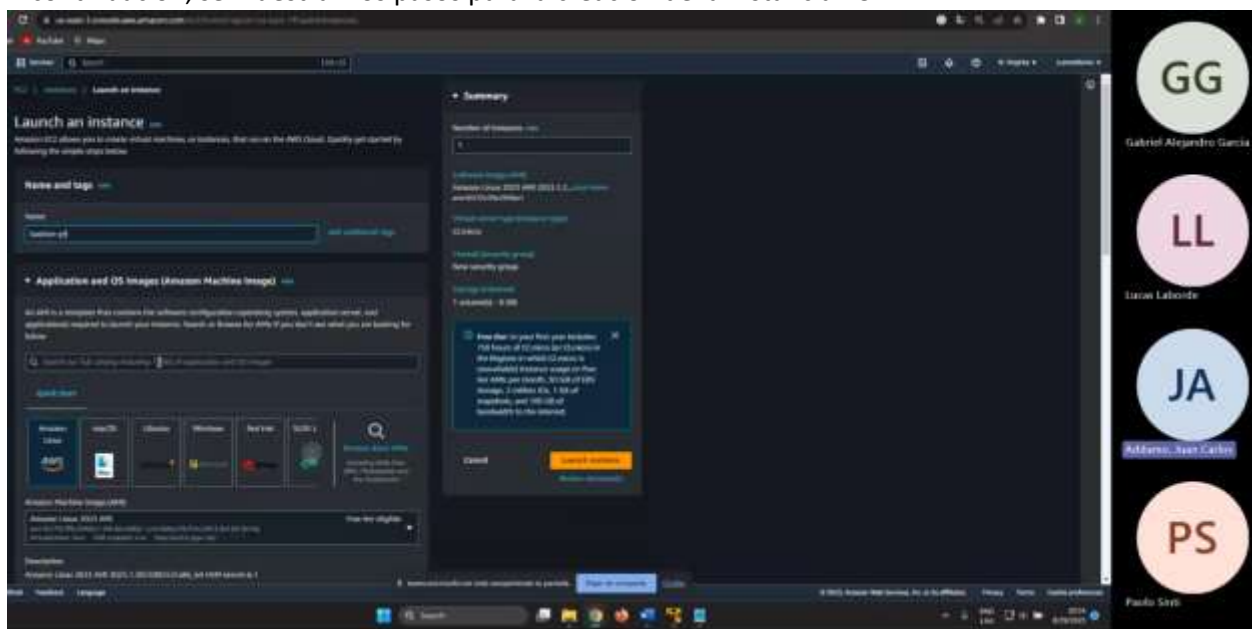
Crear instancia EC2

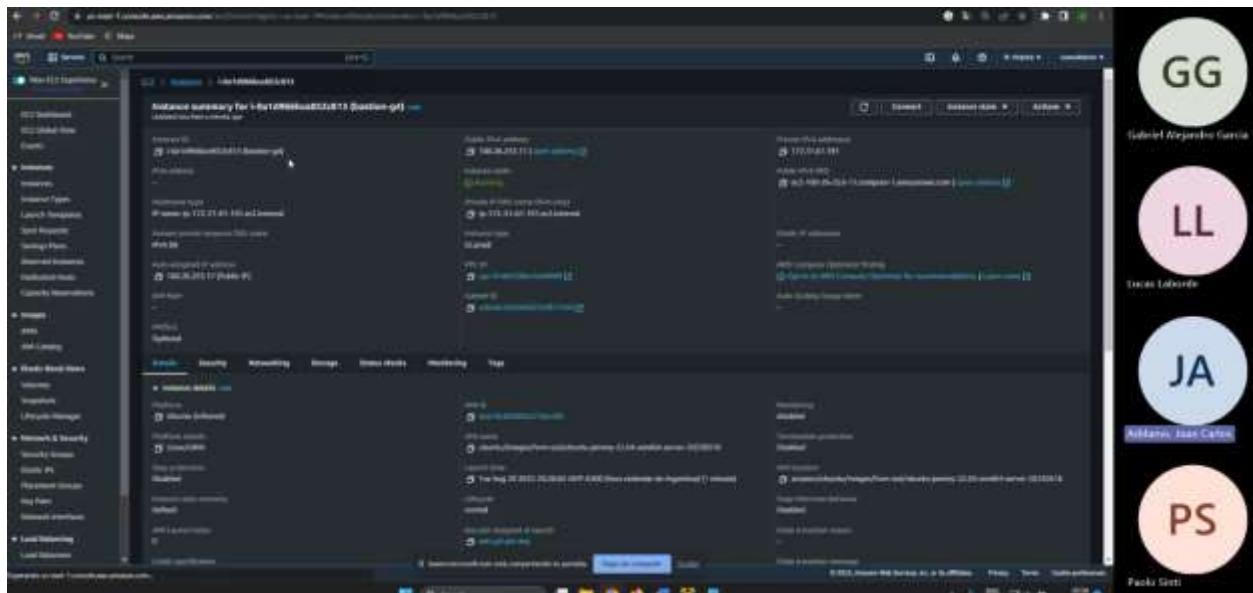
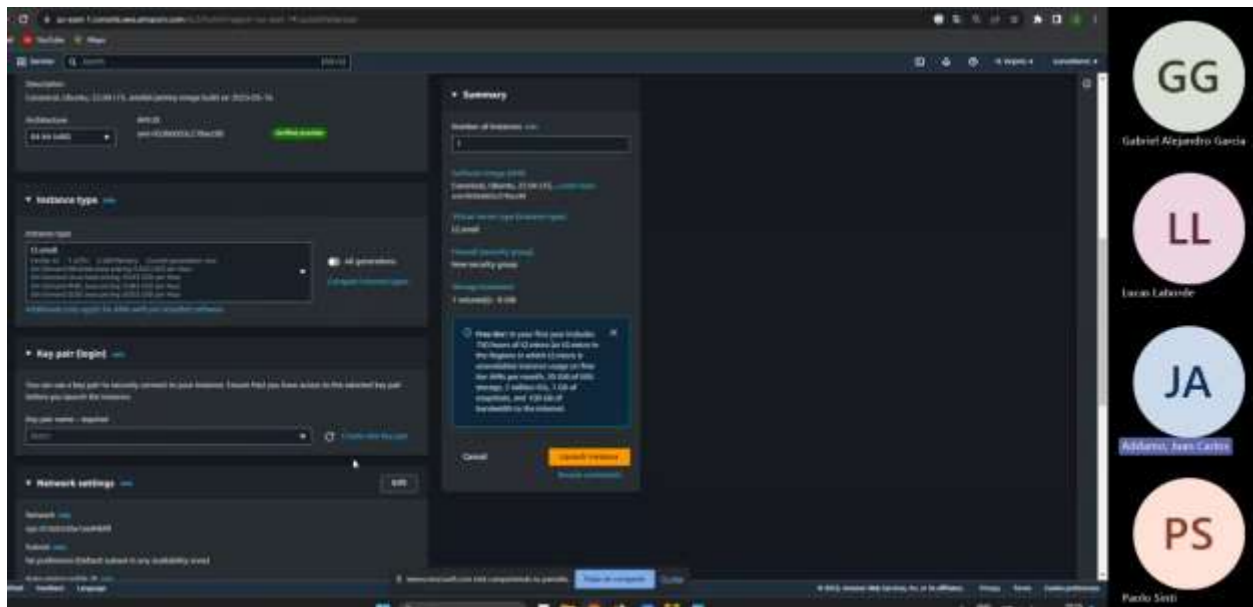
Se crea una instancia en EC2 la cuál será utilizada como bastión para la creación del cluster EKS. A continuación, se detallan las especificaciones de la instancia y los pasos para la creación de la instancia.

Datos de la instancia en EC2

- Nombre: bastion-g4
- Imagen de OS: Ubuntu server 22.04 – 64 bits
- Tipo: t2.small
- Key pair: se crea el par de claves que permitirán la conexión utilizando SSH.

A continuación, se muestran los pasos para la creación de la instancia EC2:

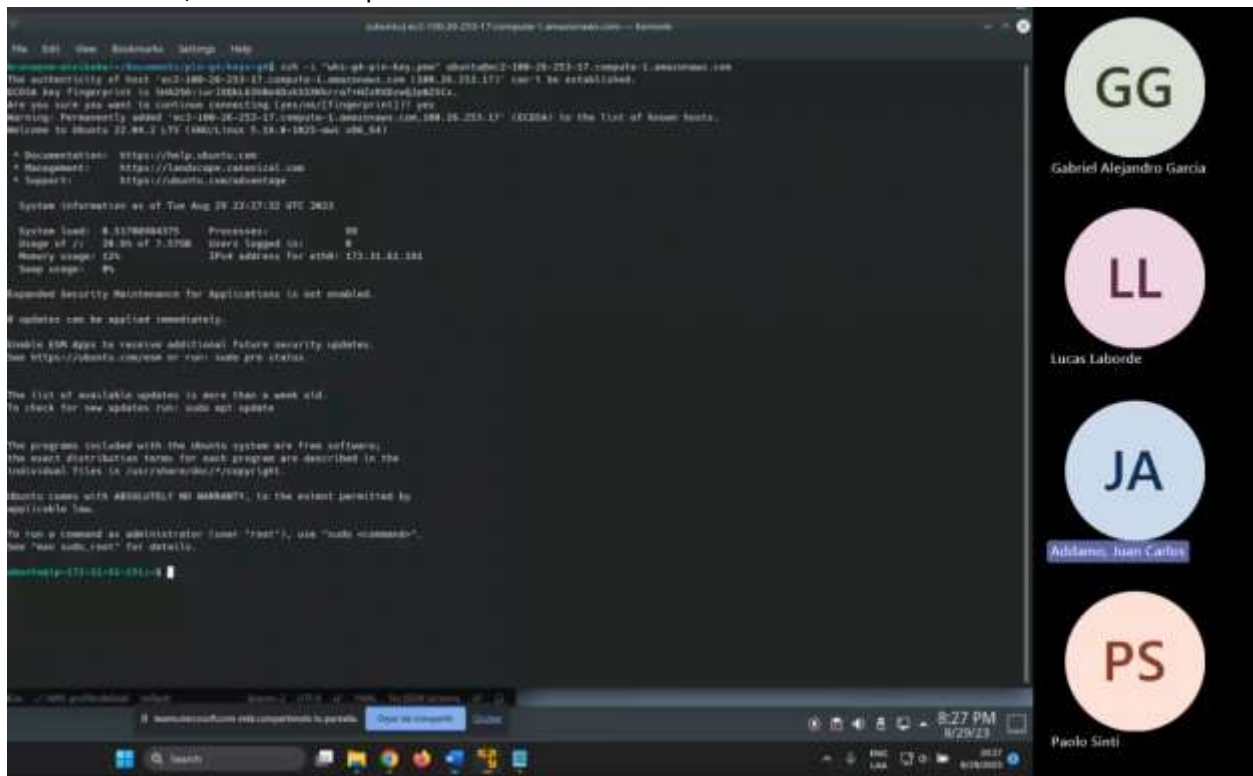




Luego de verificar que la instancia se creó correctamente, se ejecuta el siguiente comando desde una terminal Linux para conectarnos por SSH a la instancia EC2.

```
$ ssh -i "wks-g4-pin-key.pem" ubuntu@ec2-100-26-253-17.compute-1.amazonaws.com
```

A continuación, se muestra la pantalla de conexión a la instancia EC2:



Configuración de la instancia EC2

Luego de crear la instancia en EC2 se configuran las herramientas que permitirán crear y configurar el clúster EKS y sus componentes.

Instalación de AWS CLI v2

La herramienta AWS CLI v2 permite interactuar con los servicios de AWS mediante el uso de la línea de comandos.

Los comandos utilizados para la instalación de AWS CLI v2 fueron los siguientes:

```
$ sudo apt-get install unzip
```

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o  
"awscliv2.zip"
```

```
$ unzip awscliv2.zip
```

```
$ sudo ./aws/install
```

```
$ aws --version  
aws-cli/2.13.11 Python/3.11.4 Linux/5.19.0-1025-aws exe/x86_64.ubuntu.22  
prompt/off
```

[illegible]

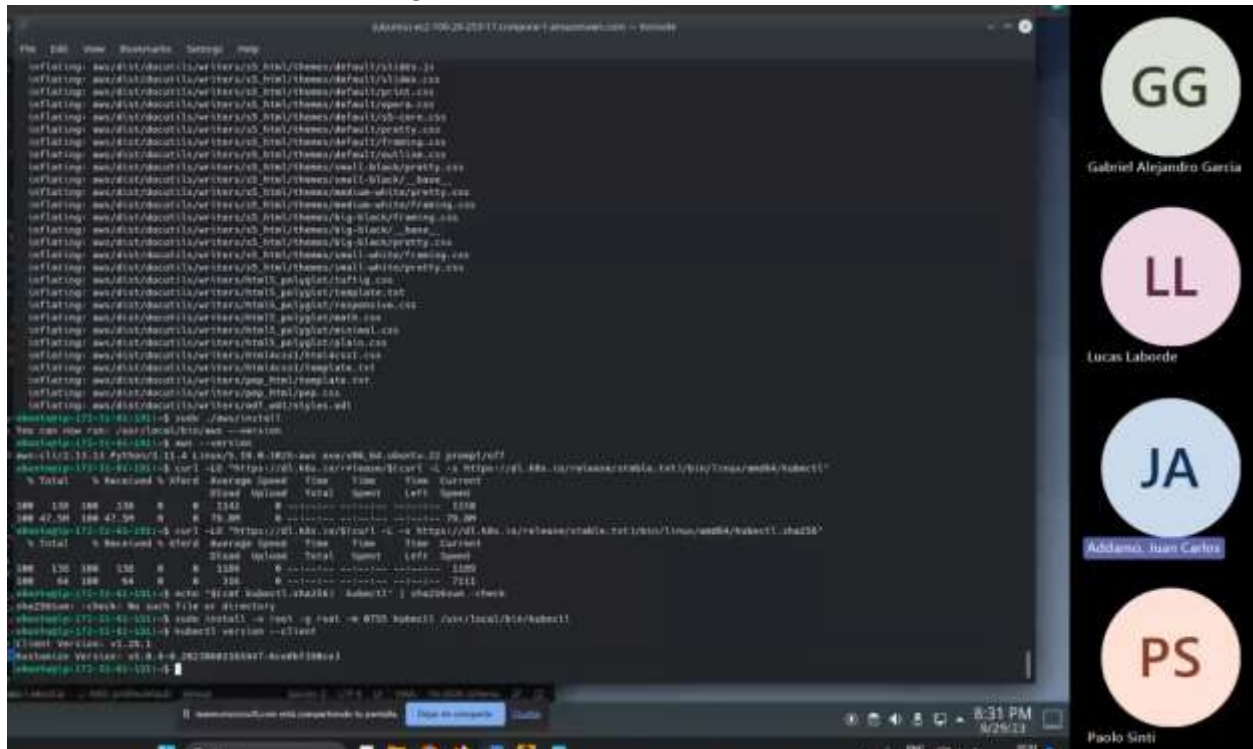
Kubectl es una interfaz de línea de comandos que permite ejecutar comando sobre un clúster de Kubernetes.

```
$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
$ echo "$(cat kubect1.sha256)    kubect1" | sha256sum --check
Ok
```

```
$ kubectl version --client
Client Version: v1.28.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bfb390ce3
```

A continuación, se muestra la imagen de la versión de Kubectl instalada en la instancia EC2:



Instalación de Eksctl

Eksctl es una herramienta de línea de comandos que permite crear, configurar y administrar un clúster de Kubernetes instalado en la plataforma de AWS.

Los comandos utilizados para la instalación de Eksctl fueron los siguientes:

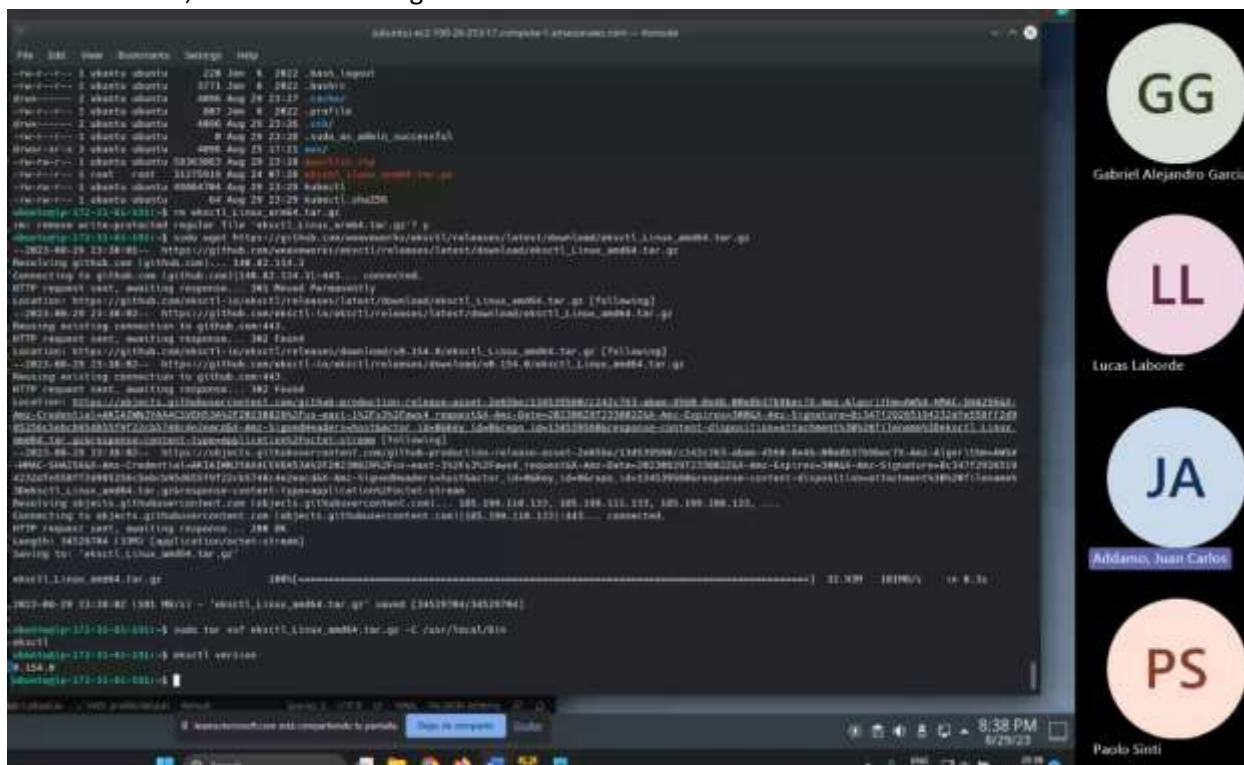
```
$ sudo wget
```

```
https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_Linux_amd64.tar.gz
```

```
$ sudo tar xvf eksctl_Linux_amd64.tar.gz -C /usr/local/bin
```

```
$ eksctl version  
0.154.0
```


A continuación, se muestra la imagen de la versión de Eksctl instalada en la instancia EC2:



Instalación de Helm en la instancia EC2

Para instalar las herramientas de monitoreo es necesario trabajar con la herramienta Helm la cual permite administrar las aplicaciones del clúster de Kubernetes.

Para ello se ejecutaron los siguientes comandos en la instancia EC2:

```
$ curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null
```

```
$ sudo apt-get install apt-transport-https --yes
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install helm
```

```
$ helm version
version.BuildInfo{Version:"v3.12.3",
GitCommit:"3a31588ad33fe3b89af5a2a54eeld25bfe6eaa5e", GitTreeState:"clean",
GoVersion:"go1.20.7"}
```


Crear y configurar un clúster de Kubernetes en AWS utilizando el servicio Amazon EKS (Elastic Kubernetes Service).

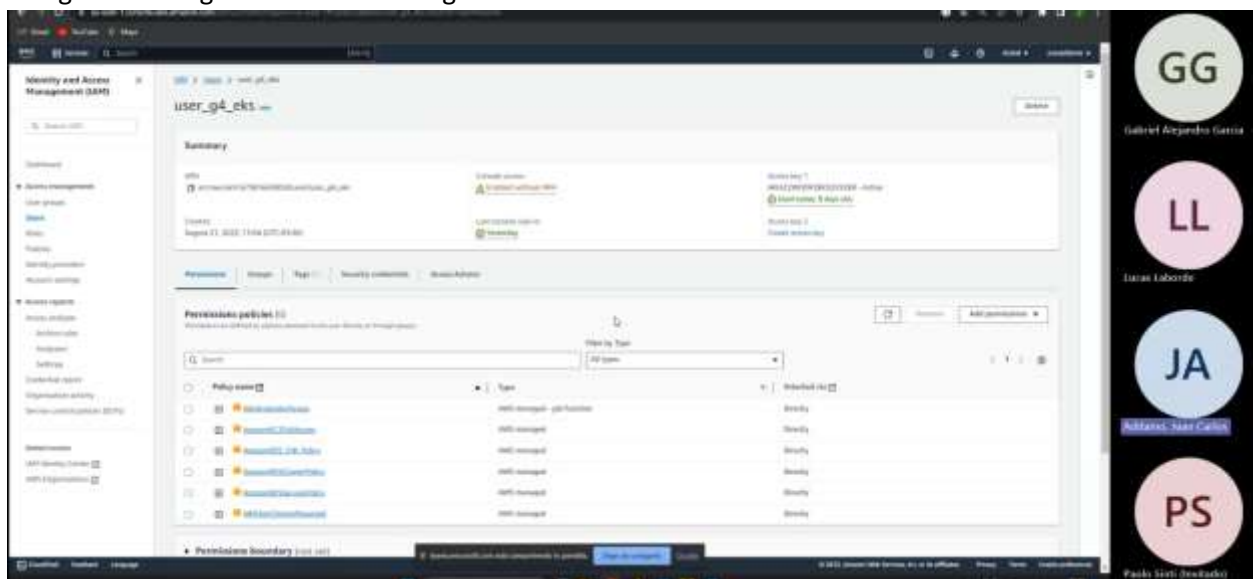
Para crear y configurar un clúster de Kubernetes vamos a utilizar las herramientas instaladas anteriormente y además vamos a crear una cuenta de usuario para tal motivo.

Cuenta de usuario con el perfil para EKS

Se crea una cuenta de usuario, la cual posee los siguientes datos:

- Nombre de la cuenta de usuario: user_g4_eks
- Pass: Autogenerada
- Se habilita el acceso por consola para la cuenta user_g4_eks:
<https://xxxxxxx0024.signin.aws.amazon.com/console>

La siguiente imagen muestra la configuración de la cuenta de usuario:



Luego, se crean las claves programáticas (“Access Key”) para la cuenta de usuario “user_g4_eks”.

The screenshot shows the AWS IAM console interface for adding an access key. The 'Use case' section includes options like 'Command Line Interface (CLI)', 'Local code', 'Application running on an AWS compute service', 'Third-party service', 'Application running outside AWS', and 'Other'. The 'Alternatives recommended' section suggests using 'AWS CloudShell' or 'AWS CLI V2'. The 'Confirmation' section has a checked box for 'I understand the above recommendation and want to proceed to create an access key'. The 'Next' button is highlighted in orange.

Una vez creadas las claves programáticas, se configura el cliente de AWS con la información obtenida, para ello se ejecuta el siguiente comando:

```
$ aws configure
AWS Access Key ID [None]: *****
AWS Secret Access Key [None]: *****
Default region name [None]: us-east-1
Default output format [None]: json
```

Creación del clúster de Kubernetes en AWS con Eksctl

Para la creación del clúster se creo un archivo YAML, con el siguiente contenido:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: g4-k8s # nombre del cluster
  version: "1.27" # version del cluster
  region: us-east-1 # region en donde se crea el cluster

iam:
  withOIDC: true

managedNodeGroups:
  - name: g4-k8s-nm
```

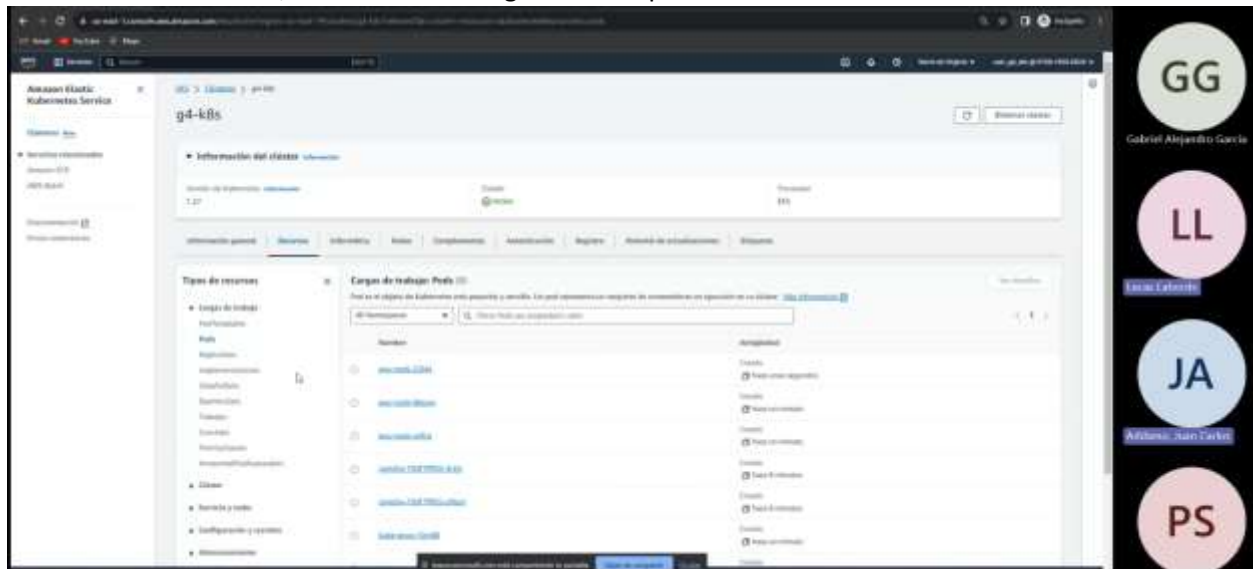

[illegible]

```
$ kubectl get nodes
```

La siguiente imagen muestra la información de los comandos ejecutados:

[illegible]

Asimismo, se ingresa a la consola de AWS con el usuario user_g4_eks y se verifica la implementación del clúster. A continuación, se muestra la imagen de la implementación del clúster:



Desplegar dos pods con Nginx y configurar el servicio Load Balancer.

En esta se realiza el despliegue de los pods y del servicio LoadBalancer en el clúster de Kubernetes creado anteriormente.

Despliegue de los pods de Nginx

Se realiza el despliegue de dos pods de Nginx que serán accedidos desde Internet en el clúster de Kubernetes recién creado, para ello se crea un archivo YAML con el siguiente contenido:

```
# nginx-deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

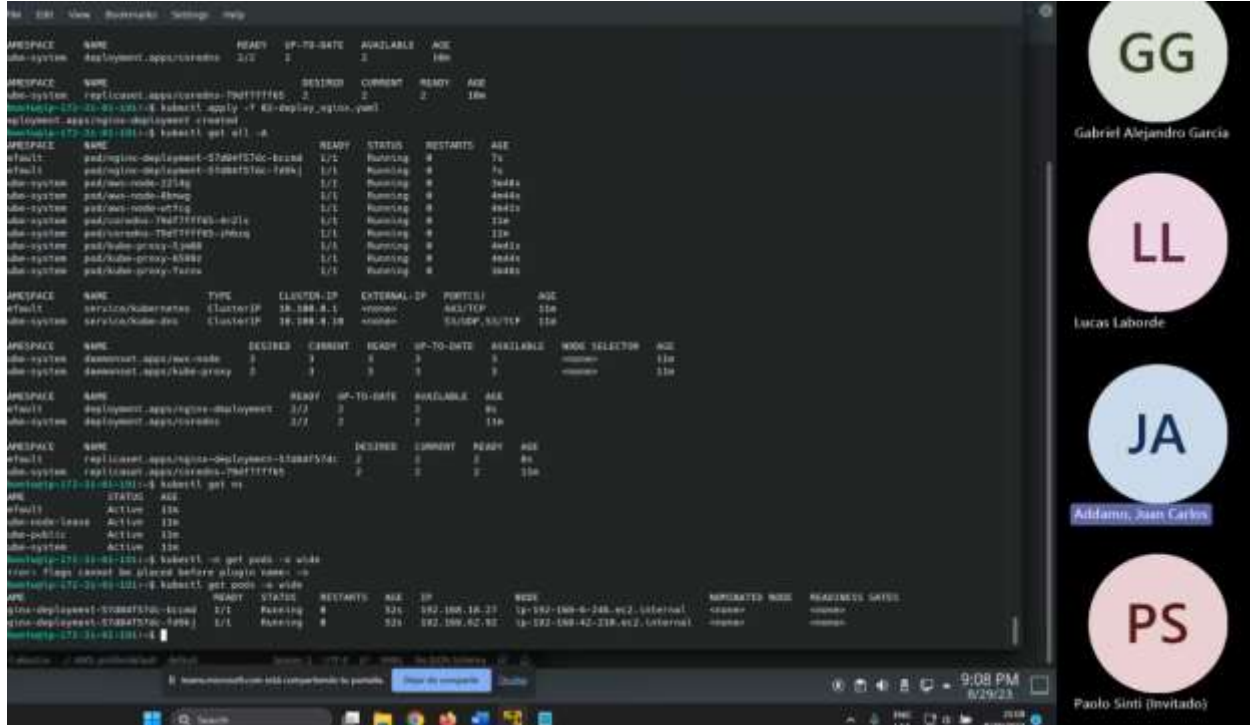
Luego se ejecuta el siguiente comando:

```
$ kubectl apply -f 02-deploy_nginx.yaml
```

Para verificar el despliegue se ejecuta el siguiente comando:

```
$ kubectl get all -A
```

En la siguiente imagen se pueden ver los detalles del despliegue:




Despliegue del servicio LoadBalancer

Para que los pods puedan ser accedidos desde Internet se crea un archivo YAML con el siguiente contenido:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

Luego de crear el archivo se ejecuta el siguiente comando:

```
$ kubectl apply -f 03-deploy_lbsevice.yaml
```


[illegible]

Welcome to nginx! x +

← → ↻ a17bb71e55d4c4519952cbf64ee57c70-536766060.us-east-1.elb.amazonaws.com ☆

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Instalar y configurar las herramientas de monitoreo en el clúster de Kubernetes.

Las herramientas de monitoreo que se instalaron en el clúster de Kubernetes creado anteriormente fueron Prometheus, la cual recopila métricas, las almacena y las pone a disposición para consultas y además envía alertas basadas en estas métricas y Grafana permite la visualización de las métricas recopiladas por Prometheus.

Instalar la herramienta Prometheus en el clúster de Kubernetes

Para realizar la instalación de Prometheus en el clúster, se crea el “namespace” prometheus y se realiza el despliegue de los pods dentro del “namespace”.

A continuación, se muestran los comandos de ejecutados:

```
$ kubectl create namespace prometheus
```

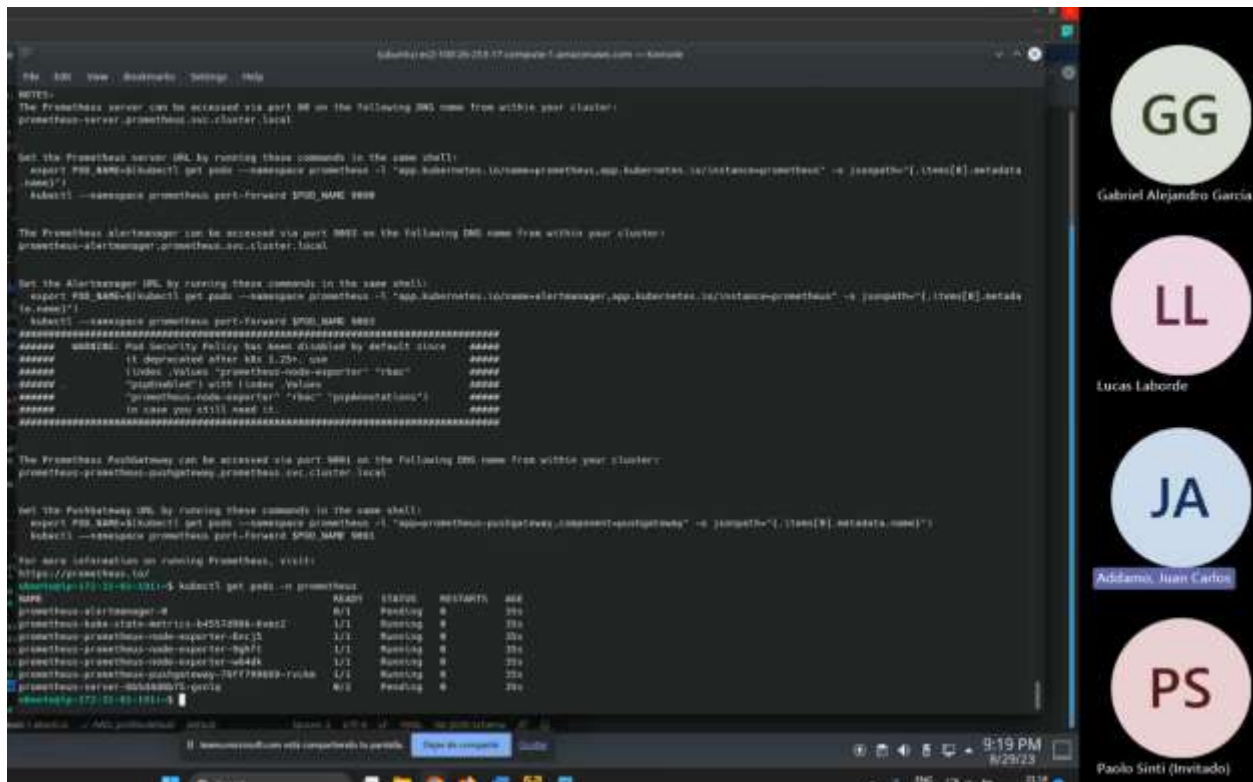
```
$ helm repo add prometheus-community https://prometheus-  
community.github.io/helm-charts
```

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

Verificamos que se hayan desplegado los pods dentro del “namespace” llamado prometheus, con el siguiente comando:

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

A continuación, se muestra la captura de pantalla en donde se pueden ver los pods de prometheus desplegados, en estado “Pending” dado que es necesario instalar el controlador EBS (Elastic Block Store) CSI (Container Storage Interface).



Instalar el driver EBS CSI

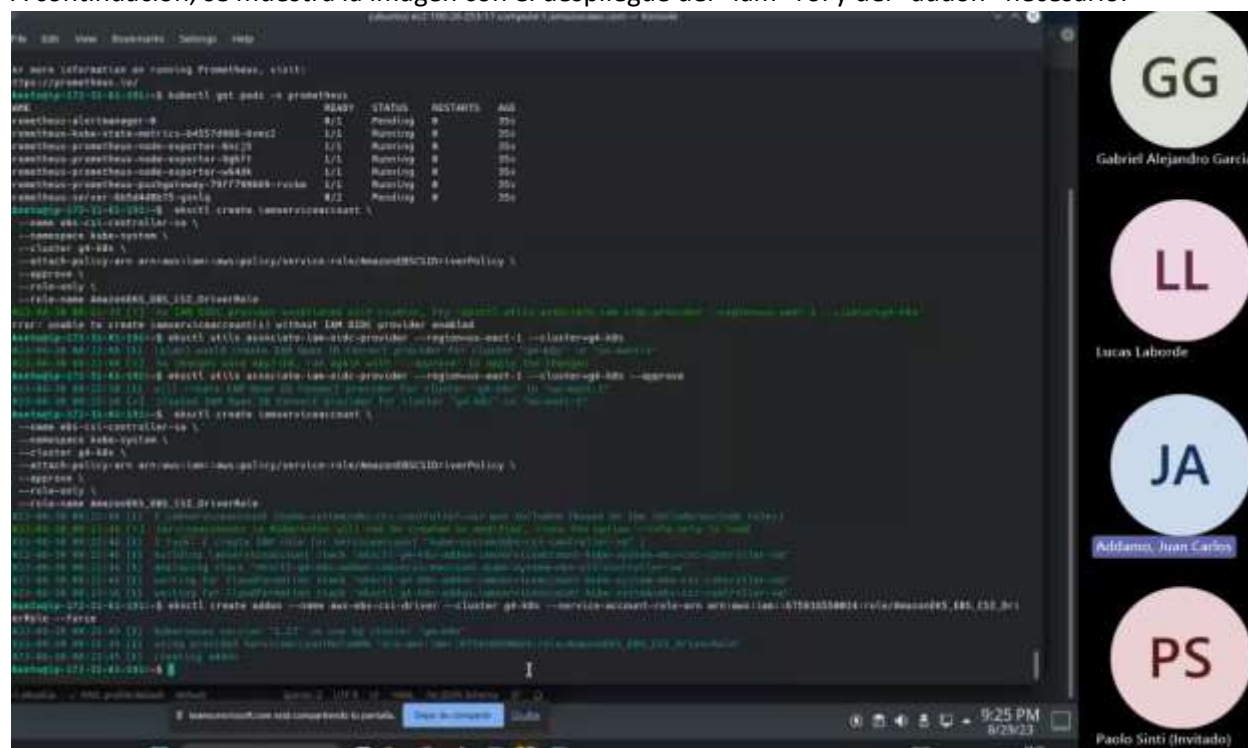
Para resolver el problema del estado “Pending” de los pods de Prometheus es necesario instalar el driver EBS (Elastic Block Store) CSI (Container Storage Interface). Este driver es una interfaz de almacenamiento de contenedores que le permite al clúster creado administrar los volúmenes.

Para instalar el driver se ejecutaron los siguientes comandos:

```
$ eksctl create iamserviceaccount \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster g4-k8s \
  --attach-policy-arn arn:aws:iam::aws:policy/service-
role/AmazonEBSCSIDriverPolicy \
  --approve \
  --role-only \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

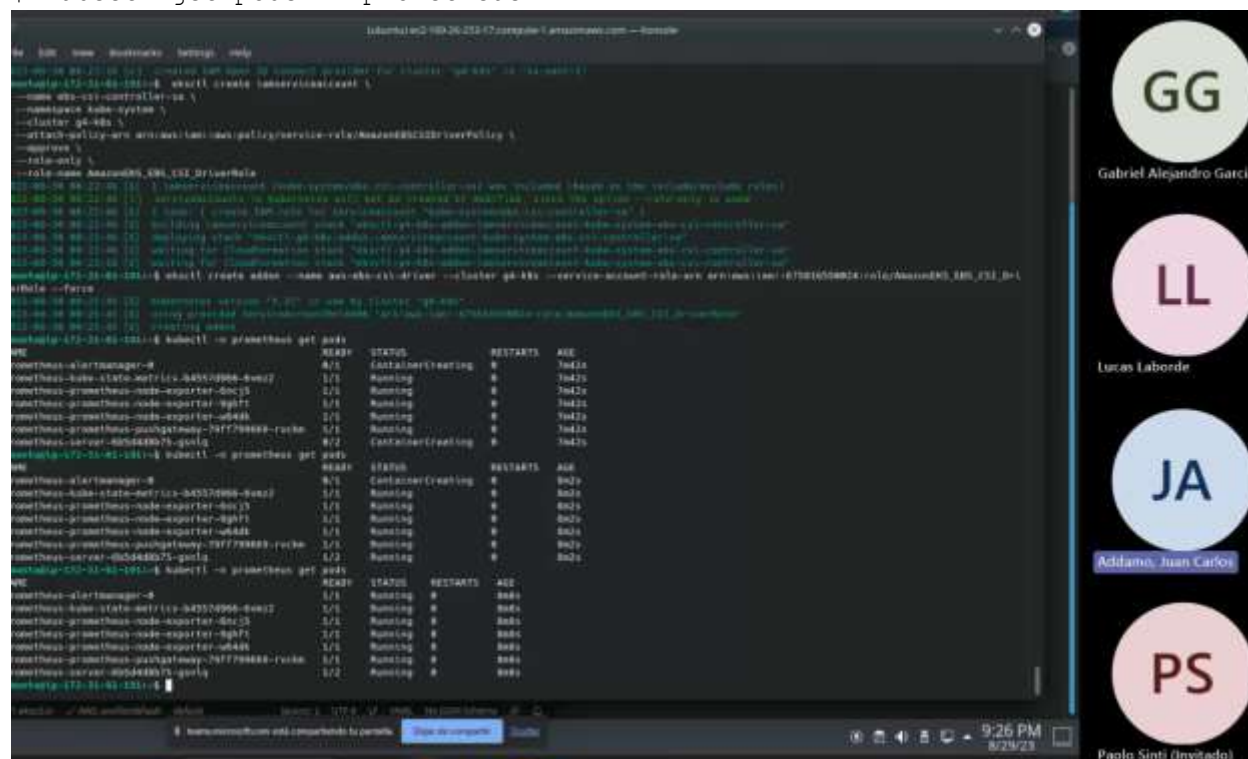
```
$ eksctl create addon --name aws-ebs-csi-driver --cluster g4-k8s --service-
account-role-arn arn:aws:iam::675816550024:role/AmazonEKS_EBS_CSI_DriverRole
--force
```

A continuación, se muestra la imagen con el despliegue del “iam” rol y del “addon” necesario:



Verificamos que el estado de todos los pods dentro del “namespace” llamado prometheus es “running”:

\$ kubectl get pods -n prometheus



Instalar y configurar la herramienta Grafana

Para realizar la instalación de Grafana en el clúster se crea el “namespace” grafana y se realiza el despliegue de los pods dentro del “namespace”.

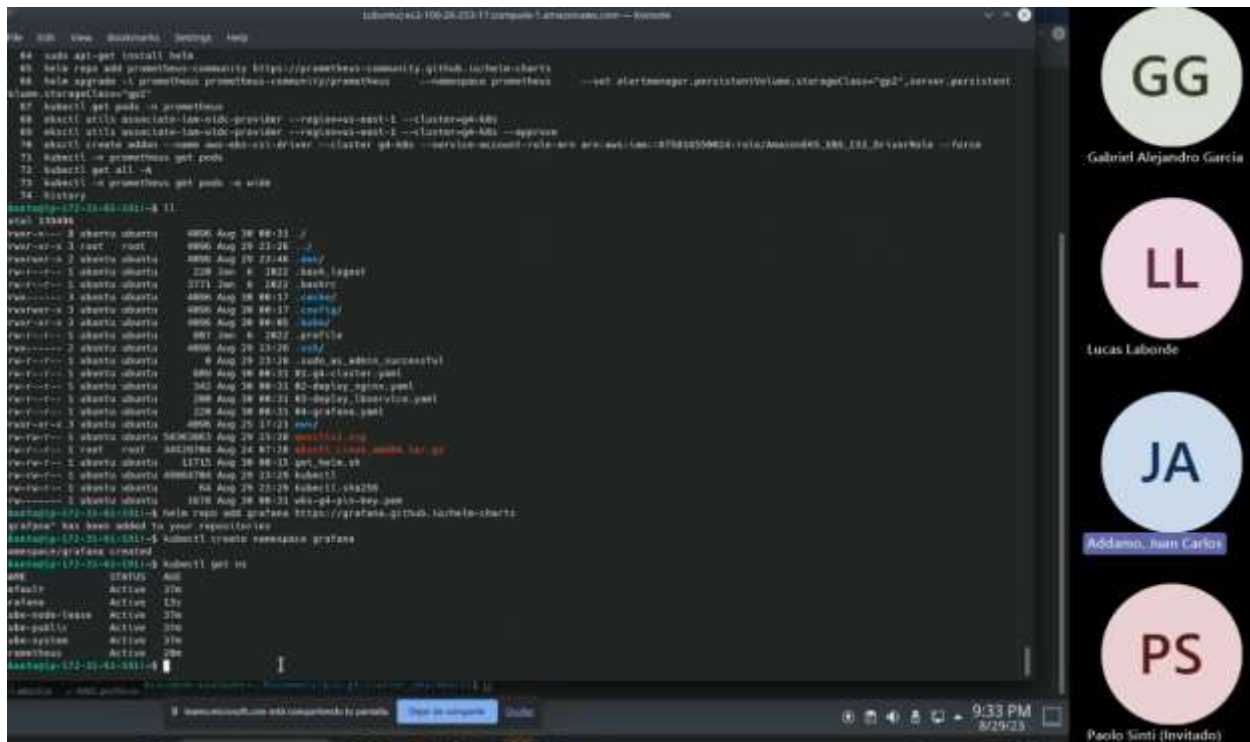
A continuación, se muestran los comandos de ejecutados:

```
$ kubectl create namespace grafana
```

```
$ helm repo add grafana https://grafana.github.io/helm-charts
```

Se verifica que existan el “namespace” con el comando:

```
$ kubectl get ns
```



Luego de ejecutar los comandos se crea un archivo YAML con el siguiente contenido:

datasources:

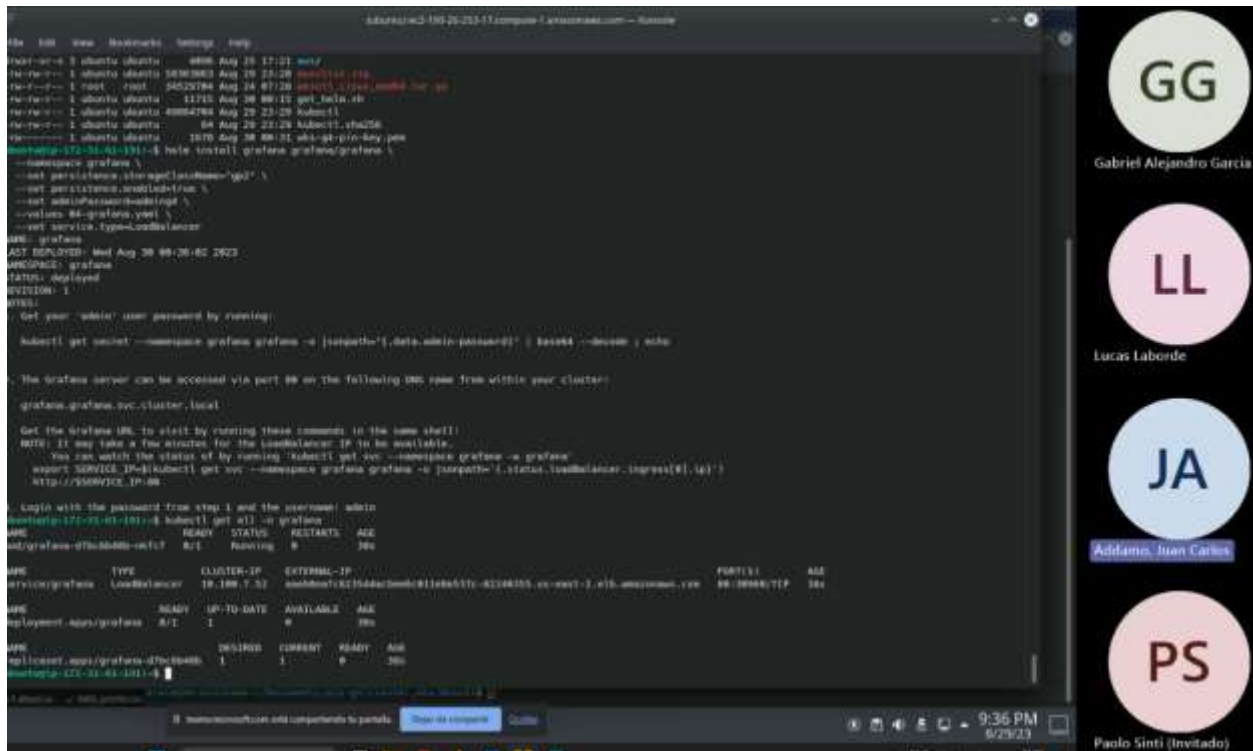
```
datasources.yaml:
  apiVersion: 1
  datasources:
  - name: Prometheus
    type: prometheus
    url: http://prometheus-server.prometheus.svc.cluster.local
    access: proxy
    isDefault: true
```

A continuación, se ejecuta el siguiente comando para implementar Grafana en el clúster:

```
$ helm install grafana grafana/grafana \
--namespace grafana \
--set persistence.storageClassName="gp2" \
--set persistence.enabled=true \
--set adminPassword=admin4 \
--values 04-grafana.yaml \
--set service.type=LoadBalancer
```

Finalizado el comando anterior, se muestra la pantalla con el resultado del comando:

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



The screenshot shows a terminal window with the following output:

```
NAME: grafana
LAST DEPLOYED: Wed Aug 30 00:26:02 2023
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 30 on the following DNS name from within your cluster:
  grafana.grafana.svc.cluster.local

3. 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 -o grafana
  export SERVICE_IP=$(kubectl get svc --namespace grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
  http://$SERVICE_IP:30

4. Login with the password from step 1 and the username: admin
kubectl exec -it $(kubectl get pods -n grafana -o jsonpath='{.items[0].metadata.name}') -- /bin/bash
NAME                READY   STATUS    RESTARTS   AGE
pod/grafana-00000000000000000000  1/1     Running   0           30s
```

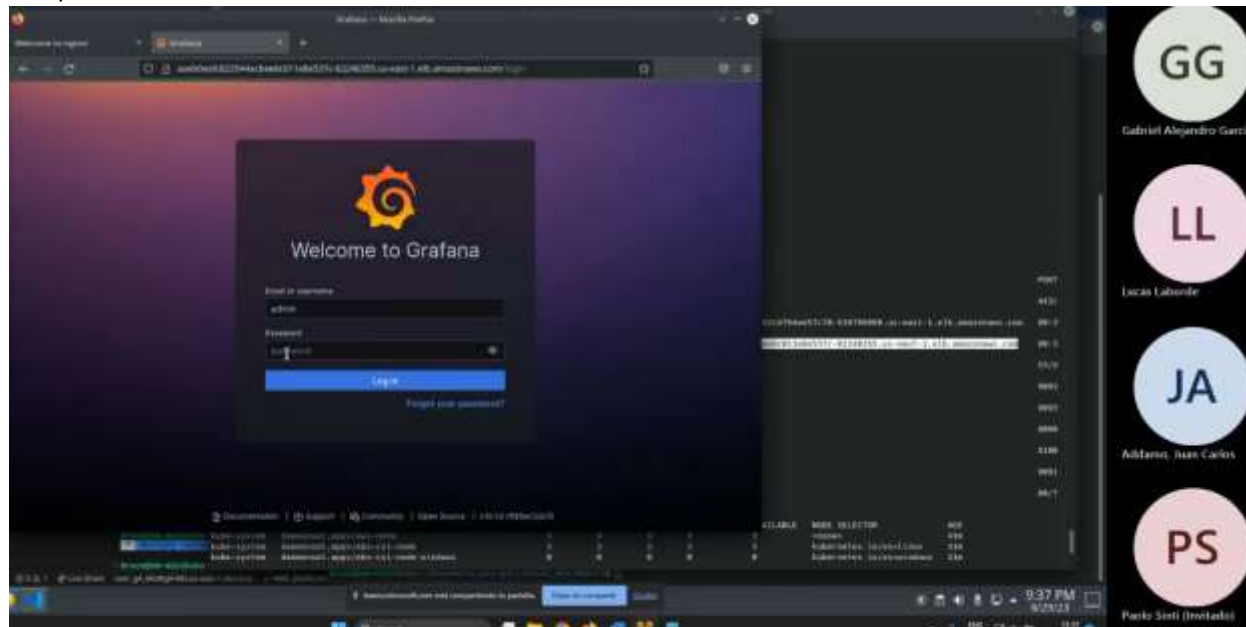
| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PARTITION | AGE |
|------------------|--------------|-------------|--|-----------|-----|
| services/grafana | LoadBalancer | 10.100.1.52 | ec2-34-202-173-110.compute-1.amazonaws.com | 0/1 | 30s |

| NAME | READY | UP-TO-DATE | AVAILABLE | AGE |
|-------------------------|-------|------------|-----------|-----|
| deployment.apps/grafana | 1/1 | 1 | 1 | 30s |

| NAME | READY | STATUS | RESTARTS | AGE |
|--|-------|---------|----------|-----|
| replicaset.apps/grafana-00000000000000000000 | 1 | Running | 0 | 30s |

The sidebar on the right shows four user avatars: GG (Gabriel Alejandro Garcia), LL (Lucas Laborde), JA (Addams, Juan Carlos), and PS (Paolo Sinti (Invitado)).

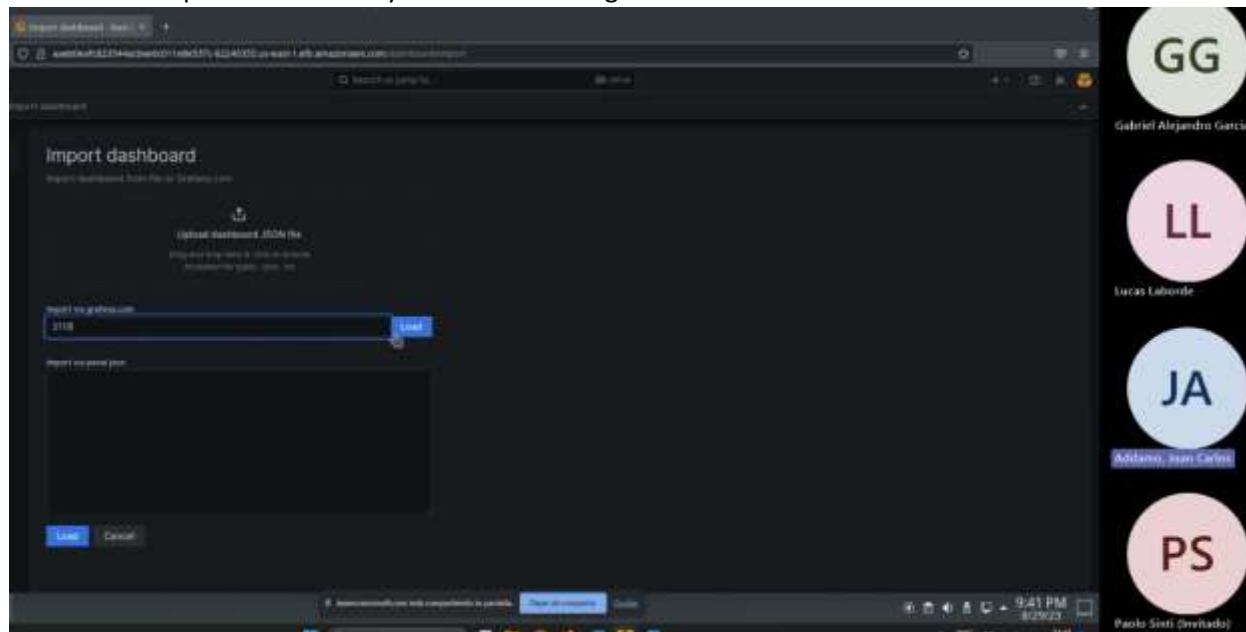
En el listado anterior se puede ver una dirección URL catalogada como “External IP” y dicha URL nos lleva a la pantalla de inicio de sesión de Grafana.



Para ingresar y configurar los tableros (dashboards) es necesario ingresar con una cuenta de usuario y una contraseña. En este caso utilizamos:

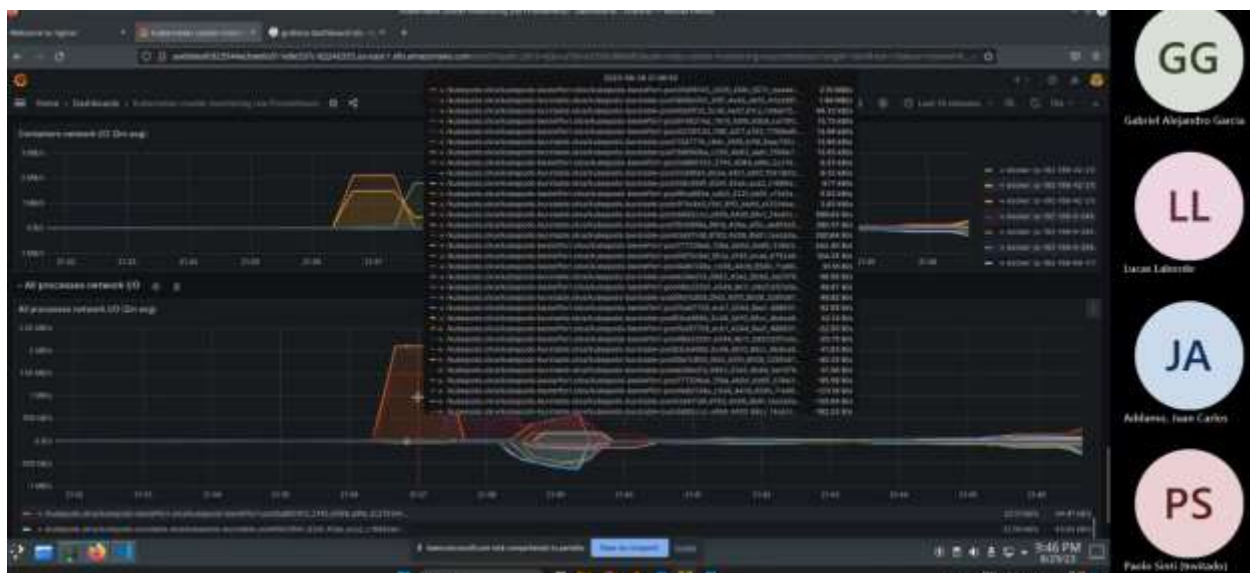
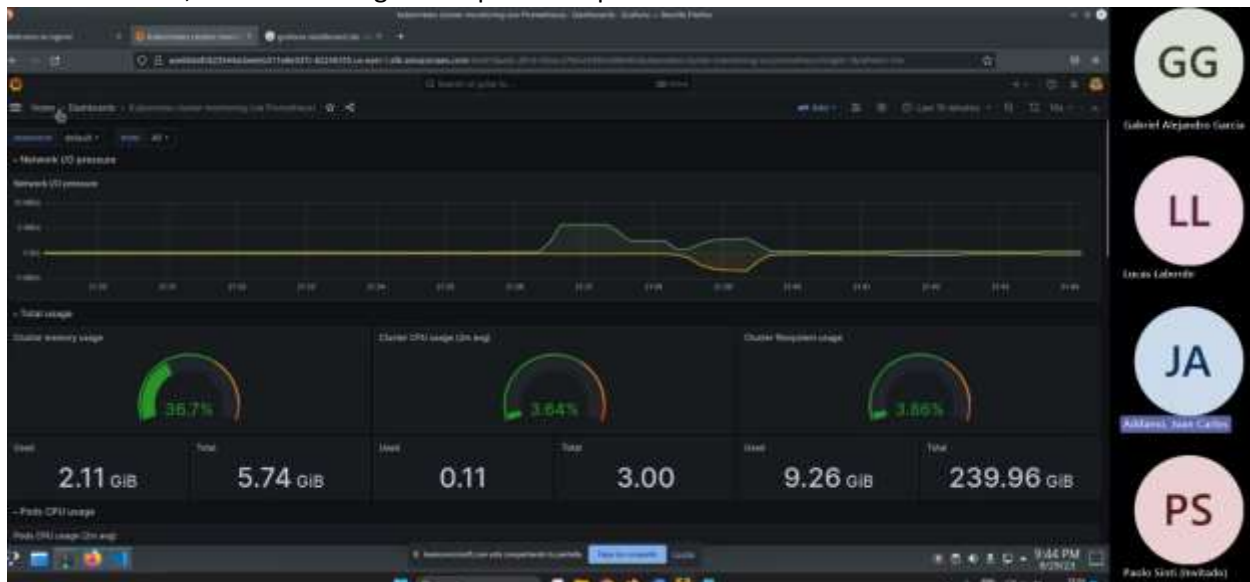
- Usuario: admin
- Contraseña: admining4

Luego de iniciar sesión en Grafana, se configura el dashboard con el código 3119 que esta configurado para el monitoreo de un clúster de Kubernetes. Para ello se ingresa en la opción “Dashboards”, se selecciona “Import dashboard” y se detalla el código mencionado anteriormente.



Una vez configurado el dashboard se puede ingresar al mismo y verificar los datos que están siendo recolectados. En este caso, se pueden ver métricas de uso de CPU, memoria, trafico de red y volúmenes.

A continuación, se muestran algunas capturas de pantalla con la información del clúster:



Una vez finalizado el laboratorio se procede a eliminar el clúster creado con el siguiente comando:

El comando anterior procede a eliminar el clúster por completo. A continuación, se muestra la captura de pantalla de los pasos de eliminación de recursos:

