PIN DevOps 2401

Repositorio Solución 1 - https://github.com/palacioea/Pin-2024 Repositorio Solución 2 - https://github.com/dnlbertoni/devops2401-pin

Grupo 01 Daniel Bertoni Emiliano Palacio

Contenido

[Enfoque de la Solución Nro: 1 2](#_Toc179824250)

[Resumen 2](#_Toc179824251)

[Creación de la Instancia EC2 en AWS 2](#_Toc179824252)

[Acceso SSH a la Instancia 4](#_Toc179824253)

[Instalación de Aplicaciones 5](#_Toc179824254)

[Creación del Cluster EKS 6](#_Toc179824255)

[Configuración de kubectl 8](#_Toc179824256)

[Instalación del EBS Driver 9](#_Toc179824257)

[Despliegue de Nginx 10](#_Toc179824258)

[Configuración de Helm y Despliegue de Prometheus 11](#_Toc179824259)

[Instalación de Grafana 11](#_Toc179824260)

[Monitoreo 12](#_Toc179824261)

[Limpieza de Recursos 14](#_Toc179824262)

[Enfoque de la Solución Nro: 2 16](#_Toc179824263)

[Resumen 16](#_Toc179824264)

[Creación de la Instancia EC2 en AWS 16](#_Toc179824265)

[Acceso SSH a la Instancia 18](#_Toc179824266)

[Aprovisionamiento de scripts 19](#_Toc179824267)

[Instalación de las aplicaciones 19](#_Toc179824268)

[Instalación del EKS 20](#_Toc179824269)

[Configuración de kubectl 20](#_Toc179824270)

[Instalación del EBS Driver 21](#_Toc179824271)

[Despliegue de Nginx 22](#_Toc179824272)

[Despliegue de Prometheus 24](#_Toc179824273)

[Instalación de Grafana 25](#_Toc179824274)

[Monitoreo 26](#_Toc179824275)

[Limpieza de Recursos 29](#_Toc179824276)

[Anexo I - Reaseguro de Limpieza 31](#_Toc179824277)

# Enfoque de la Solución Nro: 1

## Resumen

Vamos a desplegar el EC2 desde un pc de salto vía Terraform, además desplegaremos todas las herramientas que se solicitan para poder montar el EKS con un script en Shell para automatizar el despliegue.

Una vez implementado el EC2, procederemos a levantar el EKS desde línea de comandos desde el bastión previamente desplegado.

Terminada todas las capturas de pantalla, se procederá a borrar el EKS desde el bastion y luego con Terraform se destruirá el bastión construido.

Además, utilizaremos una herramienta Cloud-Nuke para poder eliminar también todos los demás recursos que se hayan creado si fuese necesario por otras vías.

## Creación de la Instancia EC2 en AWS

Utilizamos Terraform para aprovisionar una instancia EC2 en AWS. En el proceso, se generaron 16 recursos, incluyendo la creación del par de claves (KEY\_PAIR) necesario para establecer la conexión SSH. Verificamos exitosamente la creación de la instancia a través de la consola de AWS.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Acceso SSH a la Instancia

La clave PEM se descargó oportunamente, se logró la conexión SSH a la instancia de forma exitosa.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Instalación de Aplicaciones

Se creó un script que permite la instalación de todas las aplicaciones requeridas que se aprovisiono directamente desde Terraform.

Texto

Descripción generada automáticamente

## Creación del Cluster EKS

Iniciamos el proceso de creación del cluster EKS utilizando el siguiente comando:

eksctl create cluster \

--name eks-mundos-e \

--region us-east-1 \

--with-oidc \

--nodegroup-name testv3 \

--node-type t2.large \

--nodes 1 \

--nodes-min 1 \

--nodes-max 3 \

--node-volume-size 20 \

--ssh-access=false \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## Configuración de kubectl

Configuramos `kubectl` para interactuar con el cluster EKS recién creado utilizando el siguiente comando:  
**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

aws eks update-kubeconfig --name eks-mundos-e --region us-east-1

Verificación de la conexión con el cluster

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## Instalación del EBS Driver

Para habilitar el soporte de volúmenes EBS, aplicamos el controlador EBS driver con los siguientes comandos:  
Texto

Descripción generada automáticamente

kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/?ref=release-1.35

eksctl create iamserviceaccount \

--name ebs-csi-controller-sa \

--region us-east-1 \

--namespace kube-system \

--cluster eks-mundos-e \

--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 eks-mundos-e

--service-account-role-arn arn:aws:iam::xxxxxxxxxx:role/AmazonEKS\_EBS\_CSI\_DriverRole

--force

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Despliegue de Nginx

Para desplegar Nginx, aplicamos el manifiesto `nginx.yaml` usando el comando:  
**Texto

Descripción generada automáticamente**

kubectl apply -f nginx.yaml

Luego, verificamos el acceso a Nginx desde el navegador.

Texto

Descripción generada automáticamente

## Configuración de Helm y Despliegue de Prometheus

Agregamos los repositorios de Prometheus utilizando Helm:  
**Imagen que contiene Calendario

Descripción generada automáticamente**

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"

kubectl patch svc prometheus-server -n prometheus -p '{"spec": {"type": "NodePort"}}'

## Instalación de Grafana

Instalamos Grafana utilizando Helm, creando previamente el namespace:  
Texto

Descripción generada automáticamente

kubectl create namespace grafana

helm install grafana grafana/grafana \

--namespace grafana \

--set adminPassword='EKS!sAWSome' \

--values grafana.yaml \

--set service.type=LoadBalancer

## Monitoreo

Accedemos al external-ip

Imagen de la pantalla de un celular

Descripción generada automáticamente con confianza media

Agregamos Prometheus como datasource:

Patrón de fondo

Descripción generada automáticamente

importamos los dashboards 6417 y 3119 en la carpeta Dashboard y confiugramos el prometheus que creamos en el datasource.

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Verificamos importación de ambos

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

ID 6417:

Pantalla de un video juego

Descripción generada automáticamente con confianza media

ID 3119:

Captura de pantalla de un videojuego

Descripción generada automáticamente

## Limpieza de Recursos

Procedimos a limpiar todos los recursos desplegados para evitar costos innecesarios.

**Desde el bastion (EC2)**

helm uninstall prometheus --namespace prometheus

kubectl delete ns prometheus

helm uninstall grafana --namespace grafana

eksctl delete cluster --name mundos-e --region us-east-1

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Desde el pc de salto:**

terraform destroy -auto-approve

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

# Enfoque de la Solución Nro: 2

## Resumen

Vamos a desplegar el EC2 desde un pc de salto vía Terraform, además desplegaremos todas las herramientas que se solicitan para poder montar en el futuro el EKS con un script en Shell para automatizar el despliegue y gestionar el cluster.

Una vez implementado el EC2, procederemos a levantar el EKS, el Nginx y las aplicaciones de monitoreo con un script en bash aprovisionado también al EC2 desde Terraform y que solo reste aplicarlo desde el bastión.

Cade destacar que cambia es la manera de aprovisionar, ya que en el método anterior usamos **user\_data** y con este enfoque estamos usando **file** y **remote-exec** como provisioner desde Terraform

Terminada todas las capturas de pantalla, se procederá a borrar el EKS desde el bastión y luego con Terraform se destruirá el bastión construido.

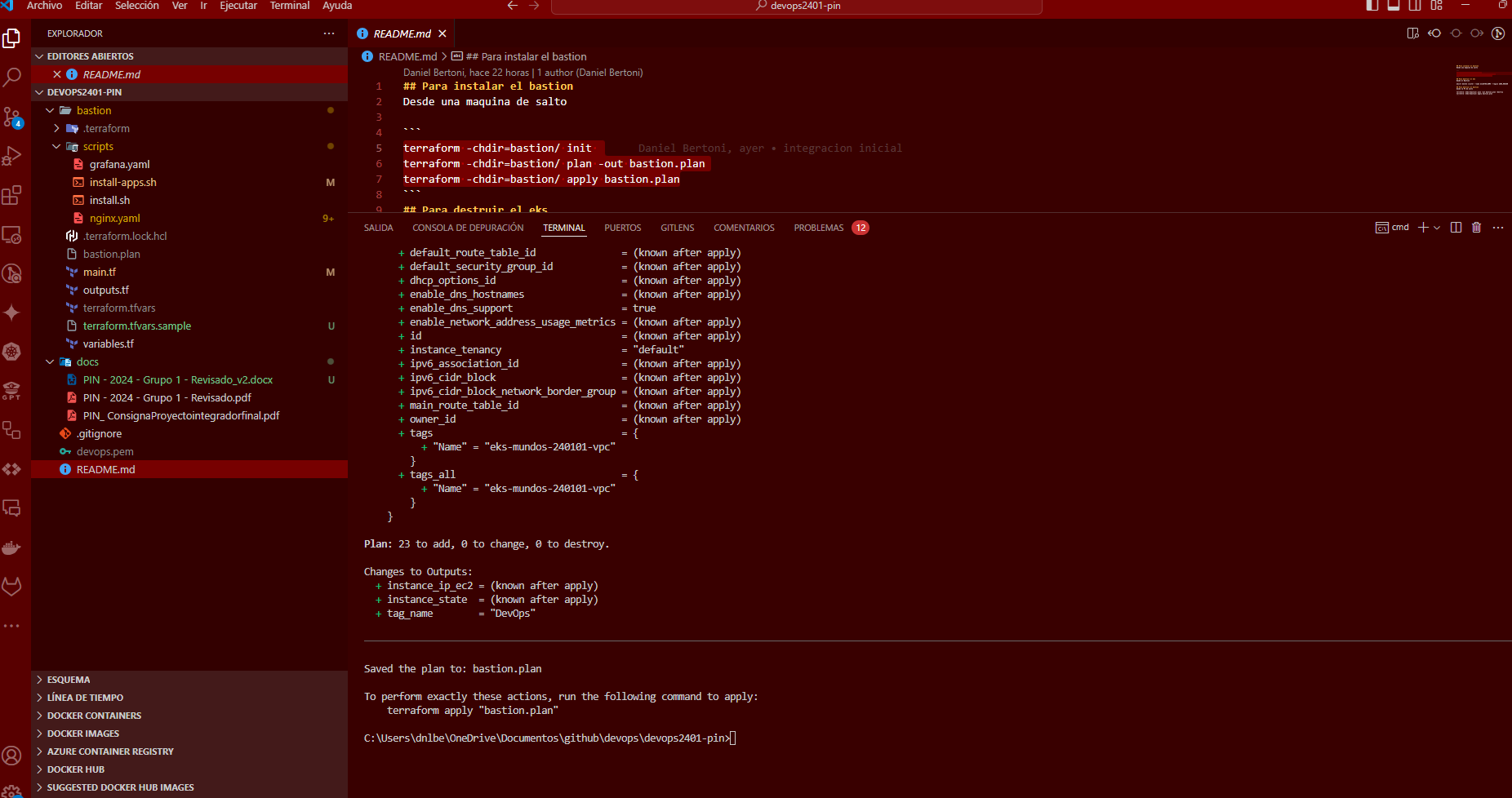
## Creación de la Instancia EC2 en AWS

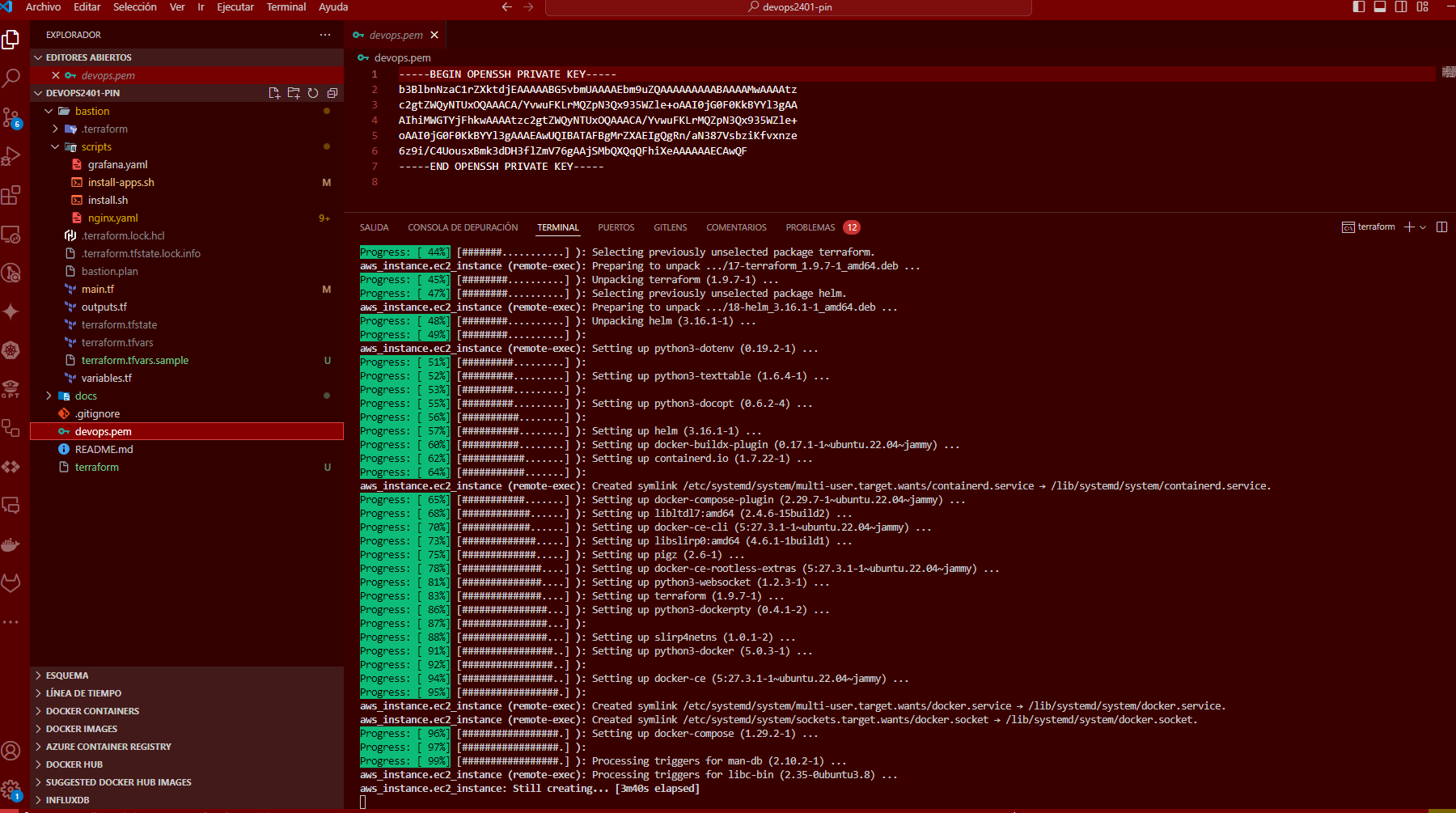
Utilizamos Terraform para aprovisionar una instancia EC2 en AWS. En el proceso, se generaron 16 recursos, incluyendo la creación del par de claves (KEY\_PAIR) necesario para establecer la conexión SSH.

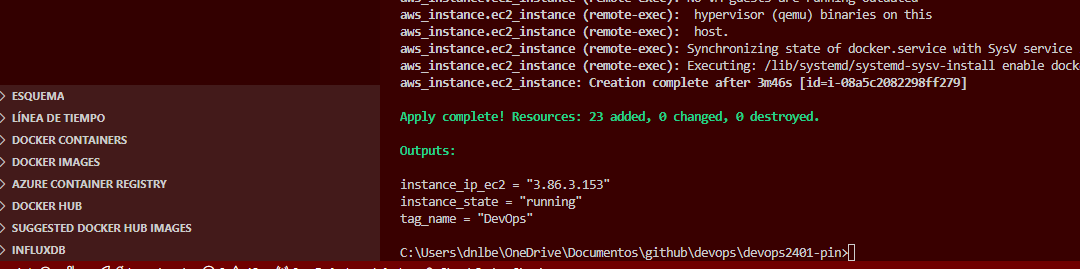
terraform -chdir=bastion/ init

terraform -chdir=bastion/ plan -out bastion.plan

terraform -chdir=bastion/ apply bastion.plan

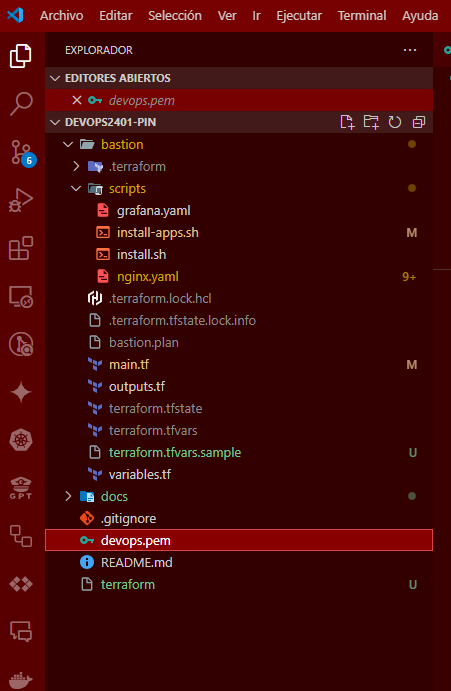


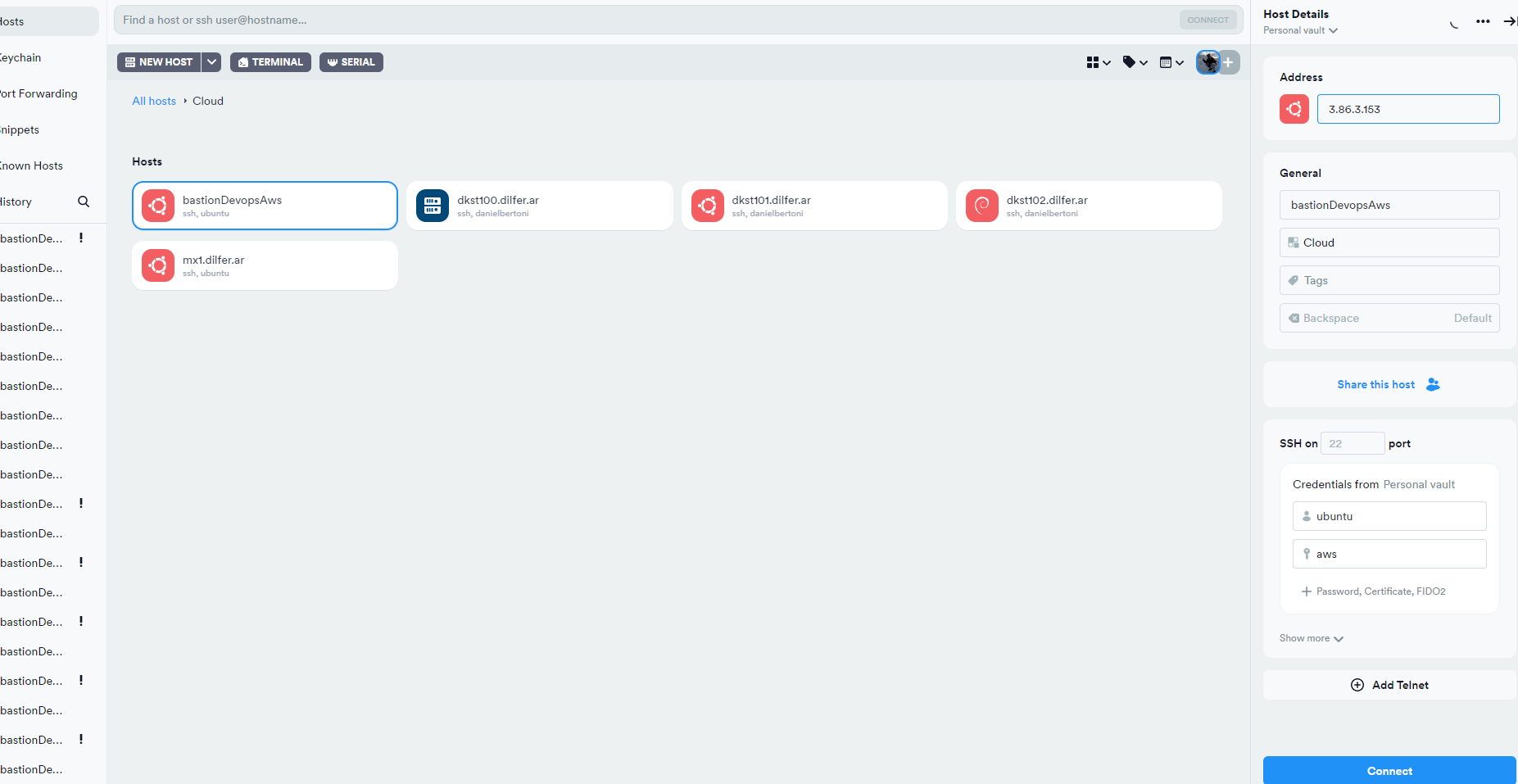


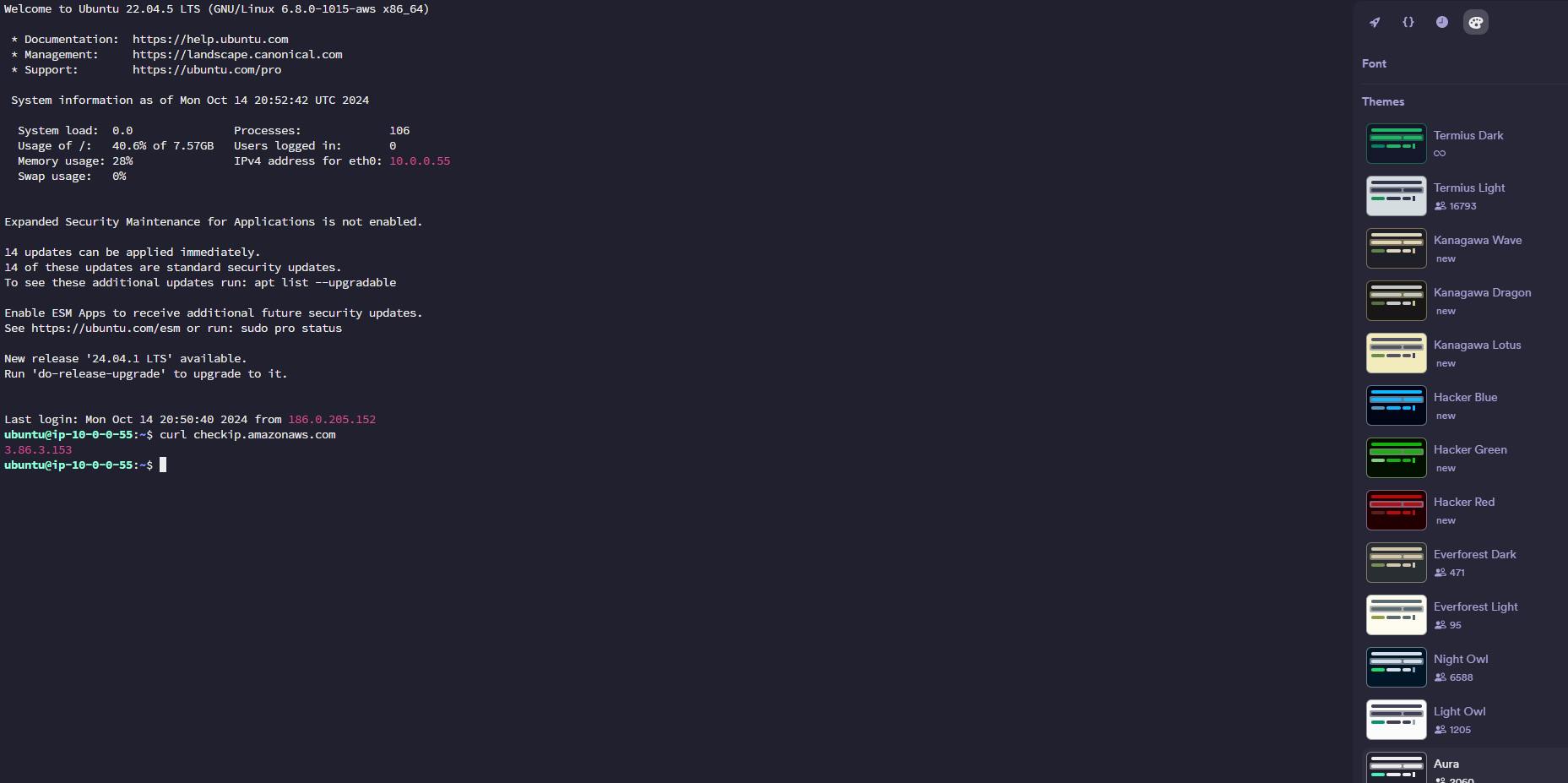


## Acceso SSH a la Instancia

La clave PEM se descargó oportunamente, se logró la conexión SSH a la instancia de forma exitosa.



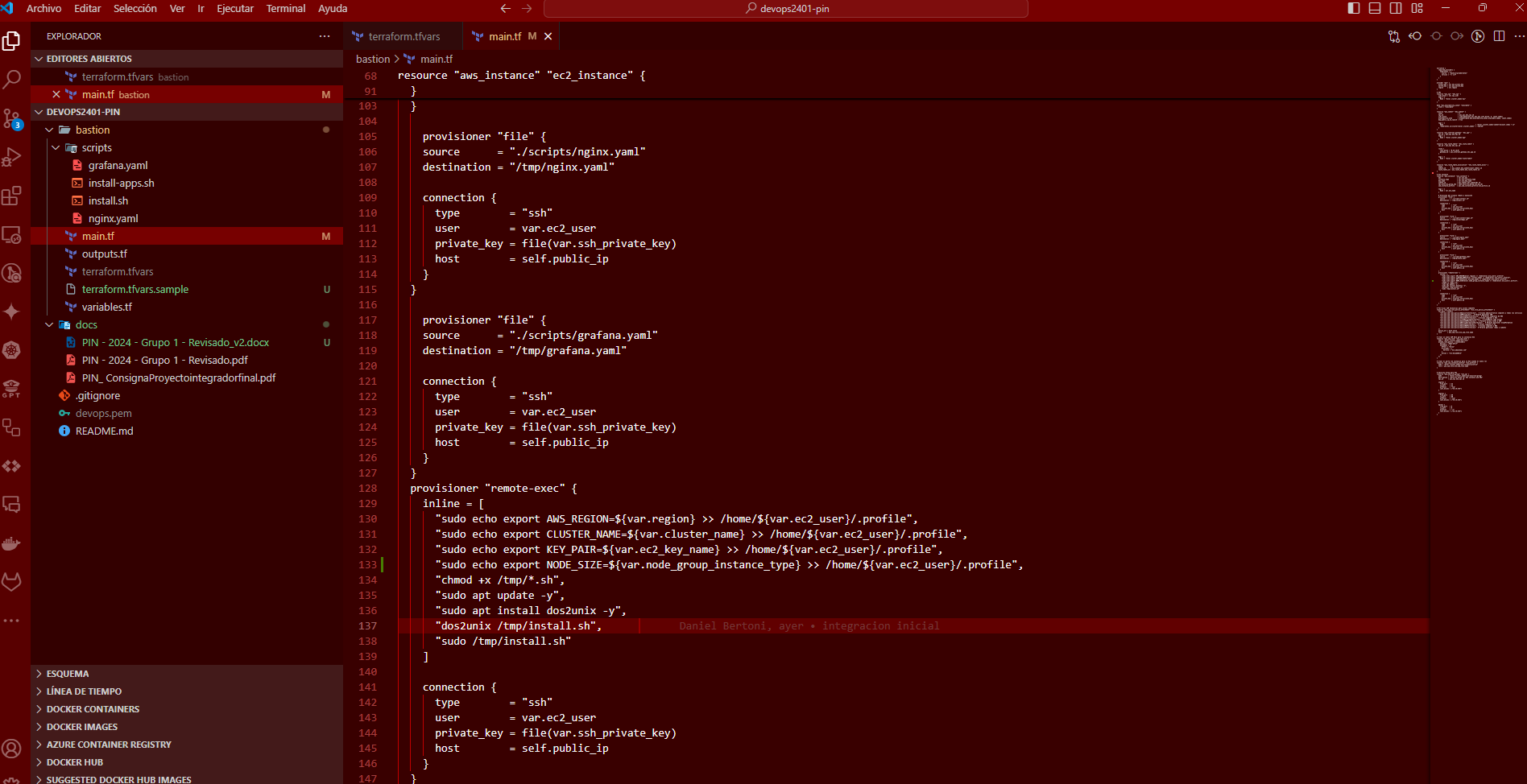




## Aprovisionamiento de scripts

Aquí usamos la misma llave que vamos a usar para conectarnos al bastión , para transferirle los archivo de script al directorio /tmp del bastión y así poder controlar de manera versionada los cambios que vayamos haciendo en los script de despliegue.

Ademas es requerido pasarle las variable de Terraform al Sistema Opertivo mediante variables de entorno.

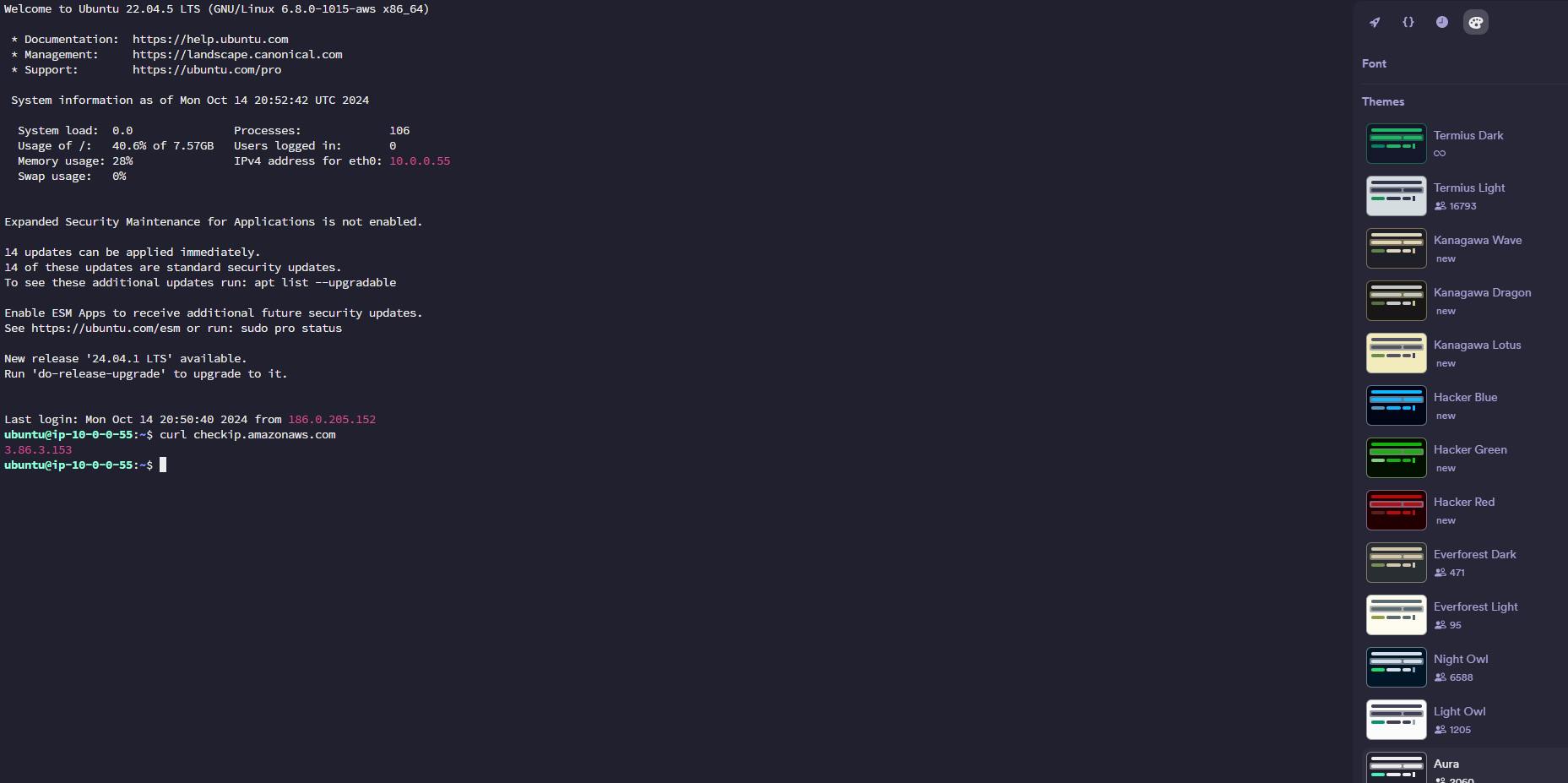


## Instalación de las aplicaciones

Iniciamos el proceso de instalación utilizando el siguiente comando:

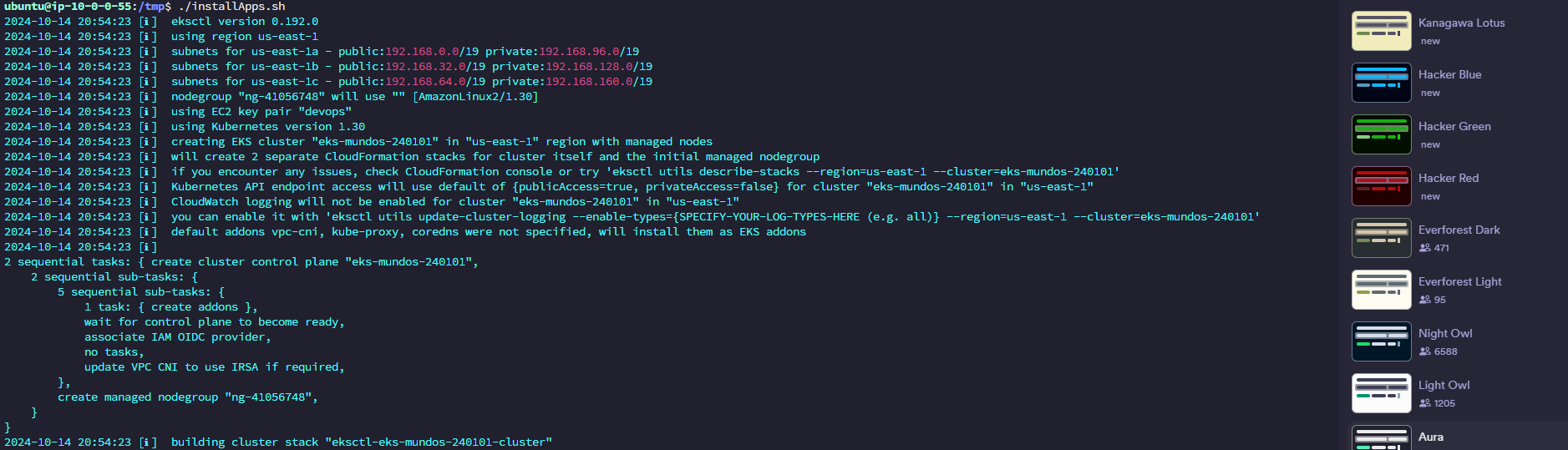
cd /tmp

./installApps.sh



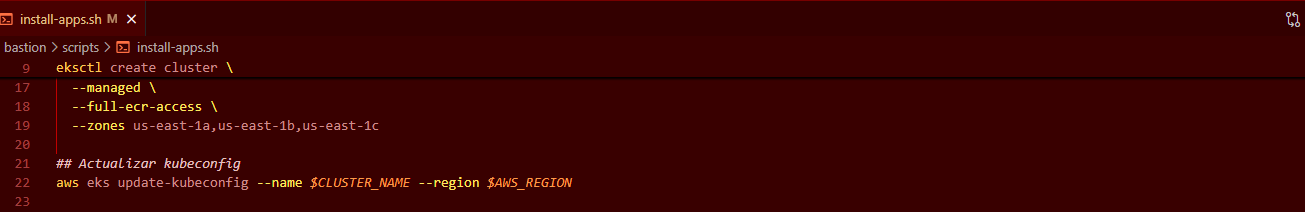
## Instalación del EKS

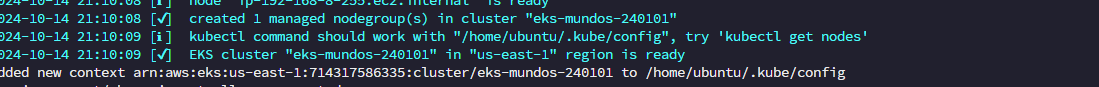
Enel script se instalación se ejecutara esta parte

## Configuración de kubectl

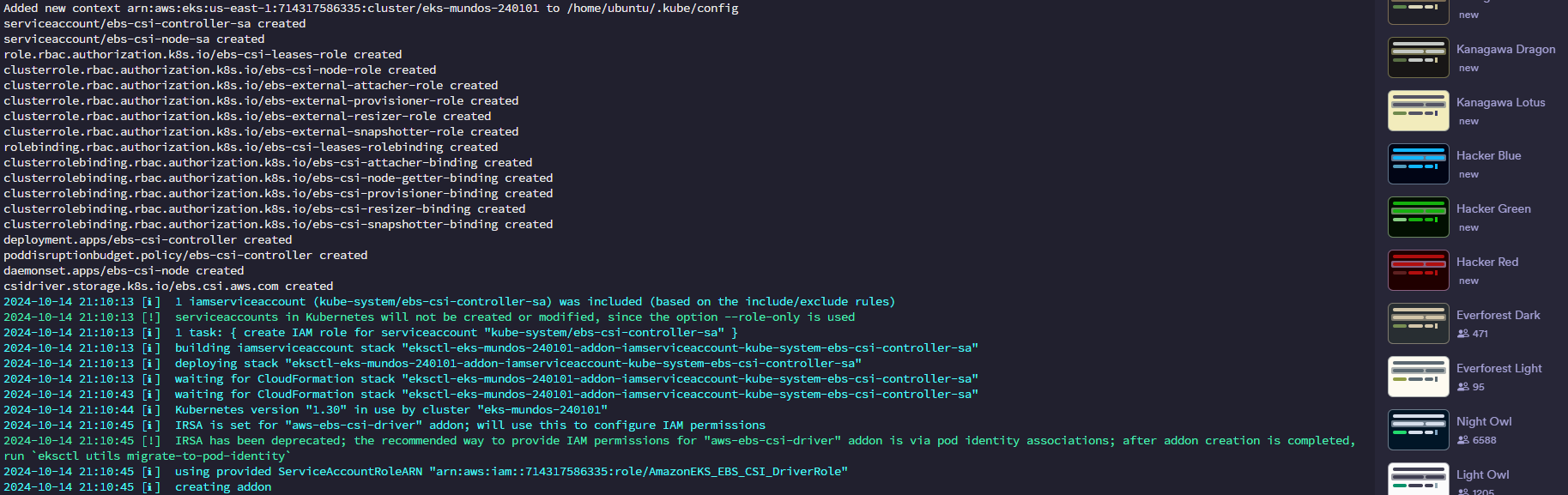
En el script de instalacion se ejecutara en esta parte





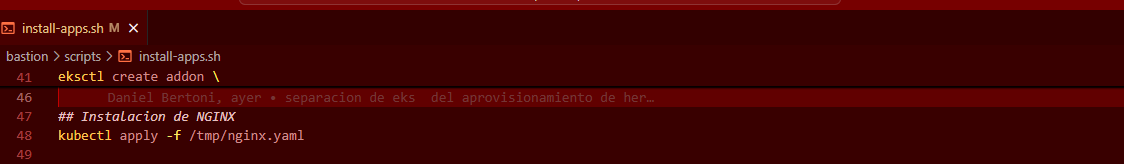
## Instalación del EBS Driver

En el script de instalacion se ejecutara en esta parte

## Despliegue de Nginx

En el script de instalacion se ejecutara en esta parte



El deployment es el siguiguiente:

apiVersion: v1

kind: Namespace

metadata:

name: devops2401

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

namespace: devops2401

labels:

app: nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.21.6

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: nginx-service

namespace: devops2401

labels:

app: nginx

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 80

selector:

app: nginx

---

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginx-ingress

namespace: devops2401

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: nginx.local

http:

paths:

- path: /

pathType: Prefix

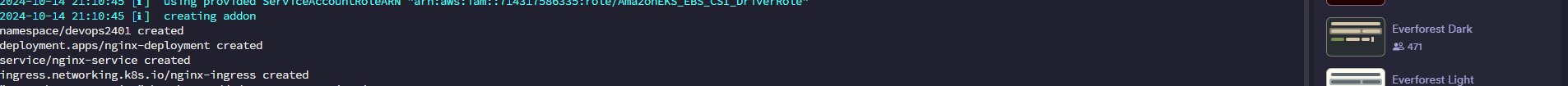
backend:

service:

name: nginx-service

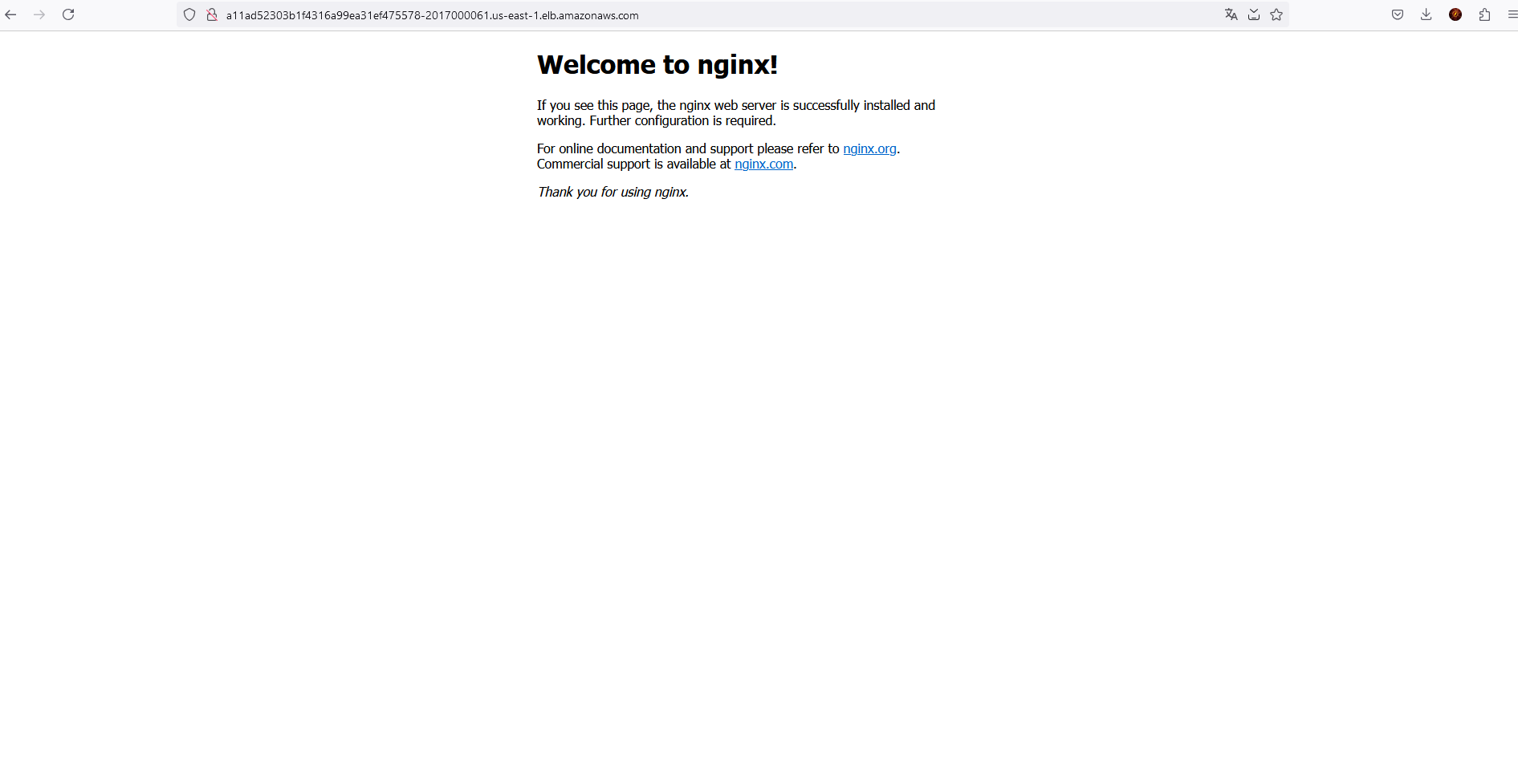
port:

number: 80



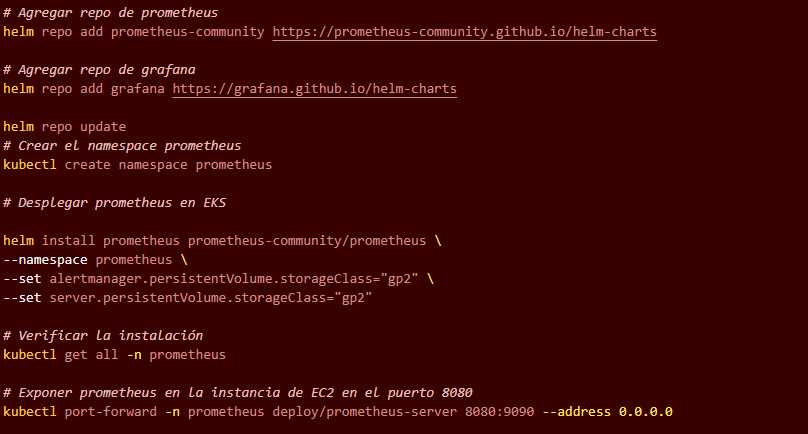


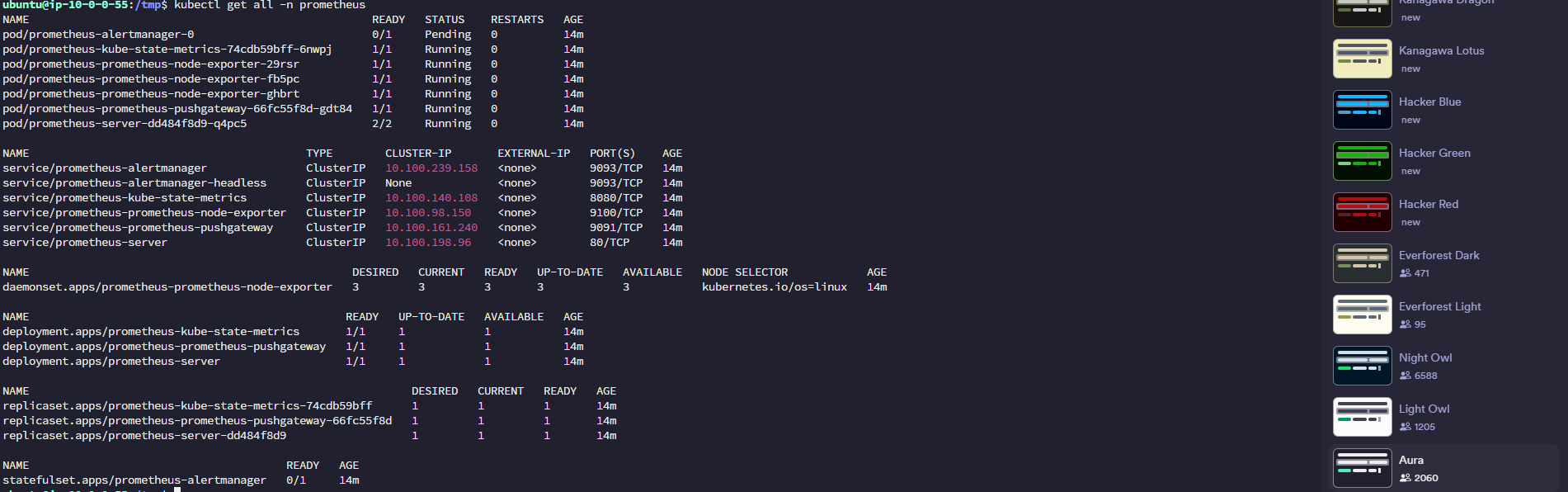
Luego, verificamos el acceso a Nginx desde el navegador.



## Despliegue de Prometheus

En el script de instalacion se ejecutara en esta parte



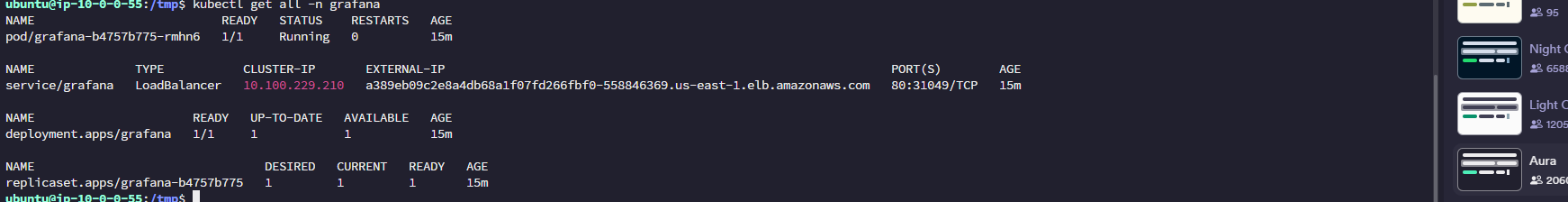


## Instalación de Grafana

En el proceso de instalacion de grafana se utiliza un archivo grafana.yaml que sera donde se configura el datasource de prometheus que se despliega en el paso anterior.

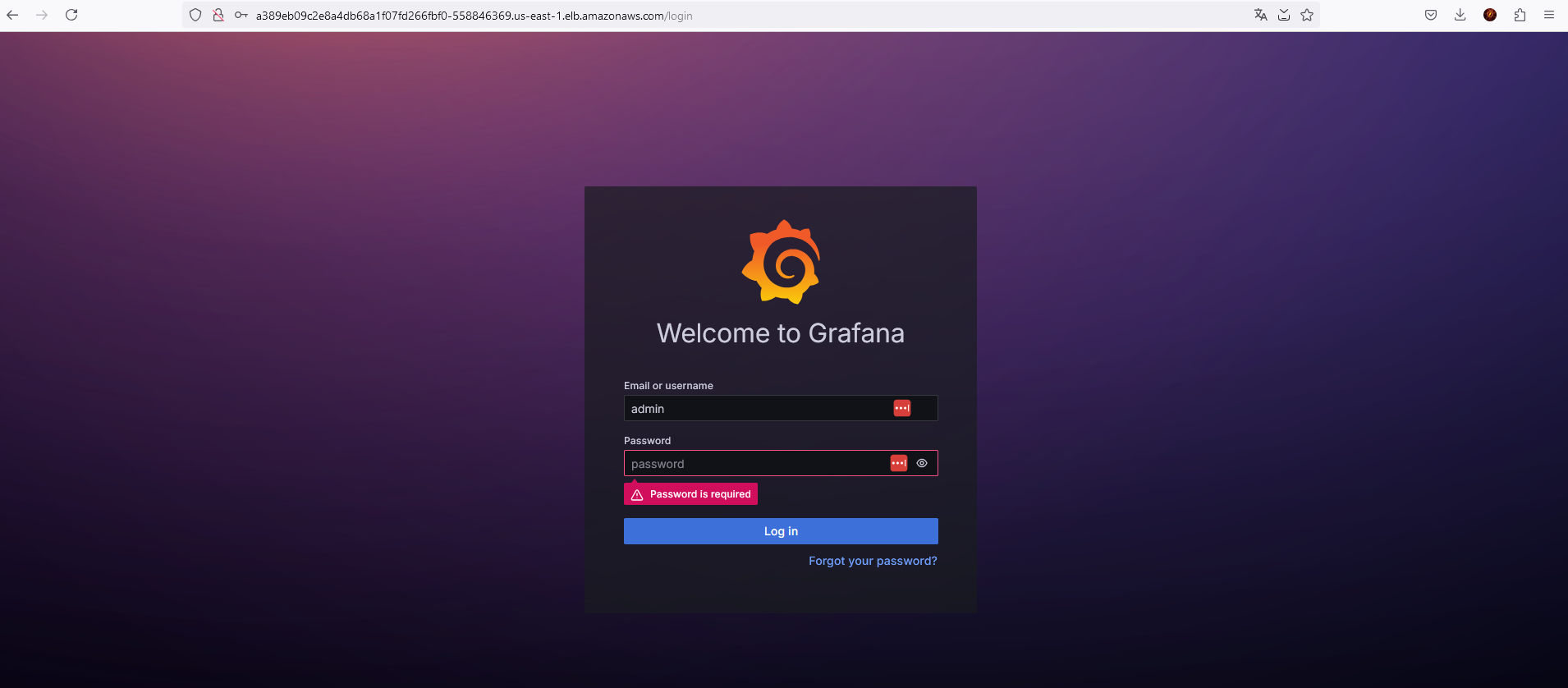
En el script de instalacion se ejecutara en esta parte



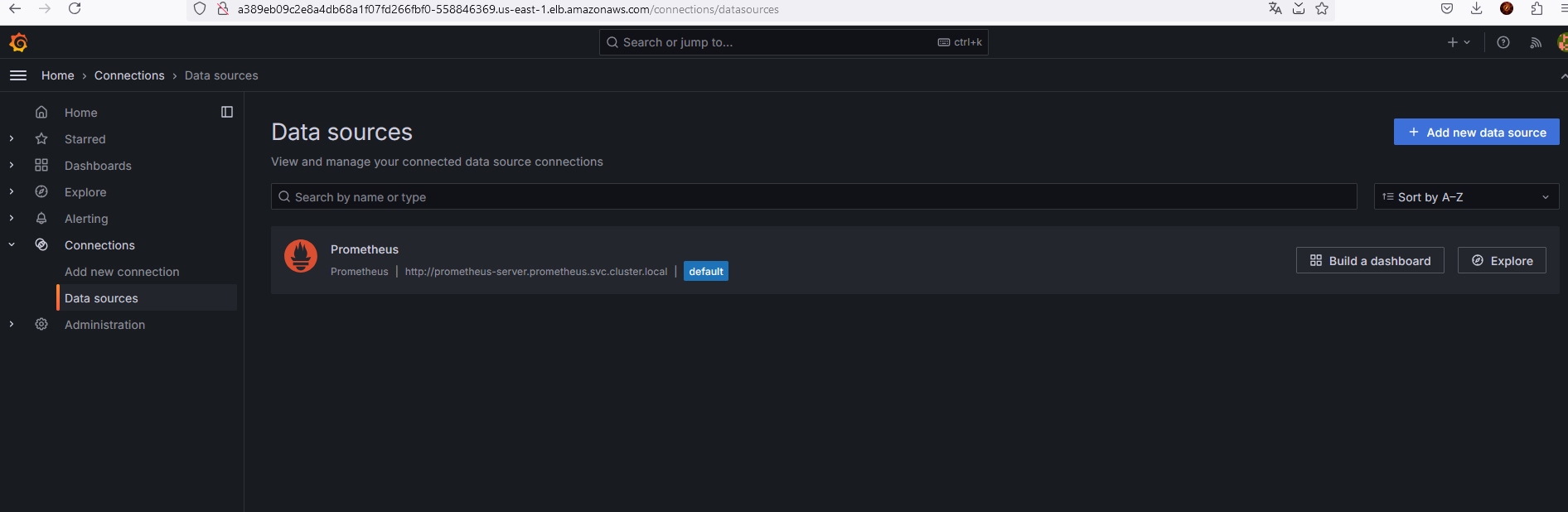


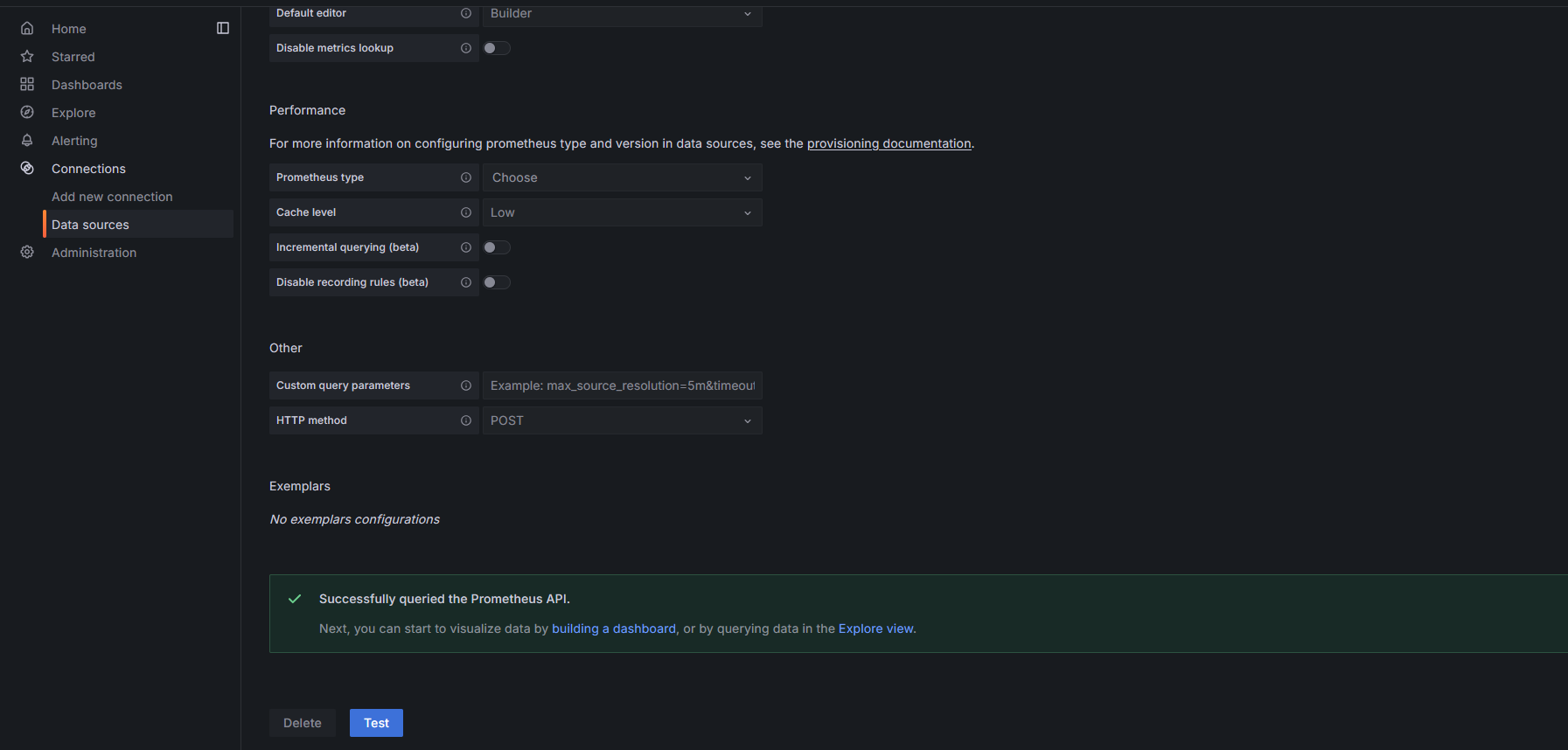
## Monitoreo

Accedemos al external-ip

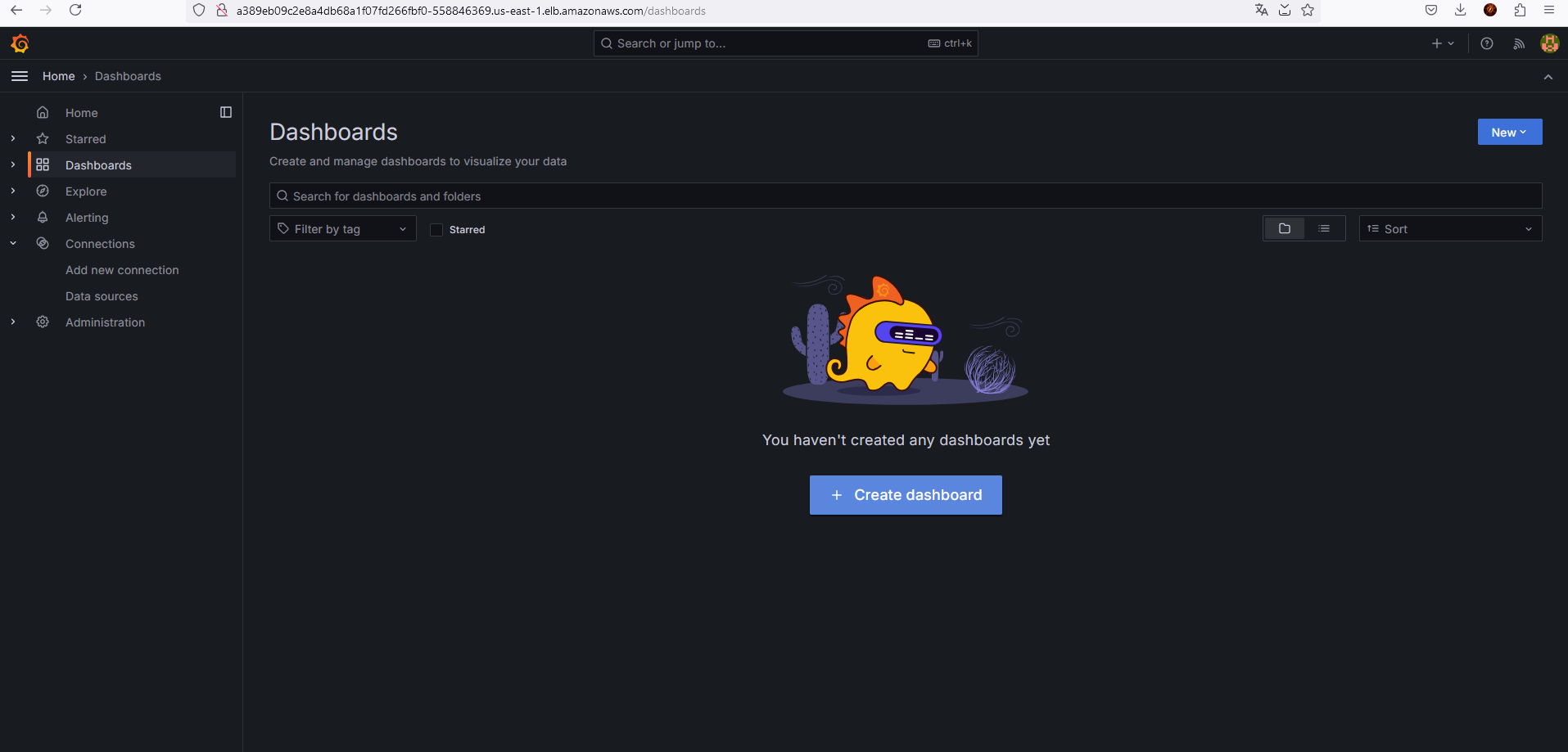


Verificamos que exista el datasource





Importamos los dashboards 6417 y 3119 en la carpeta Dashboard



Verificamos importación de ambos

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

ID 6417:

Pantalla de un video juego

Descripción generada automáticamente con confianza media

ID 3119:

Captura de pantalla de un videojuego

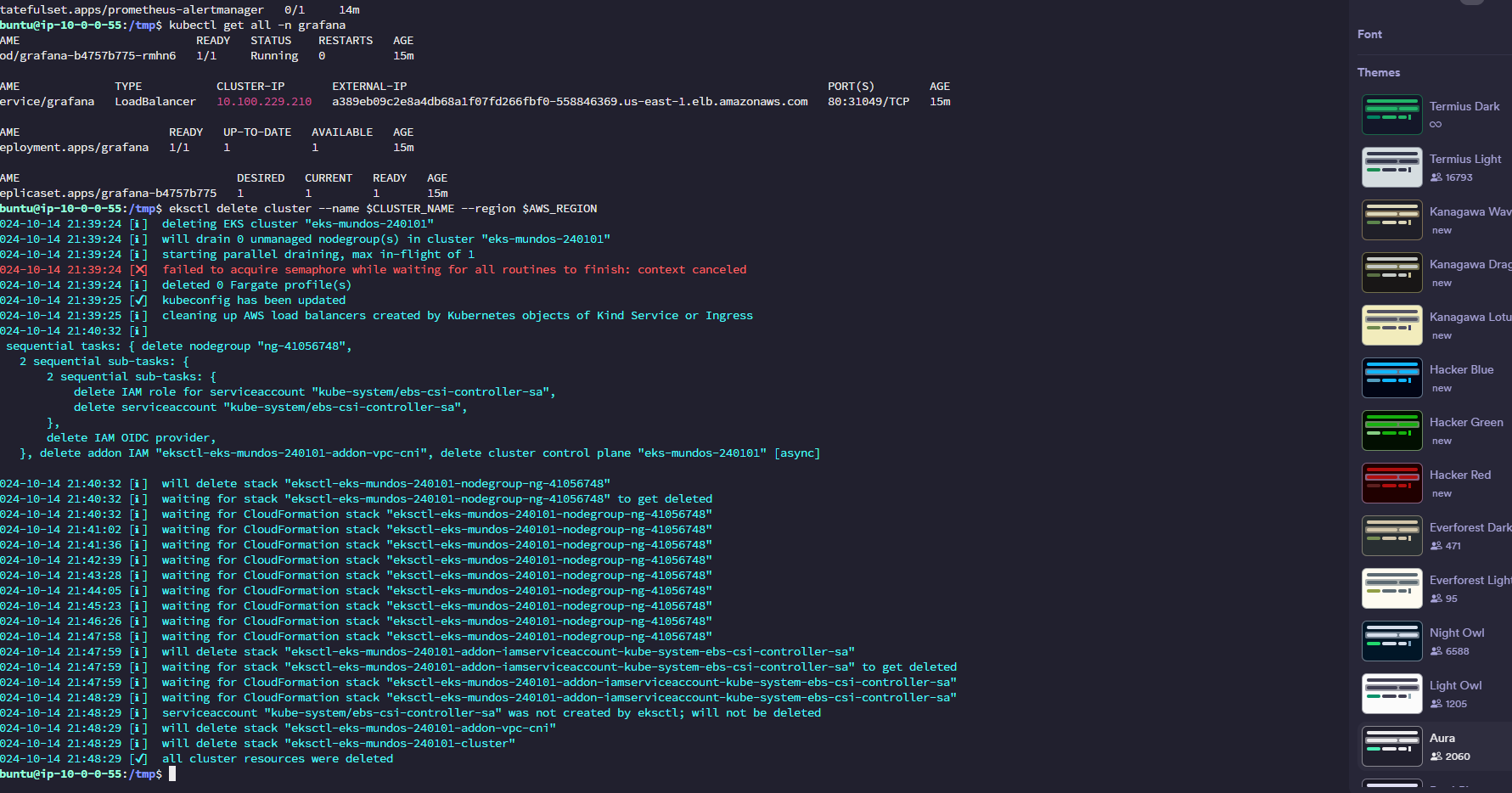
Descripción generada automáticamente

## Limpieza de Recursos

Procedimos a limpiar todos los recursos desplegados para evitar costos innecesarios.

**Desde el bastion (EC2)**

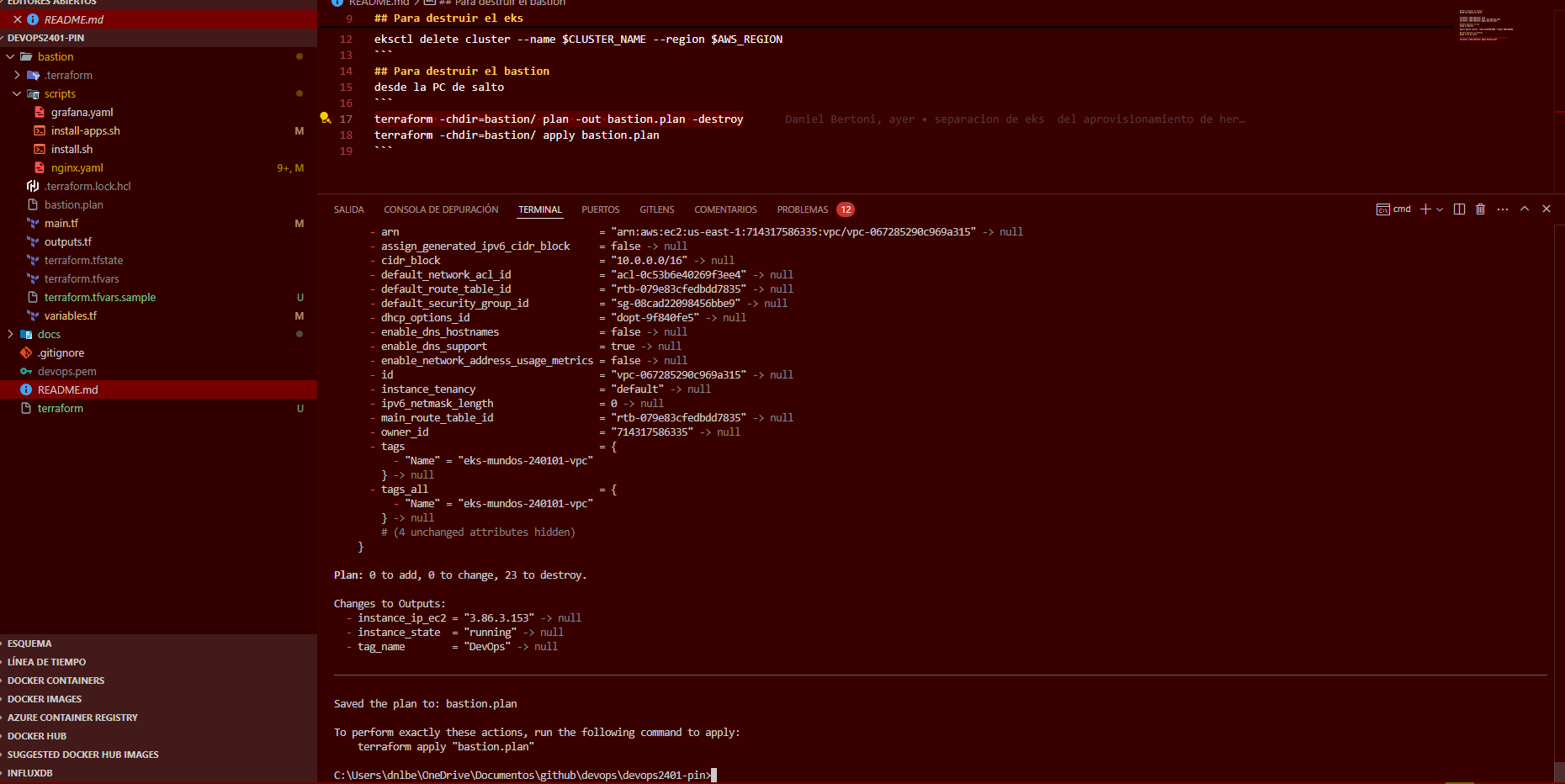
eksctl delete cluster --name $CLUSTER\_NAME --region $AWS\_REGION

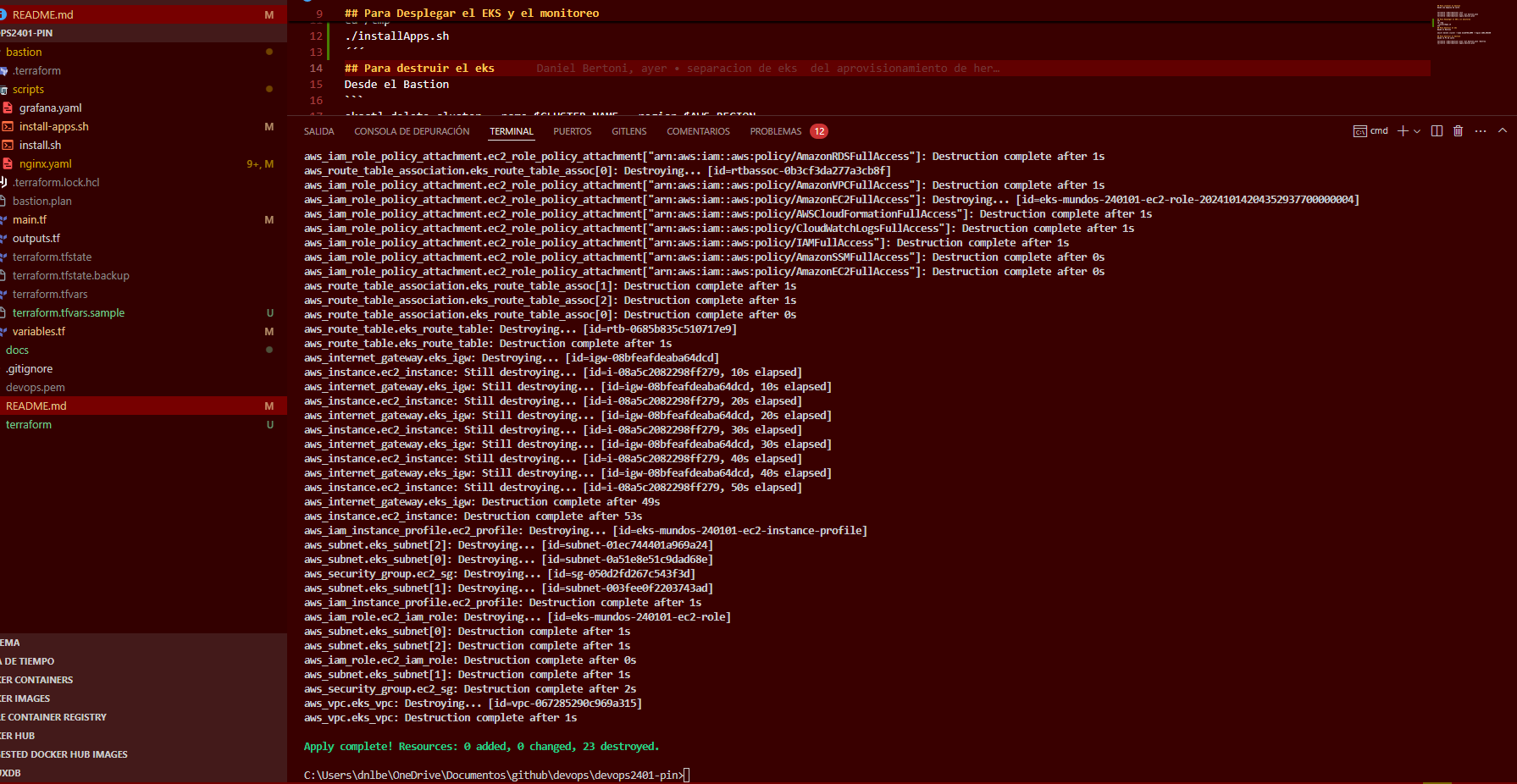


**Desde el pc de salto:**

terraform -chdir=bastion/ plan -out bastion.plan -destroy

terraform -chdir=bastion/ apply bastion.plan

****

****

## Anexo I - Reaseguro de Limpieza

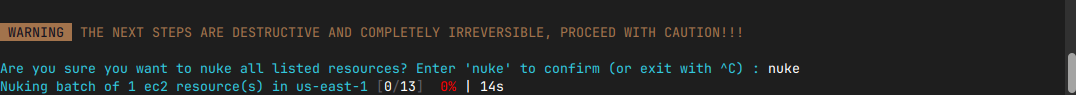
**CLOUD-NUKE - Eliminar otros servios creado a mano.**

cloud-nuke\_linux\_amd64 aws --region us-east-1

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Confirmamos**

****

****