PIN DevOps 2401

Repositorio - https://github.com/palacioea/Pin-2024

Grupo 01 Daniel Bertoni Emiliano Palacio

Contenido

[Enfoque de la Solución 2](#_Toc179754183)

[Creación de la Instancia EC2 en AWS 2](#_Toc179754184)

[Acceso SSH a la Instancia 4](#_Toc179754185)

[Instalación de Aplicaciones 5](#_Toc179754186)

[Creación del Cluster EKS 5](#_Toc179754187)

[Configuración de kubectl 7](#_Toc179754188)

[Instalación del EBS Driver 8](#_Toc179754189)

[Despliegue de Nginx 9](#_Toc179754190)

[Configuración de Helm y Despliegue de Prometheus 10](#_Toc179754191)

[Instalación de Grafana 10](#_Toc179754192)

[Monitoreo 11](#_Toc179754193)

[Limpieza de Recursos 13](#_Toc179754194)

[Anexo 15](#_Toc179754195)

## Enfoque de la Solución

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**

## Anexo

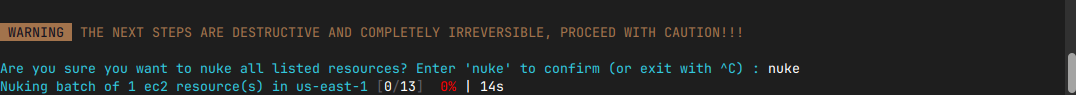
**CLOUD-NUKE - Eliminar otros servios creado a mano.**

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

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Confirmamos**

****

****