
Mundose

Devops2401

Alumno: Adolfo Del Castillo Gomez

Deploy servicios en kubernetes



Tecnologias usadas:

- Terraform
- Github
- Github Actions
- AWS
- EKS
- Kubernetes
- Helm
- Ingress
- Load Balancer
- VPC
- IAM
- Nginx
- Grafana
- Prometheus

Paso 1: Crear usuario de servicio en AWS Console para que terraform pueda conectarse a AWS

IAM > Usuarios > Crear usuario

Paso 1
[Especificar los detalles del usuario](#)

Paso 2
[Establecer permisos](#)

Paso 3
Revisar y crear

Revisar y crear

Revise las opciones seleccionadas. Después de crear el usuario, puede ver y descargar la contraseña autogenerada, si está habilitada.

Detalles del usuario

Nombre de usuario terraform-user	Tipo de contraseña de consola None	Exigir el restablecimiento de la contraseña No
-------------------------------------	---------------------------------------	---

Resumen de permisos

< 1 >

Nombre	Tipo	Usado como
admin	Grupo	Grupo de permisos

Etiquetas : *opcional*

Las etiquetas son pares clave-valor que puede agregar a los recursos de AWS para ayudar a identificar, organizar o buscar recursos. Elija las etiquetas que desee asociar a este usuario.

No hay etiquetas asociadas al recurso.

Agregar nueva etiqueta

Puede agregar hasta 50 etiqueta más.

Cancelar

Anterior

Crear usuario

Paso 2: Generamos las claves de acceso para AWS Cli para el usuario nuevo y usarlo con terraform

The screenshot shows the AWS IAM console interface. At the top, there's a navigation bar with the AWS logo, 'Servicios', a search bar, and various service icons. A green banner at the top of the console area states: 'Clave de acceso creada. Este es el único momento en el que se puede ver o descargar la clave de acceso secreta. No podrá recuperarla posteriormente. Sin embargo, puede crear una nueva clave de acceso en cualquier momento.'

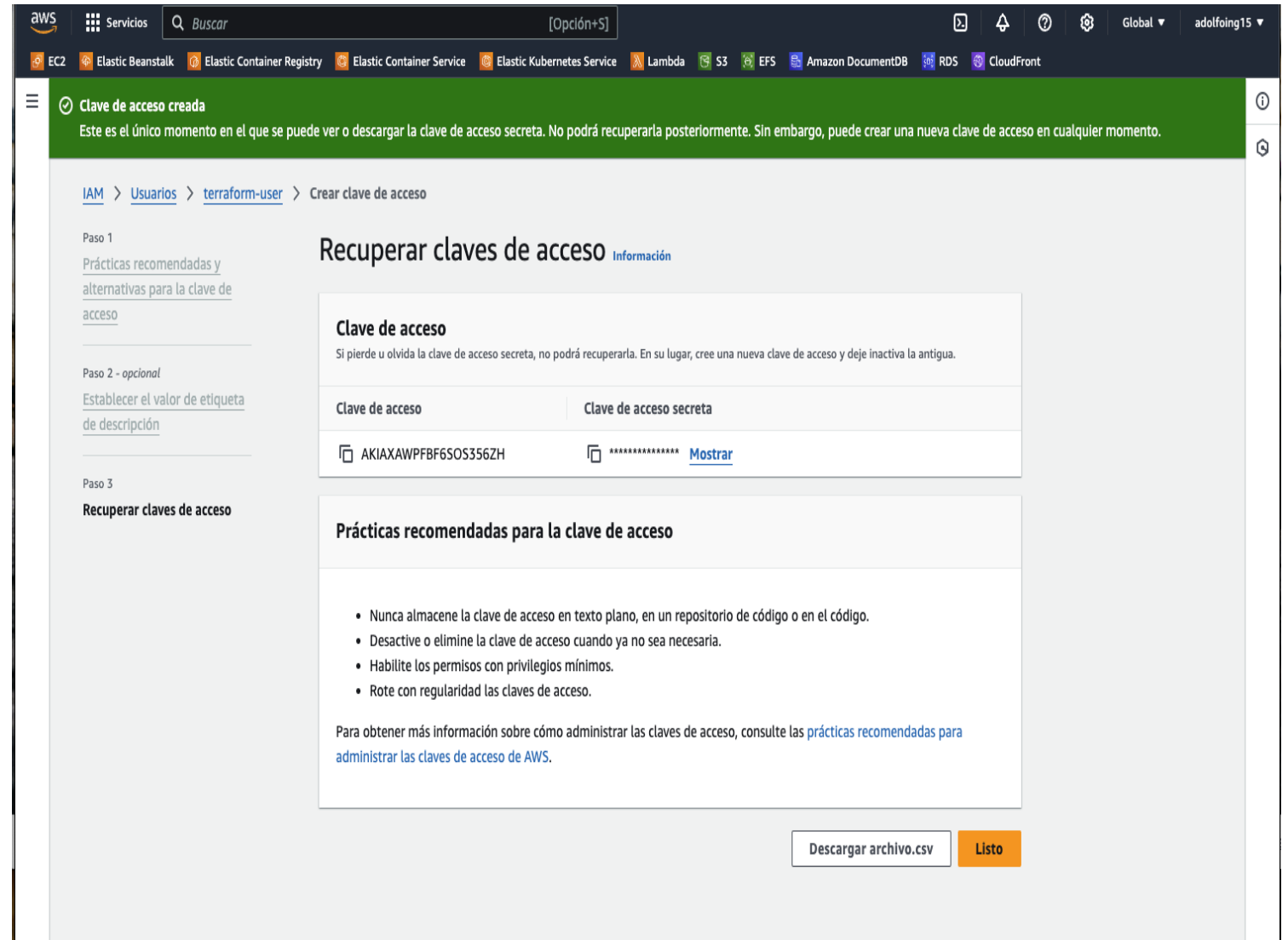
The breadcrumb trail is: IAM > Usuarios > terraform-user > Crear clave de acceso. The left sidebar shows a list of steps: Paso 1 (Prácticas recomendadas y alternativas para la clave de acceso), Paso 2 - opcional (Establecer el valor de etiqueta de descripción), and Paso 3 (Recuperar claves de acceso, which is the current step).

The main content area is titled 'Recuperar claves de acceso' with an 'Información' link. It contains a section 'Clave de acceso' with a warning: 'Si pierde u olvida la clave de acceso secreta, no podrá recuperarla. En su lugar, cree una nueva clave de acceso y deje inactiva la antigua.' Below this is a table with two columns: 'Clave de acceso' and 'Clave de acceso secreta'. The first row shows the access key 'AKIAWPF6SOS356ZH' and the secret key masked with asterisks, with a 'Mostrar' link next to it.

Below the table is a section 'Prácticas recomendadas para la clave de acceso' with a list of bullet points: 'Nunca almacene la clave de acceso en texto plano, en un repositorio de código o en el código.', 'Desactive o elimine la clave de acceso cuando ya no sea necesaria.', 'Habilite los permisos con privilegios mínimos.', and 'Rote con regularidad las claves de acceso.' At the bottom of this section, it says: 'Para obtener más información sobre cómo administrar las claves de acceso, consulte las prácticas recomendadas para administrar las claves de acceso de AWS.'

At the bottom right of the console area, there are two buttons: 'Descargar archivo.csv' and 'Listo'.

Paso 2: Generamos las claves de acceso para AWS Cli para el usuario nuevo y usarlo con terraform



The screenshot shows the AWS IAM console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various service icons. Below this, a green banner indicates that an access key has been created. The main content area is titled 'Recuperar claves de acceso' (Recover access keys) and includes a warning about the secret access key. It displays the access key ID 'AKIAWPF6SOS356ZH' and a masked secret access key with a 'Mostrar' (Show) link. A section titled 'Prácticas recomendadas para la clave de acceso' (Recommended practices for the access key) lists four guidelines: never store keys in plain text, deactivate or delete keys when not needed, use least privilege permissions, and rotate keys regularly. At the bottom right, there are buttons for 'Descargar archivo.csv' (Download CSV file) and 'Listo' (Done).

Clave de acceso creada
Este es el único momento en el que se puede ver o descargar la clave de acceso secreta. No podrá recuperarla posteriormente. Sin embargo, puede crear una nueva clave de acceso en cualquier momento.

[IAM](#) > [Usuarios](#) > [terraform-user](#) > Crear clave de acceso

Paso 1
[Prácticas recomendadas y alternativas para la clave de acceso](#)

Paso 2 - *opcional*
[Establecer el valor de etiqueta de descripción](#)

Paso 3
Recuperar claves de acceso

Recuperar claves de acceso [Información](#)

Clave de acceso
Si pierde u olvida la clave de acceso secreta, no podrá recuperarla. En su lugar, cree una nueva clave de acceso y deje inactiva la antigua.

Clave de acceso	Clave de acceso secreta
AKIAWPF6SOS356ZH	***** Mostrar

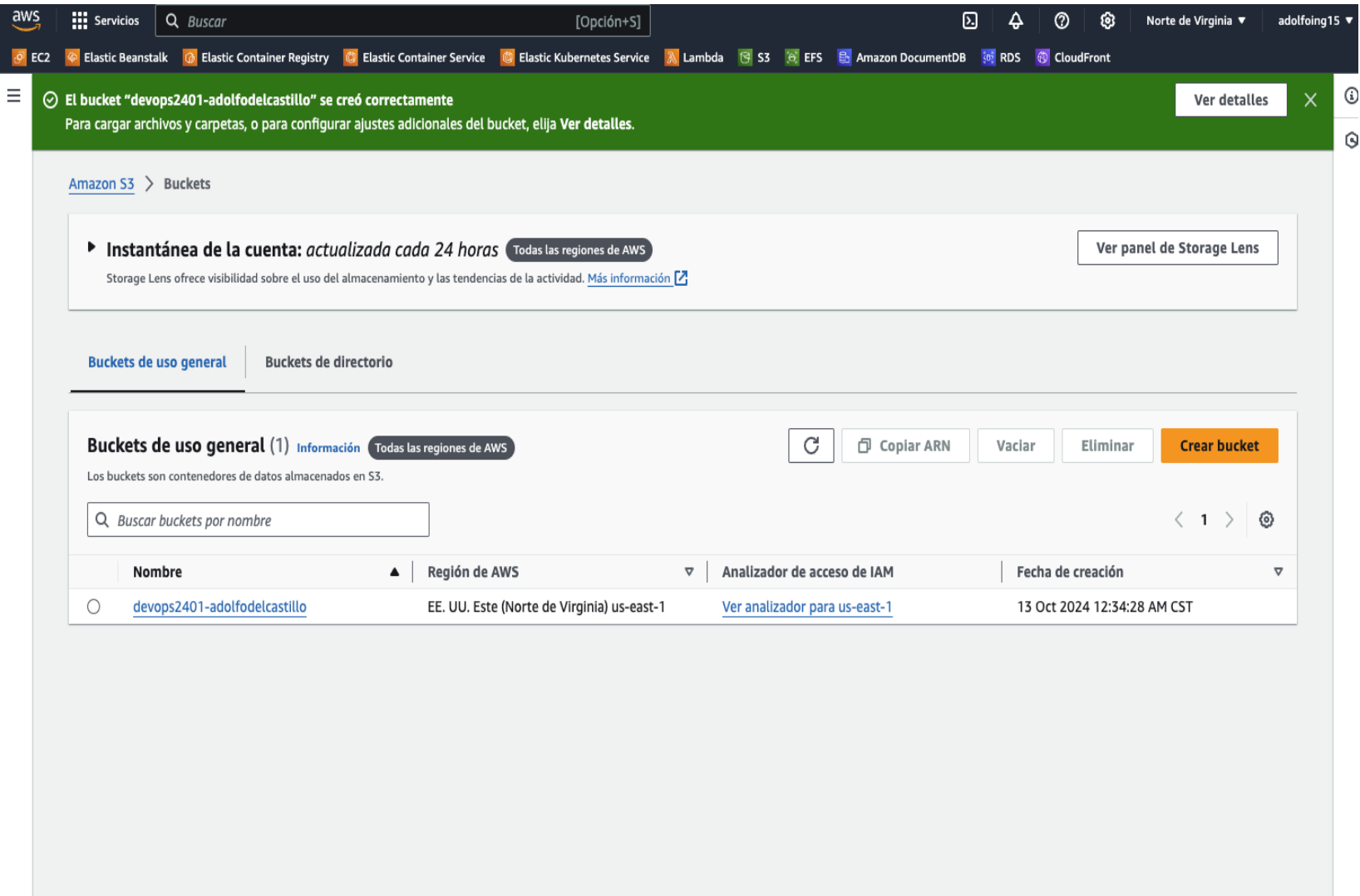
Prácticas recomendadas para la clave de acceso

- Nunca almacene la clave de acceso en texto plano, en un repositorio de código o en el código.
- Desactive o elimine la clave de acceso cuando ya no sea necesaria.
- Habilite los permisos con privilegios mínimos.
- Rote con regularidad las claves de acceso.

Para obtener más información sobre cómo administrar las claves de acceso, consulte las [prácticas recomendadas para administrar las claves de acceso de AWS](#).

[Descargar archivo.csv](#) [Listo](#)

Paso 3: Vamos a ocupar terraform asi que como buena practica generamos el bucket de s3 donde vamos a tener nuestro terraform.tfstate



Paso 4: El código de terraform consta de la creación de backend que es donde tendremos nuestro terraform.tfstate

```
terraform {  
  backend "s3" {  
    bucket = "devops2401-adolfoelcastillo"  
    key = "terraform.tfstate"  
    region = "us-east-1"  
  }  
}
```



Paso 5: Para la parte de creacion de servicios este esta todo en main.tf, lo servicios que declaramos para levantar la infraestructura de EKS son:

- VPC
- Subnets (Privada y Publica)
- Gateway
- Rutetable
- EKS Cluster
- IAM Role
- IAM Policies
- Node group

Paso 6: Para la parte del deploy del servicio ingress de nginx se uso helm para ello generamos un helmfile con la siguiente configuracion

repositories:

- name: ingress-nginx

url: <https://kubernetes.github.io/ingress-nginx>

releases:

- name: nginx

namespace: default

chart: ingress-nginx/ingress-nginx

values:

- nginx-values.yaml

Paso 7: Deploy de Nginx, en este caso usamos archivos yaml de kubernetes con el cual deployamos el pod con la imagen de nginx

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

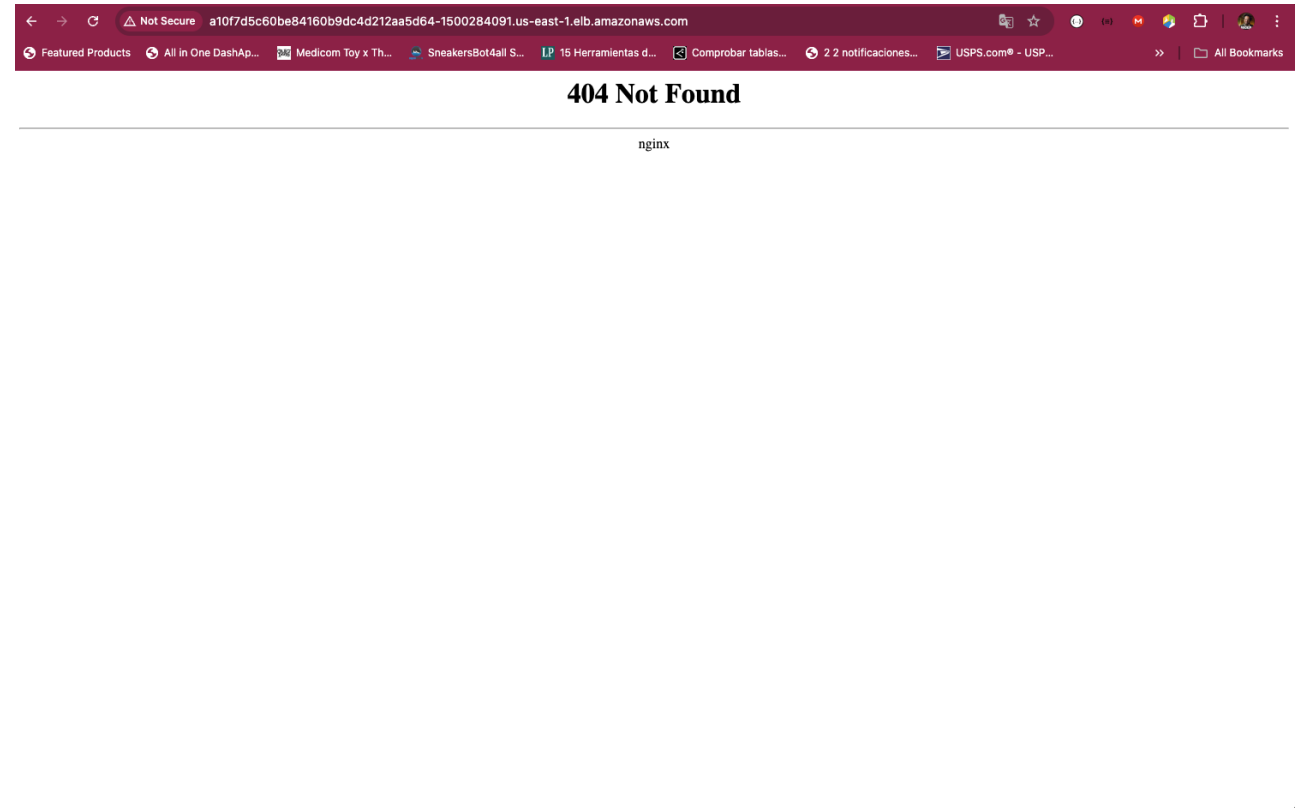
Paso 8: Deploy de Nginx, tambien se hizo deploy del ingress de nginx

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
name: nginx-ingress
namespace: default
annotations:
nginx.ingress.kubernetes.io/rewrite-target: /
spec:
rules:
- http:
paths:
- path: /
pathType: Prefix
backend:
service:
name: nginx-service
port:
number: 80
```

Paso 10: Deploy de Nginx, y el servicio del service

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: default
spec:
  type: ClusterIP
  ports:
    - port: 80
  targetPort: 80
  selector:
    app: nginx
```

Paso 11: Con estos deploy se despliega un loadbalancer en AWS el cual nos genera una URL para poder consumir por internet la pagina de NGINX



Paso 12: La parte de Grafana y prometheus se agrego con helm para que realice la configuracion necesaria y se uso archivos values para que la configuracion del ambiente sea configurable por modulo.

repositories:

- name: ingress-nginx

url: <https://kubernetes.github.io/ingress-nginx>

- name: prometheus-community

url: <https://prometheus-community.github.io/helm-charts>

- name: grafana

url: <https://grafana.github.io/helm-charts>

releases:

- name: nginx

namespace: default

chart: ingress-nginx/ingress-nginx

values:

- nginx-values.yaml

- name: prometheus

chart: prometheus-community/prometheus

namespace: monitoring

values:

- prometheus-values.yaml

- name: grafana

chart: grafana/grafana

namespace: monitoring

values:

- grafana-values.yaml

Paso 13: Se agrego CI con github actions creando tres pipelines uno para deploy de la infraestructura, para deploy de helm y para deploy de los pods

✓  `.github/workflows`

 `deploy-helm.yml`

 `deploy-kubctl.yml`

 `deploy-terraform.yml`

Paso 14: Cada pipeline utiliza los secrets de ambiente para conectarse a la cuenta de AWS y poder conectarse al cluster de EKS, esto se configuro dentro de security de github

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are encrypted and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for non-sensitive data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Environment secrets

This environment has no secrets.

[Manage environment secrets](#)

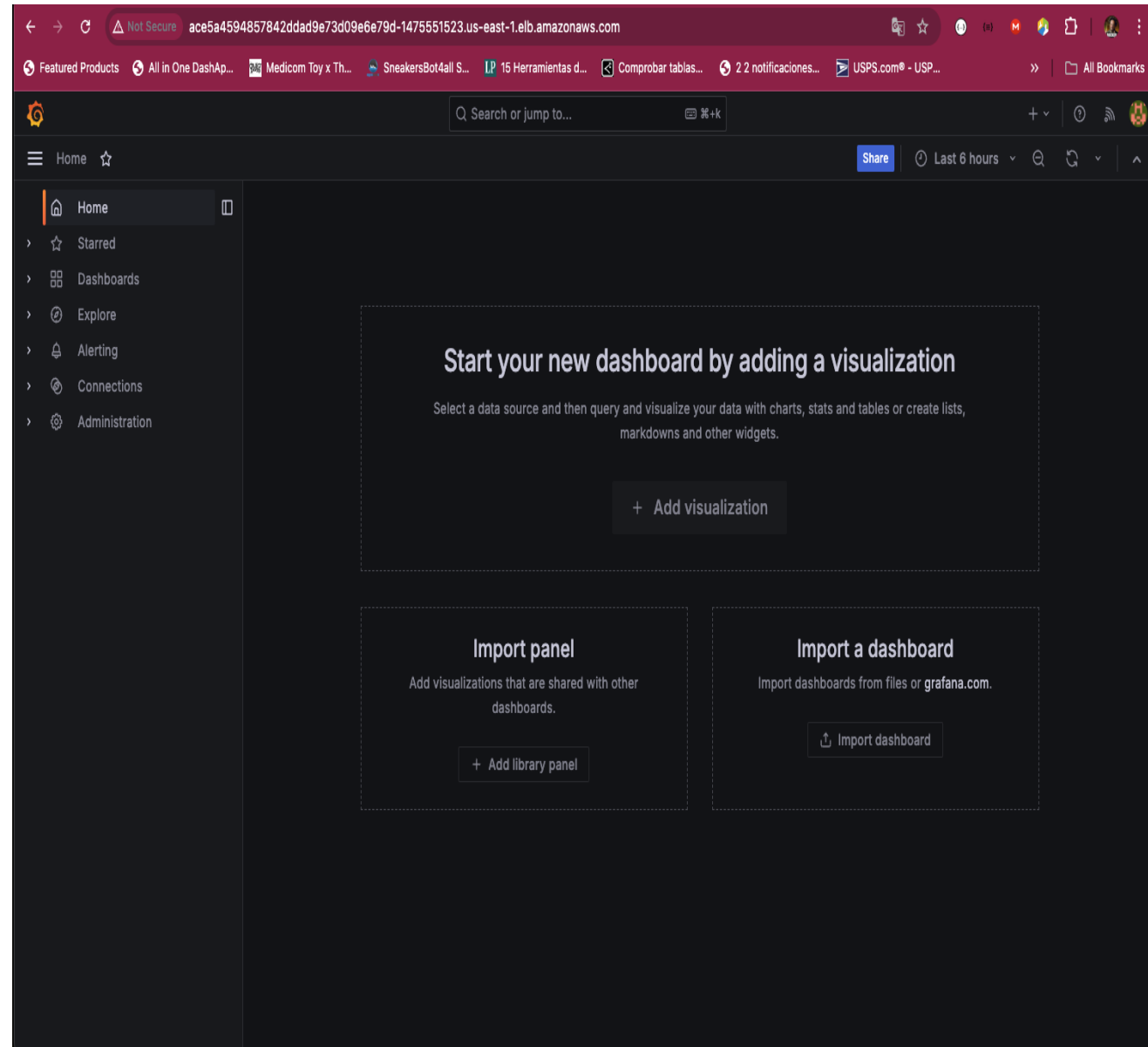
Repository secrets [New repository secret](#)

Name	Last updated
APPROVERS	2 days ago
AWS_KEY	2 days ago
AWS_SECRET_KEY	2 days ago

Paso 15: El pipeline de terraform cuenta con un step de approve o deny esto con el proposito de hacer validacion previa antes de realizar terraform apply o terraform destroy

- name: Await Manual Approval for Apply
 - uses: trstringer/manual-approval@v1
 - with:
 - secret: \${{ github.TOKEN }}
 - approvers: "adolfoDelCastillo"
 - minimum-approvals: 1
 - issue-title: "Approval needed: Deploy to AWS"
 - issue-body: "Please approve or deny the deployment of Terraform changes to the AWS environment."
 - exclude-workflow-initiator-as-approver: false

Paso 16: El servicio de Grafana se deployo usando loadbalancer para exponer el sitio en la web





Repositorio de GITHUB:

<https://github.com/SRE-Adolfo-MX/Devops2401.git>