I2BTECH - Reto DevOps

Este repositorio contiene la solución integral al reto técnico DevOps. El proyecto implementa una aplicación Node.js contenerizada, desplegada en un clúster de Kubernetes (Minikube) provisionado en una instancia EC2, utilizando prácticas modernas de CI/CD e Infraestructura como Código (IaC).

Tabla de Contenidos

Arquitectura de la Solución

La solución utiliza el patrón **Sidecar** en Kubernetes. Dentro de un mismo Pod, conviven dos contenedores:

1. **Node.js App:** La lógica de negocio (API REST).
2. **Nginx Reverse Proxy:** Maneja la entrada de tráfico, terminación de conexiones y seguridad (Basic Auth).

El tráfico fluye de la siguiente manera: `Internet` -> `Ingress Controller` -> `Service (ClusterIP)` -> `Pod (Nginx :80 -> Localhost :3000 -> Node App)`

Tecnologías Utilizadas

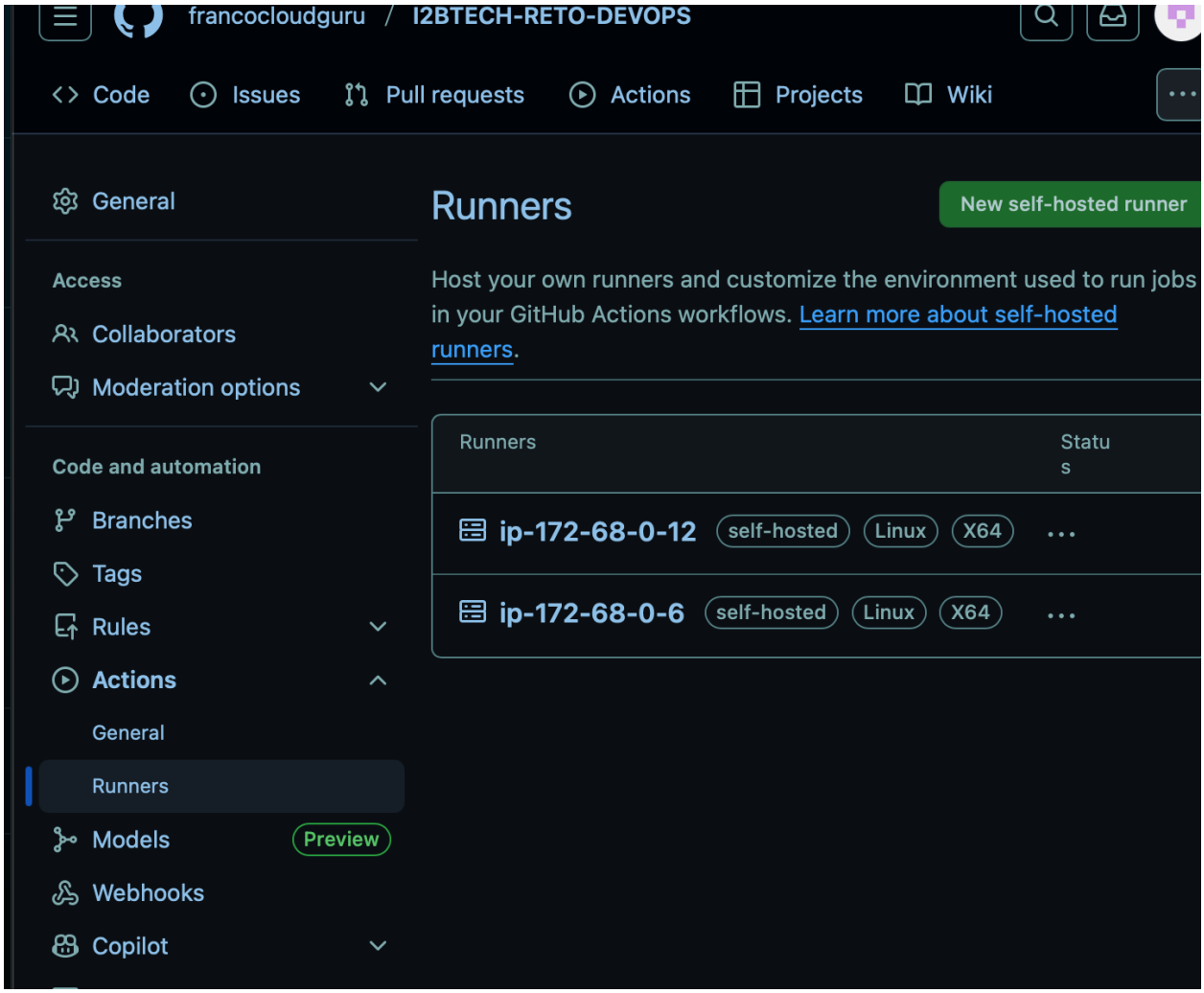
- **Aplicación:** Node.js (Express), Pino (Logging).
- **Contenerización:** Docker, Docker Compose (para pruebas locales).
- **Orquestación:** Kubernetes (Minikube v1.30+).
- **IaC & Configuración:**
  - **Ansible:** Aprovisionamiento del servidor (instalación de dependencias, Docker, Minikube).
  - **Terraform:** Despliegue de la aplicación mediante Helm Charts.
  - **Helm:** Empaquetado de la aplicación Kubernetes.
- **CI/CD:** GitHub Actions (Self-hosted runner en EC2).
- **Sistema Operativo:** Linux (Ubuntu/Debian based).

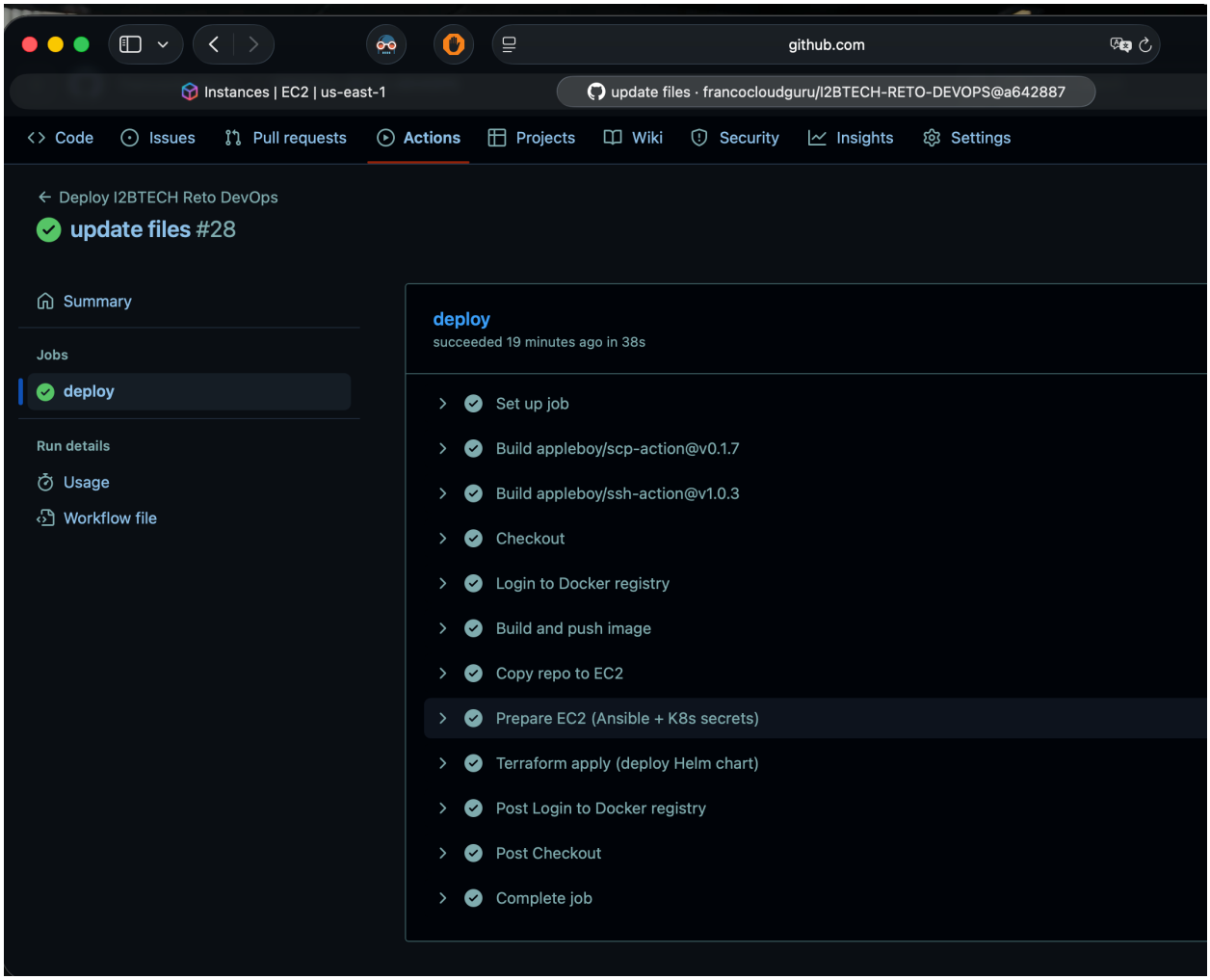
Pipeline CI/CD

El flujo de trabajo se define en `.github/workflows/deploy-pipeline.yml` y se ejecuta automáticamente al hacer push a la rama `main`.

1. **Build & Push:** Construye la imagen Docker y la sube al registro privado/público.
2. **Deploy to EC2:**
    - Copia los archivos del repositorio al servidor EC2.
    - **Ansible:** Verifica y asegura que Minikube y las herramientas necesarias estén instaladas y corriendo.
    - **K8s Secrets:** Crea los secretos necesarios (credenciales del registro, certificados TLS, htpasswd) de forma segura inyectándolos desde GitHub Secrets.
    - **Terraform:** Inicializa y aplica el plan para desplegar o actualizar el Helm Chart en el clúster.

Se utilizó la instancia EC2 Ubuntu como self hosted runner y todos los jobs funcionando de manera correcta:





Despliegue e Infraestructura

Ansible

Se encarga de preparar el "Metal". El playbook `playbook.yaml`:

- Instala dependencias del sistema (`conntrack`, `socat`, etc.).
- Configura e inicia `containerd` y los plugins CNI.
- Instala `minikube`, `kubectl` y `terraform`.
- Asegura la compatibilidad del sistema de archivos (`fs.protected_regular`).

Terraform + Helm

Terraform gestiona el estado del despliegue de la aplicación.

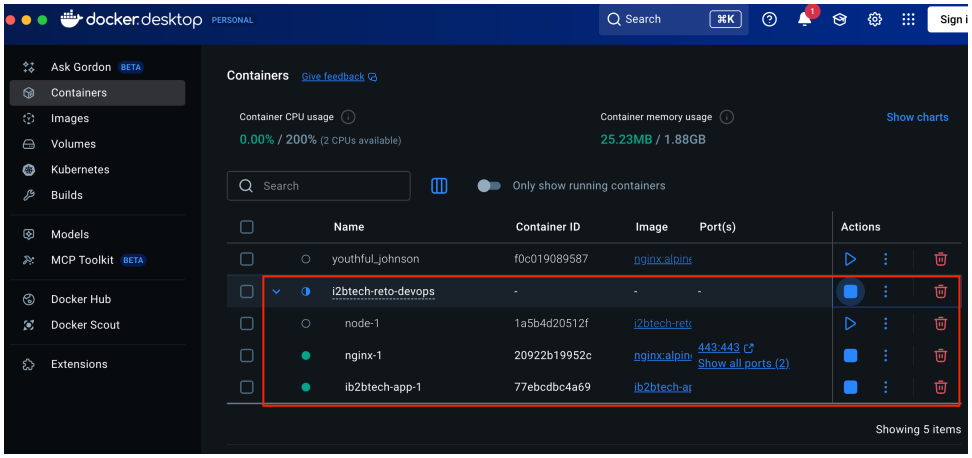
- Utiliza el provider `helm` para desplegar el chart local.
- Inyecta dinámicamente la versión de la imagen (Tag) y el repositorio desde las variables del pipeline.

Kubernetes (Helm Chart)

- **Deployment:** Configura el Pod con dos contenedores (App y Nginx).
- **PersistentVolume:** Mapea una ruta del Host (`/var/i2btech-logs`) para persistencia de logs.
- **Ingress:** Expone el servicio bajo el dominio `172.68.0.6.nip.io`.
- **ConfigMap:** Inyecta la configuración de Nginx para realizar el `proxy_pass` a `localhost:3000`.

Pruebas de Funcionamiento.

Prueba local con Docker compose:




```
franrosario@10 I2BTECH-RETO-DEVOPS % curl -k https://localhost/
{"msg":"ApiRest prueba"}%
franrosario@10 I2BTECH-RETO-DEVOPS % curl -k https://localhost/public
{"public_token":"12837asd98a7sasd97a9sd7"}%
franrosario@10 I2BTECH-RETO-DEVOPS % curl -k -I https://localhost/private
HTTP/1.1 401 Unauthorized
Server: nginx/1.29.3
Date: Sat, 13 Dec 2025 14:48:57 GMT
Content-Type: text/html
Content-Length: 179
Connection: keep-alive
WWW-Authenticate: Basic realm="Restricted Area"
```

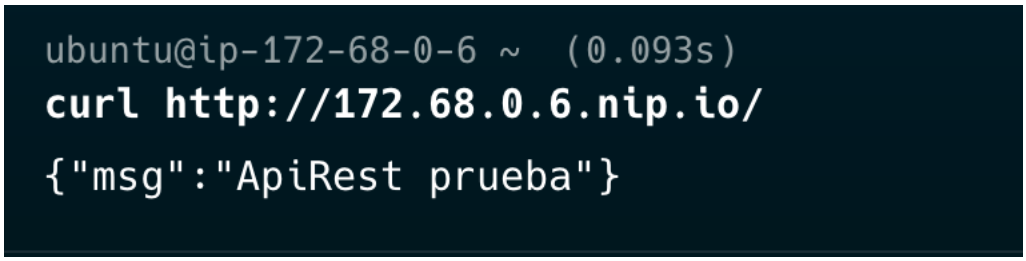
Prueba desde instancia EC2 con minikube:

A continuación se muestran los resultados de las pruebas de conectividad realizadas tras el despliegue exitoso en el entorno del reto:

1. Acceso a la ruta raíz (Pública):

curl http://172.68.0.6.nip.io/

{"msg":"ApiRest prueba"}



2.Intento de acceso a ruta privada (Sin credenciales): *El servidor Nginx intercepta la petición y solicita autenticación*.

curl http://172.68.0.6.nip.io/private

<html>

<head><title>401 Authorization Required</title></head>

<body>

<h1>401 Authorization Required</h1>

<hr> nginx/1.29.4

</body>

</html>

```
ubuntu@ip-172-68-0-6 ~  (0.102s)
curl http://172.68.0.6.nip.io/private
<html>
<head><title>401 Authorization Required</title></head>
<body>
<center><h1>401 Authorization Required</h1></center>
<hr><center>nginx/1.29.4</center>
</body>
</html>
```

3. Acceso a ruta pública:

curl http://172.68.0.6.nip.io/public

{"public_token":"12837asd98a7sasd97a9sd7"}

```
ubuntu@ip-172-68-0-6 ~  (0.144s)
curl http://172.68.0.6.nip.io/public
{"public_token":"12837asd98a7sasd97a9sd7"}
```