TEC | Tecnológico de Costa Rica

IC 7602 Redes

Proyecto Opcional

5 de septiembre de 2025

Participantes

Darío Espinoza Aguilar- 2020109109
Gerardo Gómez Brenes- 2022089271
Gianmarco Oporta Pérez- 2022141511
Ian Murillo Campos- 2020148871
Mariana Quesada Sánchez- 2021579438

Instrucciones

Para instalar toda la arquitectura y poder probar nuestros servicios, primero debemos tener kubectl y Docker instalados. Luego, desde la carpeta donde se encuentra el archivo Chart.yaml, ejecutamos el siguiente comando:

```
helm install arquitectura
```

Una vez que todos los servicios se hayan iniciado, podemos validar que todos los pods estén en funcionamiento con el siguiente comando:

```
kubectl get pods -A
```

A continuación, realizaremos pruebas para el router, el Ingress Controller y los Nginx. Para ello, vamos a utilizar los siguientes comandos.

Primero, obtenemos la dirección IP de Minikube ejecutando:

```
minikube ip
```

Una vez que tengamos la IP, ejecutamos estos comandos para realizar las peticiones:

```
curl http://{IPminikube}:30081 -> Por esta petición se ingresa
a el Ingress Controller
```

```
curl http://{IPminikube}:30081/NGINX1 -> Le hacemos la
petición para que nos devuelve la respuesta del Nginx1
```

```
curl http://{IPminikube}:30081/NGINX2 -> Le hacemos la
petición para que nos devuelve la respuesta del Nginx2
```

Para probar que el freeswitch está recibiendo solicitudes vamos a utilizar sipsak, el freeswitch tiene problemas para probarse con Zoiper. Entonces vamos a hacer una prueba sencilla con sipsak con el siguiente comando:

```
sipsak -O --udp -s sip:1000@{IPminikube}:32060 -vvv
```

Pruebas

Las pruebas e instrucciones son de las pruebas que estaremos realizando pero no se van a hacer manuales (excepción de la parte de la prueba de sipsak), creamos un .sh para la prueba de los Ngingx, el código del .sh es el siguiente:

```
#!/bin/bash
set -e

# Obtener IP del nodo en Minikube
HOST=$(minikube ip)

# Revisar estado de nginx1 y nginx2 y verificamos si esta
en running
NGINX1=$(kubectl get pods -n privado -l app=nginx1 -o
jsonpath="{.items[0].status.phase}")
NGINX2=$(kubectl get pods -n privado -l app=nginx2 -o
jsonpath="{.items[0].status.phase}")

if [[ "$NGINX1" != "Running" || "$NGINX2" != "Running" ]];
then
  echo "NGINX1 o NGINX2 no están en Running"
  exit 1
fi
echo "NGINX1 y NGINX2 están en Running"

# Probamos las peticiones hacia los ngixs por medio del
Ingress controller
echo "=== Probando acceso vía Apache Ingress (router →
apache → nginx) ==="
RESP1=$(curl -s http://$HOST:30081/Nginx1)
RESP2=$(curl -s http://$HOST:30081/Nginx2)
```

```
# Validamos las respuestas
if [[ "$RESP1" == *"Hola NGINX1"* && "$RESP2" == *"Hola
NGINX2"* ]]; then
  echo "Prueba de integración exitosa: ambos NGINX
responden correctamente"
  exit 0
else
  echo "Falló la validación de respuesta"
  echo "NGINX1 → $RESP1"
  echo "NGINX2 → $RESP2"
  exit 1
fi
```

Con este script de validación, se puede confirmar que el sistema de enrutamiento de peticiones, desde el Ingress Controller hasta los servicios de Nginx, está funcionando correctamente.

Resultados de las pruebas

Los resultados de estas pruebas, tanto las relacionadas con los Nginx, el router y el Ingress Controller, como la prueba de Sipsak, no están diseñados para ofrecer métricas de rendimiento, carga o latencia. Su principal objetivo es una validación funcional; nos dan la certeza de que la arquitectura está instalada correctamente y que cada componente se comunica de manera esperada. En esencia, confirman que los servicios están operativos y accesibles, pero no proporcionan ningún tipo de dato cuantitativo (como el tiempo de respuesta, el número de peticiones por segundo o el consumo de recursos) que sea útil para un análisis profundo o para nutrir una discusión técnica sobre el rendimiento del sistema.

Recomendaciones y conclusiones

El proyecto permitió implementar y validar una arquitectura en Kubernetes compuesta por router, Ingress Controller, Nginx y FreeSWITCH. Las pruebas realizadas confirmaron que los servicios se comunican correctamente y que la infraestructura funciona como se esperaba.

Como recomendación, sería útil complementar las validaciones con pruebas de rendimiento que midan carga y latencia, además de incorporar medidas de seguridad como certificados TLS y políticas de red entre namespaces. También se aconseja mantener configuraciones separadas por entorno para facilitar la administración y futuros despliegues.

En conclusión, el proyecto cumple con el objetivo de demostrar la viabilidad de esta arquitectura en un entorno controlado. La solución es funcional y establece una base sólida que puede ser mejorada con métricas de monitoreo, escalabilidad y prácticas de seguridad más avanzadas.

Bibliografía

Kubectl reference Docs. (n.d.). https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

nginx documentation. (n.d.). https://nginx.org/en/docs/