HA Kubernetes Cluster: Impulsando la automatización eficiente de redes

José Carlo Santizo Olivet, M.Sc. Jonathan Alberto de los Santos Chonay
Departamento de Ingeniería Electrónica, Mecatrónica, Biomédica
san20185@uvg.edu.gt, *jadelossantos@uvg.edu.gt*

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

RESUMEN

Este proyecto implementa un **clúster de Kubernetes de alta disponibilidad**[1], compuesto por seis nodos (tres de trabajo y tres de control), dos balanceadores de carga con **HAProxy**, y un **servidor NFS** para almacenamiento compartido. La implementación optimiza la disponibilidad y el rendimiento de las aplicaciones de red, creando un entorno robusto para pruebas de automatización y desarrollo. También se utilizó **Helm**[2] para gestionar eficientemente las aplicaciones del clúster. Como prueba de concepto, se intentó desplegar **ONAP** sin éxito, debido a limitaciones de recursos. Además, se desarrolló una red simulada en **GNS3** para realizar pruebas locales y controladas, integrando una API que automatiza procesos en dicha red mediante tecnologías como **FastAPI** y **Netmiko**, validando la capacidad del clúster para manejar aplicaciones críticas en redes y demostrando su eficiencia y disponibilidad para tales aplicaciones.

OBJETIVOS

OBJETIVO GENERAL

Despliegue de un clúster de *Kubernetes* de alta disponibilidad, para el despligue eficiente de aplicaciones que permitan la automatización y gestión de redes en entornos virtualizados o físicos.

OBJETIVOS ESPECÍFICOS

- Instalar y configurar herramientas como Docker, Kubelet, Kubeadm, Kubectl y Kubernetes en entornos de pruebas virtuales.
- Implementar y configurar un clúster de *Kubernetes* de alta disponibilidad, incluyendo la configuración de balanceadores de carga para distribuir el tráfico entre los nodos maestros y trabajadores.
- Configurar un servidor NFS para proporcionar almacenamiento compartido entre los nodos del clúster de Kubernetes, facilitando el acceso a volúmenes persistentes.
- Implementar una solución de red para los pods utilizando Calico, permitiendo la comunicación eficiente y segura entre los componentes del clúster.
- Desarrollar una API en *Python* para automatizar procesos de configuración y monitoreo de dispositivos de red, utilizando tecnologías como *FastAPI*, *HTML*, *Netmiko*, *PySNMP* y *Bootstrap*.
- Desplegar una aplicación desarrollada en *Python* en el clúster de *Kubernetes*, utilizando *Helm* para gestionar el despliegue, configuración y mantenimiento de la aplicación.
- Diseñar y configurar una topología de red simulada en *GNS3* para integrar y probar la comunicación entre la aplicación desplegada y dispositivos de red simulados.
- Elaborar un manual detallado de instalación y configuración del clúster de *Kubernetes*, incluyendo las configuraciones de balanceadores de carga, *Calico* y el servidor *NFS*.

CONCLUSIONES

- Se implementó exitosamente un **clúster de Kubernetes de alta disponibilidad con balanceadores de carga y almacenamiento compartido**, validando su capacidad para soportar aplicaciones críticas.
- **La API desarrollada y desplegada automatizó la gestión y monitoreo de redes simuladas**, demostrando la integración efectiva con una red virtual en GNS3.
- Se utilizaron herramientas como **Helm y Calico para optimizar el despliegue y la comunicación en el clúster**, mejorando la eficiencia y escalabilidad del entorno.
- El análisis del despliegue fallido de **ONAP** resaltó las limitaciones de los entornos locales para aplicaciones de gran escala, subrayando la utilidad de servicios en la nube.

TRABAJO FUTURO

Como trabajo futuro se busca explorar la integración de tecnologías en la nube para superar las limitaciones de recursos en entornos locales, permitiendo el despliegue exitoso de plataformas complejas como **ONAP**. También, se plantea desarrollar nuevas funcionalidades en la API que amplíen su capacidad de **automatización y monitoreo en diversas redes**, además de realizar pruebas en escenarios reales de mayor escala y complejidad para validar la estabilidad y escalabilidad del clúster

REFERENCIAS

1. Kubernetes, (2024), Creating Highly Available Clusters with kubeadm
2. The Linux Foundation, (2024), Helm docs

DEMOSTRADOR

MÉTODOS

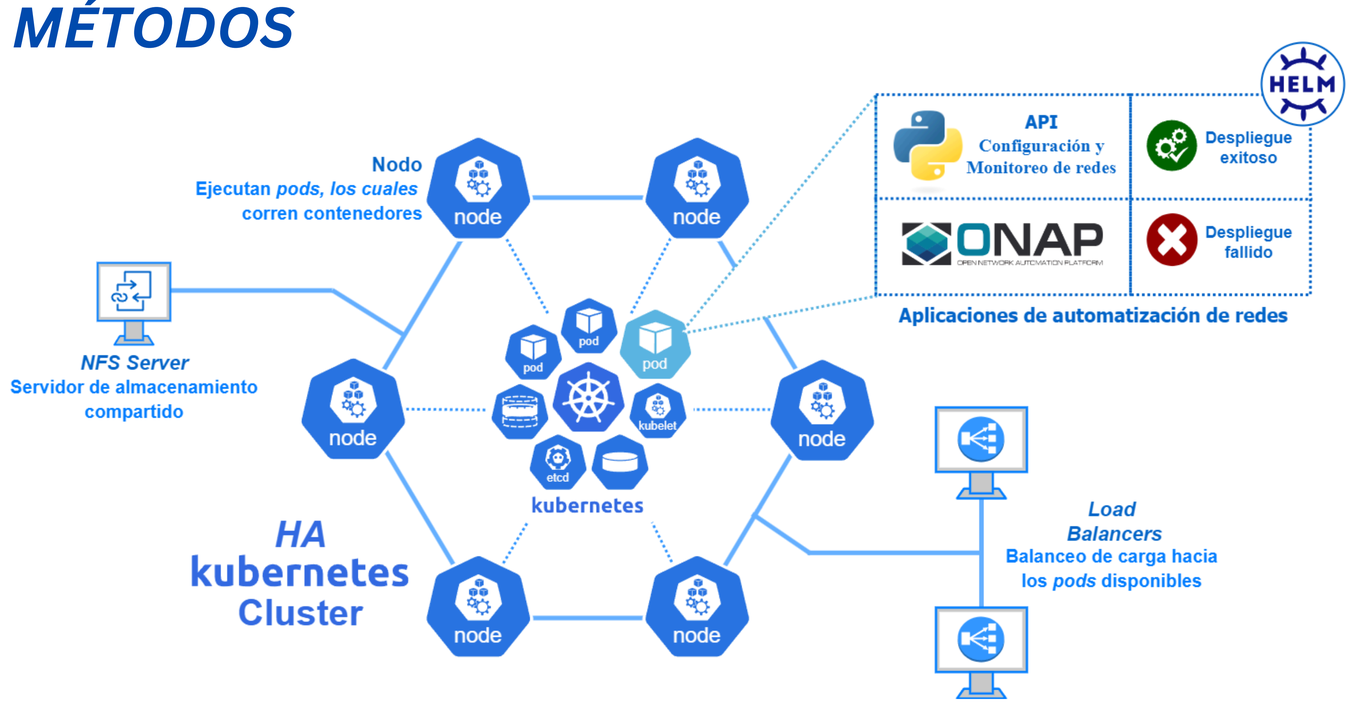

Figura 5: Arquitectura del clúster Kubernetes de alta disponibilidad con integración de la API para automatizar configuración y monitoreo de redes.

RESULTADOS

Se implementó un **Clúster de Kubernetes de alta disponibilidad**, donde se desplegó una **API, a través de Helm, para automatizar redes** y se validó su integración con dispositivos simulados en una **red en GNS3**.

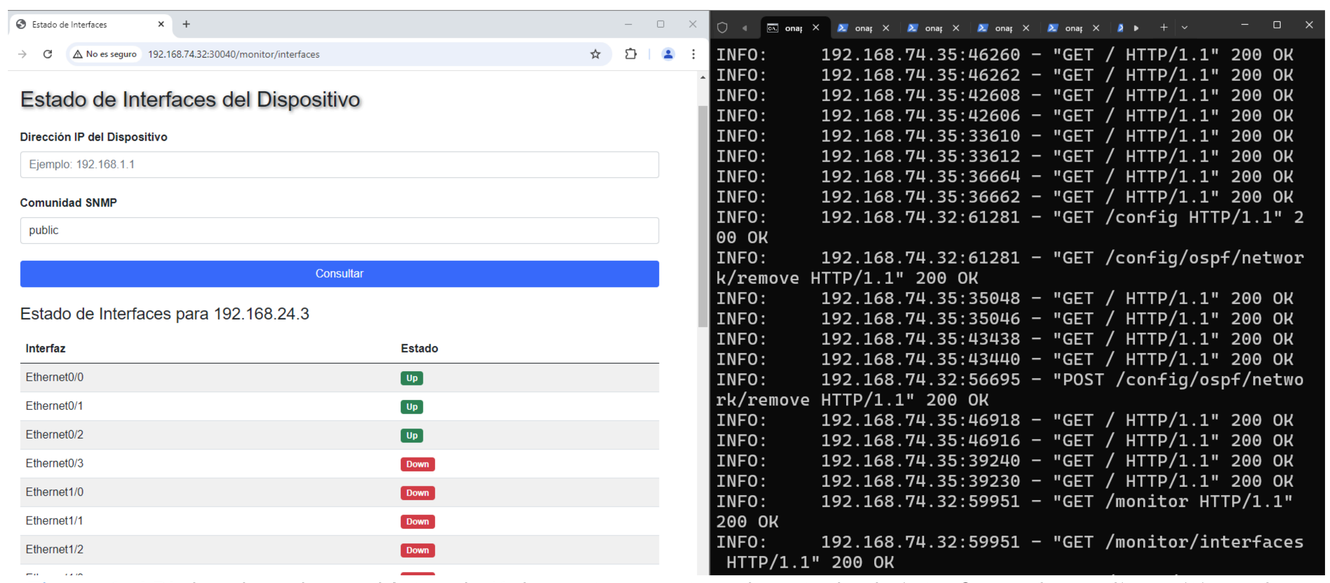

Figura 1: API desplegada en clúster de Kubernetes mostrando estado de interfaces de un dispositivos de red y los logs del pod en tiempo real.

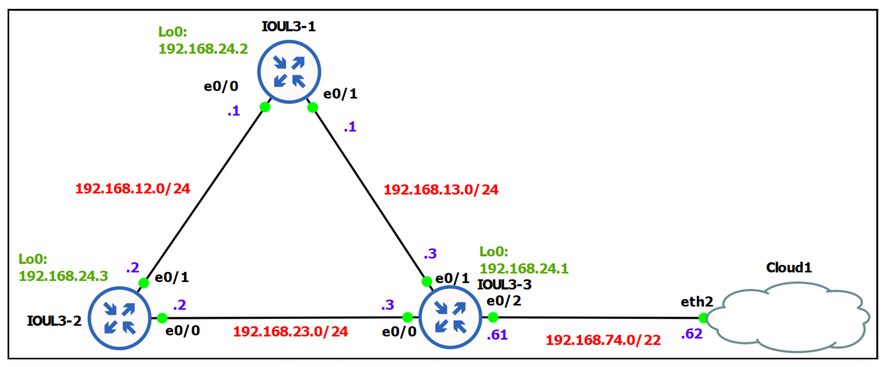

Figura 2: Topología de red simulada en GNS3 utilizada para pruebas de integración con la API desplegada.

```
onap@onapnodo1:~$ kubectl get nodes -o wide
NAME        STATUS   ROLES           AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
onapnodo1   Ready    control-plane   7d5h   v1.27.5   192.168.74.32   <none>        Ubuntu 22.04.4 LTS
onapnodo2   Ready    control-plane   7d5h   v1.27.5   192.168.74.33   <none>        Ubuntu 22.04.4 LTS
onapnodo3   Ready    control-plane   7d5h   v1.27.5   192.168.74.34   <none>        Ubuntu 22.04.4 LTS
onapnodo4   Ready    <none>          7d5h   v1.27.5   192.168.74.35   <none>        Ubuntu 22.04.4 LTS
onapnodo5   Ready    <none>          7d5h   v1.27.5   192.168.74.36   <none>        Ubuntu 22.04.4 LTS
onapnodo6   Ready    <none>          7d5h   v1.27.5   192.168.74.37   <none>        Ubuntu 22.04.4 LTS
```

Figura 3: Listado de nodos del clúster de Kubernetes desplegado, mostrando todos los nodos funcionales y en estado 'Ready'.

Figura 4: Visualización de la tabla de enrutamiento generada por la API, mostrando rutas, prefijos IP y las interfaces asociadas.