



Orquestación de contenedores para la supervisión de infraestructura de redes

Camilo Andrés Mejía Posada

Informe de prácticas académicas para optar por el título de Ingeniería de Sistemas

Asesores

Asesor interno: Fredy Alexander Rivera, PhD en Ingeniería Informática

Asesor externo: Florian Badens, Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia
2022

Cita

(Mejía Posada, 2022)

Referencia

- [1] Mejía Posada, C.A. (2022). Orquestación de contenedores para la supervisión de infraestructura de redes. [Trabajo de grado profesional]. Universidad de Antioquia, Medellín, Colombia

Estilo IEEE (2020)



Rector: John Jairo Arboleda Céspedes.

Decano/Director: Jesús Francisco Vargas Bonilla.

Jefe departamento: Diego José Luis Botia Valderram.

El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Antioquia ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por los derechos de autor y conexos.

Dedicatoria

A mis padres,
mis tías,
y a todos los compañeros que durante el paso por el Alma Máter hicieron parte de mi aprendizaje.

Agradecimientos

Al profesor y asesor Fredy Alexander Rivera por su acompañamiento y atención brindada durante mi intercambio. A mi tutor Florian Badens por los conocimientos que me aportó y la autonomía durante el desarrollo de este proyecto. A todo el conjunto de Axians Toulouse por su calurosa acogida. Finalmente, a la Universidad de Antioquia por darme las oportunidades y enseñanzas que me forjaron como profesional.

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS	2
LISTA DE FIGURAS	4
SIGLAS, ACRÓNIMOS Y ABREVIATURAS	5
RESUMEN	6
ABSTRACT	7
INTRODUCCIÓN	8
OBJETIVOS DEL PROYECTO	9
2.1. Contexto	9
2.2. Objetivos	11
METODOLOGÍA	12
3.1. Tablero de organización de proyecto - JIRA	13
3.2. Base de datos de conocimiento - Confluence	15
CONCEPCIÓN Y DESARROLLO	17
4.1. Kubernetes	17
4.2. Arquitectura	17
4.2.1. Sistema operativo	18
4.2.2. High Availability - Alta Disponibilidad	19
4.2.3. Resultado	20
4.3. Despliegue del Cluster	22
4.3.1. Rancher	22
4.4. Herramientas	23
4.4.1. CNCF	23
4.4.2. Almacenamiento Distribuido Longhorn	23
4.4.2.1. Composición	25
4.4.2.2. Nodos Storage - Almacenamiento	25
4.4.3. Exposición de servicios MetalLB - Ingress Nginx	26
4.4.3.1. Utilización de NodePort	26
4.4.3.2. Utilización de Load Balancer	27
4.4.3.3. MetalLB	28
4.4.3.4. Utilización de Ingress	29
4.4.3.5. LoadBalancer + Ingress Nginx	30
4.4.4. Aislamiento y seguridad de contenedores	31
4.4.4.1. CNI	31
4.4.4.2. Cilium	32

4.4.4.2.1. CiliumNetworkPolicy	32
4.4.4.2.2. Hubble	33
4.5. Backup	35
4.6. Tests	36
DESPLIEGUE DEL STACK	37
5.1. Stack	37
5.1.1. Configuración de Telegraf	37
5.1.2. Guardado y recuperación de datos en InfluxDB.	38
5.1.3. Construcción de tableros y visualización de datos en Grafana	39
5.2. Resiliencia aplicativa	39
5.2.1. Pod affinity y anti-affinity	40
5.2.2. Requests and Limits for containers	40
5.3. Industrialización del proceso de despliegue	40
CONCLUSIONES	41
REFERENCIAS	43

LISTA DE FIGURAS

Figura 1. Diagrama de arquitectura del stack TIG	10
Figura 2. Comparación de alojamiento de aplicaciones	11
Figura 3. Framework SCRUM	13
Figura 4. Tablero de presentaciones	14
Figura 5. Hoja de ruta	14
Figura 6. Tablero de tareas	15
Figura 7. Tablero de documentos más importantes.....	16
Figura 8. Tablero de imagenes de la instalación de la infraestructura en el datacenter.....	19
Figura 9. Diagrama de arquitectura física.....	20
Figura 10. Diagrama de arquitectura Rancher.....	21
Figura 11. Resultados del estudio comparativo entre herramientas.....	23
Figura 12. Arquitectura Longhorn.....	24
Figura 13. Representación del servicio NodePort.....	25
Figura 14. Representación del servicio LoadBalancer.....	26
Figura 15. Representación del servicio Ingress.....	28
Figura 16. ArquitecturaMetalLB + Ingress.....	29
Figura 17. CNP L3 y L4 ejemplo.....	30
Figura 18. Arquitectura Cilium y Hubble.....	31
Figura 19. Políticas CNP configuradas.....	31
Figura 20. Esquema de interconexión y flujo de red del stack en Hubble UI.....	32
Figura 21. Consola de administración Minio.....	33
Figura 22. Ejemplo de archivos de configuración Telegraf	35
Figura 23. Métricas en la plataforma de Influx.....	36
Figura 24. Tableros de métricas en Grafana.....	36
Figura 25. Scripts de despliegue y eliminación.....	38

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

SLA: Service Level Agreement

TIG : Stack compuesto por Telegraf, InfluxDB y Grafana.

Open Source : Software de libre acceso.

Multi-tenant : Una sola arquitectura pero al servicio de múltiples clientes u organizaciones.

Framework : Marco de trabajo.

SNMP : Protocolo de Gestión de Red Simple.

ICMP : Internet Control Message Protocol

TCP : Protocolo de Control de Transmisión.

CNCF : Cloud Native Computing Foundation

ARP : Address Resolution Protocol

CNI: Container Network Interface

eBPF : Tecnología que permite ejecutar programas sobre el kernel de Linux.

CIDR : Classless Inter-Domain Routing

UI : User Interface

XFS : Sistemas de ficheros altamente modulables y altamente eficiente.

CRT : Cuaderno de Tests (Cahier Recette des Tests)

RESUMEN

Durante las prácticas fui encargado, en primer lugar, de elaborar un estudio sobre la construcción y la puesta en marcha de un clúster Kubernetes para la orquestación de los contenedores, de explicar las tecnologías sub-subyacentes y presentar los diferentes bloques necesarios para la realización de una arquitectura resiliente, de alto rendimiento y tolerable a fallos con la finalidad de alojar un stack de supervisión de infraestructura de redes. Después de esta fase, analicé varias herramientas de código abierto que mejoraban el clúster, de tal forma que se pudiera cumplir con los objetivos de la empresa y los clientes en términos de SLA (Service Level Agreement). Este análisis incluyó la presentación de los componentes, estudios comparativos, qué necesidad o problema planteaban, su funcionamiento y puesta en producción en el clúster.

Además, participé en la realización de las pruebas para demostrar la eficacia del clúster y el seguimiento de los objetivos.

Finalmente, implementé una pila de tres servicios para recuperar, almacenar, procesar y trazar métricas de equipos de red de varios clientes y poder satisfacer las necesidades que habíamos establecido durante el curso del proyecto.

***Palabras clave* — Kubernetes, DevOps, Infraestructura, Monitoreo de redes, Resiliencia, Tolerancia a fallos, Sistemas distribuidos.**

ABSTRACT

During the internship, I was first of all in charge of elaborating a study on the construction and implementation of a Kubernetes cluster for the orchestration of containers, explaining the underlying technologies and presenting the different blocks necessary for the realization of a resilient, high-performance and fault-tolerant architecture to host a network infrastructure monitoring stack. After this phase, I analyzed several open source tools that enhanced the cluster, so that it can meet the objectives of the company and customers in terms of SLA (Service Level Agreement). This analysis included the presentation of the components, comparative studies, what need or problem they pose, their operation and implementation in production in the cluster.

In addition, I participated in testing to demonstrate the effectiveness of the cluster and the monitoring of the objectives.

Finally, I implemented a stack of three services to retrieve, store, process and plot metrics from network equipment of various clients in order to meet the needs that we had established during the project.

Keywords - Kubernetes, DevOps, Infrastructure, Network monitoring, Resiliency, Fault tolerance, Distributed systems.

1. INTRODUCCIÓN

A lo largo de los años, la monitorización de las infraestructuras ha desempeñado un papel fundamental en la vigilancia en tiempo real del buen funcionamiento de los equipos materiales que componen una empresa, así como en la detección rápida de las anomalías, que puedan obstaculizar los procesos comerciales, provocar la indisponibilidad de los servicios o provocar vulnerabilidades de seguridad [1].

En el marco de mi último año de ciclo Ingeniero y la realización de un intercambio académico en el INSA-Toulouse, tuve la oportunidad de realizar mi pasantía de fin de estudio en Axians C&C (Communications & Cloud), filial de Vinci Energies, en Labège, Francia.

Axians C&C diseña, implementa y mantiene soluciones para distribuir y gestionar infraestructuras de red. La compañía se actualiza con nuevas soluciones y herramientas para satisfacer las necesidades actuales de monitoreo de los equipos de los clientes.

He querido realizar mis prácticas sobre la temática de DevOps, más específicamente en el campo de los contenedores y su orquestación. Las posibilidades que ofrecen los contenedores permiten imaginar nuevos usos, implementar una arquitectura robusta, servicios resilientes y escalables que solucionan las necesidades empresariales actuales con el fin de valorizar los productos ya existentes en la empresa. El reto de este proyecto fue diseñar e implementar esta infraestructura para mejorar las soluciones propuestas en el catálogo de Axians.

Además, gracias a mi integración en un equipo de ingenieros de desarrollo y red, pude realizar en paralelo varias misiones muy enriquecedoras en la instalación y vigilancia de equipos de red.

Este informe describe el desarrollo de mi pasantía, y las etapas por las que pasé para adaptarme a un entorno laboral extranjero y llegar a la producción del stack propuesto. La primera parte tratará el contexto en el que se realizó este proyecto, por qué era necesario, las herramientas propuestas por la empresa y los problemas actuales. En la segunda parte se abordará la metodología que he seguido para el desarrollo y la administración del proyecto durante el curso. La tercera parte tratará de los estudios, despliegues y herramientas que he realizado o utilizado para la creación de la infraestructura. La cuarta parte tratará sobre el despliegue y la mejora del stack propuesto. La quinta y última parte se centrará en los resultados obtenidos y en el trabajo futuro.

2. OBJETIVOS DEL PROYECTO

2.1. Contexto

Axians C&C Toulouse, como integrador de soluciones informáticas y distribuidor de equipos de red, tiene como objetivo garantizar el correcto funcionamiento y asegurar la operatividad de los equipos dispuestos en las instalaciones de los clientes. La supervisión desempeña un papel muy importante a la hora de proporcionar un apoyo activo y eficaz en caso de problema o interrupción del servicio. Axians siempre se compromete a hacer cumplir su SLA (Acuerdo de Nivel de Servicio) con sus clientes.

Anteriormente, Axians utilizaba herramientas de monitorización ofrecidas por los proveedores de equipos de red, por lo que estas herramientas dependían en gran medida de las licencias de uso y del soporte continuo, y además de esto las herramientas tenían muy poca personalización para las necesidades de cada uno de los clientes de la agencia.

Como resultado, Axians C&C decidió llevar a cabo un estudio para encontrar un stack de monitorización con el compromiso que este sea de código abierto, personalizable y replicable para cada cliente; una solución de monitorización de infraestructura de red multi-tenant.

Este estudio encontró como solución el stack TIG, compuesto por Telegraf, InfluxDB y Grafana, como se muestra en la Figura 1.

Telegraf

Telegraf es un agente recolector de datos, entre los que destacan las métricas de uso de RAM, carga del procesador, tráfico de red, entre otros. La recolección de datos se realiza haciendo uso de los protocolos SNMP, ICMP y TCP [2]. Esta herramienta será el scraper, o recolector de información de la infraestructura del cliente hasta el service desk de la empresa.

InfluxDB

Esta herramienta se utilizará para registrar el estado de la infraestructura comunicada por Telegraf. InfluxDB es una base de datos de lectura/escritura optimizada para almacenar datos en forma de "timestamp : value", y desde la versión 2.0 también permite la visualización de datos en tiempo real [3], pero para esta tarea se utilizará la herramienta descrita a continuación.

Grafana

Grafana es una plataforma de visualización de datos. Esta herramienta permitirá la conexión con la base de datos y la visualización de las métricas en tableros de mando configurables según las necesidades del cliente [4].

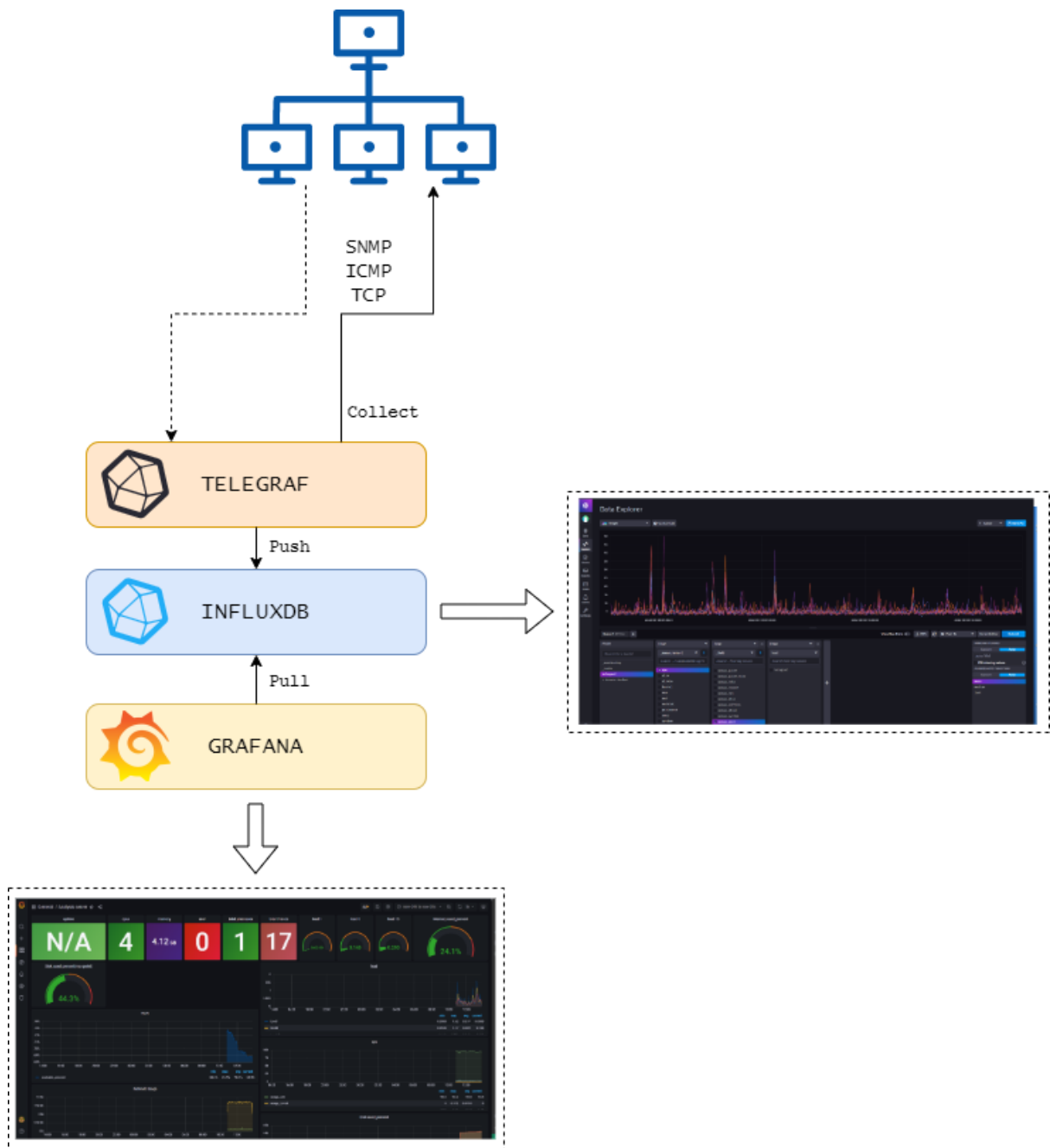


Figura 1. Diagrama de arquitectura del stack TIG

Además, se realizó un segundo estudio para encontrar una forma sencilla y manejable de desplegar el stack, teniendo en cuenta las experiencias anteriores y las tendencias actuales de despliegue en un entorno Cloud on-premise. La empresa decidió hacer una comparación entre el alojamiento en hosts físicos, en máquinas virtuales y en contenedores, una vista general de este estudio está disponible en la Figura 2.

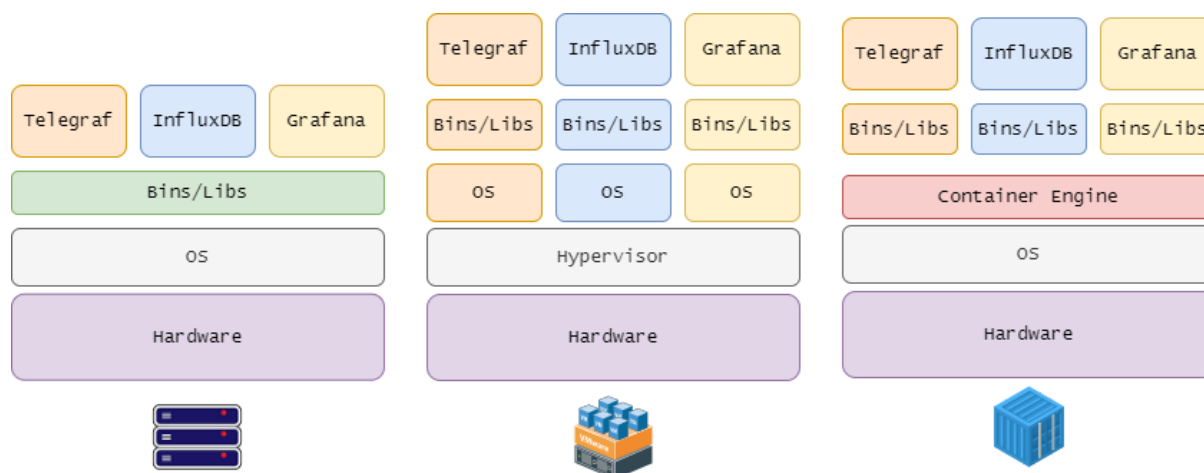


Figura 2. Comparación de alojamiento de aplicaciones

En general, el conjunto de Axians Toulouse se dio cuenta de que la implementación de la tecnología de contenedores les permitía desplegar aplicaciones de forma ligera, aislada y portátil.

En una primera instancia, también decidieron utilizar un orquestador de contenedores básico, Docker Compose, que permite desplegar múltiples contenedores de forma declarativa con una línea de comandos [5]. Pero, en consecuencia a algunas experiencias con su uso, la herramienta no es lo suficientemente potente como para gestionar el estado de las implementaciones, realizar cambios en las aplicaciones sin sufrir cortes de servicio o realizar una recuperación automática en caso de fallos.

En estas condiciones, Axians Toulouse quería implementar un orquestador de contenedores más robusto para el despliegue y la administración de sus pilas.

2.2. Objetivos

General: Implementar una arquitectura de orquestación de contenedores basada en Kubernetes para el monitoreo de la infraestructura de red de sus clientes.

Específicos:

- Identificar los requerimientos de infraestructura para la implementación del *cluster* de Kubernetes.
- Explorar y elegir las herramientas más convenientes para la construcción de un *cluster* de Kubernetes eficiente, resiliente y seguro.
- Implementar el *cluster* y el *stack* tecnológico de servicios para el monitoreo de infraestructura.

- Ejecutar pruebas de resiliencia, tolerancia a fallos y disponibilidad del *cluster*.

3. METODOLOGÍA

Durante la primera fase del curso se discutió la metodología a seguir durante el proyecto. Se adoptó el marco ágil SCRUM (Figura 3): "Esta metodología es esencialmente un marco ágil y ligero que proporciona pasos para gestionar y controlar el proceso de desarrollo de software y productos" (Srivastava, Bhardwaj, Saraswat, 2017).

El propósito de los sprints es dividir el proyecto en diferentes etapas. Para el desarrollo de éste se planificaron ocho sprints con un intervalo de tiempo previamente establecido según el grado de dificultad de las tareas que lo componen. Además, al principio de cada semana se celebraron reuniones para revisar el progreso de las tareas con todo el equipo de desarrollo y el responsable técnico del equipo de integración, en las que se discutió el progreso de cada tarea, los aprendizajes obtenidos y las definiciones de prioridades para el siguiente sprint.

Por otra parte, se realizaron presentaciones y exposiciones para entender mejor cómo funciona Kubernetes, qué herramientas utilizar, qué decisiones tomar y cómo resolver los problemas que puedan surgir durante el desarrollo. Algunos ejemplos de estas presentaciones están disponibles en la Figura 4.



Figura 3. Framework SCRUM [6]

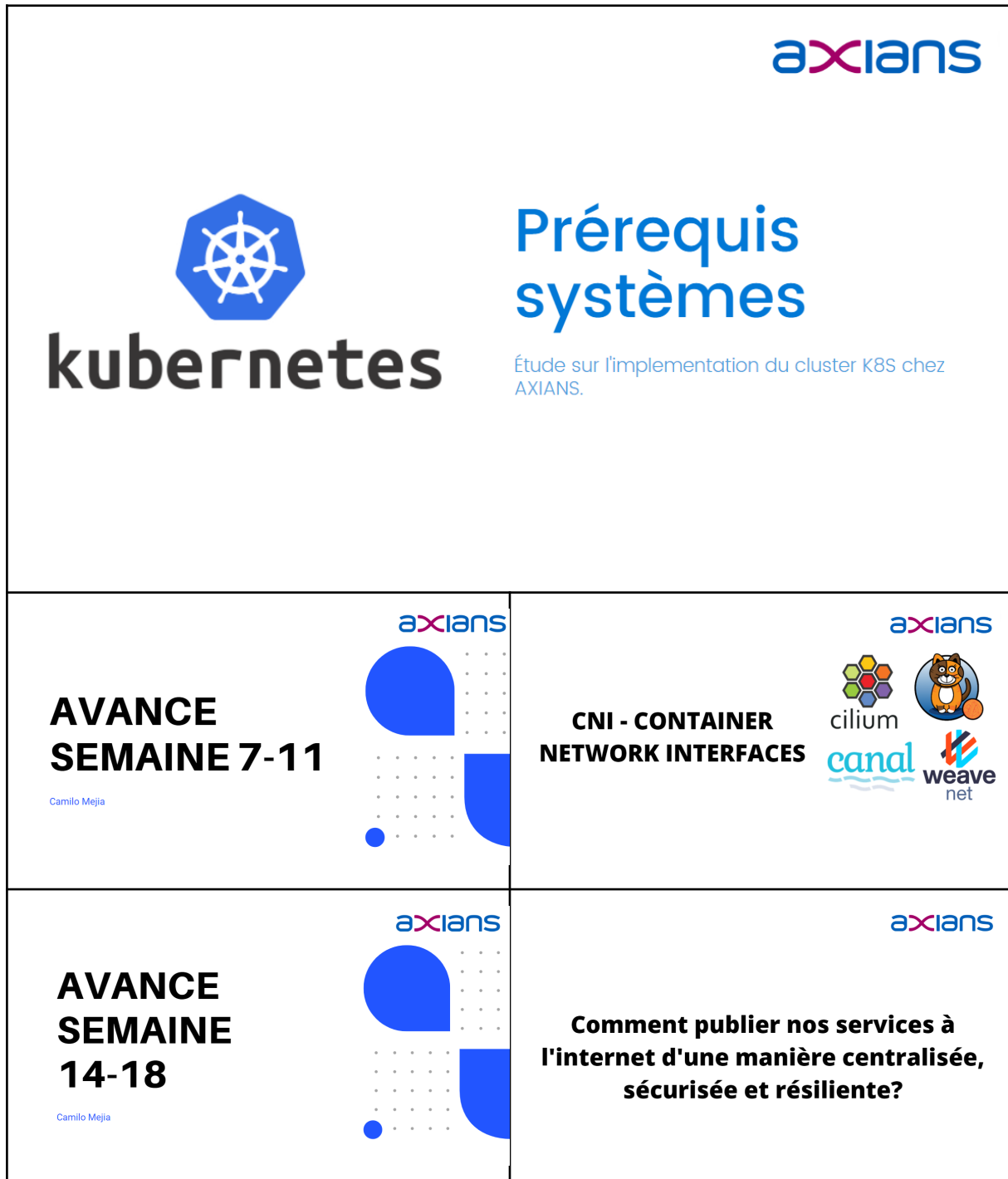


Figura 4. Tablero de presentaciones

3.1. Tablero de organización de proyecto - JIRA

Aunque no existe una relación evidente entre la planificación del proyecto y su éxito, según el PMBOK (Project Management Body of Knowledge), la falta de planificación es garantía

de fracaso. Es importante destacar que existe una relación positiva entre el esfuerzo invertido en la definición de los objetivos, los requisitos funcionales y las especificaciones técnicas a la hora de definir el éxito de un proyecto, especialmente a los ojos del usuario final. [7]

Para organizar el proyecto y especificar cada objetivo y sus requisitos, se decidió utilizar la herramienta Jira. Jira ofrece herramientas de planificación y hojas de ruta que permiten a los equipos gestionar los participantes en los proyectos, las tareas, los objetivos y los requisitos de las funciones [8]. Para el desarrollo de este proyecto se elaboró una hoja de ruta (figura 5) que abarcó desde el diseño y los estudios del proyecto hasta la ejecución y el desarrollo de las pruebas. Del mismo modo, se creó un tablero (figura 6) que separa cada tarea según su estado y progreso, con los siguientes bloques de separación definidos:

Backlog: Lista de las tareas de la hoja de ruta disponibles para realizar.

Semana actual: Lista de tareas a realizar durante la semana laboral.

Puntos de bloqueo : Lista de tareas que tienen inconvenientes.

En curso: Lista de tareas que están en curso.

Revisión/Punto semanal: Lista de tareas completadas que serán discutidas entre el equipo de desarrollo y el estudiante en las reuniones semanales para definir un estado final. La tarea que haya alcanzado los resultados esperados se colocará en el bloque "Finalizada"; de lo contrario, es probable que la tarea se modifique o tenga adiciones a sus objetivos y vuelva al estado "En curso".

Finalizada: Listas de tareas finalizadas.

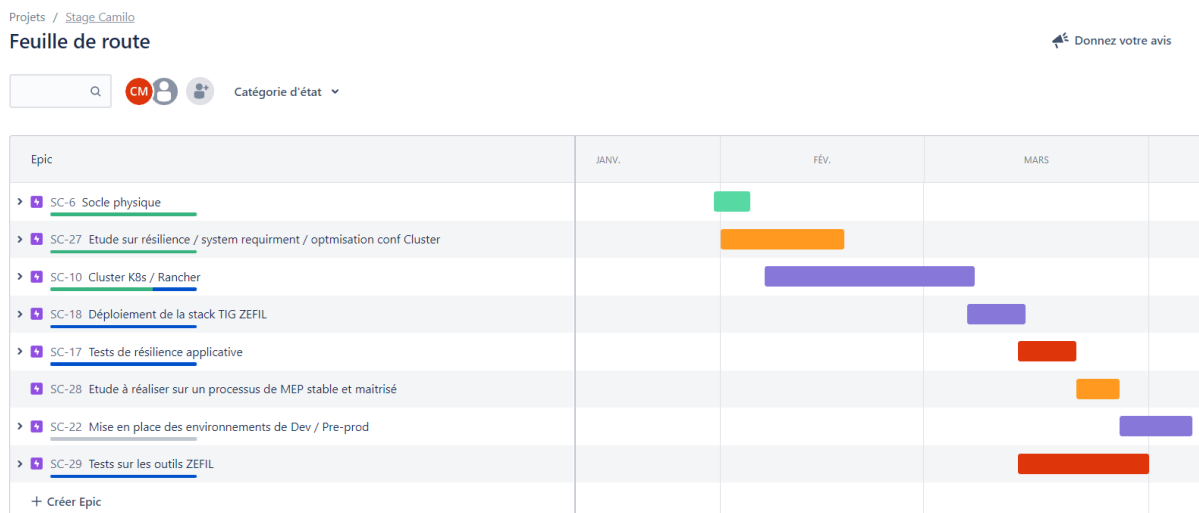


Figura 5. Hoja de ruta

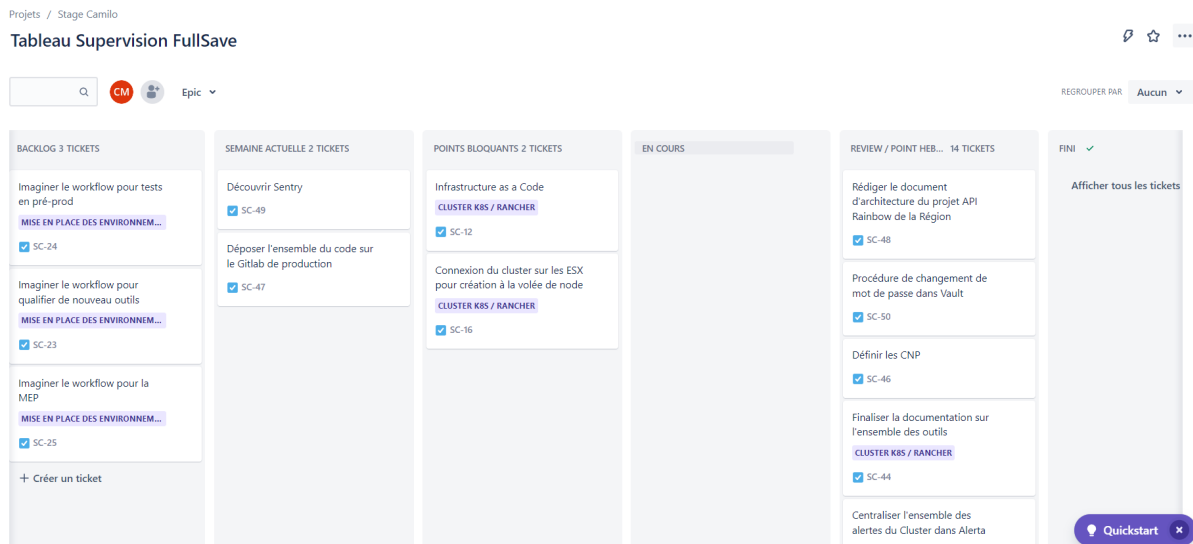


Figura 6. Tablero de tareas.

3.2. Base de datos de conocimiento - Confluence

La realización de la documentación en un proceso de desarrollo garantiza la calidad, clarifica los procesos y facilita la toma de decisiones [9]. El desarrollo de un proyecto debe estar orientado a la documentación, que es el resultado más valioso del proceso [10].

Con base en lo anterior y considerando la importancia que el proyecto podría tener en el desarrollo empresarial de la compañía, se realizaron archivos de documentación donde se representan los procesos cognitivos y tecnológicos del proyecto, el estado del arte y la toma de decisiones, para proporcionar no sólo un rastro del trabajo realizado sino también una forma de comunicar las experiencias del proyecto y los conocimientos adquiridos durante su desarrollo. Los documentos fueron escritos en la plataforma Confluence, lo cual se debe a que la misma marca Atlassian (de la cual forma parte Jira) nos ofreció esta alternativa para conectar el tablero de tareas y la documentación de cada una. Confluence es un software de colaboración en equipo que dispone de varias plantillas y bases para especificar los documentos empresariales [11].

Estos documentos trazan los estudios realizados para la introducción de nuevas tecnologías y herramientas para enriquecer el funcionamiento del clúster o, alternativamente, para resolver un problema descubierto durante el desarrollo del clúster. Para la realización de estos documentos, se tuvo en cuenta la siguiente estructura:

Problema/Necesidad: En caso de encontrar una limitación técnica durante nuestros objetivos o un deseo de mejora, formulamos una pregunta para resolver.

Explicación de la herramienta que respondió a nuestro problema: En esta parte, se da una breve explicación de la herramienta, una explicación técnica de su funcionamiento y, si

es necesario, un estudio comparativo entre esta herramienta y otras disponibles en el mercado.

Instalación/despliegue: Descripción paso a paso de la instalación de la herramienta en el clúster.

Bibliografía : Enlaces a los artículos y documentaciones que fueron seguidos para la redacción del documento.

Algunos ejemplos de la redacción de estos documentos pueden ser encontrados en la siguiente figura, en el idioma francés, donde se pueden destacar los temas de *Longhorn*, *MetalLB* y *Cilium*, temas que serán abordados en la sección [4.4](#).

LONGHORN - Réplication d'information sur le cluster

Créateur : Camillo Mejía
Dernière mise à jour : le mars 28, 2022

Problème

Pendant la construction d'un cluster disponible et fiable, on s'est posé une question:

- Comment faire la réplication d'information sur tous les nœuds afin d'augmenter la disponibilité du cluster? 🤔

Pour donner une réponse à cette question, on a examiné deux outils recommandés pour le [CNCF](#) qui servent pour faire la réplication d'information sur un cluster, on a évalué leur maturité, leur installation et leur exploitation, celles-ci sont les résultats.

Outil	Répond-il aux besoins?	Est-ce CNCF Graduated?	Est-il facile à installer?	Est-il facile à exploiter?	Permet-il de faire des snapshots et backup/restore
Rook	✓	✓	✗	✗	✓
Longhorn	✓	✗	✓	✓	✓

Après avoir fait les différentes installations, des mises en places et tests, on a découvert que **Rook** est beaucoup plus mature car il est CNCF Graduated (donc il comptait avec l'approbation du CNCF pour des environnements productifs), néanmoins il était très difficile à installer et à utiliser et ne comptait pas avec une UI native, par contre **Longhorn** répondait très bien à nos besoins, s'intégrait très facilement avec notre cluster et il comptait avec une interface facile à utiliser et très intuitive, même s'il n'est pas CNCF Graduated. **Longhorn** nous donne aussi la possibilité de faire des snapshots récurrents et de les stocker sur une base de données d'objets tels que Minio ou S3 du AWS.

Longhorn

Longhorn est un système de stockage en bloc distribué pour Kubernetes, il nous permet de:

METALLB - LoadBalancer On Premise

Créateur : Camillo Mejía
Dernière mise à jour : le mars 31, 2022

Problème

Pendant la construction du cluster et se projetant sur d'autres applications qui seront hébergées dans notre cluster, on s'est posé la question suivante:

Comment publier nos services à l'extérieur d'une manière centralisée, sécurisée et résiliente?

Nous avons donc entamer cette réflexion et avons imaginé les possibilités d'amélioration

NodePort

Un service NodePort est le moyen le plus primitif/basique pour obtenir du trafic externe directement à votre service. NodePort, comme son nom l'indique, ouvre un port spécifique sur tous les Nodes (les VMs), et tout trafic envoyé à ce port est transmis au service.

CILIUM - Surveillance et sécurité du network sur les conteneurs

Créateur : Camilo Mejia
Dernière mise à jour : le mars 28, 2022

Souhait

Avec l'intention d'avoir un isolement complet des namespace entre eux et les services qui les composent et en plus d'une meilleure sécurité et visibilité du trafic, on a décidé d'implémenter un plugin du CNI qui nous aide à accomplir ce souhait.

Container Network Interface (CNI)

Un CNI est responsable de l'insertion d'une interface réseau dans le namespace du réseau du conteneur et d'apporter les modifications nécessaires sur l'hôte. Il assigne ensuite une adresse IP à l'interface et configure les routes compatibles avec la section Gestion des adresses IP en invoquant le plugin IPAM (IP Address Management).

Cilium

Qu'est-ce que c'est?

Cilium est un CNI plugin pour sécuriser de manière transparente la connectivité réseau entre les services applicatifs avec l'utilisation du eBPF.

eBPF

eBPF est une nouvelle technologie de noyau Linux, qui permet l'insertion dynamique de la visibilité de sécurité puissante et la logique de contrôle au sein de Linux lui-même. Comme eBPF fonctionne à l'intérieur du noyau Linux, les politiques de sécurité Cilium peuvent être appliquées et mises à jour sans aucune modification du code de l'application ou de la configuration du conteneur.

Hubble

Hubble s'appuie sur Cilium et eBPF pour permettre une visibilité approfondie de la communication et du comportement des services ainsi que de l'infrastructure réseau de manière totalement transparente.

Détection rapide d'échec du node

Créateur : Camilo Mejia
Dernière mise à jour : le avr. 08, 2022

Pour que Kubernetes puisse détecter un noeud en état d'échec plus rapidement et qu'il puisse basculer les services qui étaient sur ce noeud, on a mis quelques configurations générales sur la configuration du cluster pour faire un override au **5 minutes par défaut** que Kubernetes prend pour mettre le taint NoExecute sur le noeud et basculer les services.

Comportement par défaut de Kubernetes par rapport au NodeFailure.

⚠ Cette configuration là affecte tous les déploiements présents sur le cluster, il est possible de faire un override avec [tolerations](#) sur chaque déploiement séparé.

```

1 services:
2 kubelet:
    
```

Figura 7. Tablero de documentos más importantes

4. CONCEPCIÓN Y DESARROLLO

4.1. Kubernetes

El uso de la tecnología Kubernetes para la orquestación de contenedores permite la automatización de tareas, así como la posibilidad de construir stacks tecnológicos resilientes, seguros, estables y eficientes. Además, ofrece la posibilidad de gestionar los recursos y las herramientas utilizadas de forma declarativa para garantizar la ejecución de tareas y procesos de forma estable e industrial (automatizada) [12].

4.2. Arquitectura

Para la definición de la arquitectura, es necesario conocer la composición de un clúster Kubernetes, por consiguiente se realizó una remisión directa a la documentación oficial para conocer mejor cada elemento que compone un clúster, su papel y su importancia en el ecosistema Kubernetes [13].

Un clúster Kubernetes se compone de los siguientes roles:

ETCD: Este puede ser el rol más importante, ya que el ETCD permite registrar la información del clúster (estado, cambios, implementaciones de los usuarios, etc.). Según la

documentación de Kubernetes y el sistema ETCD [14], no sólo es importante dotarlo de un buen espacio de almacenamiento como es lógico, sino que también hay que centrarse en una buena capacidad de memoria y procesador, debido a las operaciones que realizan estos nodos.

Los nodos ETCD se rigen y se basan en el protocolo RAFT [15], un protocolo de selección de líderes. Para que este protocolo se respete, es necesario asignar al menos 3 nodos ETCD para tolerar la pérdida de uno, por lo que se decidió elegir este mismo número para la construcción del clúster.

Dadas las recomendaciones y el tamaño del clúster a construir, se tomó la decisión de asignar 4 núcleos de CPU, 8GB de memoria y 40GB de almacenamiento para un correcto funcionamiento.

Control Plane: Los nodos con el rol Control Plane son los encargados de gestionar el clúster. Su labor, además de programar y desplegar aplicaciones o acciones de usuario final, es la de supervisar constantemente los eventos que se producen en el clúster para garantizar el buen funcionamiento de los despliegues, y también comprobar el estado de los pods, contenedores y nodos.

Teniendo en cuenta las recomendaciones, decidí que estos nodos deberían tener también 4 núcleos de CPU, 8 GB de RAM y 30 GB de almacenamiento.

Workers: Son los nodos responsables de sostener nuestros despliegues, nuestra fuerza de trabajo. La asignación de sus recursos depende del peso y la carga computacional de cada herramienta. En cualquier caso, siempre tenemos la posibilidad de aumentar el número de nodos de los trabajadores de forma horizontal, para que la carga en el clúster esté siempre distribuida de manera uniforme.

Por lo tanto, es lógico que se haya decidido asignar recursos informáticos más potentes, de modo que se tenga nodos eficientes, pero dejando la posibilidad de multiplicarlos si la situación lo requiere. Los recursos elegidos fueron: 8 núcleos de CPU, 16 GB de memoria y 50 GB de almacenamiento.

4.2.1. Sistema operativo

Para la elección del sistema operativo que utilizarían los nodos se realizaron estudios comparativos entre las opciones disponibles. Dado el deseo de garantizar que el clúster siga teniendo carácter de código abierto (open-source), se tomaron como candidatas a CentOS, openSUSE, rancherOS y Debian. La elección del sistema operativo fue una decisión conjunta con el equipo de desarrollo, en la que Debian 11 fue la candidata ganadora, ya que era la opción que ofrecía una mayor estabilidad como sistema operativo, una documentación y

comunidad más amplia, eficiencia, un soporte más prolongado y finalmente era el sistema operativo en el que el equipo tenía más conocimientos y experiencia.

4.2.2. High Availability - Alta Disponibilidad

Para el desarrollo de este proyecto, Axians Toulouse proporcionó 2 servidores Dell PowerEdge R630 con VMWare ESXi vSphere 7.0. Para la interconexión de la red se utilizaron dos firewalls y dos switches.

Se decidió alojar toda la infraestructura física en un centro de datos de alta disponibilidad, que permitiera, entre otras cosas, el funcionamiento, la conexión y la seguridad las 24 horas del día. Debido al estrecho contacto que mantendría con la infraestructura, Axians me permitió estar presente durante la instalación de la infraestructura en el centro de datos, donde fui testigo de todo el proceso por el que se instala y configura un servidor físico, así como su conexión con los switches y sus configuraciones de enrutamiento y los firewalls y sus configuraciones de seguridad. A continuación se muestran algunas imágenes de la instalación de los servidores en el datacenter.

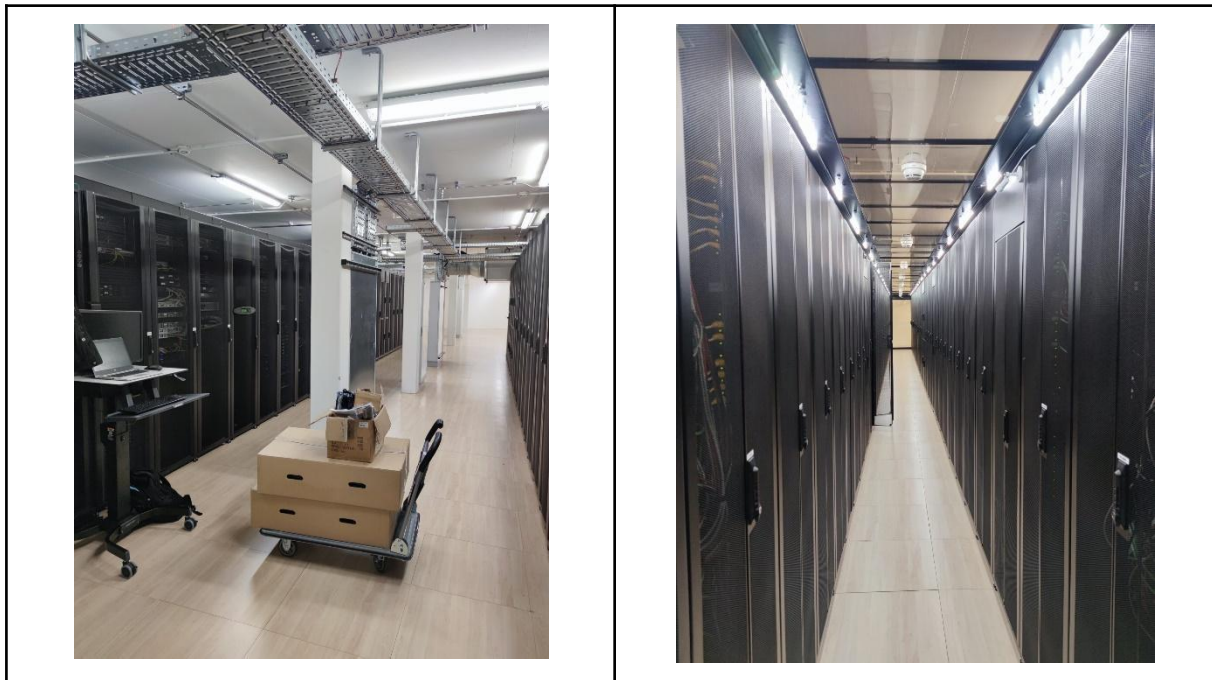




Figura 8. Tablero de imágenes de la instalación de la infraestructura en el datacenter

Teniendo en cuenta la documentación donde se habla de la verdadera Alta Disponibilidad de Kubernetes en un entorno de nube, es necesario desplegar los componentes en tres zonas de disponibilidad diferentes y separadas.

Considerando el contexto del proyecto y el hecho de que la empresa prefiere cimentar sus operaciones en una nube privada o en un entorno on-Premise, pero teniendo el fin de disponer de un clúster con verdadera resiliencia y alta disponibilidad, el equipo de desarrollo decidió utilizar un tercer servidor, esta vez alojado en las instalaciones de la empresa en Labège e interconectado a través de una VPN IPSec Site2Site con la infraestructura principal alojada en un centro de datos de alta disponibilidad como explicado anteriormente. En este tercer servidor se aloja un único nodo con el rol de ETCD para poder mantener el principio del protocolo Raft en caso de fallo en uno de los dos servidores.

4.2.3. Resultado

Finalmente, se obtuvo la arquitectura ilustrada en la Figura 9.

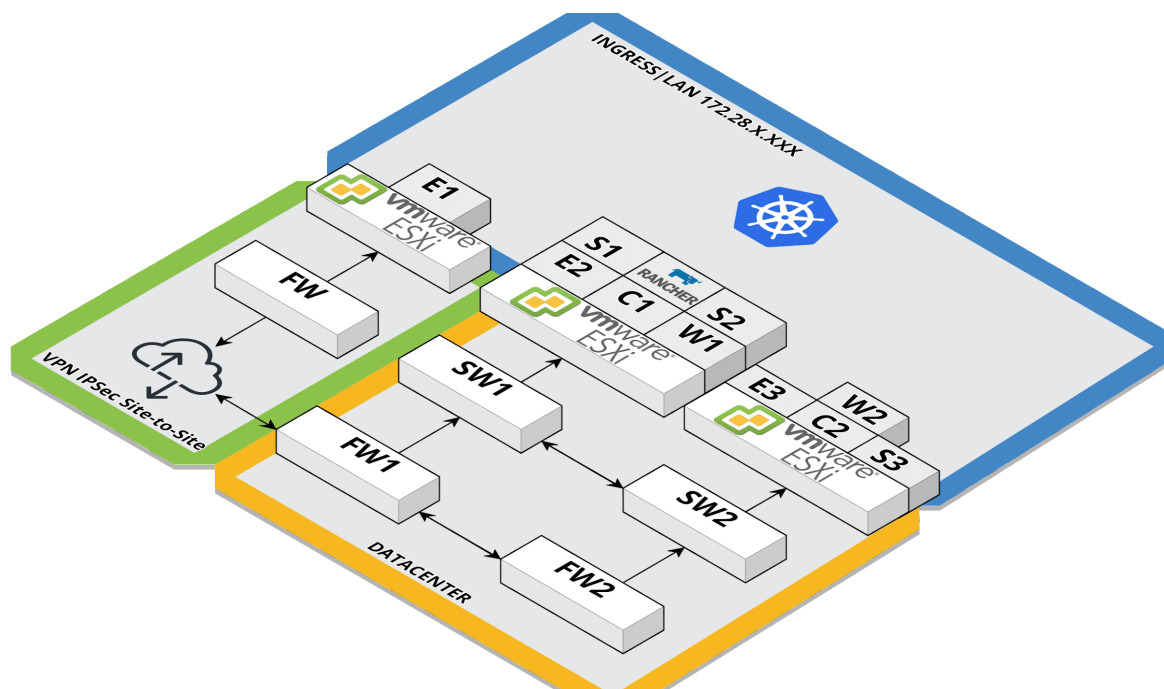


Figura 9. Diagrama de arquitectura física

Cada bloque representa una entidad física o virtual dentro del clúster, como se explica a continuación:

Bloque FWn: Firewalls físicos.

Bloque SWn: Switches físicos.

Bloque VMWare: Servidor de virtualización físico.

Bloque En: Máquinas virtuales con rol ETCD.

Bloque Cn: Máquinas virtuales con el rol de Control Plane.

Bloque Wn: Máquinas virtuales con el rol de trabajador.

Bloque Sn: Máquinas virtuales con el rol de almacenamiento, estos bloques se explicarán con más detalle en la sección 4.4.2.2.

Bloque Rancher: Máquina virtual que aloja el servidor Rancher, explicado con más detalle en la sección 4.3.1.

En esta arquitectura se destaca el uso de servidores y sus separaciones lógicas y físicas para lograr el objetivo de disponibilidad. Además, se puede destacar el seguimiento de las prácticas recomendadas y puestas a disposición para construir un cluster de Kubernetes.

4.3. Despliegue del Cluster

4.3.1. Rancher

Para la adopción de roles en cada máquina virtual del cluster se utilizó la herramienta Rancher. Esta herramienta de código abierto, también basada en Kubernetes, facilita la adopción y ejecución de Kubernetes en entornos de producción, al permitir la construcción y gestión de clusters desde un único punto [16].

Rancher permite el despliegue de los recursos necesarios para ejecutar Kubernetes utilizando la tecnología Docker. Dependiendo del rol elegido, Rancher despliega varios contenedores en la máquina virtual para que tenga las herramientas necesarias para su correcto funcionamiento dentro del clúster, además de otros contenedores que permiten la interconexión con el servidor de Rancher, que es donde se gestionan los nodos.

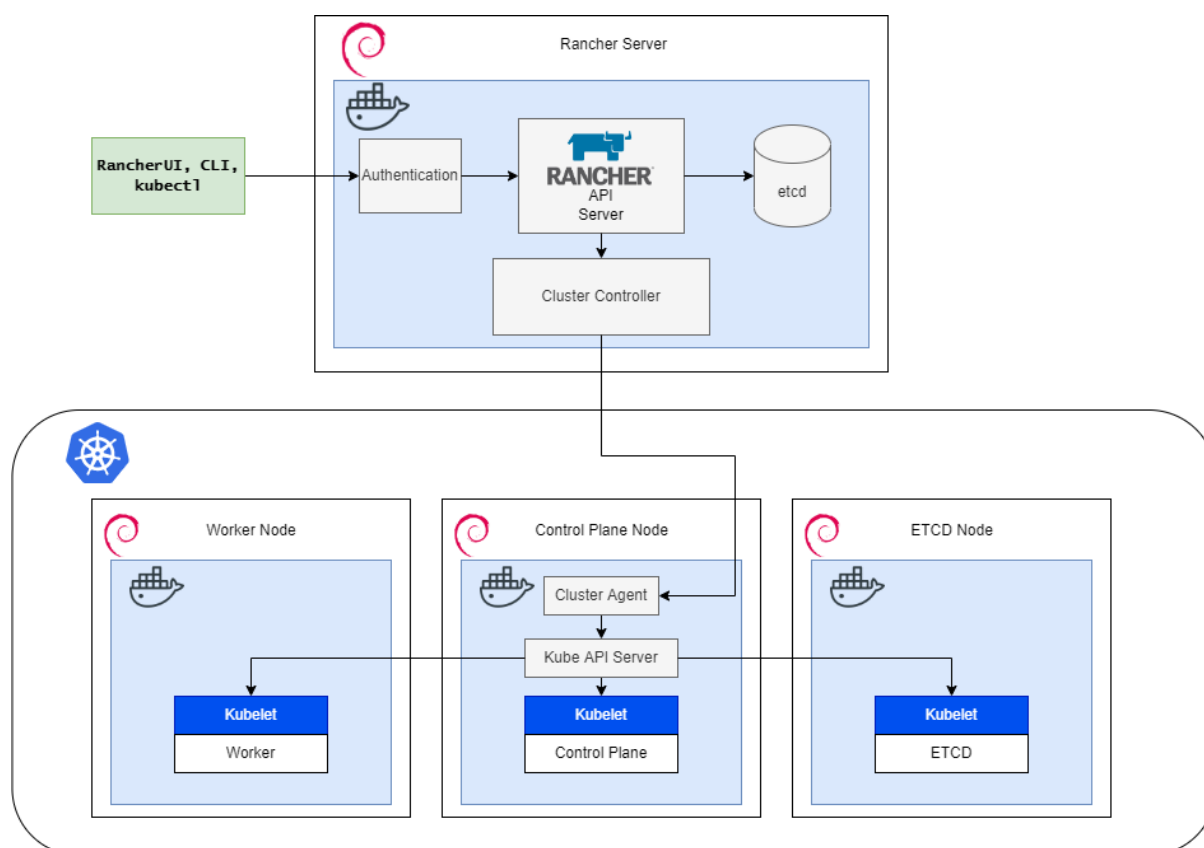


Figura 10. Diagrama de arquitectura de Rancher [17]

Es importante señalar que el despliegue de Rancher se realizó con la herramienta Docker, este contenedor es a su vez un agente de Kubernetes uninodo que utiliza las ventajas que ofrece la herramienta para gestionar múltiples clústeres. La existencia y disponibilidad del servidor Rancher es vital para la administración del cluster. Tal como se muestra en la figura 10 el servidor de Rancher sirve como puerta de entrada, autenticación y enlace hacia el cluster, sin él, los servicios desplegados seguirán funcionando con normalidad pero el cluster no será

gestionable ni administrable, ya que no habrá enlace entre el administrador y el Agente del Cluster desplegado en cada nodo del Control Plane, el cual permite la comunicación con el Servidor Kube API. La elección de Rancher fue hecha por el equipo de Axians Toulouse, una decisión con la que estuve de acuerdo, debido al deseo de usar, explorar y dominar la herramienta.

4.4. Herramientas

Durante la construcción del clúster, se enfrentaron varias dificultades a nivel técnico y funcional que impedían alcanzar los objetivos fijados. Para cada una de estas adversidades se planteó un problema a resolver y se ejecutó un estudio en el que se buscaban, se comparaban, se probaban y se discutían algunas herramientas para encontrar una solución que satisficiera a los miembros del equipo de desarrollo.

En este apartado se hablará un poco de las herramientas y estudios realizados que se han traducido en la base de conocimientos explicada en el apartado [3.2](#).

4.4.1. CNCF

Como se ha explicado anteriormente, el desarrollo del proyecto está orientado al uso de herramientas de código abierto para tener acceso a su código fuente y documentación de tal forma que se pueda adaptarlos sin tener que pagar costosas licencias [18]. El análisis de las herramientas para abordar los problemas o necesidades utilizará la web de la Cloud Native Computing Foundation (CNCf) como punto de partida.

El CNCf es el centro de código abierto sin ánimo de lucro para los proveedores de nubes nativas, que alberga proyectos como Kubernetes para hacer que la nube nativa sea universal [19].

4.4.2. Almacenamiento Distribuido | Longhorn

Una de las primeras preguntas que se formuló durante el desarrollo del proyecto fue: ***"¿Cómo replicar la información en todos los nodos para aumentar la disponibilidad del clúster?"*** Esta pregunta tiene su origen en el hecho de que la responsabilidad de alojar un servicio del stack y la información que se guarda (InfluxDB/Grafana) podría recaer en cualquier nodo con rol de Worker en el clúster, por lo que la información debería estar centralizada, ser consistente y estar disponible para que, independientemente de dónde se preste el servicio, la información sea siempre la misma.

Para responder a esta pregunta, se hizo uso de la página CNCf descrita en el artículo anterior y se realizó un estudio comparativo entre las soluciones propuestas, Rook y Longhorn, donde

se evaluó su madurez, instalación y funcionamiento, cuyos resultados se muestran en la Figura 11..

Outil	Répond-il aux besoins?	Est-ce CNCF Graduated?	Est-il facile à installer?	Est-il facile à exploiter?	Permet-il de faire des snapshots et backup/restore
Rook	✓	✓	✗	✗	✓
Longhorn	✓	✗	✓	✓	✓

Herramienta	¿Responde a las necesidades?	¿CNCF Graduated?	¿Fácil de instalar?	¿Fácil de utilizar?	¿Permite realizar snapshots, backups y restores?

Figura 11. Resultados del estudio comparativo entre herramientas

Después de hacer las diferentes instalaciones, configuraciones y pruebas, se descubrió que Rook es mucho más maduro ya que es CNCF Graduado (por lo que ha sido aprobado por CNCF para entornos productivos). Sin embargo, era muy difícil de instalar y utilizar y no estaba equipado con una interfaz de usuario nativa. Longhorn, en cambio, satisface muy bien las necesidades, se integra muy fácilmente con nuestro clúster y tiene una interfaz fácil de usar y muy intuitiva, a pesar de no estar graduada por CNCF. Longhorn también nos da la posibilidad de hacer snapshots recurrentes y almacenarlas en una base de datos de objetos como Minio o S3 en AWS.

Longhorn es un sistema de almacenamiento de bloques distribuido para Kubernetes, nos permite :

- Utilizar los volúmenes Longhorn como almacenamiento persistente para las aplicaciones *stateful* distribuidas en nuestro clúster Kubernetes.
- Particionar nuestro almacenamiento en bloque en volúmenes Longhorn para poder utilizar volúmenes Kubernetes sin un proveedor de nube pública.
- Replicar el almacenamiento en bloque en varios nodos y centros de datos para aumentar la disponibilidad.
- Guardado de los datos y de las copias de seguridad en un almacenamiento externo como AWS S3 o Minio.
- Programar snapshots recurrentes de un volumen y programar copias de seguridad recurrentes en un almacenamiento secundario compatible.
- Restaurar volúmenes a partir de una copia de seguridad.

4.4.2.1. Composición

Longhorn posee una arquitectura como la que se presenta en la Figura 12, cuya composición es la siguiente:

- Un **manager** que se ejecuta en cada nodo del clúster y se encarga de crear y gestionar los volúmenes en el clúster de Kubernetes.
- Un **CSI-Plugin** que permite la comunicación entre Kubernetes y Longhorn. El plugin CSI nos da la capacidad de crear, borrar, adjuntar, desprender, montar el volumen y tomar instantáneas del volumen.
- Un **Engine** que es desplegado cuando el gestor de Longhorn necesita crear un volumen, y es responsable de crear, almacenar y administrar la información de la réplica.

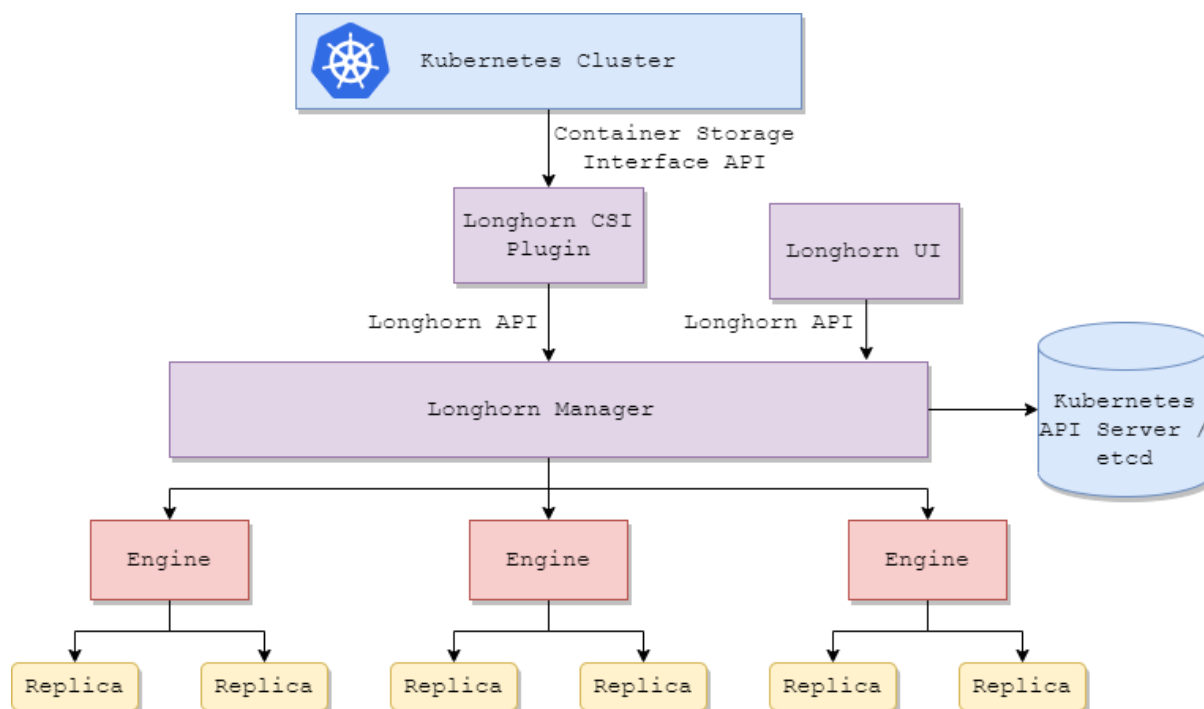


Figura 12. Arquitectura Longhorn [21]

4.4.2.2. Nodos Storage - Almacenamiento

Siguiendo las buenas prácticas al utilizar Longhorn, es aconsejable utilizar nodos de almacenamiento. El uso de nodos de almacenamiento nos permite aislar los nodos donde se aloja la información de los servicios, consiguiendo una separación lógica entre los nodos que tendrán la carga de los servicios y los nodos que tendrán los datos y métricas que producen los servicios [22]. Como resultado, estos nodos tendrán una gran capacidad de almacenamiento. Por lo tanto, se decidió asignar los siguientes recursos: 4 núcleos de CPU, 8 GB de memoria y 500GB de almacenamiento.

4.4.3. Exposición de servicios | MetalLB - Ingress Nginx

Una vez que el proyecto estaba lo suficientemente maduro, se planteó la duda acerca de *cómo publicar nuestros servicios fuera de la red de forma centralizada, segura y resiliente*. Teniendo en cuenta lo anterior, se pudo evidenciar las limitaciones que tenía en ese momento los despliegues realizados y por qué no serían lo suficientemente eficaces para lograr el objetivo original.

4.4.3.1. Utilización de NodePort

Al principio, se hacía uso del servicio NodePort, que es la forma más primitiva y básica de hacer llegar el tráfico externo directamente a nuestros servicios. NodePort, como su nombre indica, abre un puerto específico en todos los nodos (las máquinas virtuales), y cualquier tráfico enviado a ese puerto se reenvía al servicio, tal como se muestra en la Figura 13.

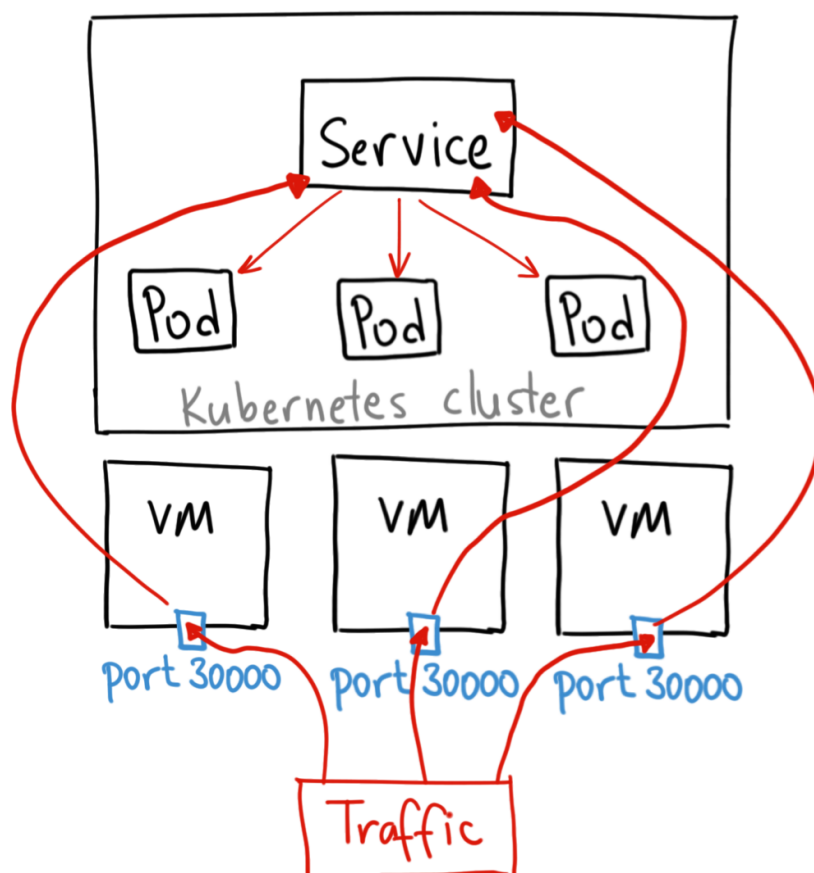


Figura 13. Representación del servicio NodePort [23]

El uso de NodePort tenía como explicación el pensamiento que nuestros servicios nunca serían expuestos hacia fuera de la red, pero igualmente se evidenciaron los siguientes defectos:

- Regulación o administración de los puertos en uso y por usar.
- Posibles cambios en la dirección IP de la máquina anfitriona.
- Múltiples puntos de acceso a nuestros servicios.
- Round Robin en DNS con la posibilidad de que el tráfico caiga en un nodo no disponible.

En vista de estos inconvenientes, se llevó a cabo una búsqueda para encontrar otra forma de exponer los servicios.

4.4.3.2. Utilización de Load Balancer

Como Kubernetes es una herramienta Cloud Native, esta espera ser desplegada en la nube pública (AWS, GCP, Azure, entre otros...) para que estos proveedores le proporcionen una pieza de infraestructura como un Load Balancer. El Load Balancer nos garantiza un único punto de entrada y la distribución de la carga entre los nodos, como se muestra en la Figura 14.

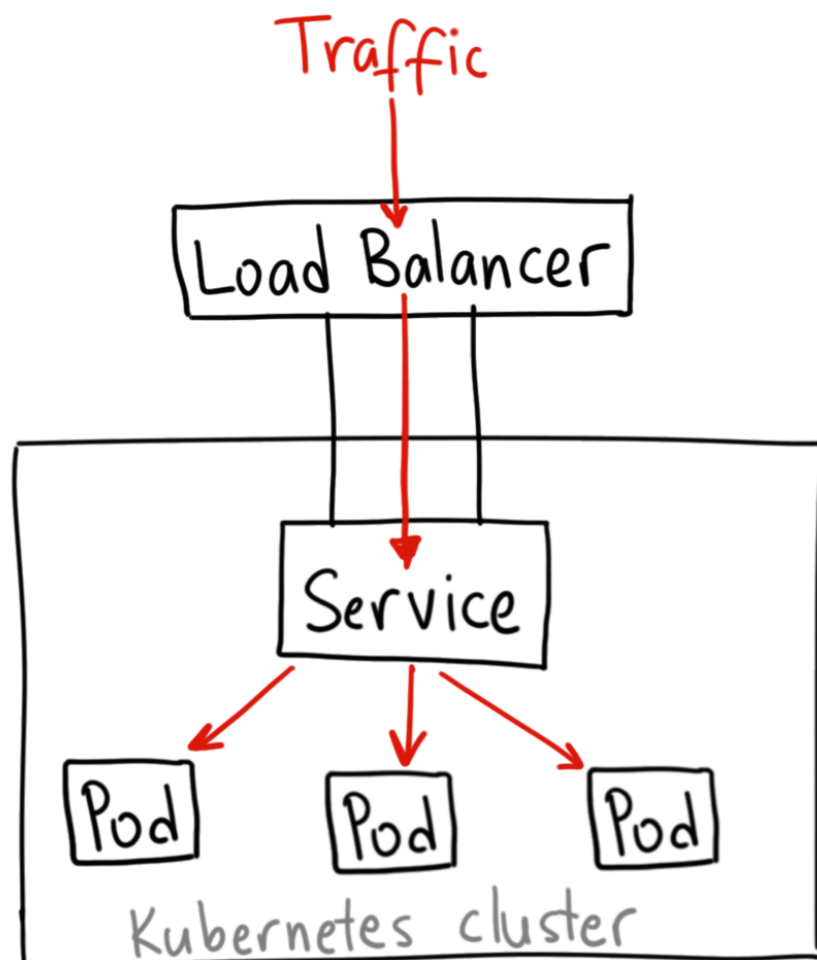


Figura 14. Representación del servicio LoadBalancer [23]

Teniendo en cuenta que la construcción del clúster se realiza en un entorno de nube privada y no se cuenta con la posibilidad de utilizar un Load Balancer físico u otro servicio externo, se comenzó la búsqueda de una herramienta que pueda suplir esta necesidad.

4.4.3.3. MetalLB

MetalLB se conecta a nuestro clúster Kubernetes y proporciona una implementación de Load Balancer de red. En definitiva, nos permite crear servicios Kubernetes tipo Load Balancer en el clúster.

MetalLB nos permite desempeñar dos funciones:

- Asignar una dirección IP.
- Anuncia la dirección a otros nodos.

La herramienta solicita un conjunto de direcciones IP que puede utilizar. MetalLB consiste en un controlador que se encargará de asignar y liberar direcciones individuales a medida que los servicios entren y salgan, y sólo distribuirá IPs que formen parte de este pool configurado. También se compone de los speakers o altavoces, que están presentes en cada nodo worker y llevan la palabra a los demás para avisar que la dirección "vive".

Una vez que MetalLB ha asignado una dirección IP externa a un servicio, debe hacer saber a la red más allá del clúster que la IP "vive" en el clúster. MetalLB utiliza protocolos de enrutamiento estándar para lograr este objetivo (ARP, NDP o BGP). Se implementó en el cluster el protocolo de nivel 2 que funciona con ARP, debido a que era el más sencillo de comprender.

Se debe tener en cuenta que no es posible hacer PING a la dirección IP que ha sido asignada por el Load Balancer.

Más allá de que MetalLB pueda parecer una muy buena solución al problema planteado, algunas problemáticas siguen en pie, tales como:

- No es un verdadero equilibrio de carga de los servicios.
- Cuello de botella de los flujos de la red.
- A igual cantidad de servicios de tipo Load Balancer, igual cantidad de direcciones IP asignadas.

4.4.3.4. Utilización de Ingress

A diferencia de todos los ejemplos anteriores, Ingress NO es un tipo de servicio. En cambio, se sitúa delante de varios servicios y actúa como un "router inteligente" o punto de entrada al clúster, como se ilustra en la Figura 15.

Se pueden realizar muchas cosas diferentes con un Ingress, y hay muchos tipos diferentes de controladores Ingress que tienen diferentes capacidades: Host pathing, Route pathing, healthcheck de nodos, entre otros.

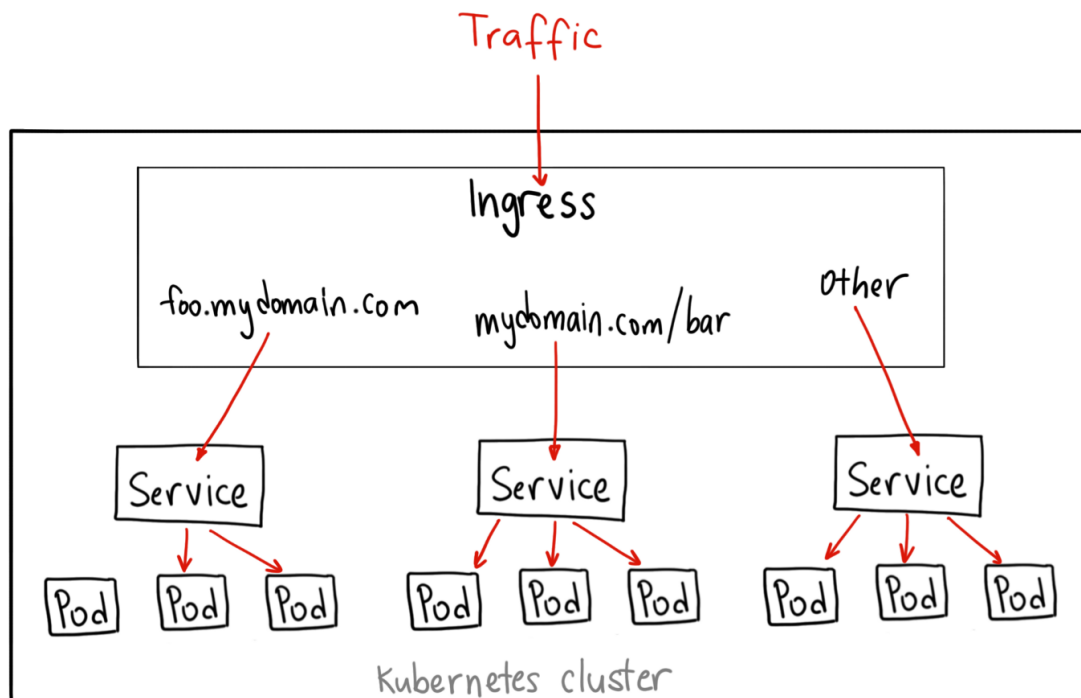


Figura 15. Representación del servicio Ingress [23]

4.4.3.5. LoadBalancer + Ingress Nginx

Después de haber realizado el estudio y entender las diferentes situaciones posibles, se encontró que la mejor manera de utilizar todos estos servicios y herramientas en la nube privada consiste en mezclar las características de MetalLB (para darnos un único punto de entrada a nuestro clúster) y un Ingress (con el poder de enrutar nuestras peticiones de manera inteligente), como se muestra en la Figura 16. Así se tendrá la posibilidad de distribuir bien la carga y centralizar la exposición de los servicios al exterior.

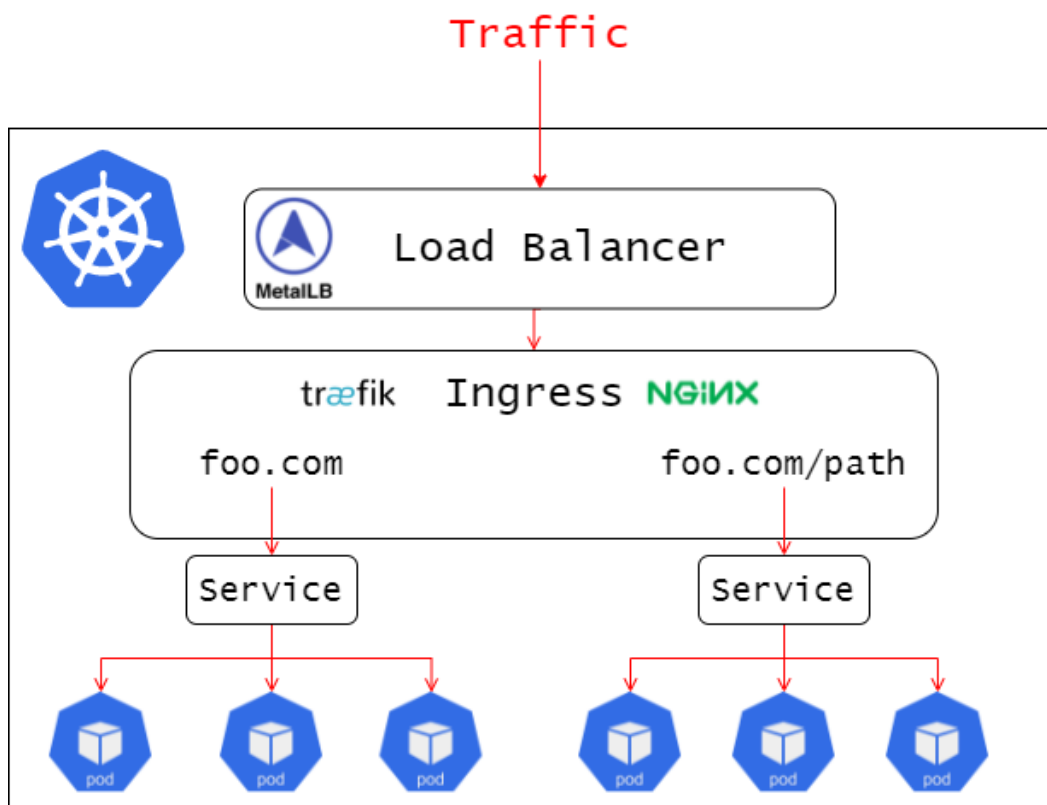


Figura 16. Arquitectura MetalLB + Ingress

Para finalizar esta parte, se decidió utilizar Ingress Nginx como servicio Ingress pues el equipo ya contaba con experiencia trabajando con Nginx, la herramienta tiene una gran documentación y se integró muy bien con el cluster.

4.4.4. Aislamiento y seguridad de contenedores

Por último, el aislamiento y la seguridad de los contenedores es uno de los objetivos para que los servicios ofrecidos sean Multitenant, es decir, que exista la capacidad de hacer un aislamiento completo entre los espacios de trabajo (namespaces) donde se encuentra cada stack y los servicios que los componen, teniendo así mejor seguridad y visibilidad del tráfico. Para lograr el objetivo descrito anteriormente, se hizo referencia nuevamente a la CNCF para encontrar un CNI que tuviera las características que se buscaba para el clúster.

4.4.4.1. CNI

Un CNI es responsable de insertar una interfaz de red en el namespace de red del contenedor y de realizar los cambios necesarios en el host. A continuación, asigna una dirección IP a la interfaz y configura las rutas de acuerdo con la sección de gestión de direcciones IP invocando el plugin IPAM (IP Address Management) [25].

4.4.4.2. Cilium

Cilium es un plugin de CNI para asegurar de forma transparente la conectividad de red entre los servicios de aplicación con el uso de eBPF, a continuación se presentan algunas ventajas al utilizar Cilium [26] :

- Implementación basada en la identidad de los recursos NetworkPolicy para aislar la conectividad de los pods en las capas L3 y L4.
- Una extensión de NetworkPolicy en forma de CRD (Custom Resource Definition) que amplía el control de la política para añadir.
- Aplicación de la política de capa 7 en la entrada y salida para los protocolos de aplicación HTTP y Kafka.
- Apoyo a la salida de los CIDR para asegurar el acceso a los servicios externos.
- Implementación de ClusterIP para proporcionar equilibrio de carga para el tráfico de pod a pod.

4.4.4.2.1. CiliumNetworkPolicy

Para la utilización de Cilium no es relevante utilizar las direcciones IP de los pods al definir las políticas de seguridad. En cambio, las etiquetas asignadas a los pods pueden utilizarse para definir estas políticas. Las políticas se aplicarán a los pods adecuados en función de las etiquetas, independientemente de dónde o cuándo se ejecuten en el clúster.

En el ejemplo siguiente podemos destacar:

- El valor de 'endpointSelector' que nos indica hacia dónde se dirigirá el tráfico.
- Los valores 'Ingress', 'fromEndpoints', 'toPorts' que nos indican desde qué pods etiquetados y hacia qué puerto podemos comunicarnos con el servicio anterior.

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: collector-to-database
  namespace: tig
spec:
  endpointSelector:
    matchLabels:
      app: database
  ingress:
    - fromEndpoints:
        - matchLabels:
            app: collector
      toPorts:
        - ports:
            - port: "8086"
```

Figura 17. Ejemplo de CNP L3 y L4

4.4.4.2.2. Hubble

Hubble aprovecha Cilium y eBPF para proporcionar una profunda visibilidad de la comunicación y el comportamiento de los servicios, las tramas de solicitud y los flujos de red, así como de la infraestructura de red de forma totalmente transparente. En la figura 18 se puede observar la utilización de Cilium dentro de los nodos y como estos pueden ser aprovechados por herramientas como Hubble para la visibilidad de flujo y el dúo de Grafana y Prometheus para las métricas de estos flujos.

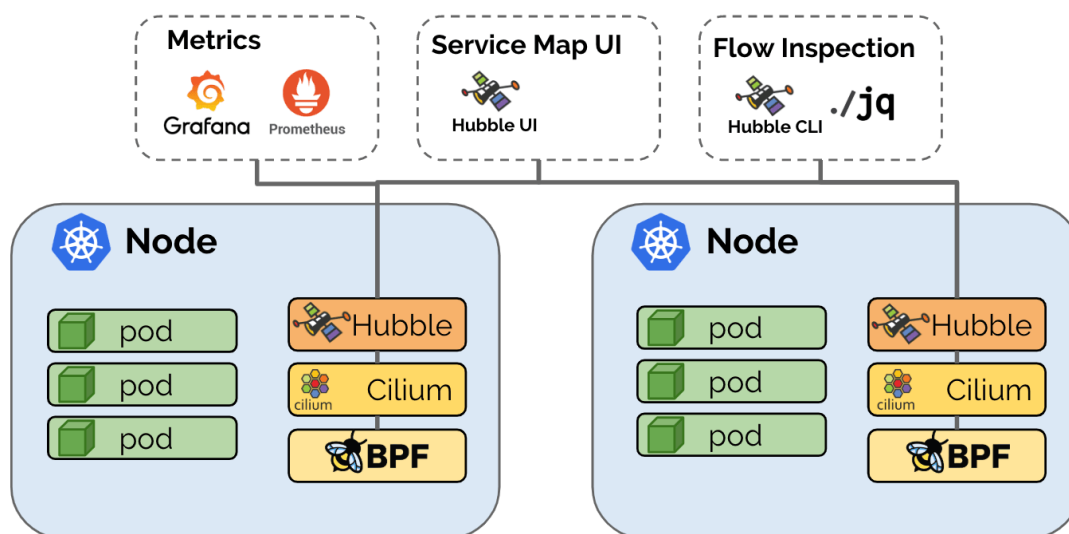


Figura 18. Arquitectura Cilium y Hubble [27]

Al utilizar Cilium y Hubble se realizaron políticas de CNP representadas en la Figura 19. En ella podemos ver que:

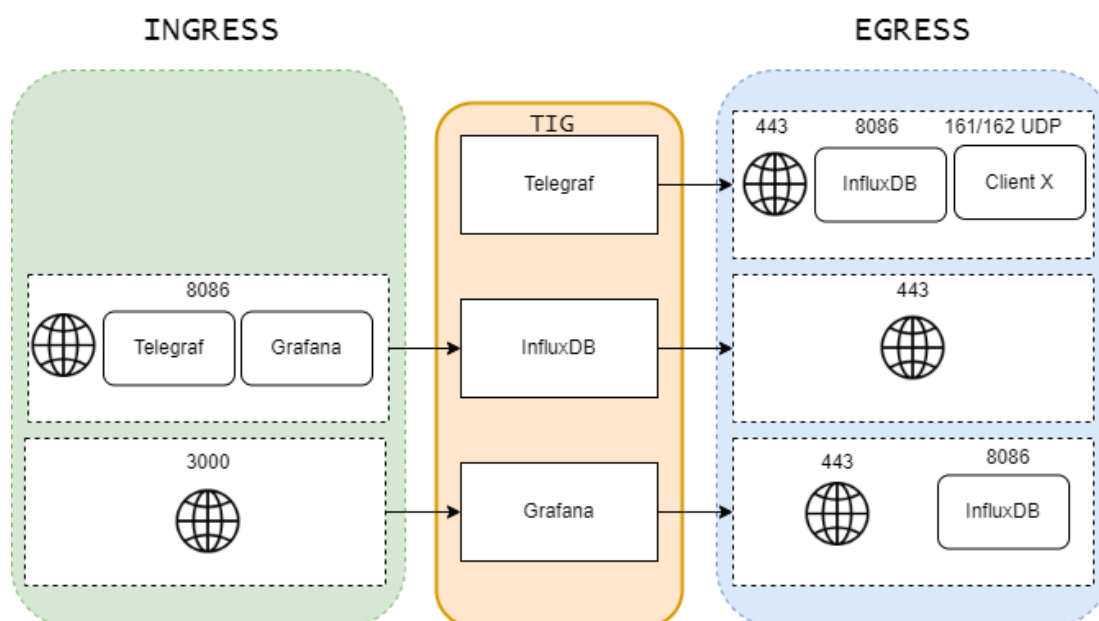


Figura 19. Políticas CNP configuradas

- El rectángulo del centro se refiere a los servicios del stack TIG.
- El rectángulo de la izquierda con fondo verde se refiere al acceso a los contenedores (Ingress). Cada rectángulo punteado se compone de una o más entidades/servicios. Estas entidades pueden acceder al servicio de su derecha.
- El rectángulo de la derecha con fondo azul evoca la salida de los contenedores (Egress). Cada rectángulo punteado se compone de una o más entidades/servicios. Estas entidades pueden ser consultadas por el servicio de su izquierda.

Para dar un ejemplo presentaré el caso del servicio InfluxDB:

- Para la política de Ingress están los servicios de Telegraf y Grafana, así como la entidad world que representa cualquier interacción fuera del clúster. Estas entidades pueden consultar el servicio InfluxDB usando el puerto 8086.
- Para la política de salida (Egress) sólo existe la entidad world en el puerto 443, esto significa que InfluxDB sólo puede consultar servicios/endpoints que utilicen el protocolo HTTPS.

Estas definiciones de políticas son más visibles una vez que se consulta la interfaz de usuario de Hubble y se verifica el flujo de tráfico y la interconexión en el servicio.

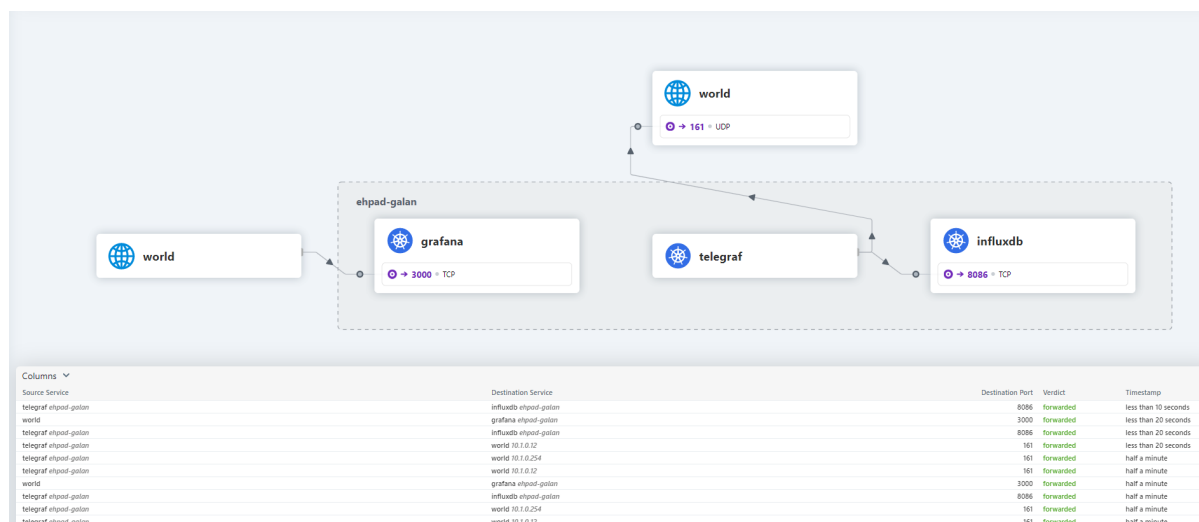


Figura 20. Esquema de interconexión y flujo de red del stack en Hubble UI

4.5. Backup

Como es normal en un entorno industrial, la preparación para los fallos y la pérdida de datos debe ser una de las principales preocupaciones de un proyecto. Debido al tamaño del proyecto y a los servicios ofrecidos, se decidió hacer copias de seguridad en la medida de lo posible. Con esto en mente, se hacen las siguientes copias de seguridad:

- **Rancher Backup:** Estas copias de seguridad registran la información de administración del servidor Rancher, es decir, la conexión con nuestro cluster y su capacidad de gestión.
- **Copia de seguridad del clúster:** Esta copia de seguridad contiene la información del clúster, específicamente la información que contiene los nodos ETCD (configuración del clúster, conexión a los nodos, despliegues, servicios, secretos, entre otros).
- **Longhorn Backup:** Estas copias de seguridad son específicas de la herramienta Longhorn, que registra información, métricas y datos en general de los servicios desplegados (cada entidad de los servicios Grafana e Influx).
- **Copia de seguridad de máquinas virtuales:** Por último, la plataforma de virtualización vSphere permite realizar copias de seguridad de la información de las máquinas virtuales, grabando realmente la información en el disco duro de las máquinas virtuales.

Para realizar las tres primeras copias de seguridad mencionadas se emplea la herramienta Minio, un servidor de almacenamiento estático de archivos/objetos [20] desplegado en una máquina virtual dentro de nuestra infraestructura pero fuera del clúster. Un vistazo de la consola de administración y los backups guardados se encuentra en la figura 21.

La última copia de seguridad listada se guarda en formato XFS en el servidor ubicado en la agencia central, guardando la información de todas las máquinas virtuales pertenecientes al clúster y al servidor Minio.

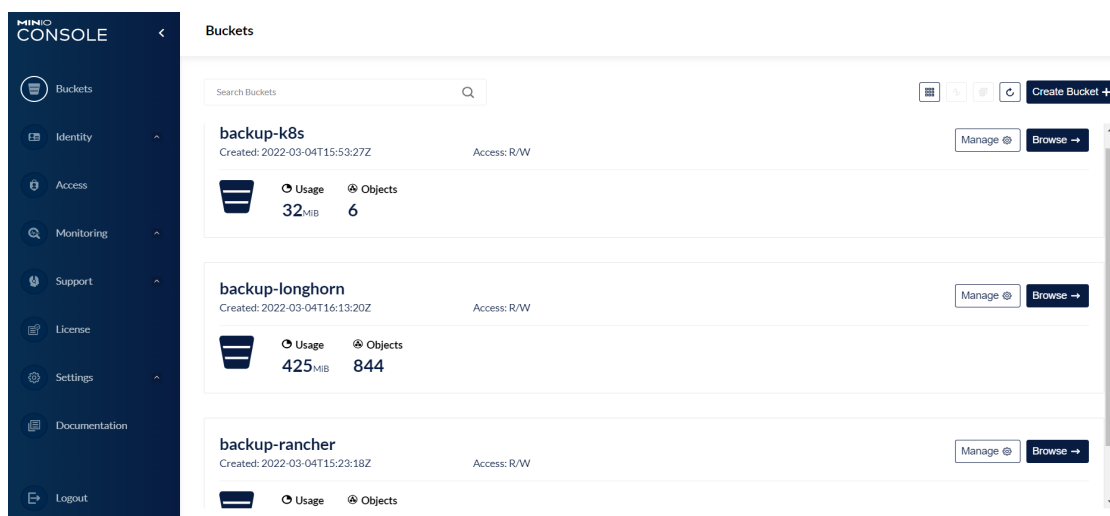


Figura 21. Consola de administración Minio

4.6. Tests

Las pruebas eran tan importantes para mí como para Axians Toulouse. Estas brindan seguridad en la evaluación del proyecto y sus objetivos, y demostrarían de una vez por todas la resiliencia, la tolerancia a los fallos y la funcionalidad de la infraestructura y el clúster en momentos de error, añadiendo el valor de los estudios realizados para la elección de herramientas que eviten una pérdida total en estas situaciones.

Axians Toulouse tiene la costumbre de elaborar un documento técnico centrado en la lista, las pruebas y la evaluación de las pruebas, de manera que este documento representa una huella del alcance del proyecto, de lo que el clúster es capaz o no de tolerar y una mirada global al funcionamiento de la solución. Este documento se llama CRT (Cahier recette des tests).

Por su propia naturaleza, este documento reúne toda la información necesaria para una buena comprensión de las pruebas realizadas. Para cada elemento, describe

- Los objetivos de la prueba.
- Las condiciones generales.
- Las acciones/eventos desencadenantes.
- Comportamiento esperado.
- Resultados de las pruebas.

Para la estructura de este documento, se hizo una separación de las pruebas en tres capas.

- **Capa de prueba de la infraestructura física:** Estas pruebas tienen como objetivo comprobar los fallos de hardware, realizando pérdidas de firewalls, switches, servidores, servicios de Internet o VPNs que conectan la agencia con el datacenter.
- **Capa de Prueba de la Infraestructura Lógica:** Estas pruebas corresponden a pérdidas que pueden ocurrir a nivel de virtualización y despliegue, es decir, pérdida de la máquina virtual de Rancher, máquinas virtuales con rol de trabajador, control plane o ETCD, eliminación de servicios/pods, entre otros.
- **Capa de pruebas de la infraestructura auxiliar:** Por último, las pruebas de la infraestructura auxiliar se refieren a las pruebas de las máquinas virtuales donde se alojan las copias de seguridad del clúster y su información, así como las copias de seguridad de las mismas máquinas virtuales. Además, se han realizado pruebas de todo el ciclo de recuperación del clúster, comprobando su copia de seguridad y su capacidad de restauración en el funcionamiento del clúster.

5. DESPLIEGUE DEL STACK

El despliegue del stack no tuvo mayores complicaciones y puede parecer sencillo después de los retos que se enfrentaron durante el proyecto. La elección de la versión de cada servicio en el stack se hizo tomando la última versión estable del repositorio de contenedores Dockerhub, donde se comprobó que era compatible y que funcionaba bien. A continuación se muestra el resultado del despliegue del stack, así como algunas mejoras que se han realizado en el despliegue de los servicios para garantizar que funcionen lo mejor posible y que el proceso sea realmente coherente con una solución multi-tenant.

5.1. Stack

5.1.1. Configuración de Telegraf

Una vez realizada la interconexión con la red de clientes, lo que sigue es la realización de la configuración de Telegraf (ver Figura 22), uniendo en una lista de direcciones IP todos los equipos de red que serán monitorizados, utilizando los protocolos ICMP y SNMP. Estas configuraciones fueron creadas con base en configuraciones previas realizadas por el tutor de la práctica. Como ejemplo se presentan algunas métricas recuperadas:

- **ICMP:** average_response_ms, packets_received, percent_packet_loss, entre otros.
- **SNMP:** hostname, cpu_use, ram_use, temperature, uptime, ifDescr, ifAdminStatus, ifOperStatus, ifInErrors, ifInErrors, ifType, entre otros.

```
#####
#                               PING CONFIG                               #
#####

[[inputs.ping]]
  urls = [
  ]
  count = 4

#####
#                               FIREWALLS CONFIG                           #
#####

[[inputs.snmp]]
  community = "$SNMP_COMMUNITY"
  version = 2

  agents = [
    "udp://xxx.xxx.xx.xx:161",
  ]

[[inputs.snmp.field]]
  oid = "RFC1213-MIB::sysUpTime.0"
  name = "uptime"

[[inputs.snmp.field]]
  oid = "RFC1213-MIB::sysName.0"
  name = "hostname"
  is_tag = true

[[inputs.snmp.table]]
  oid = "IF-MIB::ifTable"
  name = "interface"
  inherit_tags = ["hostname"]

[[inputs.snmp.table.field]]
  oid = "IF-MIB::ifDescr"
  name = "ifDescr"
  is_tag = true
```

Figura 22. Ejemplo de archivos de configuración Telegraf

5.1.2. Guardado y recuperación de datos en InfluxDB.

Gracias a la interfaz gráfica disponible en la herramienta Influx desde su versión 2.0, ahora es posible configurar las consultas de las métricas y comprobar su resultado y cambios a lo largo del tiempo. Esto sirve para una mejor comprensión de lo que se mostrará posteriormente en los tableros de Grafana. Una mirada rápida a este tablero y algunas métricas está disponible en la figura 23.



Figura 23. Métricas en la plataforma de Influx

5.1.3. Construcción de tableros y visualización de datos en Grafana

Para la visualización de los datos se realizaron tableros que pudieran representar el estado actual de los equipos monitorizados y sus estados anteriores, de forma que sea posible ayudar a las personas del centro de servicio a detectar anomalías y fallos en la infraestructura del cliente (ver Figura 24).

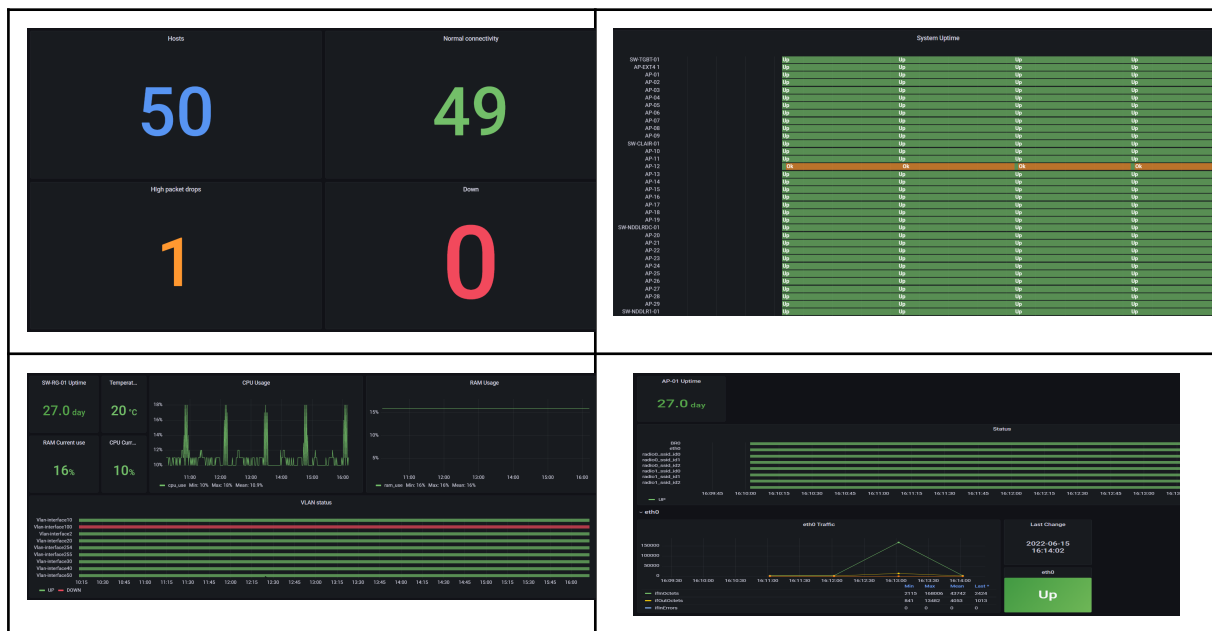


Figura 24. Tableros de métricas en Grafana

5.2. Resiliencia aplicativa

Un clúster Kubernetes es resiliente por diseño. Sin embargo, esto no significa que los servicios no se verán afectados si algo le sucede al clúster, como un fallo o una acción de drenaje en un nodo o un fallo de la aplicación en el pod.

Normalmente, un pod se asigna en un nodo en función de varios factores como:

- Disponibilidad de recursos.
- Disponibilidad del nodo.
- Tolerancia.
- Reglas de afinidad y anti-afinidad.

Si tenemos tres clones de nuestra aplicación, Kubernetes no los dispersará por los nodos trabajadores. La forma en que Kubernetes dispersará los pods depende de los factores mencionados al principio. Sin embargo, se realizó un estudio para encontrar las mejores decisiones operativas de tal forma que se pueda tener servicios y aplicaciones resilientes, es decir, aplicaciones de alta disponibilidad.

5.2.1. Pod affinity y anti-affinity

Pod affinity y anti-affinity [24] se utilizan para alojar, si es posible, los servicios en diferentes nodos. Funciona de dos maneras diferentes, una comprobación "dura" o una comprobación "suave" (`requiredDuringSchedulingIgnoredDuringExecution` y `preferredDuringSchedulingIgnoredDuringExecution`). Se puede utilizar un `hard check` cuando, por ejemplo, se requiera un redis (base de datos de valor clave) en cada nodo para que la aplicación pueda conectarse a ella más rápidamente.

5.2.2. Requests and Limits for containers

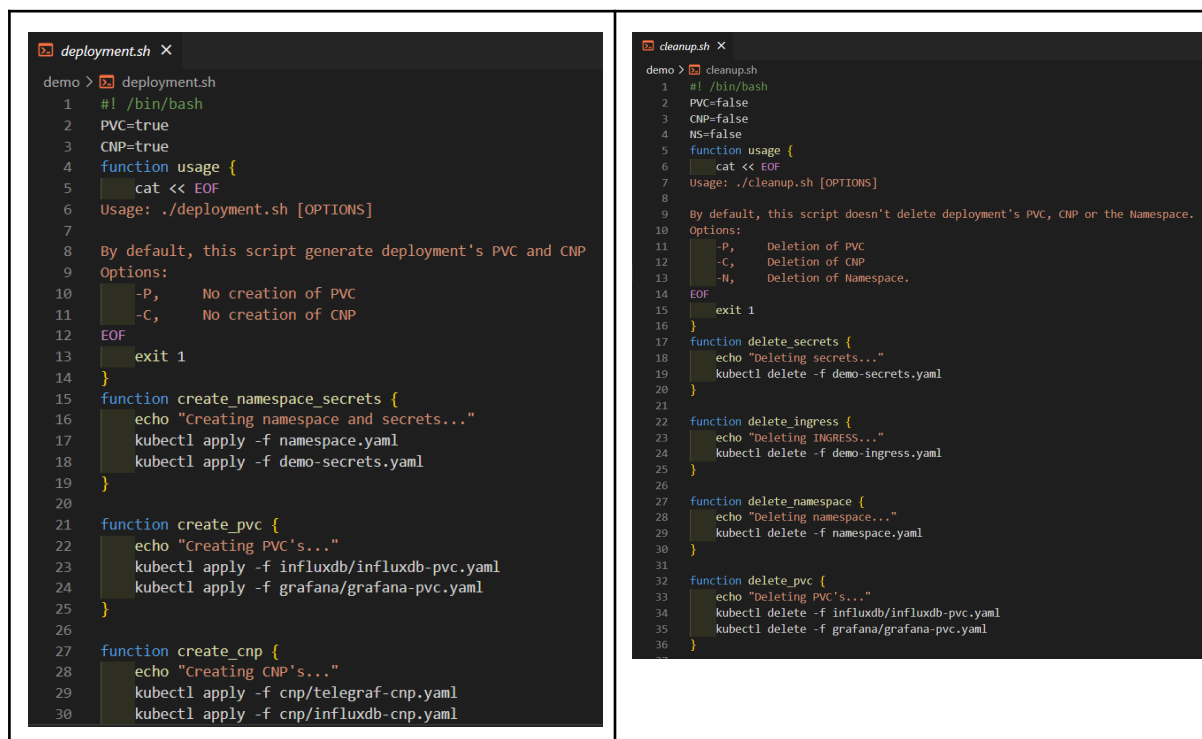
Se puede definir qué recursos (normalmente CPU y memoria) están garantizados (Requests) y cuáles serán provisionados (Limits). Esto se hace para que ningún nodo en mal estado consuma más recursos de los necesarios. Estas decisiones no son obligatorias y no garantizan que los servicios estén siempre disponibles, pero sirven para construir servicios resilientes a una serie de fallos que puede tener el clúster.

5.3. Industrialización del proceso de despliegue

En el marco de los objetivos finales se buscó la industrialización y automatización del proceso de despliegue para garantizar un resultado concreto en cada momento. Durante este último paso también se produjo la primera versión de lo que sería el proceso por que se ejecutaría a la hora de desplegar o actualizar los servicios ya mencionados, con el fin de evitar en lo posible fallos e indisponibilidades.

Para la automatización del proceso, se llevaron a cabo scripts bash que permitían el despliegue rápido y la eliminación de servicios en el clúster, ofreciendo más opciones para la creación/eliminación de volúmenes de datos, políticas de acceso, firewalling o secretos. Otro de los objetivos de este proceso era evitar en lo posible la realización de comandos en el clúster y reducir el factor de riesgo humano que esto puede provocar.

Una mirada rápida a estos scripts, las funciones definidas y la clase de comandos que ejecutan se encuentra en la siguiente figura.



```
demo deployment.sh X
demo > deployment.sh
1  #!/bin/bash
2  PVC=true
3  CNP=true
4  function usage {
5  cat << EOF
6  Usage: ./deployment.sh [OPTIONS]
7
8  By default, this script generate deployment's PVC and CNP
9  Options:
10 -P, No creation of PVC
11 -C, No creation of CNP
12 EOF
13 exit 1
14 }
15 function create_namespace_secrets {
16 echo "Creating namespace and secrets..."
17 kubectl apply -f namespace.yaml
18 kubectl apply -f demo-secrets.yaml
19 }
20
21 function create_pvc {
22 echo "Creating PVC's..."
23 kubectl apply -f influxdb/influxdb-pvc.yaml
24 kubectl apply -f grafana/grafana-pvc.yaml
25 }
26
27 function create_cnp {
28 echo "Creating CNP's..."
29 kubectl apply -f cnp/telegraf-cnp.yaml
30 kubectl apply -f cnp/influxdb-cnp.yaml
}

demo cleanup.sh X
demo > cleanup.sh
1  #!/bin/bash
2  PVC=false
3  CNP=false
4  NS=false
5  function usage {
6  cat << EOF
7  Usage: ./cleanup.sh [OPTIONS]
8
9  By default, this script doesn't delete deployment's PVC, CNP or the Namespace.
10 Options:
11 -P, Deletion of PVC
12 -C, Deletion of CNP
13 -N, Deletion of Namespace.
14 EOF
15 exit 1
16 }
17 function delete_secrets {
18 echo "Deleting secrets..."
19 kubectl delete -f demo-secrets.yaml
20 }
21
22 function delete_ingress {
23 echo "Deleting INGRESS..."
24 kubectl delete -f demo-ingress.yaml
25 }
26
27 function delete_namespace {
28 echo "Deleting namespace..."
29 kubectl delete -f namespace.yaml
30 }
31
32 function delete_pvc {
33 echo "Deleting PVC's..."
34 kubectl delete -f influxdb/influxdb-pvc.yaml
35 kubectl delete -f grafana/grafana-pvc.yaml
36 }
```

Figura 25. Scripts de despliegue y eliminación

6. CONCLUSIONES

Una vez realizadas las pruebas se pudo comprobar el funcionamiento, la resiliencia y la tolerancia a fallos de lo construido durante las prácticas, dando resultados satisfactorios ya que el cluster pasó las pruebas a la infraestructura física y lógica de forma gratificante. De manera que el clúster tiene la capacidad de soportar la pérdida de uno de sus servidores, la interconexión con la agencia y la pérdida de los elementos de infraestructura de red que lo acompañan.

También es importante señalar que si se produce un fallo en el stack desplegado, el clúster tiene la capacidad de volver a ponerlo a disposición en no más de 2 minutos y 30 segundos, lo que también satisface las necesidades de la empresa en términos de SLA con futuros clientes.

Con la realización de este proyecto se pudo dotar a la empresa de una nueva forma de desplegar sus herramientas de monitorización de forma industrializada y resiliente ante a fallos, además de ser el primer paso para el proceso de migración de servicios al clúster construido, lo que me llena de satisfacción por los objetivos alcanzados durante las prácticas.

Los objetivos planteados de este proyecto fueron cumplidos en su totalidad, y permitieron el desarrollo de nuevas ideas y funcionalidades para un trabajo futuro.

El estudio, diseño, implementación y prueba de una solución para la orquestación de contenedores fue una experiencia profesional muy gratificante para mí, de hecho, esta experiencia es el primer paso hacia el enfoque profesional en el que quiero dirigir mi carrera.

Aunque ya tenía contactos y experiencias con algunas de las herramientas puestas a disposición en el proyecto durante mi formación en Colombia y mi experiencia profesional, como Contenedores, Kubernetes y Grafana, las asignaciones y tareas realizadas en la configuración del clúster y el stack me permitieron conocer más específicamente cómo funcionan y cómo ponerlas a disposición de un caso particular de la industria, sin dejar de lado las muchas herramientas y conceptos que pude aprender de mis compañeros de proyecto y oficina.

Para terminar, me gustaría destacar que las experiencias personales que he tenido durante mi estancia en el INSA y en Axians Toulouse me han dado una perspectiva diferente de cómo solía ver las cosas más cotidianas como estudiar, trabajar e incluso comunicarme con otras personas. Desde que llegué a Axians Toulouse, puedo ver mi mejora con el idioma francés y su comprensión, ya que he podido comunicarme constantemente con mis compañeros sobre diferentes temas fuera del trabajo, lo que ha sido una agradable sorpresa para mí y lo que esperaba del entorno laboral francés.

REFERENCIAS

1. <<En un mundo digital, el monitoreo de infraestructura ha tomado un papel protagónico que debes conocer si pretendes aprovecharla al máximo>> [Online]. Available: <https://www.e-dea.co/blog/que-debe-tener-una-solucion-de-monitoreo-de-infraestructura-tecnologica>
2. <<Telegraf>> [Online]. Available: <https://www.influxdata.com/time-series-platform/telegraf/>
3. <<InfluxDB>> [Online]. Available: <https://docs.influxdata.com/>
4. <<Grafana>> [Online]. Available: <https://grafana.com/oss/grafana/>
5. <<Docker-Compose>> [Online]. Available: <https://docs.docker.com/compose/>
6. <<Agile Scrum Methodology>> [Online]. Available: <https://medium.com/@vishal.marakana/agile-scrum-methodology-c1dbd7425dcf>
7. <<An empirical analysis of the relationship between project planning and project success>> Dov Dvir, Tzvi Raz, Aaron J. Shenhar, 2003. [Online]. Available: <https://www-sciencedirect-com.gorgone.univ-toulouse.fr/science/article/pii/S0263786302000121>
8. <<Jira>> [Online]. Available: <https://www.atlassian.com/fr/software/jira/comparison/jira-vs-azure-devops>
9. <<Towards optimal quality requirement documentation in agile software development: A multiple case study>> Woubshet Behutiye, Pilar Rodríguez, Markku Oivo, Sanja Aaramaa, Jari Partanen, Antonin Abhervé, 2022. [Online]. Available: <https://www-sciencedirect-com.gorgone.univ-toulouse.fr/science/article/pii/S0164121221002090>
10. <<Document Classification for Software Quality Systems>> Kari Laitinen, 1992. Available: <https://dl.acm.org/doi/pdf/10.1145/141874.141882>
11. <<Confluence >> [Online]. Available: <https://www.atlassian.com/es/software/confluence>
12. <<¿Qué es Kubernetes?>> Redhat [Online]. Available: <https://www.redhat.com/es/topics/containers/what-is-kubernetes>
13. <<Kubernetes>> [Online]. Available: <https://kubernetes.io/docs/home/>
14. <<etcd>> [Online]. Available: <https://etcd.io/docs/v3.5/op-guide/hardware/>
15. <<The Raft Consensus Algorithm>> [Online]. Available: <https://raft.github.io/>

16. <<Rancher>> [Online]. Available: <https://rancher.com/docs/rancher/v2.6/en/>
17. <<Rancher Architecture>> [Online]. Available: <https://rancher.com/docs/rancher/v2.6/en/overview/architecture/>
18. <<Open Source ¿Qué es y cuál es su importancia?>> [Online]. Available: <https://skillnet.co/open-source-que-es-y-cual-es-su-importancia/>
19. <<CNCF - Cloud Native Computing Foundation>> [Online]. Available: <https://www.cncf.io/>
20. <<Minio>> [Online]. Available: <https://min.io/>
21. <<Volumes as a Microservice: Two years later - A look at Project Longhorn with Kubernetes>> Shen Yang, Rancher Labs, 2019 [Online]. Available: <https://www.snia.org/educational-library/volumes-microservice-two-years-later-look-project-longhorn-kubernetes-2019>
22. <<Tip: Set Longhorn To Only Use Storage On A Specific Set Of Nodes>> Phan Le, 2021 [Online]. Available: <https://longhorn.io/kb/tip-only-use-storage-on-a-set-of-nodes/>
23. <<Kubernetes NodePort vs LoadBalancer vs Ingress? When should I use what?>> Sandeep Dinesh, 2018 [Online]. Available: <https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0>
24. <<Assigning Pods to Nodes>> Kubernetes [Online]. Available: <https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/>
25. <<A brief overview of the Container Network Interface (CNI) in Kubernetes>> [Online]. Kedar Vijay, Red Hat. Available: <https://www.redhat.com/sysadmin/cni-kubernetes>
26. <<Introduction to Cilium & Hubble>> Cilium [Online]. Available: <https://docs.cilium.io/en/v1.9/intro/>
27. <<Announcing Hubble - Network, Service & Security Observability for Kubernetes>> Cilium [Online]. Available: <https://cilium.io/blog/2019/11/19/announcing-hubble>

