Implementación de una Infraestructura Orquestada, Escalable y Altamente Disponible para Servicios Web

Duvan Ferney Vargas Martinez

Trabajo de Grado para optar al título de Ingeniero de Sistemas

Director

Emilio Justiniano Carcamo Troconis

MSc. Ingeniería de sistemas e informática

Tutor

Pablo Josue Rojas Yepes

MSc. Ingeniería de sistemas e informática

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Ingeniería de Sistemas

Bucaramanga

2024

Dedicatoria

Primero que todo, dedico esto a mi familia ya que nada de esto hubiera sido posible sin ustedes, de alguna u otra forma realmente esto es de ustedes.

Especialmente a mi equipo de Dev/Ops, mis maestros, compañeros y todas aquellas personas que fueron parte de este proyecto, ya que con esto contribuímos al mejoramiento de la calidad de los servicios informáticos de la Universidad Industrial de Santander.

Finalmente, a toda aquella persona, compañía, asociación que quiera llevar a cabo la implementación de Kubernetes, apoyando la transición de máquinas físicas o virtuales a contenedores, encaminando sus necesidades al futuro, utilizando las ultimas tecnologías en cuanto a la infraestructura orquestada; y requiera de información y una guía clara y confiable para decantarse por alguna herramienta para realizar este trabajo.

Agradecimientos

Quiero expresar mi agradecimiento de manera más profunda, reconociendo el impacto significativo que cada persona y experiencia ha tenido en mi trayecto académico y personal.

A mi familia, quiero agradecerles no solo por su apoyo constante, sino por ser el pilar que sustentó cada paso en este proceso. Han sido mi fuente de fortaleza y motivación, y por ello estoy eternamente agradecido.

A mis mentores, su orientación experta ha sido más que indispensable en este arduo camino. Sus conocimientos compartidos no solo enriquecieron este trabajo, sino que también dejaron una huella duradera en mi desarrollo académico.

A mis compañeros de RSI, agradezco sinceramente por la oportunidad que me brindaron, por los conocimientos que adquirí con el transcurso del tiempo, por cada ayuda y cada hora de instrucción y aprendizaje.

Finalmente a mis amigos, las experiencias compartidas han contribuido en gran medida a mi crecimiento personal, también a Benito Antonio Martinez Ocasio, Emmanuel Gazmey Santiago y Salomon Villada Hoyos, figuras las cuales me han acompañado en los mejores y peores momentos de mi vida.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Glosario

Autoescalado: Capacidad de ajustar automáticamente el número de recursos asignados a una aplicación o servicio en función de la carga de trabajo.

AWS CLI: Interfaz de línea de comandos para interactuar con servicios de Amazon Web Services.

Autenticación centralizada: Proceso de gestionar y autenticar usuarios en un solo lugar central.

Bare-metal: Implementación de sistemas o recursos directamente en hardware físico en lugar de utilizar máquinas virtuales.

Brazohf: Arquitectura de procesador utilizada en sistemas informáticos (ARM).

Brazo64/aarch64: Arquitectura de procesador utilizada en sistemas informáticos (ARM de 64 bits).

CloudWatch: Servicio de monitoreo y observabilidad de Amazon Web Services.

Containerd: Sistema diseñado para administrar contenedores en un sistema operativo.

CaaS (Contenedores como servicio): Modelo de implementación de contenedores que proporciona una plataforma para la implementación y administración de contenedores.

Datos etcd: Almacén de datos clave-valor distribuido utilizado para la configuración de Kubernetes.

Dirección IP del Maestro: Dirección IP del nodo maestro en un clúster de Kubernetes.

Docker: Sistema diseñado para empaquetar, distribuir y ejecutar aplicaciones en contenedores.

Dockerfile: Documento de configuración que alberga las directrices para la creación de una imagen.

Entorno de prueba: Entorno diseñado para realizar pruebas y experimentos sin afectar el entorno de producción.

Entorno local: Implementación de sistemas o recursos en la infraestructura local de una organización en lugar de en la nube.

Escalabilidad: Capacidad de un sistema para manejar un aumento en la cantidad de trabajo o recursos sin degradación del rendimiento.

Hash del Certificado CA: Resumen criptográfico del certificado de autoridad (CA) utilizado para firmar otros certificados.

HELM: Herramienta para administrar paquetes de aplicaciones preconfiguradas para Kubernetes.

IaaS (Infraestructura como servicio): Modelo de implementación de servicios en la nube que proporciona infraestructura virtualizada.

IAM: Servicio de Amazon Web Services para la gestión de identidades y accesos.

Imagen: Ejecutable que incluye todo lo necesario para ejecutar una aplicación, incluyendo el código, las bibliotecas y las dependencias.

Instancias: Versiones específicas de contenedores o máquinas virtuales en ejecución.

IOPS: Operaciones de entrada/salida por segundo, una métrica que mide el rendimiento del almacenamiento.

Kubelet: Componente que ejecuta en cada nodo y garantiza que los contenedores estén en un estado de ejecución en Kubernetes.

Kubeadm: Herramienta oficial de Kubernetes para iniciar y gestionar clústeres.

Kubectl: Herramienta utilizada para interactuar con clústeres Kubernetes.

LaaS (Licencia como servicio): Modelo de licenciamiento en la nube que proporciona acceso a software basado en suscripción.

Modo Autopilot google: Modo de funcionamiento en GKE (Google Kubernetes Engine) para administrar clústeres de Kubernetes.

Modo standard google: Modo de funcionamiento en GKE (Google Kubernetes Engine) para administrar clústeres de Kubernetes.

NGINX: Servidor web y servidor proxy inverso utilizado para balanceo de carga y manejo de tráfico.

Nube pública: Servicios y recursos de infraestructura proporcionados por proveedores de servicios en la nube.

Núcleo CPU: Componente central de un procesador que realiza operaciones aritméticas y lógicas.

On-premise: Implementación de sistemas o recursos en las instalaciones físicas de una organización en lugar de en la nube.

PaaS (Plataforma como servicio): Prototipo de implementación de servicios en la nube que proporciona una plataforma para ejecutar aplicaciones.

Playbooks: Secuencias de comandos automatizadas utilizadas en Ansible para describir configuraciones y orquestar tareas.

Product-uuid: Cadena de caracteres única asignada a un producto específico.

Puerto de API del Maestro: Puerto utilizado para acceder a la interfaz de programación de aplicaciones (API) del nodo maestro en Kubernetes.

Rollbacks: Revertir un sistema o aplicación a una versión anterior o a un estado anterior.

Rolling updates: Método de actualización de software que implementa cambios gradualmente en lugar de simultáneamente.

Racks de servidores: Estructuras físicas que albergan varios servidores en un centro de datos.

SaaS (Software como servicio): Modelo de implementación de servicios en la nube que proporciona acceso a software basado en suscripción.

Scaling Groups: Grupos de escalado en AWS para gestionar conjuntos de instancias.

SSH: Protocolo de red utilizado para acceder y gestionar sistemas remotos de forma segura.

Swap: Área de almacenamiento en disco utilizada por el sistema operativo cuando la RAM física está llena.

Token de unión: Token utilizado para que los nodos se unan al clúster de Kubernetes.

VCPU: Unidad de procesamiento virtual, representa una CPU física en un entorno virtualizado.

X86-64: Arquitectura de procesador utilizada en sistemas informáticos (64 bits, Intel/AMD).

Resumen

Título: Implementación de una Infraestructura Orquestada, Escalable y Altamente Disponible para Servicios Web [*]

Autores: Duvan Ferney Vargas Martinez [**]

Palabras Clave: Kubernetes, Contenedores, Infraestructura Orquestada, Servicios Web, Escalabilidad, Disponibilidad, Clúster.

Descripción: Este proyecto se centra en implementar una infraestructura orquestada, escalable y altamente disponible para servicios web, utilizando Kubernetes. Esto se logra analizando la importancia de esta tecnología, las diferentes herramientas que hay para utilizar Kubernetes, tanto en entornos locales como en la nube; realizando una comparativa de cada una de estas, buscando la mejor opción en términos de optimización de recursos; comprendiendo los beneficios y desventajas de cada una de estas. Finalmente con la herramienta seleccionada se lleva a cabo la implementación de una prueba piloto de un clúster de Kubernetes, tomando como caso de estudio algunos servicios web, tanto propios como de la Universidad Industrial de Santander en el marco del proyecto RSI. Esto con el objetivo de garantizar la escalabilidad y alta disponibilidad, estableciendo una base sólida para futuras investigaciones en este campo en constante evolución. Tras esta investigación, es pertinente afirmar que implementar Kubernetes para desarrollar una infraestructura orquestada en RSI, basándose en una arquitectura de servicios y microservicios, trae consigo multiples ventajas; entre ellas se incluyen la escalabilidad, alta disponibilidad, portabilidad y automatización del uso de contenedores con estos servicios web.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Emilio Justiniano Carcamo Troconis

Abstract

Title: Implementation of an Orchestrated, Scalable and Highly Available Infrastructure for Web Services [*]

Authors: Duvan Ferney Vargas Martinez [**]

Keywords: Kubernetes, Containers, Orchestrated Infrastructure, Web Services, Scalability, Availability, Cluster.

Description: This project focuses on implementing an orchestrated, scalable and highly available infrastructure for web services, using Kubernetes. This is achieved by analyzing the importance of this technology, the different tools available to use Kubernetes, both in local and cloud environments; making a comparison of each of these, looking for the best option in terms of resource optimization; understanding the benefits and disadvantages of each of these. Finally, with the selected tool, the implementation of a pilot test of a Kubernetes cluster is carried out, taking as a case study some web services, both our own and those of the Industrial University of Santander within the framework of the RSI project. This with the aim of guaranteeing scalability and high availability, establishing a solid foundation for future research in this constantly evolving field. After this research, it is pertinent to affirm that implementing Kubernetes to develop an orchestrated infrastructure in RSI, based on a services and microservices architecture, brings with it multiple advantages; These include scalability, high availability, portability, and automation of the use of containers with these web services.

---

[*] Bachelor Thesis

[**] Faculty of Physicomechanical Engineering. School of Systems and Computer Engineering. Director: Emilio Justiniano Carcamo Troconis

Introducción

En la última década, las diferentes herramientas para virtualización han evolucionado para empaquetar y distribuir aplicaciones, con todas sus dependencias de manera portátil y eficiente, permitiendo que estas se ejecuten de manera consistente en diferentes entornos, tanto locales como en servidores en la nube.

Este proyecto tiene como objetivo implementar una infraestructura orquestada, escalable y altamente disponible para servicios web, haciendo uso de Kubernetes on-premise, destacando su importancia y sus implicaciones tanto positivas como desafiantes. Conforme se utiliza esta tecnología en cualquier tipo de ámbito, en este caso un enfoque empresarial, es esencial comprender cómo estas tecnologías pueden beneficiar a las diferentes organizaciones, al tiempo que abordamos cuestiones críticas como su aplicabilidad y conveniencia respecto a otras alternativas comerciales como Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure u otras plataformas.

A lo largo de este proyecto, se exploran diferentes herramientas para dicha implementación, buscando la mejor opción en cuanto a la optimización de recursos y beneficios que nos brinda. Con ello, se busca contribuir al conocimiento existente sobre este tema y proporcionar una base sólida para futuras investigaciones y desarrollos en este campo en constante evolución.

En consecuencia, se presenta el siguiente problema.

1. Planteamiento y Justificación del Problema

En un entorno en el que el uso de diversos servicios informáticos, especialmente aquellos orientados a aplicaciones web, ha experimentado un notable incremento, la necesidad de asegurar la alta disponibilidad y escalabilidad de estos servicios es crucial. Sin embargo, en la actualidad, no todas las organizaciones cuentan con la flexibilidad necesaria para abordar eficazmente este desafío. En este contexto, surge la propuesta de implementar una infraestructura contenerizada de alta disponibilidad y escalabilidad utilizando Kubernetes on-premise. Esta propuesta cobra especial relevancia para emprendimientos que apuntan a un rápido crecimiento y necesitan una flexibilidad de cobertura de sus necesidades TI.

La pregunta fundamental que orienta este estudio es la siguiente: ¿Cuál sería la solución más eficiente para llevar a cabo la migración de servicios, considerando aspectos como la escalabilidad, la alta disponibilidad y la optimización de recursos, a partir de una comprensión detallada de las necesidades de cada organización?

La respuesta a esta interrogante se deriva de un análisis minucioso de las particularidades de cada organización. Asi mismo, se examina por qué la implementación de Kubernetes on-premise es preferible en lugar de optar simplemente por servicios de nube pública como Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure u otras plataformas similares. Además, se investiga para qué tipos de organizaciones resulta conveniente contratar dichos servi-

cios en la nube. Tomando como caso de estudio la Universidad Industrial de Santander en el marco del proyecto de Renovación de los Sistemas de Información (RSI), se lleva a cabo un análisis de viabilidad comparativa entre estas soluciones, evaluando aspectos como los recursos utilizados, la escalabilidad y la disponibilidad inherente a los servicios de contenerización proporcionados por diferentes compañías.

Con el propósito de abordar este dilema, este estudio tiene los siguientes objetivos.

2. Objetivos

2.1. Objetivo General

- Implementar una solución orquestada de servicios web que priorice la escalabilidad, la alta disponibilidad y la optimización de recursos en sistemas on-premise del proyecto RSI.

2.2. Objetivos Especificos

- Evaluar la viabilidad de diversas soluciones para la orquestación de servicios web en el contexto de RSI, contrastando herramientas comerciales como Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure y soluciones on-premise basadas en Kubernetes.
- Definir criterios y métricas para la selección de soluciones destinadas a la orquestación de servicios web, acotadas a las necesidades de RSI.
- Llevar a cabo una prueba piloto de implementación de la orquestación empleando los criterios y métricas establecidos para evaluar su viabilidad.

Para lograr estos objetivos, el alcance del proyecto abarca una serie de elementos esenciales que deben ser cuidadosamente planificados y ejecutados.

2.3. Alcance

El alcance del proyecto se limita a la implementación inicial del clúster de Kubernetes, como objeto de estudio, se utilizan algunos de los servicios web de la Universidad Industrial de Santander, por su parte, se establecen métricas tales como, rangos de nodos, recursos, desafíos actuales y objetivos de TI para esta organización. No se incluye la gestión continua de la totalidad de los servicios o aplicaciones específicas.

Se diseña e instala el clúster de Kubernetes, esto realizando un análisis detallado de las necesidades organizacionales, incluidas las necesidades de capacidad, escalabilidad y recursos; designando la arquitectura del clúster, especificando la distribución de nodos maestros y trabajadores.

Se integra este clúster con herramientas y recursos existentes de la organización, como los sistemas de monitoreo; a su vez, se despliegan aplicaciones de muestra en este, realizando pruebas de integración para garantizar su funcionamiento.

Es importante documentar las prácticas y los procedimientos operativos realizados detallados, que incluya información sobre la arquitectura, las configuraciones, los procedimientos y los detalles de implementación.

Para comprender completamente el contexto y la necesidad de este proyecto, es esencial analizar el estado del arte.

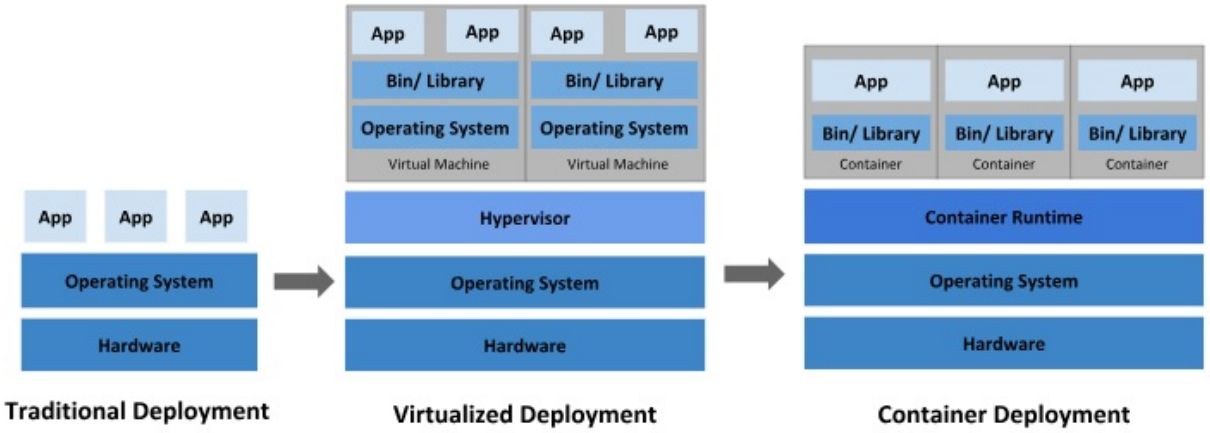
3. Estado del Arte

En el ámbito de kubernetes, al ser una tecnología DevOps, el cual “Es un marco en continua evolución que fomenta el desarrollo eficiente de aplicaciones en un corto periodo de tiempo, así como la rápida entrega de nuevas funciones de software o productos revisados para los usuarios” (NetApp, 2019); estos son muy utilizados dada su usabilidad y operación en la infraestructura TI.

En términos generales, Las organizaciones solían ejecutar aplicaciones en servidores físicos, enfrentando desafíos de asignación de recursos sin posibilidad de definir límites para las aplicaciones en un servidor, por lo cual esto se volvió costoso con el tiempo. La virtualización se introdujo como solución, permitiendo la ejecución de múltiples máquinas virtuales (VM) en la CPU de un solo servidor físico. Posteriormente, surgieron los contenedores, similares a las máquinas virtuales pero más livianos, ya que comparten propiedades del sistema operativo entre aplicaciones; al estar desacoplados de la infraestructura subyacente, son altamente portátiles a través de la nube y sistemas operativos. (Kubernetes Authors, Accedido el 3 de noviembre de 2023).

Figura 1.

Gráfico de la transición de los despliegues tradicionales hasta los contenedores.



Nota: Componentes principales de una implementación tradicional, una implementación virtualizada y una implementación de contenedores. Adaptado de: (Kubernetes Authors, Accedido el 3 de noviembre de 2023)

3.1. Antecendentes y evolución de la implementación de contenedores

A comienzos del siglo, la abstracción de máquinas virtuales (VMs), las cuales "son ordenadores de software que ofrecen la misma funcionalidad que las computadoras físicas; al igual que estas, ejecutan aplicaciones y un sistema operativo, no obstante, las máquinas virtuales son archivos informáticos que operan en una computadora física y se comportan como si fueran una computadora independiente"(«What is a Virtual Machine?», 2020), se establecen como una tecnología clave para la consolidación de servidores ya que se gestionan, mantienen fácilmente, y ofrecen alta disponibilidad; fueron principalmente utilizados ya que pueden albergar varios sistemas operativos en un solo hardware físico.

Posteriormente surge Docker y cambia la forma en la que se distribuyen y empaquetan las aplicaciones junto con sus dependencias, de una forma eficiente y portátil. Docker “es una plataforma de código abierto que posibilita a los desarrolladores la creación, implementación, ejecución, actualización y gestión de contenedores. Estos contenedores son elementos estandarizados y ejecutables que integran el código fuente de la aplicación con las bibliotecas y dependencias del sistema operativo (SO) requeridas para ejecutar dicho código en cualquier entorno” («¿Qué es Docker?», 2023).

A mediados de la decada de 2010, Kubernetes 1.0 se lanzó oficialmente, como una opción viable para la orquestación de contenedores. Empresas de todo el mundo comenzaron a adoptar Kubernetes para gestionar sus aplicaciones contenerizadas en entornos de producción. Kubernetes se unió a la Cloud Native Computing Foundation, “el propósito de esta entidad se concentra en fomentar y promover la informática nativa en la nube. Este moderno modelo computacional se impulsa mediante iniciativas de código abierto que son independientes de cualquier proveedor específico” («¿Qué es Cloud Native Computing Foundation (CNCF)?», 2022), lo que garantizó un desarrollo comunitario sólido y una amplia adopción en la industria de TI.

Sobre el año 2017, surgen proyectos como KubeVirt, el cual permitió la ejecución de máquinas virtuales (VMs) en clústeres de Kubernetes, lo cual brindó soporte a las organizaciones de gestionar tanto VMs como contenedores de manera unificada, lo que marcó un hito importante en la integración de estas tecnologías. Luego se desarrollaron herramientas y estrategias para facilitar la transición de aplicaciones de Docker a Kubernetes. Estas herramientas permitieron a las

organizaciones migrar sus aplicaciones, lo que contribuyó a la creciente adopción de Kubernetes.

3.2. Desarrollo actual

La comunidad de código abierto y las empresas líderes continúan invirtiendo en la evolución de Kubernetes para abordar las necesidades cambiantes de la infraestructura de TI moderna, estas investigaciones se centran en mejorar Kubernetes en áreas clave, como, la mejora del aislamiento de recursos y la observabilidad, siendo esta tecnología ampliamente valorada a la fecha.

Existen implementaciones on-premise, con diferentes herramientas para la gestión de estos clústeres, desde uno hasta n nodos; así como implementaciones en la nube. Estas implementaciones incluso brindan recomendaciones de qué otras herramientas y funcionalidades podemos agregar a un clúster de kubernetes, como la automatización, seguridad, escalabilidad, entre otros. Sin embargo, no existe un estudio detallado sobre qué herramienta es más conveniente utilizar de acuerdo con diferentes variables, tales como, los requisitos empresariales, recursos disponibles, desafíos actuales, objetivos de TI, incluso de los nodos necesarios para cada empresa. Así mismo, tampoco existe una guía clara de qué compone un clúster de kubernetes y para qué sirve cada herramienta; la información que se obtiene es conseguida tras una ardua investigación a detalle de diferentes autores y fuentes.

El estado del arte proporciona una visión de los desarollos en el campo de Kubernetes, destacando las soluciones existentes, sin embargo, para comprender el contexto de esta investigación, es esencial profundizar en el marco de referencia.

4. Marco de Referencia

En esta sección, se exponen los términos y el contexto necesarios para comprender a profundidad este proyecto, teniendo en cuenta el trasfondo histórico en el cual nos encontramos, es importante hacer uso de un sistema capaz de desplegar y organizar contenedores, siendo una solución viable para esto Kubernetes, siendo esta "una plataforma de código abierto diseñada para orquestar contenedores, con el propósito de automatizar la implementación, el escalado y la administración de aplicaciones contenidas" (Kubernetes, Accedido el 20 de septiembre de 2023-b). Kubernetes se ha convertido en una de las herramientas más conocidas para gestionar contenedores y ha experimentado una evolución significativa desde su creación; es pertinente destacar que Kubernetes no es un sistema PaaS tradicional, ya que no mantiene una estructura monolítica al proporcionar características que dependen de cada organización dependiendo de lo que esta requiera.

La evolución de Kubernetes se ha dado a través de múltiples versiones y mejoras desde su lanzamiento inicial en el año 2014 cuando fue lanzado por Google; desde entonces Kubernetes ha lanzado numerosas versiones que han hecho de esta una plataforma robusta con mejoras constantes. Esta plataforma se ha convertido en una herramienta muy importante para la orquestación de contenedores, utilizándose en múltiples aplicaciones y entornos. Es necesario tener en cuenta ciertos conceptos clave en el extenso mundo de Kubernetes, los cuales se exponen en este documento.

4.1. Componentes del clúster

4.1.1. Clúster

En el contexto de Kubernetes, un clúster “es un grupo de nodos máquina que ejecutan aplicaciones contenidas en contenedores.” (Red Hat, Accedido el 20 de septiembre de 2023). Este conjunto de máquinas incluye nodos físicos o virtuales que trabajan para gestionar y orquestar los contenedores de las aplicaciones. Para un funcionamiento optimo, el clúster debe componerse de varias herramientas como el planificador (scheduler), que es ün procedimiento en el plano de control que asigna Pods a Nodos; considerando las restricciones y recursos disponibles, este decide qué Nodos son destinos adecuados para cada Pod en la cola de programación"(Kubernetes, s.f.-a) ; el controlador de nodos (kubelet), el cual es ün agente que opera en cada nodo de trabajo y se responsabiliza de asegurar que los contenedores estén en ejecución en un recurso de pod específico"(Bootcamps, 2022a), además de monitorizar de que este siga ejecutandose, y finalemente el "namespace” el cual es un mecanismo que permite dividir un clúster en múltiples espacios virtuales o aislados lógicamente.

4.1.2. Contenedores

Los contenedores son una tecnología de virtualización ligera que ha revolucionado la forma en que se desarrollan, empaquetan y ejecutan aplicaciones. A diferencia de las máquinas virtuales tradicionales, los contenedores comparten el núcleo del sistema operativo subyacente y solo contienen las bibliotecas y los recursos necesarios para ejecutar una aplicación específica. Esto hace

que los contenedores sean más eficientes en cuanto a recursos y más portátiles.

La importancia de los contenedores en la orquestación radica en su capacidad para proporcionar un entorno de ejecución coherente y aislado para aplicaciones, las principales razones por las que los contenedores son esenciales en la orquestación son la portabilidad, eficiencia de recursos, aislamiento y finalmente la escalabilidad; para ejecutar estos contendeores es necesario hacer uso del “runtime", el cual es “el software encargado de llevar a cabo la ejecución de los contenedores. Kubernetes es compatible con múltiples opciones, como Docker, containerd, cri-o, rktlet, y cualquier implementación de la interfaz de tiempo de ejecución de contenedores de Kubernetes, conocida como Kubernetes CRI"(Kubernetes, C. R. I., s/f).

Es pertinente mencionar que Kubernetes y Docker son complementarios. Docker se utiliza para crear y empaquetar contenedores, mientras que Kubernetes se encarga de orquestar y administrar esos contenedores en un clúster.

4.1.3. Etcd

Etcd “es un depósito de datos consistente y altamente disponible que sirve como respaldo de Kubernetes para almacenar todos los datos del clúster"(Kubernetes, s.f.-c), este actúa como el cerebro detrás del clúster, manteniendo la sincronizcaión en todos los componentes de un clúster, garantizando la alta disponibilidad.

4.1.4. Pods

Un pod es la unidad más pequeña de implementación en Kubernetes. Manifiesta un entorno de ejecución para uno o más contenedores relacionados que comparten el mismo espacio de red y almacenamiento. Los contenedores dentro de un pod suelen estar altamente acoplados y son esenciales para la aplicación. Los pods son efímeros y pueden crearse, eliminarse o reemplazarse dinámicamente por Kubernetes en respuesta a la escalabilidad, las actualizaciones o las fallas.

4.1.5. Servicios

Un servicio es un recurso que expone una aplicación ejecutada en uno o más pods. Los servicios proporcionan una dirección IP y un puerto consistente que permiten que otras aplicaciones o servicios se comuniquen con la aplicación dentro de los pods, independientemente de su ubicación. Los servicios pueden ser de varios tipos, como ClusterIP (acceso interno al clúster), NodePort (acceso desde fuera del clúster) y LoadBalancer (uso de balanceadores de carga externos).

4.1.6. Volúmenes

Los volúmenes en Kubernetes son sistemas de almacenamiento que permiten a los pods persistir datos más allá de la vida útil de los contenedores. Los volúmenes se utilizan para compartir datos entre contenedores en el mismo pod o para acceder a datos desde fuera del pod. Los volúmenes pueden ser efímeros (temporales) o persistentes (almacenamiento duradero). Ejemplos de volúmenes persistentes son los de almacenamiento en bloque o sistemas de archivos en red

(NFS).

4.1.7. Ingress

El Ingress es un recurso en Kubernetes que se utiliza para gestionar las reglas de enrutamiento del tráfico HTTP y HTTPS desde fuera del clúster hacia los servicios dentro del clúster. Permite la configuración de rutas y reglas para dirigir las solicitudes a servicios específicos en función de encabezados, rutas o nombres de host. El Ingress se utiliza comúnmente para habilitar la exposición de servicios web a través de un controlador de Ingress compatible.

Estos conceptos son fundamentales para la construcción y la operación de aplicaciones en Kubernetes. Los pods son las unidades de ejecución de contenedores, los servicios facilitan la comunicación entre ellos, los volúmenes permiten el almacenamiento de datos, y el Ingress regula el tráfico hacia las aplicaciones expuestas. Combinados, estos conceptos proporcionan una infraestructura sólida para implementar aplicaciones escalables y altamente disponibles en un clúster de Kubernetes.

4.2. Arquitectura del clúster

Kubernetes es utilizado para administrar y automatizar la implementación, escalabilidad y operación de aplicaciones contenerizadas, por su parte, esta arquitectura se basa en una distribución de nodos maestros y nodos trabajadores que colaboran para ejecutar y gestionar las aplicaciones en contenedores; estos nodos son máquinas físicas y virtuales que son configurados para ser agregados a un clúster de Kubernetes.

4.2.1. Nodos maestros

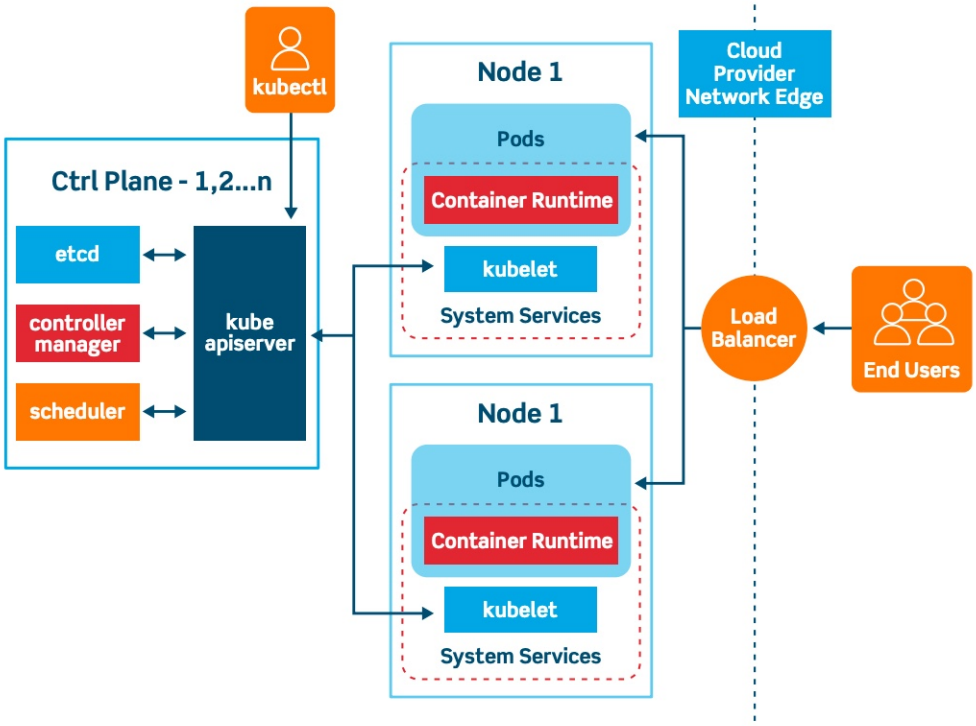
Los nodos maestros son el corazón del clúster de Kubernetes y son responsables de tomar decisiones de orquestación y gestionar el estado del clúster. Un clúster de Kubernetes típico consta de uno o más nodos maestros para garantizar la alta disponibilidad y la tolerancia a fallos.

4.2.2. Nodos trabajadores

Los nodos trabajadores son las máquinas donde se ejecutan las aplicaciones en contenedores. Están destinados a realizar el trabajo real y están gestionados por los nodos maestros. Es necesario mencionar los pods, los cuales son la unidad más pequeña de implementación en Kubernetes y pueden contener uno o varios contenedores.

Figura 2.

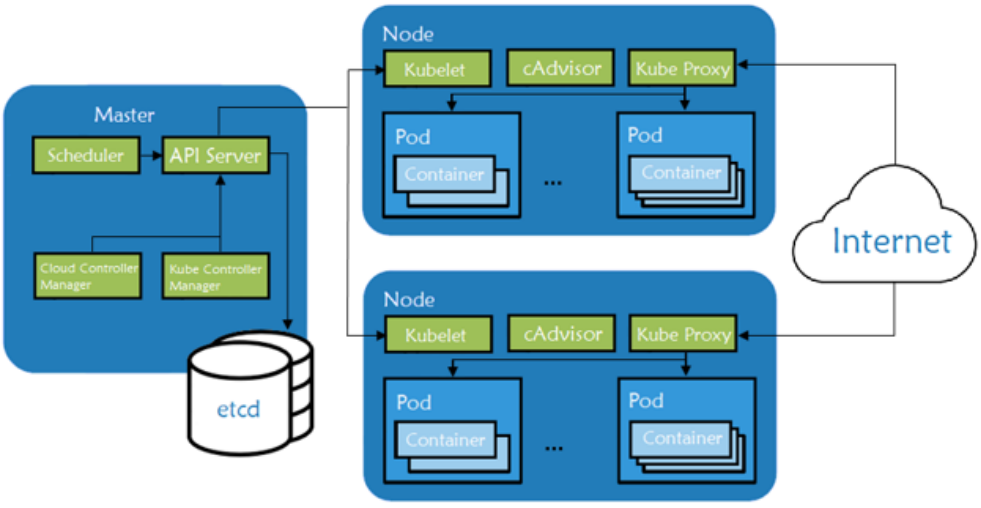
Gráfico de la arquitectura de Kubernetes.



Nota: Componentes principales de un clúster de Kubernetes. Adaptado de: («Arquitectura de Kubernetes - Taller de PAS», Accedido el 3 de noviembre de 2023)

Figura 3.

Gráfico de los nodos de un clúster de Kubernetes.



Nota: Componentes principales de un nodo. Adaptado de: («Arquitectura de Kubernetes - Taller de PAS», Accedido el 3 de noviembre de 2023)

4.3. Funcionamiento del clúster

Kubernetes gestiona y despliega contenedores mediante una serie de conceptos y componentes que permiten la orquestación eficiente de aplicaciones en un entorno de contenedores.

4.3.1. Gestión del clúster

Para empezar, se debe definir cómo se debe implementar una aplicación en Kubernetes mediante un archivo de configuración YAML, “en la creación de objetos como Pods, réplicas, servicios de implementación, entre otros, se utilizan archivos YAML” (StatDeveloper, Accedido el 20 de septiembre de 2023). Este archivo especifica detalles como la imagen del contenedor (“una imagen representa de manera estática la aplicación o servicio, incluyendo su configuración y

dependencias" (Microsoft, Accedido el 20 de septiembre de 2023)), los recursos necesarios y otros parámetros relacionados con la aplicación.

Una vez que se define la aplicación, se envía el archivo de configuración a Kubernetes a través de su API. Kubernetes interpreta esta definición y crea los recursos necesarios para la aplicación. Los recursos comunes incluyen pods, servicios, volúmenes, y otros.

Kubernetes utiliza su componente de planificación, el cual se encarga de vigilar los pods recién creados que no tengan ningún nodo asignado, para determinar en qué nodo del clúster se deben ejecutar los pods de la aplicación. El Scheduler tiene en cuenta factores como los recursos disponibles en los nodos, restricciones de afinidad y anti-afinidad, y políticas de programación definidas por el usuario, posteriormente Kubernetes crea los pods en los nodos seleccionados.

Kubernetes supervisa constantemente el estado de los pods y los recursos en el clúster. Si un pod falla o se elimina por alguna razón, Kubernetes puede tomar medidas para reemplazarlo automáticamente, como crear un nuevo pod en otro nodo.

Kubernetes proporciona servicios para exponer los pods a través de una dirección IP y un puerto consistente. Esto facilita la comunicación entre los componentes de la aplicación. Además, Kubernetes ofrece opciones para configurar el equilibrio de carga, lo que garantiza una distribución uniforme del tráfico entre los pods.

4.3.2. Despliegue del clúster

Kubernetes permite actualizaciones y despliegues controlados de la aplicación sin tiempo de inactividad. Los usuarios pueden cambiar la configuración de la aplicación o la imagen del contenedor y Kubernetes se encargará de actualizar gradualmente los pods para mantener la disponibilidad de la aplicación.

Finalmente, Kubernetes ofrece capacidades de monitorización y escalado automático. Los usuarios pueden configurar métricas y umbrales, y Kubernetes ajustará automáticamente el número de réplicas de la aplicación según la demanda para garantizar un rendimiento óptimo, así, "se utiliza en varios casos para asegurar que haya disponibilidad en un número determinado de Pods idénticos" (Kubernetes, Accedido el 20 de septiembre de 2023-d).

4.4. Orquestación de contenedores

4.4.1. Escalar aplicaciones automáticamente

Hay dos tipos de escalado automático, los cuales son esenciales para garantizar que las aplicaciones puedan mantener un rendimiento adecuado sin desperdiciar recursos.

Escalado Horizontal Automático: El HPA ajusta automáticamente el número de réplicas de un conjunto de pods para mantener un rendimiento específico basado en métricas ("cantidad de recursos consumidos por cada nodo o pod" (Kubernetes, Accedido el 20 de septiembre de 2023-c)). Esto significa que se ajustarán dinámicamente el número de réplicas de una aplicación para

satisfacer la demanda del tráfico y los recursos disponibles.

Escalado Vertical Automático: El VPA ajusta automáticamente los recursos (CPU y memoria) asignados a un pod en función de las métricas de rendimiento. Si un pod requiere más recursos para mantener su rendimiento, este los asignará, ayudando a optimizar el uso de recursos y mejorando la eficiencia.

Entonces, en términos generales, Kubernetes permite escalar aplicaciones automáticamente a través de su capacidad de escalado automático basado en métricas y políticas definidas por el usuario. esto se logra habilitando el escalado automático, por medio de un recurso llamado Horizontal Pod Autoscaler, el cual permite especificar los valores umbrales (uso de CPU y memoria) y el rango deseado de réplicas, teniendo en cuenta las métricas, si la métrica objetivo supera el umbral máximo, se aumenta el número de réplicas del pod, si cae por debajo del umbral mínimo, se reduce el número de réplicas para ahorrar recursos.

Es necesario monitorear continuamente el rendimiento y las métricas de la aplicación para asegurarse de que se mantenga dentro de los umbrales definidos, más adelante se tratará la monitorización.

4.4.2. Alta disponibilidad

Para garantizar la alta disponibilidad de las aplicaciones en un clúster, en esencia, se debe minimizar el tiempo de inactividad y garantizar que las aplicaciones sigan siendo accesibles en caso de fallas. Esto se logra con varias estrategias, una de ellas es el ya mencionado uso de réplicas.

Definir políticas de anti-afinidad para los pods, esto hace que un pod o varios nodos cercanos no programen las mismas replicas, reduciendo así que un error afecte la misma aplicación. Se debe configurar el componente Kube Proxy el cual "es un componente relevante de Kubernetes que se ejecuta en cada nodo del clúster; su tarea principal consiste en gestionar las reglas de red en los nodos y llevar a cabo la traducción de direcciones de red (NAT) para facilitar la comunicación entre los diversos pods dentro del clúster" (Kubernetes, Accedido el 20 de septiembre de 2023-a). Esto asegura que se utilicen IPs y puertos consistentes para nuestros servicios. Es pertinente agregar que es necesario establecer prácticas de restauración de datos para proteger las aplicaciones en caso de un error con nuestro clúster.

4.4.3. Configuración de recursos (CPU y memoria)

Es necesario configurar los límites de CPU y memoria para tus contenedores y pods mediante la especificación de recursos en el archivo YAML de la definición del pod. Esto para controlar y asignar recursos de manera efectiva, evitando que una aplicación utilice todos los recursos disponibles en un nodo y afecte negativamente a otras aplicaciones en el mismo clúster.

4.4.4. CI/CD

La implementación de CI/CD en un clúster de Kubernetes es esencial para automatizar y agilizar el proceso de desarrollo y despliegue de aplicaciones en contenedores. Para lograr esto se pueden utilizar herramientas como Jenkins que "es una herramienta empleada para compilar y evaluar proyectos de software de manera continua, simplificando que los desarrolladores integren

modificaciones en un proyecto y proporcionen nuevas versiones a los usuarios; es compatible con diversas plataformas y se accede a través de una interfaz web; actualmente, es la herramienta más utilizada para este cumplir con este propósito" (Sentrio, Accedido el 20 de septiembre de 2023), esta permite la configuración de tuberías de CI/CD personalizadas para aplicaciones en Kubernetes.

4.5. Seguridad del clúster

La seguridad del clúster de Kubernetes es fundamental para proteger las aplicaciones y datos, por lo cual una práctica necesaria es asignar roles y permisos necesarios para las cuentas de servicio y usuarios, esto va de la mano con hacer uso de sistemas de autenticación sólidos, como aquellos basados en tokens como OIDC (OpenID Connect).

Es importante escanear y validar las imágenes de contenedor antes de su implementación para detectar vulnerabilidades conocidas además de utilizar imágenes de contenedor oficiales y de confianza; por otra parte, se deben implementar políticas de red adecuadas para limitar la comunicación entre los pods y los clústeres, esto restringiendo el tráfico de red a nivel de pod. Es pertinente almacenar un secreto, el cual "es un objeto que almacena datos confidenciales de pequeño tamaño, como contraseñas, tokens o claves; de lo contrario, esa información podría incorporarse en una especificación de Pod o en una imagen de contenedor. Utilizar un secreto implica que no es necesario incluir datos confidenciales en el código de la aplicación" (Kubernetes, Accedido el 20 de septiembre de 2023-e), de manera segura utilizando Kubernetes Secrets; haciendo uso de esta herramienta se evita el uso de secretos codificados en YAML y archivos de configuración.

Es recomendable implementar herramientas de análisis de seguridad como Falco para detectar comportamientos anómalos, esta herramienta "ayuda en la identificación de comportamientos inusuales, posibles amenazas a la seguridad y violaciones de cumplimiento, contribuyendo así a lograr una seguridad completa durante la ejecución" («Falco», Accedido el 20 de septiembre de 2023).

Utilizar el Control de Acceso Basado en Roles (RBAC), es una esencial en Kubernetes, ya que "facilita la administración y control del acceso a recursos y acciones dentro de un clúster de Kubernetes" (Kubernetes, Accedido el 20 de septiembre de 2023-f). RBAC permite definir quién puede hacer qué en el clúster y es una parte esencial de garantizar la seguridad de tu entorno Kubernetes.

4.6. Monitorización del clúster

Para garantizar el rendimiento, la disponibilidad y la seguridad de tus aplicaciones en contenedores, es necesario implementar la monitorización en un clúster de Kubernetes. Se pueden utilizar herramientas como Prometheus, la cual "es una herramienta de supervisión ampliamente utilizada en el entorno de Kubernetes, que posibilita la recopilación de métricas, alertas, entre otras" («Prometheus monitoring», Accedido el 20 de septiembre de 2023). Prometheus se integra bien con Kubernetes y es altamente configurable; por su parte, también se puede incluir el uso de Grafana, esta "se trata de una herramienta de visualización y creación de paneles que se integra de manera fluida con Prometheus y otros sistemas de supervisión" («Kubernetes monitoring»,

Accedido el 20 de septiembre de 2023), esto puntualmente para las visualizaciones de métricas.

Estas herramientas principalmente tienen usos como recopilar las métricas del clúster, nodos, a su vez como métricas de aplicaciones para identificar problemas de rendimiento y de uso de recursos. Es fundamental agregar alertas en función de umbrales específicos para las métricas críticas.

Luego de implementar las características previamente mencionadas en el clúster de Kubernetes, resulta pertinente realizar pruebas de estrés y evaluación de para evaluar la capacidad del clúster y aplicaciones bajo cargas elevadas.

Basándonos en esta base conceptual, se debe profundizar en la metodología de investigación.

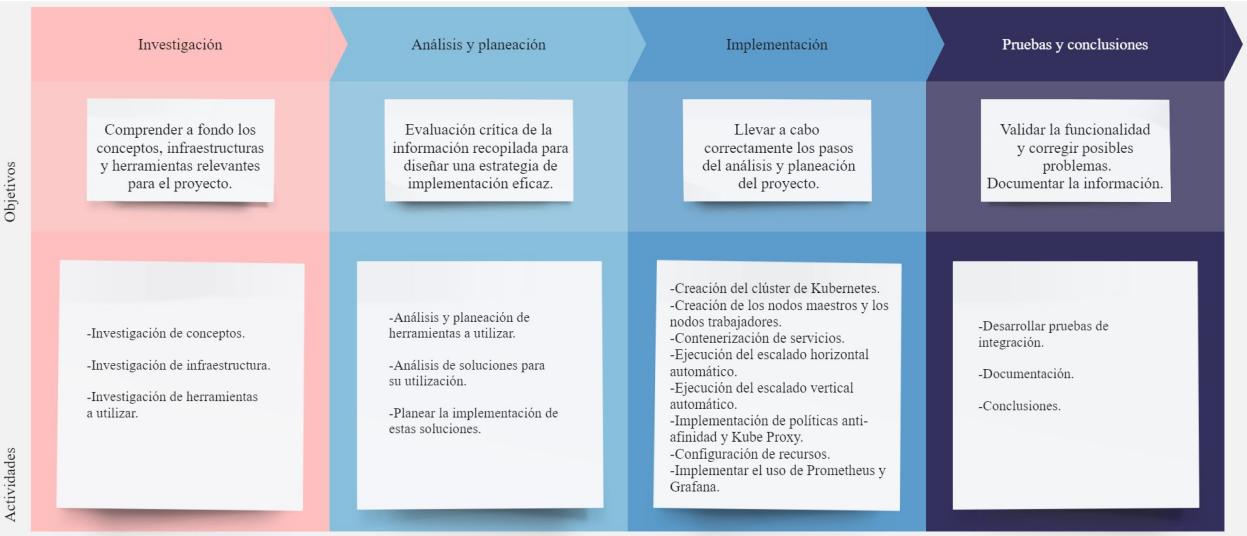
5. Metodología

Para alcanzar los objetivos propuestos, la implementación de Kubernetes on-premise siguió ciertas métricas para cumplir con la orquestación de servicios web, este caso concreto, como objeto de estudio, de la Universidad insdustrial de Santander en el marco del proyecto RSI; teniendo en cuenta que se debió diseñar y ejecutar un prototipo el cual fue propenso a cambios constantes y mejoras, fue conveniente hacer uso de la metodología de prototipado evolutivo.

El objetivo de esta metodología fue acelerar el proceso de implementación del clúster de Kubernetes mediante la creación de prototipos iterativos y pruebas constantes; partiendo del principio que esta metodología está asociada a la idea de desarrollar diferentes conceptos propuestos mediante prototipos, para su posterior evaluación, permitiendo variar constantemente nuestro prototipo incial hasta cumplir con los requerimientos y los objetivos propuestos.

Figura 4.

Gráfico de las fases y los procesos de la metodología.



Nota: Vargas, D. (2024). Diseño propio.

5.1. Investigación

Como primera instancia, luego de definir los objetivos, requerimientos del proyecto y el alcance; fue pertinente realizar una investigación detallada de conceptos, la arquitecura y las herramientas para implementar Kubernetes, además de esto, qué funcionalidades nos brindaba cada una de ellas y sus costos.

5.2. Análisis y planeación

Fue necesario realizar el análisis y estudio de las herramientas previamente investigadas, teniendo en cuenta las variables mencionadas, como, el número de nodos que estas permiten, la cantidad de recursos que consumen, dificultad de instalación y nivel de personalización, en el caso

de las herramientas públicas, sus costos económicos; posteriormente se planeó cómo desarrollar la implementación, mediante una perspectiva reduccionista, para desarrollar los componentes más simples del clúster primero.

5.3. Implementación

Se desarrollaron prototipos iniciales con las configuraciones básicas del clúster de Kubernetes con la herramienta seleccionada, realizando pruebas con estos para evaluar su viabilidad, retroalimentando con el equipo de Dev/Ops, a su vez, mejorarando los prototipos existentes para abordar los problemas identificados y agregar características necesarias.

Fue pertinente establecer un despliegue gradual, creando el clúster inicial, creando los nodos y conterizando los servicios elegidos de la Universidad Industrial de Santander en el marco del proyecto RSI; comenzando con componentes menos críticos del clúster, hasta implementar una solución de monitoreo para realizar un seguimiento continuo de escalabilidad y disponibilidad.

5.4. Pruebas y conclusiones

Finalmente, se realizó una revisión del prototipo final para asegurar que se cumplían los objetivos y el alcance definidos; fue necesario desarrollar pruebas de integración para evaluar la interacción entre diferentes componentes del clúster; por su parte, fue indispensable documentar todo el proceso de la implementación de Kubernetes, incluyendo configuraciones, procedimientos y conclusiones.

La metodología de prototipado evolutivo fue de suma importancia en la implementación de Kubernetes a través de ciclos iterativos de ejecución, evaluación y mejora; el desarrollo del proyecto se ajustó a estas fases, permitiendo la flexibilidad necesaria para garantizar el cumplimiento de los objetivos.

6. Desarrollo

En esta sección, se exploró detalladamente el proceso de desarrollo que atraviesa cada una de las etapas de la metodología previamente expuesta.

A lo largo del proyecto, se empleó la metodología de prototipado evolutivo, teniendo en cuenta las cuatro etapas de la metodología previamente expuesta. En la fase de investigación, se exploraron los requisitos y tecnologías relevantes, construyendo prototipos conceptuales para validar ideas y mejorar constantemente la ejecución de estas. En la fase de análisis, se definieron especificaciones detalladas y se diseñaron prototipos más elaborados con las herramientas investigadas, visualizando las ventajas y desventajas de estas. Durante la fase de implementación, se construyeron prototipos funcionales del sistema de manera incremental, mientras que en la fase de pruebas y conclusiones, se llevaron a cabo pruebas del sistema implementado y se extrajeron conclusiones para futuras iteraciones de implementación de esta prueba piloto. A continuación, se detalla cada una de las etapas:

6.1. Investigación

En esta fase inicial, se llevó a cabo una exhaustiva revisión de la literatura relacionada con kubernetes, la infraestructura orquestada, sus componentes y herramientas principales. Se analizaron investigaciones previas y estudios respecto a la contererización de servicios web. Esta revisión

proporcionó una comprensión sólida del estado actual del conocimiento en el área y sirvió como base para la conceptualización y diseño de la investigación.

6.1.1. Investigación de conceptos

Para empezar, se exploró a fondo el significado y propósito de Kubernetes, siendo esta una plataforma diseñada para la automatización, escalabilidad y operación de aplicaciones en contenedores, incluyendo los retos que hay al utilizarla; no solo centrandose en esta tecnología sino también en el contexto historico en el cual se empieza a utilizar esta.

Se desglozaron conceptos esenciales dentro del ecosistema de Kubernetes, incluyendo terminos como los pods (las unidades más pequeñas de despliegue), servicios (para la comunicación entre componentes), controladores (que gestionan el estado deseado del sistema), y despliegues (para la gestión declarativa de aplicaciones), entre otros previamente mencionados en el marco de referencia; a su vez incluyendo estrategias de configuración, gestión de secretos, consideraciones específicas para entornos de producción y extensiones del clúster.

Además se recopiló información de diferentes recursos educativos, como documentación oficial, tutoriales, y libros, para facilitar una comprensión profunda de los conceptos de Kubernetes; teniendo así una base solida de conocimiento para la ejecución de este proyecto.

6.1.2. Investigación de infraestructura

Existen varias tecnologías y herramientas para desarrollar una infraestructura orquestada, siendo de las herramientas más convenientes Kubernetes. Utilizar kubernetes para implementar este tipo de arquitecturas, ofrece varias ventajas significativas, especialmente en términos de escalabilidad y alta disponibilidad de servicios web, criterios fundamentales en este proyecto.

Otra herramienta muy conocida para llevar a cabo la implementación de una infraestructura orquestada es Docker Swarm, que "es una herramienta de software que posibilita la ejecución de contenedores en un conjunto de nodos; esto involucra uno o varios balanceadores de carga instalados en uno o más nodos maestros, junto con los nodos que proporcionan el servicio, desplegados en nodos trabajadores" (Making Science, 2021); sin embargo, es preferible utilizar Kubernetes pese a que Docker Swarm puede ser considerado menos complejo y más fácil de configurar que Kubernetes. Por su parte, esta herramienta puede ser una excelente opción para casos de uso más simples. Sin embargo, en entornos empresariales y aplicaciones más complejas que requieren escalabilidad, disponibilidad y otras funcionalidades avanzadas, por lo tanto, Kubernetes es conveniente para este propósito por las siguientes razones:

- Permite la gestión eficiente y escalable de contenedores, facilitando el despliegue, actualización y escalado de aplicaciones en entornos de producción.
- Facilita la escalabilidad automática de aplicaciones web. Puede ajustar dinámicamente el número de instancias de la aplicación en función de la carga de trabajo, asegurando que los

recursos estén disponibles según sea necesario.

- Está diseñado para garantizar la alta disponibilidad de las aplicaciones. Puede distribuir las cargas de trabajo entre nodos del clúster, garantizando que, incluso en caso de falla de un nodo, los servicios sigan estando disponibles.

- Permite desplegar nuevas versiones de la aplicación sin tiempo de inactividad, mediante rolling updates y a su vez, permite realizar rollbacks de manera rápida y controlada.

- Ofrece servicios integrados y balanceo de carga para distribuir el tráfico entre las instancias de la aplicación. Esto mejora la eficiencia y garantiza una distribución equitativa de las solicitudes, mejorando la experiencia del usuario y la estabilidad del sistema.

- Permite desplegar aplicaciones en múltiples nodos, ya sea en entornos locales o en la nube. Esto mejora la redundancia y la disponibilidad al distribuir las aplicaciones en diferentes ubicaciones físicas o virtuales.

- Finalmente, cuenta con un ecosistema amplio de herramientas y servicios adicionales que complementan sus capacidades. Esto incluye soluciones de monitoreo, registro, seguridad y más, que facilitan la gestión completa del ciclo de vida de las aplicaciones.

6.1.3. Investigación de herramientas a utilizar

En esta sección se muestran las herramientas más relevantes a la fecha, para implementar un clúster de Kubernetes, estas están seccionadas en herramientas on-premise y en la nube.

6.1.3.1. Herramientas on-premise. Para implementar Kubernetes on-premise, o, en otras palabras, en servidores locales, se pueden utilizar algunas herramientas, las cuales brindan diferentes utilidades; elegir cualquiera de estas va a depender de la organización la cual la requiera, teniendo en cuenta sus recursos disponibles, número de nodos necesarios, el manejo que se tiene para desarrollar dicha implementación, las mejores opciones actualmente son:

- **Kubeadm**: Kubeadm es una herramienta oficial de Kubernetes que permite crear y gestionar clústeres de Kubernetes desde cero, simplifica el proceso de inicio de clústeres Kubernetes, siendo útil para implementaciones personalizadas y configuraciones específicas.

  Los requerimientos principales que requiere esta herramienta son un host Linux compatible, 2 GB o más de RAM por máquina, 2 CPU o más, además de una conectividad de red completa entre todas las máquinas del clúster, teniendo un nombre de host, dirección MAC y product-uuid únicos para cada nodo («Installing kubeadm», 2023).

- **Kubespray**: Kubespray es una herramienta de código abierto que automatiza la implementación y administración de clústeres de Kubernetes en máquinas virtuales, siendo ideal para entornos que requieren una implementación automatizada en máquinas virtuales, permitiendo una gestión eficiente. Esta herrmaienta permite centrarse más en los proceso de configuración de las máquinas a nivel de sistema operativo.

  La opción de Kubespray está formada por una serie de inventarios, playbooks, elementos de aprovisionamiento y otras herramientas proporcionadas por Ansible, siendo esta “una plata-

forma de automatización diseñada para administrar extensos conjuntos de sistemas informáticos. Facilita la automatización de diversas tareas, como el despliegue de aplicaciones, la gestión de configuraciones, el aprovisionamiento en la nube, y la actualización de estaciones de trabajo y servidores" (Invgate, 2023). Tal que, al integrar características de Kubernetes y Ansible, Kubespray se transforma en una fusión de ambas plataformas, permitiendo la creación de clústeres con herramientas centradas en la automatización.

Las características que presenta esta herramienta abarcan la alta disponibilidad para eliminar los puntos individuales de falla, lograr redundancia y protección del sistema. También cuenta con la capacidad de realizar varias pruebas de integración de manera continua, asegurándose de supervisar la plataforma; por último, ofrece respaldo para implementaciones en servidores en la nube de diversas soluciones públicas.

La implementación de esta herramienta proporciona beneficios como la flexibilidad en los procesos de ejecución, posibilitando la rápida implementación de un clúster; además, no descuida las acciones vinculadas a la personalización, tales como las opciones de tiempo de ejecución del componente, la configuración DNS y las versiones de componentes, incluyendo su fácil manejo.

Esta es una excelente solución para organizaciones que ya están familiarizadas con Ansible; por su parte requiere que ya esté instalado AWS CLI y Terraform en nuestro sistema (KeepCoding, 2023).

- **Rancher**: Rancher es una plataforma de administración de contenedores que puede utilizarse

para implementar y gestionar clústeres de Kubernetes; proporciona una interfaz gráfica, sin embargo, esto hace que consuma más recursos.

Rancher posibilita la ejecución de contenedores en entornos de producción, simplifica la creación de clústeres, la administración de múltiples clústeres de Kubernetes, proporcionando una interfaz de usuario fácilmente accesible para la supervisión y gestión de estos clústeres. No es necesario construir una plataforma de servicios de contenedores desde el principio, en su lugar, esta distribución proporciona todo el software esencial para administrar contenedores directamente en entornos de producción.

Rancher es una herramienta valiosa cuando se trata de administrar numerosos clústeres, ya que resulta beneficiosa para satisfacer los requisitos de TI mediante el empleo de un sistema de autenticación centralizado.

Esta herramienta ofrece funcionalidades como la capacidad para establecer redes entre hosts, generando una red privada para cada entorno que facilita la comunicación segura entre contenedores a través de distintos hosts y entornos en la nube; además, proporciona un servicio de equilibrio de carga integrado y escalable para distribuir el tráfico entre contenedores o servicios.

Rancher es una herramienta de gestión sencilla gracias a sus diversas herramientas y funciones incorporadas en la distribución, incluyendo la interfaz de gestión centrada en el usuario, capaz de simplificar diversas complejidades y, en última instancia, reducir la curva de aprendizaje de Kubernetes para los equipos de DevOps; por su parte, Rancher ofrece funciones

para consolidar estos clústeres, permitiendo que un único servidor se conecte a entre uno y varios miles de ellos (Arsys, 2023).

- **OpenShift**: OpenShift es una plataforma de contenedores basada en Kubernetes, especialmente indicado para entornos empresariales que buscan características adicionales y un soporte más robusto.

  OpenShift utiliza CRI-O en lugar de Docker o Containerd, brindando algunas ventajas al ser una tecnología que no se basa en la contenerización mediante el uso de Docker.

  OpenShift se utiliza principalmente para crear entornos de desarrollo y aplicaciones. Esto permite implementar soluciones PaaS, SaaS y CaaS propias. Debido al poder y la complejidad de este software, OpenShift es utilizado principalmente para proyectos de larga duración de grandes organizaciones, sobre todo por su alto nivel de seguridad ya que esta es indispensable en los entornos empresariales.

  En un clúster de alta disponibilidad de OpenShift con etcd externo y 20 GB de espacio en disco para datos de etcd, se establece que un host maestro debe cumplir con requisitos mínimos, necesitando 1 núcleo de CPU y 1,5 GB de memoria por cada 1000 pods; por lo tanto, para un clúster de OpenShift con 2000 pods, el tamaño recomendado para un host maestro serían los requisitos mínimos de 2 núcleos de CPU y 16 GB de RAM, junto con 2 núcleos de CPU y 3 GB de RAM adicionales, sumando un total de 4 núcleos de CPU y 19 GB de RAM (Red Hat, 2011).

- **K3s**: K3s es una distribución ligera de Kubernetes diseñada para implementaciones sencillas

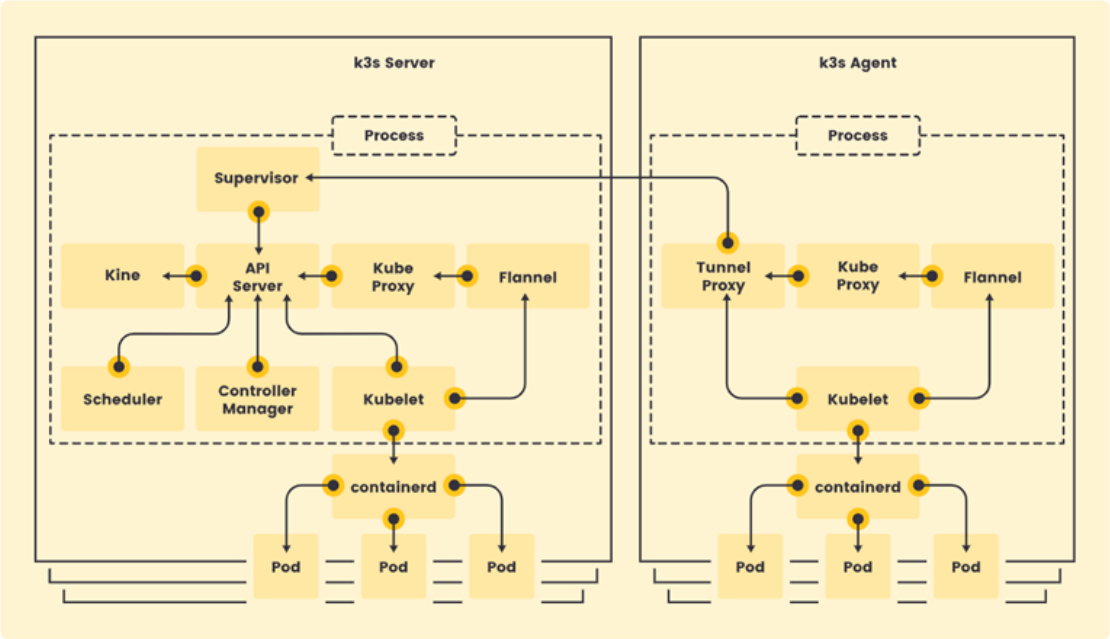
de alta disponibilidad. Está optimizada para recursos limitados además de ser fácil de instalar, adecauda para ambientes de producción en ubicaciones remotas donde la simplicidad y la eficiencia son prioritarias y con recursos limitados o dentro de dispositivos IoT.

Esta herramienta está empaquetada como un único binario de <70 MB, reduciendo las dependencias y los pasos necesarios para implementar un clúster de Kubernetes de producción. K3s está disponible para arquitecturas como x86-64, brazohf, brazo64/aarch64, s390x.

Los requisitos de hardware mínimos son 1 núcleo y 512 GB de RAM, recomendados son 2 núcleos y 1 GB de RAM, teniendo en cuenta que estos requisitos aumentan según el tamaño del clúster; por su parte, para implementar un clúster de alta disponibilidad, con una cantidad de <10 nodos se requieren 2 vCPU´s y 4 GB de RAM, con una cantidad de <100 nodos se requieren 4 vCPU´s y 8 GB de RAM, con una cantidad de <250 nodos se requieren 8 vCPU´s y 16 GB de RAM, con una cantidad de <500 nodos se requieren 16 vCPU´s y 32 GB de RAM, con una cantidad de 500<nodos se requieren 32 vCPU´s y 64 GB de RAM («K3s - Lightweight Kubernetes», 2023).

Figura 5.

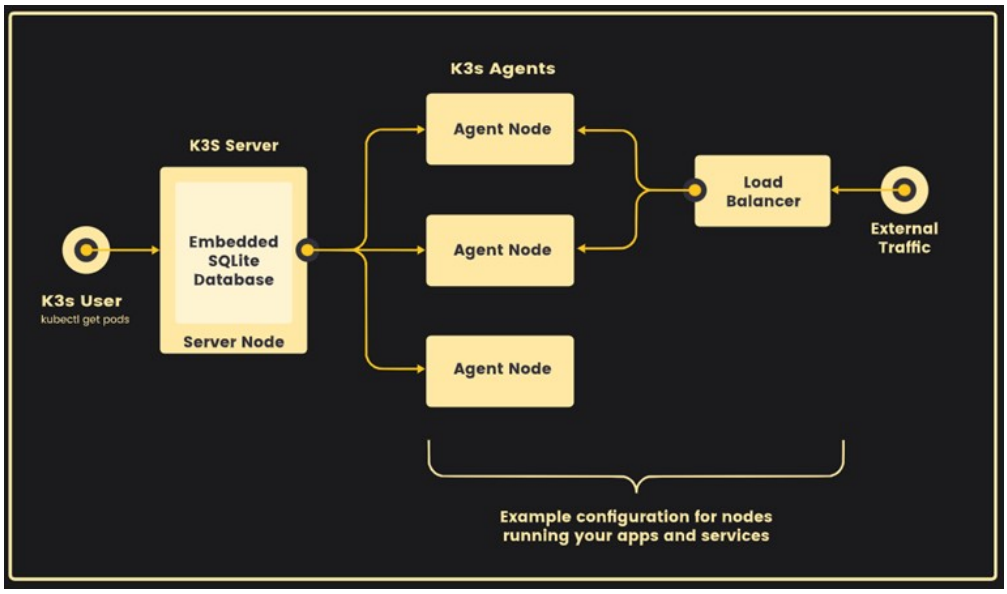
Gráfico de la arquitectura a grandes rasgos de un clúster de Kubernetes en K3s.



Nota: Arquitectura de un clúster de Kubernetes de K3s. Adaptado de: («K3s - Lightweight Kubernetes», 2023)

Figura 6.

Gráfico de la arquitectura y algunos componentes de un clúster de Kubernetes en K3s.



Nota: Componentes principales de la arquitectura de un clúster de Kubernetes de K3s. Adaptado de: («K3s - Lightweight Kubernetes», 2023)

- **Kubernetes Operations**: Kubernetes Operations o Kops es una herramienta que simplifica la creación, actualización y gestión de clústeres de Kubernetes en infraestructuras locales, también cuenta con una implementación en la nube, incluyendo AWS y GCP, esto es útil para el caso en el que se requieran migrar los servicios a la nube.

  Kops no solo lo ayuda a desarrollar un clúster Kubernetes de alta disponibilidad y grado de producción, sino que también proporciona la infraestructura de nube necesaria, cuenta con soporte con Amazon Web Services, Google Cloud Platform, DigitalOcean , OpenStack y Azure; automatizando el aprovisionamiento de clústeres de Kubernetes de alta disponibilidad. Permite personalizar los componentes para que se ajusten a necesidades específicas,

siendo una herramienta considerable para implementar soluciones on-premise con integración en la nube.

Es pertinente mencionar que Kops no proporciona una gestión integral de recursos como un administrador de clústeres completo, ya que se necesita complementar Kops con otras herramientas para la gestión de aplicaciones, monitoreo y registro; por su parte, aprender a utilizar Kops y comprender sus numerosas opciones puede llevar tiempo, lo que podría retrasar la implementación de un clúster de Kubernetes en una infraestructura local («Kops - Kubernetes Operations», 2023).

6.1.3.2. Herramientas en la nube. Al implementar Kubernetes en la nube se obtiene gran escalabilidad y la administración es simplificada, sin embargo, no siempre es la solución más económica, dependiendo de la organización es viable utilizar una solución pública, teniendo en cuenta que muchas herramientas, al tener un manejo más simple de esta, es más costosa; las principales opciones son:

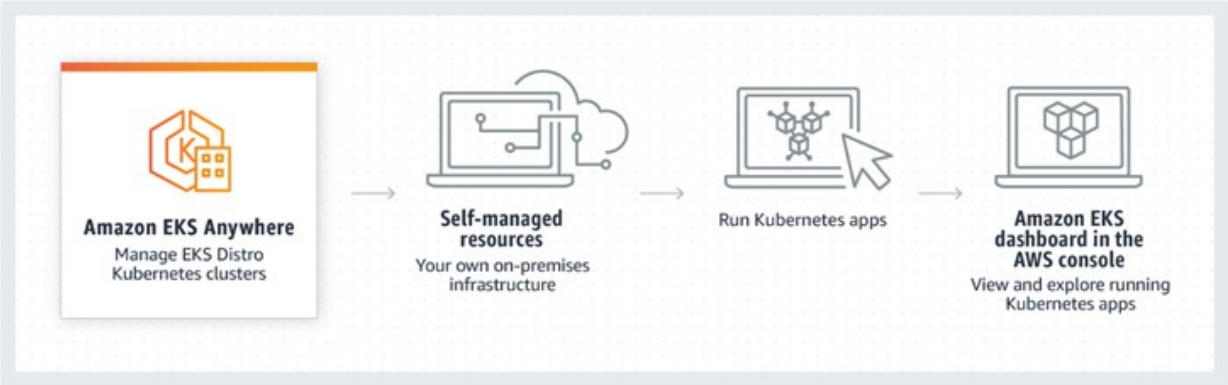
- **Amazon Elastic Kubernetes Service**: EKS es un servicio de Kubernetes que gestiona AWS, proporciona integración con otros servicios de AWS y es ampliamente utilizado para despliegues de Kubernetes, ofreciendo escalabilidad y facilidad de gestión; diseñado para ejecutar clústeres de Kubernetes tanto en la nube como en entornos locales. En la nube, EKS automatiza la administración de la disponibilidad y la escalabilidad de los nodos del plano de control de Kubernetes, proporcionando un rendimiento escalable y confiable. Con integra-

ciones con servicios clave de AWS, como CloudWatch, Auto Scaling Groups e IAM, ofrece una experiencia integral para monitorear, escalar y equilibrar aplicaciones en contenedores.

Amazon EKS ofrece una experiencia nativa de Kubernetes para funciones de malla de servicios, brindando observabilidad, control de tráfico y seguridad a las aplicaciones. Proporciona un plano de control escalable y de alta disponibilidad, automatizando tareas clave como la aplicación de parches y la gestión de nodos. El costo es de 0,10 USD por hora por clúster de EKS, con la flexibilidad de ejecutar múltiples aplicaciones en un solo clúster mediante el uso de espacios de nombres de Kubernetes y políticas de seguridad de IAM («Amazon EKS - Elastic Kubernetes Service», 2023).

Figura 7.

Gráfico del servicio de un clúster de Kubernetes en EKS.



Nota: Servicio que ofrece a grandes rasgos EKS. Adaptado de: («Amazon EKS - Elastic Kubernetes Service», 2023)

- **Google Kubernetes Engines**: GKE es un servicio de Kubernetes que gestiona Google Cloud, este ofrece una integración con otros servicios de la nube de Google, proporciona una gestión completamente automatizada de los clústeres, con autoescalado de pods y clústeres, visibilidad de costes y optimización automatizada de infraestructuras.

  La edición premium GKE Enterprise agrega funciones de gestión, gobierno, seguridad y configuración para equipos y clústeres, con una consola unificada y una malla de servicios integrada. El modo Autopilot de GKE permite gestionar la computación del clúster sin intervención, ofreciendo una experiencia de Kubernetes completa. Con una facturación por pods, se paga solo por los pods en ejecución, logrando ahorros de hasta el 85 %. Ambos modos, Autopilot y Standard, están disponibles en GKE Enterprise, siendo esta versión la más completa.

  GKE es compatible con todas las APIs de Kubernetes, autoescalado, gestión de varios clústeres y escala hasta 15,000 nodos. El autoescalado de pods horizontal se basa en el uso de CPU u otras métricas personalizadas, mientras que el autoescalado de clústeres opera por grupos de nodos y el autoescalado de pods vertical ajusta automáticamente las solicitudes de CPU y memoria según el uso continuo. La edición Enterprise tiene un costo de 0,0083 USD por vCPU y hora; por su parte la edición Standard tiene un costo de 0,10 USD por clúster y hora; así mismo, Google ofrece un número de creditos gratuito al mes para utilizar este servicio («Google Kubernetes Engine», 2023).

- **Microsoft Azure Kubernetes Service**: AKS es un servicio de Kubernetes que administra

Microsoft Azure, permite implementar, administrar y escalar clústeres de Kubernetes en la nube de Microsoft. Facilita la implementación, administración y escalabilidad de clústeres de Kubernetes, permitiendo el desarrollo y despliegue rápido de aplicaciones nativas de nube en Azure, centros de datos o en el perímetro. Proporciona canalizaciones integradas de código a nube y límites de protección.

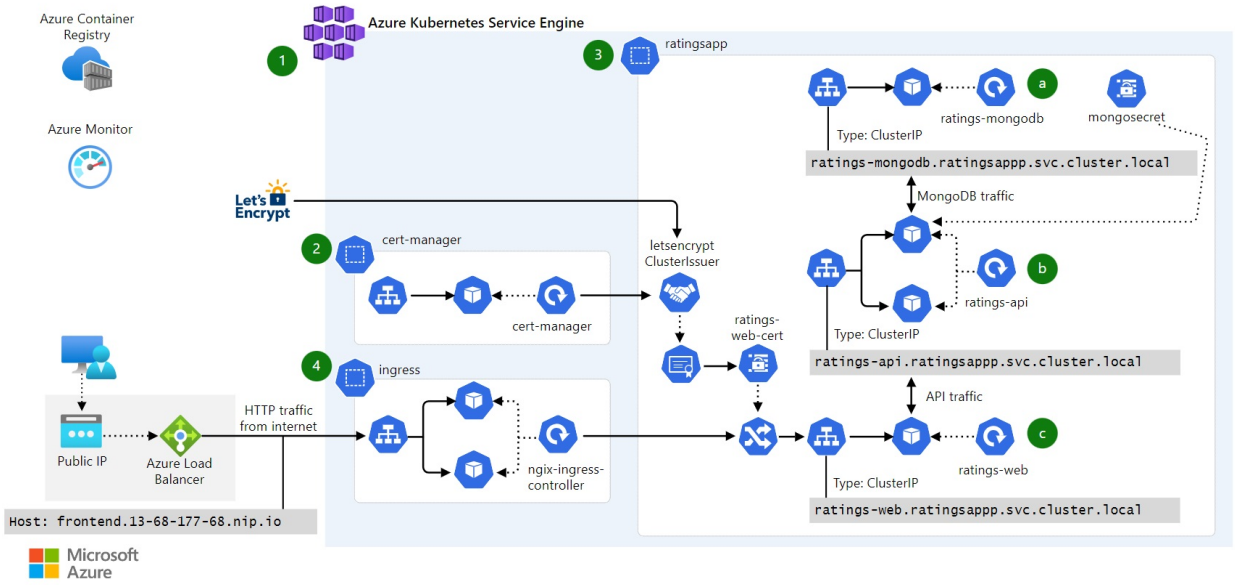
AKS ofrece una gestión unificada y gobernabilidad para clústeres Kubernetes en entornos locales, perimetrales y de nube múltiple. Interopera con servicios de seguridad, identidad, administración de costos y migración de Azure. Automatiza la administración y escalabilidad de clústeres, ofreciendo una orquestación de contenedores de nivel empresarial.

Para desarrolladores, brinda productividad desde la depuración hasta la implementación continua (CI/CD), registro y mantenimiento automatizado de nodos. La administración avanzada de identidades y acceso asegura la seguridad de los contenedores a escala.

Los precios del uso de esta herramienta dependen del nivel de automatización que se requiera, teniendo el nivel standard un costo de 0,10 USD por clúster y hora; por su parte, el nivel premium al incluir dos años de soporte técnico y un constante mantenimiento de los clústeres utilizados, tiene un costo de 0,60 USD por clúster y hora; para ambos casos teniendo un límite de 5000 nodos («Azure Kubernetes Service», 2023).

Figura 8.

Gráfico de la arquitectura y algunos componentes de un clúster de Kubernetes en AKS.



Nota: Componentes principales de un clúster de Kubernetes de AKS. Adaptado de: (Microsoft Azure, 2023)

- **IBM Cloud Kubernetes Service**: Este servicio de IBM Cloud permite la implementación y gestión de clústeres de Kubernetes en su nube. Diseñado como un servicio gestionado de contenedores basado en Kubernetes, simplifica la implementación, gestión y escalabilidad de clústeres en IBM Cloud, proporcionando una plataforma completa para desarrolladores y equipos operativos, facilitando la escalabilidad vertical y horizontal del clúster para adaptarse a las necesidades cambiantes de la carga de trabajo.

  Ofrece una plataforma completamente gestionada para simplificar la implementación y operación de clústeres de Kubernetes, brindando soporte para las versiones más recientes de Kubernetes, permitiendo a los usuarios aprovechar las últimas características y mejoras, te-

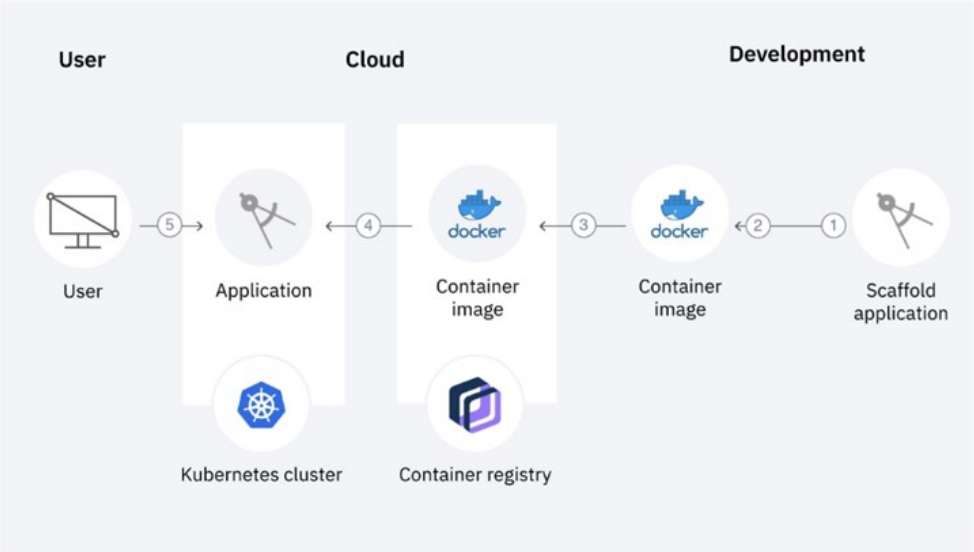
niendo integración con algunos de los servicios de IBM Cloud, permitiendo a los usuarios aprovechar servicios adicionales como bases de datos, servicios de IA, almacenamiento, etc.

Proporciona herramientas y API que facilitan la integración con flujos de trabajo de desarrollo y despliegue continuo (CI/CD) cumpliendo con estándares de seguridad y conformidad, haciéndolo adecuado para cargas de trabajo sensibles; a su vez, permite la implementación de clústeres en diferentes regiones geográficas de los centros de datos de IBM Cloud, proporcionando flexibilidad para adaptarse a requisitos de geolocalización brindando una menor latencia en sus servicios.

Los costos de esta herramienta varían dependiendo de las características del clúster, en el caso más simple, con 2 vCPU´s y 4 GB de RAM, los costos son 0,13 USD por clúster y hora («IBM Cloud Kubernetes Service», 2023).

Figura 9.

Gráfico del servicio de un clúster de Kubernetes en IBM Cloud.



Nota: Servicio que ofrece a grandes rasgos IBM Cloud. Adaptado de: («IBM Cloud Kubernetes Service», 2023)

- **DigitalOcean Kubernetes**: DOKS es un servicio de Kubernetes administrado por DigitalOcean. Es fácil de usar y está diseñado para pequeñas y medianas empresas, optimiza el uso de recursos computacionales, memoria y almacenamiento al programar instancias en todo el clúster, maximizando la eficiencia con un plano de control que rota clústeres y escala rápidamente según la carga de API mediante el escalador automático de pods horizontales.

  Utilizando una solución que permita la alta disponibilidad y escalabilidad, esta herramienta tiene un costo de 63 USD por nodo y mes («DigitalOcean Kubernetes», 2023).

- **Oracle Cloud Infraestructure Container Engine for Kubernetes**: OKE es el servicio de Kubernetes gestionado de Oracle Cloud el cual simplifica las operaciones de Kubernetes

para empresas, reduciendo tiempo y costos. Este servicio permite desplegar clústeres de Kubernetes con operaciones confiables, escalado automático y actualizaciones. OKE ofrece una experiencia sin servidor con nodos virtuales. Al crear un clúster, se puede optar por uno básico sin tarifa base, adecuado para clientes que asumen más tareas de gestión. Sin embargo, esta opción no incluye funciones avanzadas ni el acuerdo de nivel de servicio. Los clústeres básicos son ideales para quienes no requieren las funcionalidades avanzadas del clúster estándar, pero pueden migrar fácilmente si es necesario en el futuro. Esta herramienta con los componentes que necesitamos tiene un costo de 0.015 USD por nodo y hora («Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE)», 2023).

6.2. Análisis y planeación

Con base en la información previamente investigada, teniendo en cuenta los recursos disponibles y las necesidades y características con las que debe cumplir el clúster de kubernetes; se planeó como se desarrolla la implementación para cumplir con los objetivos y así contribuir al éxito de este trabajo.

6.2.1. Análisis y planeación de herramientas a utilizar

Antes de seleccionar las herramientas específicas, fue esencial realizar un análisis detallado de los requisitos del sistema y las metas del clúster Kubernetes que se requiere implementar. En esta sección se muestran los criterios y métricas tenidos en cuenta para la selección de soluciones destinadas a la orquestación de servicios web, permitiendo la escalabilidad y alta disponibilidad de

estos; por su parte, estas variables fueron acotadas a las necesidades de RSI, sin embargo, pueden tenerse en cuenta para cualquier organización.

Al seleccionar una herramienta para implementar Kubernetes on-premise, a grandes rasgos es importante considerar las siguientes variables:

- ¿Cuántos nodos brinda la herramienta para soportar las cargas de trabajo respecto al consumo de recursos?
- La capacidad de optimizar recursos físicos, en cuánto a CPU y RAM es crucial para garantizar el rendimiento adecuado de los servicios web.
- La facilidad de configuración e desarrollo de la herramienta.
- La capacidad de personalización de la herramienta para cumplir con las características necesarias del clúster.
- La conectividad de red entre los nodos del clúster y alguna posible limitación entre políticas anti afinidad.
- Considerar los aspectos de seguridad que incluye determinada herramienta.

Para seleccionar una herramienta de implementación de Kubernetes en la nube, además de las variables expuestas previamente para un entorno on-premise, se deben considerar las siguientes variables:

- Los costos asociados con la implementación en la nube, que pueden incluir tarifas de servicios, almacenamiento, ancho de banda, entre otros.
- Algunas herramientas pueden tener costos asociados con licencias, especialmente en entornos empresariales.

6.2.2. Análisis de soluciones para su utilización

Tanto para soluciones on-premise como en la nube, se realizó un análisis detallado de cada una de estas. Como objeto de estudio y en el contexto de los servicios web de RSI, se requirió de una infraestructura orquestada escalable y altamente disponible; el clúster de Kubernetes en principio, requiere una capacidad aproximada de 100 nodos inicialmente, con recursos límitados, rondando los 16 vCPU's y 32 GB de RAM, es necesaria la capacidad de personalización, conectividad de red entre los nodos del clúster manejando una baja latencia y seguridad.

Kubeadm se centra en la creación y gestión básica de clústeres, careciendo de algunas funciones de nivel empresarial que podrían ser esenciales en entornos más complejos y exigentes, aunque es una herramienta robusta y proporciona un control granular sobre la configuración, puede resultar desafiante para equipos con menos experiencia técnica. La personalización puede ser un proceso complejo; puede volverse más complejo a medida que aumenta el tamaño del clúster, la gestión de un gran número de nodos puede requerir configuraciones adicionales y una mayor carga administrativa, perjudicando la escalabilidad.

Kubespray tiene una configuración extensa y requiere de un conocimiento profundo de los

detalles internos de Kubernetes. Esto podría resultar en una curva de aprendizaje pronunciada y dificultar la implementación eficiente, a su vez, puede requerir una atención continua para actualizaciones y parches. El mantenimiento constante podría aumentar la carga operativa y la probabilidad de errores durante el ciclo de vida del clúster de Kubernetes; aunque ofrece automatización, la complejidad de la herramienta puede hacerla menos atractiva para implementaciones más simples y directas.

El uso de una herramienta como Rancher pueden consumir más recursos en comparación con soluciones más livianas, pese a que contiene una interfaz gráfica intuitiva, la inclusión de funciones adicionales puede resultar en una mayor sobrecarga; en este entorno no es conveniente el uso de esta ya que los recursos son límitados.

La amplitud de funciones de OpenShift la convierte en una buena solución para entornos empresariales, a su vez, esto puede resultar en una mayor complejidad operativa. Para implementaciones más simples, esta complejidad adicional podría no ser deseable. Su enfoque integral y robusto es una ventaja en entornos complejos, pero consume más recursos y tiempo de implementación que otras opciones más ligeras y flexibles.

K3s emerge como una opción atractiva y eficiente para implementar el clúster de Kubernetes, especialmente por el entorno con recursos limitados, demandando este escenario una distribución ligera. Una de las principales ventajas de K3s radica en su diseño optimizado para ser liviano y fácil de instalar, ocupando un espacio reducido en comparación con las implementacio-

nes convencionales de Kubernetes. Esto no solo simplifica el proceso de implementación, sino que también permite su despliegue en entornos con restricciones de recursos.

K3s se destaca por su simplicidad en la gestión de clústeres de Kubernetes. La instalación de K3s es rápida y directa, lo que reduce considerablemente las barreras de entrada y facilita su adopción, concretamente en este caso, donde la complejidad de otras herramientas podría representar un obstáculo.

Otro factor clave que respalda la elección de K3s es su eficiencia en la administración de recursos. Esta distribución de Kubernetes ha sido optimizada para garantizar un rendimiento sólido sin sacrificar funcionalidades esenciales. K3s incluye solo los componentes esenciales de Kubernetes, eliminando elementos redundantes y minimizando la sobrecarga del sistema, dado a su alto nivel de personalización se pueden agregar componentes necesarios para cumplir con los requerimientos. Esto no solo mejora el rendimiento general, sino que también contribuye a una administración más eficiente de los recursos disponibles, por su parte, con los recursos que se tienen, alcanzaría para un clúster con una capacidad de hasta 500 nodos, haciendo que sea conveniente el uso de esta.

Kops es una herramienta que simplifica la creación, actualización y gestión de clústeres de Kubernetes, al incluir una amplia variedad de funciones, Kops podría presentar una sobrecarga de características para implementaciones más simples, lo que resulta en un mayor consumo de recursos y complejidad innecesaria, esta es una buena opción de implementación, ya que es

muy eficiente sin tener unos requerimientos altos en cuanto a recursos se trata, la única dificultad considerable de esta herramienta es que no proporciona una gestión integral de recursos como un administrador de clústeres completo.

Es pertinente afirmar que al optar por una solución de Kubernetes on-premise, se tiene un control completo sobre la infraestructura de la organización ya que se puede personalizar la configuración de hardware y red según las necesidades, lo que permite una adaptación más precisa a los requisitos de las aplicaciones y de los servicios web teniendo una alta capacidad de personalización ya que el entorno no se limitaría por las restricciones impuestas por proveedores en la nube; además que se tiene un mayor control sobre la seguridad y el cumplimiento normativo, en terminos generales, para cualquier organización esto es crucial para aquellas industrias que deben cumplir con regulaciones estrictas y que requieren un control más directo sobre la gestión de datos sensibles.

A largo plazo, la implementación de una infraestructura on-premise puede ser más rentable en comparación con los costos variables asociados con los servicios en la nube, pese a que la inversión inicial puede ser más alta, además que se evitan posibles bloqueos y cambios de precios; por su parte, al tener control directo sobre la infraestructura se facilita la portabilidad ya que no está atada a un proveedor específico.

Para poder apreciar una verdadera comparativa de las herramientas expuestas anteriormente de forma cuantitativa, se elaboraron tablas comparativas y se desglosaron de forma más específica

cada una de las variables definidas en el punto anterior. Las calificaciones para cada herramienta fueron otorgadas con base en una exhaustiva investigación que incluye diversas fuentes y opiniones recopiladas en internet. A continuación, se presentan las tablas resultantes:

Tabla 1

Tabla comparativa de las herramientas para implementar Kubernetes on-premise.

Selección de herramientas para implementar Kubernetes on-premise									
Peso: 1=Baja, 3=Media, 5=Alta Importancia			Rango	5	6	2	4	1	3
Nota : 1=Débil, 3=Promedio, 5=Se adecua fuertemente			Total	206	204	294	288	298	292
Familia/ Criterio			Peso	Kubeadm	Kubespray	Rancher	OpenShift	K3s	Kops
1	Funcionalidad								
	1,01	**Interfaz de usuario**: Interfaz de usuario intuitiva y fácil de usar.Se	3	1	1	5	3	3	3
	1,02	**Confiabilidad:** Capacidad de la herramienta de mantener su nivel de ejecución bajo condiciones normales en un período de tiempo establecido	3	3	3	5	5	5	5
	1,03	**Seguridad**: Habilidad de prevenir el acceso no autorizado, sea accidental o premeditado, a los programas y datos.	5	5	5	5	5	5	5
	1,04	**Alcance:** Cantidad de nodos máximos que brinda la herramienta respecto al consumo de recursos, para soportar las cargas de trabajo.	5	3	3	5	5	5	5
	1,05	**Escalabilidad:** Capacidad de la herramienta para manejar un aumento en la carga de trabajo o en el tamaño del clúster.	5	3	3	5	5	5	5
	1,06	**Disponibilidad y tolerancia a fallos:** La herramienta proporciona mecanismos para garantizar la disponibilidad continua y la tolerancia a fallos en el clúster.	5	3	3	5	5	5	5
	1,07	**Personalización:** La capacidad de personalización de la herramienta para cumplir con las características necesarias del clúster.	5	3	5	5	5	5	5
	1,08	**Conectividad y anti-afinidad:** La conectividad de red entre los nodos del clúster y alguna posible limitación entre políticas anti afinidad.	3	3	3	3	5	3	5
	1,09	**Rendimiento:** Capacidad de la herramienta bajo carga pesada en términos de rendimiento.	5	3	3	5	5	5	5
2	Usabilidad								
	2,01	**Fácilidad:** La facilidad de configuración e desarrollo de la herramienta.	3	3	1	5	3	5	3
	2,02	**Documentación:** Documentación clara de la herramienta que facilite su implementación y uso	3	5	3	5	5	5	5
3	Eficiencia								
	3,01	**Gestión de recursos:** La capacidad de optimizar recursos físicos, en cuánto a CPU y memoria es crucial para garantizar el rendimiento adecuado del clúster.	5	3	3	5	5	5	5
	3,02	**Tiempo de implementación:** Tiempo que llevará implementar la herramienta y ponerla en producción.	3	3	3	5	5	5	5
4	Portabilidad y Mantenimiento								
	4,01	**Facilidad de mantenimiento:** Es fácil de mantener y actualizar.	3	3	3	5	5	5	3
5	Perspectiva de uso								
	5,01	**Comunidad y soporte:** Existe comunidad activa en torno a la herramienta y hay soporte del proveedor de la herramienta.	1	3	3	5	5	5	5
	5,02	**Costos:** Costos asociados a la implementación o de licencias de la herramienta.	5	5	5	3	3	5	5
			Total	206	204	294	288	298	292

Nota: Vargas, D. (2024). Diseño propio.

Tabla 2

Tabla comparativa de las herramientas para implementar Kubernetes en la nube.

Selección de herramientas para implementar Kubernetes en la nube									
Peso: 1=Baja, 3=Media, 5=Alta Importancia			Rango	2	1	3	5	4	5
Nota : 1=Débil, 3=Promedio, 5=Se adecua fuertemente			Total	300	304	294	276	288	276
Familia/ Criterio			Peso	EKS	GKE	AKS	IBM Cloud	Docks	OKE
1	Funcionalidad								
	1,01	**Interfaz de usuario**: Interfaz de usuario intuitiva y fácil de usar. Se	3	5	5	5	5	3	5
	1,02	**Confiabilidad:** Capacidad de la herramienta de mantener su nivel de ejecución bajo condiciones normales en un período de tiempo establecido	3	5	5	5	5	5	5
	1,03	**Seguridad**: Habilidad de prevenir el acceso no autorizado, sea accidental o premeditado, a los programas y datos.	5	5	5	5	5	5	5
	1,04	**Alcance:** Cantidad de nodos máximos que brinda la herramienta respecto al consumo de recusros, para soportar las cargas de trabajo.	5	5	5	5	5	5	5
	1,05	**Escalabilidad:** Capacidad de la herramienta para manejar un aumento en la carga de trabajo o en el tamaño del clúster.	5	5	5	5	5	5	5
	1,06	**Disponibilidad y tolerancia a fallos:** La herramienta proporciona mecanismos para garantizar la disponibilidad continua y la tolerancia a fallos en el clúster.	5	5	5	5	5	5	5
	1,07	**Personalización:** La capacidad de personalización de la herramienta para cumplir con las características necesarias del clúster.	5	5	5	5	5	5	5
	1,08	**Conectividad y anti-afinidad:** La conectividad de red entre los nodos del clúster y alguna posible limitación entre políticas anti afinidad.	3	5	3	3	3	3	3
	1,09	**Rendimiento:** Capacidad de la herramienta bajo carga pesada en términos de rendimiento.	5	5	5	5	5	5	5
2	Usabilidad								
	2,01	**Fácilidad:** La facilidad de configuración e desarrollo de la herramienta.	3	5	5	5	3	5	3
	2,02	**Documentación:** Documentación clara de la herramienta que facilite su implementación y uso	3	5	5	5	5	5	5
3	Eficiencia								
	3,01	**Gestión de recursos:** La capacidad de optimizar recursos físicos, en cuánto a CPU y memoria es crucial para garantizar el rendimiento adecuado del clúster.	5	5	5	5	5	5	5
	3,02	**Tiempo de implementación:** Tiempo que llevará implementar la herramienta y ponerla en producción.	3	5	5	5	3	5	3
4	Portabilidad y Mantenimiento								
	4,01	**Facilidad de mantenimiento:** Es fácil de mantener y actualizar.	3	5	5	5	3	5	3
5	Perspectiva de uso								
	5,01	**Comunidad y soporte:** Existe comunidad activa en torno a la herramienta y hay soporte del proveedor de la herramienta.	1	5	5	5	5	5	5
	5,02	**Costos:** Costos asociados a la implementación o de licencias de la herramienta.	5	3	5	3	3	3	3
			Total	300	304	294	276	288	276

Nota: Vargas, D. (2024). Diseño propio.

Hablando en términos generales, utilizar una herramienta pública es conveniente en varios casos, tales como:

- Utilizar un modelo de pago por uso que permita pagar solo por los recursos consumidos, además, si la carga de trabajo es variable, la escalabilidad elástica de la nube permite ajustar los recursos según sea necesario sin incurrir en costos innecesarios durante los períodos de baja demanda, esto evitando inversiones iniciales en hardware.
- Para proyectos que requieren una implementación rápida, la nube proporciona la capacidad de aprovisionar y configurar servicios rápidamente.
- Cuando alguna organización no cuenta con los recursos internos necesarios para gestionar y mantener una infraestructura local.

No resulta conveniente utilizar una herramienta pública en casos tales como:

- Si existen requisitos regulatorios que prohíben el almacenamiento de datos en la nube o requieren ciertos controles de seguridad específicos difíciles de cumplir en un entorno de nube pública.
- Si, a largo plazo, los costos en la nube resultan más altos que la inversión inicial en infraestructura local.
- En situaciones en las que la organización requiere un control total sobre su infraestructura, desde el hardware hasta la red, y se requieren gestionar todo internamente.

- Se requiere una personalización específica que no se puede lograr eficientemente en un entorno de nube pública.

Teniendo en cuenta los argumentos, criterios, metricas y tablas expuestas previamente, la solución más eficiente en el contexto de RSI, es K3s, por sus diferentes beneficios, sin dejar atrás de que hay buenas opciones como Kubernetes Operations o Rancher; por su parte, GKE también es una opción importante para implementaciones en la nube. Es pertinente mencionar que la elección de la herramienta a utilizar depente de cada organización y sus objetivos, sin embargo, cualquier persona puede tomar estas consideraciones confiables para decantarse por alguna solución.

6.2.3. Planear la implementación de estas soluciones

La planificacion de la implementación de la solución elegida anteriormente fue fundamental para garantizar el cumplimiento de los objetivos, se tienen los siguientes pasos para asegurar un entorno eficiente, escalable y fácil de gestionar; cada fase fue crítica para garantizar que el clúster cuente con las características necesarias.

A. La implementación del clúster de Kubernetes en K3s comenzó con la creación del clúster en sí. Esto implicó definir la arquitectura y la topología del clúster, incluyendo la distribución de nodos maestros y nodos trabajadores. Fue crucial considerar los requisitos de rendimiento y alta disponibilidad durante esta fase.

B. Una vez definida la estructura del clúster, se procedió a la creación de los nodos maestros y

los nodos trabajadores en el clúster. Se establecieron configuraciones específicas para cada tipo de nodo, asegurándose de que los nodos maestros tengan la capacidad adecuada para gestionar el plano de control, mientras que los nodos trabajadores fueron optimizados para ejecutar cargas de trabajo.

C. La implementación de estrategias de escalado horizontal automático fue esencial para adaptarse a cambios en la carga de trabajo. Aquí se definieron políticas y umbrales para escalar automáticamente el número de instancias de las aplicaciones dependiendo de la demanda.

D. El escalado vertical automático implicó ajustar automáticamente los recursos asignados a una aplicación para garantizar un rendimiento óptimo. Se definieron métricas que permitieron aumentar o disminuir la capacidad de recursos según fuera necesario.

E. Para optimizar la distribución de las cargas de trabajo y mejorar la resiliencia, se implementaron políticas anti-afinidad y se apreció la configuracion de Kube Proxy, ya que este componente se configura por defecto, verificando su funcionalidad para así facilitar la comunicación entre los servicios en el clúster.

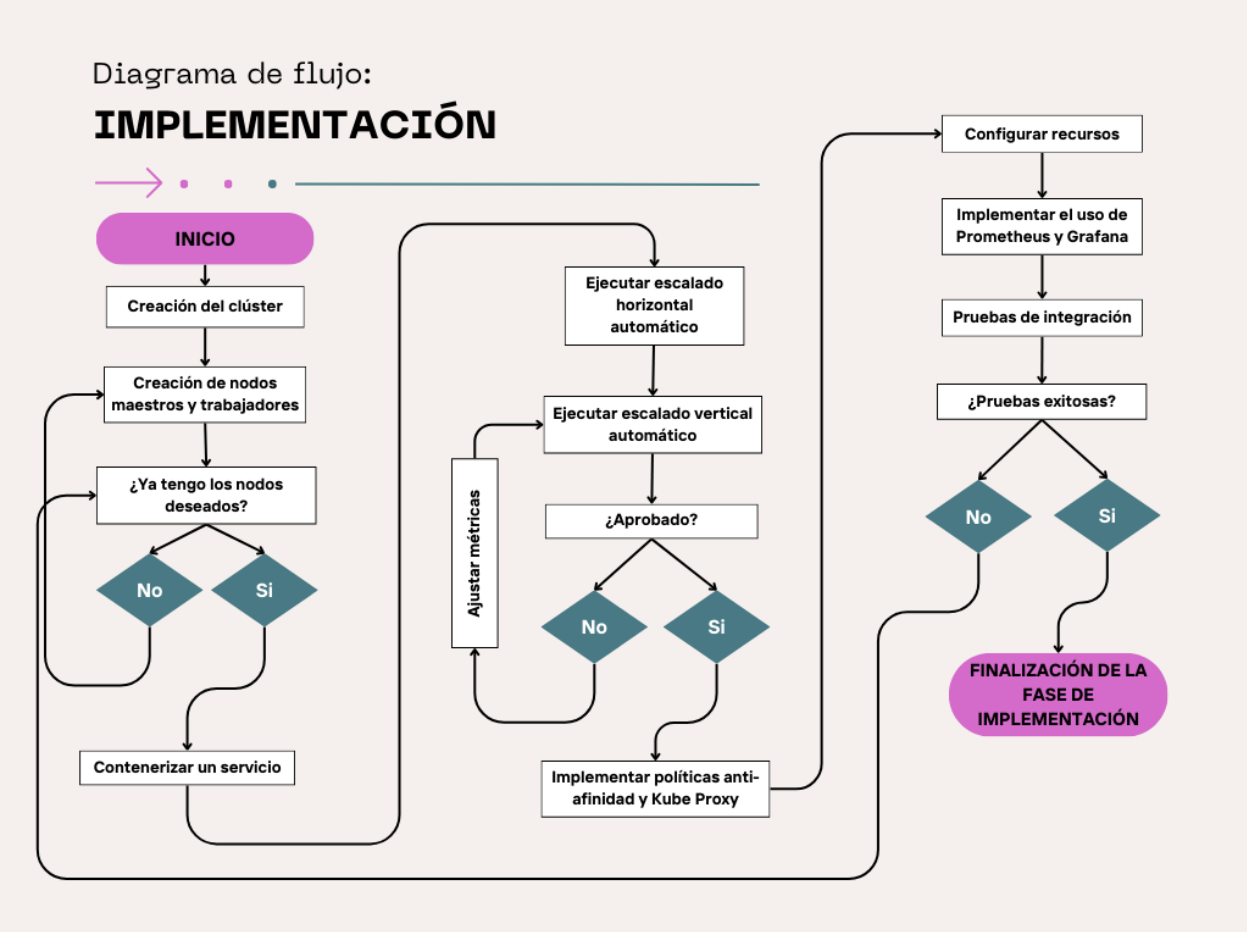
F. La configuración de recursos implicó asignar adecuadamente la capacidad de CPU, memoria y almacenamiento a cada servicio. Esto garantizó que los recursos estén distribuidos de manera eficiente y que cada carga de trabajo tuviera acceso a lo que necesitaba.

G. Finalmente, la implementación de herramientas de supervisión como Prometheus y Grafana permitió monitorear y visualizar el rendimiento del clúster. Se establecieron las métricas

clave a seguir para obtener información valiosa sobre la salud y el estado del sistema; evidentemente fue necesario realizar una prueba de integración al finalizar para garantizar que los componentes funcionen de manera correcta, sin embargo este proceso se dejó en una fase final.

Figura 10.

Diagrama de flujo de la implementación del clúster de Kubernetes.



Nota: Vargas, D. (2024). Diseño propio.

6.3. Implementación del clúster

Se procedió a implementar el plan de investigación diseñado en la fase anterior y se aplicaron las técnicas de análisis correspondientes teniendo en cuenta un enfoque sistemático, riguroso y evolutivo; manteniendo la idea de empezar con la creación de un clúster de kubernetes y a partir de esto agregarle los diferentes componentes necesarios para garantizar la escalabilidad y alta disponibilidad, características primordiales en este proyecto; es pertinente mencionar que se tomó como base una arquitectura de servicios y microservicios, siendo estos backend y frontend, para a partir de esta arquitectura, implementar una infraestructura orquestada. Este enfoque permite una integración fluida entre los servicios existentes y la nueva infraestructura, facilitando la adopción de prácticas modernas de desarrollo y operaciones en el entorno de Kubernetes.

6.3.1. Creación del clúster de Kubernetes con K3s

Para empezar se tuvieron en cuenta los requisitos necesarios de la infraestructura ya que los nodos maestros y trabajadores debían cumplir con ciertos requisitos de hardware, como una cantidad mínima de CPU, memoria y espacio de almacenamiento, como primera instancia se requirió de al menos 2 GB de RAM y 2 núcleos de CPU por nodo.

La creación del clúster fue el paso más sencillo; primero, se conectó a un servidor donde se quería instalar el clúster; luego, se ejecutó el comando para instalarlo; este comando descargó el script de instalación y lo ejecutó. A su vez, también configuró automáticamente la máquina como nodo maestro. Cabe recalcar que se debió verificar que el clúster estaba funcionando. Fue impor-

tante establecer una configuración de red adecuada para garantizar la comunicación sin problemas entre los nodos. Los puertos necesarios para K3s, como el 6443 y el 443, debieron estar abiertos, ya que eran los que necesitaba el entorno para que se comunicaran los nodos. Es pertinente mencionar que, para este caso concreto, se utilizó una base de datos externa, ya que esto traía consigo diferentes ventajas en cuanto a los requisitos de persistencia, escalabilidad, resiliencia y seguridad de las aplicaciones. Si bien agregó complejidad al diseño e implementación, también proporcionó beneficios significativos en términos de gestión de datos y operaciones.

Finalmente, se habilitaron complementos como HELM, el cual facilitó la instalación y gestión de aplicaciones. En este caso, para instalar NGINX actuando como un servidor proxy inverso que redirigió las solicitudes del cliente a los servidores de aplicación. Esto fue útil para equilibrar la carga y gestionar el tráfico entre varios servidores de aplicaciones. También se implementó un balanceador de carga para distribuir el tráfico de red entre los nodos del clúster, mejorando así la disponibilidad y la escalabilidad. Información adicional en el Apéndice A.

6.3.2. Creación de los nodos maestros y los nodos trabajadores en el clúster

La creación de nodos maestros y trabajadores fue un paso fundamental en la configuración de un clúster de Kubernetes; para entender mejor este paso, se agregaron diferentes nodos al clúster de Kubernetes, haciendo referencia a la frase çreación de los nodos maestros y los nodos trabajadores". Es pertinente mencionar que primero se configuraron las máquinas virtuales o servidores físicos que actuaron como nodos maestros y nodos trabajadores; luego, se instaló K3s en cada uno

de ellos y finalmente se configuraron para ser agregados al clúster.

Por lo general, en clúster de Kubernetes standard solo hay un nodo maestro ya que es suficiente para la mayoría de los casos y simplifica la gestión y la toma de decisiones del clúster, sin embargo, en algunos casos, hay múltiples nodos maestros para mejorar la resiliencia y la disponibilidad del control del clúster haciendo que haya una mayor complejidad en la configuración del clúster.

Teniendo en cuenta lo anterior, primero se seleccionaron los servidores destinados a actuar como nodos maestros, considerando cuidadosamente los requisitos de hardware, como la cantidad de CPU y memoria necesaria para soportar las operaciones de gestión del clúster; posteriormente, el proceso de instalación de K3s se inició mediante la ejecución de comandos específicos que descargaron e implementaron la herramienta.

En paralelo, la creación de nodos trabajadores requirió una selección cuidadosa de servidores destinados a ejecutar las aplicaciones y servicios del clúster. Siguiendo el mismo proceso de instalación, estos nodos trabajadores se conectaron al clúster, con el uso de un token de unión, el hash del certificado CA y la dirección IP y el puerto de API del nodo maestro; completando así la infraestructura necesaria para alojar y ejecutar las cargas de trabajo.

Finalmente, fue importante verificar la correcta incorporación de los nodos trabajadores al clúster para garantizar una infraestructura coherente y lista para responder a las demandas de las aplicaciones y los servicios web. Información adicional en el Apéndice A.

6.3.3. Contenerización de servicios

La contenerización de servicios fue un componente esencial en la implementación de un clúster de Kubernetes, permitiendo la encapsulación y ejecución de aplicaciones en contenedores. Este proceso fue necesario para llevar a cabo una prueba piloto de implementación de la orquestación empleando los criterios y métricas establecidos para evaluar la viabilidad del clúster.

Al abordar la contenerización de servicios en un clúster K3s, fue fundamental elegir imágenes de contenedores que representaran adecuadamente las aplicaciones y servicios deseados. Las imágenes contenían todos los elementos necesarios para ejecutar una aplicación, desde el código fuente hasta las bibliotecas y las dependencias. K3s aprovechó tecnologías como Docker para facilitar la creación y distribución de estas imágenes, garantizando una implementación consistente y eficiente en todo el clúster.

Este proceso utilizó manifestos YAML para describir la configuración y los recursos necesarios para implementar y mantener contenedores. Esto proporcionó una forma declarativa de definir el estado deseado de las aplicaciones, facilitando la administración y escalabilidad.

En este caso, se creó un Dockerfile para cada servicio backend utilizando una imagen base con Java y Maven, y frontend utilizando una imagen base con Node.js; luego, por medio de un archivo .yaml y las configuraciones de este, se desplegaron estos servicios; cabe recalcar que se debió configurar el Ingress Controller, al ser el encargado de exponer estos servicios y manejar el enrutamiento del tráfico hacia aplicaciones desplegadas en el clúster. Por eso, fue importante, para

finalizar, revisar el funcionamiento correcto de estos servicios mediante la dirección IP y el puerto o el nombre del dominio configurado previamente con este componente. Información adicional en el Apéndice A.

6.3.4. Ejecución del escalado horizontal automático

Este proceso permitió ajustar automáticamente el número de instancias de una aplicación para adaptarse a la demanda de tráfico, garantizando así un rendimiento óptimo y una utilización eficiente de los recursos disponibles.

El enfoque principal residió en definir reglas y políticas que determinaran cuándo y cómo escalar la aplicación en función de métricas específicas, como la utilización de CPU o la carga de red. Estas reglas se establecieron mediante el uso de HPA, que definieron umbrales y comportamientos de escalamiento; además, la adaptabilidad y flexibilidad de K3s facilitaron la modificación de estas configuraciones en tiempo real para ajustar el número de replicas de una aplicación en demanda del tráfico y recursos disponibles. Información adicional en el Apéndice A.

6.3.5. Ejecución del escalado vertical automático

El escalado vertical automático en un clúster de Kubernetes fue esencial para optimizar el rendimiento de las aplicaciones. El escalado vertical implicó ajustar los recursos de cada instancia, como la capacidad de CPU y memoria, para adaptarse a los requisitos cambiantes; esta funcionalidad se basó en la modificación dinámica de los recursos asignados a los contenedores de una aplicación en función de las métricas de rendimiento y la demanda actual. La configuración del

VPA en el clúster de Kubernetes también implicó definir políticas y umbrales para las métricas. Información adicional en el Apéndice A.

6.3.6. Implementación de políticas anti-afinidad y Kube Proxy

La implementación de políticas anti-afinidad en el clúster de Kubernetes fue fundamental para garantizar la alta disponibilidad y la resistencia a fallos de las aplicaciones. Las políticas anti-afinidad permitieron controlar la colocación de las instancias de una aplicación en diferentes nodos del clúster, evitando la concentración de recursos críticos en un solo lugar; esto aseguró que, en caso de fallo de un nodo, la aplicación pudiera seguir funcionando sin interrupciones, distribuyendo su carga de trabajo entre los nodos restantes; esto se realizó configurando y ejecutando un archivo .yaml.

Por otro lado, la implementación de Kube Proxy en el clúster de Kubernetes fue esencial para facilitar la comunicación y el enrutamiento del tráfico entre las instancias de la aplicación. Kube Proxy actuó como una defensa entre los servicios, gestionando la redirección de las solicitudes hacia las instancias adecuadas. Su implementación optimizó la eficiencia en la comunicación y garantizó que la carga de trabajo se distribuyera uniformemente entre los nodos del clúster; al instalar K3s, se generó un archivo .yaml en el nodo maestro con las configuraciones principales, por lo cual este componente ya estaba configurado, así que simplemente se editó este archivo para agregar las configuraciones específicas. Información adicional en el Apéndice A.

6.3.7. Configuración de recursos del clúster

La configuración de recursos se realizó mediante la definición de límites y solicitudes en los manifiestos de los pods. Estos valores indicaron al orquestador cuántos recursos debía reservar y asignar para cada instancia de la aplicación. Establecer límites evitó que una aplicación utilizara más recursos de los necesarios, mientras que las solicitudes aseguraron que el recurso necesario estuviera disponible cuando se inició la instancia.

Fue esencial evaluar las necesidades específicas de cada aplicación al configurar los recursos en K3s. La configuración adecuada garantizó que el clúster de Kubernetes utilizara eficientemente los recursos disponibles, optimizando la utilización de la infraestructura, siendo indispensable para garantizar la escalabilidad y alta disponibilidad. Información adicional en el Apéndice A.

6.3.8. Implementar Prometheus y Grafana en el clúster

La implementación de Prometheus y Grafana en el clúster de Kubernetes agregó una capa de supervisión y visualización que permitió monitorear y analizar el rendimiento del clúster, así como de las aplicaciones desplegadas, el HPA y el VPA; estas herramientas fueron esenciales para garantizar la salud, la disponibilidad y la eficiencia del sistema; utilizar estas herramientas no solo facilitó la supervisión, sino que también mejoró la capacidad de respuesta frente a eventos inesperados. Información adicional en el Apéndice A.

6.4. Pruebas

Finalmente, se llevaron a cabo pruebas para evaluar la eficacia y robustez de las soluciones propuestas. Llevando a cabo una revisión minuciosa del prototipo final para asegurar que cumple con los objetivos y el alcance definidos, incluyendo la funcionalidad, usabilidad y coherencia con las expectativas del proyecto.

Como parte integral de esta fase, se documentó detalladamente todo el proceso de implementación de kubernetes, incluyendo procedimientos seguidos durante la implementación y las conclusiones derivadas de las pruebas realizadas. siendo esta documentación crucial para garantizar la replicabilidad del proyecto.

Con estas pruebas, no solo se buscó validar la efectividad del clúster de Kubernetes implementado, sino también proporcionar información necesaria para mejorar y optimizar futuras implementaciones basadas en este proyecto. Información adicional en el Apéndice B.

6.4.1. Desarrollar pruebas de integración

La realización de pruebas de integración en el clúster de Kubernetes fue de suma importancia. Este proceso implicó integrar los componentes y funcionalidades del clúster para evaluar cómo respondían de manera conjunta. Durante las pruebas de integración, se monitorearon cuidadosamente métricas clave como el uso de CPU, la utilización de memoria y la latencia de red; para que en caso de haber un comportamiento no deseado, se realizara un ajuste de la configuración

según fuera necesario; en esta fase hubieron dos casos de prueba, los dos casos con el objetivo de evaluar el rendimiento y la escalabilidad de la implementación de la prueba piloto del clúster de Kubernetes.

En el primer caso de prueba, se implementó un clúster de Kubernetes desplegado en una máquina física con 8 GB de RAM y 6 núcleos de CPU utilizando K3s. Se crearon dos nodos, uno maestro y otro trabajador, con 1 GB de RAM y un núcleo de CPU cada uno. Se despliegaron dos servicios web, un backend y un frontend, y se simularon 10 usuarios realizando peticiones durante 30 segundos utilizando la herramienta Siege. Se recopilaron métricas de tiempo de respuesta y uso de recursos del clúster, incluyendo el análisis del uso de memoria RAM y CPU en 10 iteraciones de las pruebas, para evaluar la capacidad de manejar la carga simulada y determinar su rendimiento bajo condiciones de carga.

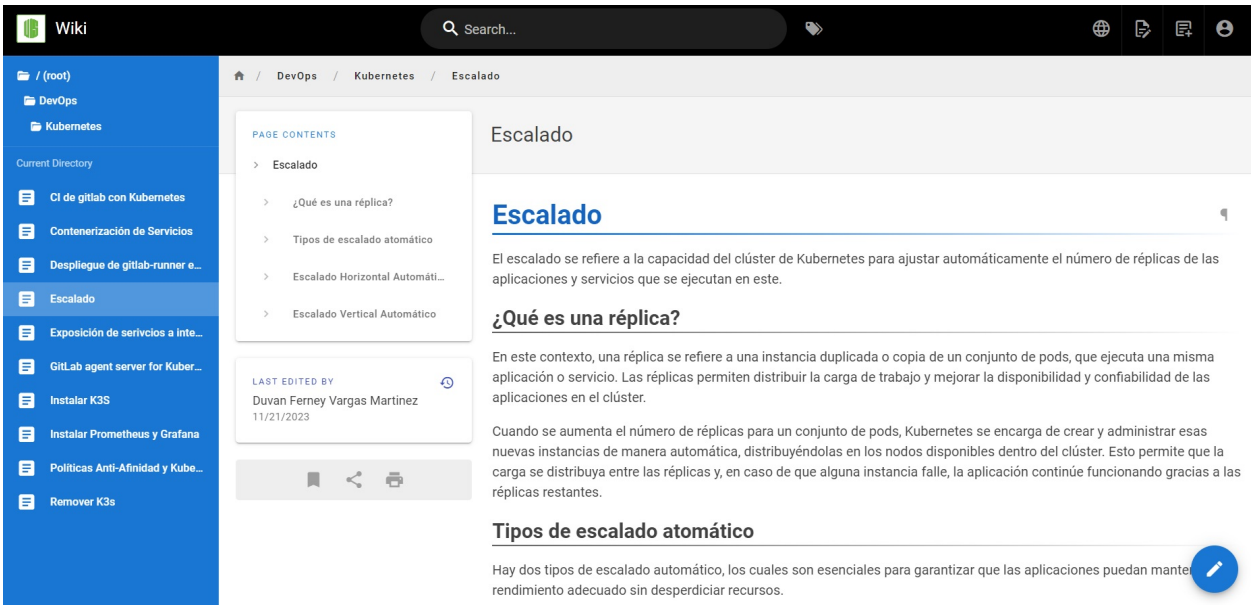
Para el segundo caso de prueba, se implementó un clúster de Kubernetes utilizando 4 máquinas físicas, cada una equipada con 8 GB de RAM y 8 núcleos de CPU. Cada máquina alojaba dos nodos con 4 GB de RAM y 3 núcleos de CPU respectivamente. Se desplegaron diferentes servicios web, tanto backend como frontend, y se llevó a cabo una simulación similar con 10 usuarios realizando peticiones durante 30 segundos mediante la herramienta Siege. Al igual que en el caso anterior, se recopilaron métricas de tiempo de respuesta y uso de recursos del cluster, incluyendo el análisis del uso de memoria RAM y CPU en 10 iteraciones de las pruebas.

6.4.2. Documentación

En este caso concreto la documentación se encuentra en la wiki de RSI, siendo esencial para garantizar una administración eficiente y una comprensión integral de la configuración y operación del entorno. Una documentación completa no solo sirve como guía para los administradores del clúster, sino que también facilita la colaboración entre equipos, mejora la resolución de problemas y permite una transición más suave en el caso de nuevas incorporaciones; esta documentación aborda la arquitectura general del clúster, la disposición de los nodos, la contenerización, la exposición de servicios, el escalado, las políticas anti afinidad y Kube proxy, el uso de Prometheus y Grafana, entre otros; manteniendo una información precisa y actualizada de los procesos mencionados anteriormente.

Figura 11.

Gráfico de la documentación de la implementación de Kubernetes en la Wiki de RSI.



Nota: Vargas, D. (2024). Diseño propio.

7. Conclusiones

Implementar Kubernetes para desarrollar una infraestructura orquestada en RSI, trae consigo multiples ventajas, no solo por el manejo de los contenedores sino también por la escalabilidad y alta disponibilidad que esta herramienta brinda para los servicios web.

En estos procesos de investigación, análisis y planeación, implementación y pruebas, se han explorado diversas herramientas y prácticas clave que impactan directamente en la eficacia y estabilidad de los entornos orquestados.

La elección de K3s como solución para implementar una infraestructura orquestada en el entorno de RSI, ha demostrado ser acertada por su enfoque simplificado y eficiente. La facilidad de instalación y configuración reduce la barrera de entrada, haciendo que la administración del clúster sea más accesible; a su vez, se destaca la importancia de planificar cuidadosamente las estrategias del desarrollo de la implemenatación, brindando así escalabilidad y alta disponibilidad, caracteristicas esenciales buscadas en este proyecto.

La implementación temprana de soluciones de monitoreo y registro, como Prometheus y Grafana, permite una visibilidad constante del rendimiento del clúster. Estas herramientas son esenciales para identificar posibles problemas y optimizar el entorno.

Es indispensable la participación activa y la colaboración entre los integrantes del equipo de Dev/Ops, para que cada uno aporte a desde su experencia con la finalidad de garantizar el éxito en la creación correcta del clúster de Kubernetes.

La implementación de una infraestructura orquestada ágil, segura y eficiente representa un paso significativo hacia el futuro por parte del proyecto de RSI, siendo esto de suma importancia para la calidad de los servicios web de la Universidad Industrial de Santander; por su parte, se sientan las bases para un despliegue Kubernetes con mayores retos enfocados a la entrega de aplicaciones de alta disponibilidad.

Utilizar una infraestructura de contenedores en lugar de una arquitectura basada en máquinas virtuales existente para servicios web en una arquitectura de servicios y microservicios ofrece

varias ventajas ya que los contenedores son portátiles y compartimentados, además, al compartir el sistema operativo del host, los contenedores son más eficientes en términos de recursos en comparación con las máquinas virtuales, lo que permite una mayor densidad de contenedores en un solo servidor físico, siendo esto muy beneficioso.

Finalmente, la automatización del despliegue de servicios web a través de herramientas de orquestación de contenedores como Kubernetes simplifica la administración y el escalamiento de aplicaciones, permitiendo una implementación eficiente de estos, en un entorno de servicios y microservicios.

8. Trabajo Futuro

La implementación y gestión exitosa del clúster Kubernetes en K3s sienta las bases para futuros desarrollos y mejoras continuas de este, enfocando todos los propósitos en la optimización de los recursos y la mejora de la calidad de los servicios web implementados por RSI.

Se podría explorar el ajuste detallado de los recursos asignados a los nodos del clúster, así como la implementación de políticas de escalado automático más sofisticadas.

Agregar pruebas de estrés para evaluar la capacidad de manejar cargas de trabajo intensivas y garantizar un rendimiento óptimo en situaciones de alta demanda. Este proceso implica simular condiciones extremas y evaluar cómo responde el clúster al ser llevado al límite, identificando posibles cuellos de botella y ajustando la configuración para optimizar su rendimiento.

Tras estas pruebas de estrés, es pertinente realizar una comparativa de cómo se desenvuelven los servicios web con la infraestructura actual contrastando con una infraestructura orquestada, para determinar si es realmente conveniente llevar a cabo una migración a esta, teniendo en cuenta diferentes métricas como el uso de recursos y la optimización de estos, garantizando así que el proyecto se desarrolle de manera óptima y se adapte a las necesidades y desafíos que puedan surgir en el futuro.

La seguridad y la conformidad siempre deben ser áreas de enfoque, podría considerarse

realizar una evaluación de seguridad más profunda, implementar controles adicionales y garantizar el cumplimiento continuo con los estándares y las políticas de seguridad.

Finalmente, es viable considerar la implementación de soluciones automatizadas de respaldo de nodos o pods y la realización de ejercicios regulares de recuperación para validar la efectividad de los procesos.

Bibliografía

¿Qué es Cloud Native Computing Foundation (CNCF)? [Accedido el 1 de octubre de 2023]. (2022, septiembre). KeepCoding Bootcamps. https://keepcoding.io/blog/que-es-cloud-native-computing-foundation/

¿Qué es Docker? [Accedido el 1 de octubre de 2023]. (2023). Ibm.com. https://www.ibm.com/mx-es/topics/docker

Amazon EKS - Elastic Kubernetes Service [Accedido el 3 de noviembre de 2023]. (2023). Amazon Web Services. https://aws.amazon.com/es/eks/

Armstrong, A. (2015, julio). *Lanzamiento de Kubernetes 1.0* [Accedido el 1 de octubre de 2023]. Storagereview.com. https://www.storagereview.com/es/news/kubernetes-1-0-released

Arquitectura de Kubernetes - Taller de PAS [Accedido el 3 de noviembre de 2023]. (Accedido el 3 de noviembre de 2023). https://aulasoftwarelibre.github.io/taller-de-pas/Sesion-2/Arquitectura/

Arsys. (2023). *Gestión de Kubernetes con Rancher* [Accedido el 3 de noviembre de 2023]. https://www.arsys.es/blog/rancher-gestion-kubernetes

Azure Kubernetes Service [Accedido el 3 de noviembre de 2023]. (2023). Microsoft Azure. https://azure.microsoft.com/es-es/pricing/details/kubernetes-service/

Bootcamps, K. (2022a, mayo). *¿Qué es Kubelet?* [Accedido el 1 de octubre de 2023]. https://keepcoding.io/blog/que-es-kubelet/

Bootcamps, K. (2022b, mayo). *¿Qué es Kubernetes Namespace?* [Accedido el 1 de octubre de 2023]. https://keepcoding.io/blog/que-es-kubernetes-namespace/

Bosch, T., Valderas Aranda, P. J., & Roldán Martínez, D. (2018). *Microservicios*. RA-MA Editorial.

Burns, B. (2020). *Guía práctica de Kubernetes: Proyectos para crear aplicaciones de éxito con Kubernetes*. Marcombo.

DigitalOcean Kubernetes [Accedido el 3 de noviembre de 2023]. (2023). DigitalOcean. https://try.digitalocean.com/kubernetes-in-minutes/?utm_campaign=amer_brand_kw_en_cpc&utm_adgroup=digitalocean_kubernetes_exact&_keyword=digitalocean%20kubernetes&_device=c&_adposition=&utm_content=conversion&utm_medium=cpc&utm_source=google&gad_source=1&gclid=EAIaIQobChMIqeKB1P_YgwMVmTjUAR3uNQngEAAYASAAEgLDg_D_BwE

Falco. (Accedido el 20 de septiembre de 2023). https://falco.org

Google Kubernetes Engine [Accedido el 3 de noviembre de 2023]. (2023). Google Cloud. https://cloud.google.com/kubernetes-engine?hl=es

IBM Cloud Kubernetes Service [Accedido el 3 de noviembre de 2023]. (2023). IBM Cloud. https://www.ibm.com/products/kubernetes-service#Overview

Installing kubeadm [Accedido el 3 de noviembre de 2023]. (2023). Kubernetes. https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

Invgate. (2023). *Introducción a Ansible* [Accedido el 3 de noviembre de 2023]. https://blog.invgate.com/es/ansible

K3s - Lightweight Kubernetes [Accedido el 3 de noviembre de 2023]. (2023). K3s. https://k3s.io/

KeepCoding. (2023). *¿Qué es Kubespray?* [Accedido el 3 de noviembre de 2023]. https://keepcoding.io/blog/que-es-kubespray/

Kops - Kubernetes Operations [Accedido el 3 de noviembre de 2023]. (2023). Kops. https://kops.sigs.k8s.io/

Kubernetes. (s.f.-a). *kube-scheduler* [Accedido el 2 de octubre de 2023]. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/

Kubernetes. (s.f.-b). *kubelet* [Accedido el 2 de octubre de 2023]. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

Kubernetes. (s.f.-c). *Operating etcd clusters* [Accedido el 2 de octubre de 2023]. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/

Kubernetes. (Accedido el 20 de septiembre de 2023-a). *kube-proxy*. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/

Kubernetes. (Accedido el 20 de septiembre de 2023-b). *Kubernetes (K8s)*. https://kubernetes.io/es/#:~:text=Kubernetes%20(K8s)%20es%20una%20plataforma,una%20f%C3%A1cil%20administraci%C3%B3n%20y%20descubrimiento.

Kubernetes. (Accedido el 20 de septiembre de 2023-c). *Pipeline de métricas de recursos*. https://kubernetes.io/es/docs/tasks/debug-application-cluster/resource-metrics-pipeline/

Kubernetes. (Accedido el 20 de septiembre de 2023-d). *Replicaset*. https://kubernetes.io/es/docs/concepts/workloads/controllers/replicaset/

Kubernetes. (Accedido el 20 de septiembre de 2023-e). *Secrets*. https://kubernetes.io/docs/concepts/configuration/secret/

Kubernetes. (Accedido el 20 de septiembre de 2023-f). *Using RBAC authorization*. https://kubernetes.io/docs/reference/access-authn-authz/rbac/

Kubernetes Authors. (Accedido el 3 de noviembre de 2023). *Kubernetes Overview* [Accedido el 3 de noviembre de 2023]. https://kubernetes.io/docs/concepts/overview/

Kubernetes, C. R. I. (s/f). *Componentes de Kubernetes* [Accedido el 2 de octubre de 2023]. https://kubernetes.io/es/docs/concepts/overview/components/

Kubernetes monitoring. (Accedido el 20 de septiembre de 2023). https://grafana.com/solutions/kubernetes/

KubeVirt.Io [Accedido el 1 de octubre de 2023]. (s/f). https://kubevirt.io

Making Science. (2021). *¿Qué es Docker Swarm?* [Accedido el 3 de noviembre de 2023]. https://www.makingscience.es/blog/que-es-docker-swarm/#:~:text=Docker%20Swarm%20es%20una%20herramienta,servicio%2C%20implementados%20en%20nodos%20trabajadores.

Microsoft. (Accedido el 20 de septiembre de 2023). *Contenedores, imágenes y registros de Docker*. https://learn.microsoft.com/es-es/dotnet/architecture/microservices/container-docker-introduction/docker-containers-images-registries

Microsoft Azure. (2023). *High Availability Kubernetes Clusters on Azure* [Accedido el 3 de noviembre de 2023]. https://learn.microsoft.com/es-es/azure/architecture/example-scenario/hybrid/high-availability-kubernetes

NetApp. (2019). *¿Qué es DevOps y para qué sirve?* [Accedido el 20 de septiembre de 2023]. https://www.netapp.com/es/devops-solutions/what-is-devops/

Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE) [Accedido el 3 de noviembre de 2023]. (2023). Oracle Cloud. https://www.oracle.com/co/cloud/cloud-native/container-engine-kubernetes/

Poniszewska-Marańda, E., & Czechowska, E. (2021). Kubernetes Cluster for Automating Software Production Environment. *Sensors (Basel, Switzerland)*, *21*(5), 1910-. https://doi.org/10.3390/s21051910

Prometheus monitoring. (Accedido el 20 de septiembre de 2023). https://www.dynatrace.com/monitoring/technologies/prometheus-monitoring/

Red Hat. (2011). *OpenShift Container Platform - Prerequisites* [Accedido el 3 de noviembre de 2023]. https://docs.openshift.com/container-platform/3.11/install/prerequisites.html

Red Hat. (Accedido el 20 de septiembre de 2023). *¿Qué es un clúster de Kubernetes?* https://www.redhat.com/es/topics/containers/what-is-a-kubernetes-cluster

Sentrio. (Accedido el 20 de septiembre de 2023). *Introducción a Jenkins: ¿Qué es, para qué sirve y cómo funciona?* https://sentrio.io/blog/que-es-jenkins/

StatDeveloper. (Accedido el 20 de septiembre de 2023). *YAML orientado a Kubernetes*. https://www.statdeveloper.com/yaml-orientado-a-kubernetes/#:~:text=Kubernetes%20utiliza%20archivos%20YAML%20como,%2C%20tipo%2C%20metadatos%20y%20especificaciones.

Swarm Mode Overview [Accedido el 1 de octubre de 2023]. (2023, agosto). Docker Documentation. https://docs.docker.com/engine/swarm/

What is a Virtual Machine? [Accedido el 1 de octubre de 2023]. (2020, marzo). VMware. https://www.vmware.com/es/topics/glossary/content/virtual-machine.html

Apéndices

Apéndice A. Guía de los Componentes de la Fase de Implementación

Creación del clúster de Kubernetes con K3s, con el nodo maestro y los nodos trabajadores

```
Es necesario tener un nodo, el cual sera el nodo maestro.


Primero se deben configurar las maquinas virtuales o servidores fisicos
que actuaran como nodos maestros y nodos trabajadores, luego se debe
instalar K3s en cada uno de ellos.


Para instalar K3s en el nodo maestro:


curl -sfL https://get.k3s.io | sh -


Para obtener el token de conexion para los nodos de trabajo que sera
utilizado mas adelante:


cat /var/lib/rancher/k3s/server/node-token


Luego es necesario unir estos nodos trabajadores al nodo maestro del
cluster:


En cada nodo de trabajo se debe ejecutar el comando:
```

```
curl -sfL https://get.k3s.io | K3S_URL=https://<IP_DEL_NODO_MAESTRO>:6443
K3S_TOKEN=<TOKEN_DEL_NODO_MAESTRO> sh -
```

Para verificar que esto se realizo de manera correcta se utiliza el siguiente comando en el nodo maestro del cluster de Kubernetes:

```
kubectl get nodes
```

El cual muestra tanto el nodo maestro como los nodos trabajadores en el cluster.

Se puede desplegar una aplicacion de prueba, como NGINX, esto se realiza con los siguientes comandos:

```
kubectl create deployment nginx --image=nginx
kubectl expose deployment nginx --port=80 --type=LoadBalancer
```

Para verificar el despliegue de esta aplicacion se puede acceder a la aplicacion de Nginx desde el navegador utilizando la ruta: http://<IP_DEL_NODO_MAESTRO>:<PUERTO_DE_NGINX>.

Contenerización de servicios

```
Para contenerizar servicios backend frontend  en un cluster de Kubernetes de K3s se
    utiliza Dockerfile, es importante mencionar que la contenerizacion de diferentes
    servicios varia dependiendo de estos.


Primero se configura el contenido del Dockerfile para el backend:


Se utiliza una imagen de java
FROM adoptopenjdk:11-jre-hotspot


Se establece el directorio de trabajo en /spring-backend
WORKDIR /spring-backend


Se copia el JAR de la aplicacion Spring Boot al contenedor
COPY target/nombre-de-tu-aplicacion.jar


Se expone el puerto en el que Spring Boot esta escuchando
EXPOSE 8080


Se utiliza el comando para ejecutar la aplicacion
CMD ["java", "-jar", "aplicacion.jar"]


Finalmente se construye y ejecuta el contenedor para el backend:


docker build -t spring-backend-image .
```

```
docker run -p 8080:8080 -d spring-backend-image



Se configura el contenido del Dockerfile para el frontend:


Se usa una imagen base de Node.js

FROM node:14


Se establece el directorio de trabajo en /angular-frontend

WORKDIR /angular-frontend


Se copian los archivos necesarios al contenedor

COPY package*.json ./


Se instalan las dependencias

RUN npm install


Se copia el resto de la aplicacion

COPY . .


Se construye la aplicacion Angular

RUN npm run build


Se expone el puerto en el que el frontend estara disponible

EXPOSE 80
```

```
Se utiliza el comando para ejecutar el servidor web

CMD ["npm", "start"]


Finalmente se construye y ejecuta el contenedor para el frontend:


docker build -t angular-frontend-image .

docker run -p 80:80 -d angular-frontend-image



Se puede verificar este proceso accediendo al backend en http://localhost:8080 y al
    frontend en http://localhost:80.
```

Ejecución del HPA y VPA

```
-HPA


Para empezar, es neceario tener el servidor de metricas (metrics-server) instalado en
     el cluster; primero se debe clonar el repositorio del servidor de metricas:


git clone https://github.com/kubernetes-sigs/metrics-server.git

cd metrics-server


Se debe aplicar la actualizacion
```

```
kubectl apply -f deploy/1.8+/


Finalmente verificamos que este instalado correctamente en nuestro cluster:


kubectl get deployment metrics-server -n kube-system
kubectl get apiservice v1beta1.metrics.k8s.io -o yaml


Posteriormente configuramos el archivo horizontalPodAutoEscaler.yaml; para este caso
    con replicas entre 2 y 10 para mantener la utilizacion promedio de la CPU en 50%.


Este archivo tiene la siguiente configuracion:


apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: myapp-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: myapp-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
```

```
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Se debe aplicar la actualizacion del archivo horizontalPodAutoEscaler.yaml

```
kubectl apply -f horizontalPodAutoEscaler.yaml
```

-VPA:

Se debe configurar el verticalPodAutoescaler.yaml para ejecutar el

```
    VerticalPodAutoescaler:
```

Este archivo tiene la siguiente configuracion:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: my-vpa
spec:
  targetRef:
```

```
    apiVersion: "apps/v1"
    kind:       "Deployment"
    name:       "my-deployment"
  updatePolicy:
    updateMode: "Auto"


Se debe aplicar la actualizacion del archivo verticalPodAutoescaler.yaml


kubectl apply -f verticalPodAutoescaler.yaml
```

Implementación de políticas anti-afinidad y Kube Proxy

```
Para empezar, se configura una politica de antiafinidad para el deployment example-
    deployment basada en el hostname del nodo. Esto asegura que los pods con la
    etiqueta app "example-app" no se programen en el mismo nodo:


apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: example-app
```

```
  template:
    metadata:
      labels:
        app: example-app
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - example-app
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: example-container
        image: example-image:tag
```

Se aplican los cambios del archivo .yaml

```
kubectl apply -f example-app.yaml
```

Tambien se puede verificar que esten funcionando las politicas anti-afinidad

```
kubectl get pods -o wide



Kube-proxy es responsable de manejar la conectividad de red en un cluster de
    Kubernetes. En este caso concreto, al ser un cluster de Kubernetes de K3s, Kube
    Proxy generalmente esta configurado por defecto; Para mostrar la configuracion de
     Kube Proxy se ejecuta el comando:


sudo kubectl get configmap -n kube-system kube-proxy -o yaml


Para ver los logs de Kube Proxy se ejecuta el comando:


kubectl logs -n kube-system -l k8s-app=kube-proxy
```

Configuración de recursos

```
Para configurar recursos se utiliza un archivo YAML que define un Deployment en un
    cluster de Kubernetes con k3s, este es un Deployment con tres replicas,
    configuracion de recursos, y un Service de tipo LoadBalancer para exponer el
    servicio externamente.


apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: mycontainer
        image: miimagen:latest
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "128Mi"
            cpu: "500m"
        ports:
        - containerPort: 80
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: myapp-service
spec:
  selector:
    app: myapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
```

```
    spec:
      containers:
      - name: mycontainer
        image: miimagen:latest
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "128Mi"
            cpu: "500m"
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  selector:
    app: myapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

```
  type: LoadBalancer


Para aplicar los cambios se ejecuta el comando:


kubectl apply -f deployment.yaml
```

Implementación de Prometheus y Grafana

```
Para empezar, es neceario tener HELM en el cluster, este es un gestor de paquetes
    para Kubernetes que facilita la instalacion y gestion de aplicaciones.


Primero se debe instalar en la maquina local


curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/
    get-helm-3
chmod 700 get_helm.sh
./get_helm.sh


Luego se instala en el cluster de Kubernetes


kubectl -n kube-system create sa tiller
kubectl create clusterrolebinding tiller --clusterrole cluster-admin --serviceaccount
    =kube-system:tiller
helm init --service-account tiller
```

```
Luego ya se puede instalar Prometheus y Grafana en el cluster, primero se anaden los
    repositorios de estas herramientas a HELM, luego si se instalan

-Prometheus:

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm install prometheus prometheus-community/prometheus

-Grafana:

helm repo add grafana https://grafana.github.io/helm-charts

helm install grafana grafana/grafana



Es necesario obtener la contrasena de Grafana

kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" |
     base64 --decode ; echo




Para acceder a Grafana, debemos obtener la direccion IP del nodo
```

```
NODE_IP=$(kubectl get node -o jsonpath='{.items[0].status.addresses[0].address}')

Luego obtenemos el puerto

GRAFANA_PORT=$(kubectl get svc grafana -o jsonpath='{.spec.ports[0].nodePort}')



Finalmente para acceder a este componente, se utiliza el comando:

echo "http://${NODE_IP}:${GRAFANA_PORT}"



Es necesario ingresar con el nombre de usuario "admin" y la contrasena obtenida
    anteriormente.
```

Apéndice B. Detalle de la implementación

Documento de la implementación

https://docs.google.com/document/d/1r-VhDJBxxf-uyMQQV2yLUDa8_3CNkj0H/edit?usp=sharing&ouid=110572191820301038464&rtpof=true&sd=true

Repositorio backend público desplegado

https://github.com/duvanfercho0511/Backend-rhlm-ips