
Presentado ante la ilustre Universidad de Los Andes como
requisito final para obtener el Título de Ingeniero de Sistemas



# Unificación y Orquestación en la Nube: Migración de Infraestructura desde Máquinas Virtuales a un clúster de MicroK8S

Por

Br. Ferreira Pace, José Alfredo

Tutor: Dr. Páez Monzón, Gerard

Asesor: Ing. Lugo, Julio

# Unificación y Orquestación en la Nube: Migración de Infraestructura desde Máquinas Virtuales a un clúster de MicroK8S

Br. Ferreira Pace, José Alfredo

Proyecto de grado – Sistemas Computacionales, páginas.

**Resumen**: La plataforma Ignis Gravitas Inc requiere actualizar y optimizar su arquitectura de software. El objetivo principal de desarrollo es mejorar la eficiencia, reducir costos y aumentar la resistencia del sistema, lo que podría tener un impacto significativo en la operación y el rendimiento de la plataforma, a través de la migración de la infraestructura de Elastic Beanstalk a MicroK8s Se realizará un análisis exhaustivo de la infraestructura actual para comprender completamente su funcionamiento y limitaciones. Posteriormente, se diseñará un plan detallado para la migración a MicroK8s incluirá los pasos, los recursos requeridos y un cronograma. Este plan se implementará primero en un entorno de prueba para identificar y resolver cualquier problema potencial antes de realizar la migración en el entorno de producción. Una vez completada la migración, se monitorea el rendimiento del sistema y se harán ajustes según sea necesario para asegurar que se están obteniendo los beneficios esperados. La justificación de este proyecto radica en los beneficios potenciales que la migración a MicroK8s puede aportar a la empresa. La transición puede ofrecer una mayor escalabilidad y flexibilidad, lo que puede facilitar un crecimiento más eficiente y una adaptación más rápida a las demandas cambiantes del negocio. Esto puede conducir a una mayor eficiencia operativa y, potencialmente, a una reducción de costos. La investigación es de tipo experimental cuantitativa y se utilizarán métodos de monitorización del rendimiento del sistema, análisis de costos y pruebas de resistencia para evaluar la eficacia de la migración.



**Palabras clave:** Kubernetes, clúster, infraestructura, virtual, migración, software, microk8s, entorno, eficiencia.

**Unification and Orchestration in the Cloud: Migration of Infrastructure from Virtual Machines to a MicroK8S Cluster**

Br. Ferreira Pace, José Alfredo

Thesis Project – Computational Systems, pages.

**Summary**: The Ignis Gravitas Inc platform requires updating and optimizing its software architecture. The main development objective is to improve efficiency, reduce costs, and increase system resilience, which could significantly impact the platform's operation and performance through the migration of the infrastructure from Elastic Beanstalk to MicroK8s. A thorough analysis of the current infrastructure will be conducted to fully understand its functionality and limitations. Subsequently, a detailed plan for the migration to MicroK8s will be designed, including the steps, required resources, and a timeline. This plan will first be implemented in a test environment to identify and resolve any potential issues before performing the migration in the production environment. Once the migration is complete, system performance will be monitored, and adjustments will be made as necessary to ensure the expected benefits are being realized. The justification for this project lies in the potential benefits that the migration to MicroK8s can bring to the company. The transition can offer greater scalability and flexibility, which can facilitate more efficient growth and quicker adaptation to changing business demands. This can lead to greater operational efficiency and potentially cost reduction. The research is of a quantitative experimental type, and methods of system performance monitoring, cost analysis, and stress testing will be used to evaluate the effectiveness of the migration.

**Keywords:** Kubernetes, cluster, infrastructure, virtual, migration, software, microk8s, environment, efficiency.

# INDICE

## Índice De Figuras

# Índice De Tablas

## Abreviaturas

**AWS** Amazon Web Services

**MicroK8s** Micro kubernetes

**IG** Ignis Gravitas

**DevOps** Conjunto de prácticas de DevOps que combina el desarrollo de software Dev y las operaciones de TI Ops

**DNS** Sistema de nombres de dominio

**IP** Protocolo de Internet

**TI** Tecnología de la información

**LTS** Soporte a largo plazo

**NPM** Administrador de paquetes de nodos

**PoC** Prueba de concepto



**RAM** Memoria de acceso aleatorio

**SSH** Secure Shell

**SSL** Capa de sockets seguros

**VNC** Computación en red virtual

**VPN** Red privada virtual

**YAML** Ain't Markup Language es un lenguaje de serialización de datos fácil de usar.

**Pod** Los Pods son las unidades de computación desplegables más pequeñas que se pueden crear y gestionar en Kubernetes

**Job** trabajo, tarea Programa, o conjunto de programas, a ejecutar por el computador como una unidad

**Worker** Los Web Workers hacen posible ejecutar la operación de un script en un hilo en segundo plano separado de la ejecución el hilo principal de la aplicación web

**Nodo** Un nodo es una máquina de trabajo en Kubernetes, previamente conocida como minion. Un nodo puede ser una máquina virtual o física

**T3** en Amazon se refiere a las instancias T3 de Amazon EC2

**CPU** Unidad central de procesamiento

**vCPU** Versión virtual de un procesador físico

**API** Interfaz de programación de aplicaciones

**EC2** Amazon Elastic Compute Cloud (EC2) es un servicio que proporciona capacidad informática en la nube segura y de tamaño modificable.

**IP2** Es una herramienta que permite detectar IPs de proxy anónimos

**Inc** Incorporated

**ECK** Elastic Cloud on Kubernetes. (Nube elástica en Kubernetes)

**IBM** Cloud Una plataforma de nube empresarial

**CI / CD** integración continua/entrega o despliegue continuo (Continue integrated / continue delivering).



**VM** Máquina virtual

**SO** Sistema operativo

**IaaS** La infraestructura como servicio

**PaaS** La plataforma como servicio

**SaaS** Software como servicio

**UI** Interfaz de usuario

**SQS** Amazon Simple Queue Service. (Amazon Servicio de colas simple).

**IT** Tecnología de información.

**RDS** Servicio de bases de datos relacionales de AWS.

**AMI** Imagen de máquina de Amazon

**AL2** Amazon Linux 2. (Máquina virtual).

**IAM** Identity and Access Management. (Administración de identidades de AWS).

**NPM** Node package mánager. (Manejador de paquetes de Node JS).

**RVM** Ruby versión mánager. (Manejador de versiones de Ruby).

**MVC**: Modelo vista controlador

## Glosario de términos

**Docker:** Tecnología de contenedores que facilita la creación y uso de contenedores Linux.

**Dockerfile:** Archivo de texto que contiene instrucciones para construir una imagen de Docker.

**Nube:** Red global de servidores remotos interconectados que funcionan como un único ecosistema.

**Broker:** Software que conecta a los usuarios con los recursos.

**Proxy reverso:** Servidor que actúa como intermediario entre los clientes y los servidores web.

**Nginx:** Software de servidor web de código abierto.

**Docker Compose:** Herramienta para definir y ejecutar aplicaciones de Docker con múltiples contenedores.

**Kompose:** Herramienta que convierte archivos Docker Compose a orquestadores de contenedores como Kubernetes.

**Elastic Beanstalk:** Servicio para desplegar y escalar aplicaciones y servicios web en AWS.



**ElastiCache:** Servicio totalmente gestionado compatible con Redis OSS.

**Volition:** Plataforma de Ignis Gravitas Inc. para impulsar el emprendimiento y la innovación.

**Monolito:** Arquitectura de software tradicional que se compila como una unidad autónoma e independiente.

**Microservicios:** Estilo de arquitectura y programación que divide una aplicación en módulos independientes y ligeros.

**Amazon Linux:** Servidores Linux en AWS.

**Kernel:** Componente central de un sistema operativo que actúa como interfaz entre el hardware y el software.

**Compilador:** Herramienta que crea un programa al analizar y traducir un lenguaje formal al lenguaje de la computadora.

**Framework**: Es un marco o esquema de trabajo utilizado por programadores para realizar el desarrollo de software.

**Kibana:** Framework visual de Elasticsearch.

**Helm Chart:** Conjunto de archivos que describen los recursos de un clúster de Kubernetes y los empaquetan como una aplicación.

**Snapshots**: Captura de los datos y dispositivos de una máquina virtual o sistema de almacenamiento en un momento específico.

**Microsoft Azure:** Plataforma de computación en la nube pública.

**Docker Swarm:** Herramienta del ecosistema Docker para gestionar un clúster de servidores.

**Apache Mesos:** Kernel de código abierto para la administración de clústeres, desarrollado en la Universidad de California.

**Virtualización:** Tecnología que permite a una computadora compartir sus recursos de hardware con varios entornos digitales separados.

**Manifiesto:e**s un archivo de configuración que describe los recursos y el estado deseado de la aplicación en el clúster.

# I INTRODUCCIÓN

En la actualidad, la infraestructura de las plataformas y páginas web gestionadas en la nube se ha convertido en un estándar de uso para la mayoría de las compañías en todo el mundo. Empresas como AWS, Google y Microsoft (Azure) ofrecen la posibilidad de almacenar y alojar plataformas y páginas web en servidores en la nube a ciertos costos.

Aunque cada una de estas empresas maneja costos diferentes, para muchas compañías es más rentable emplear la nube para gestionar la infraestructura de sus plataformas, en lugar de adquirir los equipos físicos de hardware necesarios para montar sus propios servidores. Esto, claro está, depende del caso de uso.

En mi experiencia como desarrollador de software, tanto en la empresa Ignis Gravitas Inc. durante el periodo de julio de 2020 a julio de 2022, como en otras compañías, he observado que la elección de una arquitectura correcta para una plataforma grande y su gestión son fundamentales. Además, estas decisiones pueden cambiar con el tiempo, dependiendo del crecimiento o escalamiento de la plataforma.



En diversas conversaciones con mi anterior jefe, Julio Lugo, quien desempeñó el rol de CTO en Ignis Gravitas Inc. durante gran parte del tiempo en que trabajé allí, pude observar lo tedioso que resultaba el procedimiento de gestión de la infraestructura de la plataforma de Ignis Gravitas. El manejo de la tecnología del servicio y su configuración generaban inconvenientes, como que la configuración local de la máquina para realizar los despliegues no era tan trivial.

Asimismo, en mis experiencias en otras compañías, el empleo de tecnologías en auge como Kubernetes, cuyo mantenimiento es más flexible y permite diferentes tipos de escalamiento, nos hizo plantearnos si una migración de arquitectura de las plataformas de la empresa podría no solo reducir los costos mensuales de la nube, sino también reducir el costo operacional en términos de recursos humanos.

La infraestructura de TI en la era digital actual exige eficiencia y flexibilidad. En este contexto, el Proyecto de Grado titulado “Unificación y Orquestación en la Nube: Migración de Infraestructura desde Máquinas Virtuales a un clúster de MicroK8s” se presenta como una solución innovadora para las necesidades cambiantes de las organizaciones modernas.

El objetivo principal del trabajo final es migrar la infraestructura existente de máquinas virtuales, específicamente Elastic Beanstalk de Amazon (actual de la plataforma Ignis Gravitas Inc), a un clúster de MicroK8s Elastic Beanstalk ha sido una opción popular para muchas startups y empresas debido a su facilidad de uso y la capacidad de manejar automáticamente aspectos como el monitoreo, el equilibrio de carga, la escalabilidad automática y la planificación de recursos. Sin embargo, con el tiempo, las limitaciones y desafíos asociados con Elastic Beanstalk han llevado a las organizaciones a buscar alternativas más eficientes y flexibles.

Según Mickiewicz, M (2023), "Elastic Beanstalk está diseñado para casos de uso específicos y puede que no sea compatible con todos los requisitos de la aplicación. Es posible que los desarrolladores deban modificar sus aplicaciones para que se ajusten a las limitaciones de la plataforma."

Elastic Beanstalk es un servicio de AWS que funciona bastante bien para ciertos casos de uso, pero cuando las plataformas tienen requisitos más específicos; es fundamental buscar otras alternativas de arquitectura. Debido a los gastos que pueden ocasionar en costos operativos y de mantenimiento (costo de salarios de desarrolladores) en el tiempo.



Además, Mickiewicz, M, acota varios aspectos acerca de las ventajas y desventajas de emplear este tipo de arquitectura según la implementación y el proyecto, las cuales están dispuestas en la tabla 3.

| **Aspecto** | **Ventajas** | **Desventajas** |
|---|---|---|
| Facilidad de uso | Simplifica la implementación y gestión de aplicaciones. | Control limitado sobre la infraestructura subyacente. |
| Automatización | Maneja automáticamente la escalabilidad, balanceo de carga y monitoreo. | La simplicidad puede ocultar la complejidad subyacente. |
| Integración | Se integra bien con otros servicios de AWS. | Bloqueo de proveedores, dificultando la migración a otros servicios en la nube. |

| Aspecto | Ventajas | Desventajas |
|---|---|---|
| Personalización | Permite cierta personalización a través de configuraciones y ajustes. | Personalización restringida comparada con la gestión directa en EC2. |
| Costo | Puede ser más económico para aplicaciones pequeñas y medianas. | Los costos pueden aumentar rápidamente si no se gestionan adecuadamente. |
| Tiempo de implementación | Reduce significativamente el tiempo de implementación. | Puede no ser adecuado para aplicaciones con requisitos específicos o avanzados. |

**Tabla 3:** Tabla comparativa de las ventajas y desventajas de usar Elastic Beanstalk.

En el contexto laboral de Ignis Gravitas Inc., la plataforma ha experimentado un crecimiento significativo desde sus inicios. Como resultado, los requisitos actuales son más específicos y no se adaptan bien a la arquitectura de Elastic Beanstalk. Además, la falta de capacidad de personalización de la arquitectura y el aumento de los costos de mantenimiento de la infraestructura actual son preocupantes. A esto se suman las altas tarifas de salida de datos de AWS.



Rodríguez, P. (2021) señala: “Las tarifas de salida de datos de Amazon han sido polémicas durante mucho tiempo, tanto por su elevado precio como porque muchos clientes afirman que son difíciles de calcular previamente, lo que contribuye a sorpresas desagradables en las facturas a final de mes”.

Ignis Gravitas Inc. gestiona tres dominios web y plataformas. Una de ellas es una red social para startups conocida como Gravitas, que incluye una plataforma API, y el núcleo de la compañía, la plataforma aceleradora de emprendimientos llamada Volition.

Actualmente, estas plataformas operan en un único entorno de producción. Sin embargo, la empresa se encuentra en una fase de expansión y marketing para generar ingresos.

Por ello, para el CEO Gerard Páez Monzón, es esencial cambiar la infraestructura para reducir los costos operacionales y facilitar su mantenimiento a largo plazo.

Volviendo al planteamiento de Mickiewicz, M (2023), si realizamos una tabla comparativa de sus hallazgos en los beneficios y contras de emplear el servicio de AWS de máquinas virtuales EC2, podemos visualizar que se adapta a los requerimientos actuales de la compañía Ignis Gravitas Inc.

| Aspecto | Ventajas | Desventajas |
|---|---|---|
| Control | Ofrece control total sobre la infraestructura y configuración. | Requiere más gestión y mantenimiento manual. |
| Flexibilidad | Permite personalizar completamente el entorno de ejecución. | Puede ser complejo para usuarios sin experiencia técnica. |
| Escalabilidad | Escalabilidad horizontal y vertical según las necesidades. | La escalabilidad puede ser costosa si no se gestiona adecuadamente. |
| Costo | Pago por uso, lo que puede ser económico para cargas de trabajo variables. | Los costos pueden ser impredecibles y aumentar rápidamente. |
| Integración | Se integra bien con otros servicios de AWS. | La integración puede requerir configuraciones adicionales. |
| Rendimiento | Ofrece alto rendimiento y opciones de optimización. | La optimización puede requerir conocimientos avanzados y ajustes continuos. |

**Tabla 4**: Tabla comparativa de ventajas y desventajas de emplear EC2.

Aunque crear una EC2, implica realizar más configuraciones y gestiones manuales, utilizar una EC2 permite flexibilidad, escalabilidad horizontal y vertical, mejoras en el rendimiento. Existen desventajas considerables en los requerimientos de gestión manual, las dificultades técnicas por la complejidad del desarrollo. Además, las instancias de EC2 son parte de una arquitectura que es una arquitectura per se, por lo tanto, son necesarias, pero no suficientes para cubrir la arquitectura de la compañía.

Aquí es donde entra Kubernetes. La migración a Kubernetes ofrece varios beneficios significativos. En primer lugar, Kubernetes permite una entrega continua y un despliegue sin tiempo de inactividad. Esto significa que las nuevas versiones del software pueden ser desplegadas sin interrumpir el servicio existente. En segundo lugar, Kubernetes puede resultar en una reducción de los costos de infraestructura. Al permitir una gestión más eficiente de los recursos, Kubernetes puede ayudar a las organizaciones a ahorrar en costos operativos. Además, Kubernetes facilita la creación de nuevos servicios y ofrece una mayor resistencia en caso de que alguna parte de la infraestructura falle.

Según Doerrfeld, B (2024), “Durante la última década, Kubernetes ha sido la fuerza dominante en la computación nativa de la nube y en el software empresarial en general, ya que tanto proveedores como clientes han optado por ejecutar sus aplicaciones y servicios en clústeres de contenedores en lugar de en niveles de máquinas virtuales”.



Una arquitectura de kubernetes permite escalabilidad y además portabilidad; ya que sin cambios significativos se puede migrar a diferentes entornos de la nube o diferentes plataformas, eso lo hace bastante versátil.

La opinión técnica de Matt Mickiewicz como experto en arquitectura de software, haciendo una comparativa de los beneficios de emplear kubernetes en lugar de Elastic Beanstalk, podemos destacar los siguientes puntos:

| Aspecto | EC2 con MicroK8s | Elastic Beanstalk |
|---|---|---|
| Control | Ofrece control total sobre la infraestructura y configuración de Kubernetes. | Control limitado, ya que Elastic Beanstalk abstrae gran parte de la infraestructura. |
| Flexibilidad | Permite personalizar completamente el entorno de ejecución y la orquestación de contenedores. | Personalización restringida a través de configuraciones y ajustes limitados. |
| Escalabilidad | Escalabilidad horizontal y vertical gestionada por Kubernetes, adaptándose a necesidades específicas. | Escalabilidad automática basada en reglas predefinidas, pero menos flexible. |
| Costo | Puede ser más económico a largo plazo si se gestionan adecuadamente los recursos. | Los costos pueden aumentar rápidamente si no se gestionan adecuadamente. |
| Integración | Se integra bien con otros servicios de AWS y herramientas de Kubernetes. | Integración sencilla con otros servicios de AWS, pero menos flexible fuera del ecosistema AWS. |

| Aspecto | EC2 con MicroK8s | Elastic Beanstalk |
|---|---|---|
| Rendimiento | Ofrece alto rendimiento y opciones de optimización específicas para contenedores. | Buen rendimiento, pero con menos opciones de optimización avanzada. |
| Mantenimiento | Requiere más conocimientos técnicos para la gestión y mantenimiento. | Menor mantenimiento requerido, ya que Elastic Beanstalk maneja gran parte de la infraestructura. |

**Tabla 5**: Tabla comparativa de arquitecturas EC2 con microK8s y Elastic Beanstalk.

Además, tiene una popularidad entre los desarrolladores bastante considerable, porque es un recurso de software libre que tiene una retroalimentación y mantenimiento constante; existen muchas comunidades de software para realizar consultas y una excelente documentación.



Con esta información recopilada, y ya conociendo los requerimientos de la compañía y el alcance de la plataforma; podemos realizar los siguientes análisis de los puntos de interés, en base a: reducción de costos, optimizar el mantenimiento y ofrecer escalabilidad. Lo que se traduce en realizar una comparativa sobre ambos tipos de arquitectura para estudiar la factibilidad de la solución planteada.

**Ventajas de EC2 con MicroK8s sobre Elastic Beanstalk:**

- **Mayor Control y Personalización:** EC2 con MicroK8s permite un control granular sobre la infraestructura y la configuración de Kubernetes, lo que es ideal para aplicaciones con requisitos específicos y avanzados.
- **Flexibilidad en la orquestación de contenedores:** Kubernetes ofrece una orquestación de contenedores robusta y flexible, permitiendo una mejor gestión de aplicaciones distribuidas y microservicios.

- **Escalabilidad Adaptativa**: Kubernetes permite una escalabilidad más adaptativa y precisa, ajustándose mejor a las necesidades cambiantes de la aplicación.
- **Optimización de Costos**: Con una gestión adecuada, EC2 con MicroK8s puede ser más económico a largo plazo, especialmente para aplicaciones de gran escala.
- **Integración con Herramientas de Kubernetes**: La integración con herramientas y servicios de Kubernetes proporciona un ecosistema más amplio y flexible para la gestión de aplicaciones.

**Desventajas de Elastic Beanstalk:**

- **Control Limitad**o: Elastic Beanstalk abstrae gran parte de la infraestructura, lo que limita el control directo sobre los recursos subyacentes.
- **Personalización Restringida**: Aunque permite cierta personalización, no es tan flexible como gestionar directamente los recursos en EC21.
- **Costos Potenciales:** Los costos pueden aumentar rápidamente si no se gestionan adecuadamente los recursos y la escalabilidad automática.



En conclusión, una arquitectura de EC2 con MicroK8s es una mejor alternativa para aplicaciones que requieren un alto grado de control, personalización y flexibilidad. Por otro lado, Elastic Beanstalk es más adecuado para aplicaciones que se benefician de una gestión simplificada y una integración directa con otros servicios de AWS.

Para llevar a cabo este trabajo final, se utilizará un enfoque metodológico ágil, específicamente el enfoque iterativo e incremental. Este enfoque es especialmente adecuado para proyectos de desarrollo de software complejos y con objetivos cambiantes, ya que permite adaptarse a medida que se avanza en el proyecto.

Cada iteración del trabajo final abordará una parte específica del mismo y se llevará a cabo en un plazo corto y fijo, generalmente de 1 a 2 semanas. Esto permitirá obtener rápidamente retroalimentación sobre el resultado y realizar ajustes para mejorar el proceso y los entregables. En última instancia, este enfoque garantizará que el proyecto se desarrolle de manera eficiente y efectiva, cumpliendo con los objetivos establecidos.

En resumen, la migración desde Elastic Beanstalk a Kubernetes no solo es factible, sino también altamente beneficiosa para las organizaciones modernas. A través del uso efectivo del

enfoque iterativo e incremental, este trabajo final tiene como objetivo facilitar esta transición y ayudar a las organizaciones a aprovechar al máximo las ventajas que ofrece Kubernetes.

## 1.1. Planteamiento del problema

La plataforma Ignis Gravitas Inc utiliza actualmente la infraestructura de Elastic Beanstalk, que tiene algunas limitaciones en términos de eficiencia, costos y resistencia. La migración a MicroK8s puede ayudar a superar estas limitaciones, pero también presenta algunos desafíos.

Algunas de las preguntas que se plantean en esta investigación categorizadas por objetivos son:

**Analizar la infraestructura actual:**

¿Cuáles son los componentes y servicios que componen la infraestructura actual?

¿Cómo se configuran y administran estos componentes y servicios?

¿Cuáles son los problemas o limitaciones de la infraestructura actual?



**Investigar sobre Kubernetes y MicroK8s**

¿Cuáles son las principales características y ventajas de Kubernetes y MicroK8s

¿Cómo se aplican Kubernetes y MicroK8s a la plataforma de Ignis Gravitas Inc?

¿Cuáles son los desafíos de implementar Kubernetes y MicroK8s en la plataforma de Ignis Gravitas Inc?

**Diseñar el proceso de migración:**

¿Cuáles son los pasos necesarios para la migración?

¿Cuáles son los recursos requeridos para la migración?

¿Cuál es el cronograma para la migración?

**Implementar el proceso de migración en un entorno de prueba:**

¿Cómo se probará el proceso de migración?

¿Cómo se identificarán y resolverán los problemas potenciales?

**Monitorear y optimizar la nueva infraestructura:**

¿Cómo se monitorea el rendimiento del sistema?

¿Cómo se realizarán los ajustes necesarios para optimizar la nueva infraestructura?

## 1.2. Resultados esperados

Los resultados de mi estudio ayudarán a la plataforma Ignis Gravitas Inc a realizar una migración exitosa de la infraestructura de Elastic Beanstalk a MicroK8s Los resultados podrían ayudar a:

- Identificar los costos y el tiempo estimados para la migración.
- Identificar los riesgos asociados.
- Garantizar la continuidad del servicio durante la migración.
- Medir los beneficios de la migración.
- Desarrollar un plan de migración detallado que aborde los desafíos identificados.



## 1.3. Delimitación del problema

Ignis Gravitas Inc es una empresa que ofrece una plataforma para gestionar emprendimientos de forma eficiente y segura. Su plataforma está alojada en Amazon Web Services (AWS), uno de los principales proveedores de servicios en la nube. A pesar de ello, estamos presenciando un auge tecnológico en el que otras plataformas de Amazon, distintas a la actualmente utilizada (Elastic Beanstalk), están demostrando ser beneficiosas tanto en términos de costos como de complejidad de despliegue. Entre estas, MicroK8s destaca por su capacidad para simplificar el despliegue y ofrecer una amplia gama de beneficios y opciones de escalado. Además, presenta la ventaja de poder ser migrado fácilmente a Kubernetes en el futuro, si las necesidades del negocio así lo requieren, evitando una migración costosa en términos de tiempo.

Aspectos para tener en consideración:

- **Alcance de la migración:** El proyecto se centrará exclusivamente en la migración de Elastic Beanstalk a MicroK8s dentro de la plataforma Ignis Gravitas Inc. No se considerarán otras plataformas o servicios.

- **Aspectos de mejora:** El proyecto buscará mejorar específicamente la eficiencia, reducir costos y aumentar la resistencia del sistema. Otros posibles beneficios o mejoras no serán el foco principal de este proyecto.
- **Infraestructura existente:** El proyecto asume que ya existe una infraestructura en Elastic Beanstalk que necesita ser migrada. No se considerará la creación de una nueva infraestructura desde cero.
- **Pruebas y validación:** Una vez completada la migración, se realizarán pruebas para validar que el sistema en MicroK8s funciona como se espera y que se han logrado los objetivos de mejora de eficiencia, reducción de costos y aumento de resistencia.
- **Tiempo y recursos:** El proyecto se llevará a cabo dentro de un marco de tiempo específico y con recursos limitados. Esto puede afectar a qué partes de la infraestructura se pueden migrar y cuánto detalle se puede dedicar a cada parte del proceso.

## 1.4. Objetivos

El objetivo principal de este trabajo final se centra específicamente en la migración de la infraestructura de la plataforma. A continuación, se describen los objetivos específicos y el objetivo general, que juntos conforman la meta final de lograr con éxito la migración.



### 1.4.1 Objetivo general

Migrar la infraestructura de Elastic Beanstalk a MicroK8s en la plataforma Ignis Gravitas Inc para permitir la reducción de costos, flexibilidad de montaje y facilidad de uso.

### 1.4.2 Objetivos específicos

- Analizar detalladamente la infraestructura actual de Elastic Beanstalk en la plataforma Ignis Gravitas Inc para entender su funcionamiento y limitaciones.
- Investigar sobre Kubernetes y MicroK8s y un conocimiento profundo sobre sus características, ventajas y cómo pueden ser aplicadas a la plataforma de Ignis Gravitas Inc.
- Diseñar el proceso de migración para la migración de Elastic Beanstalk a MicroK8s incluyendo los pasos necesarios, los recursos requeridos y un cronograma.

- Implementar el proceso de migración en un entorno de prueba de la plataforma de Ignis Gravitas Inc.
- Monitorear y optimizar la nueva infraestructura al finalizar la migración

**Importancia y descripción de los objetivos:**

- **Analizar la infraestructura actual:** Realizar un análisis detallado de la infraestructura actual de Elastic Beanstalk en la plataforma Ignis Gravitas Inc para entender completamente su funcionamiento y limitaciones.
- **Investigar sobre Kubernetes y MicroK8s** Adquirir un conocimiento profundo sobre Kubernetes y MicroK8s sus características, ventajas y cómo pueden ser aplicadas a la plataforma de Ignis Gravitas Inc.
- **Diseñar el proceso de migración**: Crear un plan detallado para la migración de Elastic Beanstalk a MicroK8s incluyendo los pasos necesarios, los recursos requeridos y un cronograma.
- **Implementar el proceso de migración en un entorno de prueba:** Antes de realizar la migración en el entorno de producción, es importante probar el proceso en un entorno controlado para identificar y resolver cualquier problema potencial.
- **Monitorear y optimizar la nueva infraestructura:** Después de la migración, es importante monitorear el rendimiento del sistema y hacer ajustes según sea necesario para asegurar que se están obteniendo los beneficios esperados.



## 1.5. Ámbito de la aplicación

El estudio que se propone la transición de la infraestructura de Elastic Beanstalk a MicroK8s se enfocará en varios aspectos esenciales. Primero, se concentrará en la infraestructura tecnológica, con un enfoque particular en la migración de servicios y aplicaciones de Elastic Beanstalk a MicroK8s. Este proceso implica la implementación de aplicaciones y la orquestación de servicios en MicroK8s, lo que implica la gestión y coordinación de servicios a gran escala.

Además, el estudio buscará optimizar el uso de recursos, tanto en términos de costos como de eficiencia operativa. La transición a MicroK8s puede potenciar la escalabilidad y flexibilidad del sistema, permitiendo un crecimiento más eficiente y una adaptación más ágil a

las necesidades cambiantes del negocio. La seguridad será un aspecto fundamental en este estudio, ya que el proceso de migración debe garantizar que los datos y las aplicaciones estén protegidos durante y después del proceso de migración.

Finalmente, pero no menos importante, el estudio requerirá una planificación estratégica meticulosa. Esto es vital para asegurar una transición fluida, minimizar el tiempo de inactividad y garantizar la continuidad del negocio durante el proceso de migración.

## 1.6. Alcances

- **Reducción de la complejidad:** MicroK8s puede simplificar el despliegue y ofrecer una amplia gama de beneficios y opciones de escalado.
- **Mejora en la eficiencia:** La migración puede resultar en una utilización más eficiente de los recursos, como se observó en un caso donde el número de servidores se redujo al 40% de su conteo original después de la migración.
- **Preparación para el futuro:** MicroK8s puede ser migrado fácilmente a Kubernetes en el futuro si las necesidades del negocio así lo requieren.



## 1.7. Límites

- **Tiempo y esfuerzo:** La migración requerirá tiempo y esfuerzos significativos. Se sugiere reservar al menos un mes de trabajo de desarrollo y QA para migrar los entornos de desarrollo a la nueva pila.
- **Dependencias del software:** La migración requerirá encontrar un reemplazo funcional para cualquier software propietario en el que se base el entorno actual.
- **Limitaciones del servicio:** Elastic Beanstalk puede no estar al día con todos los últimos servicios de AWS3, lo que podría afectar la migración.
- **Restricciones regionales:** La migración puede estar limitada por restricciones regionales, ya que no todos los servicios están disponibles en todas las regiones.

## 1.8. Justificación

El objetivo de este trabajo final es migrar la infraestructura tecnológica de Elastic Beanstalk a MicroK8s de la plataforma Ignis Gravitas. Esta migración puede aportar una serie de beneficios significativos para la empresa. A continuación, se presentan los principales aspectos que justifican la realización de este proyecto.

En primer lugar, la transición de Elastic Beanstalk a MicroK8s puede ofrecer una mayor escalabilidad y flexibilidad, lo que puede facilitar un crecimiento más eficiente y una adaptación más rápida a las demandas cambiantes del negocio. Esto puede conducir a una mayor eficiencia operativa y, potencialmente, a una reducción de costos.

La seguridad es un aspecto crucial en cualquier infraestructura tecnológica. La migración a MicroK8s debe realizarse de tal manera que se garantice la protección de los datos y las aplicaciones durante y después del proceso de migración. Este proyecto requerirá una planificación estratégica meticulosa para asegurar una transición fluida y minimizar el tiempo de inactividad. Esto es fundamental para mantener la continuidad del negocio durante el proceso de migración y evitar posibles pérdidas o inconvenientes para los clientes y los usuarios.



En conclusión, este proyecto se justifica por los beneficios potenciales en términos de escalabilidad, flexibilidad, seguridad y eficiencia operativa que puede proporcionar la migración a MicroK8s. Sin embargo, también se debe tener en cuenta los desafíos y riesgos que implica esta migración y tomar las medidas necesarias para mitigarlos.

## 1.9. Antecedentes

Como primer antecedente, un proyecto titulado "*Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS*", es un estudio de caso en la industria del petróleo y gas, realizado por Ali Khajeh-Hosseini, David Greenwood y Ian Sommerville del "*Cloud Computing Co-laboratory School of Computer Science University of St Andrews, UK".*

El estudio se realizó en una empresa de la industria del petróleo y gas que estaba considerando migrar su sistema de TI de un centro de datos interno a Amazon EC2.

El objetivo del estudio es ilustrar los beneficios y riesgos potenciales asociados con la migración de un sistema de TI a la nube desde una variedad de perspectivas de los interesados en toda la empresa**.**

El estudio utiliza un análisis financiero y técnico para evaluar los costos y beneficios de la migración a la nube. También se realiza un análisis de impacto en los interesados para identificar los riesgos asociados con la migración.

Los resultados muestran que la infraestructura del sistema habría costado un 37% menos durante 5 años en EC2 y que el uso de la computación en la nube podría haber eliminado potencialmente el 21% de las llamadas de soporte para este sistema. Sin embargo, el análisis de impacto en los interesados reveló que existen riesgos significativos asociados con esta migración. Aunque los beneficios del uso de la nube son atractivos, los autores argumentan que es importante que los responsables de la toma de decisiones en las empresas consideren las implicaciones generales para la organización de los cambios provocados por la computación en la nube.

Como un segundo antecedente de investigación una tesis de pregrado de la facultad de Ingeniería, Escuela Técnica de telecomunicaciones de Barcelona de la Universidad Politécnica de Catalunya, escrita por Lluís Baró Cayetano, titulada "*Creation of a Kubernetes Infrastructure*".



El proyecto implica la creación de un clúster de Kubernetes para cubrir las necesidades básicas de una empresa en un entorno de trabajo desde casa. Se desarrollarán tres escenarios de clúster diferentes:

- Uso de Docker Compose para implementar diferentes aplicaciones.
- Uso de MicroK8s para implementar las mismas aplicaciones.
- El clúster de Kubernetes estará anidado en contenedores LXC.

El proyecto ofrece una forma simple, potente y rápida de implementar aplicaciones en un servidor. Estas aplicaciones están en un entorno seguro, son fáciles de administrar, fáciles de configurar, escalables, económicas y utilizan una metodología de código abierto. El objetivo principal del proyecto es aprender sobre las tecnologías DevOps y comprender su importancia, así como desarrollar el proyecto utilizando una ideología de código abierto. Esto significa que todas las soluciones optimizadas se basarán en el intercambio de conocimientos comunitarios.

El propósito personal del proyecto es aprender sobre las tecnologías DevOps y comprender su importancia. También desarrollar el proyecto utilizando una ideología de

código abierto, lo que significa que todas las soluciones optimizadas se basarán en el intercambio de conocimientos comunitarios. El objetivo principal del proyecto es la creación de un clúster de Kubernetes que cubra las necesidades básicas de una empresa.

Esta arquitectura ha sido diseñada con el estándar de desarrollo de micro kubernetes, por lo cual, la metodología de trabajo para cualquier desarrollador con relativa experiencia en las herramientas utilizadas puede trabajar sobre ella fácilmente.

Como tercer y último antecedente de investigación, la tesis de pregrado de la facultad de Ingeniería, Escuela Técnica de telecomunicaciones de Barcelona de la Universidad Politécnica de Catalunya, titulada "*Analysis And Practice Of Micro-Services Deployment Technologies based on Containers*", escrita por Jesús López Pocino.

El proyecto se basa en el método de desarrollo DevOps y utiliza Docker y Kubernetes como herramientas principales para la gestión de contenedores. Se representará el viaje desde la antigua forma de desarrollo y concepción de arquitecturas hasta la manera actual usando contenedores. Se producirán algunas configuraciones para representar diferentes tamaños de proyectos y especificaciones.



El proyecto ofrece un análisis y evaluación de las tecnologías de contenedores disponibles que serán clave para elegir la mejor solución. Se examinarán los componentes y complementos detrás de estas tecnologías analizando el estado actual del arte.

Se producirán algunas configuraciones para representar diferentes tamaños de proyectos y especificaciones, para después analizar el producto obtenido y sus resultados.

El propósito principal es analizar y estudiar las tecnologías de contenedores y entender cómo impactan y mejoran el desarrollo y mantenimiento del proyecto. Además, se quiere exponer todas las ventajas de usar estas tecnologías.

Como parte de las especificaciones de esta investigación, fue crucial el uso de Docker y Kubernetes tienen amplia compatibilidad con los sistemas operativos más utilizados, Windows, Ubuntu y macOS. Por lo tanto, los experimentos y estudios realizados son compatibles con cualquiera de estos sistemas operativos, reforzando la característica de amplia compatibilidad. En este caso, para la base del proyecto, se utilizaron máquinas Linux y contenedores LXC para simular los dispositivos que participan en los diferentes montajes.

Este proyecto se basa en el método de desarrollo DevOps (Desarrollo y Operaciones) y utiliza las herramientas de gestión de contenedores Docker y Kubernetes como parte central de la tesis. Ambas tecnologías son de código abierto, y tienen una gran comunidad de usuarios y desarrolladores que crean software en forma de plug-in o herramienta para su uso en conjunto con Docker o Kubernetes.

# II Marco Teórico

## 2.1. Conceptos básicos de infraestructura en la nube

La infraestructura en la nube es un conjunto de servicios de computación, almacenamiento y redes que se proporcionan a través de Internet. Estos servicios se pueden utilizar para crear y ejecutar aplicaciones y servicios de forma rápida y sencilla, sin necesidad de invertir en hardware o software local.

### 2.1.1 Servicios de computación en la nube

"Los servicios de computación en la nube se dividen en tres categorías principales: IaaS (Infrastructure as a Service), PaaS (Platform as a Service) y SaaS (Software as a Service)" (Gartner, 2023, p. 5).

2.1.1.1 **IaaS (Infrastructure as a Service):** "IaaS proporciona acceso a infraestructura computacional, como servidores, almacenamiento y redes. Los proveedores de servicios en la nube ofrecen una amplia gama de recursos IaaS, que van desde servidores virtuales básicos hasta infraestructuras complejas y personalizadas". (Cisco, 2023, p. 7).



2.1.1.2 **PaaS (Platform as a Service):** PaaS proporciona una plataforma para el desarrollo, la implementación y la administración de aplicaciones. Las plataformas PaaS suelen incluir herramientas de desarrollo, servidores, almacenamiento y redes. Esto permite a los desarrolladores crear y ejecutar aplicaciones sin tener que preocuparse por la infraestructura subyacente.

2.1.1.3 **SaaS (Software as a Service):** SaaS proporciona software que se ejecuta en la nube. Los proveedores de servicios en la nube ofrecen una amplia gama de aplicaciones SaaS, que van desde aplicaciones empresariales hasta aplicaciones personales.

## 2.2. Modelos de nube

La computación en la nube ha transformado la manera en que las empresas gestionan y acceden a sus datos y aplicaciones. Los tres modelos principales de servicios en la nube privado, híbrido y público ofrecen cada uno sus propias ventajas y desventajas.

### 2.1.2 Nube pública

Según Gartner (2023, p. 6). La nube pública es un modelo en el que la infraestructura, el software y los servicios se proporcionan a través de Internet por un proveedor de servicios en la nube. Los proveedores de servicios en la nube públicos más conocidos son Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform.

### 2.1.3 Nube privada

> Una nube privada es un entorno de computación en la nube dedicado a una única organización. Todos los recursos se encuentran aislados y bajo el control de una única organización. Los entornos de nube privada son ideales para todo tipo de negocios y proyectos. Es el modelo de despliegue en la nube más común para alojar datos y aplicaciones de misión crítica que requieren un alto nivel de rendimiento, disponibilidad y seguridad. (Amazon, 2023)



La arquitectura de la nube privada y la nube pública requieren tecnologías similares.

## 2.3. La virtualización

La virtualización es una tecnología que extrae capacidades de TI del hardware físico existente. Los usuarios pueden crear máquinas virtuales o unidades de software e interactuar con ellas de la misma manera que con máquinas físicas. El software de virtualización combina recursos de hardware como almacenamiento, memoria y CPU y los asigna a máquinas virtuales cuando se necesita.

## 2.4. Software de gestión

El software de administración es necesario para los administradores para supervisar y administrar su infraestructura de TI de manera centralizada, como las unidades de software. Este programa se utiliza para garantizar el cumplimiento de la seguridad, optimizar la

asignación de recursos y establecer configuraciones uniformes en los servidores y entornos de aplicación.

## 2.5. Sistemas de automatización

Las tareas como las integraciones y el abastecimiento del servidor, que son difíciles y pueden causar errores, se simplifican con la automatización. Para que la administración de la infraestructura en la nube sea más eficiente, las organizaciones que deseen implementar un entorno de nube privada deben proporcionar recursos de automatización.

## 2.6. Nube híbrida:

"La nube híbrida es un modelo que combina la nube pública y la nube privada. Las empresas suelen utilizar la nube híbrida para combinar los beneficios de la escalabilidad y la flexibilidad de la nube pública con el control y la seguridad de la nube privada". (Amazon Web Services, 2023, p. 6)

Cada modelo tiene ventajas. La nube pública permite la escalabilidad y los bajos costos. La nube privada mejora la seguridad y el control de los recursos. Las empresas pueden aprovechar los beneficios de la nube pública y privada según sea necesario porque la nube híbrida combina lo mejor de ambos mundos.



Las tres categorías principales de infraestructura de nube privada son las siguientes:

La nube privada en las instalaciones se implementa en los recursos internos de un centro de datos. Requiere una gran inversión inicial y gastos continuos para comprar, mantener y actualizar los recursos, así como para garantizar la seguridad.

Una nube privada administrada por un tercero es un entorno para un solo inquilino que es completamente administrado por ese tercero. Por ejemplo, una empresa externa podría comprar y mantener la infraestructura de TI de su empresa en su centro de datos, así como brindar asistencia, mantenimiento, actualizaciones y administración remota de sus recursos en la nube privada.

La nube privada virtual es un tipo de nube privada que se puede integrar en una infraestructura de nube pública. Para brindar la comodidad y escalabilidad de los recursos de computación en la nube pública con control y seguridad adicionales, proporciona un entorno

aislado y seguro para ejecutar código, alojar sitios web, almacenar datos y realizar otras tareas que requieren un centro de datos tradicional.

## 2.7. Ventajas de la infraestructura en la nube

La infraestructura en la nube ofrece una serie de ventajas, entre las que se incluyen:

- **Escalabilidad:** La infraestructura en la nube se puede escalar de forma rápida y sencilla para satisfacer las necesidades cambiantes de la demanda. Esto es especialmente útil para las empresas que experimentan picos de tráfico o que necesitan aumentar la capacidad para lanzar nuevos productos o servicios.
- **Flexibilidad:** La infraestructura en la nube permite a las empresas pagar solo por los recursos que necesitan. Esto puede ayudar a las empresas a reducir sus costes de TI.
- **Ahorro de costos: "**La infraestructura en la nube puede ayudar a las empresas a reducir los costes de TI de varias maneras. Por ejemplo, las empresas pueden ahorrar dinero en hardware, software y personal". (Microsoft Azure, 2023, p. 7).
- **Acceso global:** La infraestructura en la nube permite a las empresas acceder a sus aplicaciones y datos desde cualquier lugar del mundo. Esto es especialmente útil para las empresas que tienen una base de clientes global.



## 2.8. Desafíos de la infraestructura en la nube

La infraestructura en la nube también presenta algunos desafíos, entre los que se incluyen:

- **Seguridad:** La seguridad de los datos es un reto importante en la nube. Las empresas deben tomar medidas para proteger sus datos en la nube, como utilizar la autenticación y el cifrado de datos.
- **Gestión:** La gestión de la infraestructura en la nube puede ser compleja. Las empresas deben tener un plan para gestionar sus recursos en la nube.
- **Complejidad**: La nube puede ser compleja de entender y utilizar. Las empresas deben invertir en formación para sus empleados para que puedan aprovechar las ventajas de la nube.

- **Arquitectura de la nube:** La arquitectura de la nube es el diseño de la infraestructura en la nube. Las empresas deben considerar una serie de factores al diseñar su arquitectura en la nube, como las necesidades de escalabilidad, flexibilidad y seguridad.
- **Orquestación de la nube**: La orquestación de la nube es el proceso de automatizar la gestión de la infraestructura en la nube. Las herramientas de orquestación de la nube pueden ayudar a las empresas a ahorrar tiempo y recursos.
- **Seguridad en la nube:** La seguridad en la nube es un reto importante. Las empresas deben tomar medidas para proteger sus datos en la nube, como utilizar la autenticación y el cifrado de datos.
- **Gestión del rendimiento en la nube:** La gestión del rendimiento en la nube es el proceso de garantizar que las aplicaciones y servicios en la nube funcionen de forma óptima. Las empresas deben utilizar herramientas de monitorización para supervisar el rendimiento de sus aplicaciones y servicios en la nube.

Kubernetes es una plataforma de código abierto que automatiza las operaciones de los contenedores de Linux. Fue originalmente desarrollado por Google y ahora es mantenido por la Cloud Native Computing Foundation. Kubernetes tiene una serie de características y ventajas que lo hacen una opción popular para la orquestación de contenedores. (Pons, L. ICM, 2020).



## 2.9. Características de Kubernetes

Según KeepCoding Team (2023), estas son las características más importantes de kubernetes:

- **Almacenamiento:** Kubernetes permite crear almacenamiento persistente para contenedores de aplicaciones, ya sea en la nube o local, según sus necesidades.
- **Autor reparación:** Cuando los nodos cumplen su ciclo de vida, Kubernetes puede reiniciar, reemplazar y reprogramar los contenedores que fallan. Además, se encarga de eliminar los contenedores que no responden a las verificaciones de salud del sistema establecidas por el cliente y no los presenta hasta que estén disponibles y listos para utilizarse con normalidad.
- **Descubrimiento de servicios:** Kubernetes proporciona un sistema de autodescubrimiento entre los contenedores y el enrutamiento. Esto significa que se encarga de exponer

automáticamente los contenedores a Internet o en otros contenedores mediante el uso de un nombre DNS o una dirección IP2.

- **Programación automatizada:** Kubernetes puede colocar automáticamente contenedores según sus requisitos de recursos sin afectar la disponibilidad.
- **Implementaciones y reversiones automatizadas:** Kubernetes tiene la capacidad de realizar despliegues automatizados que permiten la implementación gradual de cambios en la aplicación o la configuración mientras se mantiene un seguimiento de su estado. Si algo sale mal, Kubernetes volverá a hacer lo mismo.

## 2.10. Ventajas de Kubernetes

Según Pons, L. (ICM, 2020), estas son las ventajas de kubernetes:

- Actualizar la configuración y las credenciales de las aplicaciones sin reconstruir su imagen permite una mayor flexibilidad y eficiencia en el manejo de las aplicaciones.
- La disponibilidad y la confiabilidad del sistema se garantizan administrando las cargas de trabajo por lotes y CI, reemplazando los contenedores que fallan.
- La escalabilidad: de las aplicaciones hacia arriba y hacia abajo mediante reglas o manualmente permite que las aplicaciones se adapten a las demandas cambiantes.
- Organizar contenedores en varios hosts permite una mayor eficiencia y utilización de recursos.
- La facilidad de expansión, tanto a nivel lógico como físico, permite que las aplicaciones se expandan y adapten a medida que cambian las necesidades.
- Con el servicio EC2, se puede utilizar una plataforma elástica automatizada de servidor web en un entorno de producción sin depender de un proveedor como AWS, lo que brinda mayor independencia y flexibilidad.
- La capacidad de optimizar la estabilidad de la infraestructura, automatizar el trabajo manual en la construcción y operación de entornos de contenedores y acelerar la provisión de nuevas funciones mejora la eficiencia operativa y reduce el tiempo necesario para implementar nuevas características.

## 2.11. Orquestación de contenedores

"La orquestación de contenedores es un proceso que automatiza y simplifica el suministro, la implementación, las redes, el escalado, la disponibilidad y la gestión del ciclo de vida de los contenedores. Es esencialmente un proceso de tres pasos (o ciclo, cuando forma parte de una línea de trabajo iterativa ágil o DevOps)". (IBM, n-d).

Los contenedores son componentes de aplicaciones ligeros y ejecutables que combinan el código fuente de la aplicación con todas las bibliotecas del sistema operativo (SO) y las dependencias necesarias para ejecutar el código en cualquier entorno. Los contenedores y, más específicamente, los microservicios en contenedores o las funciones sin servidor, se han convertido en las unidades de computación de facto de las aplicaciones modernas nativas de la nube porque son más pequeños, más eficientes en recursos y más portátiles que las máquinas virtuales (VM).

Los contenedores pequeños son fáciles de implementar y administrar manualmente. Pero en la mayoría de las organizaciones, el número de aplicaciones en contenedores crece rápidamente y administrarlas a escala es imposible sin automatización, especialmente en un pipeline DevOps o integración continua/entrega continua (CI/CD).



## 2.12. La orquestación de contenedores ofrece diversos beneficios:

Según Veerash Ayyagari, (Microsoft, n-d), los siguientes son beneficios de la orquestación de contenedores:

- Reduce la complejidad de la gestión de contenedores.
- Mejorar la seguridad al reducir las posibilidades de cometer errores humanos gracias a la automatización.
- Permite la escala automática y el reinicio de clúster y contenedores.
- Ayuda a los equipos de IT a automatizar parte del trabajo y aprovechar los beneficios del uso de contenedores.
- Actualmente, Kubernetes es la plataforma de orquestación de contenedores más popular. La mayoría de los principales proveedores de nube pública, como Amazon Web Services (AWS), Google Cloud Platform, IBM Cloud y Microsoft Azure, ofrecen servicios

gestionados de Kubernetes. Docker Swarm y Apache Mesos son otras herramientas de orquestación de contenedores.

## 2.13. Estrategias de migración: Paso a paso, lift-and-shift, refactorización

Las estrategias de migración a la nube suelen caer en tres categorías principales: Paso a Paso, Lift-and-Shift y Refactorización. Cada una tiene sus propios beneficios y desafíos.

### 2.1.4 Paso a Paso (Staged Migration)

Esta estrategia implica mover partes de una aplicación a la nube de manera gradual. Por lo general, se comienza con los componentes que se beneficiarán más de la migración, como aquellos que necesitan más escalabilidad o disponibilidad. Los datos se mueven en etapas para minimizar el tiempo de inactividad y los riesgos asociados con la migración. Esta estrategia permite a las organizaciones aprender y adaptarse a medida que avanzan, pero puede llevar más tiempo completar la migración.

### 2.1.5 Lift-and-Shift (Rehosting)



En esta estrategia, las aplicaciones se mueven a la nube sin cambios significativos. Esencialmente, se “levantan” de su entorno actual y se “trasladan” a la nube. Esta puede ser la estrategia más rápida y menos arriesgada, pero no aprovecha plenamente las ventajas de la nube, como la escalabilidad automática y los servicios gestionados.

### 2.1.6 Refactorización (Replatforming/Repurchasing)

Esta estrategia implica cambiar el diseño de la aplicación para aprovechar las características específicas de la nube. Esto podría incluir la adopción de servicios gestionados, el uso de funciones sin servidor o la reescritura de partes de la aplicación para mejorar la escalabilidad y el rendimiento. La refactorización puede ofrecer los mayores beneficios a largo plazo, pero también es la más compleja y arriesgada de las tres estrategias.

## 2.14. Proceso de migración de infraestructura

Este proceso deberá ejecutarse siguiendo una serie de pasos, los cuales, puede que, requieran configuraciones adicionales dependiendo de la infraestructura actual del sistema.

## 2.15. Pasos para ejecutar la migración de Elastic Beanstalk a MicroK8s:

- **Evalúa la aplicación:** Antes de comenzar la migración, es importante entender cómo está estructurada la aplicación y cómo interactúa con Elastic Beanstalk. Esto incluye entender las dependencias, los datos almacenados y cualquier otro componente importante.
- **Configura MicroK8s:** Se deberá instalar y configurar MicroK8s en tu entorno. Esto incluirá la configuración de los nodos, el almacenamiento, la red y cualquier otro componente necesario.
- **Crea los contenedores:** Si aún no está configurado, se deberá contenerizar tu aplicación para que pueda ejecutarse en Kubernetes. Esto generalmente implica crear un Dockerfile que defina cómo se construye y se ejecuta tu aplicación.
- **Crea los manifiestos de Kubernetes:** Los manifiestos de Kubernetes definen cómo se despliega y se gestiona tu aplicación en Kubernetes. Esto incluirá la creación de Deployments, Services, Ingresses y cualquier otro recurso de Kubernetes necesario.
- **Desplegar la aplicación en MicroK8s:** Una vez creados los contenedores y manifiestos, se puede desplegar la aplicación en MicroK8s.
- **Probar la aplicación**: Después de desplegar la aplicación, se deberá realizar pruebas exhaustivas para asegurarse de que todo funciona como se espera.
- Migrar los datos.
- **Actualizar las rutas:** Finalmente, se deberá actualizar cualquier ruta o DNS para apuntar a tu nueva aplicación en MicroK8s.



## 2.16. Herramientas y recursos de migración:

Aunque la migración de Elastic Beanstalk a Kubernetes puede ser un proceso difícil, hay algunas herramientas y recursos que pueden ayudar.

Para ejecutar Elasticsearch y Kibana en Kubernetes, Elastic Cloud en Kubernetes (ECK)1 es una opción que facilita la configuración, las actualizaciones, los snapshots, el escalado, la alta disponibilidad, la seguridad y otros aspectos. ECK se basa en el patrón del operador de Kubernetes y amplía las capacidades de orquestación de Kubernetes para soportar la configuración y administración de Elasticsearch y Kibana en Kubernetes.

Además, ECK permite el despliegue de clusters en un solo entorno de Kubernetes y su conexión con clusters que se ejecutan en otros entornos. Esto puede ser útil para optimizar las operaciones de búsqueda y la migración de datos.

“Por otro lado, también puedes considerar el uso de Helm Charts oficiales de Elasticsearch y Kibana para desplegar tus aplicaciones e infraestructura de Kubernetes. “. (Elastic, The search AI Company, n.d.).

## 2.17. Necesidades específicas de la plataforma Ignis Gravitas Inc

Ignis Gravitas Inc es una plataforma de emprendimientos que se encuentra en una etapa de conocimiento y apertura de nuevas funcionalidades en búsqueda de clientes potenciales. En este momento la plataforma no cuenta con un ingreso monetario alto, no existe una cantidad de clientes fija exactamente, por lo cual es crucial ahorrar la mayor cantidad de dinero y espacio posible de los servicios consumidos de la nube de Amazon.

Por otra parte, la escalabilidad que ofrece Kubernetes y micro kubernetes en el futuro en caso de un ingreso más alto de usuarios a la plataforma, permitirá un factor de crecimiento mucho más amplio con un costo menor en el futuro, expandiendo el tamaño de la máquina virtual.



## 2.18. Riesgos asociados con la migración

Según Amazon, (2023). La migración de Elastic Beanstalk a Kubernetes puede presentar varios riesgos y desafíos, incluyendo:

- Incompatibilidad de aplicaciones: no se garantiza que las ramificaciones de plataforma basadas en AL2 sean compatibles con su aplicación actual. Además, debido a las diferencias en el sistema operativo y el entorno de ejecución, la aplicación puede comportarse o funcionar de manera diferente, aunque se implemente correctamente en la nueva versión de la plataforma.
- Diferencias en el sistema operativo y el entorno de ejecución: la AMI de Amazon Linux y Amazon Linux 2 comparten el mismo kernel de Linux, pero difieren en el sistema de inicialización, las versiones de libc, la cadena de herramientas del compilador y una variedad de paquetes.

- “Actualización de versiones particulares de la plataforma de tiempo de ejecución, herramientas de compilación y otras dependencias: Además, las versiones específicas de la plataforma del tiempo de ejecución, las herramientas de compilación y otras dependencias del servicio Elastic Beanstalk se han actualizado”. (Amazon, 2023).
- Necesidad de pruebas exhaustivas: tome su tiempo, pruebe la aplicación en un entorno de desarrollo y realice los ajustes necesarios.
- La implementación azul y verde necesaria para la actualización: cuando Elastic Beanstalk esté listo para la producción, necesitará la implementación azul y verde para realizar la actualización. Esto implica la creación de un nuevo entorno que se base en una ramificación de la plataforma de Amazon Linux 2 y la implementación de su código de aplicación en él. Mientras realiza pruebas y ajustes en el nuevo entorno de producción, el entorno actual permanecerá activo y no se verá afectado.

## 2.19. Análisis de los beneficios de la migración de Elastic Beanstalk a Micro Kubernetes en términos de eficiencia, reducción de costos y aumento de la resistencia del sistema:



La migración de Elastic Beanstalk a Kubernetes puede aumentar la eficiencia, reducir los costos y aumentar la resistencia del sistema.

La eficacia Kubernetes mejora la gestión de contenedores. Kubernetes agrupa los contenedores en "pods", que pueden gestionarse como una sola entidad, a diferencia de Elastic Beanstalk, que se basa en instancias EC2 individuales. Esto mejora el uso de los recursos porque los contenedores dentro de un pod pueden comunicarse y compartir recursos de manera más eficiente.

> Además, Kubernetes ofrece una mayor flexibilidad en la implementación y gestión de aplicaciones. Por ejemplo, permite implementaciones, donde una nueva versión de una aplicación se implementa gradualmente para un pequeño porcentaje de usuarios antes de implementarla para todos los usuarios. Esto puede mejorar la eficiencia al permitir a los equipos de desarrollo probar y solucionar problemas con nuevas versiones de aplicaciones antes de su implementación completa. (StackShare, (octubre, 2023)).

La migración a Kubernetes puede reducir significativamente los costos. Un informe de OYO Tech afirma que la migración de su aplicación monolítica habitual a Kubernetes les permitió reducir significativamente el número de entornos Elastic Beanstalk. Esto resultó en menos recursos infrautilizados o no utilizados, lo que puede ahorrar mucho dinero.

Además, Kubernetes permite el escalado automático de aplicaciones, lo que le permite cambiar automáticamente la cantidad de pods según el tráfico o la carga del sistema. Esto puede conducir a un uso más eficaz de los recursos y en una disminución de los costos.

Kubernetes también puede aumentar la resistencia del sistema. Ofrece características como la auto reparación, donde puede reiniciar automáticamente los contenedores que fallan. También permite la replicación de pods, lo que significa que si un pod falla, otro pod puede asumir su carga de trabajo." (StackShare, (Octubre, 2023) Beanstalk vs Kubernetes, https://stackshare.io/stackups/beanstalk-vs-kubernetes).

## 2.20. Variables

- **Tamaño de la infraestructura:** El tamaño de la infraestructura actual de Elastic Beanstalk puede afectar el tiempo y el costo de la migración.
- **Complejidad de las aplicaciones:** La complejidad de las aplicaciones que se ejecutan en la infraestructura actual de Elastic Beanstalk puede afectar la dificultad de la migración. (Se debe migrar tanto gravitas, como Ignis gravitas Volition).
- **Nivel de integración con los sistemas existentes:** El nivel de integración de la infraestructura actual de Elastic Beanstalk con los sistemas existentes puede afectar la dificultad de la migración.
- **Nivel de automatización:** El nivel de automatización de la infraestructura actual de Elastic Beanstalk puede afectar el tiempo y el costo de la migración.
- **Nivel de conocimiento y experiencia del desarrollador de migración:** El nivel de conocimiento y experiencia del desarrollador de migración puede afectar el éxito de la migración.
- **Tipo de aplicaciones:** El tipo de aplicaciones que se ejecutan en la infraestructura actual de Elastic Beanstalk puede afectar la dificultad de la migración. Por ejemplo, las

aplicaciones web pueden ser más fáciles de migrar que las aplicaciones de bases de datos.

- **Estrategia de migración:** La estrategia de migración elegida puede afectar el tiempo, el costo y la dificultad de la migración.
- **Presencia de recursos internos:** La presencia de recursos internos, como equipos de ingeniería y desarrollo, puede afectar el éxito de la migración.

### 2.21. ¿Cómo estas variables podrían afectar el proyecto de investigación?

- **Tamaño de la infraestructura:** Una infraestructura más grande puede requerir más tiempo y recursos para migrar.
- **Complejidad de las aplicaciones:** Aplicaciones más complejas pueden ser más difíciles de migrar.
- **Nivel de integración con los sistemas existentes:** Una mayor integración con los sistemas existentes puede complicar la migración.
- **Nivel de automatización:** Una mayor automatización puede reducir el tiempo y el costo de la migración.
- **Nivel de conocimiento y experiencia del equipo de migración:** Un equipo con más conocimiento y experiencia puede tener más probabilidades de tener éxito en la migración.
- **Tipo de aplicaciones:** Las aplicaciones web pueden ser más fáciles de migrar que las aplicaciones de bases de datos porque pueden ser más fácilmente empaquetadas y desplegadas en Kubernetes.
- **Estrategia de migración:** Una estrategia de migración gradual puede ser menos arriesgada que una estrategia de migración completa.
- **Presencia de recursos internos:** Los recursos internos pueden ayudar a acelerar la migración y reducir los costos.

## 2.22. Hipótesis

El planteamiento de la hipótesis de esta solución al problema planteado se puede definir como:

- La migración a MicroK8s puede ayudar a mejorar la eficiencia, reducir costos y aumentar la resistencia del sistema.
- Los principales desafíos de la migración son la complejidad de la implementación de Kubernetes y la integración con las aplicaciones existentes.

## 2.23. Conceptos básicos de Micro kubernetes (MicroK8s):

2.23.1.1 **Funcionalidades**: MicroK8s incluye características como DNS, almacenamiento, RBAC (Control de Acceso Basado en Roles) y más, que se pueden habilitar según las necesidades del clúster

2.23.1.2 **Gestión de clúster:** MicroK8s permite agregar nodos al clúster y gestionar la configuración de Kubernetes de manera sencilla

2.23.1.3 **Kompose**: MicroK8s es compatible con Kompose, una herramienta que convierte archivos Docker Compose en manifiestos de Kubernetes. (Kompose, 2024).



2.23.1.4 **Escalabilidad**: “MicroK8s es escalable y puede manejar desde pequeños entornos hasta clústeres de producción de alta disponibilidad”. (MicroK8s, 2024).

2.23.1.5 **Nodo**: Según la documentación de kubernetes (2024), “Un nodo es una máquina de trabajo en Kubernetes, que puede ser una máquina virt ual o física. Cada nodo está gestionado por el componente maestro y contiene los ser vicios necesarios para ejecutar pods”. Los servicios en un nodo incluyen el container runtime, kubelet y el kube-proxy.

2.23.1.6 **Pod**: Un pod es la unidad de computación más pequeña que se puede crear y gestion ar en Kubernetes4. Puede estar compuesto por uno o varios contenedores de Linux q ue comparten recursos como almacenamiento y red. (Kubernetes, 2024).

2.23.1.7 **Ingress**: Según la documentación de kubernetes (2024), "Un Ingress es un objeto de la API de Kubernetes que administra el acceso externo a los servicios en un clúster, típicamente HTTP o HTTPS. Permite mapear el tráfico a diferentes backends basado en reglas definidas".

2.23.1.8 **Ingress Class**: La clase de Ingress especifica cómo los Ingresses deben ser implemen tados por los controladores. Esto permite la implementación de reglas específicas y configuraciones para diferentes controladores de Ingress. (Kubernetes, 2024).

2.23.1.9 **Tipos de Nodos**: En Kubernetes, los nodos pueden ser de dos tipos principales: nod os maestros y nodos trabajadores. Los nodos maestros se encargan de la administraci ón del sistema, mientras que los nodos trabajadores proporcionan las cargas de trabaj o para los contenedores. (Juan García, 2021).

2.23.1.10 **Máquina Virtual**: Una máquina virtual es una emulación de una computadora que proporciona una plataforma para ejecutar sistemas operativos y aplicaciones, si mulando un entorno de hardware físico. (Juan García, 2021).

# III Marco Metodológico

## 3.1. Generalidades

"El marco metodológico es un elemento crucial en la investigación y la planificación de proyectos. Se refiere a la estructura y enfoque metodológico que guía la recopilación y análisis de datos, así como la toma de decisiones en un proyecto o estudio." (González, R, 26 de septiembre del 2023).

Tomando esto en consideración, a continuación, se describe todo el conjunto de técnicas, herramientas y estructuras que conforman este proyecto.

## 3.2. Diseño de la Investigación

El diseño de investigación se define como “los métodos y técnicas elegidos por un investigador para combinarlos de una manera razonablemente lógica para que el problema de la investigación sea manejado de manera eficiente. El diseño es una guía sobre «cómo» llevar a cabo la investigación utilizando una metodología particular.” (Muguira, A, 2023).



## 3.3. Identificación del tipo de investigación

Esta investigación es de tipo aplicada, y se considera que tiene un diseño de investigación experimental. En este tipo de diseño, se manipula una variable independiente (la infraestructura de IT de Ignis Gravitas Inc) y se observa su efecto sobre una o más variables dependientes (en este caso la eficiencia, los costos y la resistencia del sistema).

El objetivo de esta investigación es determinar si las variables dependientes se ven afectadas por un cambio en la variable independiente. En su caso, estás cambiando la infraestructura de Beanstalk Elastic a MicroK8s para ver si esto aumenta la eficiencia, reduce los costos y mejora la resistencia del sistema.

Es importante mencionar que este tipo de diseño de investigación requiere un control de variables riguroso para asegurarse de que los cambios en las variables dependientes sean el resultado de la manipulación de la variable independiente y no de otros factores.

Según Muguira, A (2023), “El diseño de la investigación experimental se utiliza para establecer una relación entre la causa y el efecto de una situación. Es un diseño de investigación donde se observa el efecto causado por la variable independiente sobre la variable dependiente.”

### 3.4. Definición del enfoque de la investigación

“La investigación cuantitativa es un método de investigación que utiliza herramientas de análisis matemático y estadístico para describir, explicar y predecir fenómenos mediante datos numéricos.” Arias, E. R. (2021).

Este enfoque permite la recopilación de datos objetivos y la medición de la frecuencia de un fenómeno observando condiciones reales. Además, los hallazgos de la investigación cuantitativa se pueden aplicar a poblaciones enteras para obtener más información.

### 3.5. Identificación de la investigación

Esta investigación es de tipo cuantitativa. Este enfoque es debido a que la recopilación y análisis de datos serán estadísticos y numéricos; ya que se medirá la eficiencia, los costos y la resistencia del sistema antes y después de la migración para hacer una comparación de las mejoras de la infraestructura del sistema y si estas, tendrán un impacto significativo en la plataforma de Ignis Gravitas Inc.



El enfoque de investigación cuantitativo se define como “un método de investigación que utiliza herramientas de análisis matemático y estadístico para describir, explicar y predecir fenómenos mediante datos numéricos”

### 3.6. Población

> “Una población es un conjunto completo de individuos u objetos que comparten características similares. La población puede comprender una nación o un grupo de personas u objetos con una característica común. Incluye a todo el grupo bien definido sobre el que cualquier investigación quiere extraer conclusiones." Población y muestra de investigación - Definición, proceso, técnicas y fórmulas. (Rivas, Y, 2022)

La "población" en este proyecto de investigación se refiere al conjunto completo de entidades que se están estudiando. La infraestructura de Elastic Beanstalk de la plataforma Ignis Gravitas Inc, que se planea migrar a MicroK8s, constituye la población de este estudio.

Todas las máquinas virtuales, aplicaciones, datos y otros componentes pertinentes de la infraestructura actual están incluidos en esta población. Es importante destacar que, en la investigación cuantitativa, la "población" puede referirse a cualquier conjunto de entidades que se estén estudiando, no necesariamente a las personas.

El objetivo general del proyecto es migrar la infraestructura de Beanstalk Elastic a MicroK8s en la plataforma Ignis Gravitas Inc para mejorar la eficiencia, reducir costos y aumentar la resistencia del sistema, razón por la cual se seleccionó esta población de estudio.

### 3.7. Muestra

"Una muestra es una parte o porción representativa de un grupo poblacional. La muestra siempre debe estar enfocada en la selección de participantes que tengan relevancia para lo que queremos investigar." Rivas, Y, (2022, 20).



La muestra de este estudio incluye componentes específicos de la infraestructura existente, como máquinas virtuales, aplicaciones y datos. La selección de esta muestra se llevará a cabo teniendo en cuenta una variedad de factores. El tamaño total de la población de estudio, los recursos disponibles para la investigación y el objetivo general de la investigación son algunos de estos factores.

El objetivo principal de la investigación es migrar la infraestructura de Elastic Beanstalk a MicroK8s en la plataforma Ignis Gravitas Inc. La migración tiene como objetivo aumentar la eficiencia, disminuir los costos y fortalecer el sistema. Es importante destacar que la plataforma tiene una sola red social y un solo monolito.

Para fines de desarrollo se tomará como muestra a la máquina virtual que almacena el monolito de Ignis Gravitas Inc.

### 3.8. Tipo de muestra

La muestra seleccionada para este estudio pertenece a la categoría de muestra no probabilística intencional. La migración de la infraestructura de Elastic Beanstalk a MicroK8s en la plataforma Ignis Gravitas Inc es el objetivo de la investigación, por lo que se ha tomado esta decisión.

“El muestreo no probabilístico intencional es aquel en el que los elementos seleccionados para la muestra son elegidos por el criterio del investigador.” Parra, A. (2023).

No todos los miembros de la población tienen la misma probabilidad de ser seleccionados para formar parte de la muestra en este tipo de muestreo. No se trata de una muestra al azar porque la selección de individuos se realiza intencionalmente o a propósito.

Debido a su relevancia y representatividad para el objetivo de la investigación, los elementos de la muestra, que pueden incluir datos, máquinas virtuales o aplicaciones, son seleccionados intencionalmente. Para alcanzar los objetivos de la investigación, esta selección estratégica permite un enfoque más dirigido y eficiente.

Es importante destacar que, aunque este tipo de muestra puede no proporcionar una representación precisa de toda la población, es adecuada para este estudio debido a su enfoque específico y dirigido. Los capítulos siguientes de esta tesis discutirán en detalle la validez y aplicabilidad de los resultados obtenidos a través de este tipo de muestra.



### 3.9. Instrumentos para la medición de resultados

Según Hernández, R. (abril, 2014, p.199), un instrumento de medición de resultados es "un proceso de vincular conceptos abstractos con indicadores empíricos, el cual se realiza mediante un plan explícito y organizado para clasificar (y con frecuencia cuantificar) los datos disponibles (los indicadores), en términos del concepto que el investigador tiene en mente."

Este estudio evaluará los resultados de la migración de la infraestructura de Elastic Beanstalk a MicroK8s en la plataforma Ignis Gravitas Inc. utilizando una variedad de instrumentos de medición. Estos programas pueden incluir sistemas de evaluación de la resistencia, herramientas de análisis de costos y software de monitoreo de rendimiento.

El diseño de la investigación incluirá la implementación de la migración y la recopilación y análisis de datos con los instrumentos de medición mencionados anteriormente. Después de

la migración, este diseño permitirá una evaluación efectiva de la eficiencia, los costos y la resistencia del sistema.

Se utilizarán los siguientes instrumentos para medir los resultados de la migración:

- **Monitorización del rendimiento del sistema:** Se utilizarán herramientas de monitorización para recopilar datos sobre el rendimiento del sistema, como el uso de CPU, la memoria y el almacenamiento, así como la latencia de la red.
- **Análisis de costos:** Para determinar cualquier reducción de costos resultante de la migración, se realizará un análisis detallado de los costos relacionados con la ejecución de la infraestructura en Elastic Beanstalk y MicroK8s
- **Pruebas de resistencia:** El sistema se someterá a pruebas de resistencia para evaluar cómo la infraestructura maneja las cargas de trabajo pesadas y las fallas del sistema.

## 3.10. Diseño de instrumento

## 3.11. Prueba de concepto PoC

La prueba de concepto del clúster de micro kubernetes, fue realizada basándose en el estándar de desarrollo de la documentación oficial del kubernetes. Esta prueba de concepto fue planificada y creada bajo la supervisión y correcciones del anterior CTO de la empresa Julio Lugo, quien es un experto en el área de desarrollo de infraestructura de software.



**Se planteó usar:**

- 1 cluster de MicroK8s dentro de una instancia de EC2 de tipo t3 mini o t3 small. (A comprobar de acuerdo con el rendimiento del cluster y los diferentes pods, jobs o workers.
- 1 o 2 Nodos dependiendo de la capacidad.
- 1 pod para cada una de las aplicaciones (1 para Ignis Gravitas volition, otro para la UI de gravitas y otro para la API de gravitas).
- Se reservó espacio para 1 servicio que servirá como worker o Job para correr la cola de mensajes de volition.

### 3.11.1 Consideraciones de memoria y capacidad de las instancias:

- Una instancia de t3 small tiene:

- **CPU**: 2 vCPU (tienen un nivel básico de rendimiento).
- **Son ráfagas de CPU**: Se puede ir más allá del nivel base de este tipo de instancia o una más pequeña si es necesario como una t3 mini.
- **Créditos de CPU**: Las instancias T3 acumulan créditos de CPU cuando la CPU no está en uso. Estos créditos se pueden usar para ráfagas de CPU en el futuro.
- **Memoria RAM**: 2GB.
- Una instancia de t3 mini tiene:
    - **CPU**: vCPU.
    - **Memoria RAM**: 1GB de RAM.
    - Son instancias de bajo costo.
    - **Precios por segundo**: Solo pagas por los recursos que consumes, incluso por fracciones de segundo.
    - **Descuentos por uso reservado**: Puedes ahorrar aún más con los Descuentos por uso reservado si te comprometes a usar una instancia T3 durante un período de tiempo específico.



**Otras características**:

- **AWS Nitro System**: Las instancias T3 se basan en AWS Nitro System, que proporciona un rendimiento y una seguridad mejorados.
- **Eficiencia energética**: Las instancias T3 están diseñadas para ser energéticamente eficientes, lo que te ayuda a ahorrar dinero y reducir tu huella de carbono.

### 3.11.2 Viabilidad de una instancia T3 para un clúster de microK8S

La viabilidad de una instancia T3 para un clúster de microK8S con 4 pods que consumen un máximo de 3 a 4 GB con 2 nodos depende de varios factores:

### 3.11.3 Carga de trabajo

- **Tipo de pods**: Los tipos de pods ejecutados en el clúster pueden afectar el uso de recursos. Los pods que realizan tareas intensivas en CPU o memoria pueden consumir más recursos que los pods más ligeros.

- **Patrones de tráfico**: Los patrones de tráfico del clúster también influyen en el uso de recursos. Si el clúster experimenta picos repentinos de tráfico, es posible que las instancias T3 no sean suficientes.

**Recursos**

- **Tamaño de la instancia**: El tamaño de la instancia T3 elegida afectará la cantidad de recursos disponibles para el clúster.
- **Memoria**: La memoria es un factor crucial, especialmente si los pods consumen mucha memoria.
- **CPU**: La CPU también es importante, particularmente si los pods realizan tareas intensivas en CPU.
- **Costo**
    - **Precio de la instancia**: El precio de la instancia T3 puede variar según el tamaño y la región.
    - **Descuentos por uso reservado**: Puedes ahorrar dinero comprometiéndote a usar una instancia T3 durante un periodo específico.



**Consideraciones adicionales**

- **Escalabilidad**: Las instancias T3 son escalables, permitiendo agregar más nodos al clúster si es necesario.
- **Disponibilidad**: Las instancias T3 están disponibles en varias regiones de AWS.

En general, una instancia T3 puede ser viable para un clúster de microK8S con 4 pods que consumen un máximo de 3 a 4 GB con 2 nodos. Sin embargo, es fundamental evaluar cuidadosamente los factores mencionados antes de tomar una decisión.

### 3.11.4 Opción factible para arquitectura con 4 pods de Kubernetes en dos nodos

La mejor opción para una arquitectura que requiere 4 pods de Kubernetes en dos nodos, en el contexto de un startup en fase inicial con bajo consumo, depende de varios factores:

**Facilidad de uso**

- **MicroK8S**: Ofrece una instalación y configuración más sencilla que Kubernetes. Ideal para startups que buscan comenzar rápidamente.

- **Kubernetes**: Requiere más tiempo y experiencia para configurar y administrar.

**Flexibilidad**

- **MicroK8S**: Menos flexible que Kubernetes, no admite todas las funciones de Kubernetes.
- **Kubernetes**: Ofrece mayor flexibilidad y control sobre la configuración del clúster.

**Escalabilidad**

- **MicroK8S**: Adecuado para pequeñas implementaciones. No es ideal para escalar a grandes clústeres.
- **Kubernetes**: Altamente escalable. Puede manejar grandes clústeres con miles de nodos.

**Costo**

- **MicroK8S**: Gratis.
- **Kubernetes**: Puede incurrir en costos adicionales por la administración del clúster.
- Consideraciones adicionales:
- **Habilidades del equipo**: Si el equipo no tiene experiencia con Kubernetes, MicroK8S puede ser una mejor opción.
- **Futuras necesidades**: Si se espera que la carga de trabajo aumente en el futuro, Kubernetes puede ser la mejor opción a largo plazo.



**En base a los detalles proporcionados:**

- **Plataforma 1**: Ruby on Rails 4 con React.
- **Plataforma 2**: Vue.js con Node.js y API.

Ambas plataformas pueden funcionar con MicroK8S o Kubernetes.

La siguiente información acerca del planteamiento de la arquitectura y sus características y desventajas fue extraído directamente de la documentación de kubernetes y AWS, con la cual se plantea porque MicroK8s sería una mejor opción y que tipo de instancia de máquina virtual funcionara para el nuevo cluster de la infraestructura:

### 3.11.5 Evaluación de opciones: MicroK8S y Kubernetes

Para una arquitectura que requiere 4 pods de Kubernetes en dos nodos, considerando que la plataforma está en período de inicio y no tiene un alto consumo, evaluemos las opciones de MicroK8S y Kubernetes.

### 3.11.6 Ventajas y desventajas de MicroK8S

#### 3.11.6.1 Ventajas

- **Simplicidad**: MicroK8S es la forma más sencilla de consumir Kubernetes, ya que abstrae gran parte de la complejidad en la gestión del ciclo de vida de los clústeres.
- **Automatización**: Su interfaz de usuario de bajo mantenimiento automatiza o simplifica operaciones como la implementación, agrupación y habilitación de servicios auxiliares necesarios para un entorno K8S de calidad de producción.
- **Aislamiento**: Los pods se ejecutan directamente en el host, ofreciendo un nivel adicional de aislamiento y evitando el riesgo de instalar algo en el host que afecte tu red.

#### 3.11.6.2 Desventajas



- **Eficiencia**: Ejecutar VMs no es tan eficiente como tener todas las cargas de trabajo directamente en la máquina anfitriona.
- **Mantenimiento**: Un clúster de varios nodos implica más mantenimiento, ya que tienes más máquinas virtuales para cuidar.

### 3.11.7 Ventajas y desventajas de Kubernetes

#### 3.11.7.1 Ventajas

- **Escalabilidad**: Kubernetes es altamente escalable y se adapta bien a entornos de producción a gran escala.
- **Comunidad y Ecosistema**: Tiene una comunidad activa y un ecosistema maduro con muchas herramientas y soluciones disponibles.
- **Flexibilidad**: Ofrece más opciones de configuración y personalización.

#### 3.11.7.2 Desventajas

- Complejidad inicial: Configurar y mantener un clúster de Kubernetes puede ser más complicado y llevar más tiempo.
- Requisitos de recursos: Kubernetes puede requerir más recursos en comparación con MicroK8S.

### 3.11.8 Recomendación

Dado que estás en una etapa de inicio y no tienes un alto consumo, MicroK8S podría ser la mejor opción. Proporciona una experiencia más simple y rápida para comenzar. Sin embargo, si anticipas un crecimiento significativo en el futuro o necesitas más flexibilidad, considera Kubernetes completo.

**Instancia actual de Volition Worker**

Recordando que la cantidad de CPU y RAM disponible para las aplicaciones puede ser menor que la cantidad total asignada a la instancia, ya que una parte se utiliza para el sistema operativo y otros servicios. Además, las instancias T3 son "ráfagas," lo que significa que pueden consumir más CPU de la asignada durante períodos cortos de tiempo.



Tras revisar en AWS con la herramienta de costos y realizar algunas pruebas, se determinó que una instancia de EC2 con un nodo de tipo 5 large puede hacer todo el trabajo y mantener la carga de los pods en la plataforma, la instancia puede ser una t3 mini. Posteriormente se procedió a la creación de un diagrama de arquitectura para realizar la implementación:

### 3.11.9 Diagrama de arquitectura de MicroK8s para Ignis Gravitas Inc

Considerando 1 Cluster de MicroK8s con una instancia de tipo EC2 t3 mini. Con 1 Nodo y 1 pod para cada una de las plataformas (Volition, Gravitas UI y Gravitas API). Adicional a esto un servicio para un worker para la cola de mensajes.

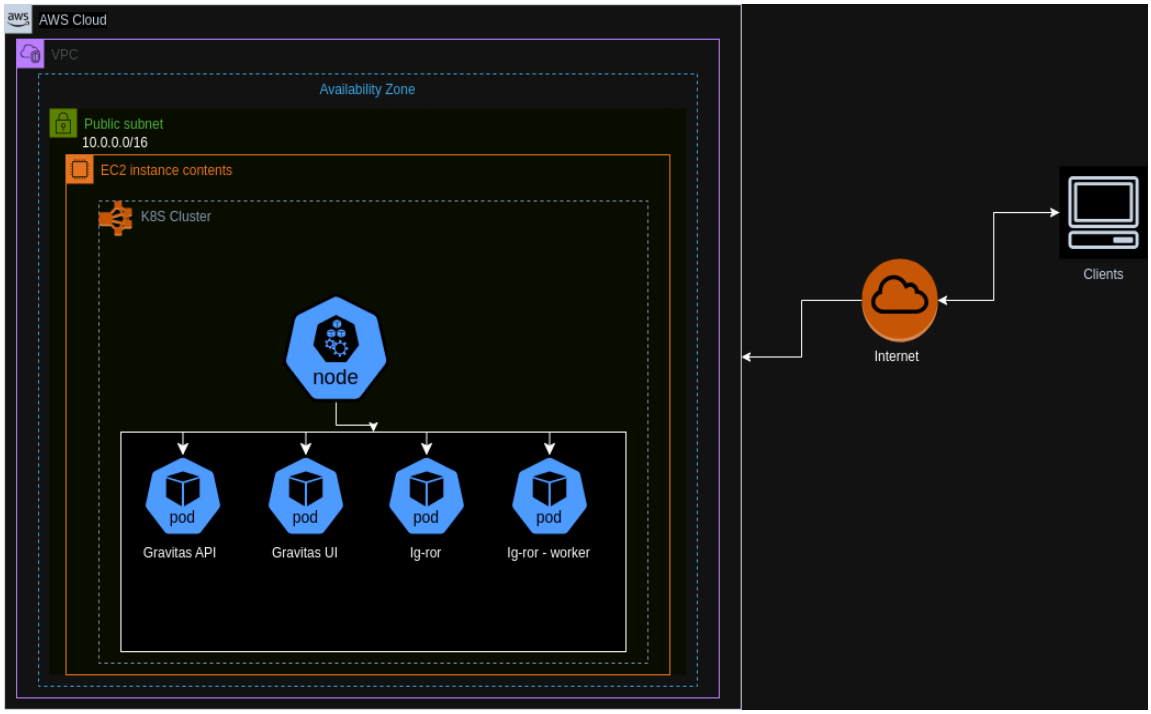


**Figura 14**: Diagrama de arquitectura de Ignis Gravitas Inc para arquitectura de MicroK8s (Modo oscuro)

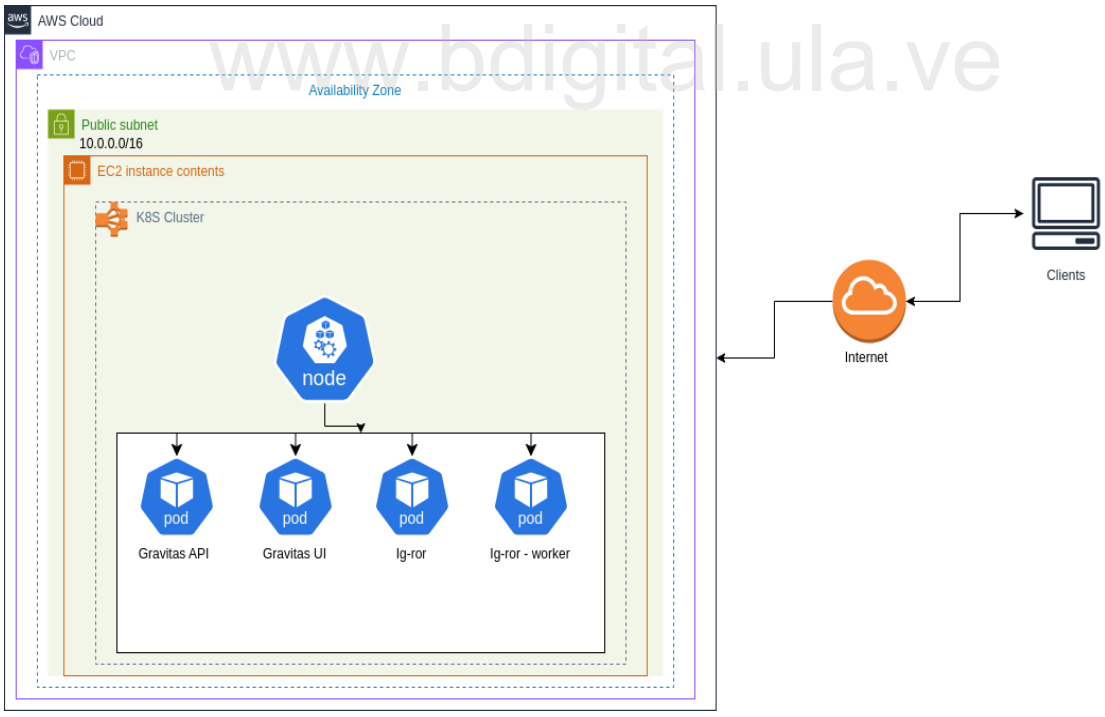


**Figura 15**: Diagrama de arquitectura de Ignis Gravitas Inc para arquitectura de MicroK8s (Modo claro)

Cada uno de estos instrumentos proporcionará datos que pueden ser utilizados para evaluar el éxito de la migración de la infraestructura a MicroK8s.

Gracias a la calculadora de precios (Amazon Pricing Tool), podemos calcular el costo mensual y anuales de las instancias que se requieren para mantener el clúster, ya que el clúster estará dentro de la instancia EC2, no se cancela este servicio, en su lugar, se cancela solo la instancia de la máquina virtual. La siguiente figura nos muestra lo que costaría aproximadamente la instancia de EC2 necesaria para crear el clúster de Microk8s

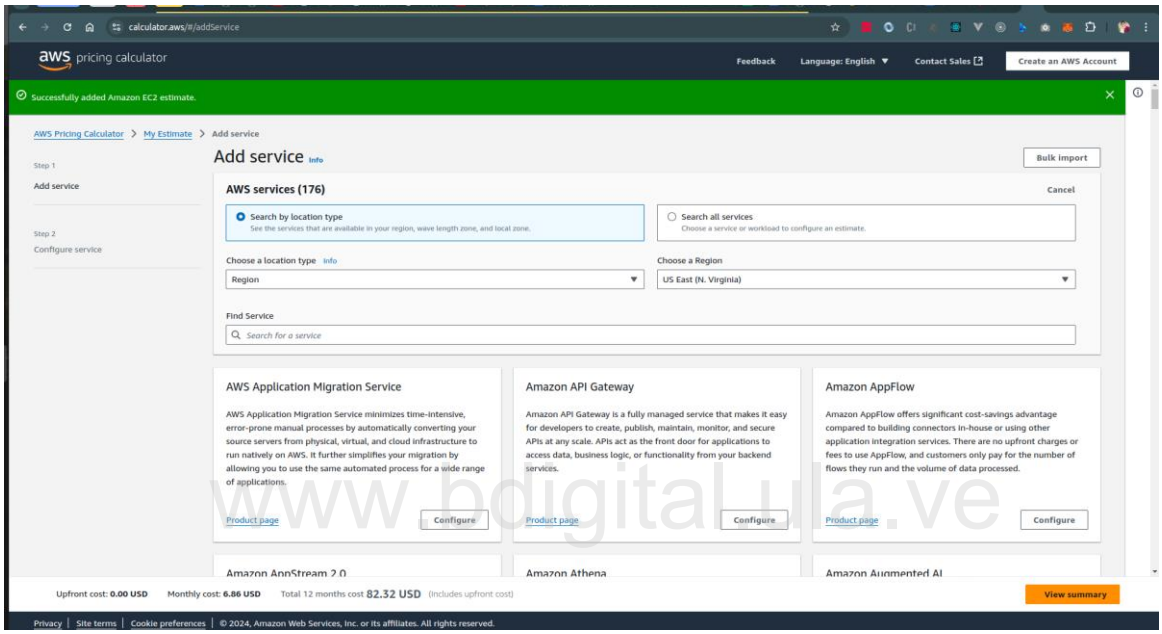


**Figura 7:** Costos para una instancia t3 micro en AWS

Actualmente la facturación de costos de la compañía Ignis gravitas Inc, es de aproximadamente 118$ mensuales promedios (costos variables debido a la fluctuación de entrada y salida de datos por los diferentes servicios empleados). Tal como se pude ver en la siguiente imagen:


**Figura 17:** Costos mensuales de la plataforma de Ignis Gravitas Inc.

El análisis de costos de la plataforma Ignis Gravitas Inc, recuperado de las facturas de la plataforma de AWS respecto al costo con la nueva infraestructura y el ahorro de costo de mantenimiento puede observarse en la tabla siguiente de recuperación de costos de infraestructura (promedios):

| **Periodo** | **AWS Instancias con Elastic Beanstalk** | **AWS instancias con microK8s en EC2** |
|---|---|---|
| 1 mes | 118$ | 82.32$ |
| 1 año | 1416$ | 987.84$ |
| 5 años | 7080$ | 4939.2$ |

**Tabla 6**: Tabla comparativa de recuperación de costos de infraestructura.

Con esta tabla de costos aproximados, podemos asegurar que el costo de infraestructura de AWS de consumo de los servicios de la plataforma de Ignis gravitas Inc. pueden reducirse

hasta en un casi 30% por mes. (Estos costos pueden tener una variabilidad mínima de acuerdo con los costos de entrada y salida de datos de la plataforma).

De igual manera, los costos de la plataforma en mantenimiento operacional con los desarrolladores también se ve significativamente mejorado, el costo de mantenimiento de la arquitectura antigua es bastante alto en comparación, ya que, con el crecimiento actual de la compañía, para los requerimientos actuales y la limitación de personal de la misma, un solo desarrollador de infraestructura puede mantener con mayor facilidad el sistema; lo que se traduce en un ahorro sustancial de recursos humanos.

Mientras que, la arquitectura antigua requiere de un profesional experto en esa tecnología, además de las limitaciones mencionadas en el marco del planteamiento del problema inicial.

### 3.11.10 Consideraciones de las imágenes de docker:

- Las actuales imágenes de docker de cada una de las aplicaciones (volition Ignis Gravitas, Gravitas UI y gravitas API deben ser actualizadas y optimizadas.
- Las nuevas versiones de docker deben ser estándar.
- Es necesario crear un nuevo dockerfile para correr los assets a través de un proxy reverso de nginx.
- Se requiere un volumen compartido entre dos imágenes, para la imagen de rails y nginx. (Cada una con un contenedor)
- El worker de Ignis gravitas no recibe peticiones, sin embargo, requiere una variable de entorno adicional que le identifica si es un worker o no.
- El worker escucha una cola de eventos que son procesados uno a uno (en SQS) con un broker en medio, que manda a la aplicación de Ruby on rails de volition (Ignis Gravitas) los mensajes de notificaciones y correos.
- Se deben crear los Helms y volúmenes persistidos para la base de datos y por servicios.
- Deben correrse las migraciones.
- Debe conectar la conexión de cable en Ignis gravitas y cambiar las configuraciones a modo producción para correr el servidor con los assets a través de la conexión con el proxy reverso de nginx.

### 3.11.11 Consideraciones adicionales de MicroK8s:

- El archivo de docker compose para MicroK8s debe poder ser transcrito a archivos de kubernetes mediante el uso de Kompose. (Según la documentación de Kompose, (2024). Los comandos ejecutados correctamente con el archivo de docker compose válido, generará los archivos YAML necesarios para el clúster (aunque requieran modificarse según cómo evolucione.

La investigación se llevó a cabo de la siguiente manera:

- **Preparación**: Se configurará el entorno MicroK8s y se prepararán todas las aplicaciones y datos para la migración antes de la migración.
- **Migración**: Los datos y las aplicaciones de Elastic Beanstalk se transferirán a MicroK8s
- **Monitorización y recopilación de datos:** Después de la migración, se observará el funcionamiento del sistema y se recopilarán datos utilizando los instrumentos de medición mencionados anteriormente.
- **Análisis de datos:** Después de la migración, los datos recolectados serán analizados para evaluar el rendimiento, los costos y la resistencia del sistema.
- **Informe de resultados:** Se elaborará un informe exhaustivo que explique los hallazgos de la investigación.



## 3.12. Metodología de desarrollo

A nivel metodológico, se propone utilizar una metodología ágil en particular, el enfoque iterativo e incremental. Esta metodología es adecuada para proyectos de desarrollo de software complejos y con objetivos cambiantes, ya que permite adaptarse a medida que se avanza en el proyecto y se comprenden los requisitos y desafíos.

El enfoque iterativo e incremental consiste en dividir el proyecto en pequeñas iteraciones, donde cada iteración produce un incremento funcional de la solución. Cada iteración aborda una parte específica del proyecto y se lleva a cabo en un plazo corto y fijo, generalmente de 1 a 2 semanas. La idea es obtener rápidamente retroalimentación del resultado y realizar ajustes para mejorar el proceso y los entregables.

A continuación, se detalla cómo se aplicará el enfoque iterativo e incremental al proyecto:

### 3.11.12 Procesamiento y codificación de datos

Los datos recopilados por los instrumentos de medición serán procesados utilizando las técnicas de análisis de datos adecuadas. Esto puede incluir análisis de varianza para comparar el rendimiento antes y después de la migración, análisis de costos para evaluar cualquier cambio en los costos operativos y análisis de resistencia para detectar cualquier mejora en la resistencia del sistema.

Es fundamental tener en cuenta que el procesamiento de datos debe llevarse a cabo de manera que se mantenga la integridad de los datos y se minimice la posibilidad de sesgo. Esto puede incluir la implementación de controles de calidad de datos y el uso de software de análisis de datos potente.

Los datos recopilados serán analizados mediante el uso de métodos estadísticos adecuados. El rendimiento del sistema se comparará antes y después de la migración para determinar si la migración ha mejorado el rendimiento del sistema. Los costos de ejecución de infraestructura también se compararon para determinar si la migración ha reducido los costos. Después de la migración, las pruebas de resistencia proporcionarán información sobre la robustez del sistema. El informe final presentará todos estos datos de manera fácil de entender.



## 3.13. Planificación Inicial y Priorización

En la primera etapa, se llevará a cabo una planificación detallada del proyecto, identificando las tareas y objetivos principales. Además, se priorizarán las funcionalidades y servicios a migrar, comenzando con la dockerización de SQS y Elasticache, y luego procediendo a la migración de la aplicación y los datos.

**Iteración 1: Dockerización de SQS y Elasticache**

La primera iteración se centrará en la dockerización de los servicios SQS y Elasticache. Se desarrollarán los Dockerfiles y se realizarán pruebas locales para garantizar el correcto funcionamiento de los contenedores.

**Iteración 2: Pruebas Locales e Integración Inicial**

En esta iteración, se implementarán los servicios dockerizados en el clúster de MicroK8s para realizar pruebas de integración iniciales con la aplicación. Se buscará asegurar que los servicios funcionen correctamente en el nuevo entorno.

**Iteración 3: Dockerización y Migración de Datos**

En esta fase, se continuará con la dockerización de la aplicación y se procederá a la migración de datos desde Elastic Beanstalk a RDS. Se realizarán pruebas para garantizar la integridad de los datos migrados.

**Iteración 4: Implementación en Entorno de Pruebas**

Se configurará un entorno de pruebas separado utilizando MicroK8s y se desplegará la aplicación migrada y los servicios dockerizados. Se llevarán a cabo pruebas de rendimiento y escalabilidad para validar la solución.

**Iteraciones 5 a 8**

**Optimización y Ajustes Finales**: Durante estas iteraciones, se realizarán mejoras continuas en el rendimiento y la estabilidad del sistema. Se ajustará la configuración de Kubernetes y MicroK8s según sea necesario, y se resolverán los problemas detectados en el entorno de pruebas.

**Iteraciones 9 a 10**



Documentación y Entrega Final: En estas últimas iteraciones, se completará la documentación del proyecto, incluyendo guías de instalación, configuración y operación. Se preparará la presentación para la defensa del proyecto de grado y se entregará el trabajo final. (Ver tabla 1).

Al aplicar el enfoque iterativo e incremental, se beneficiará de lo siguiente:

- **Retroalimentación Continua:** Obtendrá retroalimentación temprana y constante sobre el progreso y la calidad del proyecto, lo que permitirá ajustes y mejoras a medida que avanza.
- **Mitigación de Riesgos:** Al dividir el proyecto en pequeñas iteraciones, se pueden identificar y abordar problemas y desafíos antes de que se conviertan en obstáculos mayores.
- **Entregables Funcionales Tempranos:** Cada iteración producirá un incremento funcional de la solución, lo que le permitirá tener entregables parciales y útiles a lo largo del proyecto.
- **Adaptabilidad:** Podrá ajustar la planificación y las prioridades a medida que surjan nuevos requisitos o cambios en el entorno.

Es importante contar con una buena comunicación y colaboración con el equipo de desarrollo y los interesados en el proyecto para aprovechar al máximo el enfoque iterativo e incremental. Con este enfoque, se maximiza la probabilidad de éxito en la migración a MicroK8s y la entrega de una solución robusta y confiable.

## 3.14. Duración del Proyecto: 16 Semanas

A continuación, se presenta una planificación revisada para el proyecto de migración de Elastic Beanstalk a MicroK8s, teniendo en cuenta la dockerización y la reubicación de SQS y Elasticache dentro del clúster de Kubernetes. Dado que el proceso de dockerización puede llevar de 2 a 3 semanas, se ajustará el plan en consecuencia:

### 3.14.1 Semana 1: Planificación y Análisis Inicial

- Definir los objetivos y alcance actualizado del proyecto con la inclusión de dockerización.
- Investigar y comparar las características de Elastic Beanstalk y MicroK8s.
- Identificar los posibles desafíos y riesgos asociados a la dockerización y reubicación de servicios.
- Establecer un cronograma preliminar considerando el tiempo estimado para la dockerización.



### 3.14.2 Semana 2: Configuración del Entorno

- Instalar y configurar el entorno de desarrollo para trabajar con MicroK8s.
- Familiarizarse con las herramientas y comandos de MicroK8s.
- Configurar un clúster de Kubernetes local utilizando MicroK8s.

### 3.14.3 Semana 3: Dockerización de Servicios

- Comenzar el proceso de dockerización para SQS y Elasticache.
- Crear los Dockerfiles para cada servicio y realizar pruebas locales de los contenedores.

### 3.14.4 Semana 4: Dockerización de Servicios (Continuación)

- Continuar con el proceso de dockerización en caso de que no haya finalizado en la semana anterior.

- Realizar pruebas adicionales para garantizar que los contenedores funcionen correctamente.

### 3.14.5 Semana 5: Pruebas Locales y Ajustes

- Implementar los servicios dockerizados en el clúster local de MicroK8s.
- Realizar pruebas de integración para asegurar el correcto funcionamiento de los servicios en el nuevo entorno.
- Realizar ajustes y correcciones en caso de encontrar problemas.

### 3.14.6 Semana 6: Análisis de la Aplicación Actual

- Evaluar la arquitectura de la aplicación en Elastic Beanstalk y su interacción con SQS, Elasticache y RDS.
- Identificar los cambios necesarios para que la aplicación sea compatible con Kubernetes y los servicios dockerizados.

### 3.14.7 Semana 7: Preparación de la Aplicación para Kubernetes



- Implementar los cambios necesarios en la aplicación para su integración con los servicios dockerizados.
- Actualizar los archivos de configuración de Kubernetes para reflejar los cambios en la arquitectura.

### 3.14.8 Semana 8: Pruebas de Integración

- Realizar pruebas de integración para verificar la comunicación adecuada entre la aplicación y los servicios en el clúster de MicroK8s.
- Asegurarse de que los servicios dockerizados se integren correctamente con el resto de la aplicación.

### 3.14.9 Semana 9 a 10: Migración de Datos y Configuración

- Planificar y ejecutar la migración de datos desde Elastic Beanstalk a RDS.
- Realizar la configuración necesaria para que la aplicación se comunique con los servicios en el clúster de MicroK8s.

### 3.14.10 Semana 11: Implementación en un Entorno de Pruebas

- Configurar un entorno de pruebas separado utilizando MicroK8s.
- Desplegar la aplicación migrada y los servicios dockerizados en el entorno de pruebas.
- Realizar pruebas de rendimiento y escalabilidad en el nuevo entorno.

### 3.14.11 Semana 12: Optimización y Ajustes Finales

- Identificar posibles cuellos de botella y áreas de mejora en la nueva infraestructura.
- Optimizar la configuración de Kubernetes y MicroK8s para mejorar el rendimiento y la estabilidad.
- Realizar ajustes finales en la aplicación y en el entorno de Kubernetes.

### 3.14.12 Semana 13 a 14: Documentación

- Documentar todo el proceso de migración y dockerización.
- Crear guías y manuales para futuras referencias y para el equipo de operaciones.
- Preparar la documentación final del proyecto de grado.



### 3.14.13 15: Evaluación y Conclusiones

- Evaluar el éxito de la migración y si se lograron los objetivos establecidos.
- Resumir los resultados y conclusiones del proyecto.
- Identificar lecciones aprendidas y posibles mejoras para futuros proyectos similares.

### 3.14.14 Semana 16: Presentación del Proyecto de Grado

- Preparar una presentación para la defensa del proyecto de grado.
- Realizar la defensa del proyecto ante un panel de revisores o docentes.
- Entregar el trabajo final y la documentación completa.

Es importante tener en cuenta que los plazos y las tareas pueden variar según la complejidad del proyecto, los recursos disponibles y cualquier desafío imprevisto que surja durante el proceso de migración y dockerización. Se recomienda realizar un seguimiento constante del progreso y ajustar el plan según sea necesario para cumplir con los objetivos del proyecto. (Ver tabla 2).

| | **Semana** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Iteraciones** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Iteración 1 | | | | | | | | | | | | | | | | |
| Iteración 2 | | | | | | | | | | | | | | | | |
| Iteración 3 | | | | | | | | | | | | | | | | |
| Iteración 4 | | | | | | | | | | | | | | | | |
| Iteración 5 | | | | | | | | | | | | | | | | |
| Iteración 6 | | | | | | | | | | | | | | | | |
| Iteración 7 | | | | | | | | | | | | | | | | |
| Iteración 8 | | | | | | | | | | | | | | | | |
| Iteración 9 | | | | | | | | | | | | | | | | |
| Iteración 10 | | | | | | | | | | | | | | | | |

**Tabla 1**. Cronograma de iteraciones

| | Semana | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Evaluación** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Reuniones con el tutor | | | | | | | | | | | | | | | | |
| Presentación de avance del proyecto de grado | | | | | | | | | | | | | | | | |
| Entrega y revisiones | | | | | | | | | | | | | | | | |
| Entrega final | | | | | | | | | | | | | | | | |
| Defensa del proyecto de grado | | | | | | | | | | | | | | | | |

**Tabla 2.** Cronograma de evaluaciones

# IV Análisis de resultados

La migración del proyecto final (su primera versión lista para un entorno de pruebas), fue completada mediante la realización de pruebas locales, siguiendo las instrucciones detalladas de la documentación y los estándares de desarrollo de arquitectura con kubernetes y MicroK8s.

El primer paso realizado, tal como se describe en la planificación, la aplicación de la solución, en iteraciones (se empleó el sistema operativo Linux Ubuntu 20.04 LTS):

## 4.1. Pasos para la creación de la infraestructura

### 4.1.1 Instalación de dependencias

Se realizó la instalación de las dependencias necesarias localmente: NPM, RVM, ruby, react, docker, micro kubernetes, kubectl, Kompose y Docker Desktop (opcional). (Para visualizar detalle de la instalación de los paquetes, ver documentación del trabajo final).

### 4.1.2 Dockerizacion de la plataforma de Ignis gravitas volition



Fue una de las dockerizaciones más complejas, ya que requirió una optimización y solución de problemas nivel de comunicación del servidor de Puma de rails, además que, la aplicación de ruby on rails, requirió de una dockerizacion interna adicional que se comunica con un servidor proxy reverso de Nginx para poder servir los assets de la plataforma. (Imágenes).

### 4.1.3 Dockerizacion de Gravitas UI

Esta dockerizacion requirió optimización y, además, se hizo un mapeo de puertos para pasar a través del puerto 3002, debido a que la plataforma de Ignis de volition emplea el 3000. (Se requirieron ajustes de las variables de entorno del entorno de producción.)

### 4.1.4 Dockerizacion de la API de gravitas (hecha con feathers API)

En esta dockerizacion requirió de una solución de optimización y corrección de versión debido a problemas con docker y feathers.

### 4.1.5 Dockerizacion adicional del servidor proxy reverso de Nginx

Para esta dockerizacion se empleó una comunicación a modo de proxy reverso para servir los assets a través del puerto 8080 de la aplicación de Ruby on rails. El docker de Volition lanza un servidor en Puma (como lo hace Ruby on Rails por defecto para las aplicaciones creadas con el modelo MVC. Entonces el worker actuara como un Demonio (Daemon) para procesar a nivel de servidor las imágenes (assets) de la plataforma.

Los eventos de SQS de la cola de mensajes se escucharán a través de un worker o trabajo que se ejecutara en segundo plano. Al worker no le llegaran peticiones.

### 4.1.6 Migración de datos:

Se migro la base de datos de pruebas en RDS.

### 4.1.7 Implementación en el entorno de pruebas para las imágenes de docker

Se generó un entorno de pruebas locales para probar los contenedores apuntando a las variables de entorno de producción. (Se hizo monitorización de las imágenes con la herramienta de Docker Desktop). Las imágenes con las vulnerabilidades monitoreadas son las siguientes:



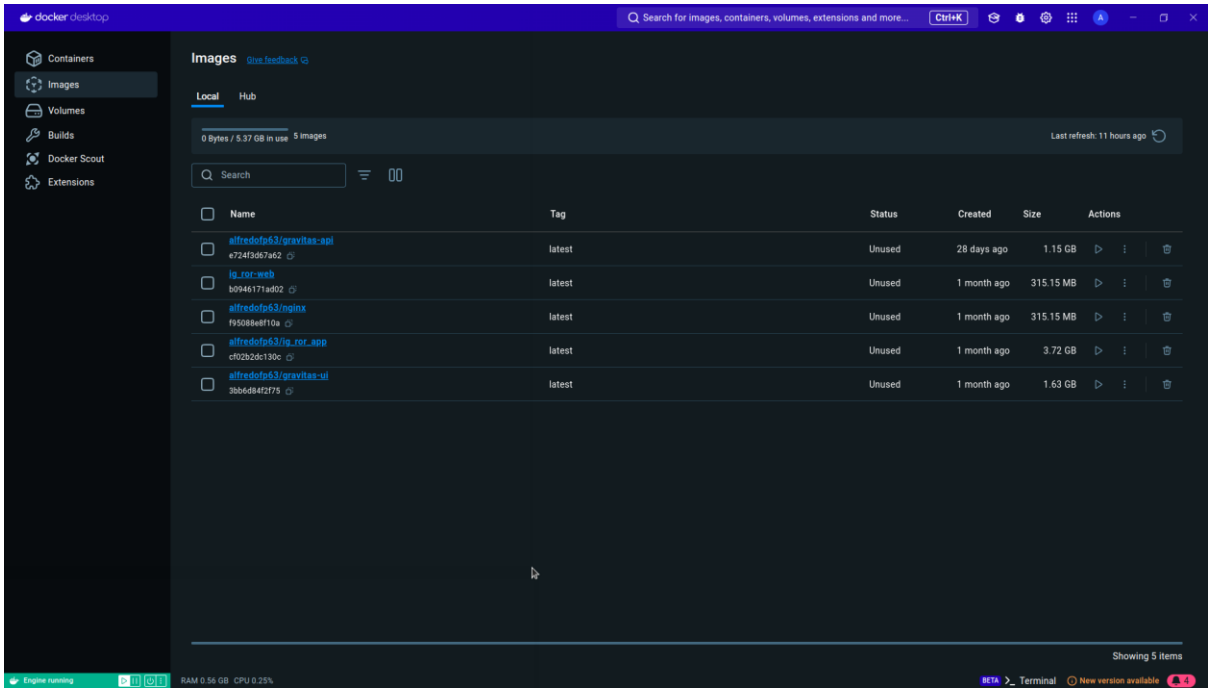


**Figura 8**: Imágenes de Docker en entorno local

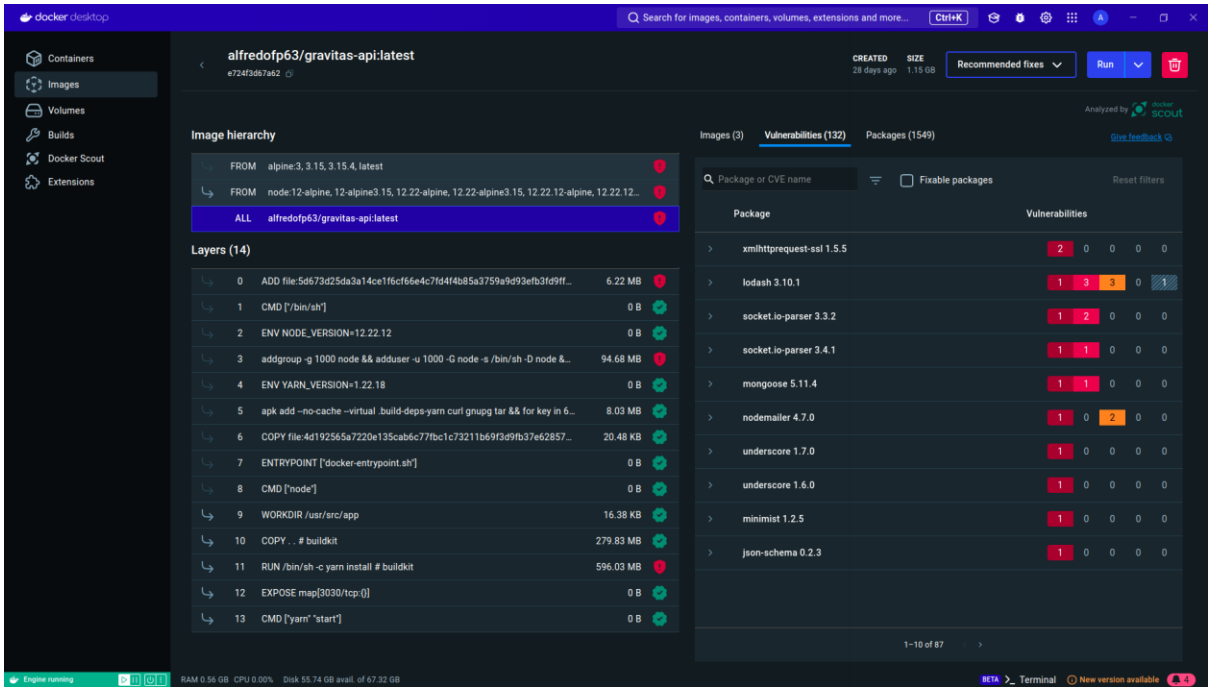


**Figura 9:** Imagen de Docker local de Gravitas API

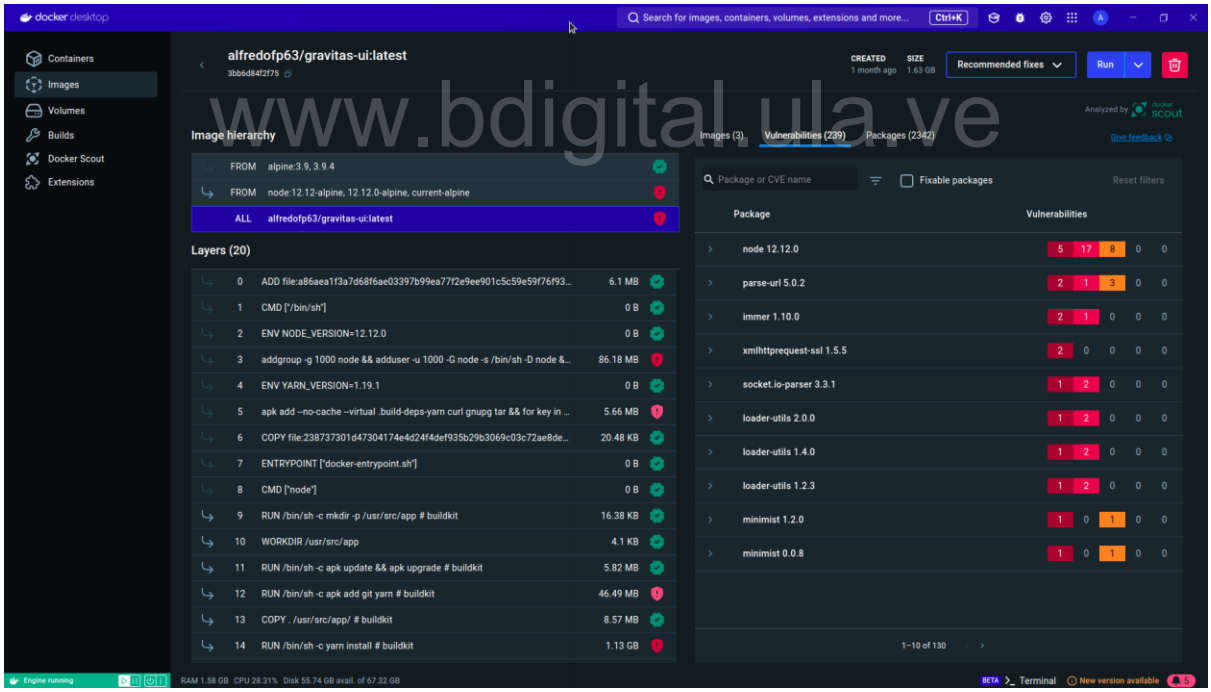


**Figura 10**: Imagen de Docker local de Gravitas UI

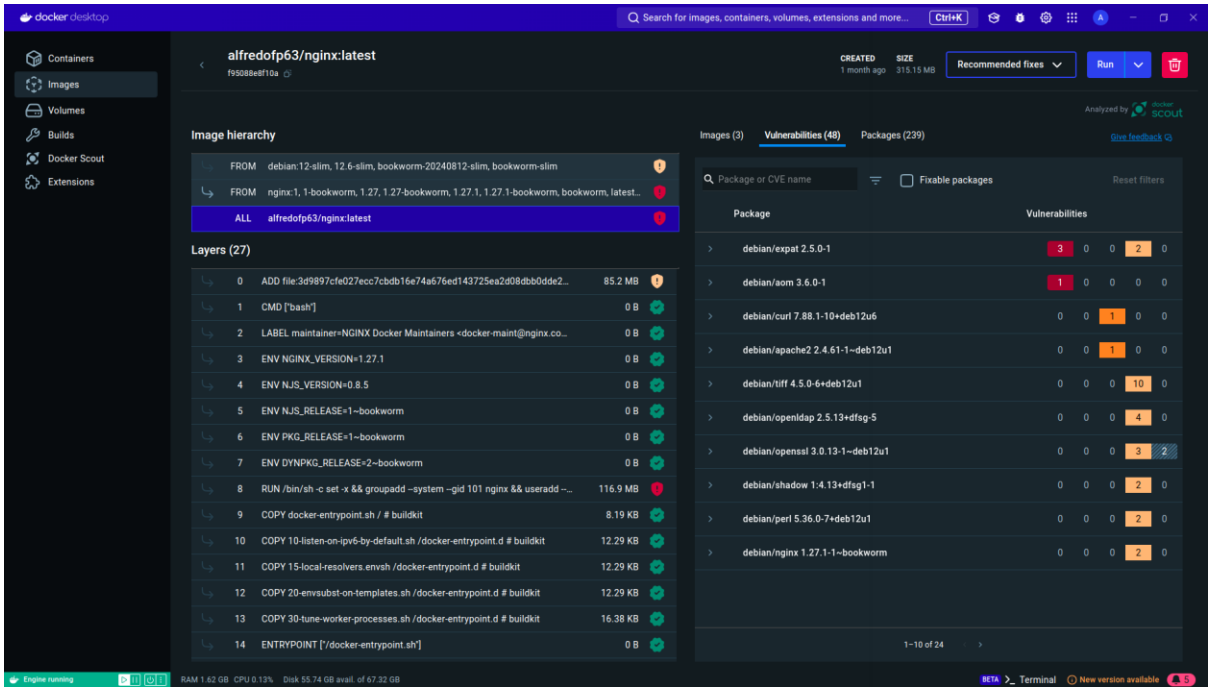


**Figura 11**: Imagen de Docker local de Nginx proxy reverso

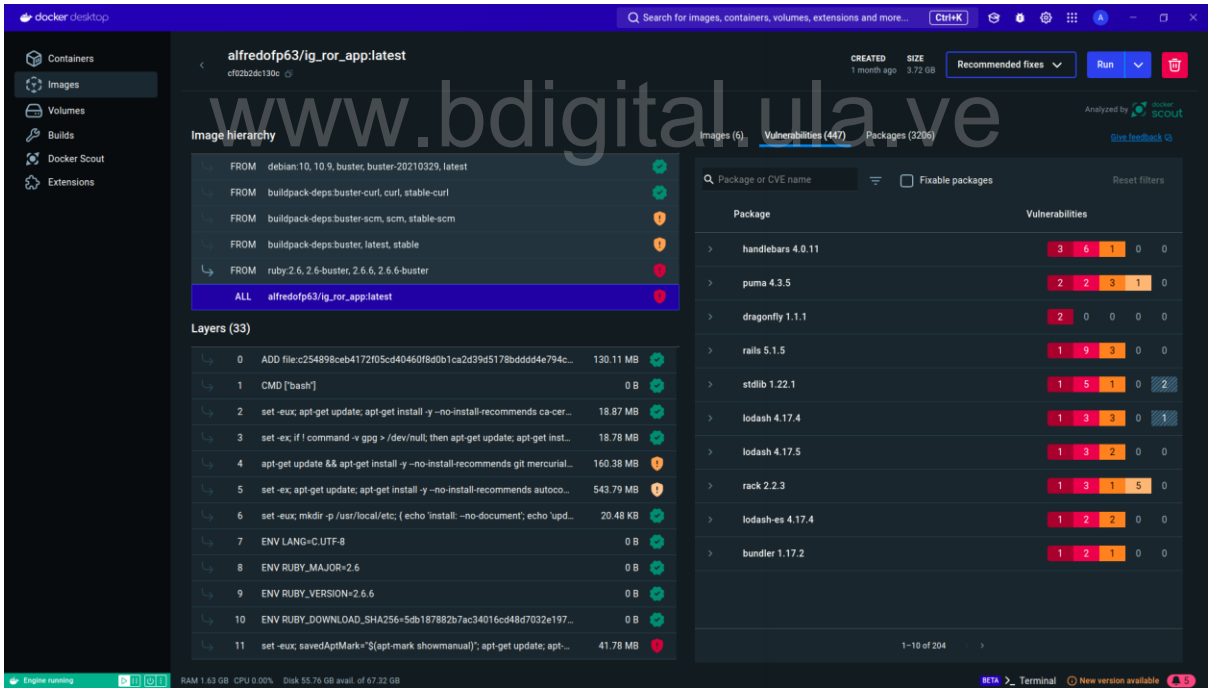


**Figura 12:** Imagen de Docker local de Volition Ignis Gravitas.

**El volumen compartido de datos:**

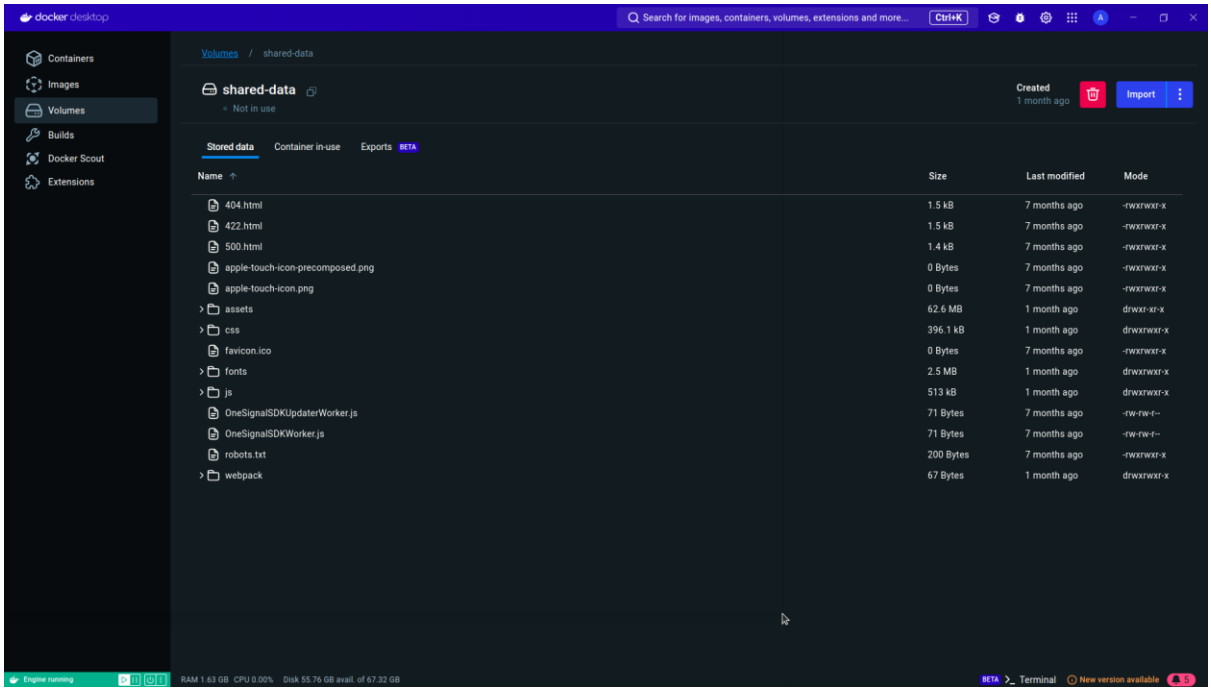


**Figura 13**: Volumen de datos compartidos.

### 4.1.8 Adecuación de imágenes en Docker Hub



Se empleó Docker Hub para subir las imágenes compiladas en una nube de administración con el código privado protegido en GitHub.

### 4.1.9 Docker compose

Una vez completada la dockerizacion, se generó un archivo de docker compose, con todas las imágenes necesarias para el cluster.

### 4.1.10 Generación de archivos YAML

Para comenzar el montaje del cluster, se empleó Kompose, como herramienta para generar los archivos YAML para el clúster de micro kubernetes.

### 4.1.11 Creación del cluster de MicroK8s

Una vez creados los archivos YAML, se continuó realizando el estándar para crear el cluster local, una vez completados los archivos y he ejecutado el comando.

### 4.1.12 Creación de pruebas locales del cluster de MicroK8

Para crear el cluster y correrlo, se ejecutaron una serie de comandos para crear el servidor. (Se generó un script para el deployment dentro del código fuente para ejecutar cada instalación de dependencia y creación del cluster).

A continuación, las imágenes del entorno de pruebas locales del cluster:

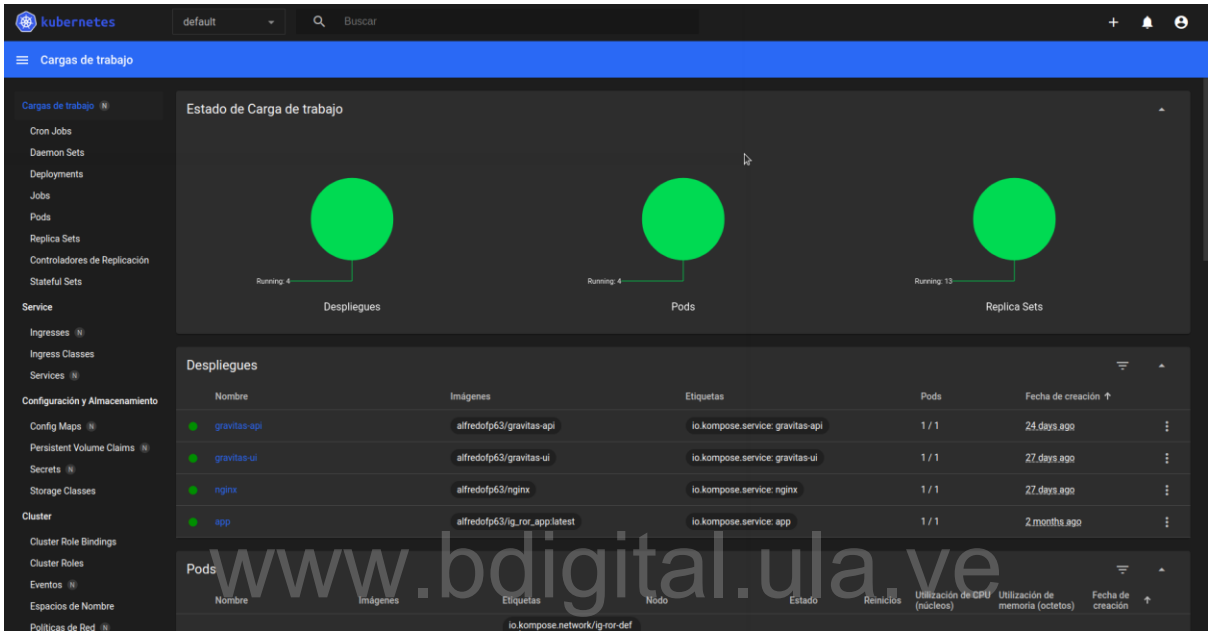


**Figura 1** Gráfico del cluster, pods y conjuntos de réplica.

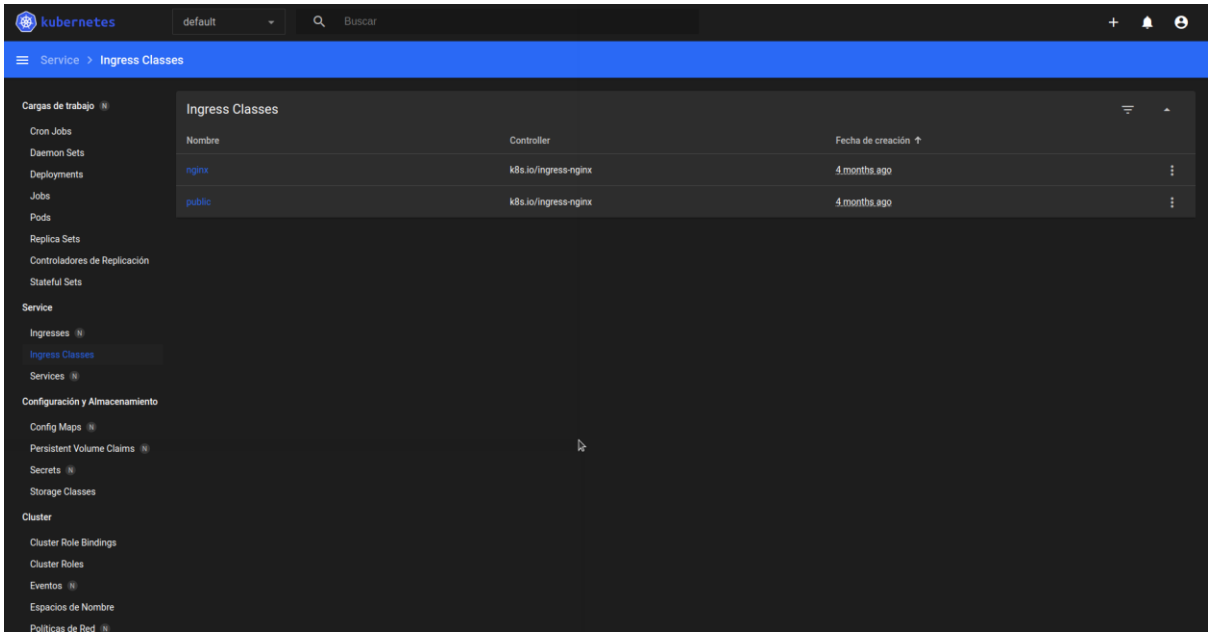


**Figura 2** Gráfico de archivos de configuración de ingress.

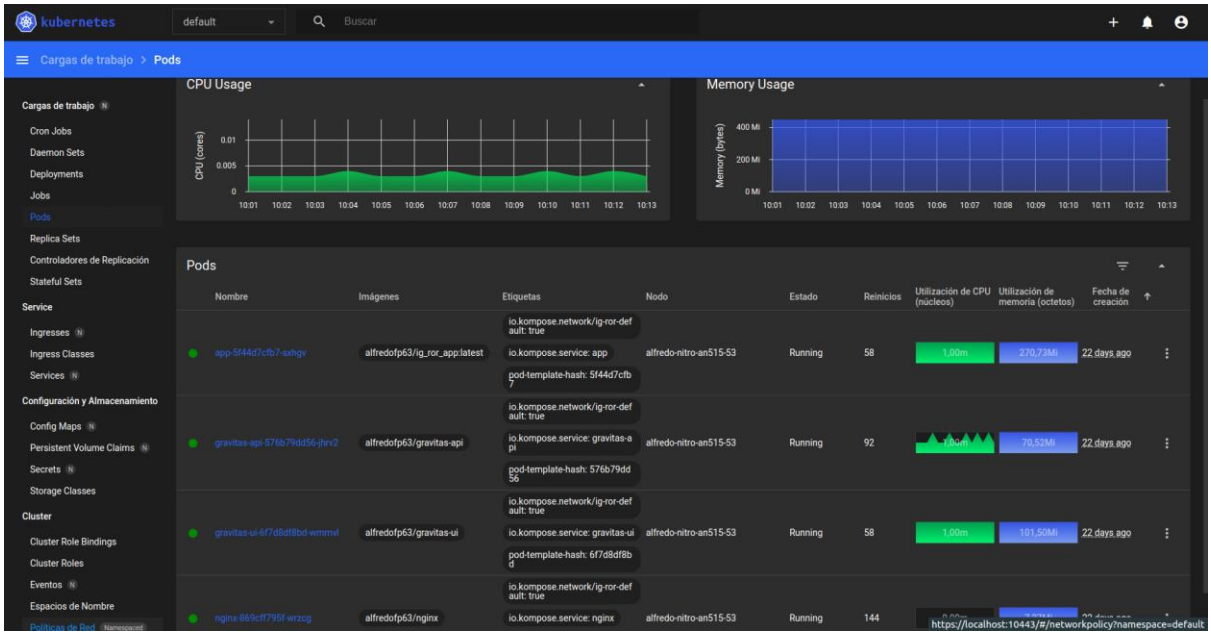


**Figura 3** Gráfico del cluster, pods y conjuntos de réplica, rendimiento de memoria.

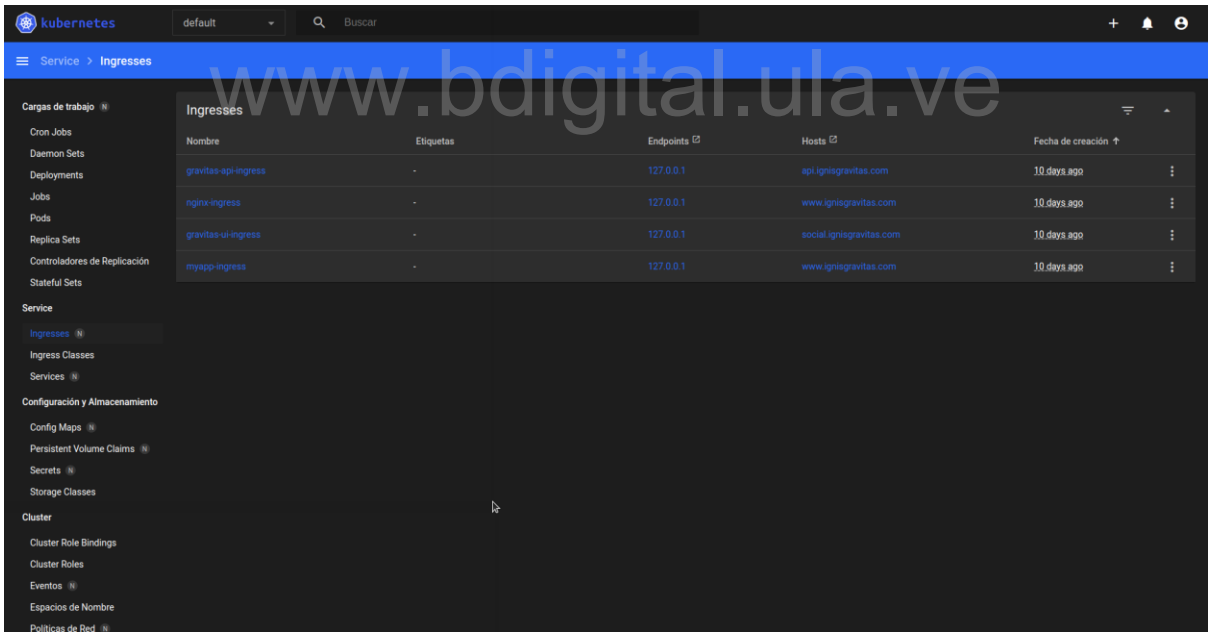




**Figura 4** Gráfico de clases de ingress.

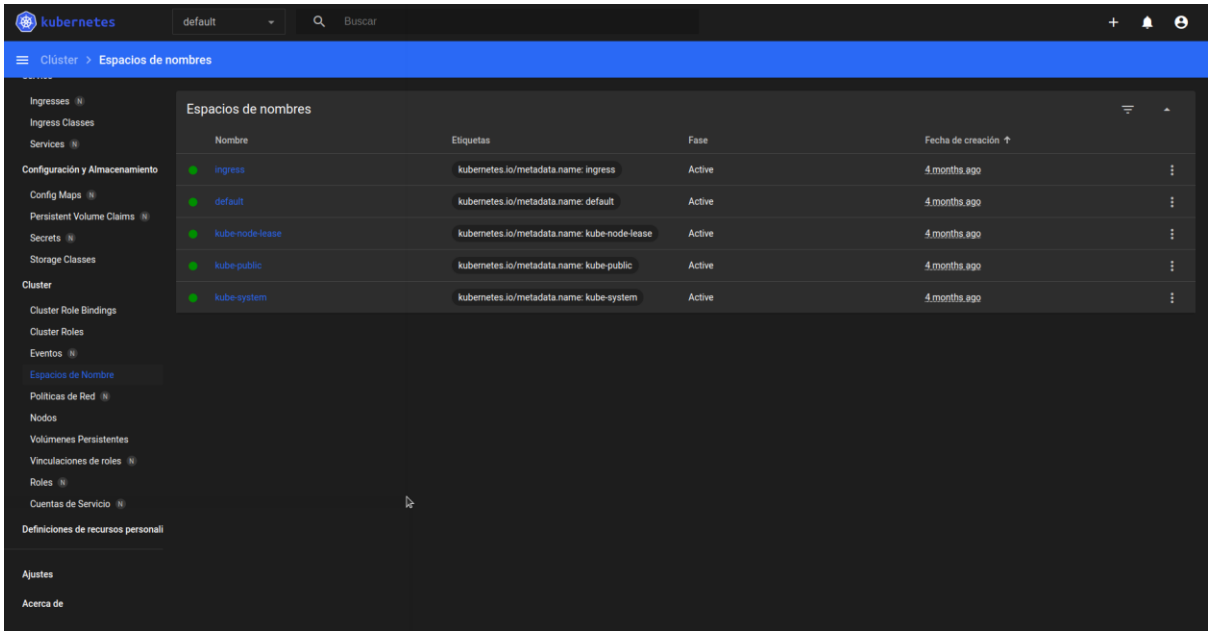


**Figura 5** Gráfico de espacio de nombres.

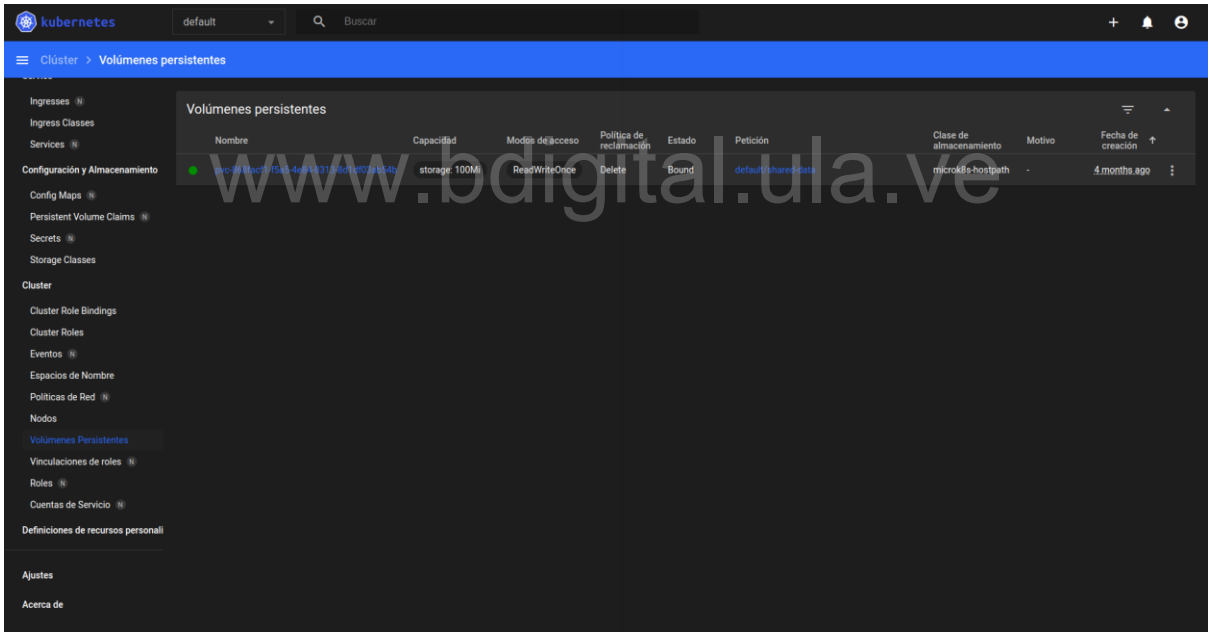


**Figura 6** Gráfico de Volúmenes persistentes.

## 4.2. Culminación de las pruebas del trabajo final:

Una vez completada la migración integral de la arquitectura desde Elastic Beanstalk a MicroK8s (Micro Kubernetes), y tras la dockerización de todos los servicios y pods de la compañía, se identificó un problema de acoplamiento en la infraestructura actual. Este problema radica en una dependencia operativa de la aplicación Ignis Gravitas (Volition), desarrollada en Ruby on Rails, con la arquitectura de Elastic Beanstalk.

Esta dependencia se debe a que Ruby on Rails opera un clúster independiente internamente para su funcionamiento, ejecutando un servidor Puma. Además, la cola de mensajes y notificaciones de la aplicación utiliza Amazon SQS, pero todo esto se gestiona a través de un worker que ejecuta un "job" en la plataforma en la nube de AWS, estando completamente integrado y acoplado a la arquitectura de Elastic Beanstalk.

Para llevar a cabo la migración del worker de la arquitectura anterior a la nueva arquitectura de MicroK8s, es necesario modificar la forma en que el sistema interactúa, envía y recibe los mensajes de la cola de SQS, adoptando una metodología estándar de Ruby on Rails. Tras diversas investigaciones y la evaluación de alternativas, se concluyó que era necesario cambiar la comunicación del worker para que utilizara Redis como intermediario. Esto se lograría mediante la creación de un pod de Sidekiq, una herramienta de software compatible con Ruby on Rails, que permite gestionar la cola de mensajes en segundo plano a través de Redis antes de procesarla con Sidekiq, facilitando así la migración del worker.



No obstante, tras varios intentos de implementar esta estrategia, surgieron problemas con la versión de Ruby del proyecto, la cual está algo desactualizada. Además, se requiere la configuración de un worker y conexiones específicas para un clúster de Kubernetes, lo cual implica un ajuste significativo en los tiempos de desarrollo, superando los plazos inicialmente previstos para la culminación de este proyecto de grado.

Este cambio de alto impacto requiere una cantidad considerable de pruebas exhaustivas antes de implementar los cambios en los entornos de producción. A pesar de estos desafíos, se logró el alcance del proyecto de migración. Se realizaron pruebas en máquinas virtuales EC2 en la nube de AWS, donde se verificó el estado del clúster y sus pods con sus respectivas réplicas funcionando correctamente.

No es viable mantener un entorno de desarrollo en ejecución permanente en el servidor debido a los costos asociados y la imposibilidad de crear y conectar el worker a la nueva arquitectura. Sin embargo, se desarrolló un script que automatiza la creación del clúster una vez se inicia una instancia EC2 de AWS. Este script permite, una vez completada la migración del worker, trasladar todos los cambios al entorno de producción de la compañía, reduciendo recursos y costos.

Este factor constituye una limitante para llevar el trabajo final a un entorno "live en producción", pero cumple con los objetivos planteados al inicio del desarrollo. Además, con la implementación de este nuevo sistema de arquitectura y sus servicios, es posible realizar una migración a otras plataformas en la nube, en caso de ser necesario para reducir costos de mantenimiento. El clúster de Kubernetes puede operar en cualquier nube con cualquier instancia o máquina virtual sin complicaciones adicionales.

## 4.3. Documentación de dependencias y de la infraestructura:

Los siguientes son los pasos de instalación de dependencias necesarias para una máquina con sistema operativo Linux, (para otros sistemas consultar la documentación de cada herramienta).



### 4.1.13 Instalación de dependencias:

#### 4.1.13.1 Instalar Node.js y React

curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -

sudo apt-get install -y nodejs

#### 4.1.13.2 Instalar React:

npm install -g create-react-app

npx create-react-app mi-app

cd mi-app

npm start

#### 4.1.13.3 Instalar Ruby on Rails

sudo apt-get update

sudo apt-get install ruby-full

#### 4.1.13.4 Instalar NVM y RVM

**Instalar NVM:**

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash

source ~/.bashrc

nvm install node

**Instalar RVM:**

\curl -sSL https://get.rvm.io | bash -s stable

source ~/.rvm/scripts/rvm



rvm install ruby

rvm use ruby --default

#### 4.1.13.5 Instalar Rails

gem install rails

#### 4.1.13.6 Instalar Docker

**Agregar el repositorio de Docker:**

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

sudo apt-get update

**Instalar Docker:**

sudo apt-get install -y docker-ce docker-ce-cli containerd.io

sudo systemctl start docker

sudo systemctl enable docker

#### 4.1.13.7 Instalar Docker Hub (Opcional)

Registrar Docker Hub:

Visita el sitio web de Docker Hub y regístrate o inicia sesión.

Sigue las instrucciones para configurar tu cuenta y obtener un token de acceso.

#### 4.1.13.8 Instalar Kubernetes

**Instalar MicroK8s:**



sudo snap install microk8s --classic

sudo microk8s.status

sudo microk8s.enable dns storage rbac

**Instalar kubectl:**

sudo apt-get install -y apt-transport-https curl

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo sh -c 'echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" > /etc/apt/sources.list.d/kubernetes.list'

sudo apt-get update

sudo apt-get install -y kubectl

#### 4.1.13.9 Instalar Kompose

sudo curl -L https://github.com/kubernetes/kompose/releases/download/v1.26.0/kompose-linux-amd64 -o kompose

sudo chmod +x kompose

sudo mv kompose /usr/local/bin/kompose

### 4.1.14 Instalar Helm:

curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null

sudo apt-get install apt-transport-https --yes

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list



sudo apt-get update

sudo apt-get install helm

### 4.1.15 Como crear y correr el cluster de MicroK8s en local

#### 4.1.15.1 Parte 1: Verificar Instalación de MicroK8s

sudo microk8s.status

sudo microk8s.enable dns storage rbac

#### 4.1.15.2 Parte 2: Configuración del Cluster

**Agregar un nodo al cluster**:

sudo microk8s add-node

**Verificar el estado del cluster:**

sudo microk8s.status

**Configurar el almacenamiento (si es necesario):**

sudo microk8s.enable

#### 4.1.15.3 **Parte 3: En caso que se requieran reiniciar los certificados locales**:

sudo microk8s refresh-certs --cert ca.crt

#### 4.1.15.4 Correr el dashboard del cluster en local:

microk8s kubectl port-forward -n kube-system service/kubernetes-dashboard 10443:443

(Se vera un mensaje como el siguiente):

Forwarding from 127.0.0.1:10443 -> 8443

Forwarding from [::1]:10443 -> 8443

La URL para deployment local: https://localhost:10443/#/pod?namespace=default

#### 4.1.15.5 Para poder acceder debemos ir a la opción acceder al sitio en opciones avanzadas y generar un token de acceso local:



token=$(microk8s kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)

**Copiamos el token con:**

microk8s kubectl -n kube-system describe secret $token

Se introduce en el navegador y accedemos al dashboard local.

Los archivos YAML y docker files y docker compose deben estar previamente generados. Esto se realizó en el análisis de la metodología de trabajo. Una vez hecho eso, es posible ejecutar:

#### 4.1.15.6 Generación de archivos de kubernetes (YAML) o manifiestos con Kompose:

$ kompose convert -f compose.yaml

$ kubectl apply -f .

$ kubectl get po

#### 4.1.15.7 Aplicar archivos de manifiestos al cluster:

microk8s kubectl apply -f resource-manifest.yaml

#### 4.1.15.8 En caso de necesitar borrar un archivo de manifiesto:

kubectl delete -f resource-manifest.yaml

#### 4.1.15.9 Habilitar ingress en el cluster:

microk8s enable ingress

### 4.1.16 Generar archivos de ingress

microk8s kubectl apply -f service-app.yml

**Nota**: Se debe realizar este comando para cada uno de los archivos de servicio de cada pod.

Script para creación e instalación de dependencias en el cluster:

README.md ● $ deploy.sh ●

$ deploy.sh

```
#!/bin/bash
# Variables
MASTER_IP="your_master_ip"
TOKEN="your_token"
POD_NETWORK_CIDR="192.168.0.0/16"
# Install Docker
sudo apt-get update
sudo apt-get install -y docker.io
sudo systemctl enable docker
sudo systemctl start docker
# Install kubeadm, kubelet, and kubectl
sudo apt-get update && sudo apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
# Initialize the Kubernetes master node
sudo kubeadm init --pod-network-cidr=$POD_NETWORK_CIDR
# Set up kubeconfig for the root user
sudo --user=root mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
# Install a pod network add-on (Calico in this case)
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
# Join worker nodes (run this command on each worker node)
# sudo kubeadm join $MASTER_IP:6443 --token $TOKEN --discovery-token-ca-cert-hash sha256:<hash>
# Install Kompose
sudo curl -L https://github.com/kubernetes/kompose/releases/download/v1.24.0/kompose-linux-amd64 -o /usr/local/bin/kompose
sudo chmod +x /usr/local/bin/kompose
# Convert Docker Compose to Kubernetes manifests
kompose convert -f docker-compose.yml
# Apply the Kubernetes manifests
kubectl apply -f .
echo "Kubernetes cluster setup and application deployment complete."
```



**Figura 16**: Script para ejecutar deployment en la maquina de EC2 e instalación de dependencias.

#### 4.1.16.1 Argumentación del script

**Pasos Explicados:**

- **Instalar Docker:** Asegura que Docker esté instalado y en funcionamiento.
- **Instalar herramientas de Kubernetes**: Instala kubeadm, kubelet y kubectl.
- **Inicializar el maestro de Kubernetes**: Configura el nodo maestro con un CIDR de red de pods especificado.
- **Configurar kubeconfig**: Configura el archivo kubeconfig para el usuario root.
- **Instalar complemento de red de pods**: Aplica el complemento de red.
- **Unir nodos de trabajo**: Proporciona un comando para unir nodos de trabajo al clúster.
- **Instalar Kompose**: Descarga e instala Kompose para convertir archivos Docker Compose.
- **Convertir y aplicar manifiestos**: Convierte archivos Docker Compose a manifiestos de Kubernetes y los aplica.

# V Conclusiones y recomendaciones

## 5.1. Conclusiones

Dependencia de la Arquitectura Anterior: La migración desde Elastic Beanstalk a MicroK8s reveló una fuerte dependencia de la aplicación Ignis Gravitas (Volition) con la arquitectura anterior, debido a la integración de Ruby on Rails con el servidor Puma y la cola de mensajes SQS gestionada por un worker en AWS.

- **Desafíos Técnicos**: La implementación de la nueva arquitectura encontró obstáculos significativos, incluyendo la necesidad de actualizar la versión de Ruby y configurar workers específicos para Kubernetes. Estos desafíos extendieron los tiempos de desarrollo más allá de lo previsto inicialmente; además de que deben realizarse configuraciones adicionales; debido a las versiones deprecadas de ruby, ruby on rails y React.
- **Pruebas y Validación**: A pesar de los desafíos, se lograron realizar pruebas exitosas en máquinas virtuales EC2, verificando el correcto funcionamiento del clúster y sus pods. Sin embargo, los costos asociados a mantener un entorno de desarrollo en ejecución permanente impiden su implementación continua.
- **Automatización y Reducción de Costos:** El desarrollo de un script automatizado para la creación del clúster en instancias EC2 representa una solución viable para reducir costos y recursos, permitiendo la migración del worker y la implementación en el entorno de producción.
- **Flexibilidad de la Nueva Arquitectura**: La nueva arquitectura basada en MicroK8s ofrece la flexibilidad de migrar a otras plataformas en la nube, lo cual puede ser beneficioso para reducir costos de mantenimiento en el futuro.

Para visualizar la tabla de recuperación de costos aproximados con la migración de la arquitectura ver tabla 6.

## 5.2. Recomendaciones

- **Actualización de Tecnologías**: Se recomienda planificar y ejecutar una actualización de la versión de Ruby utilizada en el proyecto para evitar problemas de compatibilidad y mejorar la eficiencia del desarrollo.

- **Pruebas exhaustivas**: Antes de implementar cambios en el entorno de producción, es crucial realizar pruebas exhaustivas en entornos de desarrollo y pruebas para garantizar la estabilidad y el rendimiento del sistema.
- **Optimización de Recursos**: Continuar desarrollando y mejorando scripts de automatización que permitan la creación y gestión eficiente de clústeres y pods, reduciendo así los costos operativos y de mantenimiento.
- **Documentación y Capacitación**: Documentar detalladamente todos los cambios realizados y capacitar al equipo de desarrollo en el uso de la nueva arquitectura y herramientas, asegurando una transición fluida y eficiente.
- **Evaluación de Proveedores de Nube**: Considerar la posibilidad de migrar a otros proveedores de servicios en la nube que ofrezcan mejores costos y beneficios, aprovechando la flexibilidad de la arquitectura basada en Kubernetes.

### 5.3. Argumentos y guía de procedimientos para resolver el problema de la cola de mensajes (para trabajos futuros):



Para resolver este problema de implementación, se sugiere seguir el estándar de empleo de Redis como intermediario, y mandar la cola de mensajes a sidekiq

Procedimiento para instalar y configurar sidekiq en la documentación oficial de sidekiq.

### 5.4. Procedimiento para implementar la solución (Según la documentación de kubernetes-rails):

- Instalar la gema de sidekiq.
- Crear un bundle para el job y el worker.
- Agregar el controlador
- Agregar el archivo de configuración de manifiesto YML.

## VI Referencias Bibliográfica

Amazon. (2023, noviembre). Migración de la aplicación desde una versión de plataforma heredada. (n.d.). Recuperado de: https://docs.aws.amazon.com/es_es/elasticbeanstalk/latest/dg/using-features.migration.html

Amazon. (2023, octubre). "Preguntas frecuentes sobre AWS Elastic Beanstalk". Recuperado de: https://aws.amazon.com/es/elasticbeanstalk/faqs/

Amazon. (2023, octubre). ¿Qué es la nube privada? - Explicación de la nube privada - AWS. Recuperado de: https://aws.amazon.com/es/what-is/private-cloud/

Arias, E. R. (2021, febrero). Investigación cuantitativa. Economipedia. https://economipedia.com/definiciones/investigacion-cuantitativa.html

AWS. (2024). *Amazon Pricing calculator*. Recuperado de: https://calculator.aws/

Ayyagari, V. (n-d). ¿Qué es la orquestación de contenedores? Recuperado de: https://learn.microsoft.com/es-es/azure/architecture/microservices/design/orchestration

Baró, L. (2021, January). Creation of a Kubernetes Infrastructure. Degree Thesis submitted to the Faculty of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona Universitat Politècnica de Catalunya [Catalunya]. Recuperado de: https://upcommons.upc.edu/bitstream/handle/2117/344394/Creation%20of%20a%20Kubernetes%20Infrastructure.pdf?sequence=2

Colli, M. (2023) Kubernetes rails. *Deploying a Rails application to Kubernetes*. Recuperado de: https://kubernetes-rails.com/#workers

De loof, N. (2019). GitHub. *Compose-on-kubernetes*. Recuperado de: https://github.com/docker/compose-on-kubernetes/blob/master/docs/install-on-microk8s.md

Doerrfeld, B. (11 de septiembre de 2024). Computer World. *El futuro de Kubernetes y la infraestructura en la nube*. Recuperado de:

https://www.computerworld.es/article/3515312/el-futuro-de-kubernetes-y-la-infraestructura-en-la-nube.html

Elasticsearch en Kubernetes | Desplegar Y orquestar Elasticsearch en Kubernetes. (n.d.). Elastic. Recuperado de: https://www.elastic.co/es/elastic-cloud-kubernetes

Flowcharts, (2024), Ferreira, P. *Diagrama de arquitectura de Ignis Gravitas Inc*. https://app.diagrams.net/?libs=general;aws4#G1Hze14i-MgkWKEOrAdVPTjUsPtgLgvoL7#%7B%22pageId%22%3A%22Ht1M8jgEwFfnCIfOTk4-%22%7D

García, J. (2021). Tecno Simple. *¿Qué es un nodo de kubernetes?* Recuperado de: https://tecno-simple.com/que-es-un-nodo-de-kubernetes/

Hernández, R. Fernández, C y Baptista, P. (abril, 2014, p.199). Metodología de la investigación. México, México D.F. Editorial Mc Graw-Hill. Recuperado de: https://perio.unlp.edu.ar/catedras/wp-content/uploads/sites/151/2021/08/Hernandez-Sampieri.-Metodologia-de-la-investigacion.pdf

IBM. (n-d). Orquestación de contenedores para microservicios. Recuperado de: https://www.ibm.com/mx-es/topics/container-orchestration

Keep coding. (2023, May 24). Kubernetes: Que es Y características. Recuperado de: https://keepcoding.io/blog/que-es-kubernetes/

Khajeh, H., Greenwood, D., & Sommerville, I. (n.d.). Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. Cloud Migration. Recuperado de: https://arxiv.org/ftp/arxiv/papers/1002/1002.3492.pdf

Kompose. (2024). *Kompose*. Recuperado de: https://kompose.io/

Kubernetes, (2024). Kubernetes. *Documentación de Kubernetes - Ingress*. Recuperado de: https://kubernetes.io/es/docs/concepts/architecture/nodes/

Kubernetes, (2024). Kubernetes. *Documentación de Kubernetes - Nodos*. Recuperado de: https://kubernetes.io/docs/concepts/services-networking/ingress/

Kubernetes, (2024). Kubernetes. *Documentación de Kubernetes - Pods*. Recuperado de: https://kubernetes.io/es/docs/concepts/workloads/pods/pod/

Kubernetes, (2024). Kubernetes. *Documentación de Kubernetes*. Recuperado de: https://kubernetes.io/

Kubernetes. (2024). *Kubernetes documentation*. Recuperado de: https://kubernetes.io/

López, J. (2021, January). Analysis And Practice of Micro-Services Deployment Technologies based on Containers. A Degree Thesis Submitted to the Faculty of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona Universitat Politècnica de Catalunya [Catalunya]. Recuperado de: https://upcommons.upc.edu/bitstream/handle/2117/344392/Jesus_Thesis.pdf?sequence=2

Mickiewicz, M. *AWS Elastic Beanstalk vs EC2: A Detailed Comparison* Sitepoint. (30 de marzo del 2023). Recuperado de: https://www.sitepoint.com/aws-elastic-beanstalk-vs-ec2/

MicroK8s (2024). MicroK8s. *The effortless Kubernetes Zero-ops, pure-upstream, HA Kubernetes, from developer workstations to production* Recuperado de: https://microk8s.io/



Muguira, A. (2023, febrero). Diseño de investigación. Elementos y características. QuestionPro. https://www.questionpro.com/blog/es/diseno-de-investigacion/

Parra, A. (2023, febrero). Muestreo intencional. Características Y ejemplos. QuestionPro. https://www.questionpro.com/blog/es/muestreo-intencional/

Población y muestra de investigación - Definición, proceso, técnicas y fórmulas. (2022, octubre). Organizadoresgraficos.org. https://www.organizadoresgraficos.org/poblacion-y-muestra-de-investigacion/

Población y muestra: Definición Y diferencias con UN ejemplo. (2022, 20). Matemente. https://www.matemente.com/poblacion-y-muestra/

Pons, L. (2021, September 6). ¿Qué es Kubernetes Y cuáles son sus ventajas? Recuperado de: https://www.icm.es/2020/03/07/kubernetes/

Rodríguez, P. (4 de octubre de 2021). *El talón de Aquiles de AWS son sus altas tarifas de salida de datos, y sus rivales empiezan a explotarlo: guerra de precios contra el gigante de la*

*nube.* Xataka. Recuperado de: https://www.xataka.com/pro/talon-aquiles-aws-sus-altas-tarifas-salida-datos-sus-rivales-empiezan-a-explotarlo-guerra-precios-gigante-nube

Ruby on Rails. (2024). *Ruby on Rails guides*. Recuperado de: https://guides.rubyonrails.org/active_job_basics.html

Sidekiq. (2023). Sidekiq. *Getting Started*. Recuperado de: https://github.com/sidekiq/sidekiq/wiki/Getting-Started

Sidekiq. (2023). Sidekiq. *Simple, efficient background processing for Ruby*. Recuperado de: https://github.com/sidekiq/sidekiq/tree/v5.1.3?tab=readme-ov-file

StackShare, (Octubre, 2023) Beanstalk vs Kubernetes | What are the differences? (n.d.). Stack Share. Recuperado de: https://stackshare.io/stackups/beanstalk-vs-kubernetes

Woodward, C. (2020). Thinking in Microservices. *Docker Compose to Kubernetes part II*. Recuperado de: http://thinkmicroservices.com/blog/2020/kubernetes/docker-compose-to-kubernetes-part-2.html

Woodward, C. (2020). Thinking in Microservices. *Docker Compose to Kubernetes part III*. Recuperado de: http://thinkmicroservices.com/blog/2020/kubernetes/docker-compose-to-kubernetes-part-3.html

Woodward, C. (2020). Thinking in Microservices. *From Docker-Compose to Kubernetes Part I. Kompose*. Recuperado de: http://thinkmicroservices.com/blog/2020/kubernetes/docker-compose-to-kubernetes-part-1.html