

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

ESCUELA DE INFORMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍA DE LA INFORMACIÓN

INFRAESTRUCTURA KUBERNETES PARA IMPLEMENTACIÓN DE APLICACIONES EN CONTENEDORES EN ENTORNO DE PRUEBAS EN LA UNIDAD DE TECNOLOGÍAS DE LA INFORMACIÓN DE LA PUCE-IBARRA

DIANA ABIGAIL SÁNCHEZ VILLACÍS

TUTOR: GALO HERNÁN PUETATE HUERA

IBARRA – ECUADOR

JULIO, 2024

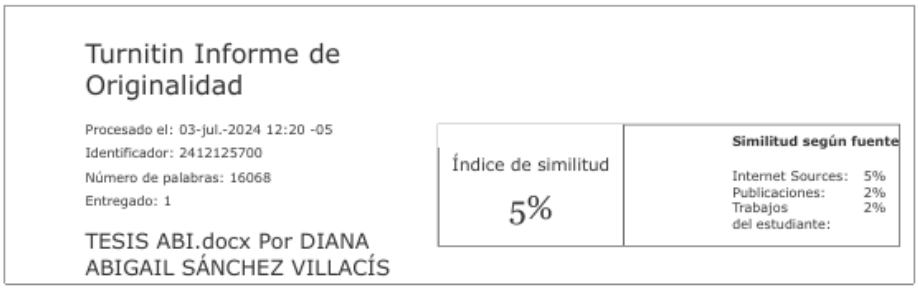
Ibarra, 04 de julio de 2024

CERTIFICACIÓN TUTOR

En mi calidad de Tutor del Trabajo de integración curricular titulado: INFRAESTRUCTURA KUBERNETES PARA IMPLEMENTACIÓN DE APLICACIONES EN CONTENEDORES EN ENTORNO DE PRUEBAS EN LA UNIDAD DE TECNOLOGÍAS DE LA INFORMACIÓN DE LA PUCE-IBARRA, presentado por el estudiante DIANA ABIGAIL SÁNCHEZ VILLACÍS con cédula de ciudadanía N° 2101131262, para obtener el Título de INGENIERA EN TECNOLOGÍAS DE LA INFORMACIÓN.

Certifico que el trabajo cumple con todos los parámetros establecidos, mediante el cual el estudiante demuestra el desarrollo de competencias en el campo de conocimiento de su profesión con un nivel de argumentación coherente, para ser sometido a la evaluación por parte de los lectores.

Adicionalmente, se adjunta el certificado de porcentaje de originalidad de TURNITIN.



Turnitin Informe de Originalidad

Procesado el: 03-jul.-2024 12:20 -05
Identificador: 2412125700
Número de palabras: 16068
Entregado: 1

TESIS ABI.docx Por DIANA ABIGAIL SÁNCHEZ VILLACÍS

Índice de similitud	Similitud según fuente	
5%	Internet Sources:	5%
	Publicaciones:	2%
	Trabajos del estudiante:	2%

(f):__ _____

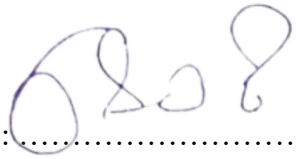
Mgs. GALO HERNÁN PUETATE HUERA

TUTOR DE TRABAJO

C.C.: 0401375787

PÁGINA DE APROBACIÓN DEL TRIBUNAL

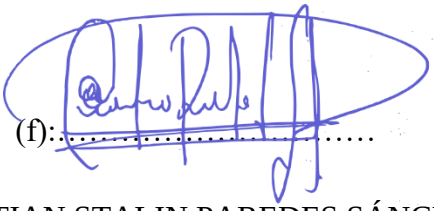
El tribunal examinador, aprueba el presente trabajo en nombre de la Pontificia Universidad Católica del Ecuador Ibarra:



(f):……

Mgs. GALO HERNÁN PUETATE HUERA

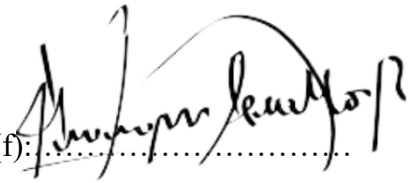
C.C.: 0401375787



(f):……

Ing. CRISTIAN STALIN PAREDES SÁNCHEZ

C.C.: 0401355912



(f):……

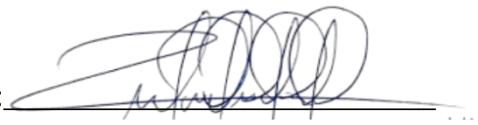
Mcs. ÁLVARO MAURICIO CEVALLOS RAMÍREZ

C.C.: 1002494019

ACTA DE CESIÓN DE DERECHOS

Yo, *DIANA ABIGAIL SÁNCHEZ VILLACÍS*, declaro conocer y aceptar la disposición del Art. 165 del Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación, que manifiesta textualmente: "Se reconoce facultad de los autores y demás titulares de derechos de disponer de sus derechos o autorizar las utilizaciones de sus obras o prestaciones a título gratuito y oneroso, según las condiciones que determinen. Esta facultad podrá ejercerse mediante licencias libres, abiertas y otros modelos alternativos de licenciamiento o la renuncia".

Ibarra, 04 de julio de 2024

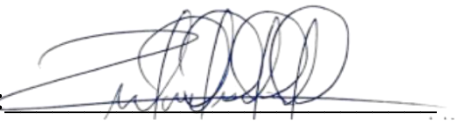
(f):

DIANA ABIGAIL SÁNCHEZ VILLACÍS

C.C.: 2101131262

AUTORÍA

Yo, *DIANA ABIGAIL SÁNCHEZ VILLACÍS*, portadora de la cédula de ciudadanía N° 2101131262 declaro que el presente trabajo de investigación es de total responsabilidad de la autora, y eximo expresamente a la Pontificia Universidad Católica del Ecuador Ibarra de posibles reclamos o acciones legales.

(f):

DIANA ABIGAIL SÁNCHEZ VILLACÍS

C.C.: 2101131262

DEDICATORIA

A mis padres, Aurora y Jorge, pilares fundamentales de mi vida. Mamá, tu amor incondicional y tu fortaleza han sido mi inspiración constante. Papá, tu sabiduría, tu ejemplo de trabajo duro y creativo es de admiración, encendiendo en mí la curiosidad por aprender e innovar. Este logro es tan suyo como mío.

A mi hermana Emily, por las risas compartidas y la felicidad de verte crecer. Tu presencia ilumina mi vida.

AGRADECIMIENTOS

Al concluir esta etapa de mi vida, quiero expresar mi más profundo agradecimiento a quienes hicieron posible este logro:

A mi familia, por su apoyo, sus palabras de aliento en los momentos más complicados, su paciencia durante todo este proceso y por creer en mí. Sin ustedes, nada de esto habría sido posible.

A mi prima, Ana Sánchez, por compartir su conocimiento, su experiencia; sus duras palabras y consejos fueron fundamentales para el desarrollo de este trabajo.

A la Unidad de Tecnologías de la Información de la PUCE-IBARRA, por abrirme sus puertas y permitirme desarrollar mi proyecto en sus instalaciones. Su apoyo en este proyecto tecnológico ha permitido el éxito de esta investigación.

A mi profesora de titulación, PhD. Laura Guerra Torrealba, de la Escuela de Informática e Inteligencia Artificial, por su paciencia y consejos, que me ayudaron a no perder el rumbo, completando exitosamente el proyecto.

A la PUCE-IBARRA, por ser mi alma mater y brindarme la oportunidad de crecer tanto personal como profesionalmente.

A todos aquellos que de una u otra manera contribuyeron a la realización de este trabajo, mi más sincero agradecimiento.

ÍNDICE DE CONTENIDOS

CERTIFICACIÓN TUTOR ..... **ii**
PÁGINA DE APROBACIÓN DEL TRIBUNAL ..... **iii**
ACTA DE CESIÓN DE DERECHOS ..... **iv**
AUTORÍA ..... **v**
RESUMEN ..... **xvi**
ABSTRACT ..... **xvii**
INTRODUCCIÓN **1**
OBJETIVOS 2
Objetivo General: 2
Objetivos Específicos: 2
CAPÍTULO I. ESTADO DEL ARTE **4**
1.1. Antecedentes 4
1.1.1. Antecedentes Internacionales 4
1.1.2. Antecedentes Nacionales. 5
1.2. Marco Teórico 5
1.2.1. Fundamentos de Kubernetes y Contenedores 5
1.2.2. Contenedores y su Relación con Kubernetes 8
1.2.3. Impacto de Kubernetes Para la Orquestación de Contenedores 8
1.2.4. Arquitectura Kubernetes 9
1.2.5. Implementación y Gestión de Clústeres Kubernetes 12
1.2.6. Despliegue de Aplicaciones en Kubernetes 15
1.2.7. Herramientas para el Monitoreo de Clústeres Kubernetes 16
1.2.8. Almacenamiento en Kubernetes 17
1.2.9. Modelo de Red en Kubernetes 17
CAPÍTULO II. MATERIALES Y MÉTODOS **19**
2.1. Aspectos Generales de la Investigación 19
2.2. Procedimiento de la Investigación 20
2.3. Fase de Planificación: 22
2.3.1. Recolección de Datos. 23

2.3.2. Determinación de Aplicaciones Contenerizadas. 25
2.3.3. Recursos Disponibles. 27
2.3.4. Análisis de los sistemas operativos compatibles con Docker y Kubernetes. 27
2.3.5. Diseño de la Arquitectura Kubernetes. 29
2.4. Fase de Desarrollo: 32
2.5. Fase de Entrega 32
2.6. Fase de Uso 32
CAPÍTULO III. RESULTADOS 33
3.1. Desarrollo del Entorno de Pruebas: 33
3.1.1. Instalación del Sistema Operativo en los Servidores 33
3.1.2. Instalación Inicial en los Servidores del Clúster Kubernetes Maestro - Workers 41
3.1.3. Instalación en el Nodo Maestro. 50
3.1.4. Instalación en los Nodos Worker 58
3.1.5. Despliegue del Dashboard, Instalación de Helm y Configuración de Metrics Server 60
3.1.6. Implementación de Aplicaciones en la Arquitectura 70
3.2. Entrega del Entorno de Pruebas: 75
3.2.1. Validación del Clúster 75
3.2.2. Formalización de la Transferencia 77
3.3. Uso del Entorno de Pruebas: 77
CONCLUSIONES 78
RECOMENDACIONES 80
REFERENCIA BIBLIOGRÁFICA 81
ANEXOS 84
ANEXO I 84
ANEXOS II 87
ANEXO III 88
ANEXO IV 89
ANEXO V 90
ANEXO VI 91
ANEXO VII 94
ANEXO VIII 95
ANEXO IX 97

ANEXO X 100

ÍNDICE DE TABLAS

Tabla 1 *Sugerencias de Seguridad para la Infraestructura de Kubernetes*..... 14

Tabla 2 *Cronograma Macro de la Fase de Planificación de la Metodología DevOps.*..... 22

Tabla 3 *Características del Servidor Base* 27

Tabla 4 *Comparación de Sistemas Operativos Compatibles con Docker y Kubernetes* 28

Tabla 5 *Recursos Asignados al Nodo Maestro* 31

Tabla 6 *Recursos Asignados a los Nodos Trabajadores* 31

ÍNDICE DE FIGURAS

Figura 1 *Diferencias Arquitectónicas: Virtualización vs. Contenedores* 7

Figura 2 *Arquitectura Básica de un Clúster Kubernetes* 11

Figura 3 *Flujo de Peticiones en un Clúster* 16

Figura 4 *Fases de la Metodología DevOps.* 21

Figura 5 *Arquitectura de Software [Eventos PUCE-I]* 26

Figura 6 *Arquitectura del Clúster Kubernetes* 30

Figura 7 *Elección del Idioma para el Sistema Operativo* 34

Figura 8 *Configuración del Teclado del Sistema Operativo* 35

Figura 9 *Pantalla de la Configuración de Red* 36

Figura 10 *Configuración Manual de la Red* 37

Figura 11 *Configuración del Almacenamiento para el Sistema Operativo* 38

Figura 12 *Pantalla de Resumen del Sistema de Archivos* 39

Figura 13 *Configuración del Perfil Inicial del Sistema Operativo* 40

Figura 14 *Pantalla de Instalación* 41

Figura 15 *Actualización del Sistema Operativo* 42

Figura 16 *Instalación de los Paquetes Https y Certificados* 42

Figura 17 *Creación de la Carpeta que Almacena Claves GPG e Instalación de la Clave GPG de Kubernetes* 43

Figura 18 *Instrucción para Añadir un Repositorio de Paquetes de Kubernetes y Actualización del Servidor* 43

Figura 19 *Instalación de los Componentes Esenciales para Trabajar con Kubernetes*..... 44

Figura 20 *Comando para Evitar las Actualizaciones No Deseadas*..... 44

Figura 21 *Versiones de las Herramientas de Kubernetes*..... 44

Figura 22 *Desactivar el Espacio de Intercambio Archivos o Dispositivos* 45

Figura 23 *Comentar la Línea de swap.img en el Archivo fstab*..... 45

Figura 24 *Módulos del Kernel modprobe overlay y modprobe br_netfilter*..... 45

Figura 25 *Modificar el Archivo /etc/sysctl.d/kubernetes.conf* 46

Figura 26 *Aplicar y Recargar la Configuración del Kernel*..... 46

Figura 27 *Configuraciones de los Módulos del Kernel Automáticos al Iniciar el Sistema*..... 47

Figura 28 *Instalación de Varios Paquetes y Dependencias* 47

Figura 29 *Clave para Verificar la Autenticidad de los Paquetes de Docker y Agregar Repositorio APT al Sistema*..... 48

Figura 30 *Instalación de Actualizaciones y Actualización del Sistema*..... 48

Figura 31 *Instalación de containerd.io*..... 49

Figura 32 *Restablecer, Iniciar y Verificar el Estado del Servicio containerd*..... 49

Figura 33 *Imágenes para el Clúster de Kubernetes en el Nodo Maestro*..... 50

Figura 34 *Ingresar la IP de los Servidores del Clúster para el Nodo Maestro y Trabajadores* . 51

Figura 35 *Inicialización del Clúster de Kubernetes* 52

Figura 36 *Salida Exitosa en la Creación del Control Plane* 52

Figura 37 *Acceso al Archivo de Configuración del Clúster Kubernetes y Cambio de la Propiedad del Archivo de Configuración para el Usuario Actual*..... 53

Figura 38 *Información Sobre el Clúster Kubernetes* 53

Figura 39 *Descarga del Archivo tigera-operator.yaml y Descarga del Archivo custom-resources.yaml* 54

Figura 40 *Creación de Recursos del Clúster de Kubernetes a partir del archivo tigera-operator.yaml* 55

Figura 41 *Verificación del Reemplazo en la Instancia cidr* 56

Figura 42 *Creación de Recursos del Clúster de Kubernetes a partir del Archivo custom-resources.yaml* 56

Figura 43 *Información de los Pods en Tiempo Real* 57

Figura 44 *Muestra los Pods en Estado: Running* 57

Figura 45 *Información del Nodo creado control-plane* 57

Figura 46 *Información Sobre los Pods e Información de los Servicios* 58

Figura 47 *Conexión Fallida* 59

Figura 48 *Comando kubeadm en Nodo Maestro* 59

Figura 49 *Comando kubeadm Para Registrar el Nodo Worker en el Clúster* 59

Figura 50 *Verificación de Comunicación entre los Nodos Worker y el Nodo Maestro* 60

Figura 51 *Verificación de la Instalación del Gestor de Paquetes Helm* 61

Figura 52 *Instalación del Dashboard a Través del Gestor de Paquetes Helm* 62

Figura 53 *Recursos de Kubernetes Creados por el Despliegue del Dashboard* 62

Figura 54 *Archivo values.yaml* 63

Figura 55 *Actualización del Dashboard Modificado a NodePort* 63

Figura 56 *Verificación del Tipo de Servicio NodePort*..... 63

Figura 57 *Acceso al Dashboard Mediante el Puerto 32481*..... 64

Figura 58 *Recursos Creados Después de Aplicar el Archivo user.yaml* 65

Figura 59 *Token Generado para el Usuario admin-user* 66

Figura 60 *Ingresando al Dashboard con el Token Generado* 66

Figura 61 *Página de Inicio del Dashboard* 67

Figura 62 *Descarga del Archivo components.yaml* 68

Figura 63 *Modificaciones Realizadas en el Archivo components.yaml*..... 69

Figura 64 *Salida del Comando kubectl para Visualizar los Recursos de Metrics Server* 70

Figura 65 *Métricas de los Nodos del Clúster* 70

RESUMEN

La presente investigación aborda la implementación de una infraestructura de Kubernetes para el despliegue de aplicaciones en contenedores en un entorno de pruebas dentro de la Unidad de Tecnologías de la Información de la Pontificia Universidad Católica del Ecuador Ibarra (PUCE-Ibarra). El objetivo principal es mejorar la eficiencia operativa y la calidad de los servicios de TI mediante la utilización de Kubernetes, centrándose en la portabilidad, fiabilidad y disponibilidad de las aplicaciones contenerizadas.

El estudio comienza con una fase de planificación, donde se recolecta y analiza información relevante para diseñar una arquitectura de clúster adecuada. Se evalúan los recursos disponibles y se seleccionan las aplicaciones a desplegar. Posteriormente, se desarrolla el entorno de pruebas, incluyendo la instalación y configuración del sistema operativo, Kubernetes, y herramientas adicionales para el monitoreo y gestión de contenedores.

La fase de entrega implica pruebas para validar la funcionalidad del clúster y su capacidad para gestionar aplicaciones en contenedores. Finalmente, se realiza la transferencia formal del entorno de pruebas al departamento de TI de la PUCE-Ibarra, asegurando que esté listo para su uso operativo.

Los resultados demuestran que la implementación de Kubernetes mejora significativamente la gestión de aplicaciones en contenedores, proporcionando un entorno más ágil, escalable y confiable. Esta solución beneficia no solo al personal de TI, sino también a docentes y estudiantes que dependen de aplicaciones tecnológicas para sus actividades académicas.

Palabras clave: Kubernetes, contenedores, entorno de pruebas, portabilidad, fiabilidad, disponibilidad.

ABSTRACT

This research addresses the implementation of a Kubernetes infrastructure for deploying containerized applications in a test environment within the Information Technology Unit of the Pontificia Universidad Católica del Ecuador, Ibarra (PUCE-Ibarra). The primary objective is to enhance operational efficiency and the quality of IT services through the use of Kubernetes, focusing on the portability, reliability, and availability of containerized applications.

The study begins with a planning phase, where relevant information is collected and analyzed to design an appropriate cluster architecture. Available resources are evaluated, and the applications to be deployed are selected. Subsequently, the test environment is developed, including the installation and configuration of the operating system, Kubernetes, and additional tools for monitoring and container management.

The delivery phase involves testing to validate the functionality of the cluster and its capacity to manage containerized applications. Finally, the formal transfer of the test environment to the IT department of PUCE-Ibarra is carried out, ensuring it is ready for operational use.

The results demonstrate that the implementation of Kubernetes significantly improves the management of containerized applications, providing a more agile, scalable, and reliable environment. This solution benefits not only IT personnel but also faculty and students who rely on technological applications for their academic activities.

Keywords: Kubernetes, containers, test environment, portability, reliability, availability.

INTRODUCCIÓN

Durante las últimas décadas, el mundo ha sido testigo de constantes cambios en los entornos descritos por los marcos de la infraestructura en Tecnologías de la Información (TI). Desde la tecnología centralizada en computadoras masivas, pasando por la llegada de internet que revolucionó la conectividad, hasta la actualidad donde el paradigma ha cambiado gracias al surgimiento de recursos compartidos por la red o "nubes" y sistemas virtualizados. Esta evolución se debe a que, según Fernández (2017), factores como el procesamiento de grandes volúmenes de datos (Big Data), la creciente presencia del Internet de las Cosas (IoT), la ubicuidad de los sistemas e incluso los modelos de virtualización de sistemas y aplicaciones a través de la nube, ha ocasionado situaciones en las que las infraestructuras de TI tradicionales ya no son compatibles y no pueden adaptarse a los cada vez más exigentes escenarios emergentes.

En consecuencia, ha surgido una creciente necesidad de desplegar aplicaciones, servicios y microservicios de manera más ágil, reduciendo costos y optimizando recursos. Esto demanda una mejora constante en las tecnologías de vanguardia. Un destacado ejemplo de estas tecnologías es Kubernetes, una plataforma de código abierto que, según resaltan Delgado Pedraza & Pineda Parra (2019), brinda una organización estructurada a los contenedores que conforman una aplicación, lo que facilita su gestión y permite ejercer un control efectivo sobre ellos.

La Pontificia Universidad Católica del Ecuador - Ibarra (PUCE-I) es una reconocida institución académica que se esfuerza por mantenerse a la vanguardia en la educación superior e impulsar la investigación. Como lo menciona Aragón Puetate (2018), la universidad responde a las necesidades formativas de pregrado y posgrado en la provincia. En este contexto, la Unidad de Tecnologías de la Información desempeña un rol fundamental dentro de la PUCE-I, dedicándose al desarrollo de proyectos que contribuyen a mejorar el entorno universitario y abordar desafíos tecnológicos emergentes.

Es necesario destacar que dentro de la PUCE-I, la Unidad de Tecnologías de la Información se ha enfrentado a desafíos en la gestión de su infraestructura tecnológica a lo largo de los años, como la escalabilidad y elasticidad para satisfacer la demanda, desplegar las aplicaciones, recuperación ante desastres y complejidad de gestionar la parte operativa al diversificar la infraestructura. Hoy

en día, han adoptado la tecnología de contenedores, que permite la separación virtual entre las aplicaciones, así como simplifica su despliegue. Sin embargo, la gestión manual de los contenedores ha ocasionado interrupciones en la disponibilidad continua de los servicios cuando el contenedor en el cual se aloja la aplicación se detiene, dando posibles problemas en la continuidad del negocio.

Es por ello que este trabajo tuvo como propósito implementar un entorno de pruebas con infraestructura basada en la plataforma de Kubernetes utilizando una aplicación de prueba contenerizada en Docker dentro del clúster. Esta iniciativa no solo beneficia al personal de la Unidad de TI, sino también a docentes y estudiantes que utilizan aplicaciones en su experiencia académica diaria. Además, el uso de servicios o aplicaciones portables y fiables beneficia al personal administrativo y contribuye a la continuidad operativa de la PUCE-I.

Finalmente, con esta investigación se buscaba dar respuesta a la siguiente interrogante: "¿Cómo mejorar la eficiencia operativa y la calidad de los servicios de Tecnología de la Información y gestión de sistemas mediante la implementación de Kubernetes en la portabilidad, disponibilidad y fiabilidad de aplicaciones contenerizadas en Docker?".

Con esto en mente, a continuación, se presentan los objetivos del estudio.

OBJETIVOS

Objetivo General:

- Implementar un entorno de pruebas con infraestructura Kubernetes para la ejecución de aplicaciones en contenedores para la Unidad de Tecnologías de la Información de la PUCE-Ibarra.

Objetivos Específicos:

- Analizar la metodología a aplicarse para la implementación de una arquitectura de contenedores utilizando las mejores prácticas vigentes, revisando la literatura científica actualizada.

- Analizar el orquestador de contenedores a implementar, con énfasis en la portabilidad, fiabilidad y disponibilidad de aplicaciones, revisando la literatura científica actualizada.

- Definir la arquitectura Kubernetes en base a las necesidades y recursos disponibles de la Unidad de Tecnologías de la Información de la PUCE-I.

- Implementar la plataforma Kubernetes en los servidores de la Unidad de Tecnologías de la Información, siguiendo los requisitos de la arquitectura definida.

- Implementar un orquestador de contenedores con una interfaz gráfica para facilitar la administración y visualización de la plataforma Kubernetes.

- Evaluar la infraestructura implementada y la capacidad de la arquitectura, mediante pruebas de validación en un entorno de pruebas.

El presente documento se estructura en tres (3) capítulos: El primero aborda la consulta bibliográfica y conceptual que fundamenta la investigación. El segundo capítulo se centra en la descripción de las herramientas tecnológicas, la metodología del proyecto, el desarrollo del entorno de pruebas con infraestructura Kubernetes. Finalmente, el tercer capítulo presenta los resultados obtenidos tras haber culminado el proyecto, dar respuesta a la pregunta de investigación y realizar la evaluación de la utilización de Kubernetes en la Unidad de Tecnologías de la Información de la PUCE-I.

CAPÍTULO I. ESTADO DEL ARTE

Dentro de este capítulo se presenta una revisión de la literatura existente y los trabajos previos relacionados con la implementación de infraestructura Kubernetes para el despliegue de aplicaciones en contenedores. Se abordan los antecedentes tanto a nivel internacional como nacional, con el fin de comprender el estado actual del arte y los avances más relevantes en esta área. Además, se analizan los fundamentos teóricos y conceptuales necesarios para contextualizar el problema de investigación y sentar las bases del proyecto.

1.1. Antecedentes

1.1.1. Antecedentes Internacionales

En el ámbito internacional, Packard et al. (2021), han llevado a cabo una implementación de la tecnología Kubernetes en clústeres multi-master y self-hosted. Los autores resaltan que la experiencia con Kubernetes fue en gran medida exitosa, permitiendo probar de manera rápida e iterativa los componentes, además mencionan que la administración de esta plataforma de orquestación de contenedores es relativamente sencilla y ofrece beneficios significativos para el personal que utiliza un enfoque Devops.

Por otra parte, los investigadores concluyen que uno de los principales beneficios es que Kubernetes fomenta la adopción de mejores prácticas de reproducibilidad, obligando a los desarrolladores a definir la instalación, implementación y almacenamiento en formatos de lectura fáciles, como los archivos YAML, antes de desplegarlos en el clúster. Esta práctica estandariza y permite mejorar la gestión de configuración de las aplicaciones.

En esta misma línea, el trabajo de Paredes Calle, (2021) resalta que la implementación de microservicios en un clúster de Kubernetes es una práctica recomendada para mejorar la escalabilidad en la arquitectura de las aplicaciones. Kubernetes cuenta con políticas de escalamiento automático y es líder en la orquestación de contenedores.

Poniszewska-Maranda y Czechowska (2021), en su estudio sobre el despliegue de clústeres Kubernetes, señalan la importancia de automatizar las operaciones del clúster a través de script, siguiendo buenas prácticas de “infraestructura como código”.

1.1.2. Antecedentes Nacionales.

Ecuador ha realizado esfuerzos para promover la implementación de tecnologías emergentes como los contenedores y las plataformas de orquestación. En este contexto, el trabajo de Montes Bravo (2022) tuvo como propósito implementar servidores web con hosts virtuales utilizando la plataforma de orquestación de contenedores Minikube, basada en Kubernetes. Los resultados permitieron concluir que esta tecnología mejora la escalabilidad y la respuesta ante las demandas de los hosts virtuales, brindando mayor confiabilidad y disponibilidad.

Los trabajos investigados resaltan que Kubernetes mejora significativamente la escalabilidad y disponibilidad de las aplicaciones, gracias a sus políticas de escalamiento automático y su enfoque de orquestación de contenedores. Además, fomenta la adopción de mejores prácticas de reproducibilidad y estandarización en el desarrollo y despliegue de aplicaciones, al obligar a los desarrolladores a definir la instalación, implementación y almacenamiento en formatos de lectura fácil, como los archivos YAML.

Por otro lado, los estudios enfatizan la importancia de automatizar las operaciones del clúster Kubernetes a través de scripts, siguiendo buenas prácticas de "infraestructura como código", lo que facilita la gestión y mantenimiento de la infraestructura.

En el contexto nacional, los esfuerzos realizados en Ecuador para promover la implementación de tecnologías emergentes como los contenedores y las plataformas de orquestación han arrojado resultados positivos.

1.2. Marco Teórico

1.2.1. Fundamentos de Kubernetes y Contenedores

Contenedores. Microsoft (2023) describe a un contenedor como un “un silo aislado y ligero para ejecutar una aplicación en el sistema operativo host”. Dicho en otras palabras, un contenedor es un espacio separado y eficiente destinado a ejecutar una aplicación en el sistema operativo principal, aprovechando el kernel de este sistema operativo.

Los contenedores se fundamentan en la virtualización a nivel de sistema operativo, lo cual difiere de la virtualización tradicional de máquinas virtuales. En lugar de virtualizar el hardware completo, los contenedores comparten el kernel del sistema operativo host.

Esta característica de compartir el kernel del host es lo que permite que los contenedores sean más livianos y eficientes. Al no requerir virtualizar el hardware completo, los contenedores pueden aprovechar los recursos del sistema anfitrión de manera más óptima, lo que los hace ideales para despliegues ágiles de aplicaciones.

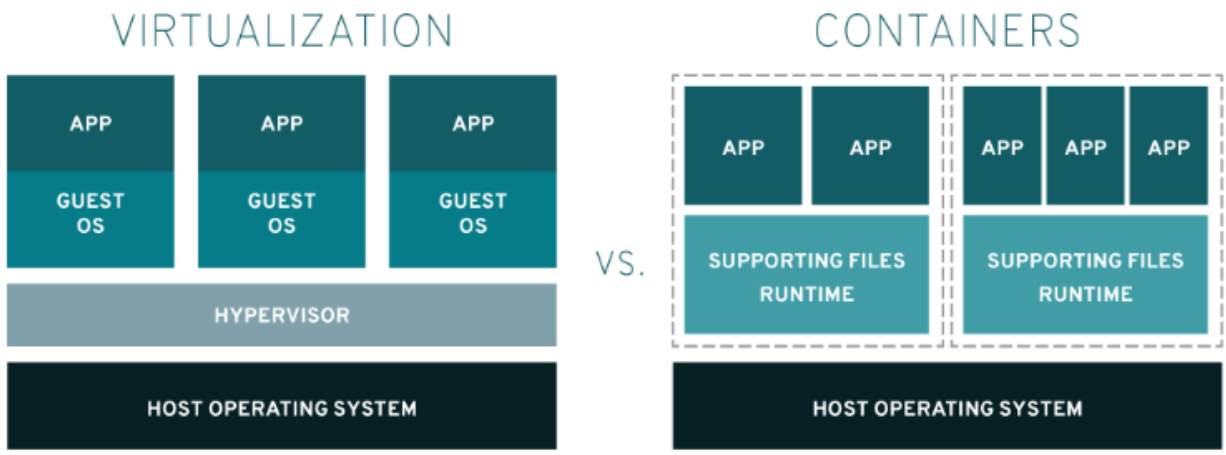
Contenerización. IBM (s.f) presenta la definición de contenerización como "empaquetado de código de software con solo las bibliotecas de sistema operativo (SO) y las dependencias necesarias para ejecutar el código para crear un único ejecutable ligero, denominado contenedor, que se ejecuta de manera coherente en cualquier infraestructura".

Esta definición enfatiza la ligereza de los contenedores al empaquetar únicamente las bibliotecas y dependencias necesarias, lo que los hace más eficientes en términos de recursos. Además, la portabilidad y la capacidad de ejecución coherente en cualquier infraestructura son características fundamentales que facilitan el despliegue y la escalabilidad de las aplicaciones.

Diferencias Entre Contenedores y Máquinas Virtuales. Tanto los contenedores como las máquinas virtuales son soluciones tecnológicas que posibilitan separar las aplicaciones de los recursos de la infraestructura de TI.

Como se ilustra en la **¡Error! No se encuentra el origen de la referencia.**, la virtualización, se ejecuta sobre un sistema operativo hospedador (*host*), cada una con su propio sistema operativo *guest* en el que se encuentran las aplicaciones, es decir, implica la copia digital de una máquina física. En cambio, los contenedores comparten el mismo sistema operativo *host*, pero cada contenedor empaqueta una aplicación junto con sus archivos y dependencias de soporte en un entorno aislado y portátil conocido como *runtime*.

Figura 1
Diferencias Arquitectónicas: Virtualización vs. Contenedores



Nota. Muestra las arquitecturas de virtualización y de contenedores. Tomado de Containers vs VMs, de Red Hat, 2023, https://www.redhat.com/en/topics/containers/containers-vs-vms

Kubernetes. Kubernetes es una plataforma de código abierto, portable y extensible diseñada para automatizar el despliegue de aplicaciones de manera sencilla. Según Kubernetes Documentation (2023), esta solución se basa en una configuración declarativa, lo que significa que el usuario proporciona una descripción detallada del estado deseado del sistema, y la plataforma se encarga de interpretar esta especificación y realizar las acciones necesarias para alcanzar ese estado.

Al ser un proyecto de código abierto, Kubernetes cuenta con un ecosistema en constante crecimiento que facilita el acceso a servicios, documentación, soporte y herramientas relacionadas. Este ecosistema enriquecido contribuye a su uso generalizado.

Esta solución se ha consolidado como líder en la orquestación de contenedores, gracias a su arquitectura robusta y escalable. En su núcleo, Kubernetes se fundamenta en la tecnología de contenedores, como Docker. Esta última, según su página oficial Docker (s.f.), facilita el ciclo de vida del desarrollo de aplicaciones al brindar una forma estandarizada y portable de empaquetar y distribuir software.

Una de las características clave de Kubernetes es su arquitectura de clústeres, donde un conjunto de servidores, ya sean físicos o virtuales, actúan como nodos trabajadores (workers) que se unen para formar una infraestructura unificada y gestionada por un plano de control (control-plane) centralizado.

Otra característica fundamental es su capacidad de escalabilidad horizontal, tanto en el número de nodos como en la cantidad de cargas de trabajo. Gracias a su arquitectura distribuida y mecanismos de escalamiento automático, puede adaptarse de manera dinámica a las demandas cambiantes de recursos, asegurando una mejora en la disponibilidad de las aplicaciones.

Finalmente, es importante saber que esta solución, además de ser una plataforma de orquestación de contenedores, también se ha convertido en un estándar para la industria del desarrollo de software y la infraestructura, especialmente en la nube. Su popularidad se debe a su capacidad de simplificar la gestión de aplicaciones en entornos distribuidos, que ofrece una amplia gama de servicios, herramientas y soporte a los usuarios.

1.2.2. Contenedores y su Relación con Kubernetes

La relación entre estos temas es fundamental, debido a que Kubernetes surgió como una solución que gestiona, automatiza, orquesta el despliegue, el escalamiento y la administración de aplicaciones empaquetadas en contenedores.

En un entorno de Kubernetes, los contenedores se agrupan en unidades lógicas llamadas Pods, que representan la unidad básica de la arquitectura. Estos Pods se ejecutan en nodos trabajadores dentro de un clúster de Kubernetes. Esta plataforma introduce una capa de abstracción sobre los contenedores, proporcionando recursos como deployments, services y otros. De esta manera, la combinación de contenedores y Kubernetes ha permitido a las organizaciones adoptar prácticas de desarrollo y despliegue más ágiles, escalables y eficientes.

1.2.3. Impacto de Kubernetes Para la Orquestación de Contenedores

A medida que los entornos se vuelven más complejos y escalables, la necesidad de una solución de orquestación de contenedores se ha vuelto crucial. En este sentido, Kubernetes ha tenido un impacto significativo y positivo en esta área. En primer lugar, permite la portabilidad de las

aplicaciones, mejorando la disponibilidad de los contenedores y las aplicaciones que se encuentran dentro.

Como confirma el trabajo de Pan et al. (2019), en el que realiza una comparación entre distintos orquestadores de contenedores, incluyendo a Kubernetes y Docker Swarm, Kubernetes se destaca por su flexibilidad, potencia y su rico conjunto de funciones, lo que lo convierte en la opción preferida para manejar requisitos complejos de orquestación de contenedores.

1.2.4. Arquitectura Kubernetes

Una arquitectura de Kubernetes es desarrollada de acuerdo al tipo del negocio, tomando en cuenta las necesidades y recursos disponibles de la empresa. En este contexto, el ambiente de pruebas es desarrollado para la Pontificia Universidad Católica del Ecuador - Ibarra, específicamente para la Unidad de TI. Debido a la limitación de recursos disponibles para este proyecto, se opta por utilizar una arquitectura de clúster básico.

La arquitectura de clúster básico de Kubernetes es una solución adecuada cuando los recursos de la organización son limitados. Esta configuración permite desplegar y gestionar aplicaciones de manera eficiente, sin requerir una infraestructura compleja. Al utilizar este enfoque, la Unidad de TI de la Pontificia Universidad Católica del Ecuador podrá aprovechar los beneficios de Kubernetes, como la disponibilidad y la facilidad de administración, adaptándose a las necesidades y restricciones de su entorno. A continuación, se detallan los principales conceptos de la tecnología Kubernetes:

- Nodos maestro o control-plane

Conocido como nodo maestro, es un servidor físico o virtual que alberga una colección de servicios que lo conforman. Según lo expresa Poulton (2023), estos servicios del sistema ejecutados en el nodo maestro son los responsables de gestionar, supervisar y administrar toda la infraestructura del clúster.

En otras palabras, el nodo maestro actúa como el cerebro de la operación de Kubernetes, orquestando y coordinando las diversas tareas y recursos del clúster. Para llevar a cabo esta función

crítica, el control-plane interactúa con varios componentes clave, tales como el kube-api-server y el kube-scheduler, entre otros. Seguidamente, se detallan estos componentes:

  - kube-controller-manager: ejecuta los controladores necesarios para Kubernetes como el controlador de replicación, nodos, endpoints y tokens.

  - kube-apiserver: es el componente central para todas las solicitudes y peticiones realizadas al clúster.

  - kube-scheduler: monitorea los pods que no tienen asignado un nodo worker y selecciona el adecuado de acuerdo a factores como los recursos, software y/o restricciones de hardware.

  - etcd: Es una base de datos distribuida en donde se almacena toda la información relacionada con el clúster de Kubernetes.

- Nodos worker

En la arquitectura de Kubernetes, un nodo worker se refiere a un servidor individual cuya función primordial es ejecutar los pods asignados. Actúan como las unidades de procesamiento del clúster, encargándose de la ejecución efectiva de las cargas de trabajo empaquetadas en pods.

Cada nodo worker cuenta con componentes específicos de Kubernetes, como kubelet y kube-proxy, que operan bajo la supervisión y coordinación del nodo maestro o control-plane. Estos componentes son:

  - kubelet: es el “agente nodo” principal que se ejecuta en cada nodo y garantiza que los contenedores descritos bajo un objeto YAML o JSON (PodSpec) estén en buen estado.

  - kube-proxy: se encarga del reenvío de conexiones de flujo TCP, UDP y SCTP.

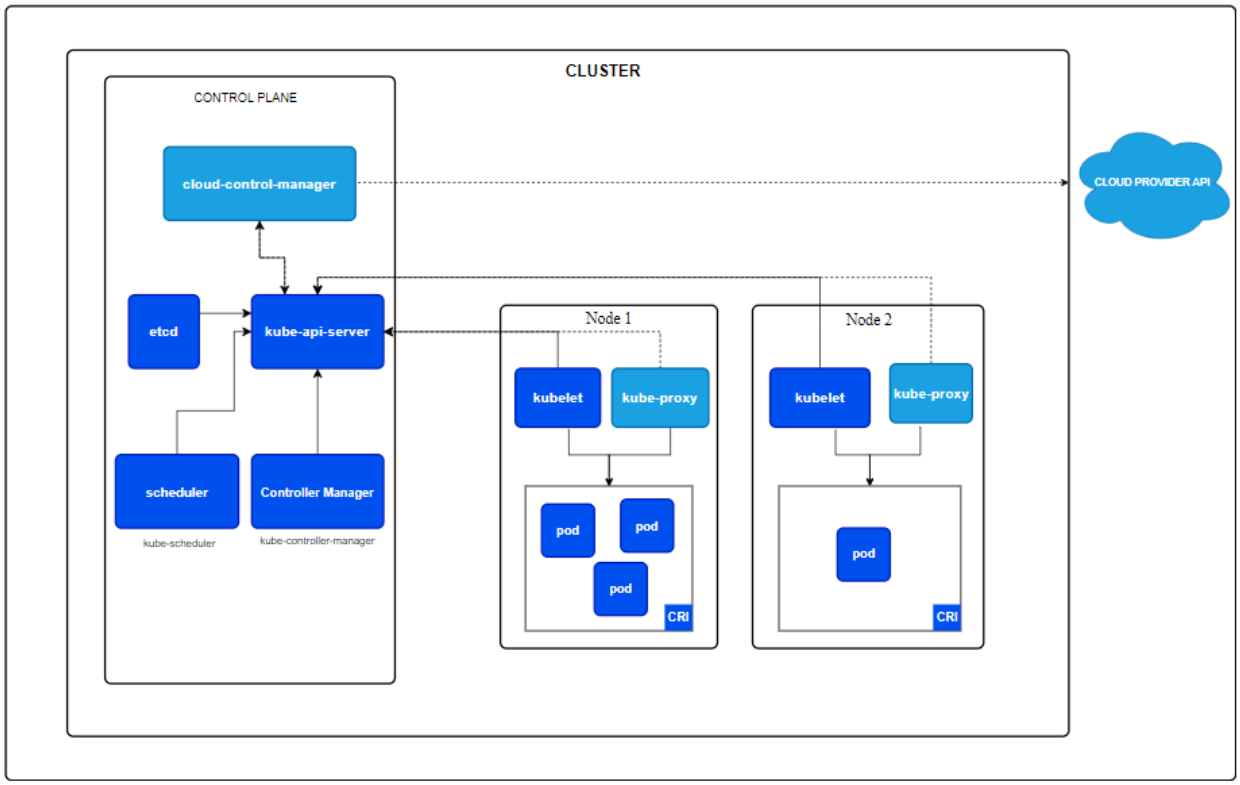
  - Container runtime: es el software encargado de los contenedores, es decir, permite que se inicien, se ejecuten y se detengan de manera adecuada.

- Pods

De acuerdo con Vohra (2016, p.40), un pod se define como una unidad indivisible que agrupa uno o más contenedores que comparten recursos y una misma IP. Esta estructura permite que los contenedores dentro de un pod se comuniquen de manera eficiente, al mismo tiempo que mantienen un aislamiento adecuado con respecto a otros pods en el clúster.

En la **¡Error! No se encuentra el origen de la referencia.**, se ilustra la arquitectura básica de un clúster en el contexto de Kubernetes. Según Sayfan (2017, p.4), un clúster se define como un conjunto de servidores, recursos de almacenamiento y capacidad de red que se emplean de manera conjunta para la ejecución de diversas tareas que conforman un sistema completo. Cabe mencionar que un sistema de gran escala puede estar compuesto por múltiples clústeres.

Figura 2
Arquitectura Básica de un Clúster Kubernetes



*Nota.*Recuperado de Cluster Architecture, de The Linux Foundation, 2023, Kubernetes (https://kubernetes.io/docs/concepts/architecture/). CC BY 4.0

Un clúster de Kubernetes está compuesto principalmente por un conjunto de nodos worker y al menos un nodo maestro o control-plane. Estos componentes se comunican de manera constante y coordinada a través de una red virtual que los interconecta, permitiendo el intercambio seguro y eficiente de información.

El control plane actúa como el cerebro de las operaciones para gestión y orquestación de todo el clúster. Este componente central interactúa con varios elementos clave, como el kube-api-server, que actúa como puerta de entrada para todas las solicitudes y operaciones en el clúster; el kube-scheduler, responsable de programar y asignar los pods (unidades básicas de despliegue) a los nodos trabajadores adecuados; y el kube-controller-manager, que supervisa el estado del clúster y toma medidas correctivas en caso de ser necesario.

Por otro lado, los nodos trabajadores son los encargados de ejecutar las cargas de trabajo empaquetadas en pods. Cada nodo cuenta con componentes como kubelet, que se comunica con el control-plane para recibir instrucciones y reportar el estado del nodo, y kube-proxy, que facilita la comunicación entre los pods y el exterior del clúster.

Además, Kubernetes utiliza el concepto de etcd, un almacén de datos distribuido y confiable que mantiene el estado actual del clúster. Todos los cambios en el clúster, como la creación, modificación o eliminación de recursos, se registran en etcd, lo que garantiza la consistencia y la capacidad de recuperación en caso de fallas.

1.2.5. Implementación y Gestión de Clústeres Kubernetes

Consideraciones Técnicas y Requerimientos. Antes de iniciar la implementación de un clúster Kubernetes, es esencial realizar un análisis de los requerimientos técnicos y las necesidades específicas del entorno. Para este proyecto se tomó en cuenta algunas consideraciones clave que son:

Arquitectura de la aplicación: Conocer la arquitectura de las aplicaciones que se desplegarán en el clúster y requisitos de recursos.

Infraestructura subyacente: Decidir si se utilizará una infraestructura en la nube, on-premises o híbrida. Para este caso, se escogió la infraestructura on-premises debido a los recursos disponibles en la Unidad de TI.

Almacenamiento persistente: Evaluar las necesidades de almacenamiento persistente para las aplicaciones, ya sea mediante volúmenes locales, proveedores de almacenamiento en la nube o sistemas de almacenamiento de red.

Requisitos de red: Entender los requisitos de red, como el enrutamiento y el acceso externo. Kubernetes ofrece soluciones integradas para la gestión de redes.

Para el tema de redes y seguridad de las mismas, existen los Container Network Interface (CNI) que son plugins o complementos necesarios para administrar la red, existen varios, los más conocidos son: Flannel, Calico, Weave Net, Cilium, entre otros.

Monitoreo: Kubernetes proporciona un dashboard propio que permite visualizar el estado y los procesos que se llevan a cabo dentro del clúster de Kubernetes, permitiendo detectar problemas, cabe mencionar que también hay soluciones de terceros.

Configuración Inicial de un Clúster Kubernetes. Según las prácticas recomendadas de configuración (2023) de Kubernetes, se detallan los primeros pasos de la configuración:

Se debe instalar y configurar el entorno de ejecución de contenedores, conocido como container runtime, en todos los nodos que formarán parte del clúster. Es crucial asegurarse de que los nodos cumplan con los requisitos mínimos de hardware y sistema operativo establecidos por Kubernetes.

Una vez que los nodos estén listos, el siguiente paso es instalar Kubernetes en sí. Existen varios métodos de instalación disponibles, como kubeadm, minikube (para entornos de desarrollo) o herramientas específicas de proveedores de nube. El proceso implica inicializar el clúster con el nodo maestro (master node) y unir los nodos trabajadores (worker nodes) al clúster.

Después de la instalación, es necesario configurar la red de contenedores utilizando una solución CNI ya que permite la comunicación entre los diferentes componentes del clúster. Además, se debe configurar el enrutamiento de ingreso y salida para exponer los servicios.

Una vez que la red esté configurada, es recomendable instalar complementos esenciales, como monitoreo de recursos y, por último, es importante realizar pruebas para asegurar un entorno correctamente configurado.

Estrategias de Aprovisionamiento y Escalado de Nodos. Microsoft Azure (s.f.) en la página en la que discute sobre el escalado vertical frente a escalado horizontal y habla sobre estos conceptos, siendo el escalado o reducción vertical como el proceso de disminuir los recursos asignados a un nodo individual, como CPU, memoria RAM o almacenamiento. Así mismo, habla sobre el escalado horizontal como agregar o eliminar nodos del clúster.

Ambos enfoques tienen sus ventajas y desafíos. El escalado vertical puede ser una solución rápida y eficiente a corto plazo, pero también tiene limitaciones en cuanto a la capacidad máxima de un solo nodo. Por otro lado, el escalado horizontal ofrece una mayor flexibilidad y escalabilidad, pero puede requerir una planificación y coordinación más cuidadosa.

El escalado y aprovisionamiento de nodos en un entorno on-premise puede ser un proceso más complejo y requerir una planificación minuciosa, ya que implica la adquisición y configuración de hardware físico.

Seguridad en Clústeres Kubernetes. La seguridad del clúster y sus aplicaciones es una consideración primordial en la implementación para proteger los datos y las cargas de trabajo.

En la Tabla 1, se muestran algunas sugerencias para proteger el clúster de Kubernetes de acuerdo al área de interés.

Tabla 1
Sugerencias de Seguridad para la Infraestructura de Kubernetes

Área de interés para la infraestructura de Kubernetes	Sugerencias
Acceso de red al Plano de Control	El acceso público al plano de control de Kubernetes en Internet está restringido y controlado mediante listas de control de acceso a direcciones IP específicas.
Acceso a la red de los Nodos	Los nodos deben configurarse para aceptar conexiones únicamente desde el plano de control en puertos específicos.

Acceso a etcd	El acceso a etcd (base de datos de Kubernetes) debe limitarse al plano de control y, preferiblemente, utilizar TLS para las comunicaciones seguras.
Encriptación etcd	Es una buena práctica encriptar todas las unidades de almacenamiento de etcd, ya que contiene el estado completo del clúster.

Nota. Adaptado de *Vista general de Seguridad Cloud Native*, Kubernetes, 2023, Kubernetes (https://kubernetes.io/es/docs/concepts/security/overview/)

Estas características de seguridad a nivel de infraestructura implican múltiples capas de protección a los componentes, limita los permisos de acceso y reduce el riesgo de compromisos de seguridad en los clústeres.

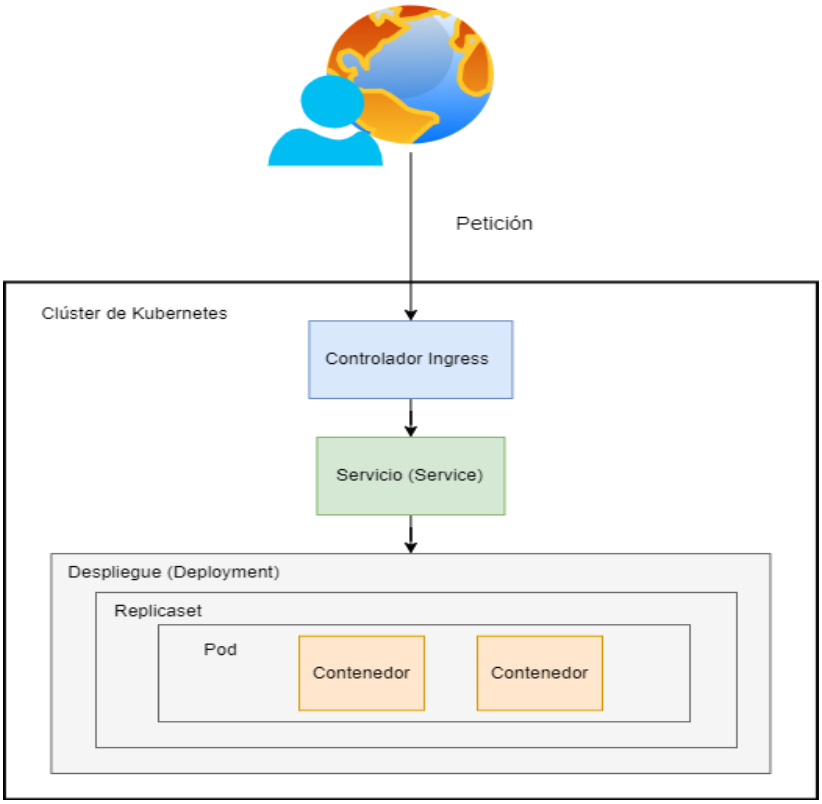
1.2.6. Despliegue de Aplicaciones en Kubernetes

Desplegar aplicaciones en esta plataforma es un proceso clave que usa el componente Deployment. Un Deployment define mediante un archivo declarativo YAML o JSON el estado deseado de la aplicación, incluyendo el número de réplicas, los recursos asignados, entre otros. Kubernetes se encarga de mantener el estado deseado realizando las acciones necesarias.

Además de los Deployments, Kubernetes también ofrece otros objetos útiles para el despliegue, como los Services y los Ingress. Un Service actúa como un balanceador de carga interno, proporcionando acceso a los Pods de una aplicación mediante una dirección IP estable y un nombre de DNS. El Ingress, por otro lado, permite exponer los servicios de la aplicación a través de un punto de entrada único, facilitando el enrutamiento del tráfico.

En la Figura 3 se muestra el flujo de una petición externa hacia una aplicación desplegada en un clúster de Kubernetes, mostrando la relación jerárquica entre estos componentes y cómo Kubernetes orquesta y administra el ciclo de vida de las aplicaciones.

Figura 3
Flujo de Peticiones en un Clúster



Nota. Adaptado de *Ingress*, Kubernetes, 2023, Kubernetes (https://kubernetes.io/docs/concepts/services-networking/ingress/)

1.2.7. Herramientas para el Monitoreo de Clústeres Kubernetes

Monitorear los clústeres es fundamental para garantizar la disponibilidad y fiabilidad de las aplicaciones desplegadas. Existen varias interfaces de terceros que permiten visualizar el estado de los contenedores de las aplicaciones ejecutadas, así como administrar los recursos del clúster de manera gráfica; sin embargo, Kubernetes tiene una propia interfaz que es liviana y permite visualizar los recursos como nodos, pods, implementaciones, servicios y más. Además, ofrece la capacidad de realizar operaciones básicas de administración, como escalar o reiniciar implementaciones, crear o eliminar recursos, y acceder a los registros de los contenedores para tener una visibilidad amplia del estado de las aplicaciones, facilitando la detección y resolución de problemas.

1.2.8. Almacenamiento en Kubernetes

El almacenamiento en Kubernetes debe configurarse correctamente para un buen funcionamiento de las aplicaciones en la infraestructura.

Es bien sabido que cuando un contenedor está en funcionamiento los datos de la aplicación se mantienen; sin embargo, cuando el contenedor se detiene o se reinicia los datos almacenados se pierden. Por ello Kubernetes ha introducido el concepto de volúmenes que, según la documentación en el apartado Volumes (2024), es un directorio que se monta dentro de uno o más contenedores en un mismo Pod.

Kubernetes soporta varios tipos de volúmenes, pero se caracterizan generalmente por ser persistentes o efímeros, entonces, cuando un pod se elimina junto con los contenedores dentro, esta tecnología borra los volúmenes efímeros y es capaz de mantener los volúmenes persistentes.

1.2.9. Modelo de Red en Kubernetes

El modelo de red es esencial para la comunicación entre los componentes del clúster. En este modelo, cada Pod tiene su propia dirección IP, lo que elimina la necesidad de enlaces entre pods y puertos. Además, la comunicación de pod a pod se logra a través de direcciones IP únicas, lo que permite la comunicación directa entre pods, incluso en diferentes nodos del clúster.

Según IBM (s.f.), en su web dedicada a la explicación del modelo de red, Kubernetes ofrece cuatro tipos de comunicación básica:

1. Redes de contenedor a contenedor. - Permiten la comunicación dentro de un mismo pod, utilizando localhost para compartir recursos de red.

2. Redes de pod a pod. - Facilitan la comunicación entre pods en el mismo nodo o en diferentes nodos, utilizando direcciones IP únicas y un servicio DNS para asignar nombres a los pods.

3. Redes de pod a servicio. - Definen un conjunto lógico de pods y permiten la exposición externa del tráfico a través de servicios, como ClusterIP, NodePort y LoadBalancer.

4. Redes externas al servicio. - Permiten el acceso a servicios externos desde fuera del clúster.

CAPÍTULO II. MATERIALES Y MÉTODOS

En este capítulo, se presentan los aspectos generales del proyecto, detallando las fases de Planificación y Desarrollo de la metodología elegida. Además, se incluyen algunas consideraciones relevantes que deben tenerse en cuenta para la implementación de la infraestructura Kubernetes para la implementación de aplicaciones basadas contenedores, acorde con el alcance establecido.

2.1. Aspectos Generales de la Investigación

El alcance de este proyecto se centró en la implementación de un prototipo de infraestructura Kubernetes en entorno de pruebas para la ejecución de algunas aplicaciones en contenedores en la Unidad de Tecnologías de la Información de la PUCE-Ibarra. Este entorno de pruebas tiene como propósito principal evaluar la viabilidad y los beneficios de adoptar Kubernetes como solución para abordar los desafíos actuales en la gestión de aplicaciones en contenedores.

El tipo de investigación en la que se enmarca el desarrollo del presente trabajo de titulación es aplicado, en el contexto de las nuevas tecnologías para el despliegue de aplicaciones web en diferentes entornos.

El enfoque utilizado fue cualitativo, lo cual permitió entender los aspectos relacionados con hardware, software, redes, infraestructura como código (IaC) y seguridad, mediante datos e información que se recopilaron del personal técnico de la Unidad de Tecnologías de la Información.

La Unidad de Tecnologías de la Información de la PUCE-Ibarra ha enfrentado diversos desafíos en la gestión de su infraestructura tecnológica a lo largo de los años. Entre los principales desafíos se encuentran la dificultad en el despliegue ágil y eficiente de aplicaciones, la falta de capacidad de recuperación ante fallas o situaciones de desastre, y el aumento de la complejidad operativa a medida que la infraestructura se diversifica.

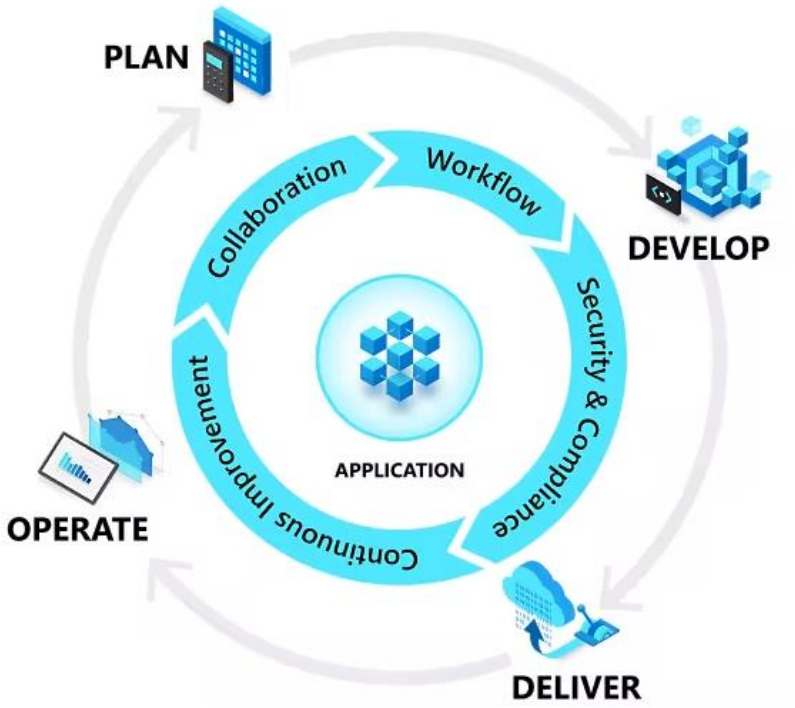
2.2. Procedimiento de la Investigación

Para llevar a cabo la implementación de la infraestructura de Kubernetes para la ejecución de aplicaciones en contenedores de manera eficiente, se adoptó una metodología ágil. Esta metodología se basa en los principios de DevOps, los cuales promueven una colaboración estrecha entre los equipos de operaciones y desarrollo.

De acuerdo al sitio de Microsoft Azure (2018) en el que habla sobre DevOps, establece cuatro fases primordiales: planificación, desarrollo, entrega y uso. En el contexto de este trabajo, centrado en la parte operativa, las fases de la metodología mencionada se adaptan al análisis de la plataforma y herramientas, recursos disponibles (hardware, software, redes), diseño de arquitectura, instalación, configuración y pruebas de la infraestructura, en lugar del desarrollo tradicional de aplicaciones. Estos elementos van a garantizar que el entorno de pruebas sea eficiente y sencillo de mantener.

A continuación, en la Figura 4, se presentan las fases de la metodología:

Figura 4
Fases de la Metodología DevOps.



Nota. Tomado de Microsoft Azure (2018).

Dado el objetivo de "Implementar un entorno de pruebas con infraestructura Kubernetes para la ejecución de aplicaciones en contenedores para la Unidad de Tecnologías de la Información de la PUCE-Ibarra", se establece que DevOps es la metodología más adecuada debido a que es altamente compatible con la tecnología Kubernetes, la flexibilidad y escalabilidad es esencial en un entorno de pruebas que está en constante cambio y la adaptabilidad a nuevas tecnologías.

De acuerdo con las fases de la metodología DevOps mencionada anteriormente, el procedimiento de la investigación se estructuró de la siguiente manera:

2.3. Fase de Planificación:

Durante la fase de planificación, se llevó a cabo un análisis que incluyó la recolección de datos, la identificación de los recursos disponibles, la determinación de las aplicaciones contenerizadas y la definición de la arquitectura de Kubernetes a implementar.

Dentro de esta fase, se llevaron a cabo las siguientes actividades según el siguiente cronograma macro (ver Tabla 2). En el Anexo I se detallan las actividades y duración específicas de todas las fases de la metodología.

Tabla 2
Cronograma Macro de la Fase de Planificación de la Metodología DevOps.

Actividad	Descripción	Duración
1. Recolección de datos	Fase de recopilación de información relevante para el proyecto.	5 días
1.1. Técnicas de recolección de datos	Realización de entrevistas con el personal clave de la Unidad de TI para obtener detalles y requisitos.	2 días
1.2. Análisis de las entrevistas	Análisis y procesamiento de los datos recopilados en las entrevistas.	3 días
2. Determinación de aplicaciones contenerizadas	Identificación de las aplicaciones que serán desplegadas en contenedores dentro del clúster de Kubernetes.	5 días
3. Recursos disponibles	Evaluación de los recursos disponibles de los servidores.	3 días
4. Análisis del sistema operativo	Selección y análisis del sistema operativo más adecuado para el entorno de Kubernetes.	3 días

5. Diseño de la Arquitectura Kubernetes	Fase de diseño de la arquitectura de Kubernetes para el entorno de pruebas.	10 días
5.1. Topología del clúster	Definición de la topología del clúster, incluyendo la cantidad y distribución de nodos maestros y trabajadores.	9 días
5.2. Definición de recursos de los nodos Master y Worker	Determinación de los recursos (CPU, memoria, almacenamiento) asignados a los nodos maestros y trabajadores.	1 día

Durante esta fase de planificación, se recopiló información crucial, se identificaron los recursos necesarios y se establecieron los cimientos para la implementación. De acuerdo al cronograma definido anteriormente se realizaron las siguientes actividades:

2.3.1. Recolección de Datos.

Para comprender el contexto específico de la Unidad de TI de la PUCE-Ibarra, se emplearon técnicas cualitativas de recolección de datos. Estas técnicas permitieron obtener información pertinente sobre las necesidades, desafíos y recursos disponibles en la Unidad respecto a la gestión de contenedores. La principal técnica utilizada fueron las entrevistas semiestructuradas, las cuales se realizaron con miembros clave del personal de TI, incluyendo al encargado de la Unidad, al encargado de redes y al desarrollador responsable de las aplicaciones.

Se realizaron entrevistas individuales con miembros clave de la Unidad de Tecnologías de la Información. Antes de las entrevistas, se preparó una guía de preguntas que aborde los temas esenciales, que se puede ver en el Anexo II, III y IV. Sin embargo, la secuencia y la formulación de las preguntas se adaptan según el flujo de la conversación y las respuestas del entrevistado.

En el desarrollo de esta investigación, se basó en consideraciones éticas y respeto por la privacidad de los participantes. Se utilizaron entrevistas semiestructuradas como una técnica de recolección de datos, y antes de llevar a cabo estas entrevistas, se obtuvo el consentimiento informado de cada uno de los participantes, es decir, fueron debidamente informados sobre los objetivos de la investigación y los procedimientos a seguir. Asimismo, se les aseguró que la información

recopilada sería utilizada únicamente con fines de investigación y que se tomarían todas las medidas necesarias para garantizar la confidencialidad tanto de los involucrados como de los recursos puestos a disposición para el proyecto.

Análisis de las Entrevistas. A partir de las entrevistas realizadas al jefe de la Unidad de TI y al desarrollador de software se pueden identificar varios aspectos relevantes para el análisis:

- Proceso actual de implementación y gestión de aplicaciones: En la entrevista con el jefe de la Unidad de TI, se pudo conocer que actualmente el proceso de implementación y gestión de aplicaciones se realiza de manera manual y poco eficiente. Esto genera desafíos en términos de portabilidad, disponibilidad y fiabilidad de las aplicaciones.

- Familiaridad con contenedores y Kubernetes: Los entrevistados manifestaron tener conocimientos sobre la tecnología de contenedores y Kubernetes, pero reconocieron la necesidad de profundizar en esta área para aprovechar al máximo sus beneficios.

- Beneficios esperados de Kubernetes: Entre los beneficios esperados al utilizar Kubernetes, mencionaron la disponibilidad, la facilidad de implementación y actualización de aplicaciones, así como la optimización de recursos.

- Recursos disponibles para implementar un entorno de pruebas: El encargado de la Unidad de TI indicó que cuentan con recursos de hardware y software limitados para este proyecto, pero suficientes para la implementación de un entorno de pruebas con Kubernetes.

- Aplicaciones a probar en Kubernetes: El encargado en desarrollo de software mencionó que les gustaría probar aplicaciones web contenerizadas con Docker, realizadas internamente. Posteriormente se definió la aplicación.

- Interacción con la infraestructura Kubernetes: Los entrevistados expresaron su interés en interactuar con la infraestructura Kubernetes de manera sencilla y amigable, mediante una interfaz gráfica o en su defecto una herramienta de línea de comandos.

- Requisito del sistema operativo: Se identificó la necesidad de utilizar un sistema operativo basado en Unix, que tenga compatibilidad para trabajar con Docker y Kubernetes. La

elección del sistema operativo puede variar según las necesidades específicas del proyecto y la familiaridad del equipo con una distribución en particular.

En general, el análisis de las entrevistas resalta la necesidad y el interés de la Unidad de Tecnologías de la Información en adoptar soluciones como Kubernetes para mejorar la gestión de aplicaciones, así como la disposición a explorar y aprender sobre estas nuevas tecnologías.

2.3.2. Determinación de Aplicaciones Contenerizadas.

Una parte fundamental del proyecto fue la selección de las aplicaciones que se utilizarían para probar el entorno de Kubernetes. Dado que la Unidad de Tecnologías de la Información ya ha adoptado la tecnología de contenedores para pocas aplicaciones, se decidió aprovechar una de las aplicaciones que ya se encontraban contenerizadas y en uso.

En conjunto con el desarrollador de software, se estableció la aplicación realizada internamente, que se utilizaron para probar el clúster de Kubernetes, considerando los aspectos de contenerización con Docker y relevancia de la aplicación.

La aplicación ha sido seleccionada para ser probadas en el entorno de pruebas de Kubernetes debido a varias razones importantes:

- Utilización de contenedores Docker: Las aplicaciones ya se encuentran contenerizadas utilizando Docker, lo que simplifica su migración al entorno de Kubernetes y la integración en el clúster es más sencilla y eficiente.

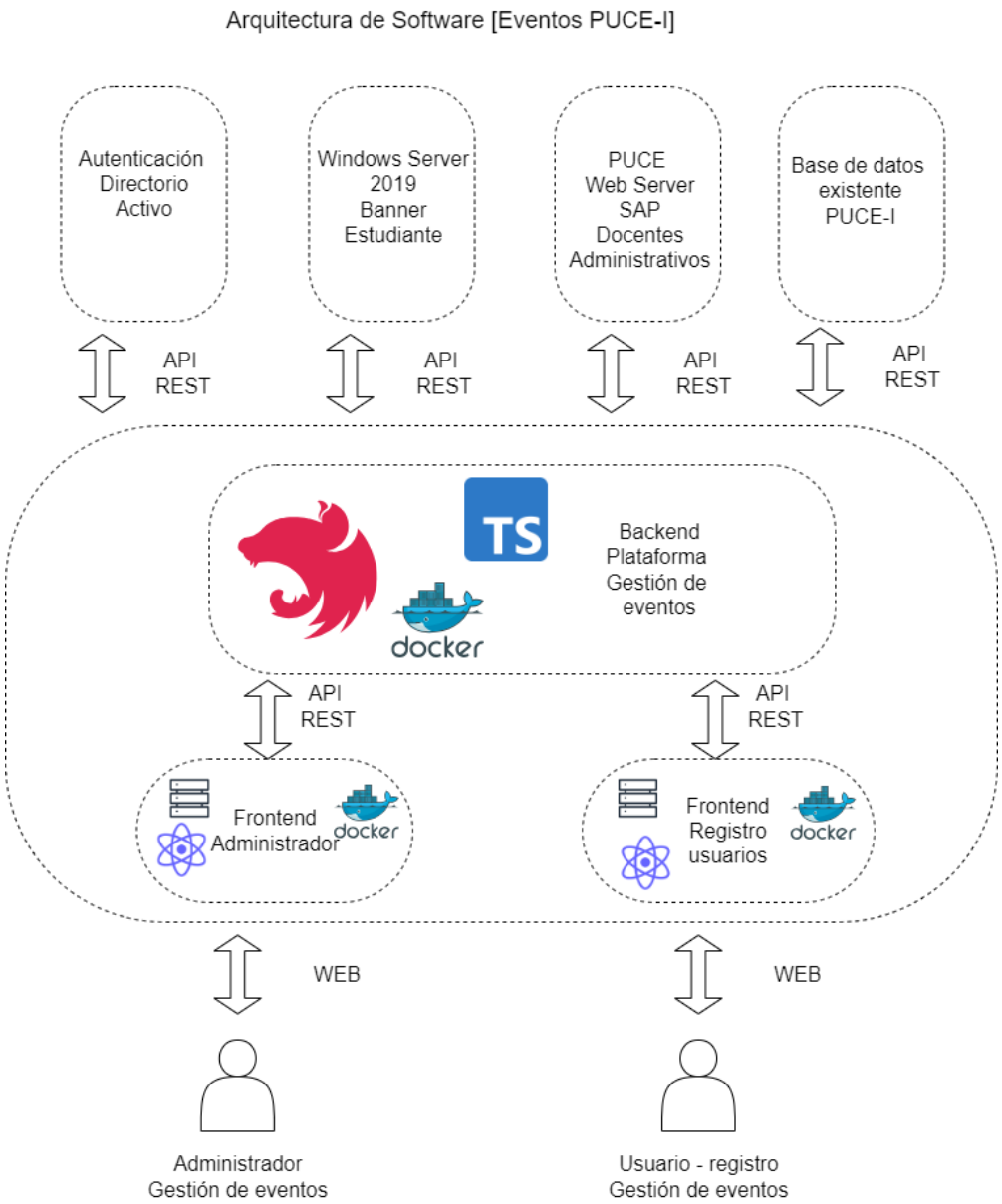
- Importancia de las aplicaciones: Ambas aplicaciones son útiles y necesarias, sin embargo, no son aplicaciones críticas. Además, es importante mencionar que no todas las aplicaciones de la Unidad de TI de la PUCE-I desarrolladas se encuentran con tecnologías de contenedores por lo que se puso a disposición el software mencionado en esta sección.

A continuación, se detallan la aplicación definida para realizar las pruebas en la infraestructura de Kubernetes:

Aplicación de Gestión de Eventos. Esta aplicación web está diseñada para facilitar la gestión y organización de eventos institucionales. En la Figura 5 se observa la arquitectura de software de la aplicación, en la que igualmente se diferencian los componentes que la forman.

Figura 5

Arquitectura de Software [Eventos PUCE-I]



Nota. Arquitectura de software de la aplicación Gestión de Eventos, que muestra los diferentes componentes y servicios involucrados. Elaboración propia

2.3.3. Recursos Disponibles.

Para llevar a cabo de manera exitosa el trabajo propuesto en este documento, fue fundamental contar con los recursos adecuados tanto en hardware como en software. En este apartado, se detallan los recursos disponibles en la Unidad de TI de la PUCE-Ibarra para este proyecto.

Es importante destacar que, si bien los recursos son limitados, se consideró que eran suficientes para la implementación de un entorno de pruebas a pequeña escala y para optimizar el uso de este recurso y poder alojar el entorno de pruebas de Kubernetes, se aprovechó la tecnología de virtualización.

En cuanto a los recursos de hardware disponibles, se contó con un servidor físico base de gama media. En la Tabla 3 se pueden ver las características:

Tabla 3
Características del Servidor Base

Características	Servidor base
Marca	Hewlett Packard
Modelo	ProLiant BL460c
Generación	Gen8
CPU	8 CPUs x Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
RAM	32 GB
Almacenamiento	600 GB
Hipervisor	Esxi Vmware

Nota. Elaboración propia

2.3.4. Análisis de los sistemas operativos compatibles con Docker y Kubernetes.

Si bien estas tecnologías son multiplataformas, lo que significa que pueden ejecutarse en diferentes sistemas operativos. Existen algunos sistemas operativos cuentan con mejor soporte y

compatibilidad que otros. Por lo tanto, es importante analizar las opciones disponibles como se muestra a continuación en la Tabla 4.

Tabla 4

Comparación de Sistemas Operativos Compatibles con Docker y Kubernetes

Sistema Operativo	Compatibilidad con Docker	Compatibilidad con Kubernetes	Justificación
Ubuntu	Excelente	Excelente	Ubuntu es una distribución de Linux muy popular y ampliamente utilizada en entornos de desarrollo y producción. Ofrece excelente compatibilidad con Docker y Kubernetes, con una gran comunidad de usuarios y abundante documentación.
RockyLinux	Excelente	Excelente	Conocida por su estabilidad y como reemplazo de CentOS. RockyLinux ofrece excelente compatibilidad con Docker y Kubernetes. Sin embargo, durante las pruebas iniciales, se encontraron algunos desafíos en la configuración que consumieron tiempo adicional, lo que llevó a explorar otras opciones.
CentOS	Excelente	Excelente	CentOS, anteriormente una de las distribuciones de Linux empresariales más populares, era ampliamente utilizada en entornos de producción debido a su estabilidad y soporte a largo plazo. Sin embargo, CentOS ha sido descontinuado y reemplazado.
AlmaLinux	Excelente	Excelente	AlmaLinux es una distribución de Linux empresarial, derivada de Red Hat Enterprise Linux (RHEL), que surgió como reemplazo directo de CentOS. Ofrece una excelente compatibilidad con Docker y Kubernetes, así como un rendimiento óptimo. Sin embargo, al ser relativamente nueva, se encontró que algunos paquetes y recursos específicos para la integración de este proyecto estaban más maduros y mejor documentados en Ubuntu.

Windows Server	Buena	Buena	Docker y Kubernetes son compatibles con Windows Server, pero su implementación puede ser más compleja y requerir configuraciones adicionales. Además, algunas características avanzadas de Docker y Kubernetes pueden tener un soporte limitado en Windows Server.

Después de analizar la compatibilidad de los sistemas operativos, realizar pruebas iniciales para cada opción, se determinó que Ubuntu 20.04 LTS Server era la opción más adecuada para implementar el entorno de pruebas. Esta elección se basó en la excelente compatibilidad de Ubuntu con Docker y Kubernetes, la estabilidad y soporte a largo plazo ofrecidos por la versión LTS, además de la documentación específica para la integración requerida, la eficiencia en la configuración y resolución de problemas observada durante las fases iniciales del proyecto.

2.3.5. Diseño de la Arquitectura Kubernetes.

El presente trabajo propone el uso de la plataforma de código abierto Kubernetes como una solución para brindar portabilidad, disponibilidad y fiabilidad a las aplicaciones. Considerando todo esto, se diseñó una arquitectura de clúster, el monitoreo gráfico, el monitoreo por consola y las redes.

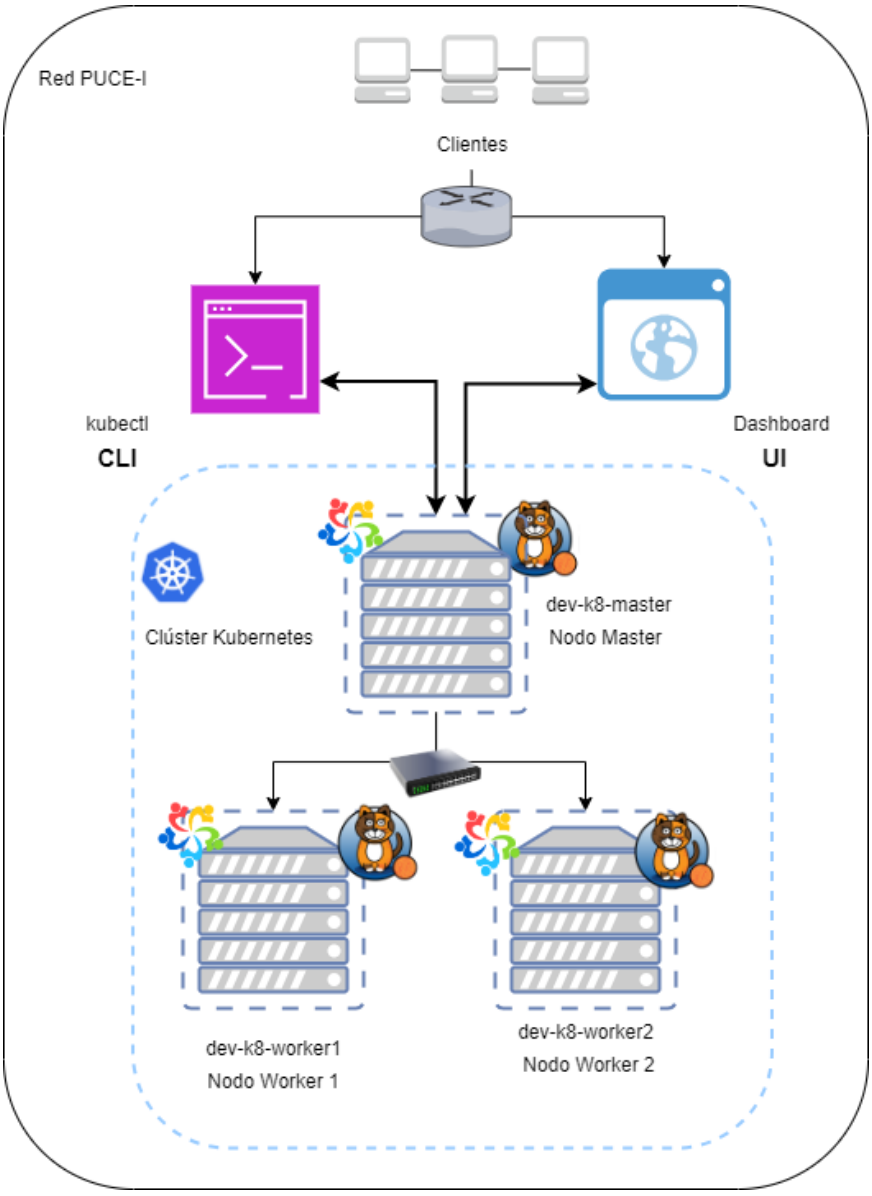
Topología del Clúster. En la Figura 7 se puede apreciar el diagrama detallado del clúster que se implementó, consta de un nodo maestro (dev-k8-master) y dos nodos trabajadores (dev-k8-worker1 y dev-k8-worker2). El nodo maestro es responsable de gestionar y coordinar las operaciones del clúster, mientras que los nodos trabajadores son los encargados de ejecutar las aplicaciones en contenedores (pods).

Los usuarios finales (Clientes) pueden gestionar las aplicaciones desplegadas en el clúster a través de la interfaz de usuario (Dashboard UI) o mediante la interfaz de línea de comandos (CLI) conocida como kubectl.

La comunicación entre los diferentes componentes del clúster se realiza a través de la red de la PUCE-I que interconecta todos los nodos. Por otra parte, para la comunicación entre los pods, que tienen los contenedores, se comunican por la red de pods definida en la implementación y

gestionada por Calico, un CNI (*Container Network Interface*), que en la Figura 6 se muestra con el símbolo de un gato.

Figura 6
Arquitectura del Clúster Kubernetes



Nota. Elaboración propia.

Nodo Maestro. El nodo maestro (dev-k8-master) es el componente principal que gestiona y coordina todas las operaciones del clúster de Kubernetes. En este proyecto, se destinaron los siguientes recursos (ver Tabla 5) para el nodo maestro:

Tabla 5
Recursos Asignados al Nodo Maestro

Recurso	Asignación
CPU Virtual	4
RAM	7 GB
Almacenamiento	70 GB
Sistema operativo	Ubuntu
Versión del sistema operativo	20.04.6 LTS
Arquitectura del hardware	x86_64

Nodos Trabajadores. Los nodos trabajadores (dev-k8-worker1 y dev-k8-worker2) son los encargados de ejecutar las cargas de trabajo (pods) de las aplicaciones en contenedores. En este proyecto, se asignaron los siguientes recursos (ver Tabla 6) para cada nodo trabajador:

Tabla 6
Recursos Asignados a los Nodos Trabajadores

Recurso	dev-k8-worker1	dev-k8-worker2
CPU Virtual	4	4
RAM	7 GB	7 GB
Almacenamiento	70 GB	70 GB
Sistema operativo	Ubuntu	Ubuntu
Versión del sistema operativo	20.04.6 LTS	20.04.6 LTS
Arquitectura del hardware	x86_64	x86_64

2.4. Fase de Desarrollo:

Durante esta fase, se procedió a la instalación del entorno de pruebas utilizando las tecnologías de Docker y Kubernetes. Se instalaron estas tecnologías en los servidores seleccionados

Se configuró el entorno de pruebas, incluyendo la creación del clúster de Kubernetes, definición de recursos (nodos, pods, servicios, almacenamiento), configuración de redes virtuales e integración de herramientas de monitoreo.

Durante todo el proceso de implementación, se documentó cada paso, incluyendo el despliegue del Dashboard y aplicaciones.

2.5. Fase de Entrega

Abarca la puesta en marcha de las aplicaciones objetivo, asegurando su correcto funcionamiento dentro del clúster de Kubernetes. Se detalla la matriz de pruebas realizada, donde se evaluaron diversos escenarios y se verificó el cumplimiento de los requisitos establecidos. Además, se incluyen capturas de pantalla que muestran el producto final listo para su uso, ejemplificando la interfaz de usuario y las funcionalidades implementadas.

2.6. Fase de Uso

En esta fase, se llevó a cabo la entrega oficial la solución al Departamento de Tecnologías de la Información de la PUCE-Ibarra. Esta fase implica la transferencia formal de la infraestructura implementada. Como constancia de esta entrega, se adjunta el acta de recepción firmada por los representantes del departamento.

CAPÍTULO III. RESULTADOS

En este capítulo, se presentan los resultados obtenidos durante las fases finales del proyecto, lo cual comprende el detalle de la fase de Desarrollo, fase de Entrega y la fase de Uso de la metodología escogida. Estas etapas cruciales marcan la culminación del proceso de implementación de la infraestructura de Kubernetes para la ejecución de aplicaciones en contenedores en un entorno de pruebas.

3.1. Desarrollo del Entorno de Pruebas:

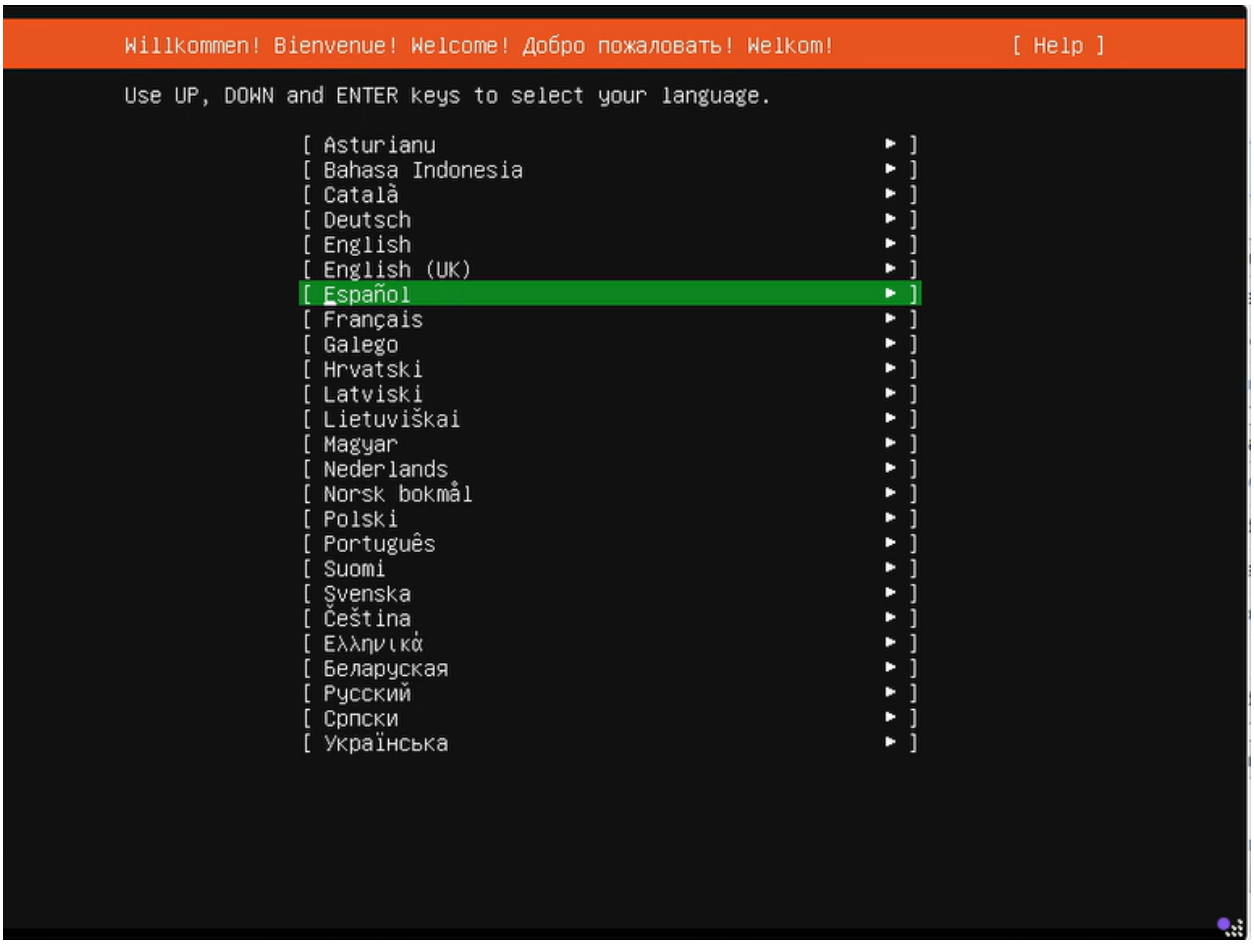
3.1.1. Instalación del Sistema Operativo en los Servidores

Se descargó el software libre Ubuntu 20.04.6 LTS en su imagen mínima y se instaló en los servidores seleccionados. Esta distribución de Linux fue elegida por su compatibilidad nativa con Docker y Kubernetes, así como por su estabilidad y soporte a largo plazo.

Es importante destacar que se utilizó el mismo sistema operativo en los tres servidores, por lo que la instalación es la misma. A continuación, se muestra el proceso de instalación del sistema operativo.

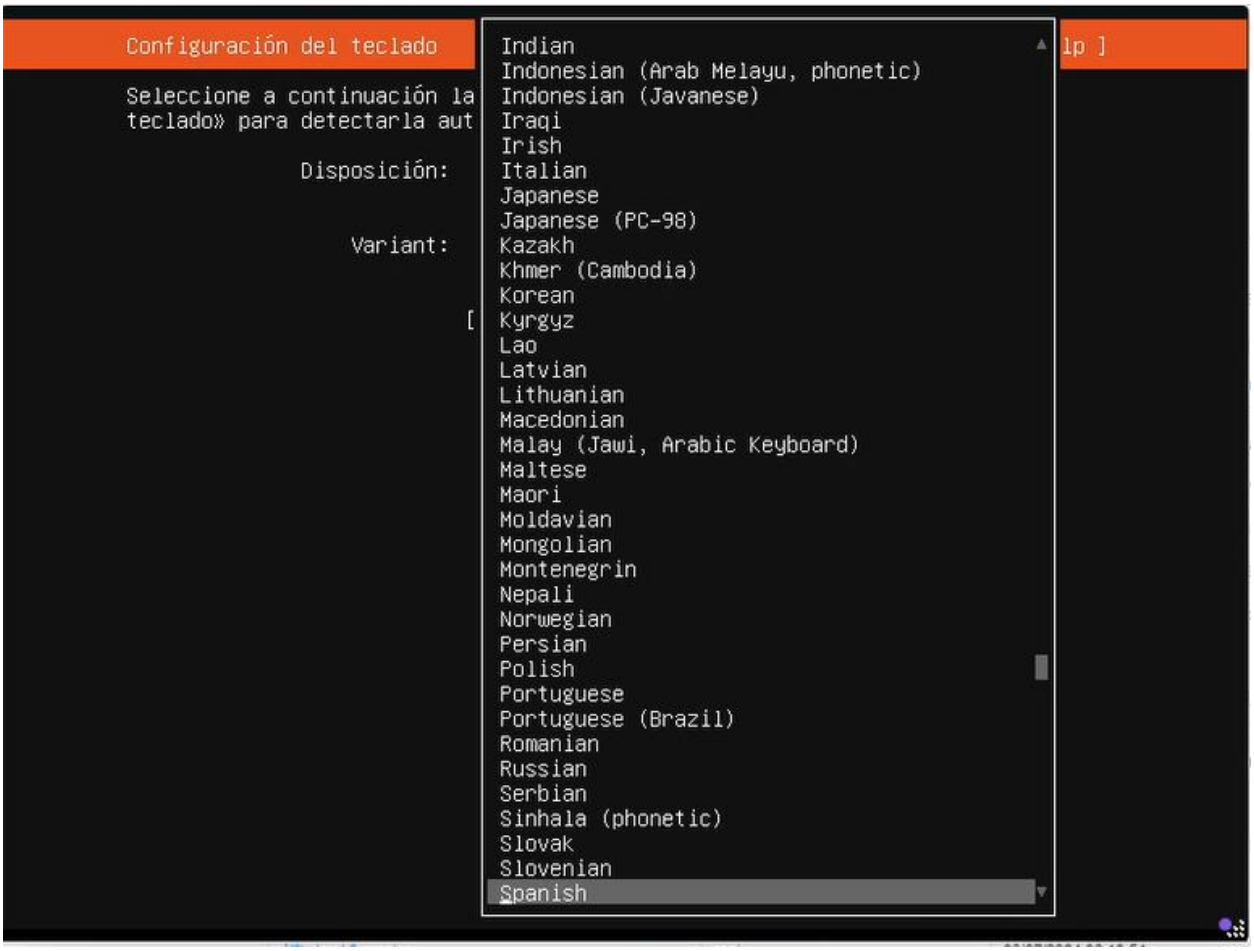
1. Una vez que el servidor inició desde el medio de instalación, se mostró una interfaz en la que se configuró el idioma, como se ve en la Figura 7. Se seleccionó el idioma español.

Figura 7
Elección del Idioma para el Sistema Operativo



2. Al continuar, inmediatamente apareció una pantalla en la que se estableció el idioma del teclado, seleccionando el teclado en español (ver Figura 8).

Figura 8
Configuración del Teclado del Sistema Operativo



3. Se aplicó la configuración y seguidamente se mostró la pantalla de configuración de la red (ver Figura 9). En la que se establecieron las ip, red, dns y gateway de forma manual como se muestra en la Figura 11, en cada uno de los servidores virtuales. Esta información fue definida con el encargado de redes de la universidad.

Figura 9
Pantalla de la Configuración de Red

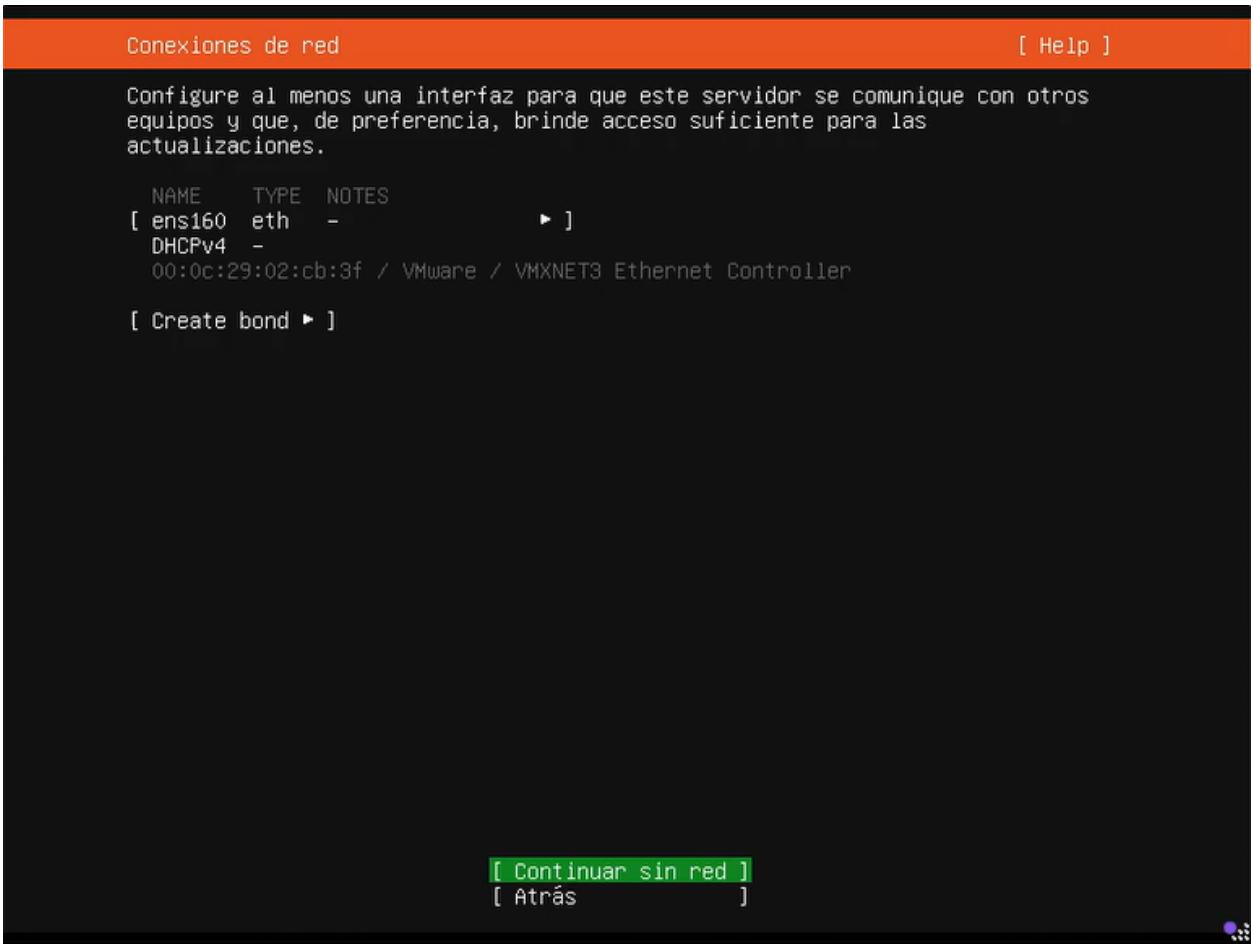
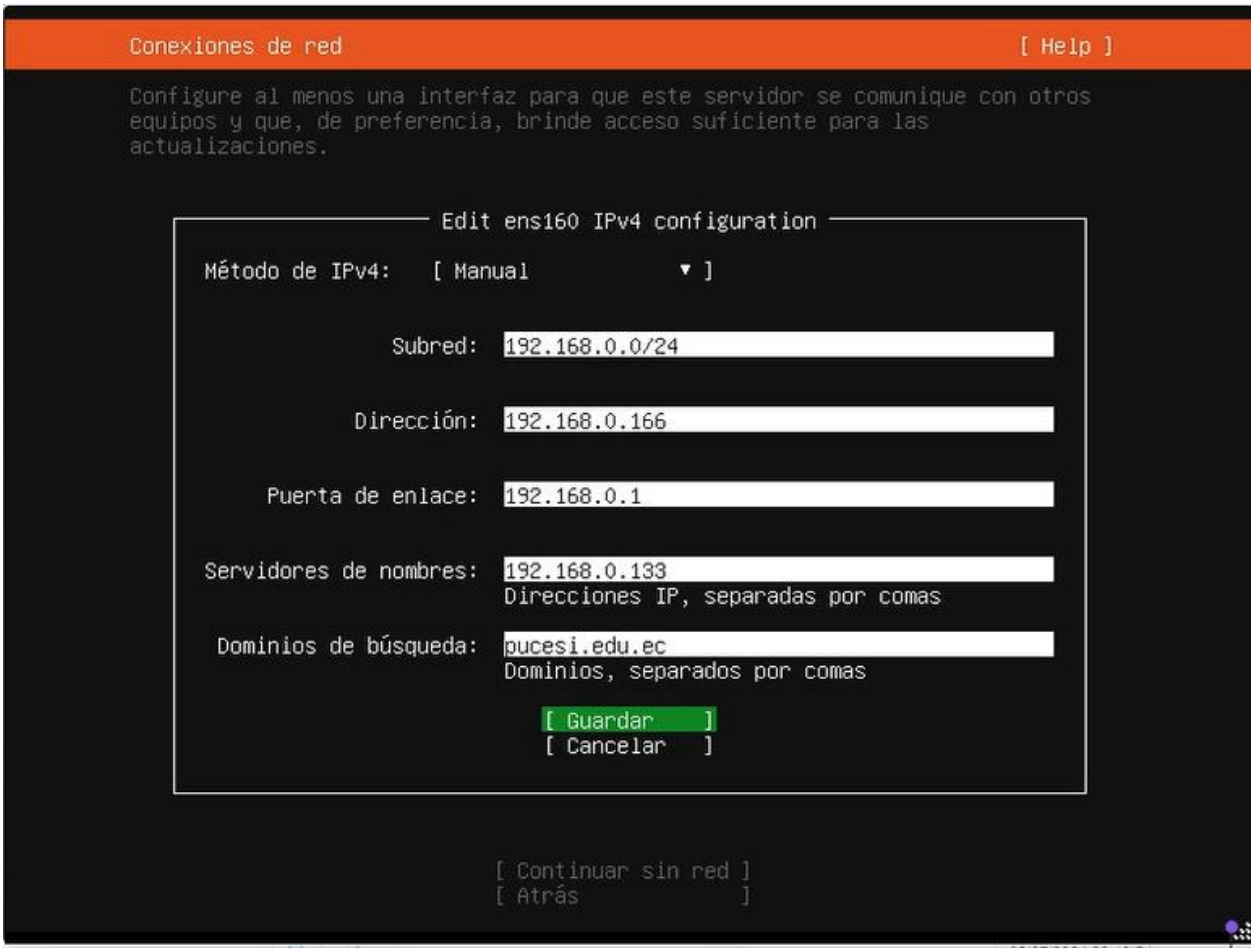


Figura 10
Configuración Manual de la Red



4. Finalizando la configuración de la red, se guarda y se continuó hasta el apartado del almacenamiento, se verificó el disco correcto en el que se va a instalar el sistema operativo, como se observa en la Figura 11.
5. Posteriormente, se muestra el resumen del sistema de archivos del sistema operativo (Ver Figura 12).

Figura 11

Configuración del Almacenamiento para el Sistema Operativo

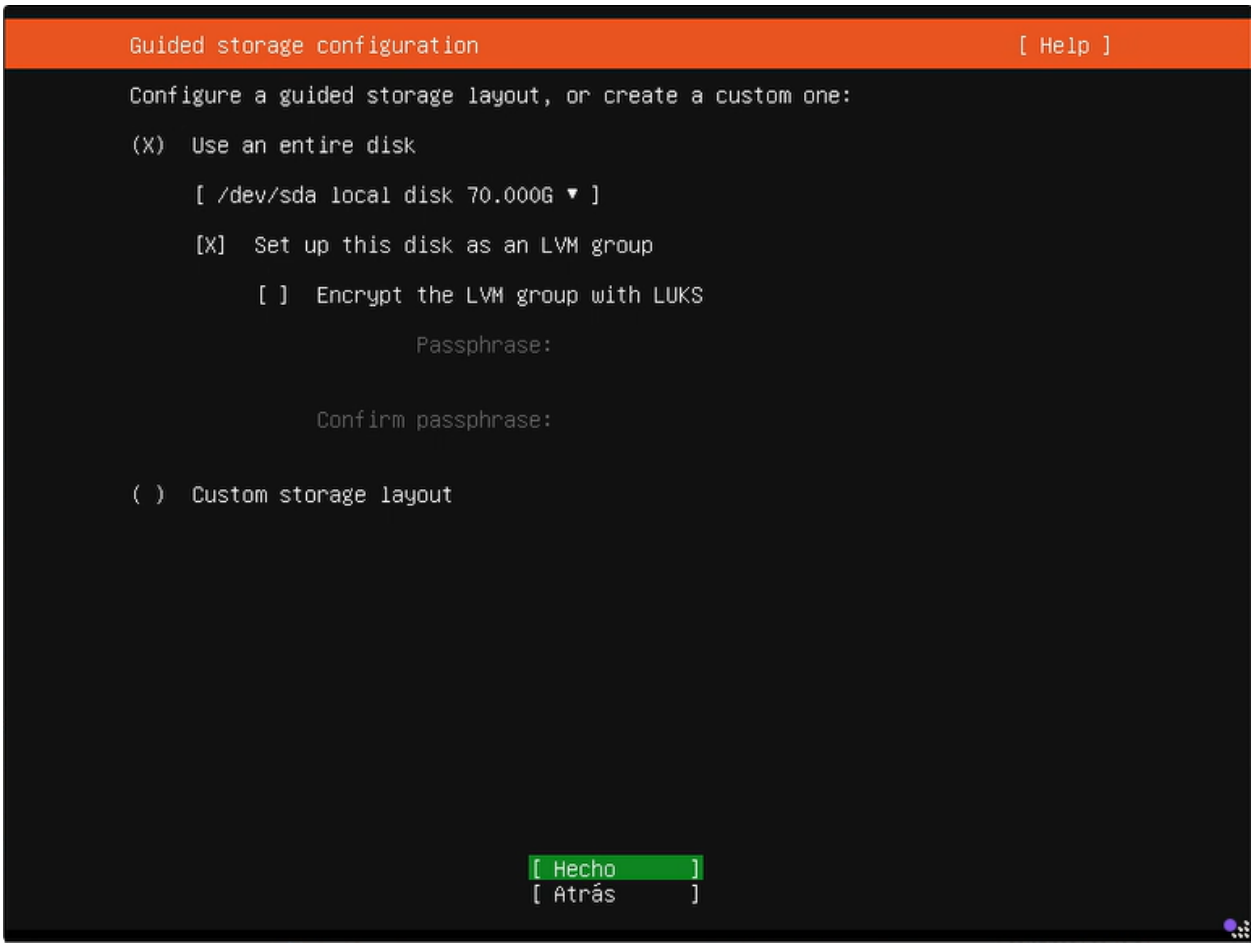
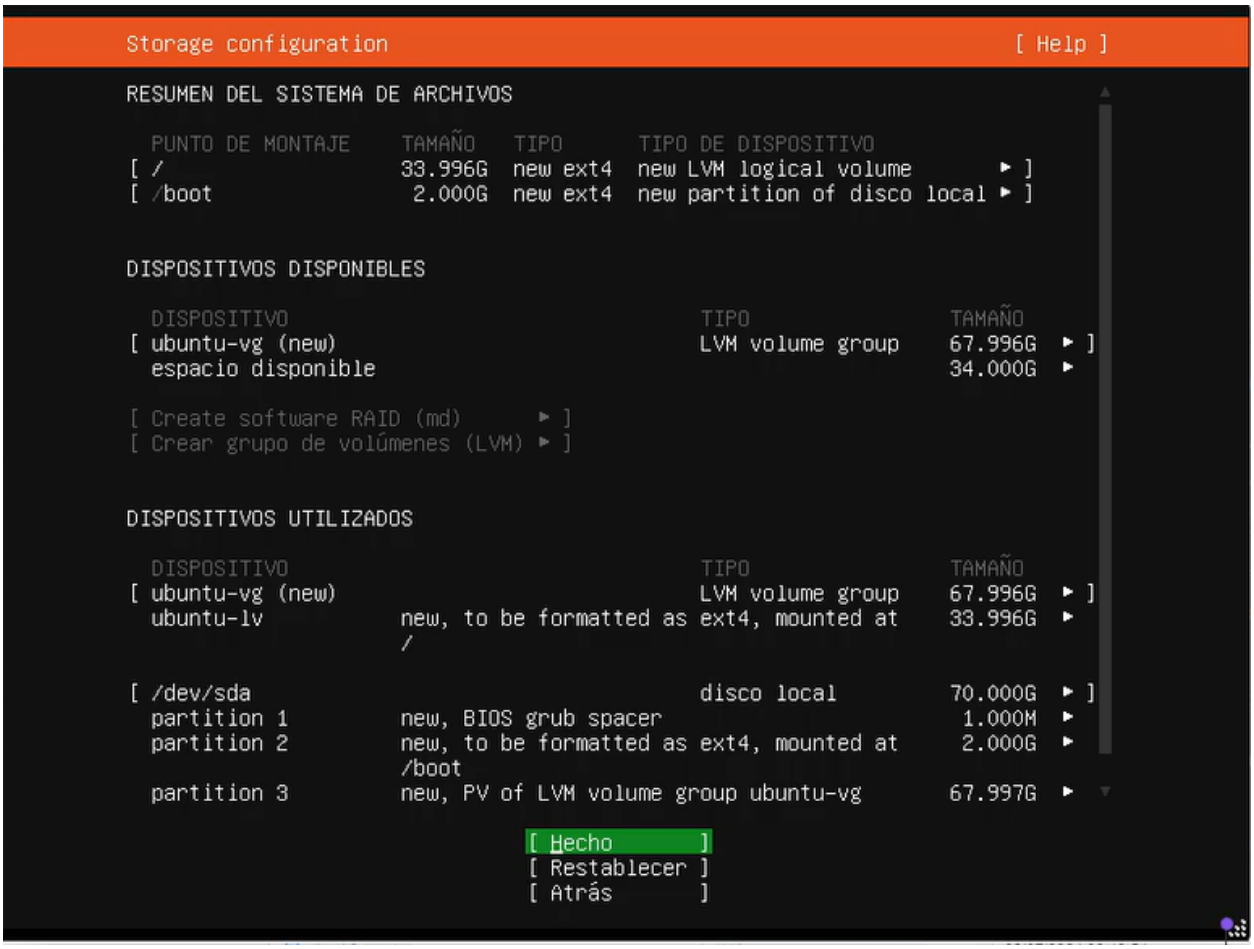


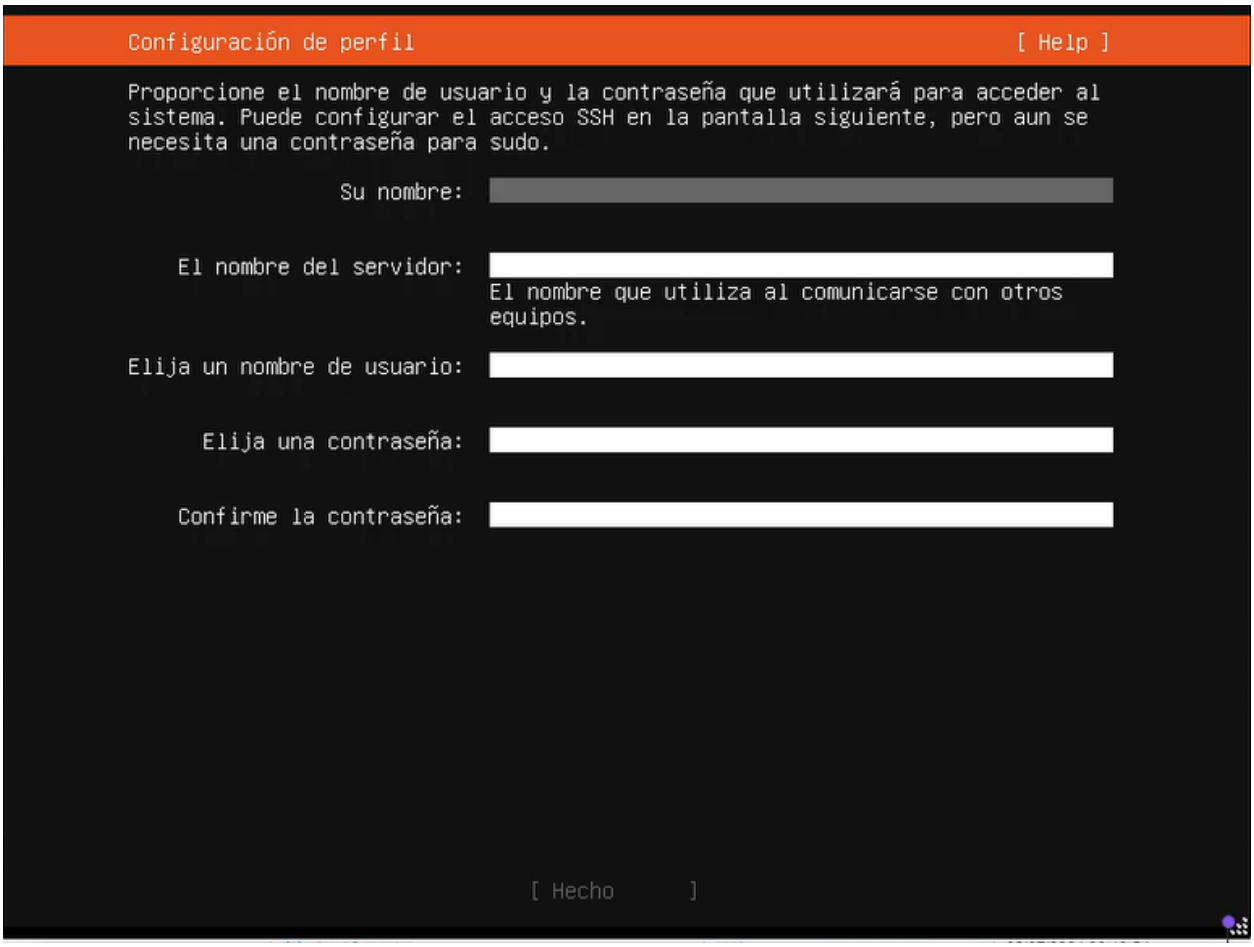
Figura 12
Pantalla de Resumen del Sistema de Archivos



6. Una vez configurado correctamente el sistema de archivos, se definió una cuenta de usuario principal, esta tiene acceso al perfil de super usuario (root) como se visualiza en la Figura 13.

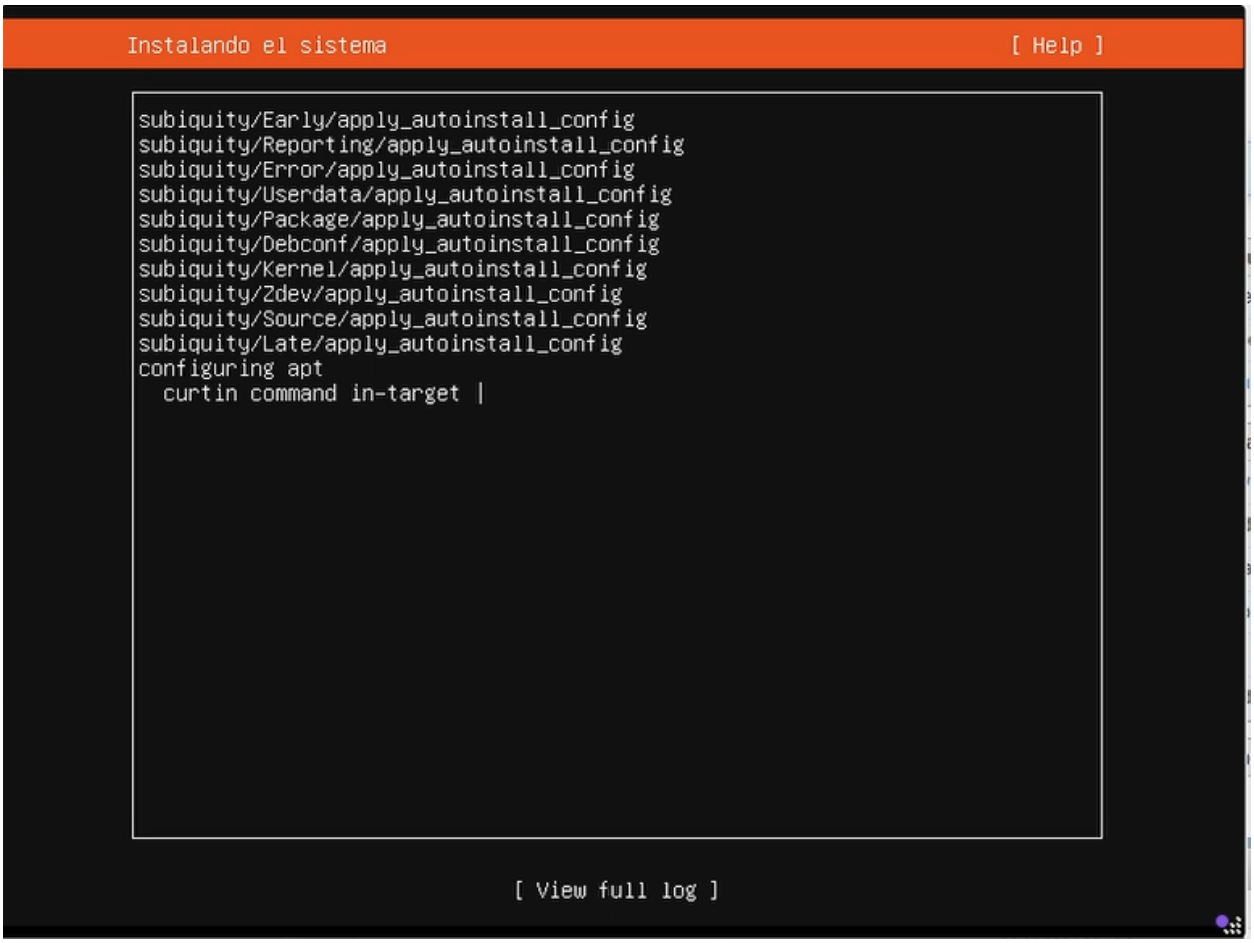
Figura 13

Configuración del Perfil Inicial del Sistema Operativo



7. Una vez completada la información, se continuó hasta que inicie la instalación del sistema operativo (Ver Figura 14). Al terminar el proceso se reinició el sistema y se inicia sesión con el perfil configurado durante la instalación.

Figura 14
Pantalla de Instalación



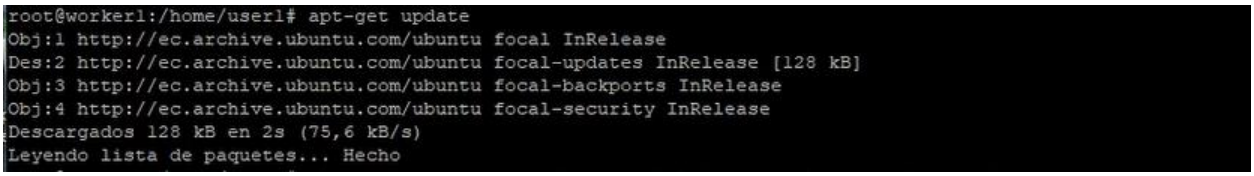
3.1.2. Instalación Inicial en los Servidores del Clúster Kubernetes Maestro - Workers

Después de tener el sistema operativo base instalado, se procedió a la instalación inicial en todos los servidores de la arquitectura. La instalación de estas herramientas es idéntica en todos los servidores. A continuación, se detallan los pasos:

1. Antes de proceder con la instalación, se ingresó al servidor como usuario root y actualizó el sistema operativo (Ver Figura 15) mediante el siguiente comando:

```
# apt-get update
```

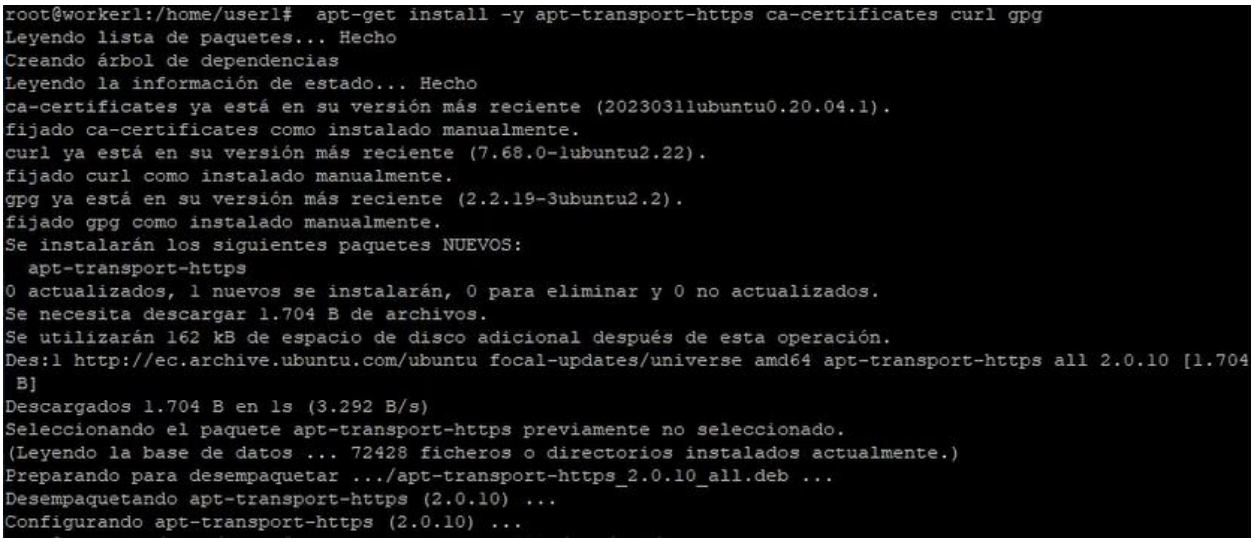
Figura 15
Actualización del Sistema Operativo



2. Se instalaron los paquetes necesarios para manejar conexiones HTTPS, certificados, y otras herramientas esenciales. Esto se logró ejecutando el comando mostrado en la Figura 16, que instaló el siguiente comando:

```
#  apt-get install -y apt-transport-https ca-certificates curl gpg
```

Figura 16
Instalación de los Paquetes Https y Certificados



3. Para preparar el sistema para la instalación de Kubernetes, se creó un directorio específico para almacenar las claves GPG. Se creó el directorio /etc/apt/keyrings (ver Figura 17) con los permisos adecuados con el comando:

```
# sudo mkdir -p -m 755 /etc/apt/keyrings
```

4. Se procedió a descargar e instalar de la clave GPG de Kubernetes. Como se detalla en la Figura 17, se empleó el comando curl para descargar la clave y gpg para procesarla y almacenarla de forma segura en el directorio recién creado.

```
# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

Figura 17

Creación de la Carpeta que Almacena Claves GPG e Instalación de la Clave GPG de Kubernetes



5. Una vez descargada la clave, se añadió el repositorio de Kubernetes al sistema. La Figura 18 muestra el comando que agregó el repositorio, asegurando que el sistema pudiera acceder a los paquetes oficiales de Kubernetes.:

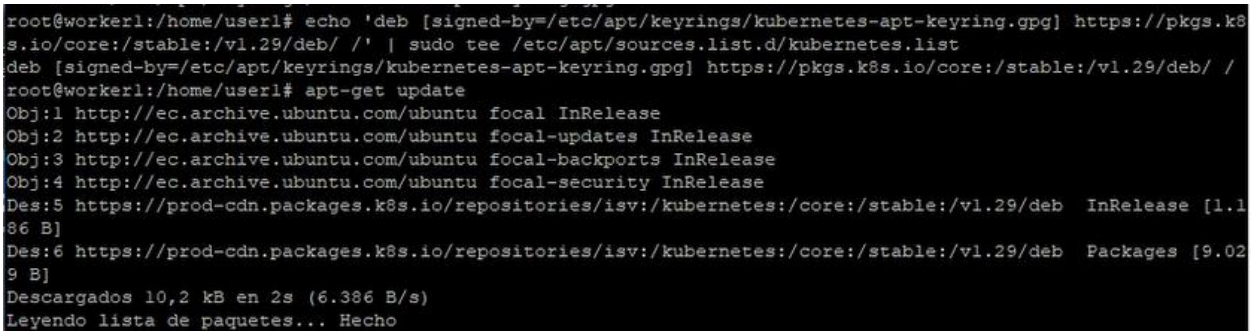
```
# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

6. Fue importante actualizar nuevamente el sistema para reconocer los nuevos paquetes disponibles. Esto se realizó con el comando mostrado en la Figura 18.

```
# apt-get update
```

Figura 18

Instrucción para Añadir un Repositorio de Paquetes de Kubernetes y Actualización del Servidor

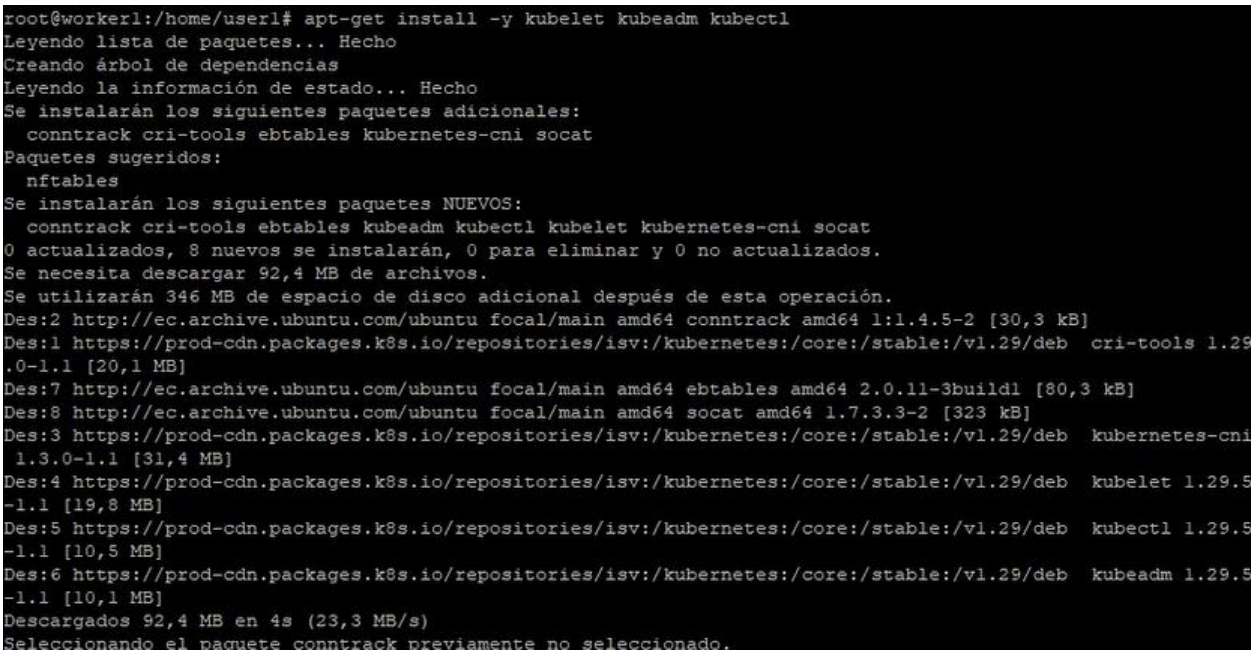


7. Se instalaron los componentes esenciales de Kubernetes: kubelet, kubeadm, y kubectl. Este paso se ejecutó con el comando ilustrado en la Figura 19.

```
# apt-get install -y kubelet kubeadm kubectl
```

Figura 19

Instalación de los Componentes Esenciales para Trabajar con Kubernetes

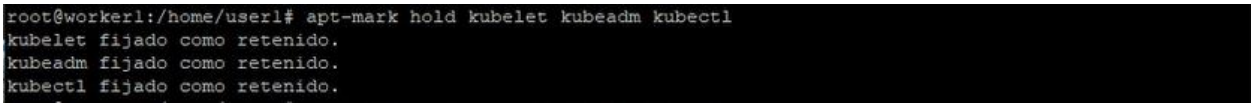


8. Para evitar actualizaciones no deseadas que podrían afectar la estabilidad del clúster, se marcaron estos paquetes como retenidos, utilizando el comando (observar Figura 20):

```
# apt-mark hold kubelet kubeadm kubectl
```

Figura 20

Comando para Evitar las Actualizaciones No Deseadas

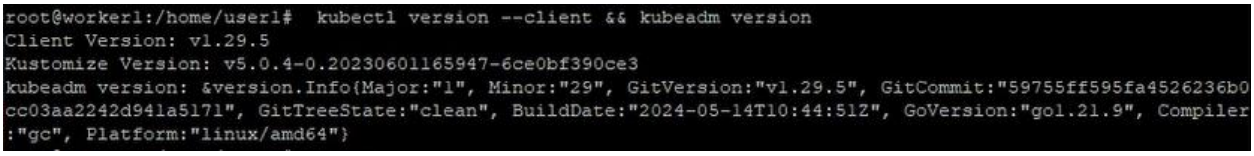


9. Para verificar que la instalación se había realizado correctamente, se consultaron las versiones de las herramientas instaladas. Como se ve en la Figura 21, los comandos kubectl version --client y kubeadm version proporcionan esta información.

```
# kubectl version --client && kubeadm version
```

Figura 21

Versiones de las Herramientas de Kubernetes



10. Un requisito importante de Kubernetes es que el espacio de intercambio (swap) esté desactivado. Para lograr esto, primero se desactivó temporalmente con el comando swapoff -a, y luego se editó el archivo /etc/fstab para hacer el cambio permanente. Las Figuras 22 y 23 ilustran este proceso.

```
# swapoff -a
# nano /etc/fstab
```

Figura 22
Desactivar el Espacio de Intercambio Archivos o Dispositivos

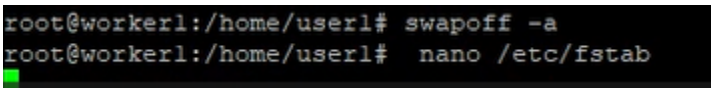
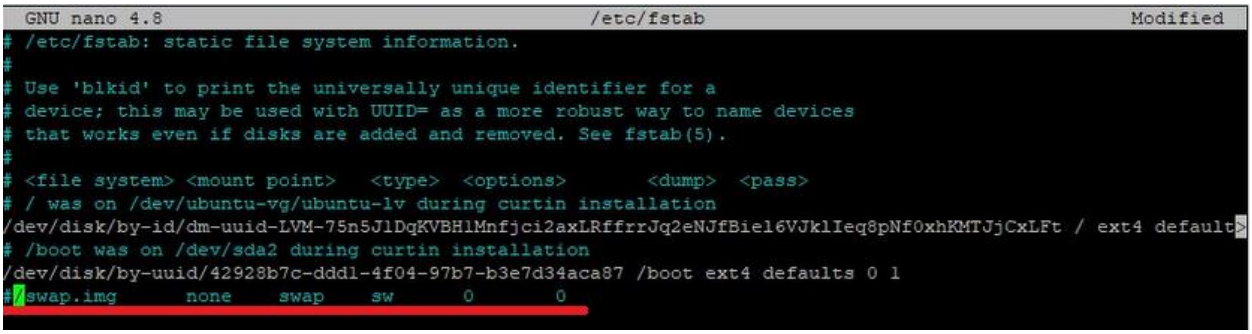


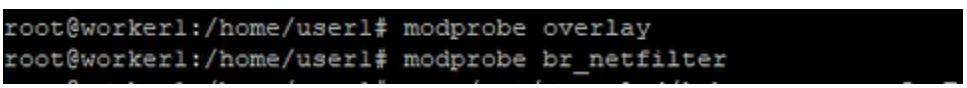
Figura 23
Comentar la Línea de swap.img en el Archivo fstab



11. En el contexto de contenedores, fue importante habilitar ciertos módulos del kernel. El módulo overlay, se cargó para permitir el uso eficiente del almacenamiento en contenedores. De manera similar, el módulo br_netfilter, cargado con los comandos en la Figura 24, fue esencial para el manejo de políticas de red en entornos con redes puente.

```
# modprobe overlay
# modprobe br_netfilter
```

Figura 24
Módulos del Kernel modprobe overlay y modprobe br_netfilter



12. Se modificó el archivo /etc/sysctl.d/kubernetes.conf. Como se ve en la Figura 25, se usó el comando tee para agregar configuraciones que permitieron el reenvío de paquetes y el procesamiento de tráfico a través de bridges.

```
# tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

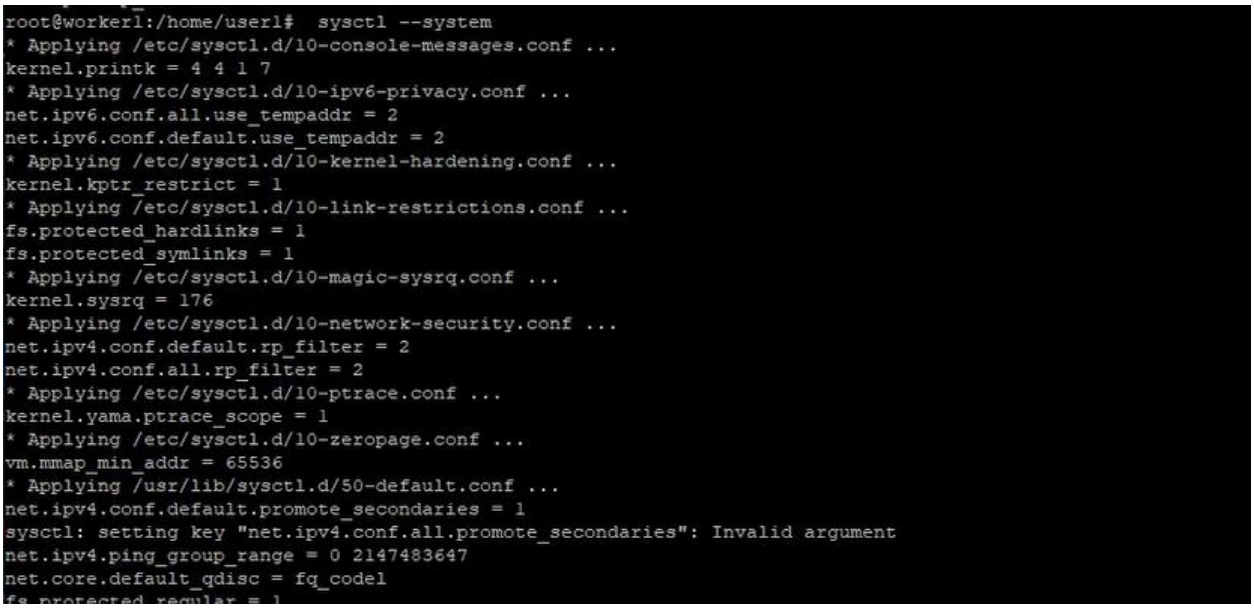
13. Estas configuraciones se aplicaron inmediatamente mediante el comando sysctl --system, como se mostró en la Figura 26.

```
# sysctl --system
```

Figura 25
Modificar el Archivo /etc/sysctl.d/kubernetes.conf



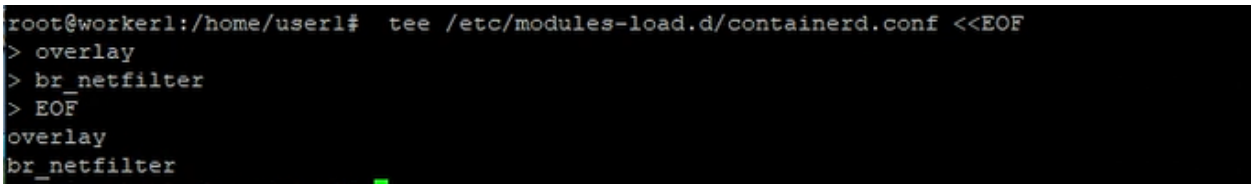
Figura 26
Aplicar y Recargar la Configuración del Kernel



14. Para asegurar que los módulos necesarios se cargaran automáticamente al iniciar el sistema, se creó el archivo /etc/modules-load.d/containerd.conf. La Figura 27 mostró cómo se utilizó tee para agregar los módulos overlay y br_netfilter a este archivo.

```
# tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
```

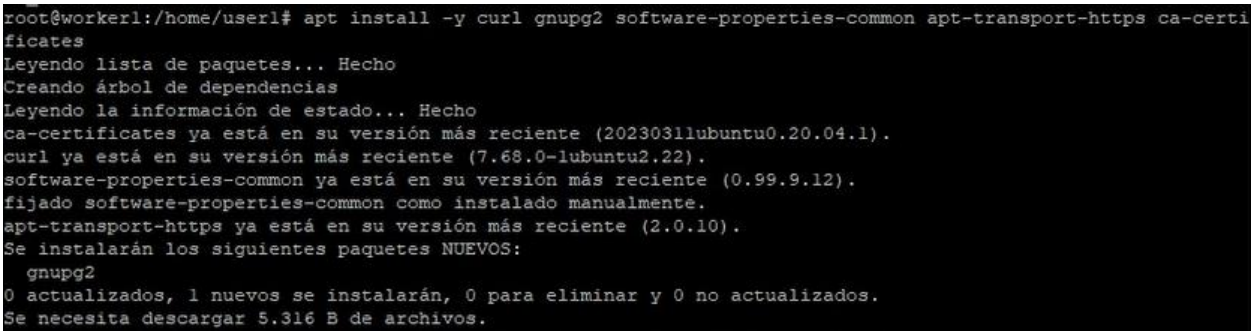
Figura 27
Configuraciones de los Módulos del Kernel Automáticos al Iniciar el Sistema



15. Luego, se procedió a instalar varios paquetes y dependencias esenciales para configurar un sistema para instalar software o agregar repositorios que utilizan conexiones seguras. Esto se hizo con el comando mostrado en la Figura 28.

```
# apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
```

Figura 28
Instalación de Varios Paquetes y Dependencias



16. El siguiente paso fue añadir el repositorio de Docker al sistema. La Figura 29 mostró cómo se descargó, almacenó la clave GPG de Docker y cómo se agregó el repositorio al sistema.

```
# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker-archive-keyring.gpg
```

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

Figura 29

Clave para Verificar la Autenticidad de los Paquetes de Docker y Agregar Repositorio APT al Sistema



17. Siguiendo las mejores prácticas, se actualizó nuevamente el sistema con los comandos mostrados en la Figura 30.

```
# apt update
# apt upgrade -y
```

Figura 30

Instalación de Actualizaciones y Actualización del Sistema



18. Se instaló containerd.io, como se vio en la Figura 31. Este componente fue fundamental para el funcionamiento de los contenedores en Kubernetes.

```
# apt install -y containerd.io
```

Figura 31
Instalación de containerd.io



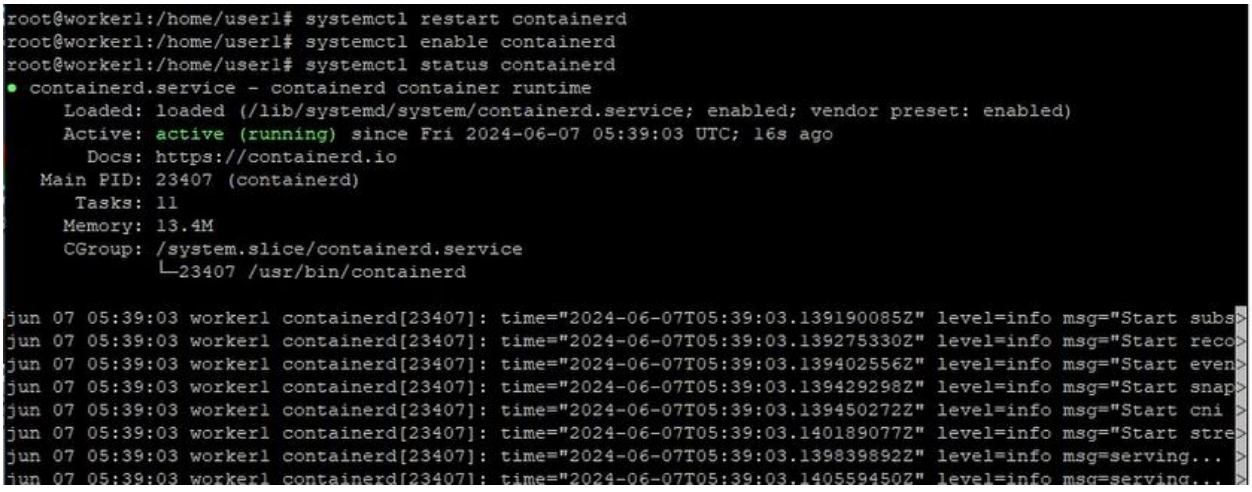
19. La configuración de containerd requirió crear un directorio específico y generar un archivo de configuración predeterminado mediante los siguientes comandos:

```
# mkdir -p /etc/containerd
# containerd config default|sudo tee /etc/containerd/config.toml
```

20. Después de configurar, se reinició el servicio, se habilitó para que iniciara, y se verificó su estado (ver Figura 32).

```
# systemctl restart containerd
# systemctl enable containerd
# systemctl status containerd
```

Figura 32
Restablecer, Iniciar y Verificar el Estado del Servicio containerd



3.1.3. Instalación en el Nodo Maestro.

Después de completar la configuración inicial común a todos los nodos, se procedió con la instalación específica del nodo maestro. A continuación, se detalla el proceso de instalación:

1. El primer paso fue descargar las imágenes necesarias para el clúster de Kubernetes. Utilizando el comando kubeadm config images pull, como se muestra en la Figura 33, se aseguró que todas las imágenes requeridas estuvieran disponibles localmente en el nodo.

```
# kubeadm config images pull
```

Figura 33
Imágenes para el Clúster de Kubernetes en el Nodo Maestro

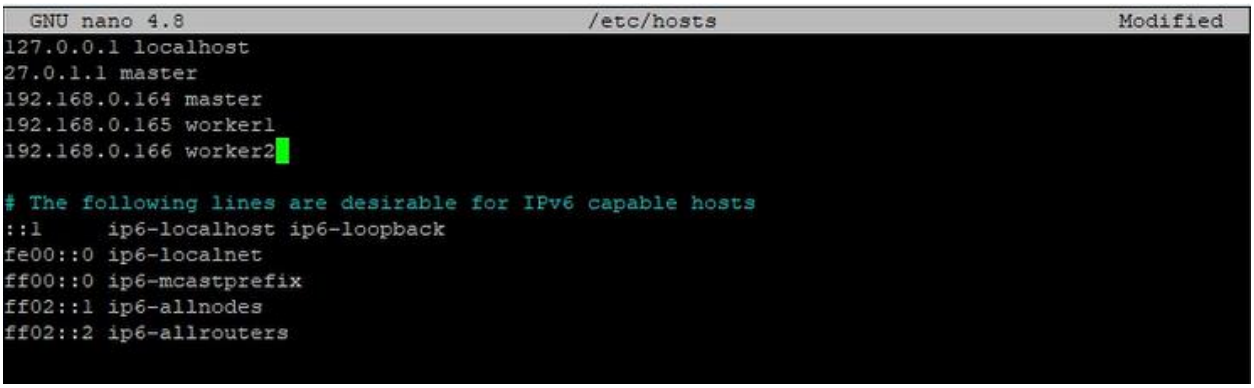


2. A continuación, se agregó la resolución de nombres para el clúster. Utilizando el comando nano /etc/hosts, como se ilustra en la Figura 34, se abrió el archivo de configuración de hosts del sistema. Se ingresaron las direcciones IP de los nodos o servidores del clúster, este paso se realizó en todos los nodos.

```
# nano /etc/hosts
```

Figura 34

Ingresar la IP de los Servidores del Clúster para el Nodo Maestro y Trabajadores



3. El siguiente paso fue la inicialización del clúster de Kubernetes. Esto se logró ejecutando el comando kubeadm init con parámetros específicos, como se ve en la Figura 35. Este comando configuró el pod network CIDR, especificó el socket de containerd, habilitó la carga de certificados, y estableció el endpoint del control plane.

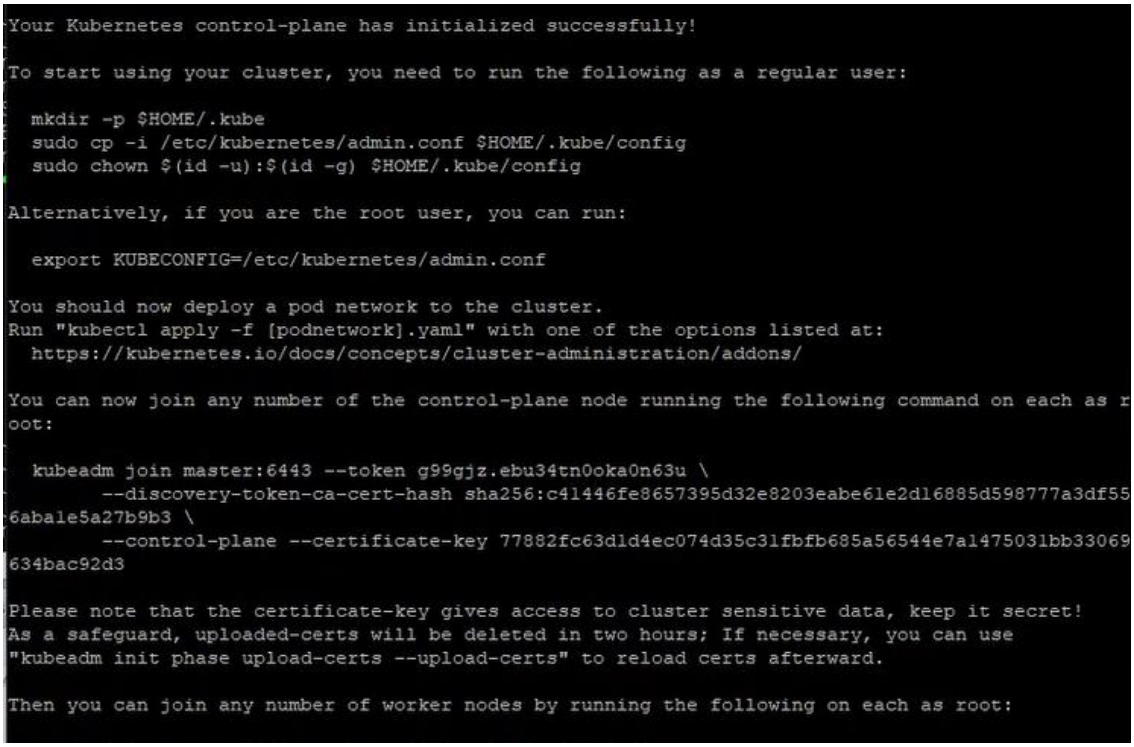
```
# kubeadm init --pod-network-cidr=172.24.0.0/16 --cri
socket=unix:///run/containerd/containerd.sock --upload-certs --control-plane-
endpoint=master
```

En la Figura 36 se observa una salida correcta en la creación del control plane o nodo maestro e información primordial para completar la configuración del clúster.

Figura 35
Inicialización del Clúster de Kubernetes



Figura 36
Salida Exitosa en la Creación del Control Plane



4. Se salió del modo root y se creó el directorio .kube en el directorio home del usuario utilizando el comando mkdir -p $HOME/.kube.

```
$ mkdir -p $HOME/.kube
```

5. Para que el usuario pudiera interactuar con el clúster, se copió el archivo de configuración generado por kubeadm init al directorio .kube. Esto se hizo con el comando sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config , como se ilustra en la Figura 37.

```
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

6. Para asegurar que el usuario tuviera los permisos adecuados, se cambió la propiedad del archivo de configuración utilizando el comando sudo chown $(id -u):$(id -g) $HOME/.kube/config, como se muestra en la Figura 37.

```
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Figura 37
Acceso al Archivo de Configuración del Clúster Kubernetes y Cambio de la Propiedad del Archivo de Configuración para el Usuario Actual

```
master@master:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
master@master:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

7. Para facilitar el uso de kubectl, se estableció la variable de entorno KUBECONFIG con la ruta al archivo de configuración. Esto se hizo con el comando export KUBECONFIG=$HOME/.kube/config.

```
$ export KUBECONFIG=$HOME/.kube/config
```

8. Para verificar que el clúster se había inicializado correctamente, se utilizó el comando kubectl cluster-info, como se muestra en la Figura 38. Este comando proporcionó detalles sobre los servicios del clúster.

```
$ kubectl cluster-info
```

Figura 38
Información Sobre el Clúster Kubernetes

```
master@master:~$ kubectl cluster-info
Kubernetes control plane is running at https://master:6443
CoreDNS is running at https://master:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/pr
oxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
master@master:~$
```

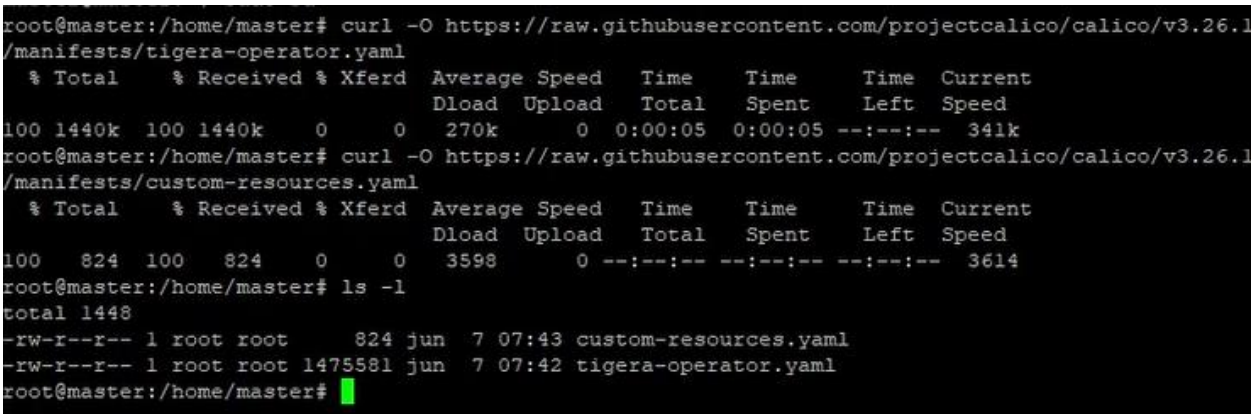
9. La siguiente fase fue la configuración de la red del clúster utilizando Calico. Primero, se descargaron dos archivos esenciales desde el repositorio de GitHub de Calico. Usando los

comandos curl -O, como se ve en las Figura 39, se descargaron tigera-operator.yaml y custom-resources.yaml.

```
# curl -O https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/tigera operator.yaml
# curl -O https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/custom resources.yaml
```

Figura 39

Descarga del Archivo tigera-operator.yaml y Descarga del Archivo custom-resources.yaml

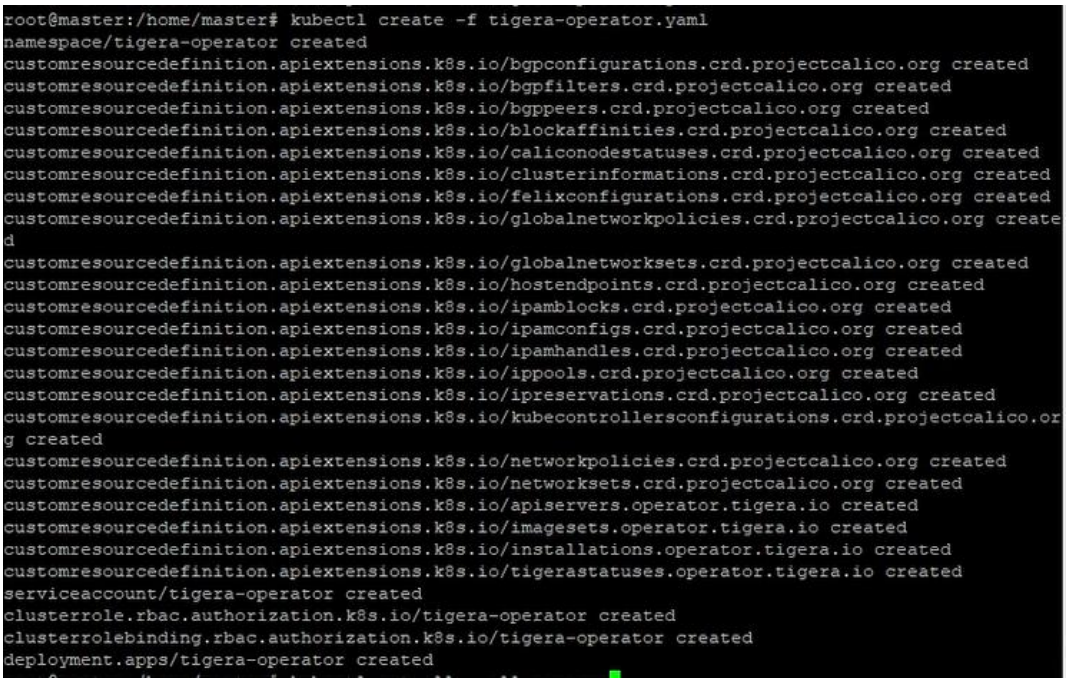


10. Luego, se crearon los recursos de Kubernetes definidos en tigera-operator.yaml usando el comando kubectl create -f tigera-operator.yaml, como se ilustra en la Figura 40.

```
# kubectl create -f tigera-operator.yaml
```

Figura 40

Creación de Recursos del Clúster de Kubernetes a partir del archivo tigera-operator.yaml

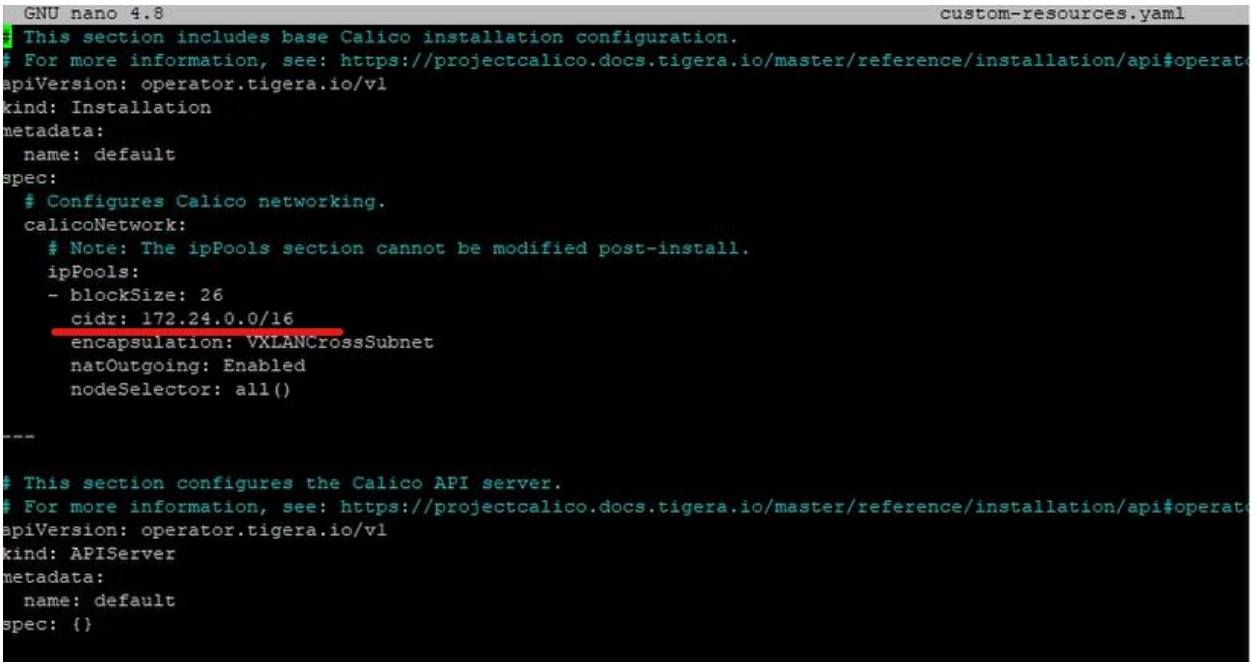


11. Para adaptar la configuración de red a las necesidades del clúster, se modificó el archivo custom-resources.yaml. Utilizando el comando sed, se reemplazó el CIDR predeterminado por el especificado durante la inicialización del clúster y la modificación se verificó usando nano custom-resources.yaml, como se muestra en la Figura 41.

```
# sed -ie 's/192.168.0.0/172.24.0.0/g' custom-resources.yaml
# nano custom-resources.yaml
```

Figura 41

Verificación del Reemplazo en la Instancia cidr

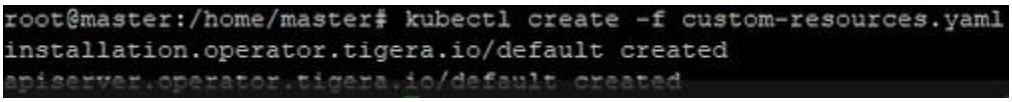


12. Después, se crearon los recursos definidos en custom-resources.yaml con el comando kubectl create -f custom-resources.yaml, como se ilustra en la Figura 42.

```
# kubectl create -f custom-resources.yaml
```

Figura 42

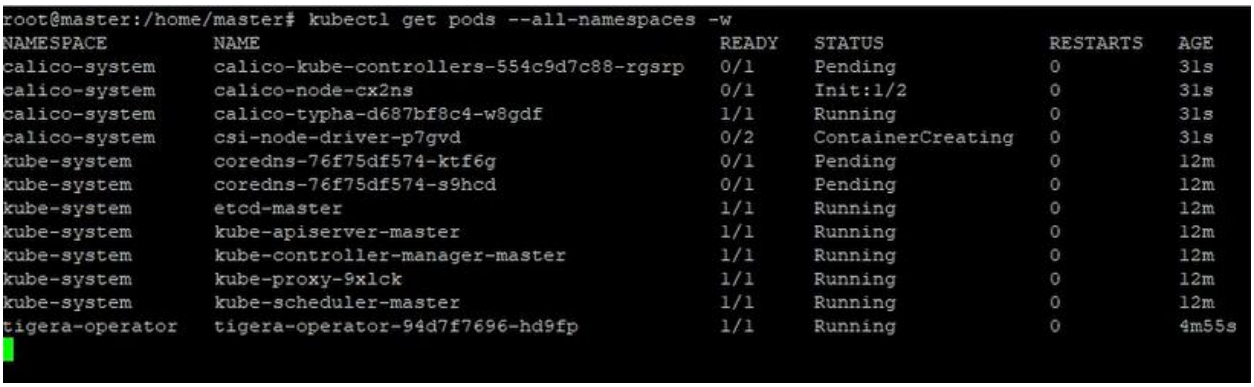
Creación de Recursos del Clúster de Kubernetes a partir del Archivo custom-resources.yaml



13. Para monitorear el despliegue de estos recursos, se utilizó el comando kubectl get pods --all-namespaces -w, como se ve en la Figura 43. Este comando proporcionó una vista en tiempo real del estado de los pods.

```
# kubectl get pods --all-namespaces -w
```

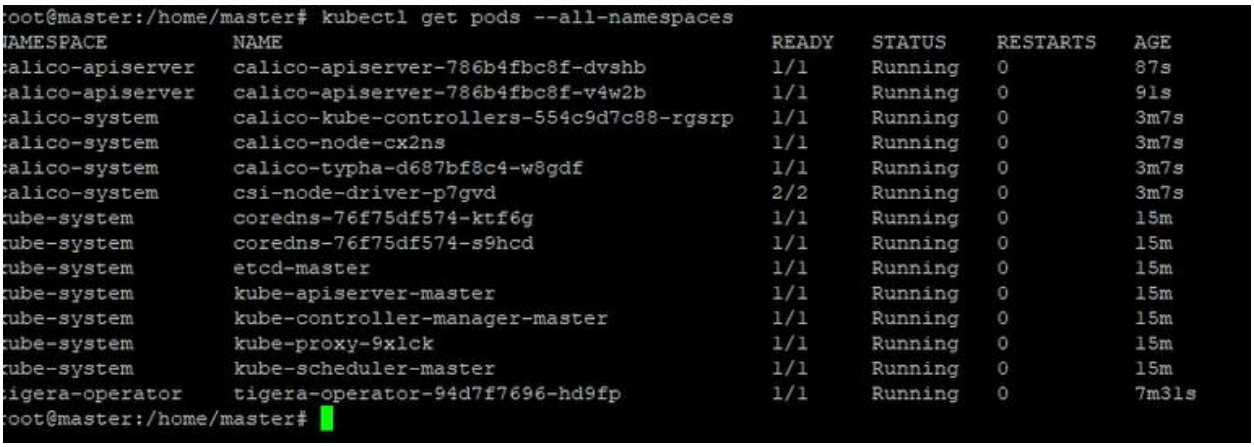
Figura 43
Información de los Pods en Tiempo Real



14. Una vez que todos los pods estuvieron en estado "Running", se verificó su estado final con el comando kubectl get pods --all-namespaces, como se muestra en la Figura 44.

```
# kubectl get pods --all-namespaces
```

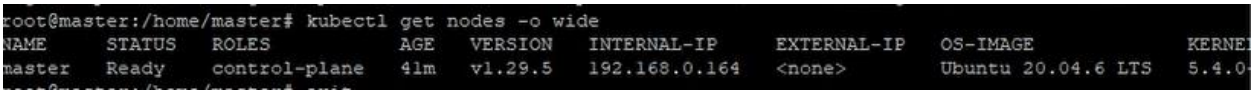
Figura 44
Muestra los Pods en Estado: Running



15. La configuración del nodo maestro se verificó utilizando el comando kubectl get nodes -o wide, como se ilustra en la Figura 45. Este comando proporcionó información detallada sobre el nodo creado.

```
# kubectl get nodes -o wide
```

Figura 45
Información del Nodo creado control-plane



10. Finalmente, se revisó el estado de los componentes del sistema de Kubernetes. Utilizando el comando kubectl -n kube-system get pod, se obtuvo información sobre los pods en el namespace kube-system. Además, se verificaron los servicios del clúster con el comando kubectl get svc, como se ve en la Figura 46.

```
# kubectl -n kube-system get pod
# kubectl get svc
```

Figura 46
Información Sobre los Pods e Información de los Servicios



Nota: los pods se están ejecutando en el espacio de nombres kube-system.

3.1.4. Instalación en los Nodos Worker

Después de instalar las tecnologías principales en toda la arquitectura y configurar el nodo principal (maestro), que es el que gestiona y orquesta contenedores de manera eficiente. Se procede a establecer los nodos worker y creación del clúster en base al diseño de la arquitectura establecida.

1. Para verificar la conectividad básica con el servidor de la API de Kubernetes en el nodo maestro, se intentó establecer una conexión HTTPS utilizando el comando curl https://k8scp:6443, como se ve en la Figura 47. Como era de esperar en este punto, la conexión falló debido a la falta de credenciales, indicando que se necesitaban obtener las credenciales apropiadas desde el nodo maestro.

```
# curl https://master:6443
```

Figura 47
Conexión Fallida

```
root@worker1:/home/user1# curl https://master:6443
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

Nota: La conexión es fallida debido a la falta de credenciales desde el nodo maestro.

2. El proceso de obtener estas credenciales y el comando para unir el nodo trabajador al clúster se realizó en el nodo maestro. Utilizando el comando kubeadm token create --print-join-command, como se muestra en la Figura 48, se generó un nuevo token y el comando completo para que el nodo trabajador se uniera al clúster.

```
$ kubeadm token create --print-join-command
```

Figura 48
Comando kubeadm en Nodo Maestro

```
master@master:~$ kubeadm token create --print-join-command
kubeadm join master:6443 --token f8m92c.yvz4die7mvwlgnmc --discovery-token-ca-cert-hash sha256:c41446fe8657395d32e8203eabe61e2d16885d598777a3df556aba1e5a27b9b3
```

3. En el nodo trabajador se ejecutó el comando proporcionado por el nodo maestro, que tenía la estructura kubeadm join master:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>, como se ilustra en la Figura 49. Este comando inició el proceso de unión del nodo trabajador al clúster, estableciendo una conexión segura con el nodo maestro y configurando el nodo para recibir y ejecutar pods.

Figura 49
Comando kubeadm Para Registrar el Nodo Worker en el Clúster

```
user1@worker1:~$ sudo kubeadm join master:6443 --token f8m92c.yvz4die7mvwlgnmc --discovery-token-ca-cert-has
9b3
[sudo] password for user1:
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

4. La verificación final de que el nodo trabajador se había unido exitosamente al clúster se realizó desde el nodo maestro. Utilizando el comando kubectl get nodes, como se muestra en la Figura 50, se pudo observar el nodo trabajador listado junto con el nodo maestro, confirmando que la comunicación y la configuración del clúster se habían establecido correctamente.

```
$ kubectl get nodes -o wide
```

Figura 50

Verificación de Comunicación entre los Nodos Worker y el Nodo Maestro



3.1.5. Despliegue del Dashboard, Instalación de Helm y Configuración de Metrics Server

Después de establecer la infraestructura básica del clúster con nodos maestro y trabajadores, el siguiente paso fue mejorar la gestión y monitoreo del clúster. Esto se logró mediante el despliegue del Dashboard de Kubernetes, se utilizó el gestor de paquetes Helm, y la configuración de Metrics Server para obtener métricas detalladas del clúster. A continuación, se muestran los pasos seguidos:

1. El proceso comenzó con la instalación de Helm, una herramienta para la gestión de aplicaciones en Kubernetes. Para instalar Helm en el nodo maestro, se utilizó los siguientes comandos:

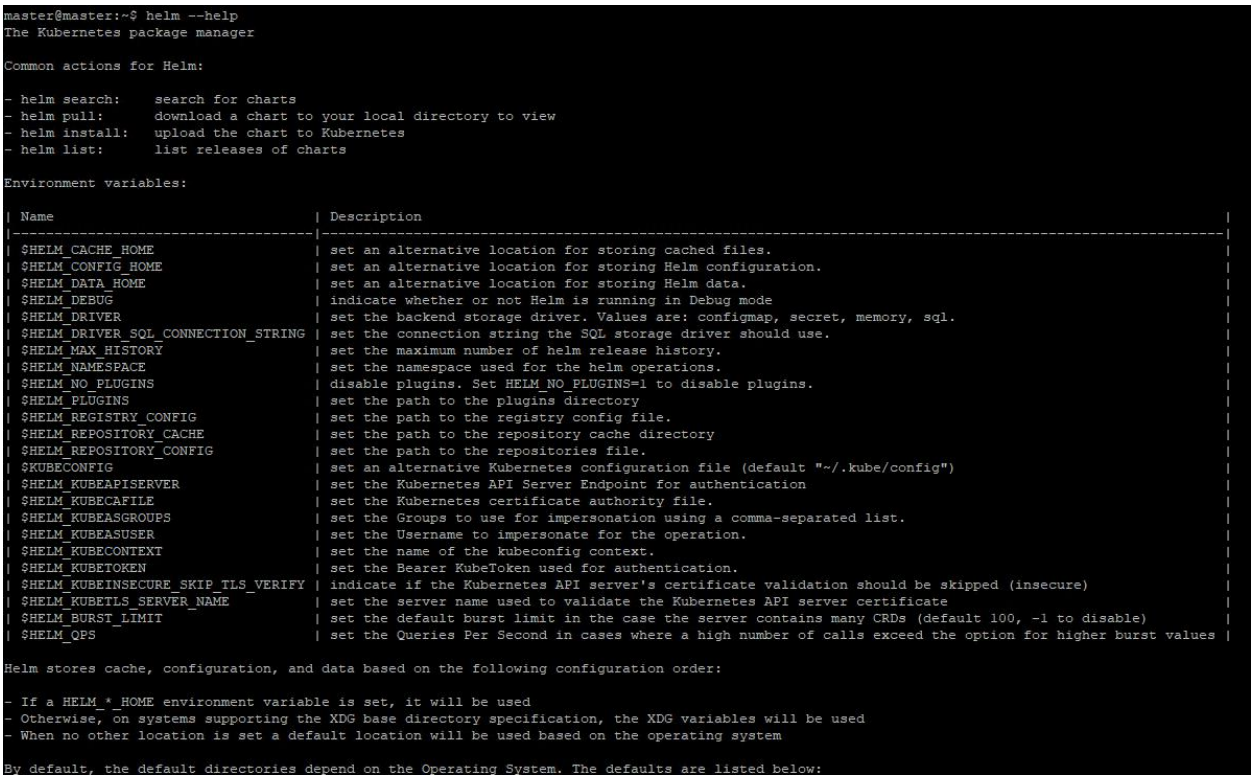
```
#curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
# chmod 700 get_helm.sh
# ./get_helm.sh
# helm --help
```

El primer comando (curl) descargó el script de instalación de Helm desde el repositorio oficial de GitHub. El segundo comando (chmod 700) otorgó permisos de ejecución exclusivos al usuario actual. El tercer comando ejecutó el script, llevando a cabo la

instalación de Helm. Finalmente, helm --help verificó la instalación y mostró la documentación de ayuda, como se ve en la Figura 51.

Figura 51

Verificación de la Instalación del Gestor de Paquetes Helm



2. El Dashboard de Kubernetes, una interfaz web intuitiva, se desplegó para facilitar la gestión visual y el monitoreo del clúster (Ver Figura 52). Para ello se utilizó el gestor de paquetes recientemente instalado. Además se verificó el tipo de servicio (ver Figura 53), siguiendo los siguientes comandos:

```
$ helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/
$ helm upgrade --install kubernetes-dashboard kubernetes-dashboard/kubernetes-dashboard --create-namespace --namespace kubernetes-dashboard
$ kubectl get all
```

Figura 52

Instalación del Dashboard a Través del Gestor de Paquetes Helm

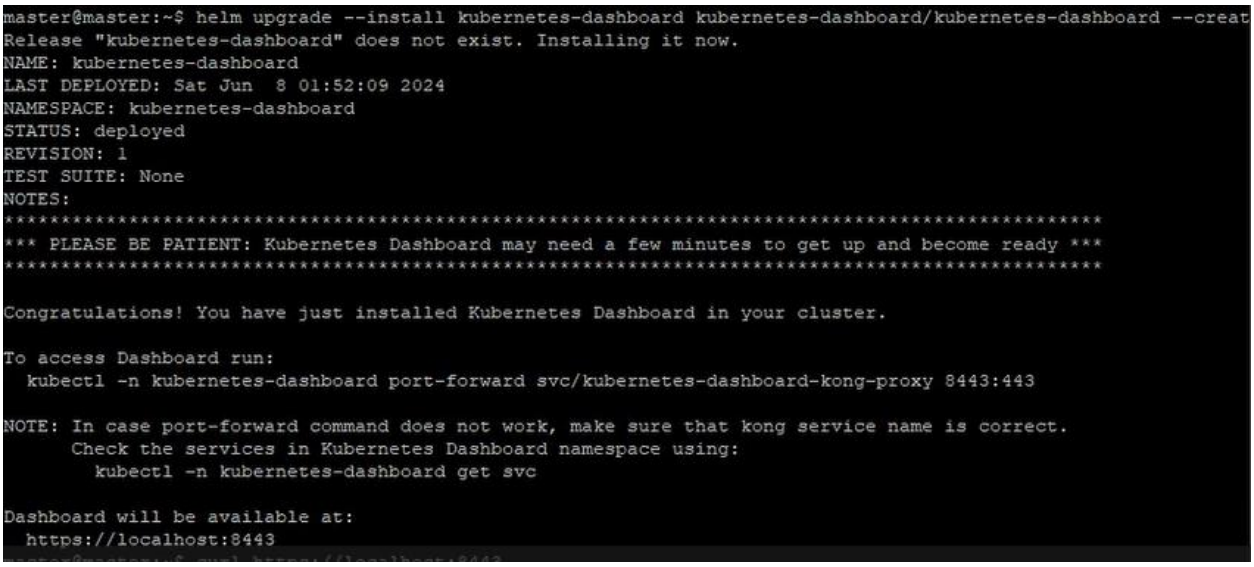
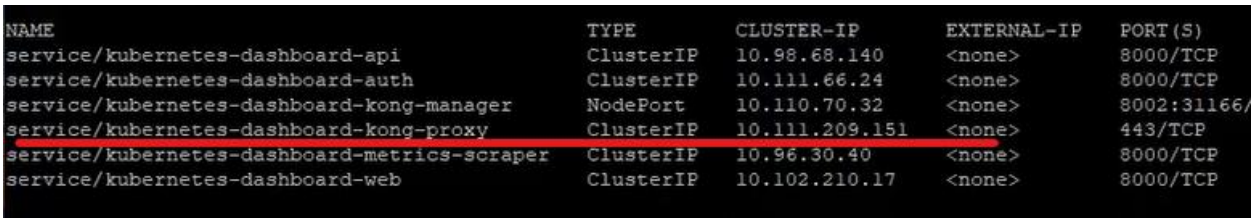


Figura 53

Recursos de Kubernetes Creados por el Despliegue del Dashboard



3. Se ha desplegado correctamente la interfaz Dashboard en el clúster. Sin embargo, por defecto el tipo de servicio es de tipo “ClusterIP”, es decir, solo es accesible dentro del clúster. Para exponer el servicio del Dashboard externamente se utilizó el reenvío de puertos con el uso de “NodePort” como tipo de servicio. Para ello se creó un archivo con el nombre values.yaml y se agregó lo siguiente (ver Figura 54):

```
kong:
 proxy:
   type: NodePort
 http:
   enabled: true
```

Figura 54
Archivo values.yaml

```
 GNU nano 4.8                                                    values.yaml
kong:
  proxy:
    type: NodePort
  http:
    enabled: true
```

4. Se actualizó el servicio existente a una nueva versión utilizando un archivo YAML creado. Este proceso se llevó a cabo mediante la ejecución de los siguientes comandos, cuyos resultados se muestran en las Figuras 55 y 56, respectivamente:

```
$ helm upgrade kubernetes-dashboard kubernetes-dashboard/kubernetes-dashboard -f values.yaml -n kubernetes-dashboard
$ kubectl get svc –all-namespaces
```

Figura 55
Actualización del Dashboard Modificado a NodePort

```
master@master:~/recursosk8$ helm upgrade kubernetes-dashboard kubernetes-dashboard/kubernetes-dashboard -f values.
Release "kubernetes-dashboard" has been upgraded. Happy Helming!
NAME: kubernetes-dashboard
LAST DEPLOYED: Sat Jun  8 01:58:15 2024
NAMESPACE: kubernetes-dashboard
STATUS: deployed
REVISION: 2
TEST SUITE: None
NOTES:
*************************************************************************************************
*** PLEASE BE PATIENT: Kubernetes Dashboard may need a few minutes to get up and become ready ***
*************************************************************************************************

Congratulations! You have just installed Kubernetes Dashboard in your cluster.

To access Dashboard run:
  kubectl -n kubernetes-dashboard port-forward svc/kubernetes-dashboard-kong-proxy 8443:443

NOTE: In case port-forward command does not work, make sure that kong service name is correct.
      Check the services in Kubernetes Dashboard namespace using:
        kubectl -n kubernetes-dashboard get svc

Dashboard will be available at:
```

Nota: el tipo de servicio se actualizó usando el archivo values.yaml

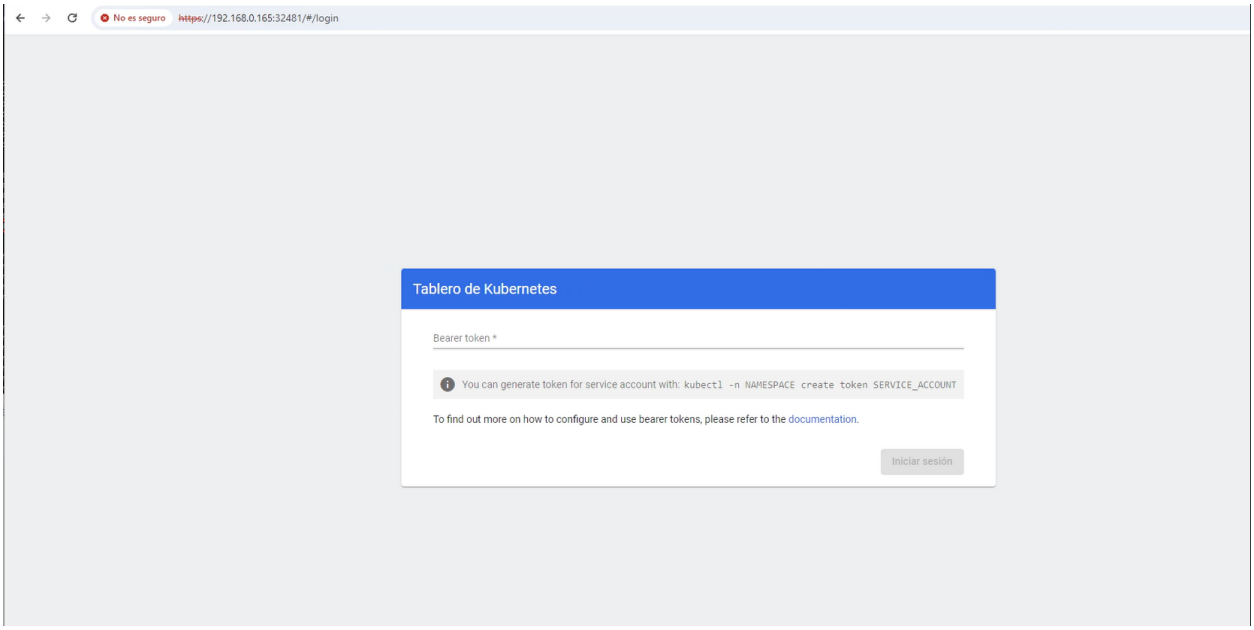
Figura 56
Verificación del Tipo de Servicio NodePort

```
kubernetes-dashboard   kubernetes-dashboard-api               ClusterIP   10.98.68.140    <none>   8000/TCP
kubernetes-dashboard   kubernetes-dashboard-auth              ClusterIP   10.111.66.24    <none>   8000/TCP
kubernetes-dashboard   kubernetes-dashboard-kong-manager      NodePort    10.110.70.32    <none>   8002:31166/TCP,8445:30546/TCP
kubernetes-dashboard   kubernetes-dashboard-kong-proxy        NodePort    10.111.209.151  <none>   443:32481/TCP
kubernetes-dashboard   kubernetes-dashboard-metrics-scraper   ClusterIP   10.96.30.40     <none>   8000/TCP
kubernetes-dashboard   kubernetes-dashboard-web               ClusterIP   10.102.210.17   <none>   8000/TCP
```

5. Para acceder al servicio del Dashboard, fue necesario ingresar la URL https://192.168.0.164:32481 en un navegador web, como se ilustra en la Figura 57.

Figura 57

Acceso al Dashboard Mediante el Puerto 32481



6. Para garantizar un acceso seguro al Dashboard, se creó un usuario con privilegios de administrador mediante un archivo YAML. Este archivo, denominado user.yaml, contenía los diferentes recursos como ServiceAccount, ClusterRoleBinding y Secret, tal como se muestra a continuación:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: v1
kind: Secret
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
  annotations:
    kubernetes.io/service-account.name: "admin-user"
type: kubernetes.io/service-account-token
```

7. Mediante el comando kubectl apply -f user.yaml, se crearon los recursos definidos en el archivo user.yaml, como se muestra en la Figura 58. Posteriormente, el comando kubectl get secret permitió obtener el token de acceso al Dashboard, cuyo resultado se aprecia en la Figura 59.

```
$ kubectl apply -f user.yaml
$ kubectl get secret admin-user -n kubernetes-dashboard -o jsonpath={".data.token"} | base64 -d
```

Figura 58
Recursos Creados Después de Aplicar el Archivo user.yaml

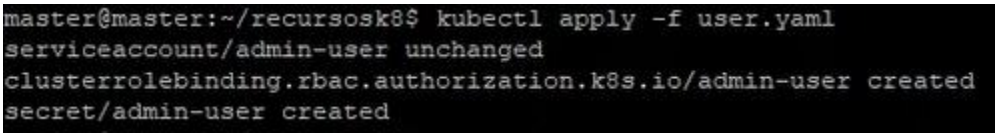


Figura 59

Token Generado para el Usuario admin-user



8. Finalmente, se ingresó el token generado en la página de inicio del Dashboard, como se puede apreciar en las Figuras 60 y 61, respectivamente.

Figura 60

Ingresando al Dashboard con el Token Generado

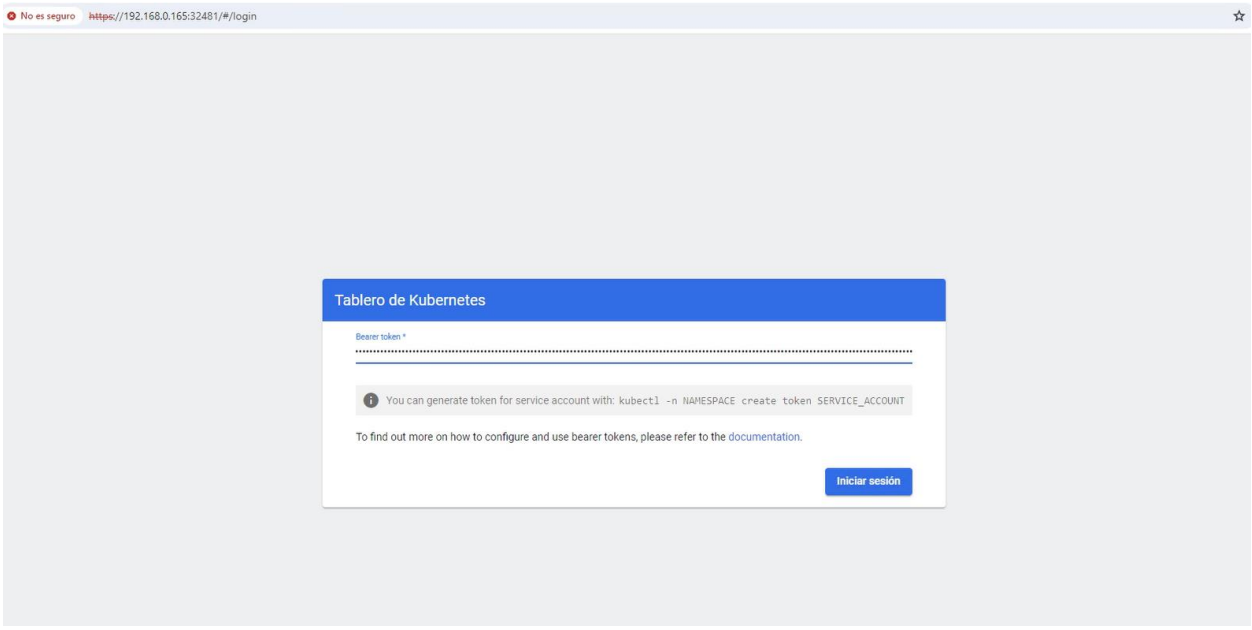
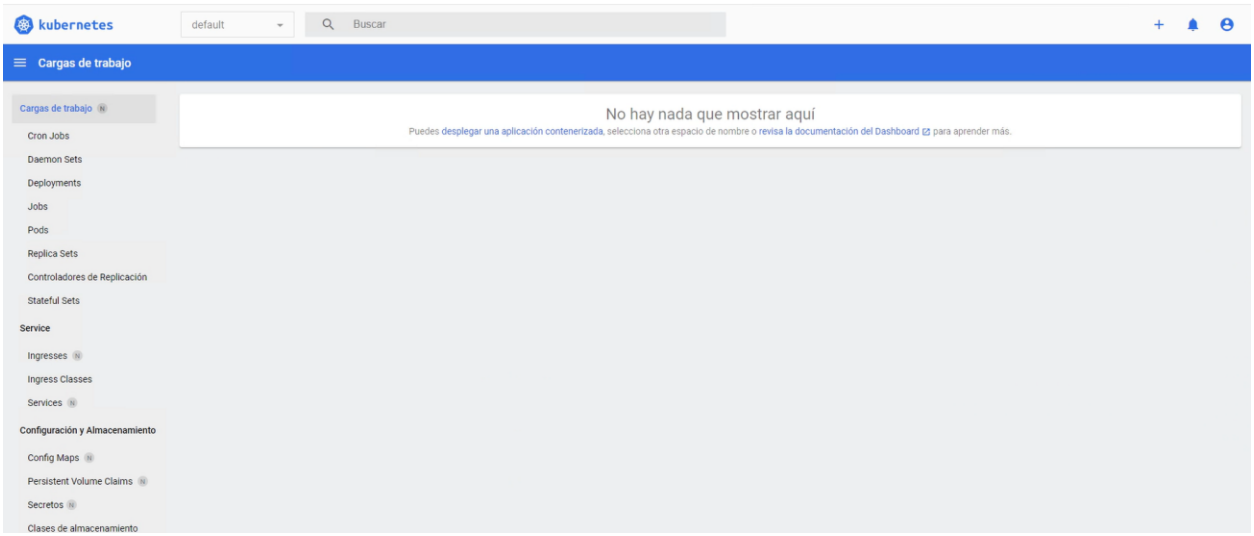


Figura 61
Página de Inicio del Dashboard

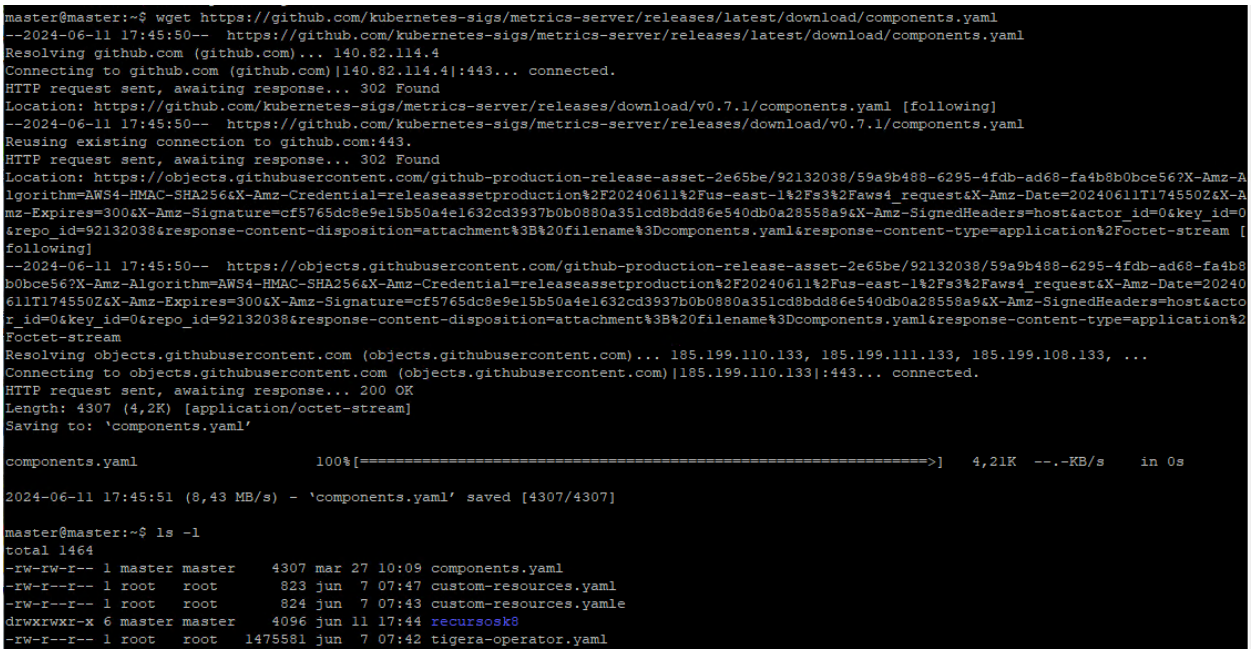


9. Metrics Server es un componente para recopilar métricas de recursos, como CPU y memoria, de los nodos y pods del clúster. Estas métricas son utilizadas por el Dashboard de Kubernetes. Para instalar Metrics Server, se descargó el documento https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml y se listó los archivos para verificar su descarga como se observa en la Figura 62 mediante los siguientes comandos:

```
$wget https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
$ ls -l
```

Figura 62

Descarga del Archivo components.yaml

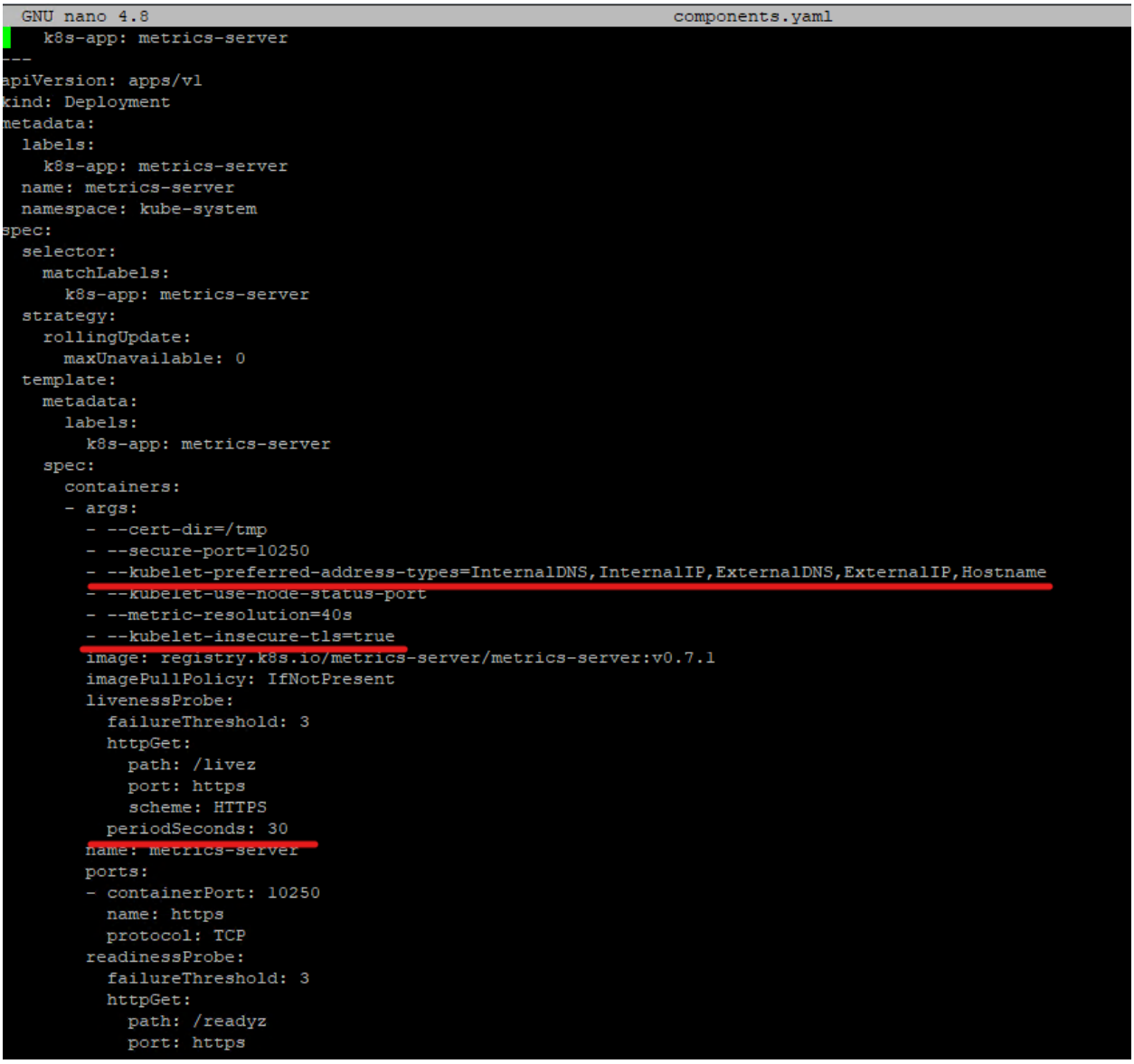


10. Se abrió el archivo de manifiesto descargado en el editor de texto nano y se realizó las modificaciones que se muestran en la Figura 63.

```
$ nano components.yaml
```

Figura 63

Modificaciones Realizadas en el Archivo components.yaml



11. Después de realizar las modificaciones, se guardan los cambios.
12. Para crear los recursos definidos en el archivo components.yaml, como Deployments, ServiceAccounts, ClusterRoles, y otros recursos necesarios para el despliegue de Metrics Server se aplicó el archivo de manifiesto modificado al clúster de Kubernetes utilizando el siguiente comando:

```
$ kubectl apply -f components.yaml
```

13. Se verificó que Metrics Server se haya instalado correctamente ejecutando el siguiente comando (ver Figura 64):

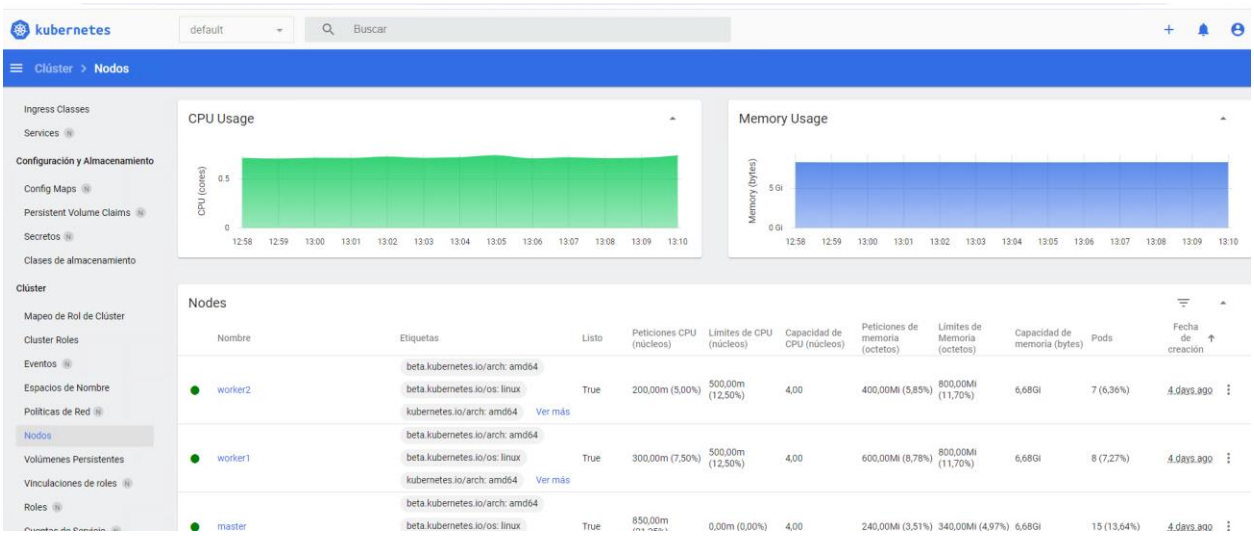
```
$ kubectl get all -n kube-system | grep metrics-server
```

Figura 64
Salida del Comando kubectl para Visualizar los Recursos de Metrics Server

```
master@master:~/recursosk8$ kubectl get all -n kube-system | grep metrics-server
pod/metrics-server-66485759bc-j2nlq   1/1     Running   0          2d20h
service/metrics-server   ClusterIP   10.109.9.142   <none>        443/TCP                  2d20h
deployment.apps/metrics-server   1/1     1            1           2d20h
replicaset.apps/metrics-server-66485759bc   1         1         1       2d20h
```

14. Finalmente, se ingresó nuevamente al Dashboard para ver las métricas de de CPU y RAM, como se observa en la Figura 65.

Figura 65
Métricas de los Nodos del Clúster



3.1.6. Implementación de Aplicaciones en la Arquitectura

Después de haber establecido una infraestructura sólida para el clúster de Kubernetes, el siguiente paso fundamental fue implementar aplicaciones en esta arquitectura de orquestación de contenedores. La implementación de aplicaciones en Kubernetes implica varios componentes clave, como Deployments, Servicios y otros recursos de Kubernetes. En esta sección, se detalla el proceso de implementación, incluyendo el despliegue de aplicaciones y la exposición de los servicios correspondientes.

Antes de desplegar las aplicaciones y crear los servicios es importante conocer la arquitectura de las aplicaciones que se van a desplegar, considerando la aplicación de Gestión de Eventos definida se consideraron las dependencias entre los servicios y las necesidades de red. Se identificaron los componentes de la aplicación y sus interacciones entre los frontend, el backend y la base de datos. Además, se evaluaron las necesidades de conectividad en la red, asegurando que todos los servicios pudieran comunicarse de manera eficiente.

A continuación, se detallan los pasos seguidos para el despliegue de la aplicación:

1. Se llevó a cabo la creación de imágenes de contenedores para cada componente de la aplicación, en este caso fueron tres, mediante el comando docker build. Este proceso aseguró la encapsulación adecuada de cada componente con sus dependencias.
2. Se estableció un registro de contenedores en Docker Hub, que es un repositorio de imágenes Docker, para el almacenamiento y gestión de las imágenes creadas. Se utilizó el comando docker login para autenticarse y posteriormente se enviaron las imágenes al repositorio mediante docker push, garantizando así su disponibilidad para el despliegue.
3. Se elaboró un archivo YAML para definir los recursos de Kubernetes necesarios. Este incluyó Deployments, Services y ConfigMaps.

   El archivo de configuración eventos-app.yaml incluyó definiciones para:

   - Un Deployment y Service para el backend (backend-eventos-gestion)
   - Un Deployment y Service para el frontend de administración (frontend-eventos-admin)
   - Un Deployment y Service para el frontend de contador (frontend-eventos-counter)

   A continuación, se detalla el archivo de configuración utilizado:

   ```
   ---
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: backend-eventos-gestion
   spec:
     replicas: 1
   ```

```
  selector:
    matchLabels:
      app: backend-eventos-gestion
  template:
    metadata:
      labels:
        app: backend-eventos-gestion
    spec:
      containers:
      - name: backend-eventos-gestion
        image: dasanchez1/backend-eventos-gestion:v1
        ports:
        - containerPort: 9101  # El backend escucha por el puerto 9101
        envFrom:
        - configMapRef:
            name: backend-env-config
---
apiVersion: v1
kind: Service
metadata:
  name: backend-eventos-gestion-service
spec:
  selector:
    app: backend-eventos-gestion
  type: NodePort
  ports:
  - port: 9101
    targetPort: 9101
    nodePort: 30114
---
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: frontend-eventos-admin
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend-eventos-admin
  template:
    metadata:
      labels:
        app: frontend-eventos-admin
    spec:
      containers:
      - name: frontend-eventos-admin
        image: dasanchez1/frontend-eventos-admin:v1
        ports:
        - containerPort: 443   # Puerto HTTPS expuesto por el frontend
        envFrom:
        - configMapRef:
            name: front-admin-env-config
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-eventos-admin-service
spec:
  selector:
    app: frontend-eventos-admin
  type: NodePort
  ports:
```

```
  - port: 443  # Puerto del servicio (HTTPS)
    targetPort: 443  # Puerto del contenedor
    nodePort: 30443  # Puerto expuesto en el nodo (HTTPS)

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-eventos-counter
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend-eventos-counter
  template:
    metadata:
      labels:
        app: frontend-eventos-counter
    spec:
      containers:
      - name: frontend-eventos-counter
        image: dasanchez1/frontend-eventos-counter:v1
        ports:
        - containerPort: 443  # Puerto HTTPS expuesto por el frontend
        envFrom:
        - configMapRef:
            name: front-counter-env-config
---
apiVersion: v1
kind: Service
metadata:
```

```
  name: frontend-eventos-counter-service
spec:
  selector:
    app: frontend-eventos-counter
  type: NodePort
  ports:
  - port: 443  # Puerto del servicio (HTTPS)
    targetPort: 443  # Puerto del contenedor
    nodePort: 30444  # Puerto expuesto en el nodo (HTTPS)
```

Cada Deployment especificó detalles como el número de réplicas, la imagen del contenedor a utilizar y los puertos expuestos. Los Services se configuraron como tipo NodePort para permitir el acceso externo a los componentes de la aplicación.

La aplicación de esta configuración en la infraestructura se realizó mediante el comando kubectl apply -f eventos-app.yaml, lo que desplegó efectivamente todos los componentes de la aplicación en el clúster de Kubernetes.

3.2. Entrega del Entorno de Pruebas:

En la fase de entrega se aseguró que la infraestructura de Kubernetes implementada esté completamente operativa y lista para ser utilizada por el departamento de Tecnologías de la Información de la PUCE-Ibarra. Esta fase incluye la validación final de la funcionalidad del clúster y la formalización de la transferencia de la infraestructura.

3.2.1. Validación del Clúster

Pruebas de funcionalidad del clúster

Se realizaron diversas pruebas para validar la funcionalidad básica del clúster de Kubernetes. Estas pruebas incluyeron la verificación de la conectividad entre nodos, el despliegue y escalabilidad de aplicaciones, así como la funcionalidad del Dashboard de Kubernetes.

Matriz de Pruebas y Resultados

La matriz de pruebas es un instrumento fundamental para documentar y evaluar la efectividad de las pruebas realizadas sobre la infraestructura de Kubernetes implementada. A continuación, se presenta la matriz de pruebas detallada que incluye el objetivo de cada prueba, el procedimiento a seguir, los criterios de éxito y los resultados obtenidos.

Tabla 7
Matriz de Pruebas de Funcionalidad del Clúster y Resultados

ID	Tipo de Prueba	Descripción	Procedimiento	Resultado Esperado	Resultado Obtenido
1	Prueba de Conectividad entre Nodos	Verificar la conectividad entre el nodo maestro y los nodos trabajadores.	Ejecutar ping y kubectl get nodes desde el nodo maestro y los nodos trabajadores.	Los nodos deben responder a los pings y estar en estado "Ready" al ejecutar kubectl get nodes.	Los nodos responden correctamente y están en estado "Ready" (ver Anexo V).
2	Prueba de Despliegue de Aplicaciones	Desplegar una aplicación de prueba en el clúster.	Crear un Deployment y un Service para una aplicación de prueba.	La aplicación debe desplegarse sin errores y ser accesible a través del Service.	La aplicación se despliega correctamente y es accesible (ver Anexo VI).
3	Prueba de Escalabilidad	Escalar la aplicación de prueba aumentando el número de réplicas.	Usar kubectl scale para aumentar las réplicas de la aplicación.	Las réplicas adicionales deben iniciarse y estar en estado "Running".	Las réplicas adicionales se inician y están en estado "Running" (ver Anexo VII).
4	Prueba de Funcionalidad del Dashboard	Acceder y utilizar el Dashboard de Kubernetes para gestionar el clúster.	-Iniciar sesión en el Dashboard. -Realizar tareas de gestión como	Los usuarios deben poder realizar tareas de gestión sin problemas.	El Dashboard es funcional y permite la gestión del clúster (ver Anexo VIII).

			despliegue de aplicaciones y monitoreo de recursos.		
5	Prueba de Disponibilidad	Simular fallos en nodos y verificar la recuperación del clúster y la disponibilidad de las aplicaciones.	-Apagar nodos trabajadores. -Verificar la redistribución de pods y la disponibilidad de las aplicaciones.	Las aplicaciones deben permanecer disponibles o recuperarse rápidamente.	Las aplicaciones se redistribuyen y permanecen disponibles tras el fallo de nodos (ver Anexo IX).

3.2.2. Formalización de la Transferencia

Como parte integral de esta formalización, se llevó a cabo una reunión final de entrega, en la cual se presentaron los resultados del proyecto, se realizó una demostración del funcionamiento del entorno.

La formalización fue realizada mediante la carta de finalización del proyecto, firmado por parte del jefe de la Unidad de TI. Este documento, que se puede consultar en el Anexo X, certifica la conclusión satisfactoria del proyecto y la aceptación formal del entorno de Kubernetes por parte de la Unidad de TI.

3.3. Uso del Entorno de Pruebas:

La fase de Uso del entorno de Kubernetes marca el inicio de la operación regular del sistema por parte de la Unidad de Tecnología de la Información (TI). Es importante destacar que esta fase queda fuera del alcance de la presente investigación, la cual se ha centrado en el diseño, implementación y entrega del entorno de Kubernetes. Una vez completada la entrega formal, la responsabilidad de la operación diaria y evolución de la infraestructura recae en la Unidad de TI. Esta transición representa el punto en el que el proyecto de investigación concluye, habiendo cumplido su objetivo de proporcionar una infraestructura inicial robusta y funcional de Kubernetes.

CONCLUSIONES

Tras finalizar la implementación de la infraestructura basada en Kubernetes para el despliegue de aplicaciones en contenedores en el entorno de pruebas de la Unidad de Tecnologías de la Información, y en conformidad con los requisitos de la arquitectura establecida en el proyecto, se llegó a las siguientes conclusiones:

- La adopción de Kubernetes ha permitido una gestión más eficiente y automatizada de las aplicaciones, reduciendo significativamente la carga operativa manual. Esto ha resultado en una mayor disponibilidad y fiabilidad de los servicios tecnológicos ofrecidos por la Unidad de TI.
- La capacidad de Kubernetes para escalar aplicaciones de manera horizontal ha proporcionado una mayor flexibilidad para ajustarse a las demandas variables. Esto es crucial para satisfacer las necesidades de los usuarios finales y asegurar un rendimiento óptimo bajo diferentes cargas de trabajo.
- El uso de Kubernetes ha simplificado los procesos de implementación y actualización de aplicaciones, permitiendo realizar estas tareas de manera más rápida.
- Durante el desarrollo del proyecto, se encontraron desafíos al trabajar con sistemas operativos como AlmaLinux o Rocky Linux. Es importante destacar que estos problemas no se debieron a incompatibilidades inherentes de estos sistemas con Kubernetes, sino a la falta de documentación específica y experiencia previa con estas distribuciones en el contexto de Kubernetes. Esta situación subraya la importancia de la disponibilidad de recursos y documentación actualizados para diversas plataformas, así como la necesidad de una formación continua en las tecnologías.
- Se evidenció la crucial importancia de una correcta instalación y configuración del plugin CNI (Container Network Interface). La experiencia demostró que una implementación inadecuada del CNI puede resultar en un clúster Kubernetes no funcional o con limitaciones significativas. Este aspecto resalta la necesidad de prestar especial atención a la capa de red en la implementación de Kubernetes, ya que es fundamental para la comunicación entre pods y servicios.

- Se observó una mejora en la utilización de recursos de hardware, gracias a la capacidad de Kubernetes para distribuir eficientemente las cargas de trabajo entre los nodos disponibles.

RECOMENDACIONES

Basado en los resultados obtenidos, se presentan las siguientes recomendaciones para futuras implementaciones y mejoras continuas en la infraestructura de TI:

- Es fundamental que el personal de TI reciba capacitación continua en el uso y administración de Kubernetes y tecnologías relacionadas. Esto asegurará que puedan aprovechar al máximo las capacidades de la plataforma.
- Se recomienda continuar con la automatización de procesos operativos y de gestión de aplicaciones mediante herramientas de CI/CD (Integración Continua/Despliegue Continuo). Esto no solo mejorará la eficiencia, sino que también reducirá el riesgo de errores humanos.
- Implementar soluciones de monitoreo y seguridad robustas para el clúster de Kubernetes. Esto incluye el monitoreo continuo de recursos, la implementación de alertas tempranas y la aplicación de medidas de seguridad avanzadas para proteger los datos y las aplicaciones.
- Mantener una documentación detallada y actualizada de la arquitectura, configuraciones y procedimientos operativos.
- Usar aplicaciones desacopladas, es decir, diseñar y desarrollar aplicaciones siguiendo una arquitectura de microservicios.

REFERENCIAS BIBLIOGRÁFICA

Aragón Puetate, J. A. (2018). *Sistema de evaluación docente para apoyo de la gestión universitaria de la PUCE-SI basado en arquitectura de software tres capas.* [Informe de pregrado, Pontificia Universidad Católica del Ecuador Sede Ibarra]. Obtenido de https://dspace.pucesi.edu.ec/handle/11010/133

Delgado Pedraza, L., & Pineda Parra, O. D. (2019). *MODELO PARA LA ORQUESTACIÓN DE MICROSERVICIOS CON KUBERNETES APLICADO AL SERVICIO DE CONTROL DE VERSIONES GIT*. [TRABAJO DE GRADO, UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS]. Obtenido de https://repository.udistrital.edu.co/bitstream/handle/11349/22424/DelgadoPedrazaLeonardo2019.pdf?sequence=1&isAllowed=y

Docker. (s.f.). *Docker*. Recuperado el 22 de March de 2024, de Docker: Accelerated Container Application Development: https://www.docker.com

Fernández Ameijeiras, J. A. (2017). *Orquestación de aplicaciones corporativas mediante virtualización de contenedores usando Docker.* [Tesis de grado de ingeniería informática, Universitat Oberta de Catalunya]. Obtenido de http://hdl.handle.net/10609/60465

Google Cloud. (26 de January de 2023). *Deployment | Google Kubernetes Engine (GKE).* Recuperado el 22 de March de 2024, de Google Cloud: https://cloud.google.com/kubernetes-engine/docs/concepts/deployment?hl=es-419

Hurtado, D. (2018). *TECNOLOGÍA DE CONTENEDORES DE SOFTWARE EN ENTORNOS DE PRUEBAS.* [Tesis de Grado, Pontificia Universidad Católica del Ecuador].

IBM. (s.f.). *¿Qué son las redes de Kubernetes?* Recuperado el 30 de Marzo de 2024, de IBM: https://www.ibm.com/mx-es/topics/kubernetes-networking

IBM. (s.f.). *En qué consiste la contenerización.* Recuperado el 22 de March de 2024, de IBM: https://www.ibm.com/es-es/topics/containerization

Kubernetes Documentation. (19 de September de 2023). *Overview.* Recuperado el 20 de March de 2024, de Kubernetes: https://kubernetes.io/docs/concepts/overview/#why-you-need-kubernetes-and-what-can-it-do

Microsoft. (31 de March de 2023). *Contenedores frente a máquinas virtuales.* Recuperado el 22 de March de 2024, de Microsoft Learn: https://learn.microsoft.com/es-es/virtualization/windowscontainers/about/containers-vs-vm

Microsoft Azure. (2018). *¿Qué es DevOps? Explicación de DevOps.* Recuperado el 15 de April de 2024, de Microsoft Azure: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops

Microsoft Azure. (s.f.). *Escalado horizontal frente a escalado vertical.* Recuperado el 28 de March de 2024, de Microsoft Azure: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/scaling-out-vs-scaling-up

Montes Bravo, J. D. (2022). *IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SERVIDOR WEB CON HOSTS VIRTUALES MEDIANTE KUBERNETES.* [Tesis de formación de tecnólogo, Escuela Politécnica Nacional]. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/23057/1/CD%2012476.pdf

Nedeltcheva, G. N., Xiang, B., Niculut, L., & Benedetto, D. (2023). Challenges Towards Modeling and Generating Infrastructure-as-Code. *Association for Computing Machinery*. Obtenido de https://doi.org/10.1145/3578245.3584937

Packard, M., Stubbs, J., Drake, J., & Garcia, C. (2021). Real-World, Self-Hosted Kubernetes Experience. *Association for Computing Machinery*. Obtenido de https://doi.org/10.1145/3437359.3465603

Pan, Y., Ian, C., Brasileiro, F., Jayaputera, G., & Sinnott, R. (2019). A Performance Comparison of Cloud-Based Container Orchestration Tools. *IEEE International Conference on Big Knowledge*, 191-198. Obtenido de https://www.researchgate.net/publication/338263213

Paredes Calle, D. (2021). *IMPLEMENTACIÓN DE MICROSERVICIOS CON KUBERNETES PARA MEJORAR LA ESCALABILIDAD EN LA ARQUITECTURA DE LA APLICACIÓN EN UNA EMPRESA DEDICADA AL COMERCIO ELECTRÓNICO.* [Trabajo de suficiencia profesional, UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR]. Obtenido de https://repositorio.untels.edu.pe/jspui/handle/123456789/876

Perré, Y. (2022). *Kubernetes: gérez la plateforme de déploiement de vos applications conteneurisées.* Éditions ENI.

Poniszewska-Maranda, A., & Czechowska, e. (2021). Kubernetes Cluster for Automating Software Production Environment. *Sensors, 21*(1910). Obtenido de https://doi.org/10.3390/s21051910

Poulton, N. (2023). *The kubernetes book.* NIGEL POULTON LTD.

Prácticas Recomendadas de Configuración. (28 de June de 2023). Recuperado el 28 de March de 2024, de Kubernetes: https://kubernetes.io/es/docs/concepts/configuration/overview/

Vohra, D. (2016). *Kubernetes Microservices with Docker.* Apress. Obtenido de https://books.google.com.ec/books?hl=es&lr=&id=rJUFDAAAQBAJ&oi=fnd&pg=PR3&dq=kubernetes+docker&ots=dE9CZTy5tj&sig=4iH3nQyFDSu5S9rrlzsnAZozcus&redir_esc=y#v=onepage&q=kubernetes%20docker&f=false

Volumes. (2024). Recuperado el 29 de March de 2024, de Kubernetes: https://kubernetes.io/es/docs/concepts/storage/volumes/

ANEXOS

ANEXO I

Cronograma de actividades

Actividad	Descripción	Duración
	Fase de Planificación	
1. Recolección de datos	Fase de recopilación de información relevante para el proyecto.	5 días
1.1. Técnicas de recolección de datos	Realización de entrevistas con el personal clave de la Unidad de TI para obtener detalles y requisitos.	2 días
1.2. Análisis de las entrevistas	Análisis y procesamiento de los datos recopilados en las entrevistas.	3 días
2. Determinación de aplicaciones contenerizadas	Identificación de las aplicaciones que serán desplegadas en contenedores dentro del clúster de Kubernetes.	5 días
3. Recursos disponibles	Evaluación de los recursos disponibles de los servidores.	3 días
4. Análisis del sistema operativo	Selección y análisis del sistema operativo más adecuado para el entorno de Kubernetes.	3 días

5. Diseño de la Arquitectura Kubernetes	Fase de diseño de la arquitectura de Kubernetes para el entorno de pruebas.	10 días
5.1. Topología del clúster	Definición de la topología del clúster, incluyendo la cantidad y distribución de nodos maestros y trabajadores.	9 días
5.2. Definición de recursos de los nodos Master y Worker	Determinación de los recursos (CPU, memoria, almacenamiento) asignados a los nodos maestros y trabajadores.	1 día
	Fase de Desarrollo	
6. Desarrollo del entorno de pruebas		30 días
6.1. Instalación del sistema operativo en los servidores	Instalación del sistema operativo seleccionado en los servidores asignados para el clúster de Kubernetes.	2 días
6.2. Instalación inicial en los servidores del clúster de Kubernetes Maestro-Workers	Configuración inicial de los servidores para el clúster de Kubernetes, incluyendo la configuración de red y otros requisitos previos.	15 días
6.3. Instalación en el nodo Maestro	Instalación y configuración del nodo maestro de Kubernetes, responsable del control del clúster.	2 días
6.4. Instalación en los nodos Worker	Instalación y configuración de los nodos trabajadores de Kubernetes, responsables de ejecutar las cargas de trabajo.	1 día
6.5. Despliegue del Dashboard, instalación de Helm y	Implementación del Dashboard de Kubernetes, instalación de Helm (gestor de paquetes) y	5 días

configuración de Metric Server	configuración de Metric Server para monitoreo de recursos.	
6.6. Implementación de aplicaciones en la arquitectura	Despliegue de aplicaciones contenerizadas en el clúster de Kubernetes y configuración de servicios correspondientes.	5 días
	Fase de Entrega	
7. Validación del Clúster	Realización de pruebas para garantizar que el entorno de Kubernetes esté funcionando correctamente y cumpla con los requisitos establecidos.	10 días
7.1. Pruebas de funcionalidad del clúster	Ejecución de pruebas para verificar la funcionalidad básica del clúster de Kubernetes, asegurándose de que los nodos maestro y trabajador operen correctamente.	5 días
7.2. Matriz de pruebas y resultados	Involucrar a usuarios clave para validar que el entorno cumple con sus expectativas y necesidades operativas.	3 días
8. Formalización de la Transferencia	Formalización de la transferencia del entorno de pruebas a la unidad responsable.	1 día
	Fase de Uso	
9. Uso del entorno de pruebas	-	-

ANEXOS II

Entrevista 1

Fecha: 20 de febrero del 2024

Lugar: Oficina de la Unidad de TI en la PUCE-Ibarra

Duración: 50 minutos

Entrevistado: Franklin Sánchez (jefe de la Unidad de TI).

1. ¿Cómo se realiza actualmente el proceso de implementación y gestión de aplicaciones?
2. ¿Cuáles son los principales desafíos que enfrenta actualmente la Unidad en cuanto a infraestructura y gestión de aplicaciones?
3. ¿Qué estrategias o soluciones han implementado para abordar estos desafíos?
4. ¿Están familiarizados con la tecnología de contenedores y Kubernetes?
5. ¿Qué beneficios esperan obtener al utilizar Kubernetes para la implementación de aplicaciones en contenedores?
6. ¿Cuáles son los recursos de hardware, softwares disponibles y sistema operativo para implementar un entorno de pruebas con Kubernetes?
7. ¿Qué aplicaciones o servicios les gustaría probar en la infraestructura de Kubernetes?
8. ¿Cómo le gustaría interactuar con la infraestructura Kubernetes en un ambiente de pruebas?

ANEXO III

Entrevista 2

Fecha: 20 de febrero del 2024

Lugar: Oficina de la Unidad de TI en la PUCE-Ibarra

Duración: 40 minutos

Entrevistado: Cristian Paredes (Área de desarrollo de software).

1. ¿Cuáles son los principales desafíos que enfrenta actualmente la Unidad en cuanto a infraestructura y gestión de aplicaciones?
2. ¿Están familiarizados con la tecnología de contenedores y Kubernetes?
3. ¿Qué beneficios esperan obtener al utilizar Kubernetes para la implementación de aplicaciones en contenedores?
4. ¿Cuáles son los requisitos y características específicas de estas aplicaciones (arquitectura, servicios externos)?
5. ¿Cuáles son los recursos de hardware, softwares disponibles y sistema operativo para implementar un entorno de pruebas con Kubernetes?
6. ¿Cómo le gustaría interactuar con la infraestructura Kubernetes en un ambiente de pruebas?

ANEXO IV

Entrevista 3

Fecha: 20 de febrero del 2024

Lugar: Oficina de la Unidad de TI en la PUCE-Ibarra

Duración: 30 minutos

Entrevistado: Paul Enríquez (Área de redes y comunicación).

1. ¿Están familiarizados con la tecnología de contenedores y Kubernetes?
2. ¿Qué beneficios esperan obtener al utilizar Kubernetes para la implementación de aplicaciones en contenedores?
3. ¿Cuáles son los recursos de hardware, softwares disponibles y sistema operativo para implementar un entorno de pruebas con Kubernetes?
4. ¿Cómo le gustaría interactuar con la infraestructura Kubernetes en un ambiente de pruebas?

ANEXO V

Figura 66
Nodos del Clúster en Estado "Ready"

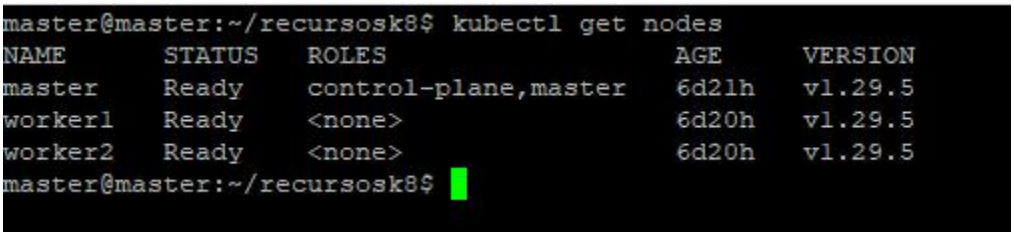


Figura 67
Ping desde el Nodo Master al Worker1

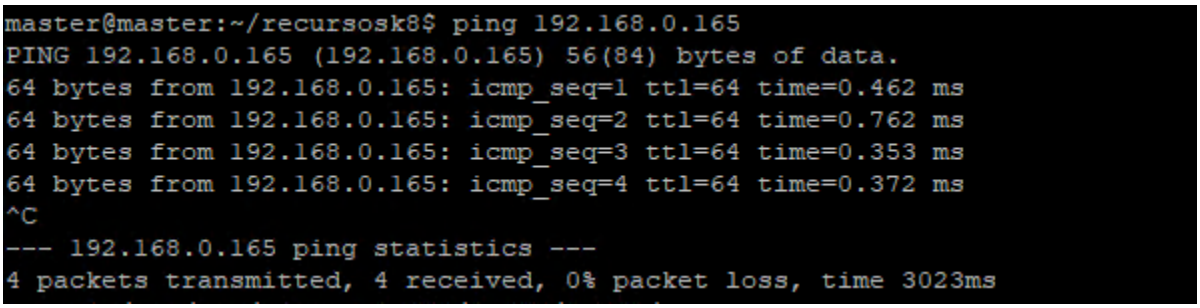
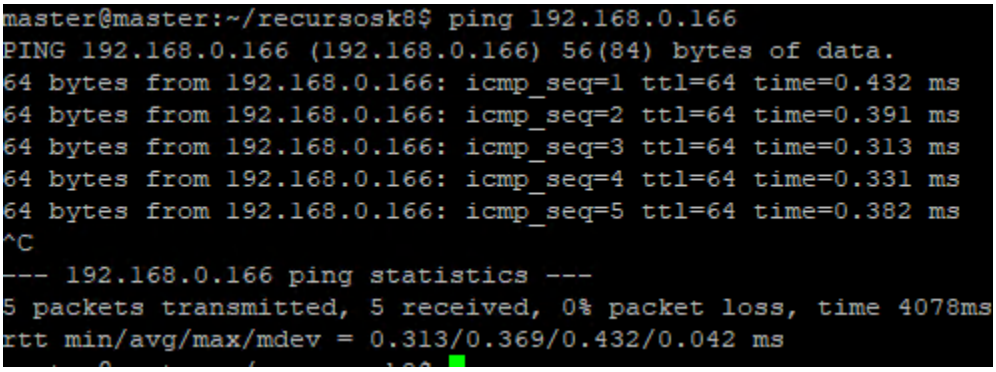


Figura 68
Ping desde el Nodo Master al Worker2



ANEXO VI

Figura 69
Creación de Recursos Pods, Services, Deployments para una Aplicación

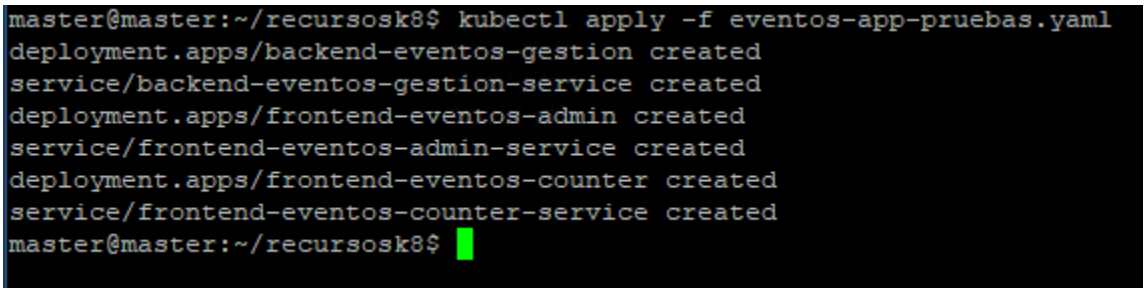


Figura 70
Pods Creados Funcionando Correctamente

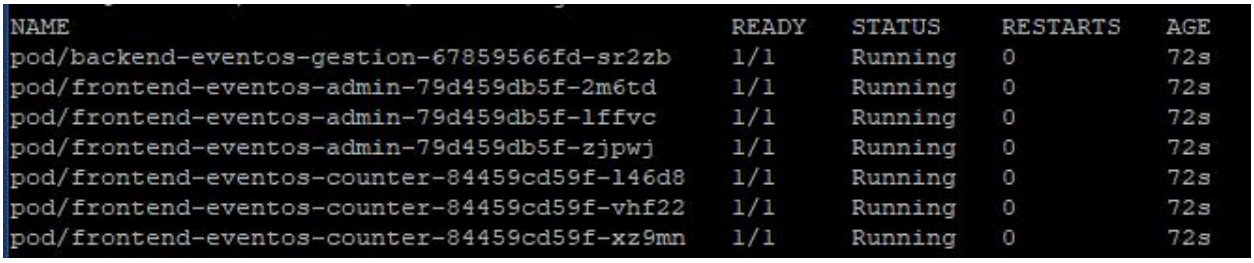


Figura 71
Services de la Aplicación

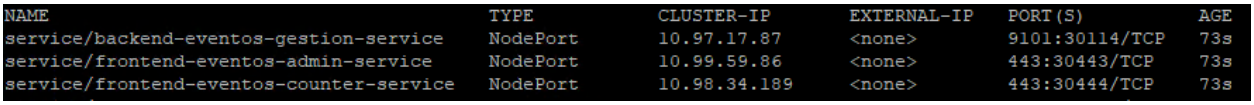


Figura 72
Deployments de la Aplicación

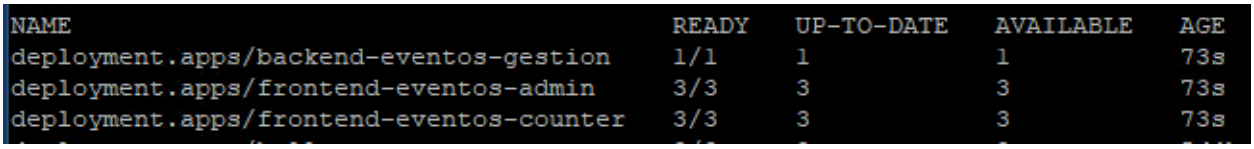


Figura 73
Réplicas de los Recursos de la Aplicación

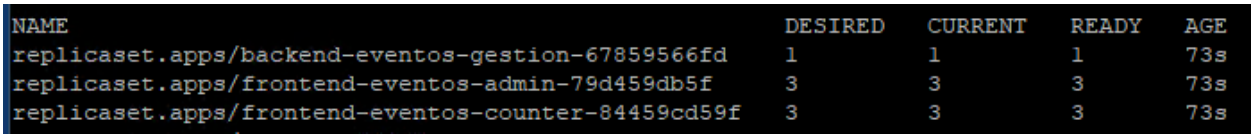


Figura 74

Recursos Creados Mostrados desde el Dashboard (UI)

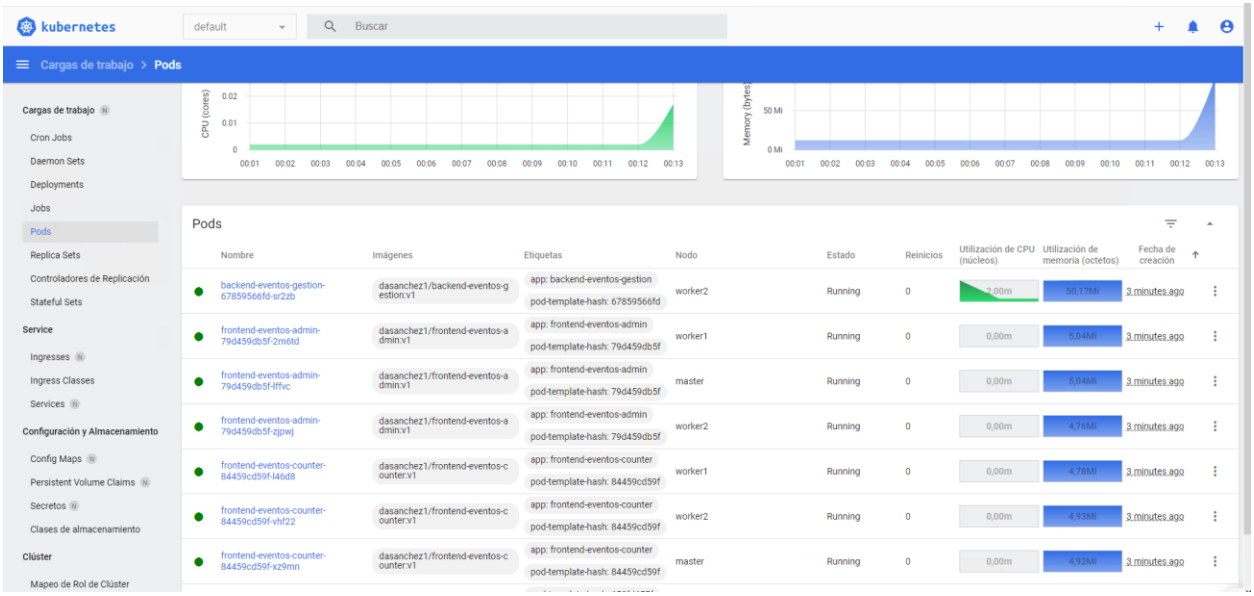


Figura 75

Pantalla de Administración de la Aplicación

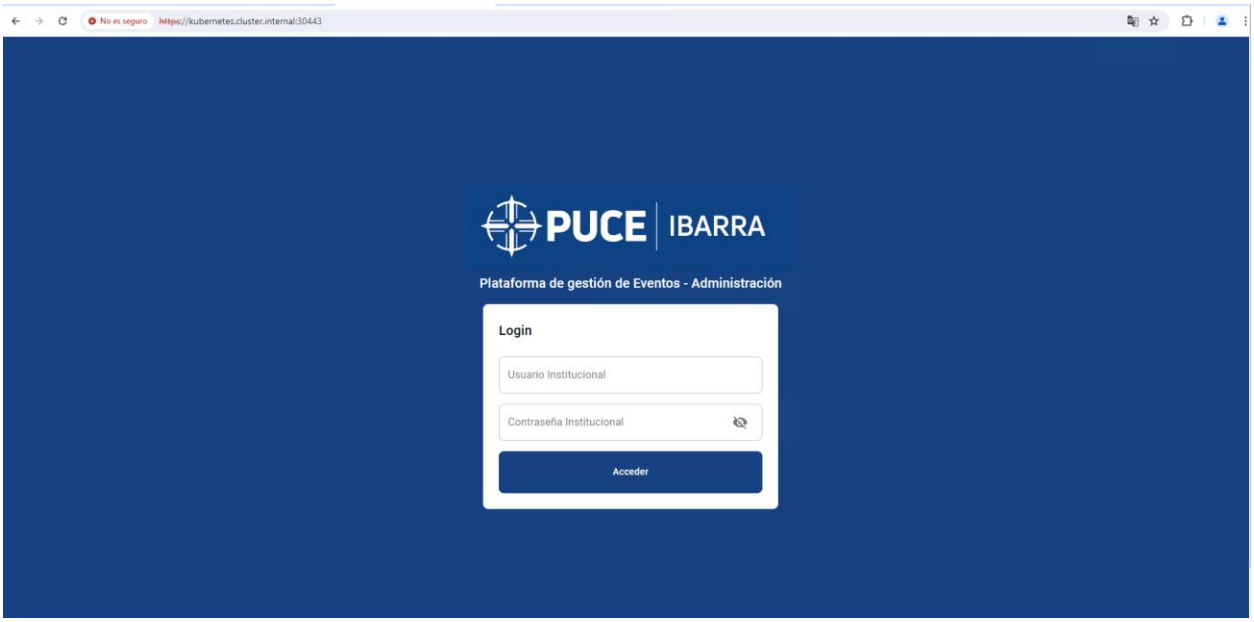
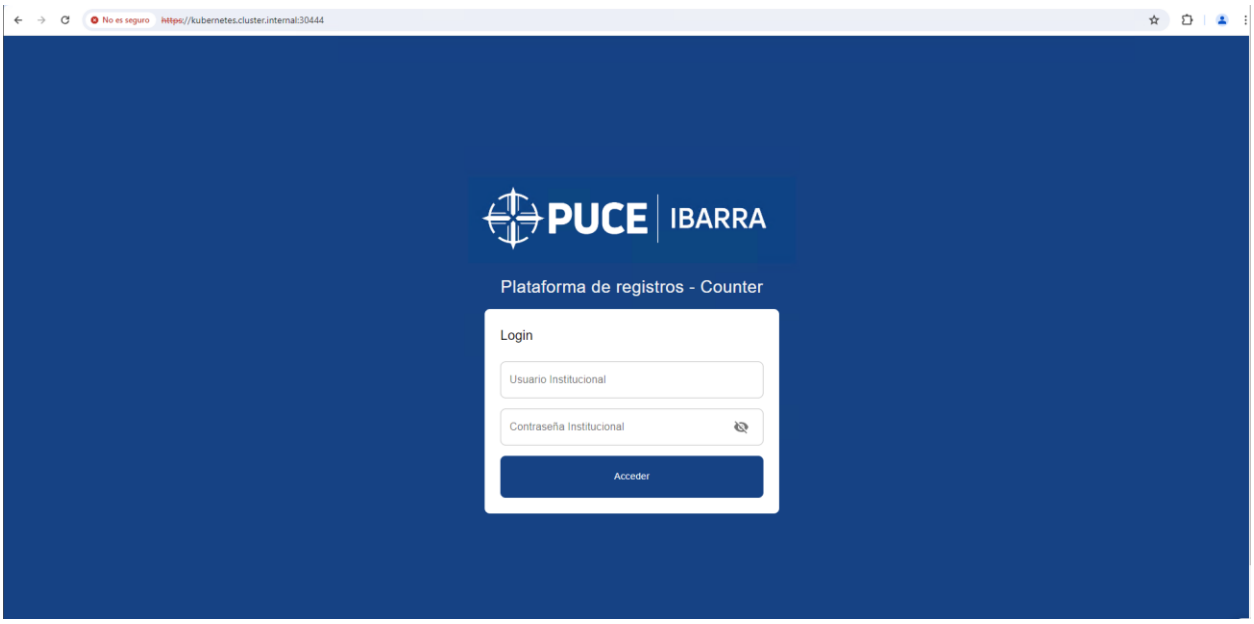


Figura 76
Pantalla de Counter de la Aplicación



ANEXO VII

Figura 77
Comando para Escalar Horizontalmente la Aplicación

```
master@master:~/recursosk8$ kubectl scale --current-replicas=3 --replicas=5 deployment/frontend-eventos-counter
deployment.apps/frontend-eventos-counter scaled
```

Figura 78
Pods Creados al Escalar Horizontalmente la Aplicación

Pods

Nombre	Imágenes	Etiquetas	Nodo	Estado	Reinicios	Utilización de CPU (núcleos)	Utilización de memoria (octetos)	Fecha de creación
frontend-eventos-counter-84459cd59f-j7blh	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	worker1	Running	0	0,00m	4,79Mi	a minute ago
frontend-eventos-counter-84459cd59f-m8lk5	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	worker2	Running	0	0,00m	4,98Mi	a minute ago
backend-eventos-gestion-67859566fd-sr2zb	dasanchez1/backend-eventos-gestion:v1	app: backend-eventos-gestion pod-template-hash: 67859566fd	worker2	Running	0	1,00m	49,27Mi	14 minutes ago
frontend-eventos-admin-79d459db5f-2m6td	dasanchez1/frontend-eventos-admin:v1	app: frontend-eventos-admin pod-template-hash: 79d459db5f	worker1	Running	0	0,00m	5,41Mi	14 minutes ago
frontend-eventos-admin-79d459db5f-lffvc	dasanchez1/frontend-eventos-admin:v1	app: frontend-eventos-admin pod-template-hash: 79d459db5f	master	Running	0	0,00m	5,04Mi	14 minutes ago
frontend-eventos-admin-79d459db5f-zjpwj	dasanchez1/frontend-eventos-admin:v1	app: frontend-eventos-admin pod-template-hash: 79d459db5f	worker2	Running	0	0,00m	4,76Mi	14 minutes ago
frontend-eventos-counter-84459cd59f-l46d8	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	worker1	Running	0	0,00m	4,78Mi	14 minutes ago
frontend-eventos-counter-84459cd59f-vhf22	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	worker2	Running	0	0,00m	4,93Mi	14 minutes ago
frontend-eventos-counter-84459cd59f-xz9mn	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	master	Running	0	0,00m	4,92Mi	14 minutes ago

ANEXO VIII

Figura 79
Pantalla de Login del Dashboard (UI)

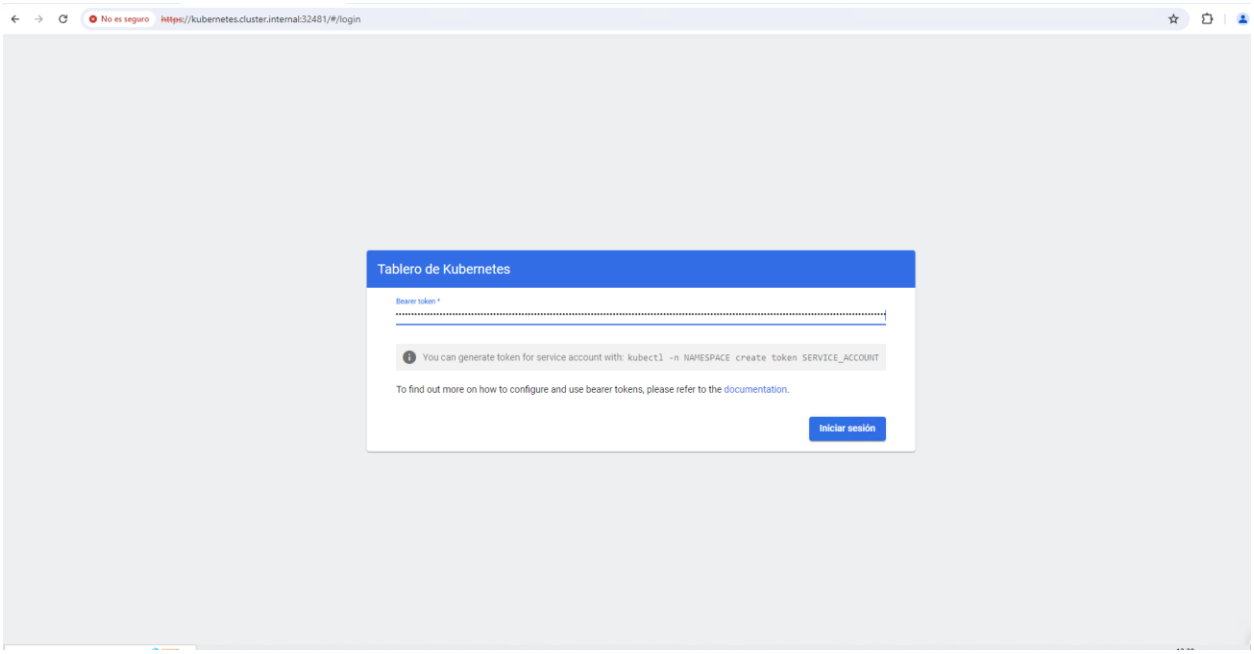


Figura 80
Despliegue desde la Interfaz Gráfica

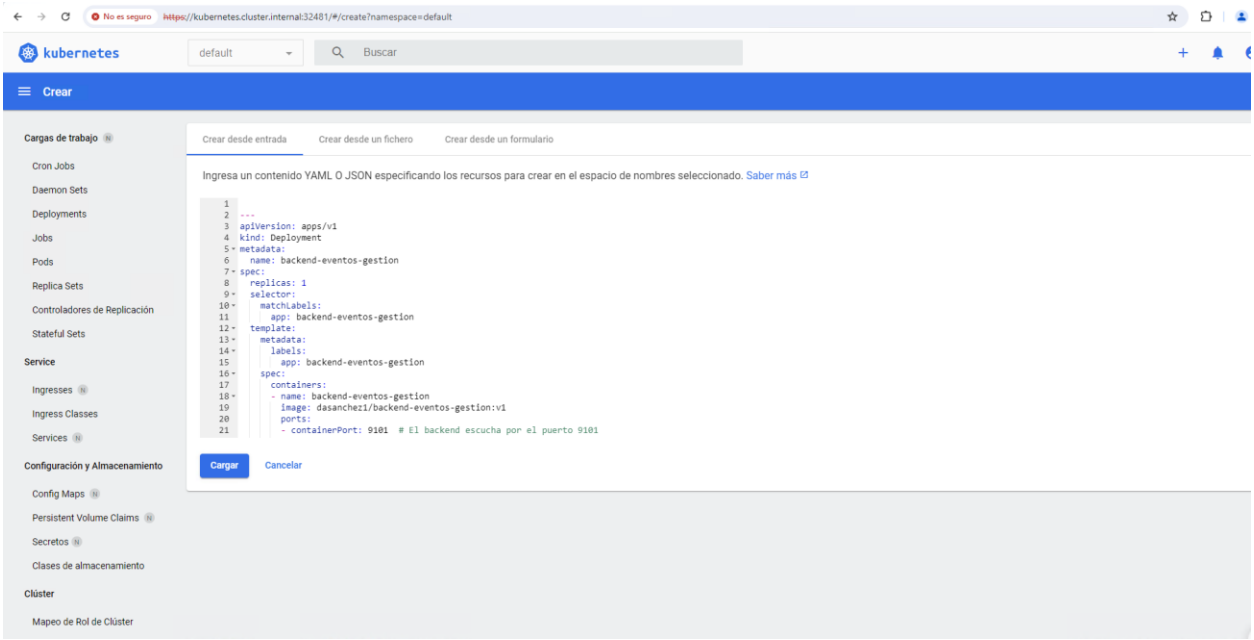


Figura 81
Creación de Recursos desde la Interfaz Gráfica

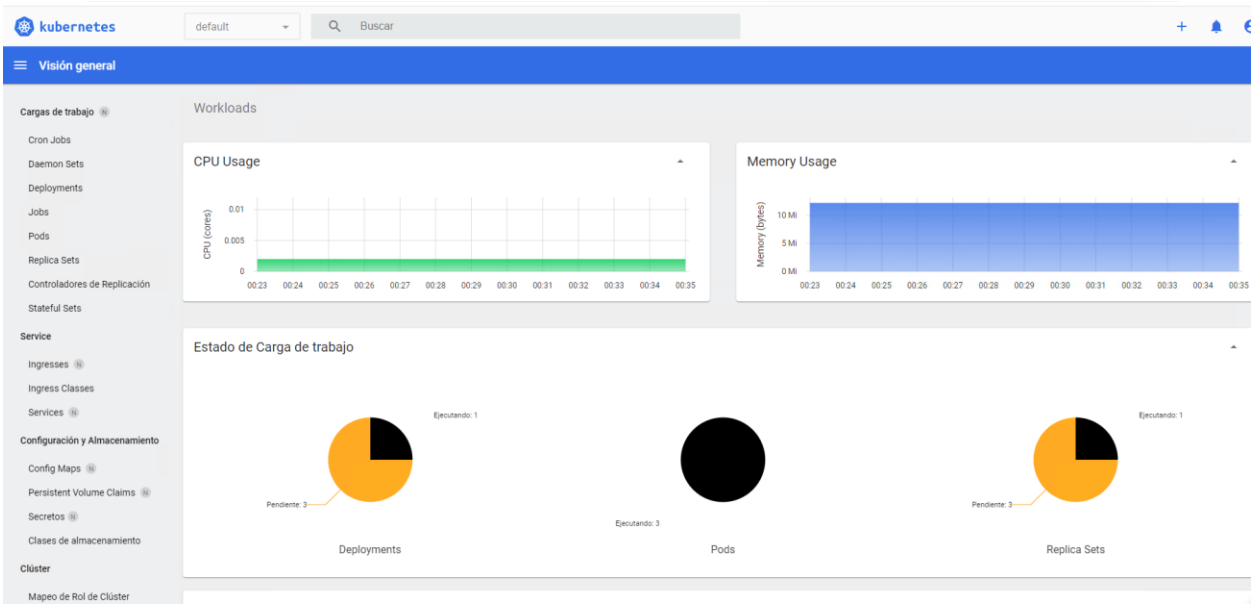


Figura 82
Pods Funcionando Correctamente

Pods

Nombre	Imágenes	Etiquetas	Nodo	Estado	Reinicios	Utilización de CPU (núcleos)	Utilización de memoria (octetos)	Fecha de creación
backend-eventos-gestion-67859566fd-rvrt4	dasanchez1/backend-eventos-gestion:v1	app: backend-eventos-gestion pod-template-hash: 67859566fd	worker2	Running	0	-	-	33 seconds ago
frontend-eventos-admin-79d459db5f-2xq52	dasanchez1/frontend-eventos-admin:v1	app: frontend-eventos-admin pod-template-hash: 79d459db5f	worker2	Running	0	-	-	33 seconds ago
frontend-eventos-admin-79d459db5f-f4g68	dasanchez1/frontend-eventos-admin:v1	app: frontend-eventos-admin pod-template-hash: 79d459db5f	worker1	Running	0	-	-	33 seconds ago
frontend-eventos-admin-79d459db5f-wn8tc	dasanchez1/frontend-eventos-admin:v1	app: frontend-eventos-admin pod-template-hash: 79d459db5f	master	Running	0	-	-	33 seconds ago
frontend-eventos-counter-84459cd59f-c8lls	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	worker2	Running	0	-	-	33 seconds ago
frontend-eventos-counter-84459cd59f-qckr5	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	master	Running	0	-	-	33 seconds ago
frontend-eventos-counter-84459cd59f-tnv56	dasanchez1/frontend-eventos-counter:v1	app: frontend-eventos-counter pod-template-hash: 84459cd59f	worker1	Running	0	-	-	33 seconds ago

ANEXO IX

Figura 83

Pods antes de Apagar el Nodo Worker2

```
master@master:~/recursosk8$ kubectl get pods -o wide
NAME                                      READY   STATUS    RESTARTS   AGE     IP               NODE      NOMINATED NODE
backend-eventos-gestion-67859566fd-rvrt4  1/1     Running   0          4m18s   172.24.189.120   worker2   <none>
frontend-eventos-admin-79d459db5f-2xq52   1/1     Running   0          4m18s   172.24.189.121   worker2   <none>
frontend-eventos-admin-79d459db5f-f4g68   1/1     Running   0          4m18s   172.24.235.182   worker1   <none>
frontend-eventos-admin-79d459db5f-wn8tc   1/1     Running   0          4m18s   172.24.219.117   master    <none>
frontend-eventos-counter-84459cd59f-c8lls 1/1     Running   0          4m18s   172.24.189.122   worker2   <none>
frontend-eventos-counter-84459cd59f-qckr5 1/1     Running   0          4m18s   172.24.219.116   master    <none>
frontend-eventos-counter-84459cd59f-tnv56 1/1     Running   0          4m18s   172.24.235.183   worker1   <none>
hello-658fd655f-5srb5                     1/1     Running   0          5d4h    172.24.219.79    master    <none>
hello-658fd655f-814xc                     1/1     Running   0          5d4h    172.24.189.77    worker2   <none>
hello-658fd655f-zps6d                     1/1     Running   0          5d4h    172.24.235.144   worker1   <none>
master@master:~/recursosk8$
```

Figura 84

Terminando los Pods del Nodo Worker2

```
master@master:~/recursosk8$ kubectl get pods -o wide
NAME                                      READY   STATUS        RESTARTS   AGE     IP               NODE      NOMINATED NODE
backend-eventos-gestion-67859566fd-rvrt4  1/1     Terminating   0          17m     172.24.189.120   worker2   <none>
backend-eventos-gestion-67859566fd-v78xm  1/1     Running       0          4m55s   172.24.235.184   worker1   <none>
frontend-eventos-admin-79d459db5f-2xq52   1/1     Terminating   0          17m     172.24.189.121   worker2   <none>
frontend-eventos-admin-79d459db5f-bv4ph   1/1     Running       0          4m55s   172.24.235.186   worker1   <none>
frontend-eventos-admin-79d459db5f-f4g68   1/1     Running       0          17m     172.24.235.182   worker1   <none>
frontend-eventos-admin-79d459db5f-wn8tc   1/1     Running       0          17m     172.24.219.117   master    <none>
frontend-eventos-counter-84459cd59f-c8lls 1/1     Terminating   0          17m     172.24.189.122   worker2   <none>
frontend-eventos-counter-84459cd59f-qckr5 1/1     Running       0          17m     172.24.219.116   master    <none>
frontend-eventos-counter-84459cd59f-tnv56 1/1     Running       0          17m     172.24.235.183   worker1   <none>
frontend-eventos-counter-84459cd59f-v29tn 1/1     Running       0          4m55s   172.24.235.185   worker1   <none>
hello-658fd655f-5srb5                     1/1     Running       0          5d5h    172.24.219.79    master    <none>
hello-658fd655f-814xc                     1/1     Terminating   0          5d5h    172.24.189.77    worker2   <none>
hello-658fd655f-c7k2p                     1/1     Running       0          4m55s   172.24.235.188   worker1   <none>
hello-658fd655f-zps6d                     1/1     Running       0          5d5h    172.24.235.144   worker1   <none>
master@master:~/recursosk8$
```

Figura 85

Pods Creados el Nodo Worker1

```
master@master:~/recursosk8$ kubectl get pods -o wide
NAME                                      READY   STATUS    RESTARTS   AGE     IP               NODE      NOMINATED NODE
backend-eventos-gestion-67859566fd-v78xm  1/1     Running   0          13m     172.24.235.184   worker1   <none>
frontend-eventos-admin-79d459db5f-bv4ph   1/1     Running   0          13m     172.24.235.186   worker1   <none>
frontend-eventos-admin-79d459db5f-f4g68   1/1     Running   0          26m     172.24.235.182   worker1   <none>
frontend-eventos-admin-79d459db5f-wn8tc   1/1     Running   0          26m     172.24.219.117   master    <none>
frontend-eventos-counter-84459cd59f-qckr5 1/1     Running   0          26m     172.24.219.116   master    <none>
frontend-eventos-counter-84459cd59f-tnv56 1/1     Running   0          26m     172.24.235.183   worker1   <none>
frontend-eventos-counter-84459cd59f-v29tn 1/1     Running   0          13m     172.24.235.185   worker1   <none>
hello-658fd655f-5srb5                     1/1     Running   0          5d5h    172.24.219.79    master    <none>
hello-658fd655f-c7k2p                     1/1     Running   0          13m     172.24.235.188   worker1   <none>
hello-658fd655f-zps6d                     1/1     Running   0          5d5h    172.24.235.144   worker1   <none>
master@master:~/recursosk8$ kubectl get pods -o
```

Figura 86

Estado "NotReady" en el Worker2

```
NAME      STATUS     ROLES                  AGE     VERSION
master    Ready      control-plane,master   6d22h   v1.29.5
worker1   Ready      <none>                 6d21h   v1.29.5
worker2   NotReady   <none>                 6d21h   v1.29.5
master@master:~/recursosk8$
```

Figura 87
Terminando los Pods del Nodo Worker1

```
master@master:~/recursosk8$ kubectl get pods -o wide
NAME                                    READY   STATUS        RESTARTS   AGE    IP              NODE      NOMINATED NODE
backend-eventos-gestion-67859566fd-592kf    1/1     Running       0          11s    172.24.189.127   worker2   <none>
backend-eventos-gestion-67859566fd-v78xm    1/1     Terminating   0          22m    172.24.235.184   worker1   <none>
frontend-eventos-admin-79d459db5f-45746     1/1     Running       0          11s    172.24.189.72    worker2   <none>
frontend-eventos-admin-79d459db5f-62f42     1/1     Running       0          11s    172.24.189.125   worker2   <none>
frontend-eventos-admin-79d459db5f-bv4ph     1/1     Terminating   0          22m    172.24.235.186   worker1   <none>
frontend-eventos-admin-79d459db5f-f4g68     1/1     Terminating   0          35m    172.24.235.182   worker1   <none>
frontend-eventos-admin-79d459db5f-wn8tc     1/1     Running       0          35m    172.24.219.117   master    <none>
frontend-eventos-counter-84459cd59f-c8hc2   1/1     Running       0          11s    172.24.189.75    worker2   <none>
frontend-eventos-counter-84459cd59f-ggtxh   1/1     Running       0          11s    172.24.189.126   worker2   <none>
frontend-eventos-counter-84459cd59f-qckr5   1/1     Running       0          35m    172.24.219.116   master    <none>
frontend-eventos-counter-84459cd59f-tnv56   1/1     Terminating   0          35m    172.24.235.183   worker1   <none>
frontend-eventos-counter-84459cd59f-v29tn   1/1     Terminating   0          22m    172.24.235.185   worker1   <none>
hello-658fd655f-5srb5                       1/1     Running       0          5d5h   172.24.219.79    master    <none>
hello-658fd655f-b4fh8                       1/1     Running       0          11s    172.24.189.73    worker2   <none>
hello-658fd655f-c7k2p                       1/1     Terminating   0          22m    172.24.235.188   worker1   <none>
hello-658fd655f-z7ntt                       1/1     Running       0          11s    172.24.189.66    worker2   <none>
hello-658fd655f-zps6d                       1/1     Terminating   0          5d5h   172.24.235.144   worker1   <none>
master@master:~/recursosk8$
```

Figura 88
Estado "NotReady" en el Worker1

```
master@master:~/recursosk8$ kubectl get nodes
NAME       STATUS     ROLES                  AGE     VERSION
master     Ready      control-plane,master   6d22h   v1.29.5
worker1    NotReady   <none>                 6d21h   v1.29.5
worker2    Ready      <none>                 6d21h   v1.29.5
master@master:~/recursosk8$
```

Figura 89
Vista de los Pods Terminando su Ejecución desde la Interfaz Gráfica

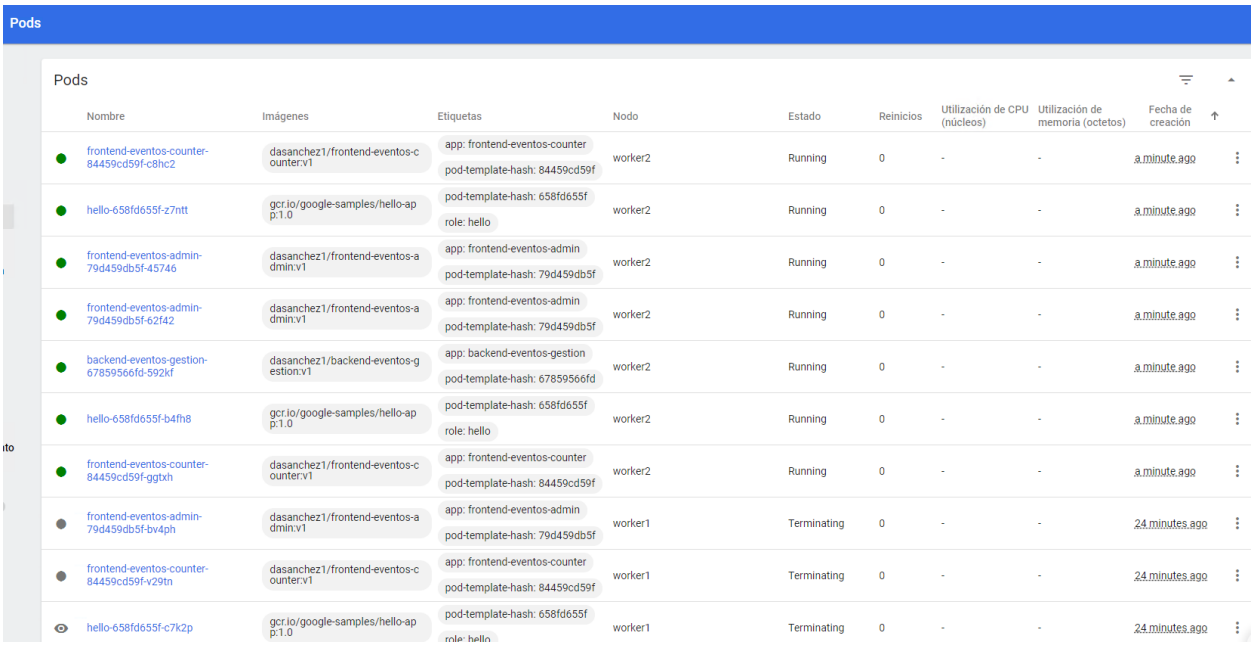
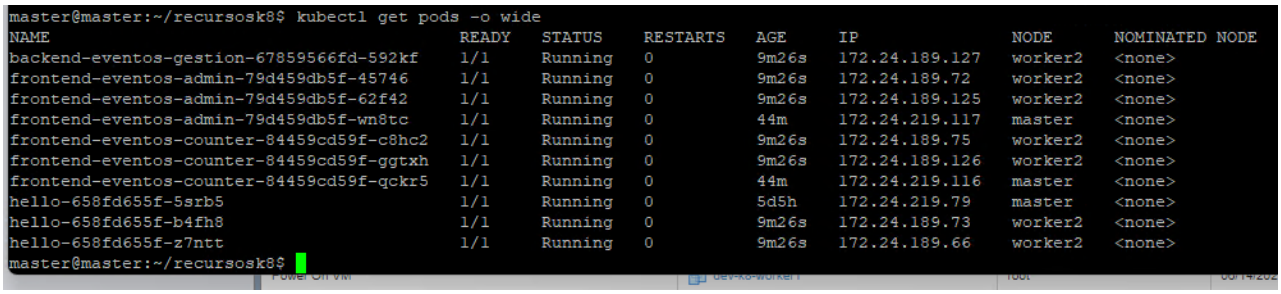


Figura 90

Pods Creados el Nodo Worker2



ANEXO X

Pontificia Universidad Católica del Ecuador
Seréis mis testigos

IBARRA

UNIDAD DE TECNOLOGÍAS DE LA INFORMACIÓN

Ibarra a, 04/07/2024

Carta de Entrega del trabajo de titulación en PUCE-Ibarra

Estimada Diana Abigail Sánchez,

En nombre de la Unidad de Tecnologías de la Información de la PUCE-IBARRA, me complace informarle que hemos recibido y revisado el proyecto de titulación: "Infraestructura Kubernetes para implementación de aplicaciones en contenedores en entorno de pruebas en la Unidad de Tecnologías de la Información de la PUCE-IBARRA" que ha desarrollado en nuestra unidad.

Queremos expresar nuestra satisfacción con el trabajo realizado, demostrando ser una contribución valiosa para nuestra unidad, proporcionando una solución moderna y eficiente para el despliegue de aplicaciones en contenedores. La implementación de Kubernetes en nuestro entorno de pruebas nos permitirá mejorar significativamente nuestra capacidad de desarrollo y despliegue de aplicaciones.

Agradecemos su dedicación y profesionalismo demostrado a lo largo del desarrollo de este proyecto. Estamos seguros de que los conocimientos y la experiencia que ha adquirido serán de gran valor para su carrera profesional.

Atentamente,

Msc. Franklin Sánchez
Jefe de la Unidad de Tecnologías de la Información - PUCE IBARRA

Dirección: Av. Jorge Guzmán Rueda y Av. Aurelio Espinosa Pólit. Ciudadela "La Victoria".
Teléf: (593-6) 2615 500 / 2615 453 **Ext.** 1000 **Cel.** 099 236 27 13 / 098 138 3498
Ibarra - Ecuador / www.pucesi.edu.ec