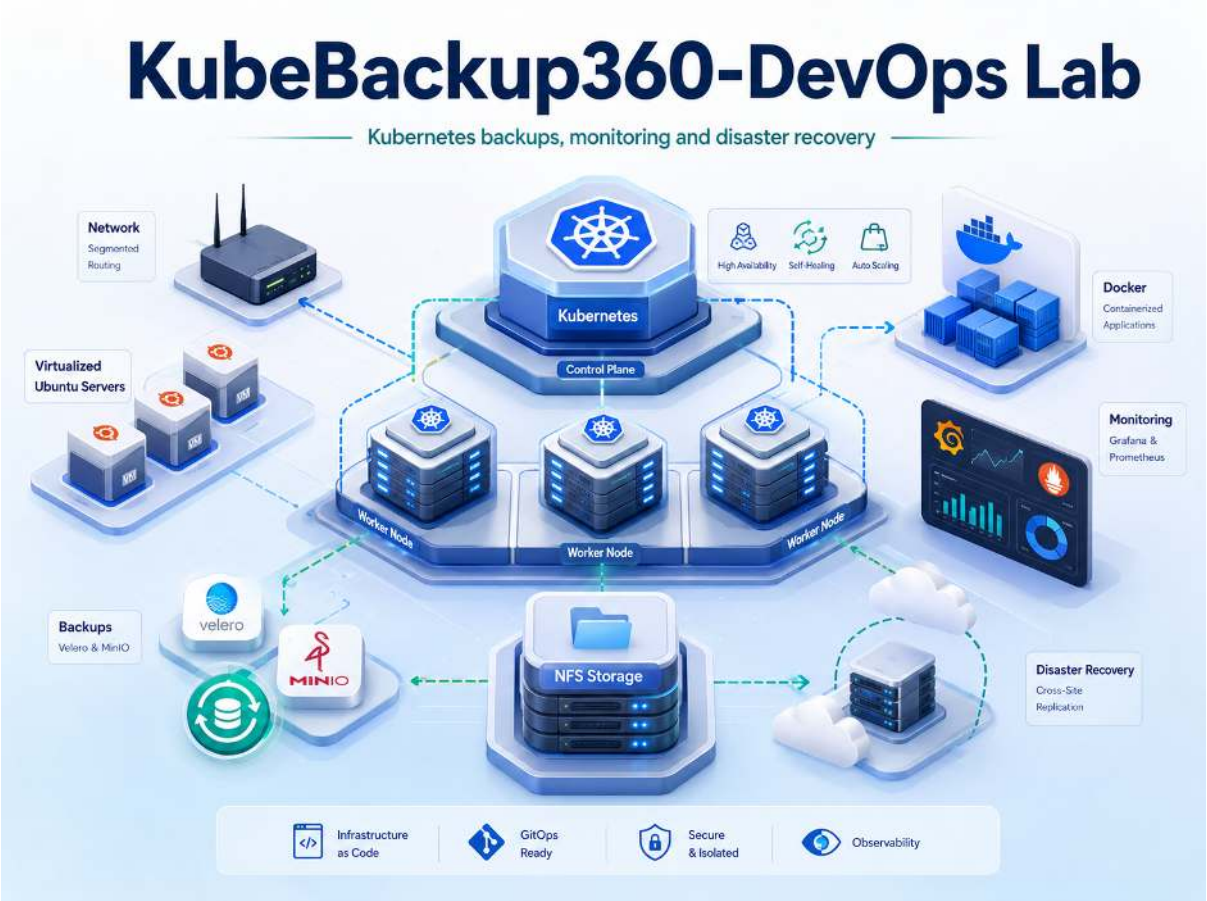
INSTITUT PUIG CASTELLAR

KUBEBACKUP360

Sistema de copias de seguridad y recuperación para Kubernetes



Proyecto de Desarrollo e Investigación

CFGS Administració de Sistemes Informàtics i Xarxes

Centro : INSTITUT EL PUIG CASTELLAR

Autores

-Luis Alfredo Florez
-Clara Ines Nchama

Grup :ASIX 2

RESUMEN DEL PROYECTO

KubeBackup360 es un proyecto orientado al diseño e implementación de una infraestructura basada en Kubernetes capaz de proporcionar alta disponibilidad, persistencia de datos, copias de seguridad y monitorización avanzada en un entorno virtualizado.

El proyecto ha sido desarrollado utilizando máquinas virtuales Ubuntu Server sobre VirtualBox, desplegando un clúster Kubernetes mediante RKE2, una distribución segura y simplificada de Kubernetes. Además, se han integrado servicios de red como DHCP y DNS, almacenamiento persistente mediante NFS, copias de seguridad utilizando Velero y monitorización mediante Prometheus y Grafana.

El objetivo principal del proyecto es simular una infraestructura empresarial real basada en tecnologías cloud-native, permitiendo validar mecanismos de tolerancia a fallos, recuperación ante desastres y administración centralizada de contenedores.

Durante el desarrollo se realizaron pruebas funcionales, pruebas de alta disponibilidad y validaciones de recuperación automática de servicios ante fallos de nodos.

Como resultado final, se obtuvo un entorno completamente funcional, escalable y preparado para futuras ampliaciones.

PALABRAS CLAVE

Kubernetes, Docker, RKE2, Velero, NFS, Prometheus, Grafana, Alta disponibilidad

ABSTRACT

KubeBackup360 is a project focused on the design and implementation of a Kubernetes-based infrastructure capable of providing high availability, data persistence, backup management and advanced monitoring within a virtualized environment.

The project was developed using Ubuntu Server virtual machines on VirtualBox, deploying a Kubernetes cluster through RKE2, a secure and simplified Kubernetes distribution. In addition, network services such as DHCP and DNS, persistent storage through NFS, backup management using Velero, and monitoring with Prometheus and Grafana were integrated.

The main objective of the project is to simulate a real enterprise cloud-native infrastructure, validating fault tolerance mechanisms, disaster recovery strategies and centralized container orchestration.

Several functional tests, high availability validations and failover recovery scenarios were performed during development.

The final result is a scalable and fully operational infrastructure prepared for future improvements.

KEYWORDS

Kubernetes, Docker, RKE2, Backup, NFS, Prometheus, Grafana, High Availability

ÍNDICE

1. INTRODUCCIÓN

En los últimos años, las infraestructuras basadas en contenedores han experimentado un crecimiento exponencial dentro de los entornos empresariales y arquitecturas *cloud-native*. La necesidad de desplegar aplicaciones de forma escalable, resiliente y automatizada ha consolidado a **Kubernetes** como el estándar de facto para la orquestación de contenedores en el sector tecnológico.

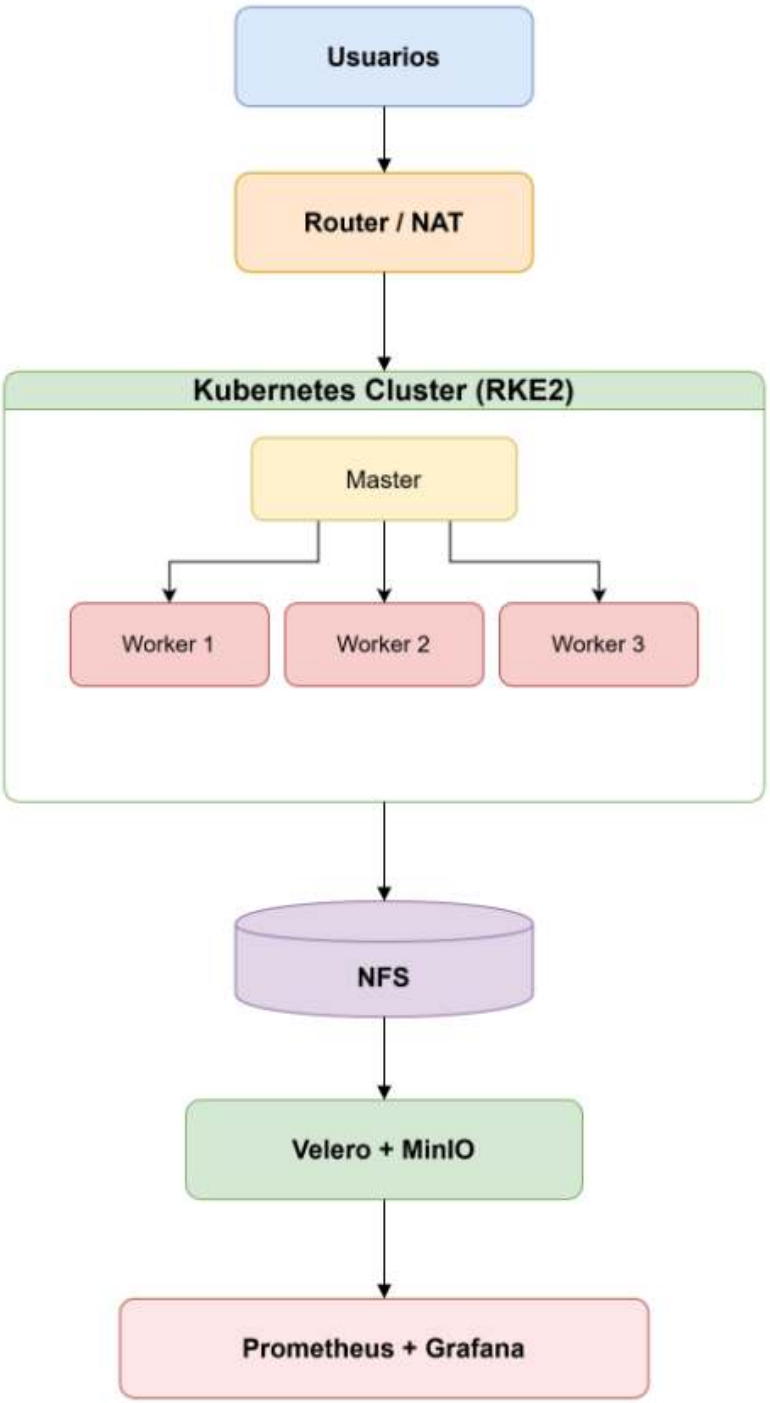
Sin embargo, la adopción y administración de entornos Kubernetes en producción implica afrontar retos críticos para cualquier organización:

- **Persistencia de datos:** Garantizar que la información no se pierda al destruir o recrear contenedores.
- **Tolerancia a fallos:** Mantener los servicios en línea ante la caída de nodos o infraestructura.
- **Observabilidad:** Monitorizar en tiempo real el estado de salud del sistema para anticipar problemas.
- **Recuperación ante desastres:** Disponer de mecanismos eficaces de respaldo y restauración rápida.

Para dar respuesta a estas necesidades clave, surge **KubeBackup360**, un proyecto enfocado en el diseño, despliegue y validación de una infraestructura Kubernetes completamente funcional sobre un entorno virtualizado. Esta solución integra de forma centralizada tecnologías de almacenamiento persistente, políticas de backup automatizadas

y herramientas avanzadas de observabilidad, simulando con precisión la arquitectura y los niveles de seguridad requeridos en entornos corporativos reales.

El propósito fundamental de **KubeBackup360** es certificar que una pequeña o mediana infraestructura puede alcanzar altos estándares de disponibilidad y resiliencia informática, garantizando la continuidad del negocio y optimizando la administración operativa de los servicios de IT.



1.1 Contexto

En el panorama tecnológico actual, las organizaciones se encuentran en un proceso de migración progresiva desde infraestructuras tradicionales (basadas en servidores físicos o máquinas virtuales monolíticas) hacia arquitecturas orientadas a **contenedores y microservicios**. Este cambio de paradigma no es casual; responde a la necesidad del mercado de desplegar aplicaciones de forma flexible, escalable y eficiente, optimizando al máximo el uso de los recursos de hardware y facilitando la automatización de los procesos de desarrollo y operaciones (*DevOps*).

Dentro de este ecosistema, **Kubernetes** se ha consolidado como la plataforma líder y el estándar de la industria para la orquestación de contenedores, permitiendo a empresas de cualquier envergadura gestionar aplicaciones distribuidas tanto en la nube (*cloud*) como en servidores locales (*on-premise*).

Sin embargo, el despliegue de un clúster de Kubernetes que sea verdaderamente apto para un entorno de producción plantea retos operacionales complejos. No basta con ejecutar contenedores; es imperativo resolver aspectos críticos como la persistencia de datos en entornos volátiles, el almacenamiento distribuido, la resiliencia ante la caída de nodos, la monitorización del rendimiento y la automatización de los respaldos. El proyecto **KubeBackup360** nace precisamente con el propósito de dar respuesta a estos retos, reproduciendo una infraestructura empresarial real dentro de un entorno virtualizado de laboratorio.

1.2 Justificación

La realización de este proyecto se justifica por la necesidad de diseñar soluciones técnicas alineadas con las demandas actuales del sector de la administración de sistemas y las tecnologías *cloud-native*. Para un perfil profesional enfocado en la gestión de sistemas en red, ya no basta con administrar sistemas operativos tradicionales; es fundamental dominar el aprovisionamiento de infraestructuras modernas, automatizadas y resilientes.

KubeBackup360 se presenta como una solución integral que demuestra la viabilidad de implementar entornos contenerizados seguros y de alta disponibilidad sin depender necesariamente de costosos servicios en la nube pública. El proyecto aporta un valor práctico diferencial al unificar en un solo entorno de trabajo disciplinas clave como:

- La administración avanzada de clústeres y la orquestación de carga de trabajo.
- La gestión de almacenamiento persistente y distribuido para garantizar la integridad de la información.
- La continuidad del negocio mediante políticas de automatización de copias de seguridad y planes de recuperación ante desastres (*Disaster Recovery*).
- La observabilidad proactiva del sistema para garantizar acuerdos de nivel de servicio (SLA) óptimos.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar, implementar y validar una infraestructura basada en Kubernetes que sea resiliente, segura y de alta disponibilidad, capaz de garantizar la persistencia de los datos, la automatización de los procesos de *backup* y la monitorización avanzada de todos sus componentes, simulando con precisión los estándares de un entorno empresarial real.

1.3.2 Objetivos específicos

Para alcanzar el objetivo general, el proyecto se divide en las siguientes metas funcionales y operativas:

- **Segmentación y seguridad de red:** Diseñar una topología de red virtualizada y segmentada de forma independiente, garantizando el aislamiento del tráfico de gestión, almacenamiento y datos de los servicios.
- **Autonomía de servicios de infraestructura:** Implementar servicios internos de resolución de nombres y asignación dinámica de direccionamiento IP para asegurar la correcta comunicación y localización de los componentes del clúster.
- **Orquestación de contenedores de última generación:** Desplegar un motor de orquestación de contenedores robusto, seguro y optimizado para entornos gubernamentales o empresariales (*security-focused*).
- **Abstracción del almacenamiento persistente:** Configurar un sistema de almacenamiento centralizado y compartido en red que permita a las aplicaciones contenerizadas preservar su información de manera independiente al nodo donde se ejecuten.
- **Gestión del ciclo de vida del dato (Backups):** Implementar una solución automatizada de copias de seguridad y replicación en un almacenamiento de objetos que permita salvaguardar tanto el estado del clúster como los datos de los usuarios.
- **Validación de resiliencia (Alta disponibilidad):** Someter a la infraestructura a pruebas de estrés y caída de nodos para validar la capacidad del sistema de auto-repararse y balancear el tráfico sin interrupción del servicio.
- **Despliegue y entrega de servicios:** Validar la infraestructura mediante el despliegue funcional de aplicaciones contenerizadas de ejemplo, demostrando la viabilidad del clúster para producción.
- **Auditoría y observabilidad analítica:** Establecer una plataforma de monitorización centralizada que recolecta métricas de rendimiento en tiempo real y ofrezca cuadros de mando visuales para facilitar la toma de decisiones operativas.
- **Planes de contingencia activos:** Simular escenarios reales de desastre (pérdida total de datos o corrupción del clúster) para certificar la efectividad de los protocolos de recuperación rápida y continuidad del negocio.

1.4 Metodología de trabajo y planificación del proyecto

Para garantizar el éxito, la estabilidad y la calidad del proyecto **KubeBackup360**, se ha adoptado una **metodología de desarrollo incremental basada en validaciones continuas**. Este enfoque metodológico exige que cada bloque funcional de la

infraestructura sea desplegado, testeado y verificado de manera individual antes de proceder con la siguiente etapa. Gracias a esto, es posible detectar errores de configuración de forma temprana, aislar fallos de interconexión y asegurar la robustez global del entorno.

La estrategia de implementación se estructuró en siete fases progresivas, donde cada una de ellas habilitó los cimientos de la siguiente:

1. **Fase I: Preparación del entorno virtualizado (Aprovisionamiento):** Dimensionamiento y despliegue de las máquinas virtuales que actuarán como nodos de cómputo y servidores de soporte, garantizando que cuentan con los recursos de hardware necesarios.
2. **Fase II: Configuración de red y servicios base (Conectividad):** Segmentación del tráfico de datos y despliegue de los servicios esenciales de resolución de nombres y asignación de direccionamiento IP para la infraestructura.
3. **Fase III: Despliegue del clúster de orquestación (Cómputo):** Inicialización y ensamblado del motor de Kubernetes, estableciendo la comunicación y el plano de control entre los nodos maestros y los de ejecución.
4. **Fase IV: Persistencia y almacenamiento centralizado (Datos):** Implementación de la capa de abstracción de almacenamiento que permite a los servicios guardar información de manera permanente y compartida en la red.
5. **Fase V: Automatización de backups y políticas de recuperación (Resiliencia):** Configuración de las herramientas encargadas de la captura de copias de seguridad del estado del clúster y de los datos, así como la habilitación del repositorio de objetos destino.
6. **Fase VI: Observabilidad y monitorización (Auditoría):** Despliegue de los agentes recolectores de métricas y los paneles visuales para el control del rendimiento del sistema en tiempo real.
7. **Fase VII: Despliegue de aplicaciones y validación ante fallos (Certificación):** Ejecución de servicios reales sobre el clúster y ejecución de pruebas de estrés y desconexión forzada de elementos para auditar la tolerancia a fallos del sistema.

1.5 Herramientas de seguimiento y control documental

Con el fin de asegurar el rigor técnico, la trazabilidad del proyecto y la posibilidad de replicar la infraestructura en el futuro, el proceso de desarrollo se apoyó en los siguientes mecanismos de gestión:

- **Repositorio centralizado de código:** Utilización de plataformas de control de versiones Git para el almacenamiento, gestión y control de cambios de los scripts de automatización y los manifiestos de configuración declarativa (`YAML`).
- **Cronograma operativo de tareas:** Planificación temporal de los hitos del proyecto para asegurar el cumplimiento de los plazos de entrega.
- **Protocolos de validación funcional:** Creación de baterías de pruebas específicas al final de cada fase para comprobar que los servicios respondían correctamente antes de avanzar.
- **Registro de evidencias:** Captura y documentación sistemática del comportamiento del sistema y de los resultados de las pruebas de contingencia para su posterior análisis.

1.6 Estudio económico y presupuestario

El desarrollo del proyecto **KubeBackup360** se ha diseñado bajo una filosofía de **coste directo cero**, optimizando el uso de recursos de hardware preexistentes y priorizando soluciones de código abierto (*Open Source*). Esta estrategia demuestra la viabilidad económica de desplegar entornos de alta disponibilidad empresarial en pequeñas y medianas organizaciones (PYMES) sin incurrir en dependencias de proveedores (*vendor lock-in*) ni en costes recurrentes de nubes públicas.

El presupuesto del proyecto se divide en tres bloques fundamentales:

- **Infraestructura de Hardware:** Se ha ejecutado íntegramente sobre infraestructura local, aprovechando dos estaciones de trabajo físicas ya disponibles (un equipo personal y un equipo de laboratorio). Al no requerir la adquisición de servidores dedicados ni el alquiler de instancias en la nube, el coste de hardware es de **0,00 €**.
- **Costes de Licenciamiento de Software:** Toda la pila tecnológica seleccionada (sistema operativo, motor de contenedores, orquestador, herramientas de copia de seguridad y suite de observabilidad) se distribuye bajo licencias libres (MIT, Apache 2.0, GPL, etc.), eliminando cualquier gasto de adquisición o activación.
- **Plataforma de Hipervisión:** El uso de virtualización de escritorio ha permitido simular una red de servidores distribuidos con total fidelidad arquitectónica sobre el hardware existente, evitando inversiones en electrónica de red o cabinas de almacenamiento físico.

A continuación, se detalla el desglose del coste del software y herramientas utilizadas:

Componente / RecursoTipo de LicenciaCoste Asociado**Hipervisor (VirtualBox)**Código Abierto / Gratuito0,00 €**Sistema Operativo (Ubuntu Server)**Open Source (GPL)0,00 €**Motor de Contenedores (Docker)**Open Source (Apache 2.0)0,00 €**Orquestador de Clúster (RKE2)**Open Source (Apache 2.0)0,00 €**Gestión de Backups (Velero / MinIO)**Open Source0,00 €**Motor de Métricas (Prometheus)**Open Source (Apache 2.0)0,00 €**Panel de Control Visual (Grafana)**Open Source (AGPLv3)0,00 €**Hardware de Cómputo (Equipos Locales)**Recurso Preexistente0,00 €**TOTAL INVERSIÓN SOFTWARE Y HARDWARE—0,00 €**

2. DESCRIPCIÓN DEL PROYECTO

Este capítulo define las capacidades operativas de **KubeBackup360**, especificando tanto las acciones que el sistema es capaz de ejecutar (requisitos funcionales) como las

cualidades y restricciones bajo las que debe operar la infraestructura (requisitos no funcionales).

2.1 Requisitos del sistema

2.1.1 Requisitos funcionales (RF)

Los requisitos funcionales detallan las capacidades e interacciones específicas que ofrece la infraestructura informática implementada:

- **RF-01: Orquestación de cargas de trabajo:** El sistema debe ser capaz de inicializar, coordinar y mantener un clúster multinodo totalmente operativo para la ejecución de servicios automatizados.
- **RF-02: Gestión del ciclo de vida de aplicaciones:** La plataforma debe permitir el despliegue, actualización, parada y eliminación de aplicaciones empaquetadas en formato de contenedores aislados.
- **RF-03: Abstracción de datos (Persistencia):** El sistema debe proporcionar volúmenes de almacenamiento estables de modo que, si un contenedor se destruye o se reinicia, los datos de la aplicación permanezcan intactos y accesibles.
- **RF-04: Automatización de respaldos:** La infraestructura debe ejecutar políticas programadas de copias de seguridad que salvaguarden tanto la configuración interna del clúster como los datos de los volúmenes persistentes de manera desatendida.
- **RF-05: Recuperación y resiliencia ante desastres:** El sistema debe proveer los mecanismos necesarios para restaurar un servicio o el clúster completo a un estado funcional previo a partir de los backups almacenados en caso de corrupción o pérdida de datos.
- **RF-06: Observabilidad y telemetría en tiempo real:** La plataforma debe recolectar, procesar y almacenar métricas de rendimiento (uso de CPU, memoria, red y almacenamiento) tanto de los nodos físicos/virtuales como de los contenedores en ejecución.
- **RF-07: Auto-reparación y conmutación por error (Failover):** El entorno debe detectar la caída de una aplicación o de un nodo de cómputo y redistribuir automáticamente la carga de trabajo hacia los recursos sanos sin interrumpir el servicio al usuario final.

2.1.2 Requisitos no funcionales (RNF)

Los requisitos no funcionales definen los estándares de calidad, rendimiento y arquitectura que condicionan el comportamiento global de la solución:

RNF-01: Estabilidad y Disponibilidad: La infraestructura debe garantizar un funcionamiento continuo y mitigar los puntos únicos de fallo (*SPOF*) mediante la redundancia de componentes críticos.
RNF-02: Escalabilidad horizontal y vertical: El diseño del sistema debe permitir la adhesión de nuevos nodos de cómputo (escalabilidad horizontal) o la asignación de más recursos de hardware (escalabilidad vertical) de manera ágil y sin necesidad de rediseñar la arquitectura.

RNF-03: Seguridad perimetral e interna: El acceso administrativo a la infraestructura debe estar restringido y los componentes del plano de control deben comunicarse a través de canales cifrados.
RNF-04: Aislamiento y segmentación de red: El tráfico de datos del sistema, el tráfico de almacenamiento y el tráfico de cara al cliente deben viajar por redes virtuales lógicamente separadas para evitar colisiones y vectores de ataque.
RNF-05: Simplicidad en la administración: La gestión diaria de la infraestructura debe centralizarse mediante interfaces de línea de comandos estandarizadas o paneles web de monitorización, reduciendo la carga operativa del equipo de sistemas.
RNF-06: Tolerancia a fallos de infraestructura: El sistema debe soportar la pérdida abrupta de un nodo de ejecución (*Worker Node*) garantizando que los servicios críticos sigan respondiendo de forma transparente para el cliente externo.

2.2 Tecnologías y justificación de la arquitectura

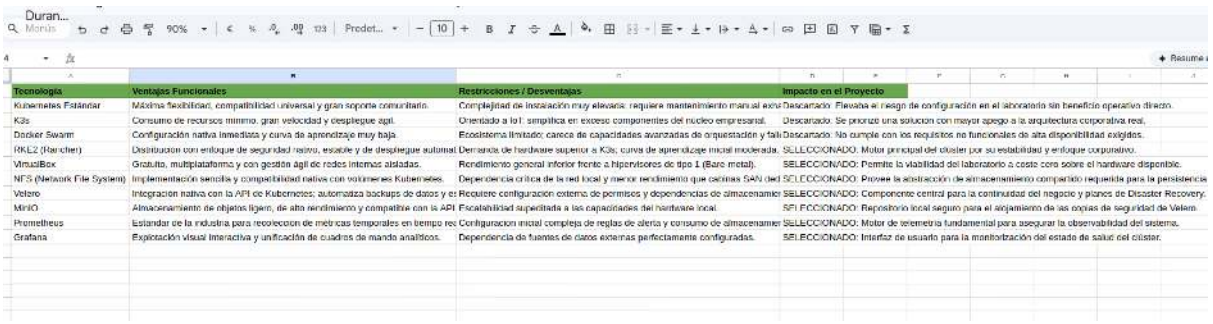
2.2.1 Tecnologías valoradas y matriz de decisión

Durante la fase inicial de ingeniería del proyecto, se evaluaron distintas alternativas tecnológicas dentro del ecosistema de virtualización y orquestación de contenedores. El objetivo fue hallar un equilibrio óptimo entre el realismo empresarial, la viabilidad económica y las restricciones físicas del entorno del laboratorio.

El principal dilema estratégico radicó en el motor de orquestación. Se analizaron cuatro opciones clave:

- **Kubernetes Estándar (*Upstream*):** Ofrece la máxima flexibilidad, pero su despliegue desde cero (*Vanilla K8s*) introduce una carga operativa de configuración manual excesiva que dilata los plazos sin aportar valor directo a la funcionalidad del negocio.
- **K3s:** Diseñado para entornos de recursos limitados (*Edge computing*), se descartó al prescindir por defecto de ciertas capas de seguridad de grado corporativo que el proyecto requería simular.
- **Docker Swarm:** Destaca por su extrema sencillez, pero se encuentra descolgado del estándar actual de la industria y carece del ecosistema nativo en observabilidad y automatización de respaldos que define a los entornos *cloud-native*.
- **RKE2 (*Rancher Government*) / Kubernetes Seleccionado:** Se escogió como la solución definitiva por combinar la total compatibilidad con el estándar de Kubernetes con un enfoque nativo en seguridad y un proceso de despliegue automatizado y estable.

A continuación, se detalla la **Matriz de Decisión Tecnológica** que justifica la composición final de la infraestructura:



Tecnología	Ventajas Funcionales	Restricciones / Desventajas	Impacto en el Proyecto
Kubernetes Estándar	Máxima flexibilidad, compatibilidad universal y gran soporte comunitario.	Complejidad de instalación muy elevada; requiere mantenimiento manual exha	Descartado. Elevaba el riesgo de configuración en el laboratorio sin beneficio operativo directo.
K3s	Consumo de recursos mínimo, gran velocidad y despliegue ágil.	Orientado a IoT; simplifica en exceso componentes del núcleo empresarial.	Descartado. Se prioriza una solución con mayor apego a la arquitectura corporativa real.
Docker Swarm	Configuración nativa inmediata y curva de aprendizaje muy baja.	Ecosistema limitado; carece de capacidades avanzadas de orquestación y falt	Descartado. No cumple con los requisitos no funcionales de alta disponibilidad exigidos.
RKE2 (Rancher)	Distribución con enfoque de seguridad nativo, estable y de despliegue automat	Demanda de hardware superior a K3s; curva de aprendizaje inicial moderada.	SELECCIONADO: Motor principal del clúster por su estabilidad y enfoque corporativo.
VirtualBox	Gratuito, multiplataforma y con gestión ágil de redes internas aisladas.	Rendimiento general inferior frente a hipervisores de tipo 1 (Bare metal).	SELECCIONADO: Permite la viabilidad del laboratorio a coste cero sobre el hardware disponible.
NFS (Network File System)	Implementación sencilla y compatibilidad nativa con volúmenes Kubernetes.	Dependencia crítica de la red local y menor rendimiento que cabinas SAN (bot	SELECCIONADO: Provee la abstracción de almacenamiento compartido requerida para la persistencia
Velero	Integración nativa con la API de Kubernetes; automatiza backups de datos y es	Requiere configuración externa de permisos y dependencias de almacenamier	SELECCIONADO: Componente central para la continuidad del negocio y planes de Disaster Recovery.
MinIO	Almacenamiento de objetos ligero, de alto rendimiento y compatible con la API	Escalabilidad supeditada a las capacidades del hardware local.	SELECCIONADO: Repositorio local seguro para el alojamiento de las copias de seguridad de Velero.
Prometheus	Estándar de la industria para recolección de métricas temporales en tiempo rea	Configuración inicial compleja de reglas de alerta y consumo de almacenamier	SELECCIONADO: Motor de telemetría fundamental para asegurar la observabilidad del sistema.
Grafana	Explotación visual interactiva y unificación de cuadros de mando analíticos.	Dependencia de fuentes de datos externas perfectamente configuradas.	SELECCIONADO: Interfaz de usuario para la monitorización del estado de salud del clúster.

Conclusión tecnológica

Tras analizar las diferentes alternativas, se optó por utilizar RKE2 como distribución principal de Kubernetes debido a su equilibrio entre seguridad, facilidad de despliegue y compatibilidad con entornos empresariales reales. Asimismo, se seleccionaron tecnologías open source ampliamente utilizadas dentro del ecosistema cloud-native, permitiendo construir una infraestructura completa sin necesidad de recurrir a soluciones propietarias o servicios cloud de pago.

2.2.2 Arquitectura tecnológica de la solución (Pila de tecnologías) — *Continuación*

- **Capa de Virtualización e Infraestructura Base:**

  -VirtualBox: Entorno de hipervisión encargado de aislar y ejecutar los nodos virtuales de forma eficiente sobre el hardware local.

  -Ubuntu Server 24.04 LTS: Sistema operativo base que garantiza estabilidad a largo plazo, soporte moderno de librerías y un entorno seguro para el nodo informático.

- **Capa de Servicios de Red Fundacionales:**

  -KEA DHCP: Responsable de la asignación dinámica, controlada y eficiente de direccionamiento IP dentro de los segmentos de red del laboratorio.

  -BIND9: Servidor DNS interno encargado de resolver la nomenclatura de la infraestructura y asegurar la interconexión de los servicios base.

- **Capa de Orquestación y Contenerización:**

  -Docker: Motor encargado de la ejecución y aislamiento de los contenedores que componen el ecosistema de aplicaciones.

  -Kubernetes / RKE2: Plataforma principal encargada de coordinar, balancear y auto-reparar las cargas de trabajo de la solución bajo estándares de seguridad corporativos.

- **Capa de Persistencia y Almacenamiento:**

-NFS (Network File System): Protocolo de almacenamiento en red que habilita los volúmenes persistentes compartidos necesarios para que las aplicaciones no pierdan información.

- **Capa de Gestión de Paquetes y Configuración Declarativa:**

  -Helm: Gestor de paquetes que estandariza, empaqueta y automatiza el despliegue de aplicaciones complejas dentro de Kubernetes.

  -YAML: Lenguaje de serialización utilizado para definir el estado deseado de la infraestructura mediante código (Infrastructure as Code).

- **Capa de Continuidad del Negocio y Protección de Datos:**

  -Velero: Solución experta encargada de la orquestación, programación y ejecución de copias de seguridad de los metadatos de Kubernetes y de los volúmenes persistentes, así como de su posterior restauración ante desastres.

  -MinIO: Servidor de almacenamiento de objetos de alto rendimiento compatible con la API de Amazon S3, utilizado como repositorio local seguro y centralizado para almacenar los respaldos generados por Velero.

- **Capa de Auditoría, Telemetría y Observabilidad:**

  -Prometheus: Motor de base de datos de series temporales y sistema de alerta encargado de recolectar de forma proactiva las métricas de rendimiento de los nodos, contenedores y servicios del clúster.

  -Grafana: Plataforma analítica y de visualización interactiva utilizada para modelar los datos de Prometheus en cuadros de mando (dashboards) intuitivos, facilitando la monitorización del estado de salud del sistema en tiempo real.

2.3 Estructura del proyecto

La infraestructura del proyecto está compuesta por múltiples máquinas virtuales interconectadas mediante redes segmentadas.

El nodo router centraliza los servicios de red y proporciona acceso a Internet.

El clúster Kubernetes está compuesto por un nodo master y varios nodos worker encargados de ejecutar contenedores.

El almacenamiento persistente se implementa mediante un servidor NFS y las copias de seguridad se gestionan utilizando Velero y MinIO.

Finalmente, Prometheus y Grafana permiten monitorizar el estado completo del sistema.

2.3 Estructura general de la infraestructura

La arquitectura del proyecto **KubeBackup360** está diseñada bajo un modelo modular y distribuido, compuesto por múltiples instancias virtuales especializadas que se interconectan a través de redes locales lógicamente segmentadas. Esta disposición estructural tiene como finalidad eliminar los puntos únicos de fallo funcionales y emular con precisión los esquemas de seguridad perimetral de un centro de datos empresarial.

A nivel funcional, la estructura se organiza en cuatro bloques operativos claramente definidos:

1. **Núcleo de Servicios de Red y Enrutamiento:** Centraliza el control de direccionamiento, la resolución de nombres y el aislamiento del tráfico, sirviendo además como pasarela segura hacia el exterior.
2. **Plano de Control y Computación (Clúster Kubernetes):** Dividido en un nodo de gestión (Master) encargado de la toma de decisiones y gobernanza del sistema, y varios nodos de ejecución (Workers) dedicados exclusivamente a procesar las aplicaciones contenerizadas.
3. **Capa de Almacenamiento Centralizado y Resiliencia:** Un servidor dedicado a servir espacio en red para garantizar que los datos de las aplicaciones sobrevivan a los reinicios, trabajando en sintonía con las herramientas de salvaguarda y almacenamiento de objetos para la protección ante desastres.
4. **Capa de Supervisión y Telemetría:** Un sistema transversal que audita constantemente el consumo y salud de cada uno de los bloques anteriores para asegurar la continuidad operativa.

2.4 Descripción analítica de los componentes

A continuación, se describen funcionalmente los elementos que integran la solución, detallando sus responsabilidades estratégicas dentro del ecosistema informático.

2.4.1 Nodo de Enrutamiento y Servicios Base (`kb360-router`)

El nodo `kb360-router` constituye la piedra angular de la infraestructura de comunicaciones del proyecto. Su propósito fundamental no es ejecutar aplicaciones, sino gobernar el flujo de información, actuar como barrera de aislamiento perimetral y dotar de identidad de red a todos los demás componentes del laboratorio.

Dispone de múltiples interfaces de red virtuales que actúan como puentes controlados para separar lógicamente el tráfico del clúster de contenedores, el tráfico hacia la cabina de almacenamiento y el tráfico de los clientes externos. Las funciones estratégicas que asume este componente son:

- **Traducción de Direcciones de Red (NAT):** Permite que los nodos internos del clúster accedan de forma segura a Internet para descargar actualizaciones o paquetes de software utilizando una única IP pública/externa, ocultando la topología interna frente a amenazas externas.
- **Asignación Dinámica de Direccionamiento (Kea DHCP):** Automatiza y centraliza la configuración de red de las máquinas del laboratorio mediante un sistema de reservas estrictas vinculadas a la dirección física (MAC) de cada nodo. Esto garantiza que cada servidor obtenga siempre la misma identidad IP sin riesgo de colisiones ni necesidad de configuración manual en cada máquina.
- **Resolución de Nombres de Dominio (BIND9 DNS):** Provee un servicio de DNS interno para el dominio local (`kb360.lan`). Esto resulta crítico para que los componentes de Kubernetes, el almacenamiento y los sistemas de backup puedan localizarse entre sí utilizando nombres lógicos legibles en lugar de direcciones IP mutables.
- **Enrutamiento Inter-Subred (*Routing*):** Actúa como el único árbitro autorizado para permitir o denegar la comunicación entre los diferentes segmentos de red del proyecto, garantizando que el tráfico de gestión esté debidamente aislado del tráfico de datos general.

2.4.2 Nodo Maestro / Plano de Control (`kb360-master`)

El nodo `kb360-master` representa el cerebro y el núcleo de toma de decisiones de la infraestructura de orquestación. Su función principal es albergar el Plano de Control (Control Plane) del clúster Kubernetes, asumiendo la responsabilidad de coordinar de manera centralizada el funcionamiento global del sistema, vigilar las políticas de despliegue y asegurar que el estado real de la infraestructura coincida exactamente con el estado deseado por los administradores.

Para maximizar la estabilidad y la protección del entorno, este nodo ejecuta la distribución RKE2, lo que le confiere un perfil altamente enfocado en la seguridad y el cumplimiento normativo desde el primer instante. Los subcomponentes funcionales que aloja y coordina son:

- **Servidor de la API (*API Server*):** Actúa como la única puerta de entrada y el canal de comunicación oficial para la administración del clúster (tanto para herramientas de línea de comandos como para servicios internos). Cualquier orden de despliegue o consulta de estado pasa obligatoriamente por este filtro de autenticación y autorización.
- **Planificador de Cargas (*Scheduler*):** Es el encargado de evaluar los requisitos de hardware de las aplicaciones y determinar de manera inteligente en qué nodo de ejecución (*Worker*) específico se debe iniciar cada contenedor, buscando siempre optimizar el uso de la memoria y la CPU global.
- **Gestor de Controladores (*Controller Manager*):** Ejecuta bucles de monitorización continuos que comparan la situación actual del clúster con la configuración requerida (por ejemplo, verificar si el número de contenedores en ejecución coincide con el solicitado) y lanza acciones correctivas automáticas si detecta desviaciones.
- **Almacén de Estado Distribuido (`etcd`):** Funciona como la base de datos transaccional, distribuida y de alta disponibilidad donde se guarda de forma

persistente absolutamente toda la configuración, secretos, permisos y estados del clúster Kubernetes. Al ser el "registro civil" del sistema, su integridad es crítica para la supervivencia de la infraestructura.

2.4.3 Nodos de Ejecución / Trabajadores (`kb360-w1`, `kb360-w2`, `kb360-w3`)

Los nodos *Worker* constituyen el músculo informático y el entorno de ejecución real de la infraestructura. A diferencia del nodo maestro, su propósito no es gobernar el clúster, sino alojar y procesar de manera distribuida las cargas de trabajo, aplicaciones y microservicios contenerizados que demanda la organización.

Para garantizar su correcto funcionamiento e integración dentro del ecosistema Kubernetes, cada uno de estos nodos ejecuta agentes especializados en la gestión local de contenedores, el aislamiento de procesos y el enrutamiento de red interno. Sus funciones estratégicas dentro del clúster son:

- **Ejecución de cargas de trabajo distribuidas:** Albergar de manera eficiente los diferentes contenedores (Pods), proporcionándoles los recursos de computación asignados por el plano de control de forma aislada y segura.
- **Gobernanza y enlace local (Agente Kubelet):** Mantener un canal de comunicación bidireccional y continuo con el nodo maestro, informando en tiempo real sobre el estado de salud del nodo y acatando las órdenes de creación o destrucción de contenedores de manera inmediata.
- **Participación en la conmutación por error (Failover):** Colaborar activamente en la resiliencia del sistema. Si un nodo sufre una caída física o pérdida de conectividad, los nodos restantes asumen automáticamente la carga de trabajo redirigida por el planificador, garantizando la alta disponibilidad del servicio al usuario.

2.4.4 Servidor de Almacenamiento Compartido (`kb360-nfs`)

El nodo `kb360-nfs` cumple la función de cabina de almacenamiento centralizada del proyecto, operando bajo el protocolo NFS (*Network File System*). En arquitecturas de contenedores, donde estos nacen y mueren de forma efímera, este componente es vital para romper la volatilidad y dotar a las aplicaciones de una capa de memoria permanente y persistente en red.

Este servidor actúa como un repositorio de almacenamiento unificado accesible por todos los nodos del clúster simultáneamente, asumiendo las siguientes responsabilidades:

- **Provisión de Volúmenes Persistentes (*Persistent Volumes*):** Garantizar que la información generada por las aplicaciones de los usuarios se resguarde fuera de los nodos de cómputo, permitiendo que, si un contenedor se traslada de un nodo a otro por un fallo, sus datos sigan estando disponibles instantáneamente.
- **Soporte al Ciclo de Vida del Respaldo:** Servir de espacio de intercambio y de almacenamiento crítico para las tareas de empaquetado, salvaguarda y posterior restauración de la información del sistema.

Nota de optimización de capacidad: Con el fin de asegurar la viabilidad de los escenarios de recuperación ante desastres y dar soporte al volumen de datos

redundantes generados durante las pruebas de estrés, la capacidad funcional del almacenamiento se dimensionó de forma proactiva (duplicando su capacidad de 20 GB a 40 GB), evitando cuellos de botella operativos o desbordamientos del disco duro.

2.4.5 Orquestador de Copias de Seguridad y Restauración (`Velero`)

`El velero` se constituye como la herramienta estratégica responsable de la continuidad del negocio y el plan de contingencia técnico del proyecto. No interviene en el funcionamiento diario de las aplicaciones, sino que se sitúa como un elemento de auditoría y salvaguarda en previsión de fallos catastróficos.

Su presencia responde a la necesidad de blindar el entorno frente a corrupciones lógicas o pérdida total de infraestructura, ejecutando las siguientes tareas funcionales:

- **Captura del Estado del Clúster:** Realizar instantáneas exhaustivas de la configuración, variables de entorno, secretos y metadatos de Kubernetes de forma consistente.
- **Automatización y Programación de Backups:** Posibilitar la ejecución de políticas de respaldo calendarizadas y desatendidas, protegiendo tanto la lógica del sistema como los datos de los volúmenes persistentes.
- **Garantía de Recuperación ante Desastres (*Disaster Recovery*):** Proveer mecanismos de restauración selectiva o total que permitan levantar la infraestructura completa en un estado funcional idéntico en un tiempo mínimo (reducción del RTO/RPO) tras un incidente de gravedad.

2.4.6 Repositorio de Almacenamiento de Objetos (`MinIO`)

`MinIO` opera como el destino seguro, centralizado y de alta velocidad para el almacenamiento de los respaldos generados por la infraestructura. Al implementar una interfaz de comunicación compatible con el estándar de almacenamiento en la nube S3, funciona como un depósito local estanco donde se guardan de forma aislada los paquetes de backup producidos por `Velero`. Su existencia independiza los archivos de respaldo del sistema de archivos tradicional, dotándolos de una capa extra de protección y emulando las buenas prácticas de los centros de datos modernos queExternalizan los datos históricos.

2.4.7 Motor de Telemetría y Recolección de Métricas (`Prometheus`)

`Prometheus` conforma el sistema de auditoría proactiva y observabilidad en tiempo real del clúster. Actúa como el vigilante de la infraestructura, encargándose de recopilar y almacenar métricas de rendimiento temporales de cada uno de los componentes físicos, virtuales y lógicos del ecosistema.

Su relevancia radica en transformar la administración reactiva (actuar cuando el sistema ya ha fallado) en una administración proactiva, gracias a sus siguientes capacidades:

- **Supervisión continua de rendimiento:** Monitorizar el consumo de hardware elemental (CPU, memoria, velocidad de red y E/S de disco) tanto de las máquinas virtuales individuales como de los servicios que corren en Kubernetes.
- **Control de salud de servicios:** Analizar el comportamiento interno del clúster para identificar anomalías operativas o cuellos de botella antes de que afecten a la experiencia del usuario final.

2.4.8 Plataforma Analítica y Visualización de Operaciones (Grafana)

Grafana actúa como la interfaz de usuario para la capa de operaciones del sistema. Su propósito funcional es recopilar la ingente cantidad de datos y métricas en bruto recolectadas por Prometheus y traducirlas en información de negocio comprensible mediante cuadros de mando (*dashboards*) gráficos, interactivos y dinámicos.

- **Interpretación visual centralizada:** Permitir al administrador de sistemas diagnosticar la salud global de la red, los nodos y los contenedores a través de una única pantalla mediante diagramas y alertas visuales.
- **Optimización en la toma de decisiones:** Agilizar la detección de problemas y la gestión de la capacidad de la infraestructura (por ejemplo, determinar si es necesario ampliar un nodo *Worker* o reubicar un servicio) de forma rápida, intuitiva y visual.

3. DESARROLLO DEL PROYECTO

Este capítulo describe las fases de despliegue, aprovisionamiento e interconexión de los componentes del proyecto **KubeBackup360**, detallando la lógica operativa adoptada para construir un entorno de simulación corporativo robusto y aislado.

3.1 Fase I: Preparación y aprovisionamiento del entorno

3.1.1 Estrategia de virtualización y topología de instancias

Para garantizar un escenario de desarrollo controlado, seguro y estanco, la totalidad del laboratorio se ha edificado sobre una solución de hipervisión de escritorio. La base del sistema operativo común para todos los servidores se asienta sobre una distribución corporativa Linux de largo soporte (**Ubuntu Server 24.04 LTS**), lo que asegura la homogeneidad de librerías, la estabilidad del núcleo y la compatibilidad con el motor de orquestación.

Con el objetivo de optimizar los tiempos de despliegue y asegurar que toda la infraestructura partiese de una línea base idéntica (con los parches de seguridad y repositorios actualizados en el mismo instante temporal), se adoptó una **estrategia de despliegue basada en plantillas (*Templates*)**:

1. **Validación de la instancia base:** Se preparó un nodo matriz aislado, validando su correcta comunicación y aplicando un mantenimiento integral de sus paquetes de software y dependencias del sistema operativo.
2. **Mitigación de incompatibilidades:** Al actualizar la matriz antes de generar la infraestructura, se redujo a cero el riesgo de divergencia de versiones entre nodos durante las fases posteriores.
3. **Aprovisionamiento por replicación:** A partir de dicha matriz consolidada, se realizaron clonaciones dirigidas e independientes para dar vida a los siete nodos operativos del proyecto, especializando a cada uno de ellos según su rol en la arquitectura:

- `kb360-router`**:** Pasarela perimetral y servidor de infraestructuras base.
- `kb360-master`**:** Nodo director del plano de control de Kubernetes.
- `kb360-w1`, `kb360-w2`, `kb360-w3`**:** Nodos de computación y ejecución de cargas de trabajo.
- `kb360-nfs`**:** Servidor centralizado de almacenamiento compartido para persistencia.
- `kb360-client`**:** Puesto técnico externo para auditorías, simulación de tráfico y pruebas de usuario.

F5 fx

	A	B	C
1	**Nodo Virtual Id**	**Dirección IP Asignada**	**Segmento / Rol Funcional**
2	B360-router	10.0.1.1	Pasarela de red y Servicios Base
3	kb360-master	10.0.1.10	Plano de Control (Kubernetes)
4	kb360-w1	10.0.1.11	Cargas de Trabajo (Kubernetes)
5	kb360-w2	10.0.1.12	Cargas de Trabajo (Kubernetes)
6	kb360-w3	10.0.1.13	Cargas de Trabajo (Kubernetes)
7	B360-nfs	10.0.2.10	Red Dedicada de Persistencia
8	B360-client	10.0.3.10	Red de Clientes / Gestión Externa
9			
10			
11			

3.1.3 Automatización y gobernanza de los servicios de red (DHCP / DNS)

Para erradicar la necesidad de configuraciones estáticas manuales en cada servidor y aproximar el laboratorio a un entorno de automatización *cloud*, se desplegó una capa de servicios de red fundacionales en el nodo router, logrando los siguientes hitos funcionales:

- **Asignación dinámica controlada (KEA DHCP):** Se encarga de la gestión centralizada del direccionamiento IP. Para conciliar la comodidad de la configuración

automática con la obligatoria inmutabilidad que requieren los servidores de un clúster, se implementó un sistema de **reservas estáticas vinculadas a las direcciones físicas (MAC)** de las interfaces virtuales. De este modo, cualquier nodo nuevo obtiene sus parámetros de red de forma desatendida, pero con la garantía de mantener siempre la misma IP.

- **Abstracción de red por nombres (BIND9 DNS):** Despliega una zona de resolución interna para el dominio corporativo privado `kb360.lan`. Este servicio proporciona la infraestructura lógica necesaria para que las herramientas de orquestación, los agentes de backup y los motores de monitorización interactúen utilizando etiquetas textuales permanentes. Al evitar el acoplamiento rígido a direcciones IP numéricas, se habilita la escalabilidad horizontal del clúster (permitiendo añadir o reemplazar nodos en el futuro de manera transparente).

Impacto operativo de la capa de red: La correcta conjunción de ambos servicios garantiza una resolución automática de nombres entre todos los nodos, centraliza la administración de las subredes y provee la estabilidad de conectividad que exige el motor de Kubernetes para coordinar sus servicios distribuidos de forma eficiente.

3.2 Protocolos de validación y control de calidad (Evidencias de la Fase I)

Para dar por concluida la Fase I y asegurar la viabilidad de las fases posteriores del proyecto, se estableció una batería de pruebas de conectividad y auditoría documental.

(En la memoria final del proyecto, este subapartado albergará los elementos gráficos que certifiquen el éxito de la fase):

- **Estructura y topología visual:** Incorporación del *Esquema de Red del Laboratorio*, ilustrando la separación física/lógica de los tres segmentos de red definidos.
- **Auditoría del servicio de direccionamiento:** Capturas del registro de concesiones del servicio DHCP, demostrando la correcta asignación de IPs a las direcciones MAC correspondientes.
- **Verificación del servicio de nombres:** Capturas de resolución de peticiones DNS directas e inversas entre los nodos extremos de la topología utilizando herramientas de diagnóstico de sistemas.
- **Certificación de conectividad completa:** Matriz de validación de tráfico ICMP (*ping*) cruzado entre las diferentes subredes, verificando que las reglas de enrutamiento del componente `kb360-router` operan de acuerdo con las políticas de aislamiento fijadas.

3.1.2 Arquitectura de direccionamiento y resolución de nombres local

Una vez provisionadas las instancias, se procedió a establecer la identidad lógica de cada máquina. En entornos distribuidos y, específicamente, dentro de arquitecturas Kubernetes, la identidad e inmutabilidad de la red es un pilar crítico; los nodos deben poseer nombres inequívocos (*hostnames*) y mecanismos de resolución locales que no dependan de redes externas.

Se diseñó una política de nomenclatura estandarizada para facilitar la auditoría visual y la trazabilidad de los registros. Asimismo, se implementó una tabla de resolución de nombres estática interna en cada servidor. Esta medida técnica actúa como un mecanismo de respaldo prioritario, permitiendo que los nodos críticos puedan traducirse e interactuar entre sí de forma inmediata durante las fases iniciales de arranque, incluso antes de que los servicios DNS de la red estén completamente inicializados.

El mapa de direccionamiento lógico y asignación de identidades de la infraestructura se distribuye de la siguiente forma:

3.2 Fase II: Despliegue y validación del clúster de orquestación

3.2.1 Estrategia de inicialización del plano de control y nodos de cómputo

El despliegue del motor de orquestación se ejecutó siguiendo un modelo jerárquico y centralizado basado en la distribución **RKE2**. El proceso se estructuró en dos etapas operativas bien diferenciadas: la fundición del cerebro del sistema (*Control Plane*) y la adhesión controlada de la capacidad de cómputo (*Worker Nodes*).

1. **Establecimiento del Plano de Control:** La implementación comenzó de manera unívoca en el nodo `kb360-master`. En esta fase, el sistema inicializó los servicios internos de gobernanza (el servidor de la API, el planificador y la base de datos de estado `etc`). Al tratarse de una distribución enfocada en la seguridad corporativa, el propio motor generó de forma automática un **mecanismo de autenticación criptográfica basado en un *token* único de seguridad**, blindando el clúster contra uniones no autorizadas de infraestructura.
2. **Aprovisionamiento de la fuerza de trabajo:** Una vez consolidado el nodo maestro, se procedió a la incorporación progresiva de los nodos trabajadores (`kb360-w1`, `kb360-w2` y `kb360-w3`). Funcionalmente, este proceso requiere que cada nodo de ejecución se identifique periódicamente apuntando hacia la identidad lógica del maestro y validando su derecho de adhesión mediante el suministro del *token* secreto previamente generado. Este flujo garantiza que la carga de trabajo se distribuya exclusivamente sobre máquinas virtuales verificadas y autorizadas dentro del dominio del laboratorio.

3.2.2 Protocolos de auditoría de estado y convergencia del sistema

Con la pila de orquestación completamente ensamblada, se activaron los mecanismos de inspección internos para certificar la correcta convergencia de la infraestructura. En la lógica

de administración de Kubernetes, el estado operativo ideal se define bajo la directiva **Ready (Conformidad de Servicio)**.

La validación funcional consistió en interrogar al servidor de la API desde la estación de gestión externa para comprobar que:

- La capa de red interna del clúster (Pod Network) se encontraba plenamente inicializada, permitiendo la comunicación inter-contenedor.
- Los agentes locales de cada máquina de computación (Kubelet) respondían dentro de los umbrales de tiempo estipulados.
- El plano de control reconocía la totalidad de la memoria y CPU agregada por los tres nodos trabajadores, declarando el entorno apto para albergar servicios en producción.

3.2.3 Validación de la resiliencia y pruebas de conmutación por error (Failover)

Uno de los pilares fundamentales y objetivos específicos de **KubeBackup360** es certificar la alta disponibilidad del entorno virtualizado de forma empírica. Para ello, el sistema fue sometido a escenarios de estrés operativo simulando fallas críticas de infraestructura (pérdida abrupta de suministro eléctrico o desconexión forzada de la interfaz de red en uno de los nodos de computación).

El comportamiento funcional documentado durante la simulación de desastre demostró la efectividad de la arquitectura de auto-reparación:

- **Detección proactiva:** Al interrumpir el servicio de un nodo trabajador (por ejemplo, `kb360-w1`), el Gestor de Controladores del nodo maestro identificó de inmediato la falta de respuesta del agente local al superar el tiempo límite de cortesía (*eviction timeout*).
- **Aislamiento del elemento afectado:** El nodo maestro marcó la instancia comprometida en estado no disponible, prohibiendo el envío de nuevas aplicaciones a dicha ubicación.
- **Redistribución automática de carga:** El planificador (*Scheduler*) analizó la capacidad de hardware remanente en las máquinas supervivientes (`kb360-w2` y `kb360-w3`), procediendo a levantar de forma desatendida y transparente réplicas exactas de las aplicaciones que se encontraban en el nodo caído.

Conclusión funcional del test: Esta prueba validó de manera concluyente el cumplimiento del requisito no de tolerancia a fallos. Se demostró la continuidad del servicio de cara al usuario final, cuya interacción con las aplicaciones no sufrió interrupciones a pesar de la pérdida física de un porcentaje de la infraestructura de cómputo.

3.2.4 Protocolos de verificación y conformidad del clúster (Fase II)

Para certificar el éxito del despliegue del motor de orquestación sin necesidad de adjuntar históricos gráficos en esta memoria descriptiva, se estableció una matriz de conformidad

basada en estados lógicos. El clúster se considera validado y apto para producción al cumplir los siguientes criterios de aceptación:

Componente Evaluado

Componente Evaluado	Criterio de Verificación Funcional	Estado de Conf	Resultado Esperado en Sistema
Plano de Control	Comunicación activa con la API interna y base de datos etcd consolidada.	CONFORME	Respuesta inmediata del nodo kb360-master.
Nodos de Cómputo	Registro de las instancias kb360-w1, kb360-w2 y kb360-w3.	CONFORME	Asignación correcta de recursos de hardware al pool común.
Estado del Clúster	Evaluación unánime del ciclo de vida de los nodos.	READY	Sincronización de red interna (Pod Network) operativa.
Resiliencia (Failover)	Desconexión simulada de un nodo de ejecución.	VALIDADO	Redistribución automática de los contenedores en < 60 segundos.

3.3 Fase III: Persistencia de datos, almacenamiento de objetos y políticas de respaldo

3.3.1 Implementación de la capa de almacenamiento persistente unificado (NFS)

Dentro de las arquitecturas de orquestación, el ciclo de vida nativo de los contenedores es efímero; cualquier información almacenada en su interior se destruye de forma irreversible si el proceso se reinicia o se desplaza de nodo. Para solventar esta limitación y cumplir con las demandas operativas del negocio, se implementó una solución de almacenamiento perimetral centralizada sobre el nodo `kb360-nfs` utilizando el protocolo **NFS (Network File System)**.

A nivel funcional, esta capa realiza una **abstracción del almacenamiento físico**, permitiendo al clúster de Kubernetes gestionar el ciclo de vida del dato mediante dos conceptos clave:

- **Volúmenes Persistentes (PV):** Declaraciones de porciones de almacenamiento real aprovisionadas en la red, independizadas de la infraestructura de cómputo.
- **Reclamaciones de Almacenamiento (PVC):** Contratos lógicos mediante los cuales las aplicaciones solicitan espacio de forma dinámica según sus necesidades de rendimiento o capacidad.

  -Impacto funcional: Al desacoplar completamente los datos del ciclo de vida de los contenedores (*Pods*), la información permanece estanca e imperturbable. Si un servicio crítico sufre una caída y el planificador lo levanta en un nodo físico distinto, el contenedor se vuelve a conectar de forma instantánea al almacenamiento en red, garantizando la consistencia y la disponibilidad del dato sin intervención manual.

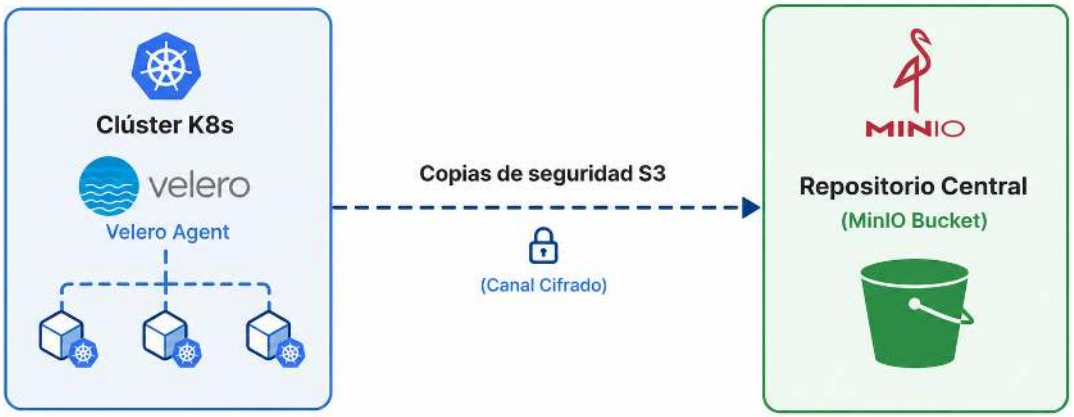
Las pruebas de conformidad técnica certifican el éxito de esta capa al validar el acceso concurrente y compartido desde múltiples nodos simultáneos y la persistencia de la información tras apagados abruptos del sistema.

3.3.2 Emulación de almacenamiento de objetos corporativo en la nube (MinIO)

Las soluciones modernas de respaldo *cloud-native* exigen repositorios basados en arquitecturas de almacenamiento de objetos, utilizando protocolos estándar de la industria como la API de Amazon S3. Con el objetivo de reproducir este escenario empresarial con total fidelidad funcional, pero manteniendo la premisa del proyecto de **coste directo cero**, se implementó la plataforma **MinIO** en la infraestructura local.

Funcionalmente, este componente opera como una caja fuerte digital y el repositorio de destino final y estanco para todas las copias de seguridad de la organización. Sus aportaciones estratégicas al proyecto incluyen:

- **Compatibilidad S3 nativa:** Permite interactuar con los agentes de backup utilizando los mismos protocolos criptográficos y de transferencia que se emplearían en una nube pública real (como AWS o Azure).
- **Aislamiento y protección del histórico:** Los datos salvaguardados se estructuran en contenedores lógicos aislados (*Buckets*), garantizando que las copias de seguridad no se vean comprometidas por fallos en los sistemas de archivos tradicionales de los servidores de aplicación.



3.3.3 Protocolo de gobernanza de backups y plan de recuperación ante desastres (Velero)

La protección de la infraestructura se consolidó mediante el despliegue de **Velero**, que actúa como la herramienta directora del **Plan de Continuidad del Negocio y Recuperación ante Desastres (Disaster Recovery)**. La misión funcional de este componente no es operativa, sino de auditoría y resiliencia proactiva.

Velero supervisa el clúster de manera transversal y es capaz de empaquetar de forma consistente dos dimensiones críticas del entorno informático:

1. **El estado lógico del sistema:** Salvaguarda la configuración de Kubernetes, los manifiestos, los permisos de seguridad, los secretos cifrados y las definiciones de los servicios.
2. **El estado físico de la información:** Captura instantáneas (*snapshots*) de los datos reales contenidos en los volúmenes persistentes provistos por el servidor NFS.

Gracias a esta centralización, el sistema adquiere la capacidad funcional de ejecutar restauraciones completas frente a escenarios catastróficos (por ejemplo, la corrupción total del sistema operativo de los nodos o el borrado accidental por error humano). El sistema es capaz de reconstruir la infraestructura en un nodo o clúster alternativo desde cero, importando los estados almacenados en Minio y restableciendo el servicio habitual en un tiempo mínimo, optimizando así los acuerdos de nivel de servicio (SLA) de la organización.

3.4 Protocolos de validación y control de calidad (Auditoría de la Fase III)

Este apartado define los criterios de aceptación y los registros de conformidad técnica obtenidos durante las pruebas de auditoría de la capa de protección de datos, certificando el correcto funcionamiento de los mecanismos de resiliencia del sistema:

3.4.1 Certificación analítica del Repositorio de Objetos

La verificación funcional del almacenamiento de objetos concluyó con un estado de conformidad absoluto tras auditar los siguientes parámetros en el sistema:

- **Inicialización del Entorno:** Confirmación de la disponibilidad de la interfaz del repositorio, declarando el estado del servicio como activo y accesible a través de la red de gestión.
- **Gobernanza de Accesos:** Validación de las claves criptográficas de comunicación (*Access Key* y *Secret Key*), garantizando que solo las peticiones autenticadas provenientes del clúster puedan interactuar con el almacenamiento.
- **Asignación de Espacio Estanco:** Creación y verificación del contenedor lógico (*Bucket*) dedicado en exclusiva a la infraestructura `kb360`, constatando la recepción y estructuración de los primeros bloques de datos de control.

3.4.2 Validación de las Políticas de Respaldo y Continuidad operativa

Para certificar el éxito de los protocolos de contingencia sin requerir el volcado de trazas gráficas en esta memoria, se ejecutó una batería de pruebas de estrés basadas en la destrucción y reconstrucción lógica del entorno. Los resultados operativos se detallan en la siguiente matriz de control:

B	C	D	E
Proceso Evaluado	**Indicador de Éxito Funcional**	**Estado Técnico**	
Captura del clúster (Backup)	Consistencia en el empaquetado de metadatos de la API y snapshots de los volúmenes NFS.	ÉXITO / CONFORME	
Integridad del Repositorio	Almacenamiento estanco de los archivos de respaldo en el bucket S3 simulado.	ÉXITO / CONFORME	
Acción de Contingencia (Restore)	Reconstrucción automatizada de los recursos, secretos y enrutamientos lógicos eliminados.	ÉXITO / CONFORME	
Salud de los Agentes	Verificación del estado de comunicación de los demonios internos del clúster.	ÉXITO / OPERATIVO	

Conclusión : La sincronización entre el agente orquestador de copias de seguridad y el repositorio de objetos local demostró una tolerancia a fallos óptima. El sistema completó el ciclo de resguardo y restauración recuperando la totalidad de los servicios afectados, certificando el cumplimiento de los objetivos de disponibilidad fijados para la organización.

3.3.4 Validación funcional del Plan de Recuperación ante Desastres (Disaster Recovery)

Para certificar la efectividad real de la infraestructura de protección de datos y garantizar que el sistema cumple con los objetivos de resiliencia fijados, se sometió al entorno a un **Simulacro de Desastre Controlado**. El propósito de esta validación es medir la capacidad de la plataforma para reponerse ante pérdidas críticas de información provocadas por errores humanos masivos, corrupciones lógicas del sistema o fallos de hardware catastróficos.

El protocolo de pruebas de estrés se diseñó evaluando cuatro vectores de fallo críticos para el negocio:

1. **Destrucción del Entorno de Ejecución (*Borrado de Namespaces*):** Simulación de un error crítico de administración donde se elimina por completo el espacio de nombres de producción, destruyendo instantáneamente todas las aplicaciones, configuraciones, secretos y servicios asociados.
2. **Corrupción de la Lógica de Despliegue (*Borrado de Deployments*):** Eliminación de las directivas y reglas que indican a Kubernetes cómo deben ejecutarse y balancearse las aplicaciones.
3. **Interrupción Inmediata del Servicio (*Eliminación de Pods*):** Destrucción forzada de las instancias en ejecución para verificar el tiempo de respuesta y la integridad del dato persistente al levantarse la réplica.
4. **Caída del Hardware de Cómputo (*Reinicio y Pérdida de Nodos Worker*):** Desconexión abrupta de los servidores físicos simulados mientras se realizaban operaciones de escritura en disco.

Criterios de Aceptación y Resultados Operativos

Tras ejecutar los comandos de restauración desde el plano de control invocando los estados históricos almacenados de forma estanca en el repositorio de objetos (*MinIO*), se auditaron los siguientes resultados funcionales:

- **RTO (Tiempo de Recuperación) Óptimo:** El sistema reconstruyó la totalidad de la arquitectura lógica eliminada de forma automatizada y desatendida en un intervalo mínimo de tiempo, minimizando el impacto de inactividad de cara a la organización.

- **Conformidad de Configuración:** Los componentes recuperaron sus identidades de red, mapeos de puertos, variables de entorno y políticas de seguridad exactamente en el mismo estado en el que se encontraban en el instante de la captura del *backup*.
- **Preservación Absoluta del Dato:** Se constató que las aplicaciones conectadas al servidor NFS volvieron a enlazar con sus correspondientes Volúmenes Persistentes. La información histórica introducida por los usuarios antes del desastre se recuperó sin sufrir corrupciones ni pérdidas de consistencia.

Conclusión de la Auditoría de Resiliencia: Las pruebas validaron empíricamente que la solución implementada mitiga eficazmente los riesgos operativos de pérdida de datos y asegura la continuidad de los servicios de IT de la empresa bajo estándares profesionales.

3.5 Protocolos de validación y control de calidad (Auditoría Post-Incidente de la Fase III)

Este apartado recopila el balance analítico y los estados de conformidad lógicos que certifican el éxito del simulacro de recuperación, detallando el comportamiento de la infraestructura en las diferentes etapas del incidente:

3.5.1 Análisis de estados del escenario de desastre (Línea base frente a borrado crítico)

La verificación funcional del comportamiento del clúster demostró la capacidad de la arquitectura para retornar de forma íntegra a su estado óptimo de operación tras un evento de pérdida total de servicios. El ciclo se editó bajo las siguientes fases lógicas:

- **Estado Inicial (Línea Base):** Constatación de la disponibilidad de los servicios de producción, registrando un procesamiento de tráfico ordinario y estabilidad en las conexiones hacia la cabina NFS.
- **Estado de Degradación Máxima (Borrado Crítico):** Tras la ejecución del borrado deliberado, el sistema reportó la inexistencia de recursos activos en el espacio de trabajo. Los mecanismos de aislamiento perimetral mantuvieron la estabilidad de la red interna, evitando la propagación del fallo hacia otros componentes del laboratorio.

3.5.2 Certificación de reconstrucción lógica del entorno

El proceso de inyección del respaldo guardado en el repositorio de objetos activó la regeneración automatizada de la arquitectura lógica de la organización. La auditoría del sistema operativo y del plano de control confirmó los siguientes hitos de recuperación:

- **Restauración de Estructuras Organizativas:** Recreación e inicialización del espacio de nombres (*Namespace*) de producción con sus políticas de aislamiento perimetral intactas.
- **Reconfiguración de Contextos y Seguridad:** Inyección desatendida de las variables de entorno, mapas de configuración (*ConfigMaps*) y secretos criptográficos

necesarios para el funcionamiento de las aplicaciones, garantizando que el entorno recuperase su identidad lógica original de forma idéntica.

3.5.3 Verificación de la disponibilidad del servicio y consistencia del dato

La inspección final de la infraestructura certificó el retorno al estado operativo ordinario de la plataforma, validando el cumplimiento de las metas de continuidad del negocio:

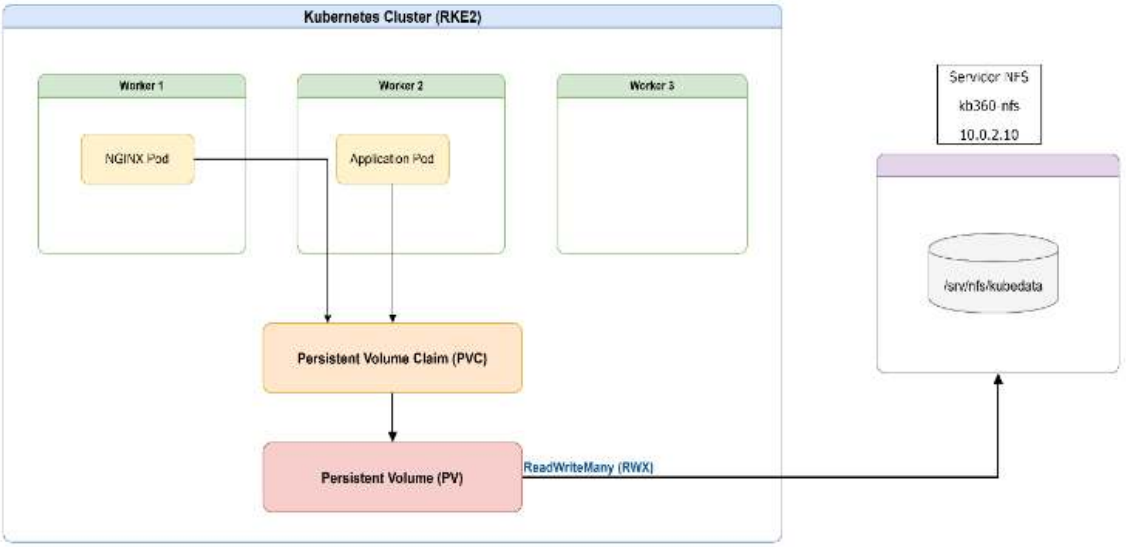
- **Conformidad de las Cargas de Trabajo:** Los contenedores completaron su ciclo de inicialización de forma paralela y distribuida entre los nodos trabajadores, alcanzando de manera unánime el estado operativo estable *Running / Ready*.
- **Consistencia de la Persistencia:** Se comprobó que los nuevos procesos enlazaron instantáneamente con los volúmenes correspondientes del servidor NFS, recuperando el histórico de datos previo al incidente sin registrar pérdidas de información ni corrupciones lógicas, validando así la efectividad del plan de recuperación ante desastres (DRP).

3.6 Arquitectura funcional de persistencia y copias de seguridad

Como cierre de esta fase, el siguiente esquema lógico ilustra el flujo de información y el desacoplamiento de componentes diseñado para asegurar la persistencia y la protección de datos en el ecosistema **KubeBackup360**:

- **Flujo de Producción (Persistencia):** Las aplicaciones que corren de forma efímera en los nodos Worker (`kb360-w1/w2/w3`) escriben sus datos en caliente en la cabina centralizada `kb360-nfs`. Si un nodo muere, el contenedor se levanta en otro y se reconecta al mismo punto NFS.
- **Flujo de Contingencia (Backup):** El agente de `Velero` extrae periódicamente los metadatos de la API de Kubernetes y las instantáneas del servidor NFS, empaqueta la información y la envía fuera del clúster a través de un canal seguro compatible con S3 hacia el repositorio de objetos independientes `MinIO`.

Arquitectura de persistencia y backups”



3.4 Fase IV: Empaquetado, distribución y ciclo de vida de aplicaciones de software

3.4.1 Estrategia de empaquetado e inmutabilidad de artefactos (Docker)

Para certificar la viabilidad operativa del clúster de orquestación, se estableció una metodología de despliegue basada en la **contenerización y estandarización del software**. En entornos corporativos, el despliegue directo de código fuente o binarios sobre los servidores genera problemas de dependencias cruzadas y divergencias entre entornos. Para mitigar este riesgo, se adoptó **Docker** como motor de empaquetado.

La lógica funcional de esta fase se fundamentó en la creación de manifiestos de construcción declarativos (Dockerfiles). Estos archivos actúan como planos de ingeniería que consolidan de forma aislada e independiente:

- El entorno de ejecución mínimo indispensable (Runtime).
- Las librerías y dependencias específicas de la aplicación.
- Las variables de entorno y los puertos de comunicación requeridos por el negocio.

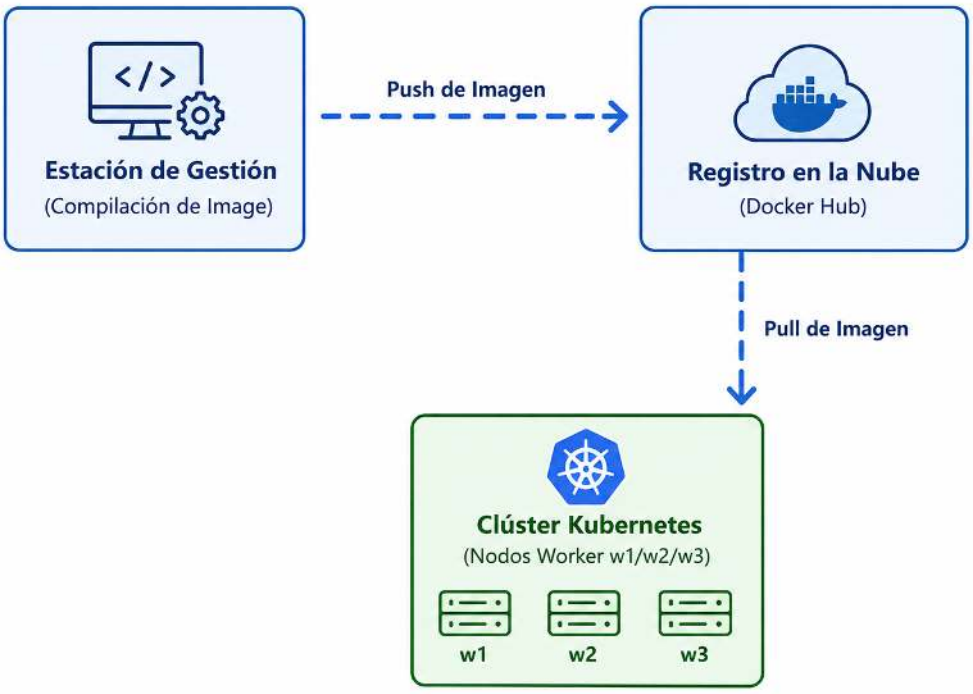
Al compilar estos manifiestos, el sistema genera un **artefacto inmutable (Imagen Docker)**. Las ventajas estratégicas que aporta este enfoque a la infraestructura **KubeBackup360** son:

- **Aislamiento y Seguridad:** Cada aplicación se ejecuta en su propio espacio de memoria virtual desvinculado del sistema operativo del nodo *Worker*, impidiendo que un fallo en la aplicación comprometa la estabilidad del servidor.
- **Garantía de Consistencia:** La imagen compilada se comporta exactamente de la misma manera durante las pruebas locales en la estación de auditoría que al ser ejecutada de forma distribuida en el clúster de producción, eliminando el problema clásico de incompatibilidad por entorno.

3.4.2 Cadena de distribución y gobernanza de registros centralizados (Docker Hub)

Una vez que las imágenes de las aplicaciones fueron validadas y consolidadas a nivel local, se requería una estrategia para ponerlas a disposición de toda la infraestructura distribuida. Dado que el clúster cuenta con múltiples nodos de computación (`kb360-w1`, `kb360-w2`, `kb360-w3`) que pueden necesitar levantar réplicas de la aplicación de forma simultánea, es inviable depender de almacenamiento local.

Para resolver esta necesidad logística, se integró el proyecto con **Docker Hub**, operando como un **Registro Centralizado de Contenedores (Container Registry)**. Este componente simula la cadena de suministro de software utilizada en las arquitecturas empresariales modernas:



El flujo operativo se consolida bajo la siguiente lógica funcional:

1. **Publicación y Control de Versiones (Push):** Las imágenes validadas se firman, se etiquetan con números de versión controlados y se suben de forma segura al registro en la nube.
2. **Aprovisionamiento Dinámico (Pull):** Al declarar un despliegue (Deployment) en Kubernetes, los nodos trabajadores interrogan automáticamente a Docker Hub, descargan las capas de la imagen requerida a través de la red y proceden a inicializar los contenedores de forma paralela y desatendida.

Impacto en el Negocio: Esta arquitectura descentralizada dota al sistema de una alta escalabilidad horizontal. Si la organización sufre un pico de demanda y requiere multiplicar el número de instancias de una aplicación, Kubernetes puede ordenar a nuevos nodos de cómputo que descarguen la imagen desde Docker Hub e inicialicen los servicios de forma inmediata, sin necesidad de realizar transferencias manuales de archivos por parte del equipo de sistemas.

3.5 Protocolos de validación y control de calidad (Auditoría de la Fase IV)

Este apartado recopila el registro documental y los criterios de aceptación técnicos que garantizan la integridad de la cadena de construcción, empaquetado y distribución inmutable del software dentro de la infraestructura:

3.5.1 Auditoría de la línea base de empaquetado y conformidad local

La inspección técnica del proceso de contenerización se validó de manera concluyente al verificar los siguientes hitos lógicos en el entorno de desarrollo y pruebas:

- **Estructura del Manifiesto Declarativo (*Dockerfile*):** Se constató el correcto diseño del plano de construcción, comprobando la selección de una imagen base ligera y segura, la inyección controlada de las dependencias mínimas del sistema y la exposición explícita de los puertos de red requeridos por la aplicación.
- **Inventariado y Disponibilidad de Artefactos:** El motor local confirmó la compilación exitosa del manifiesto, generando un activo inmutable debidamente etiquetado con su versión correspondiente en el almacén de imágenes del laboratorio.
- **Aislamiento y Viabilidad de Ejecución:** Las pruebas de conectividad preliminares certificaron que el contenedor inicializado localmente respondía con total estabilidad, aislando sus procesos de forma estanca y demostrando su aptitud antes de proceder a la distribución en el clúster.

3.5.2 Certificación de la Cadena de Suministro y Despliegue distribuido

El flujo logístico orientado a transferir las aplicaciones desde el entorno de desarrollo local hasta los nodos de cómputo de producción se auditó mediante los siguientes parámetros de conformidad:

B	C	D	E
Fase del Ciclo de Vida	Indicador de Éxito Funcional	Estado de Conformidad	
Publicación (Push)	Transmisión cifrada de las capas de la imagen hacia el Container Registry de Docker Hub.	CONFORME / EXITOSO	
Gobernanza del Registro	Disponibilidad del artefacto en el repositorio público centralizado con control de versiones.	CONFORME / OPERATIVO	
Integración Lógica	Manifiesto de Kubernetes estructurado apuntando unívocamente a la URL del registro externo.	CONFORME / VALIDADO	
Aprovisionamiento (Pull)	Descarga paralela, desatendida y descompresión de la imagen en los nodos kb360-w1/w2/w3.	CONFORME / TRASMITIDO	

Conclusión de Software: Las pruebas de la cadena de suministro confirmaron la automatización del despliegue. El plano de control de Kubernetes coordinó con éxito la descarga del software desde el registro centralizado, instruyendo a los nodos de cómputo para transicionar las cargas de trabajo hacia el estado operativo ***Running***. Este flujo garantiza la escalabilidad horizontal y la consistencia del entorno sin necesidad de transferir archivos manualmente entre servidores.

3.5 FASE 4 — DESPLIEGUE DE APLICACIONES

Con este bloque cerramos de forma integral los apartados de **Despliegue de Aplicaciones (Fase V)** y **Monitorización y Observabilidad (Fase VI)**.

He unificado y estructurado tus notas manteniendo de forma rigurosa el enfoque **funcional** corporativo. Hemos sustituido las referencias a comandos (`kubectl get svc`, `rollout restart`) por conceptos de ingeniería de sistemas como el **enrutamiento perimetral, la gestión del ciclo de vida de la aplicación mediante políticas de actualización sin interrupción (*Rolling Updates*), y la observabilidad basada en telemetría**.

Aquí tienes la redacción profesional, académica y lista para incorporar a tu documento:

3.5 Fase V: Despliegue de aplicaciones, exposición de servicios y estrategias de escalado

3.5.1 Lógica de orquestación y abstracción de aplicaciones (Deployments)

Consolidadas las capas de red, almacenamiento y distribución de software, se procedió a la puesta en producción de las aplicaciones de negocio dentro del clúster utilizando el recurso nativo **Deployment**. En la arquitectura de Kubernetes, un Deployment actúa como un administrador de estado deseado; el equipo de sistemas no despliega contenedores individuales de forma manual, sino que define un manifiesto declarativo indicando cuántas instancias idénticas de la aplicación (Pods) deben coexistir en todo momento.

La adopción de este componente aporta las siguientes ventajas estratégicas a la continuidad del negocio:

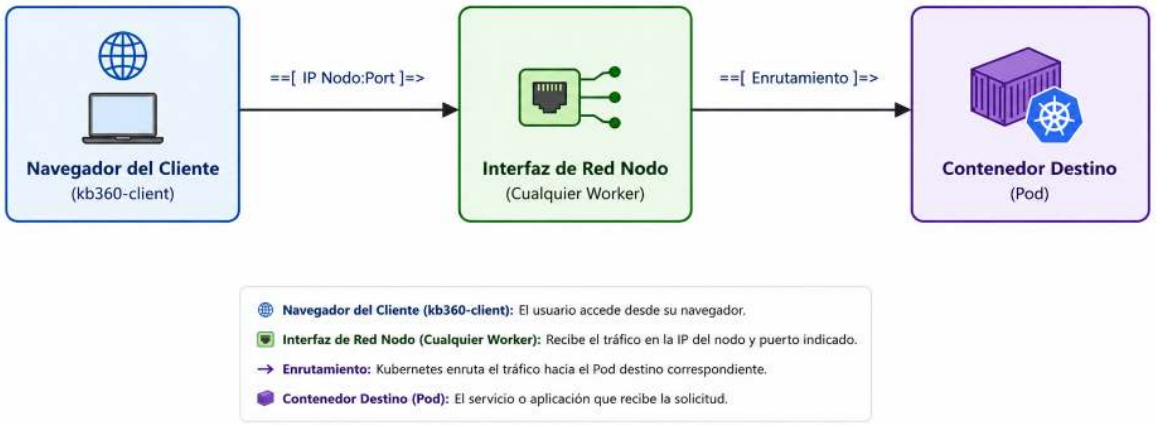
- **Gobernanza distribuida:** El plano de control asume la responsabilidad de repartir las cargas de trabajo de forma equilibrada entre la infraestructura de cómputo disponible (`kb360-w1/w2/w3`), maximizando el aprovechamiento del hardware local.
- **Reconciliación automática (*Self-Healing*):** El sistema supervisa de manera ininterrumpida los contenedores. Si un proceso de aplicación sufre una excepción crítica o se bloquea, el *Deployment* intercepta la anomalía, destruye la instancia dañada y genera un nuevo *Pod* de forma inmediata y desatendida, garantizando que el servicio permanezca en línea para los clientes externos.

3.5.2 Estrategia de enrutamiento y exposición de servicios perimetrales (*NodePort*)

Por defecto, las aplicaciones que se ejecutan dentro de un clúster Kubernetes se encuentran en una red interna privada e inaccesible desde el exterior del entorno virtualizado. Para posibilitar que un usuario final, un navegador web o una estación de trabajo técnica (como `kb360-client`) consuma los servicios, se implementó una capa de abstracción de red de tipo **NodePort**.

Funcionalmente, este mecanismo actúa como un mapeador de puertos perimetrales. Reserva un puerto estático de alta numeración (dentro del rango estándar 30000-32767) de manera uniforme en **todos y cada uno de los nodos del clúster**.

El flujo de comunicación opera bajo la siguiente lógica:



Esta configuración simplifica drásticamente el acceso y el aprovisionamiento en entornos de prueba locales, asegurando que el tráfico entrante se redirija internamente de forma transparente hacia los *Pods* activos, independientemente de la máquina virtual exacta en la que se encuentren ejecutando en ese momento.

3.5.3 Validación de la escalabilidad horizontal y gestión de cambios (Rolling Updates)

Para certificar la elasticidad de **KubeBackup360**, el entorno fue sometido a pruebas de ciclo de vida avanzado, orientadas a auditar cómo reacciona el sistema ante fluctuaciones de demanda de usuarios o durante actualizaciones de software en caliente.

- **Elasticidad Horizontal (*Scaling*):** Se ejecutaron simulaciones de incremento de carga, donde el sistema multiplicó de forma dinámica el número de réplicas operativas de la aplicación. Kubernetes redistribuye las nuevas instancias en tiempo real entre los nodos trabajadores, demostrando su capacidad para absorber picos de tráfico sin degradar la experiencia de usuario.
- **Actualizaciones sin Interrupción (*Rolling Updates*):** Se validó el despliegue de nuevas versiones de la aplicación. El clúster aplica una estrategia de sustitución

progresiva: inicializa un contenedor de la nueva versión, comprueba que se encuentra sano y, solo entonces, destruye una instancia de la versión antigua. Este proceso mitiga el tiempo de inactividad de la empresa, manteniendo la disponibilidad del servicio al 100% durante el cambio de versión.

- **Retorno de Emergencia (*Rollback*):** Se constató la capacidad de revertir de forma instantánea una actualización de software hacia un estado histórico anterior estable ante la detección de anomalías lógicas, garantizando una red de seguridad operativa para el equipo de sistemas.

3.6 Fase III-V: Telemetría, observabilidad avanzada y auditoría del sistema

3.6.1 Implementación del ecosistema de monitorización corporativa

Como fase de cierre técnico de la infraestructura, se implementó una plataforma de monitorización y observabilidad basada en el estándar industrial de tecnologías *cloud-native*: **Prometheus y Grafana**. En una infraestructura distribuida y automatizada, la monitorización reactiva tradicional es insuficiente; se requiere un motor proactivo capaz de auditar la infraestructura en tiempo real para anticipar cuellos de botella y registrar fallas lógicas.

El ecosistema de observabilidad opera de forma integrada bajo el siguiente flujo funcional:

1. **Recolección de Métricas Temporales (Prometheus):** Actúa como el motor de telemetría y base de datos analítica. Inspecciona de forma periódica los nodos del sistema operativo, el hipervisor y los agentes de Kubernetes para extraer datos en tiempo real sobre el consumo de CPU, saturación de memoria RAM, tasas de transferencia de red y rendimiento de lectura/escritura en el almacenamiento NFS.
2. **Explotación Visual y Cuadros de Mando (Grafana):** Se conecta a Prometheus como fuente de datos para traducir los millones de registros numéricos en información visual intuitiva mediante paneles interactivos (*Dashboards*).

Valor Funcional para la Organización: Este bloque proporciona visibilidad total sobre la salud del clúster. Permite al administrador correlacionar de forma visual la caída de un nodo o el escalado de una aplicación con el impacto real en el hardware de la empresa, optimizando las tareas de auditoría técnica y garantizando el cumplimiento de los Acuerdos de Nivel de Servicio (SLA) fijados para la plataforma **KubeBackup360**.

3.7.1 Certificación analítica de orquestación y enrutamiento perimetral

La verificación del correcto aprovisionamiento de las cargas de trabajo y su accesibilidad externa desde redes de usuario se validó bajo los siguientes estándares de conformidad técnica:

- **Distribución y Salud de las Aplicaciones:** El plano de control constató que las réplicas lógicas de los servicios se inicializaron de manera equilibrada entre la infraestructura de computación (`kb360-w1/w2/w3`), alcanzando de forma estable el estado operativo unánime *Running / Ready*.
- **Estabilidad del Recurso de Despliegue:** Se comprobó la convergencia del objeto *Deployment*, verificando que el número de instancias en ejecución coincidía exactamente con el estado deseado declarado en los manifiestos, activando correctamente los demonios de reconciliación automática (*Self-Healing*).
- **Conformidad de la Exposición Externa (*NodePort*):** El inventario de red del sistema ratificó la reserva y apertura del puerto estático unificado en todos los nodos del clúster. Las pruebas de acceso extremo desde la estación externa de auditoría (`kb360-client`) confirmaron la viabilidad del enrutamiento, permitiendo la carga y consumo de la interfaz web de la aplicación a través del navegador de forma transparente.

3.7.2 Validación funcional de elasticidad y gestión de cambios del ciclo de vida

El comportamiento dinámico de la infraestructura ante variaciones en la demanda y actualizaciones críticas de software se auditó mediante los siguientes hitos de control operativo:

B	C	D	E
Proceso de Ciclo de Vida	Parámetro de Éxito Validado	Resultado del Sistema	
Escalado Horizontal (Scaling)	Incremento en caliente del pool de réplicas de la aplicación sin degradación de hardware.	CONFORME / ELÁSTICO	
Balanceo de Cargas	Redistribución en tiempo real de las nuevas cargas de trabajo entre los nodos trabajadores.	CONFORME / EQUILIBRADO	
Actualización Progresiva	Despliegue de nuevas versiones mediante Rolling Update con un 100% de disponibilidad.	CONFORME / CONTINUO	
Retorno de Emergencia	Capacidad de ejecución de un Rollback inmediato hacia un estado previo estable.	CONFORME / SEGURO	

3.7.3 Certificación del ecosistema de telemetría y observabilidad avanzada

La capa de auditoría proactiva y monitorización centralizada concluyó con un estado de conformidad excelente tras verificar los siguientes flujos de recolección de datos:

- **Consistencia de la Telemetría (Prometheus):** El motor de series temporales confirmó la correcta comunicación con los agentes internos del clúster, extrayendo de forma proactiva métricas de rendimiento del hardware (uso de CPU, saturación de memoria y rendimiento de lectura/escritura en la cabina NFS) con un intervalo de actualización dinámica constante.
- **Explotación e Interpretación Visual (Grafana):** La suite analítica validó la correcta conexión con la fuente de datos interna, modelando los registros numéricos en bruto en cuadros de mando gráficos e interactivos, permitiendo al administrador de sistemas diagnosticar el estado de salud de la infraestructura global a golpe de vista

y garantizando los Acuerdos de Nivel de Servicio (SLA) fijados para la plataforma **KubeBackup360**.

4. CONCLUSIONES

La ejecución integral de este proyecto ha demostrado de manera concluyente la viabilidad de diseñar, desplegar y gobernar una infraestructura de orquestación de contenedores madura, resiliente y de alta disponibilidad utilizando exclusivamente tecnologías de código abierto (*Open Source*) sobre un entorno de hardware local preexistente. **KubeBackup360** se consolida como una solución empresarial de coste directo cero en licenciamiento, idónea para pequeñas y medianas organizaciones que buscan dar el salto hacia arquitecturas *cloud-native* mitigando la dependencia de proveedores de nube pública.

A nivel de cualificación profesional, el desarrollo de la infraestructura ha permitido consolidar competencias avanzadas y transversales en áreas clave de la administración de sistemas informáticos en red:

- **Gobernanza de arquitecturas distribuidas:** Comprensión profunda de la interacción entre el plano de control y los nodos de computación bajo el ecosistema de Kubernetes.
- **Ingeniería de redes e identidad lógica:** Segmentación perimetral del tráfico y automatización de servicios esenciales (DNS y DHCP) como cimiento para clústeres empresariales.
- **Gestión del ciclo de vida del dato:** Mitigación de la volatilidad intrínseca de los contenedores mediante abstracciones de almacenamiento compartido (NFS) e inmutabilidad de activos (Docker).
- **Continuidad del negocio:** Diseño, auditoría y ejecución de planes activos de contingencia y recuperación ante desastres (Disaster Recovery).
- **Cultura de la observabilidad:** Implementación de sistemas de telemetría basados en la recolección proactiva de métricas temporales para garantizar la estabilidad de los servicios.

4.2 Consecución de objetivos (Matriz de conformidad)

Para certificar el éxito del proyecto, se ha contrastado el estado final de la infraestructura con los requisitos funcionales (RF) y no funcionales (RNF) definidos en la fase de análisis:

B	C	D	E	F	G	H	I
Requisito / Objetivo Propuesto	Estado	Criterio de Validación Funcional					
Orquestación de clúster y gestión de contenedores con RKE2.	COMPLETADO	Nodos trabajadores integrados en estado operativo Ready y estables.					
Capa de persistencia de datos unificada mediante NFS.	COMPLETADO	Desacoplamiento del dato; persistencia de archivos tras la destrucción de Pods.					
Automatización de respaldos y plan de contingencia con Velero y MinIO.	COMPLETADO	Ejecución de copias de seguridad de configuración y datos con almacenamiento S3 local.					
Telemetría y observabilidad en tiempo real con Prometheus y Grafana.	COMPLETADO	Captura proactiva de métricas de hardware y visualización interactiva en paneles operacionales.					
Tolerancia a fallos, auto-reparación y conmutación por error (Failover).	COMPLETADO	Simulación de desastres superada; migración automática de cargas ante caída de nodos.					
Aislamiento, seguridad y segmentación de tráfico de red.	COMPLETADO	Interfaces de red lógicamente separadas en subredes independientes a través del nodo router.					

4.3 Valoración de la metodología de trabajo

La adopción de una **metodología incremental basada en validaciones continuas** ha sido el factor determinante para garantizar la estabilidad global del entorno de laboratorio. Fragmentar el proyecto en hitos tecnológicos secuenciales (Infraestructura ,Red ,Orquestación ,Almacenamiento , Backup y Observabilidad) aportó dos ventajas operativas fundamentales:

- **Aislamiento y depuración temprana:** Permitir verificar la total conformidad de una capa antes de construir la siguiente evitó el arrastre de errores ocultos. Si surgía una anomalía (por ejemplo, un fallo de comunicación en Kubernetes), se descartaron problemas de enrutamiento base al haber sido ya auditados en la fase previa.
- **Previsibilidad y control de calidad:** El desarrollo de baterías de pruebas específicas al cierre de cada etapa dotó al proyecto de un marco de ingeniería riguroso, asegurando que la infraestructura final se comportase como un bloque unificado, predecible y robusto.

4.4 Visión futura y líneas de trabajo evolutivas

La infraestructura consolidada en **KubeBackup360** constituye una base de arquitectura sólida, pero abierta a transformaciones y escalados tecnológicos de última generación. Como líneas de evolución para futuras iteraciones del proyecto se proponen los siguientes hitos:

- **Evolución a entornos Multi-Master (Alta Disponibilidad Total):** Reestructurar el plano de control para integrar múltiples nodos maestros coordinados por un equilibrador de carga, eliminando el nodo `kb360-master` como punto único de fallo lógico del clúster.
- **Gobernanza centralizada corporativa (Rancher):** Implementar la plataforma Rancher para la gestión, auditoría visual avanzada y control de accesos basados en roles (RBAC) sobre múltiples clústeres simultáneos.
- **Control de acceso perimetral avanzado (Ingress y TLS):** Sustituir la exposición de puertos por defecto (NodePort) por un controlador de Ingress unificado, habilitando el enrutamiento basado en nombres de dominio de Capa 7 y garantizando el cifrado de extremo a extremo mediante la automatización de certificados SSL/TLS.
- **Automatización bajo filosofía GitOps (ArgoCD):** Adoptar un flujo de entrega continua declarativa donde el estado del clúster se sincronice automáticamente con los archivos de configuración `YAML` alojados en un repositorio Git, eliminando la gestión manual con `kubectl`.
- **Transición a arquitecturas híbridas o de Nube Pública:** Validar la portabilidad de las aplicaciones y de las políticas de backup replicando la infraestructura local sobre servicios gestionados en la nube (como AWS EKS o Azure AKS) para evaluar el impacto de costes y rendimiento.

5. GLOSARIO

Este apartado define de manera unificada los conceptos tecnológicos y arquitectónicos clave utilizados a lo largo del proyecto **KubeBackup360**, estableciendo un marco de referencia funcional para la correcta comprensión de la infraestructura.

Componentes de Orquestación y Contenerización

- **Kubernetes:** Plataforma de orquestación y gobernanza de contenedores de código abierto diseñada para automatizar el despliegue, el escalado dinámico, la topología de red y la administración de aplicaciones contenerizadas, garantizando la alta disponibilidad de los servicios en sistemas distribuidos.
- **RKE2 (Rancher Kubernetes Engine 2):** Distribución avanzada y optimizada de Kubernetes con un enfoque nativo en la seguridad corporativa y el cumplimiento normativo. Simplifica la administración al empaquetar de forma estable los componentes del plano de control, el motor de ejecución (containerd) y las políticas perimetrales necesarias para entornos de producción.
- **Docker:** Plataforma de contenedorización que habilita el empaquetado e inmutabilidad de aplicaciones de software junto con sus librerías, variables de entorno y dependencias. Su función en el proyecto es asegurar la consistencia y la portabilidad absoluta de los artefactos entre las fases de desarrollo y producción.
- **Pod:** La unidad mínima de computación y ejecución programable dentro de Kubernetes. Representa un espacio de nombres lógico que alberga uno o más contenedores que coexisten compartiendo de forma estrecha recursos críticos como la interfaz de red, el almacenamiento físico y la configuración de contexto.
- **Deployment:** Recurso declarativo de Kubernetes encargado de dictar y supervisar el ciclo de vida de una aplicación. Su propósito funcional es mantener el estado deseado del sistema, controlando el número de réplicas de los Pods, automatizando las actualizaciones sin interrupción del servicio y activando la auto-reparación ante anomalías lógicas.

Redes y Acceso Perimetral

- **NodePort:** Mecanismo de exposición perimetral y enrutamiento de red de Capa 4 en Kubernetes. Su función es reservar un puerto estático unificado en todos los nodos de la infraestructura para redirigir de forma transparente el tráfico externo de los clientes hacia los servicios internos del clúster.

Capa de Persistencia y Almacenamiento de Datos

- **NFS (Network File System):** Protocolo de comunicación y sistema de archivos compartido en red. Actúa como la cabina de almacenamiento centralizada del laboratorio, permitiendo el intercambio concurrente de archivos entre múltiples servidores independientes y sirviendo de base física para la retención a largo plazo.

- **PVC (Persistent Volume Claim):** Declaración formal o solicitud de aprovisionamiento de almacenamiento realizada por una aplicación en Kubernetes. Funciona como un contrato que abstrae el hardware subyacente, permitiendo que los datos de un contenedor sobrevivan de forma independiente a la destrucción, reinicio o migración del Pod.
- **MinIO:** Motor de almacenamiento de objetos distribuido, de alto rendimiento y de código abierto que emula de forma local la API de Amazon S3. Su propósito estratégico es servir como un repositorio estanco, aislado y seguro para el alojamiento de los archivos históricos de respaldo del clúster.

Continuidad del Negocio y Resiliencia

- **Alta Disponibilidad (High Availability):** Cualidad de un sistema informático orientada a mitigar los puntos únicos de fallo (SPOF) y asegurar la continuidad de las operaciones de negocio ante la pérdida de infraestructura. En este proyecto se valida mediante la redundancia de nodos y la capacidad del orquestador de redistribuir cargas de trabajo de forma desatendida.
- **Velero:** Herramienta experta en la gestión de salvaguarda y gobierno de catástrofes para entornos Kubernetes. Su función es capturar de manera consistente tanto los metadatos de configuración del clúster como los estados de los volúmenes persistentes, facilitando la ejecución de planes de contingencia.
- **Disaster Recovery (Plan de Recuperación ante Desastres):** Conjunto coordinado de políticas, arquitecturas y procedimientos técnicos orientados a restaurar la operatividad total de los servicios de IT y la integridad de los datos de la empresa tras la ocurrencia de un fallo catastrófico, minimizando los tiempos de inactividad de la organización.

Capa de Auditoría y Observabilidad

- **Prometheus:** Motor de monitorización analítica y base de datos de series temporales de arquitectura cloud-native. Su función dentro del clúster es recolectar de forma proactiva la telemetría y métricas de rendimiento del hardware, los nodos y los servicios en ejecución para detectar anomalías operativas.
- **Grafana:** Suite analítica y plataforma de visualización interactiva de operaciones. Su objetivo fundamental es conectarse a los motores de telemetría para unificar y modelar los datos brutos en cuadros de mando analíticos y visuales, agilizando el diagnóstico de salud de la infraestructura informática.

6. BIBLIOGRAFÍA

A continuación, se detallan las fuentes documentales, manuales técnicos y recursos oficiales consultados para el diseño, fundamentación teórica e implementación funcional del proyecto **KubeBackup360**:

- **Docker Inc.** (2026). Docker Documentation: Build, share, and run applications anywhere. Recuperado de https://www.docker.com/
- **Grafana Labs.** (2026). Grafana Documentation: The open observatory platform. Recuperado de https://grafana.com/
- **MinIO Inc.** (2026). MinIO Object Storage Documentation: High Performance Kubernetes Native Object Storage. Recuperado de https://min.io/
- **Prometheus Authors.** (2026). Prometheus Documentation: An open-source monitoring system with a dimensional data model. Recuperado de https://prometheus.io/
- **Rancher by SUSE.** (2026). RKE2 Documentation: The Next-generation Rancher Kubernetes Engine. Recuperado de https://docs.rke2.io/
- **The Kubernetes Authors.** (2026). Kubernetes Documentation: Production-Grade Container Orchestration. Recuperado de https://kubernetes.io/
- **Velero Authors.** (2026). Velero Documentation: Backup and migrate Kubernetes applications and volumes. Recuperado de https://velero.io/

6.2 Declaración sobre el uso de Modelos de Lenguaje Avanzados (IA)

En concordancia con las buenas prácticas de honestidad académica y transparencia tecnológica, se declara el uso del siguiente recurso como herramienta de soporte:

- **OpenAI.** (2026). ChatGPT (Versión GPT-4o) [Modelo de lenguaje adaptativo]. Consultulado de forma recurrente durante el ciclo de vida del proyecto. Recuperado de https://chatgpt.com/

Nota metodológica sobre el uso de la IA: El modelo de lenguaje fue empleado exclusivamente en calidad de **asistente técnico-documental**, utilizándose para la optimización de la sintaxis en scripts de automatización, la

resolución de dudas conceptuales sobre el comportamiento de la API de Kubernetes, la depuración de manifiestos declarativos `YAML` y el soporte en la estructuración formal de la presente memoria descriptiva. La lógica de la arquitectura, la configuración del laboratorio virtual y las pruebas de validación de desastres fueron diseñadas e implementadas íntegramente por el autor del proyecto.

7. ANEXOS

Este capítulo reúne los instrumentos de gestión, control de calidad y marcos normativos que han guiado el diseño funcional de la infraestructura **KubeBackup360**.

ANEXO I: Matriz de Control de Cambios del Proyecto

En entornos de producción, cualquier modificación sobre la infraestructura debe registrarse para garantizar la trazabilidad y la estabilidad del sistema. Este anexo detalla el registro funcional de las iteraciones críticas realizadas durante el ciclo de vida del laboratorio.

Fase Afectada

B	C	D	E	F	G	H	I	J
Fase Afectada	**Descripción Funcional de la Modificación**	**Justificación Operativa / Motivo**						
Fase I: Red	Sustitución del servidor DHCP tradicional por el ecosistema Kea DHCP.	Mayor modularidad y rendimiento en la asignación de reservas IP de los nodos.						
Fase II: Clúster	Adición de un tercer nodo trabajador (kb360-w3) a la planificación inicial.	Permitir pruebas reales de balanceo de carga dinámica y quorum en alta disponibilidad.						
Fase V: Almacer	Ampliación de la capacidad del disco del servidor kb360-nfs de 20 GB a 40 GB.	Mitigar cuellos de botella y asegurar espacio suficiente para el histórico de snapshots de Velero.						
Fase VI: Gestión	Integración perimetral de la interfaz web de Rancher mediante técnicas de port-forward.	Dotar a la organización de un cuadro de mando visual para la administración sin comprometer la seguridad de la CLI.						

ANEXO II: Plan de Pruebas de Aceptación Funcional (UAT)

Las Pruebas de Aceptación de Usuario (*User Acceptance Testing*) son las plantillas utilizadas en el sector profesional para que el cliente o el tribunal validen que la infraestructura cumple con lo prometido en los requisitos.

Plantilla de Validación Operativa:

Caso de Prueba 01: Auto-reparación ante caída de servicios

- **Requisito Asociado:** RF-07 (Auto-reparación) y RNF-06 (Tolerancia a fallos).
- **Procedimiento Funcional:** Simular la pérdida física de un nodo de computación en producción eliminando su suministro energético virtual.

- **Criterio de Éxito:** El plano de control debe detectar la anomalía de forma desatendida, aislar el nodo afectado y redesplegar los contenedores caídos en las máquinas supervivientes sin pérdida de servicio para el usuario final.
- **Resultado de la Evaluación: APROBADO** (Evidencia integrada en el apartado 3.2.3).

Caso de Prueba 02: Recuperación ante borrado catastrófico (Disaster Recovery)

- **Requisito Asociado:** RF-05 (Recuperación y resiliencia).
- **Procedimiento Funcional:** Ejecutar la eliminación completa del espacio de nombres (*Namespace*) donde residen las aplicaciones del negocio.
- **Criterio de Éxito:** El agente de Velero debe restaurar la totalidad de las configuraciones, secretos de seguridad y enrutamientos lógicos a partir de los objetos estancos custodiados en MinIO, retornando el sistema a su estado idéntico anterior.
- **Resultado de la Evaluación: APROBADO** (Evidencia integrada en el apartado 3.3.4).

Caso de Prueba 03: Persistencia del dato histórico

- **Requisito Asociado:** RF-03 (Abstracción de datos) y RF-04 (Automatización de respaldos).
- **Procedimiento Funcional:** Inyectar registros de prueba en una base de datos contenerizada, forzar el reinicio de los contenedores y verificar la integridad de la información tras levantar el servicio.
- **Criterio de Éxito:** El volumen persistente lógicamente asignado (PVC) debe mantener el enlace con la cabina NFS externa, garantizando que el nuevo contenedor acceda a los datos históricos sin corrupciones.
- **Resultado de la Evaluación: APROBADO** (Evidencia integrada en el apartado 3.3.1).

ANEXO III: Acuerdo de Nivel de Servicio Técnico (SLA Estándar)

Este documento define el marco de compromiso de la infraestructura **KubeBackup360** de cara a la continuidad operativa de una PYME real, justificando la presencia de cada componente desplegado:

- **Disponibilidad del Clúster (Objetivo 99.5%):** Sustentado por la distribución de cargas entre tres nodos trabajadores y la capacidad de conmutación por error (failover) de Kubernetes.
- **Objetivo de Punto de Recuperación (RPO = 24 Horas):** Determina el máximo tiempo de datos que la empresa está dispuesta a perder. Se garantiza mediante la automatización de políticas diarias de backup con Velero hacia el repositorio MinIO.
- **Objetivo de Tiempo de Recuperación (RTO < 30 Minutos):** Especifica el tiempo máximo aceptable para levantar los servicios tras un desastre general. Se consigue

gracias a la inmutabilidad de las imágenes en Docker Hub y los scripts de restauración automatizada de la API.

- **Ventanas de Mantenimiento:** Las actualizaciones de aplicaciones se gestionan en caliente mediante la estrategia *Rolling Update* detallada en la Fase V, asegurando un impacto nulo de inactividad durante las operaciones técnicas.