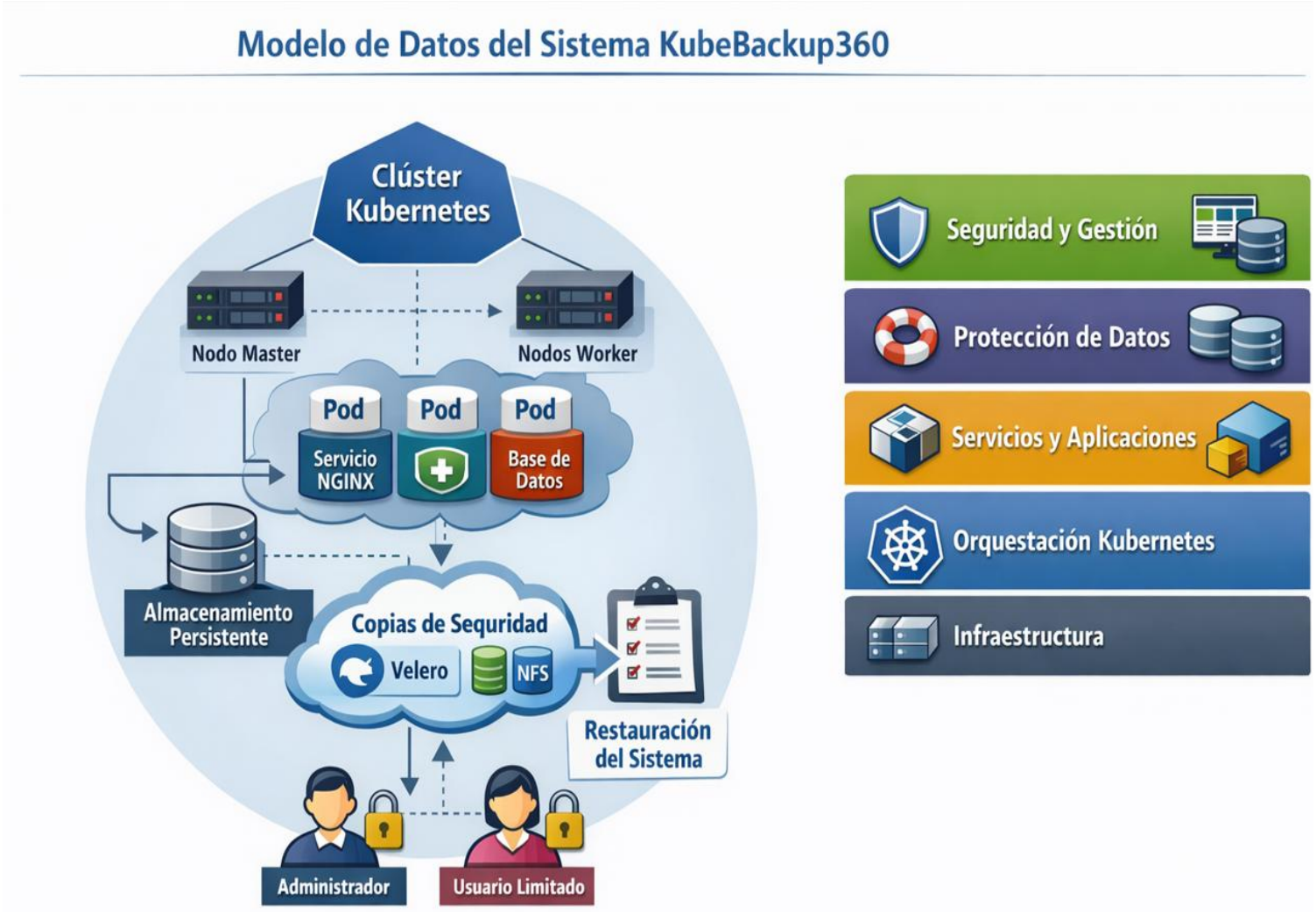
Institut Puig Castellar



ALUMNE/GRUP:

Clara Inés Nchama Ondo Mangue

Ilias Brahimi Bouaqaiq

Luis Alfredo Florez Maldonado

1. Introducción y contexto

El proyecto **KubeBackup360** surge ante la necesidad de disponer de un sistema fiable, automatizado y seguro para la gestión de copias de seguridad en entornos basados en contenedores y orquestados mediante Kubernetes. En infraestructuras modernas, donde los servicios se ejecutan de forma distribuida y dinámica, la pérdida de datos, errores de configuración o fallos en los nodos pueden provocar interrupciones graves si no existe una estrategia adecuada de respaldo y recuperación.

El problema principal que se pretende resolver es la **falta de un mecanismo centralizado y automatizado** que permita proteger tanto la configuración del clúster como los datos persistentes de las aplicaciones, garantizando la continuidad del servicio y una rápida recuperación ante incidentes. En muchos entornos de aprendizaje o laboratorio, estas tareas se realizan de forma manual o no se contemplan, lo que dificulta la comprensión de cómo funcionan las infraestructuras reales en producción.

El usuario o cliente final del sistema será, principalmente, el **administrador de sistemas** o el **estudiante de Administración de Sistemas Informáticos en Red (ASIR)**, que necesita gestionar un clúster Kubernetes, supervisar su estado y asegurar la integridad de la información. *De forma indirecta, también se benefician los usuarios de las aplicaciones desplegadas en el clúster, ya que dispondrán de servicios más estables y protegidos frente a fallos*.

La solución propuesta consiste en el diseño e implementación de una **infraestructura Kubernetes gestionada con Rancher**, que integre un sistema de **copias de seguridad automatizadas mediante Velero y almacenamiento NFS**, junto con herramientas de **monitorización y seguridad**.

El propósito del proyecto es doble: por un lado, ofrecer una solución técnica funcional que permita realizar backups y restauraciones de forma controlada; por otro, servir como **guía didáctica y práctica**, documentada paso a paso, que facilite el aprendizaje y la comprensión de cómo se construyen y administran infraestructuras modernas, resilientes y seguras.

2. Análisis de requisitos

2.1. Requisitos funcionales (RF) organizados por fases

A continuación, se enumeran los **requisitos funcionales del sistema KubeBackup360**, organizados por **fases de desarrollo**. Esta estructura permite que el proyecto sea **progresivo y escalable**, de forma que cada fase pueda considerarse un entregable independiente. Si por tiempo, recursos o conocimientos no se completa todo el proyecto, se podrá cerrar en la fase alcanzada manteniendo un resultado funcional y defendible.

FASE 1 — Preparación del entorno y virtualización (VirtualBox + Ubuntu)

Código	Descripción del requi sito funcional
RF1	El sistema permitirá crear y configurar las máquinas virtuales necesarias para el laboratorio (nodo master, nodos worker y nodos auxiliares si aplican).
RF2	El sistema permitirá configurar la conectividad de red entre todas las máquinas del laboratorio (red interna y acceso de administración).
RF3	El sistema permitirá habilitar acceso remoto seguro para administración (por ejemplo, mediante SSH) en los nodos del laboratorio.

FASE 2 — Despliegue del clúster Kubernetes (RKE2)

Código	Descripción del requisito funcional
RF4	El sistema permitirá instalar y poner en funcionamiento un clúster Kubernetes basado en RKE2.
RF5	El sistema permitirá unir nodos worker al nodo master/control-plane mediante el mecanismo de registro (token).
RF6	El sistema permitirá verificar el estado del clúster y de los nodos (por ejemplo, nodos en estado *Ready*).
RF4	El sistema permitirá instalar y poner en funcionamiento un clúster Kubernetes basado en RKE2.

FASE 3 — Gestión centralizada del clúster (Rancher)

Código	Descripción del requisito funcional
RF7	El sistema permitirá administrar el clúster mediante una interfaz web de gestión centralizada.
RF8	El sistema permitirá visualizar desde la interfaz el estado de nodos, pods, namespaces y servicios.
RF9	El sistema permitirá realizar operaciones básicas de gestión desde la interfaz (consulta, despliegue y supervisión de recursos).

FASE 3.5 — Creación y despliegue de contenedores personalizados (Docker → RKE2)

Código	Descripción del requisito funcional
RF10	El sistema permitirá construir imágenes de contenedores personalizadas a partir de un Dockerfile.
RF11	El sistema permitirá probar localmente las imágenes construidas antes de desplegarlas en el clúster.
RF12	El sistema permitirá publicar imágenes en un repositorio de imágenes accesible por el clúster (por ejemplo, Docker Hub).
RF13	El sistema permitirá desplegar las imágenes publicadas dentro del clúster mediante manifiestos Kubernetes (Deployment/Service).
RF14	El sistema permitirá exponer el servicio desplegado para acceso y verificación (por ejemplo, mediante NodePort o port-forward).

FASE 4 — DNS interno y alta disponibilidad básica (CoreDNS + resiliencia)

Código	Descripción del requisito funcional
RF15	El sistema permitirá la resolución de nombres interna entre servicios y pods mediante DNS del clúster.
RF16	El sistema permitirá mantener la disponibilidad de un servicio mediante réplicas (múltiples pods) controladas por Kubernetes.
RF17	El sistema permitirá simular la caída de un nodo worker y comprobar la reubicación automática de pods en otros nodos disponibles.

FASE 5 — Sistema de copias de seguridad y restauración (Velero + NFS)

Código	Descripción del requisito funcional
RF18	El sistema permitirá configurar un almacenamiento externo para backups accesible por el clúster (por ejemplo, un servidor NFS).
RF19	El sistema permitirá instalar y operar un sistema de backups para Kubernetes (Velero).
RF20	El sistema permitirá ejecutar copias de seguridad manuales de los recursos del clúster y/o namespaces seleccionados.
RF21	El sistema permitirá programar copias de seguridad automáticas con una periodicidad definida.
RF22	El sistema permitirá comprobar que las copias generadas se almacenan correctamente en el destino externo.
RF23	El sistema permitirá restaurar recursos del clúster a partir de una copia de seguridad disponible.
RF24	El sistema permitirá simular un incidente (borrado de un deployment/servicio/PVC según el alcance) y recuperar el estado usando restauración.

FASE 6 — Monitorización centralizada (Prometheus + Grafana)

Código	Descripción del requisito funcional
RF25	El sistema permitirá recolectar métricas del clúster y de los nodos mediante una solución de monitorización.
RF26	El sistema permitirá visualizar métricas en paneles gráficos (dashboards) accesibles por interfaz web.
RF27	El sistema permitirá supervisar indicadores de rendimiento (CPU, memoria, red, pods activos, etc.).
RF28	El sistema permitirá detectar cambios en el estado del clúster a través de métricas (por ejemplo, caída y recuperación de un nodo o pod).

FASE 7 — Seguridad y control de acceso (TLS + RBAC + NetworkPolicies)

Código	Descripción del requisito funcional
RF29	El sistema permitirá asegurar el acceso a los servicios de gestión mediante HTTPS/TLS.
RF30	El sistema permitirá definir roles y permisos para limitar acciones sobre recursos del clúster (RBAC).
RF31	El sistema permitirá aplicar restricciones de comunicación entre pods/namespaces mediante políticas de red (NetworkPolicies).
RF32	El sistema permitirá validar que un usuario sin permisos no puede ejecutar acciones críticas (por ejemplo, borrar recursos).

FASE 8 — Pruebas finales y documentación

Código	Descripción del requisito funcional
RF33	El sistema permitirá ejecutar pruebas de validación por escenarios (caída de nodos, restauración de backups, verificación de métricas y seguridad).
RF34	El sistema permitirá registrar evidencias del funcionamiento (capturas, logs y resultados de comandos) para la memoria del proyecto.
RF35	El sistema permitirá verificar criterios de éxito definidos (servicio accesible, backup restaurable, métricas disponibles y accesos controlados).

2.2. Requisitos no funcionales (RNF)

Los requisitos no funcionales definen **cómo debe comportarse el sistema KubeBackup360**. No describen funciones concretas, sino características de **calidad**, como rendimiento, seguridad, compatibilidad, disponibilidad, facilidad de uso y mantenibilidad. *Estos requisitos permiten establecer criterios objetivos de aceptación y sirven como guía durante el despliegue y las pruebas en un entorno de laboratorio académico.*

Código	Descripción del requisito no funcional
RNF1	El sistema deberá ejecutarse en un entorno de laboratorio local, sin exposición a Internet, utilizando una red interna entre las máquinas virtuales.
RNF2	El sistema deberá ser compatible con un equipo anfitrión con Windows 11 y recursos limitados, funcionando correctamente con 16 GB de memoria RAM y almacenamiento SSD.
RNF3	El sistema deberá mantener estabilidad durante la ejecución simultánea de varias máquinas virtuales, evitando bloqueos o caídas del entorno de laboratorio.
RNF4	El sistema deberá ofrecer una disponibilidad básica, de forma que los servicios se mantengan operativos ante la caída de pods individuales.
RNF5	La administración del clúster deberá poder realizarse desde un navegador web moderno dentro de la red local.
NF6	El sistema deberá ser escalable a nivel de laboratorio, permitiendo añadir o retirar nodos virtuales sin necesidad de rediseñar la arquitectura.
RNF7	Las copias de seguridad deberán almacenarse en un sistema externo al clúster, accesible en red local y disponible para procesos de restauración.
RNF8	El sistema deberá registrar información relevante mediante logs para facilitar la verificación de backups, restauraciones y pruebas de fallo.
RNF9	El sistema deberá utilizar tecnologías estándar y ampliamente documentadas del ecosistema Kubernetes, facilitando su comprensión y mantenimiento en un contexto educativo.
RNF10	El rendimiento del sistema deberá ser adecuado al entorno de laboratorio, priorizando la estabilidad y la funcionalidad frente a la alta carga o el uso intensivo de recursos.
RNF11	La configuración del sistema deberá ser clara y ordenada, organizada por fases, para facilitar su seguimiento, modificación y documentación.
RNF12	El sistema deberá permitir la recuperación de servicios y configuraciones en un tiempo razonable tras un fallo simulado, sin requerir procesos manuales complejos.

2.3. Restricciones

Las restricciones del proyecto definen las **condiciones y limitaciones bajo las cuales se desarrollará el sistema KubeBackup360**. Estas restricciones establecen el marco técnico, temporal y material del proyecto, y deben tenerse en cuenta durante todas las fases de diseño, implementación y validación.

Lenguajes o tecnologías obligatorias

El proyecto deberá desarrollarse utilizando tecnologías concretas, seleccionadas por su adecuación al entorno académico y por su uso habitual en entornos profesionales. Las tecnologías obligatorias del proyecto son:

- VirtualBox como plataforma de virtualización.
- Ubuntu Server 24.04 LTS como sistema operativo de las máquinas virtuales.
- Kubernetes en su distribución RKE2 como plataforma de orquestación de contenedores.
- Rancher como herramienta de gestión y administración visual del clúster.
- Docker para la creación de imágenes de contenedores personalizadas.
- Velero como herramienta de copias de seguridad y restauración del clúster.
- NFS como sistema de almacenamiento para las copias de seguridad.
- Prometheus y Grafana como herramientas de monitorización.
- kubectl y Helm como herramientas de administración y despliegue.

Recursos disponibles (tiempo, equipo, materiales)

El desarrollo del proyecto se realizará en un **entorno de laboratorio local**, utilizando un único equipo físico con las siguientes características aproximadas:

- Sistema operativo anfitrión: Windows 11.
- Memoria RAM disponible: 16 GB.
- Almacenamiento: SSD con espacio limitado para máquinas virtuales y copias de seguridad.
- Equipo humano: proyecto desarrollado a nivel académico, con recursos humanos limitados y aprendizaje progresivo.
- Tiempo de desarrollo: limitado al calendario académico, por lo que el proyecto se plantea de forma modular y por fases, permitiendo finalizarlo en la fase alcanzada.

Dependencias o limitaciones técnicas

El proyecto presenta las siguientes dependencias y limitaciones técnicas:

- Dependencia del correcto funcionamiento de la plataforma de virtualización (VirtualBox).
- Limitación de recursos de hardware, que condiciona el número de máquinas virtuales y servicios que pueden ejecutarse simultáneamente.
- Dependencia de la red local para la comunicación entre nodos y servicios, sin acceso directo a Internet en el entorno de laboratorio.
- Necesidad de compatibilidad entre las versiones de Kubernetes, Rancher y las herramientas asociadas.
- Posibles limitaciones derivadas del nivel de conocimiento técnico inicial, que condicionan el alcance de algunas fases avanzadas.

3. Análisis de usuarios y roles

El proyecto **KubeBackup360** se desarrolla en un **entorno académico y de laboratorio**, por lo que no está orientado a un uso multiusuario real. El sistema será gestionado **exclusivamente por el administrador**, y la existencia de otros usuarios se limita a **fines demostrativos**, con el objetivo de validar el funcionamiento del control de accesos y permisos.

Por este motivo, el proyecto contempla **únicamente dos roles**, directamente relacionados con su alcance real:

Roles definidos en el sistema

Rol	Descripción	Permisos principales
Administrador	Rol principal del sistema. Representa al administrador de sistemas o estudiante ASIR responsable de diseñar, desplegar, configurar y mantener toda la infraestructura.	Control total del clúster Kubernetes: gestión de nodos, despliegue y eliminación de servicios, configuración de copias de seguridad y restauraciones, acceso completo a monitorización y aplicación de políticas de seguridad.
Usuario	Rol secundario creado únicamente para **simular el uso del sistema** y demostrar el funcionamiento del control de accesos. No representa un usuario final real.	Acceso limitado a determinados recursos o servicios, como la consulta de información o el uso de aplicaciones desplegadas, sin permisos de administración ni modificación de la infraestructura.

Consideraciones del modelo de usuarios

- El **Administrador** es el único rol utilizado de forma continua durante el desarrollo del proyecto.

- El rol **Usuario** se implementa de forma **básica**, únicamente para demostrar el uso de **RBAC (Role-Based Access Control)**.

- No se contempla el rol de visitante ni un sistema complejo de gestión de usuarios, ya que **no forma parte de los objetivos del proyecto**.

- Esta simplificación permite mantener el proyecto enfocado en la **infraestructura, la seguridad y la administración del clúster**, sin añadir complejidad innecesaria.

4. Casos de uso / Escenarios de uso

Este apartado tiene como objetivo describir los **principales casos de uso del proyecto KubeBackup360**, mostrando el **producto final obtenido** y su **finalidad**.

El proyecto se centra en la implementación de un sistema de copias de seguridad en Kubernetes, capaz de proteger la configuración y los datos del clúster y de recuperar el sistema ante fallos simulados.

El resultado final es un **entorno Kubernetes de laboratorio**, funcional y controlado, que permite desplegar servicios, realizar copias de seguridad, restaurar el sistema tras incidencias y supervisar su estado. Todo ello se gestiona aplicando un **control de accesos básico por roles**, adecuado a un contexto académico.

Los casos de uso tienen como actor principal al **Administrador del sistema**, incorporando de forma puntual un **usuario con permisos limitados** para demostrar el funcionamiento de la seguridad por roles.

Caso de uso 1 — Despliegue y uso de servicios en el clúster

Elemento	Descripción
Código	CU1
Nombre	Uso de servicios desplegados en Kubernetes
Actor principal	Administrador del sistema
Descripción	El administrador utiliza el clúster Kubernetes para desplegar y ejecutar servicios de prueba, como un servidor web NGINX y, de forma opcional, una base de datos MariaDB, simulando una carga de trabajo real.
Resultado esperado	Servicios operativos y accesibles dentro del entorno de laboratorio.

Caso de uso 2 — Realización de copias de seguridad del sistema

Elemento	Descripción
Código	CU2
Nombre	Crear copia de seguridad del clúster
Actor principal	Administrador del sistema
Descripción	El administrador ejecuta una copia de seguridad del clúster Kubernetes, incluyendo la configuración y los servicios desplegados, utilizando el sistema de backups integrado.
Resultado esperado	Copia de seguridad creada correctamente y almacenada en el sistema externo definido.

Caso de uso 3 — Recuperación del sistema tras un fallo

Elemento	Descripción
Código	CU3
Nombre	Restaurar el clúster desde una copia de seguridad
Actor principal	Administrador del sistema
Descripción	El administrador simula un fallo eliminando un servicio o recurso del clúster y utiliza una copia de seguridad previa para restaurar el sistema a su estado funcional.
Resultado esperado	Servicio restaurado correctamente y clúster operativo tras la recuperación.

Caso de uso 4 — Supervisión del estado del clúster

Elemento	Descripción
Código	CU4
Nombre	Monitorizar el estado del sistema
Actor principal	Administrador del sistema
Descripción	El administrador consulta el estado del clúster mediante herramientas de monitorización, visualizando métricas de uso de recursos y el estado de los servicios.
Resultado esperado	Información clara y actualizada del estado del clúster y sus servicios.

Caso de uso 5 — Control de acceso por roles

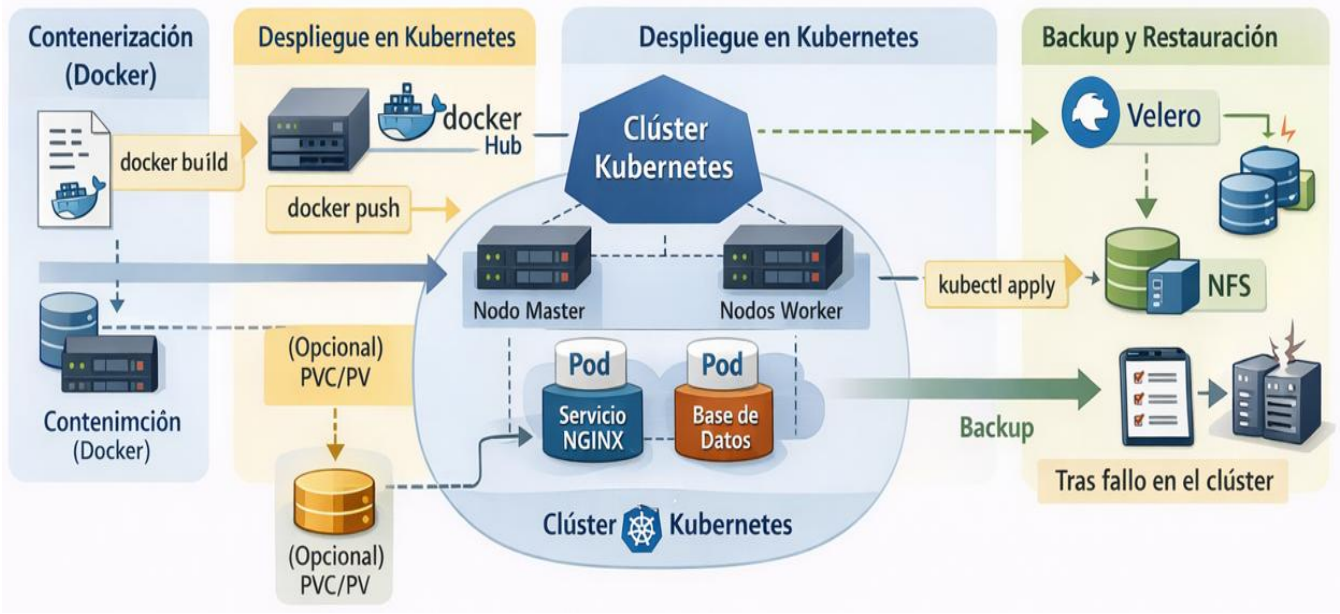
Elemento	Descripción
Código	CU5
Nombre	Validar seguridad mediante roles
Actor principal	Administrador del sistema / Usuario
Descripción	El administrador crea un usuario con permisos limitados y comprueba que este solo puede acceder a los recursos permitidos, sin capacidad de modificar la infraestructura.
Resultado esperado	Acceso restringido correctamente según el rol asignado, validando el control de permisos del sistema.

5. Modelo de datos o estructura de la información

En el proyecto KubeBackup360, el modelo de datos no se basa en una aplicación con una base de datos centralizada, sino en la estructura de información que se genera y gestiona a lo largo del ciclo de vida de los contenedores y del clúster Kubernetes.

Este modelo describe qué información existe, cómo se organiza y qué elementos son objeto de copia de seguridad y restauración.

El sistema integra tecnologías de contenedorización (Docker), orquestación (Kubernetes) y protección de datos (Velero), por lo que el modelo de datos se entiende como un flujo estructurado de información, desde la creación de imágenes hasta la recuperación tras un fallo.



5.1 Entidades principales del sistema

1. Imágenes de contenedores (Docker)

Representan las aplicaciones empaquetadas listas para ser ejecutadas.

Se crean a partir de un *Dockerfile* y contienen todo lo necesario para ejecutar un servicio (por ejemplo, NGINX o MariaDB).

2. Registro de imágenes (Docker Hub)

Almacén externo donde se publican las imágenes Docker creadas.

Kubernetes obtiene las imágenes desde este registro para desplegar los servicios en el clúster.

3. Clúster Kubernetes (RKE2)

Entidad central del sistema. Gestiona y coordina todos los recursos necesarios para ejecutar los contenedores y servicios.

4. Nodos del clúster (Master y Workers)

- El nodo master controla el estado y la planificación del clúster.
- Los nodos worker ejecutan los pods que contienen los servicios.

5. Manifiestos de Kubernetes (YAML)

Archivos de configuración que describen cómo deben desplegarse los servicios (Deployments, Services, StatefulSets).

Estos manifiestos conectan las imágenes Docker con el clúster.

6. Pods

Unidad mínima de ejecución en Kubernetes. Cada pod ejecuta uno o varios contenedores basados en imágenes Docker.

7. Servicios (Services)

Permiten acceder a los pods de forma estable, independientemente de que los pods se creen o eliminen dinámicamente.

8. Almacenamiento persistente (PV y PVC)

Elementos encargados de mantener datos de forma persistente, especialmente importantes para servicios con estado, como bases de datos.

Estos datos forman parte del contenido crítico a respaldar.

9. Copias de seguridad (Backups con Velero)

Conjunto de datos y configuraciones del clúster que se almacenan de forma segura en un sistema externo (NFS).

10. Restauraciones (Restore)

Proceso mediante el cual se recupera el estado del sistema utilizando una copia de seguridad previa, tras un fallo o eliminación accidental.

11. Usuarios y roles (RBAC)

Definen quién puede acceder y qué acciones puede realizar sobre el sistema, diferenciando entre administrador y usuario con permisos limitados.

5.2 Relación entre entidades (flujo de la información)

- El administrador crea imágenes Docker a partir de un Dockerfile.

- Las imágenes se publican en Docker Hub como repositorio externo.

- Kubernetes descarga las imágenes desde el registro y las ejecuta en los pods de los nodos worker.

- Los pods ofrecen servicios accesibles mediante Services.

- Algunos servicios utilizan almacenamiento persistente (PVC/PV) para conservar datos.

- Velero realiza copias de seguridad del estado del clúster y de los recursos asociados.

- Las copias se almacenan en un servidor NFS externo.

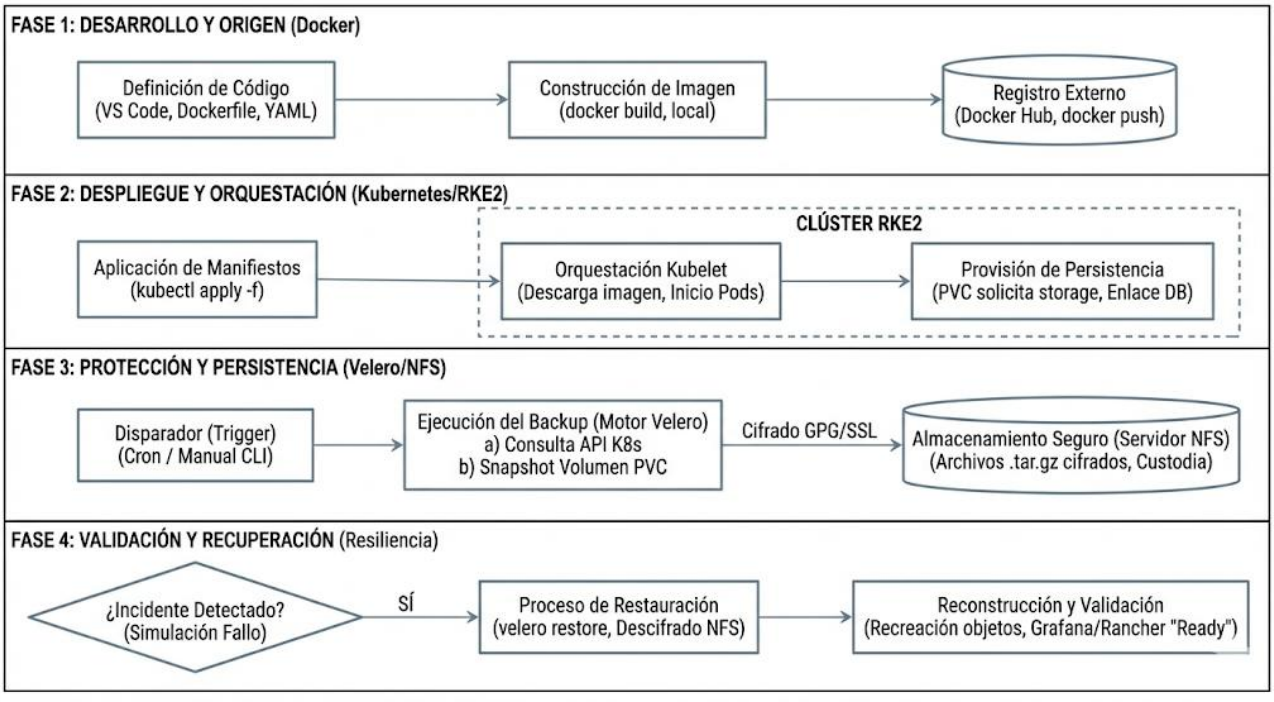
- Ante un fallo, el sistema puede restaurarse utilizando las copias de seguridad.

- El acceso y las operaciones están controladas mediante roles y permisos (RBAC).

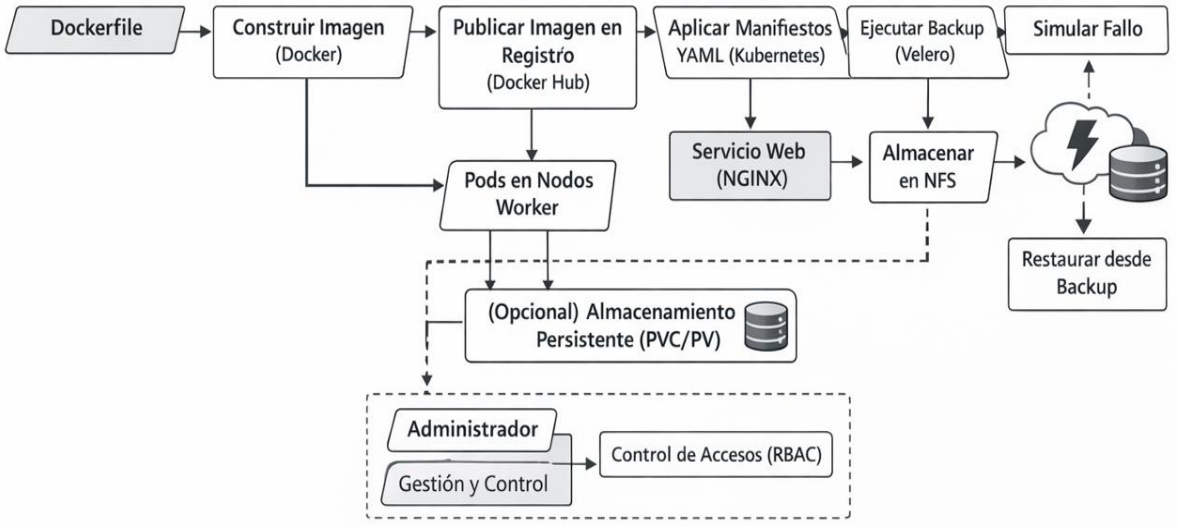
5.3 Modelo conceptual resumido

KubeBackup360 gestiona un flujo completo de información que comienza con la creación de contenedores Docker y finaliza con la restauración del sistema tras un fallo. Las imágenes Docker constituyen la base de los servicios, Kubernetes se encarga de su ejecución y Velero protege tanto la configuración como los datos persistentes. Todo el sistema se administra mediante roles, garantizando un control de acceso adecuado en un entorno académico de laboratorio.

Diagrama de flujo del ciclo de vida operativo y de resiliencia del sistema KubeBackup360.



Ciclo operativo de desarrollo, despliegue y protección en KubeBackup360.



6. Diseño de la interfaz

El proyecto KubeBackup360 no contempla el desarrollo de pantallas o interfaces gráficas propias, ya que su finalidad no es la creación de una aplicación software, sino el diseño e implementación de un sistema de copias de seguridad en Kubernetes dentro de un entorno académico de laboratorio.

En proyectos de administración de sistemas e infraestructuras, **como es el caso de KubeBackup360**, la interacción con el sistema se realiza habitualmente mediante herramientas profesionales ya existentes, ampliamente utilizadas en entornos reales.

El desarrollo de una interfaz propia no aportaría valor adicional al objetivo del proyecto y supondría una complejidad innecesaria, alejándose del enfoque formativo del ciclo de Administración de Sistemas Informáticos en Red (ASIR).

Por este motivo, el diseño de la interfaz del sistema se basa en el uso combinado de interfaces web de gestión y monitorización, junto con interfaces de línea de comandos, que permiten administrar, supervisar y validar el correcto funcionamiento del clúster.

El acceso a todas las interfaces se realiza en un entorno local, utilizando port-forward, por ser la opción más sencilla, segura y adecuada a un proyecto académico de FP.

6.1 Interfaz de gestión del clúster (Rancher)

Nombre de la interfaz: Panel de gestión del clúster (Rancher UI).

Funcionalidad principal:

Proporcionar una visión global del estado del clúster Kubernetes, permitiendo al administrador supervisar nodos, pods, namespaces y servicios de forma visual y centralizada.

Uso dentro del proyecto:

- Visualización del estado general del clúster.
- Gestión básica de recursos desplegados.
- Verificación del control de accesos por roles (RBAC).

Casos de uso relacionados:
- CU1: Uso de servicios desplegados en Kubernetes.
- CU5: Validación de la seguridad mediante roles.

Acceso:
Mediante port-forward desde el entorno local del laboratorio.

6.2 Interfaz de monitorización (Grafana)

Nombre de la interfaz: Panel de monitorización del sistema.

Funcionalidad principal:

Visualizar métricas de rendimiento del clúster, como el uso de CPU, memoria y el estado de los nodos, permitiendo comprobar la estabilidad del sistema y su comportamiento tras operaciones de backup y restauración.

Uso dentro del proyecto:
- Supervisión del estado del clúster.
- Comprobación del funcionamiento del sistema tras fallos simulados.
- Validación visual del correcto rendimiento general.

Casos de uso relacionados:
- CU4: Supervisión del estado del clúster.

Acceso: Mediante port-forward desde el entorno local.

6.3 Interfaz de línea de comandos (CLI)

Nombre de la interfaz: Consola de administración (kubectl y Velero).

Funcionalidad principal:

Permitir al administrador realizar tareas técnicas avanzadas que requieren un mayor nivel de control y precisión.

Uso dentro del proyecto:
- Despliegue de servicios mediante archivos YAML.
- Ejecución de copias de seguridad del clúster.
- Restauración del sistema tras fallos o eliminaciones accidentales.

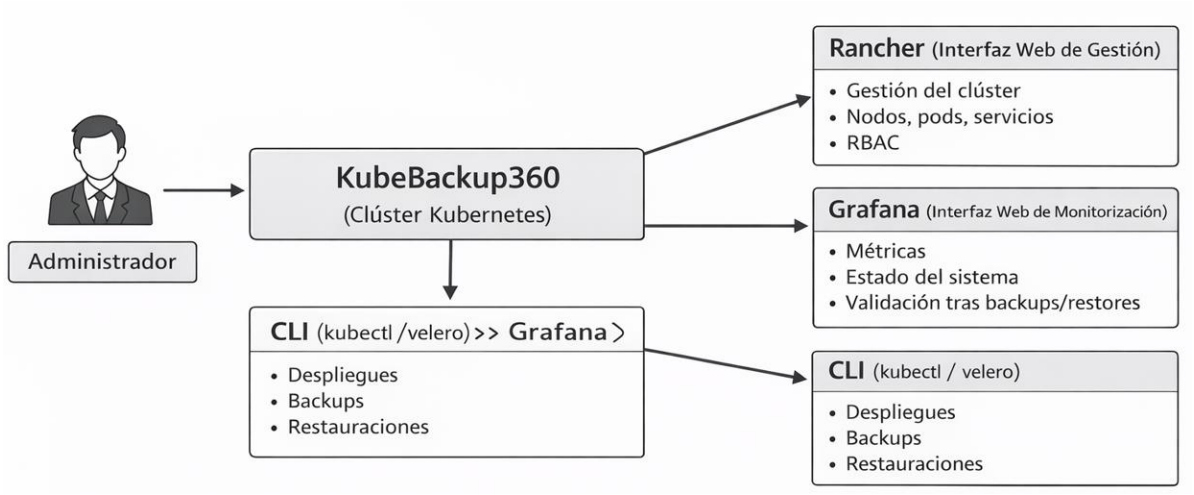
Casos de uso relacionados:
- CU2: Realización de copias de seguridad.
- CU3: Recuperación del sistema tras un fallo.

6.4 Flujo de interacción del administrador

El uso del sistema sigue un flujo operativo claro y ordenado:

- El administrador utiliza Rancher para la gestión visual y supervisión diaria del clúster.
- Consulta Grafana para verificar el estado y rendimiento del sistema.
- Accede a la CLI cuando es necesario realizar operaciones técnicas específicas, como backups o restauraciones.

Interacción del administrador con las interfaces de gestión, monitorización y control del sistema KubeBackup360



7. Planificación técnica

El objetivo de este apartado es planificar el desarrollo del proyecto **KubeBackup360**, definiendo las **tecnologías y herramientas utilizadas**, la **organización del trabajo en grupo** y un **cronograma realista** que permita completar el proyecto dentro del plazo establecido.
El proyecto se desarrolla por **un grupo de dos personas** y está planteado para estar **completamente finalizado antes de la revisión de finales de abril**, funcionando como una entrega completa previa a la entrega definitiva.

7.1 Lenguajes, tecnologías y frameworks

Al tratarse de un proyecto de **infraestructura y administración de sistemas**, no se utilizan lenguajes de programación ni frameworks de desarrollo de aplicaciones tradicionales. Las tecnologías empleadas son las propias del ecosistema de contenedores y Kubernetes:

- **Docker**: creación de imágenes de contenedores mediante Dockerfile.
- **Kubernetes (RKE2)**: orquestación de contenedores y gestión del clúster.
- **YAML**: definición de recursos Kubernetes (Deployments, Services, PVC, etc.).
- **Bash**: uso básico para la ejecución de comandos y administración del sistema.
- **Helm**: uso básico para el despliegue de herramientas como Rancher, Grafana o Velero.

No se emplean frameworks de desarrollo web o de aplicaciones, ya que el proyecto no consiste en una aplicación software, sino en una infraestructura técnica.

7.2 Base de datos

El proyecto no requiere una base de datos propia para su funcionamiento.

No obstante, se contempla el uso de **MariaDB de forma opcional**, únicamente como **servicio de prueba**, con el objetivo de simular un sistema con datos persistentes y demostrar el funcionamiento de las copias de seguridad y restauración.

Además, se utilizará un **servicio web NGINX** como aplicación principal de demostración.

7.3 Herramientas de diseño, desarrollo y documentación

Durante el desarrollo del proyecto se emplean las siguientes herramientas:

- **Visual Studio Code (VS Code)**: edición de archivos YAML, Dockerfile y documentación técnica.
- **draw.io**: creación de diagramas y esquemas
- **Microsoft Works**: redacción de la memoria del proyecto.
- **GitHub**: repositorio Git para el control de versiones del proyecto, de forma básica y colaborativa.
- **VirtualBox**: plataforma de virtualización para el entorno de laboratorio.

El uso de GitHub permite mantener un histórico de cambios y facilitar el trabajo en grupo, aunque no se adopta una metodología avanzada de control de versiones.

7.4 Reparto de tareas (trabajo en grupo)

El proyecto KubeBackup360 está desarrollado por tres personas, que trabajan de forma conjunta en todas las fases del proyecto, con responsabilidades principales diferenciadas para facilitar la organización y el seguimiento del trabajo.

A continuación, se detalla el reparto principal de tareas:

Luis Alfredo, Responsable principal de:

- Preparación del entorno de virtualización.
- Instalación y configuración del clúster Kubernetes (RKE2).
- Contenerización de servicios mediante Docker.
- Implementación del sistema de copias de seguridad y restauración (Velero y NFS).

Clara Inés, Responsable principal de:

- Documentación del proyecto.
- Elaboración de diagramas, esquemas e infografías.
- Implementación y validación de la monitorización del sistema (Prometheus y Grafana).
- Apoyo en la validación final del proyecto.

Ilias Brahimi, Responsable principal de:

- Apoyo en la configuración y gestión del clúster Kubernetes.
- Verificación del despliegue de servicios y pruebas funcionales.
- Colaboración en tareas de seguridad básica (roles y permisos).
- Soporte en pruebas de fallos y restauraciones.

Todos los miembros del grupo participan en las distintas fases del proyecto, colaborando, revisando y validando conjuntamente los resultados obtenidos, lo que garantiza una comprensión global del sistema y un desarrollo equilibrado del proyecto.

7.5 CRONOGRAMA DEL PROYECTO

El proyecto **KubeBackup360** se desarrolla entre el **19 de enero de 2026** y el **8 de mayo de 2026**.

La organización del cronograma está diseñada para el trabajo en paralelo de tres integrantes, optimizando el tiempo y garantizando que cada fase técnica cuente con su correspondiente validación y documentación técnica.

- Fecha de inicio: 19 de enero de 2026.
- Fecha objetivo de finalización: 8 de mayo de 2026.
- Ritmo de trabajo: Lunes a viernes (jornada técnica) con soporte documental en fines de semana.

DISTRIBUCIÓN TEMPORAL Y REPARTO DE TRABAJO

FASE 1: INFRAESTRUCTURA Y CLÚSTER BASE (ENERO)

Objetivo: Disponer de la infraestructura base operativa y conectividad total entre nodos.

Luis Alfredo:
- Creación y configuración de las máquinas virtuales (Master y Workers) en VirtualBox.
- Instalación y configuración de los sistemas operativos Ubuntu Server 24.04.
- Despliegue inicial del clúster Kubernetes mediante RKE2.

Ilias Brahimi:
- Configuración de la red interna y direccionamiento IP estático para los nodos.
- Verificación de la conectividad SSH y estado de salud de los servicios de sistema.
- Pruebas iniciales de validación de los pre-requisitos de Kubernetes.

Clara Inés:
- Inicio de la documentación técnica y funcional del proyecto.
- Elaboración de los diagramas de arquitectura de red y topología del clúster.
- Redacción de los apartados de introducción, contexto y análisis de requisitos.

OBJETIVO CLAVE: Infraestructura virtual estable y nodos en estado "Ready".

FASE 2: GESTIÓN Y DESPLIEGUE DE SERVICIOS (FEBRERO)

Objetivo: Clúster gestionable mediante interfaz web y despliegue de cargas de trabajo.

Luis Alfredo:
- Instalación de Rancher como panel de gestión visual centralizado.
- Creación de imágenes Docker personalizadas para los servicios web (NGINX) y bases de datos.
- Publicación de imágenes en el registro externo (Docker Hub).

Ilias Brahimi:
- Unión de nodos trabajadores al clúster y balanceo de carga básico.
- Verificación del acceso a servicios desplegados y pruebas de estabilidad de red.
- Configuración de accesos externos mediante port-forwarding.

Clara Inés:

- Documentación técnica de los procesos de contenedorización y despliegue.
- Actualización de los esquemas de flujo de datos y persistencia.
- Redacción de los manuales de usuario para la interfaz de Rancher.

OBJETIVO CLAVE: Clúster gestionado visualmente y aplicaciones operativas con persistencia.

FASE 3: PROTECCIÓN Y MONITORIZACIÓN (MARZO)

Objetivo: Núcleo funcional de backup y observabilidad operativa al 100%.

Luis Alfredo:

- Configuración del servidor NFS externo para almacenamiento de copias.
- Instalación y configuración de Velero para la gestión de backups.
- Implementación de rutinas de backup automatizadas (Schedules).

Ilias Brahimi:

- Ejecución de simulacros de fallo y validación de los procesos de restauración.
- Pruebas de resiliencia del almacenamiento ante caídas de nodos.
- Auditoría de integridad de los datos recuperados.

Clara Inés:

- Despliegue y configuración del stack de monitorización (Prometheus y Grafana).
- Diseño de dashboards personalizados para métricas de clúster y estado de backups.
- Documentación de los resultados de las pruebas de desastre y recuperación.

OBJETIVO CLAVE: Plan de recuperación ante desastres (DRP) validado y sistema monitorizado.

FASE 4: VALIDACIÓN FINAL Y SEGURIDAD (ABRIL / MAYO)

Objetivo: Sistema securizado, validación técnica global y defensa del proyecto.

Luis Alfredo:

- Implementación de certificados TLS y configuración avanzada de seguridad.
- Ajustes finales de rendimiento en el clúster y limpieza de recursos.

Ilias Brahimi:

- Validación de las políticas de red (NetworkPolicies) y control de accesos (RBAC).
- Revisión funcional completa de todos los casos de uso del proyecto.

Clara Inés:

- Cierre de la memoria técnica e integración de todas las evidencias (capturas, logs).
- Preparación del material visual y guion para la defensa final del proyecto.

OBJETIVO CLAVE: Proyecto KubeBackup360 finalizado y listo para la defensa.

CONSIDERACIONES FINALES DEL CRONOGRAMA

- **Paralelismo**: El reparto de tareas entre tres integrantes permite cubrir simultáneamente la infraestructura técnica, la validación de red y la documentación continua.

- **Resiliencia**: El cronograma incluye márgenes específicos para la corrección de errores e imprevistos técnicos propios de entornos Kubernetes.

- **Madurez**: El sistema se considera funcionalmente completo al finalizar marzo, dedicando abril exclusivamente a la securización avanzada y validación final.

Diagrama de Gantt – Planificación técnica
Proyecto: KubeBackup360
Periodo: 19 de enero – 8 de mayo
Planificación: semanal, de lunes a viernes
Nota: Se parte de VirtualBox instalado y OVAs disponibles.

PROYECTO KUBEBACKUP360

FASE 1: INFRAESTRUCTURA Y COMUNICACIONES

MES: ENERO

SEMANA 1: Despliegue del entorno virtual (19 – 23 enero)

- LUNES: Planificación de recursos (CPU/RAM) y creación de la VM "Base" (Ubuntu 24.04).
- MARTES: Clonación de máquinas para nodos Master y Workers (1 Master + 2 Workers).
- MIÉRCOLES: Configuración de red interna y asignación de direccionamiento IP estático.
- JUEVES: Instalación de OpenSSH-server y configuración de usuarios de sistema.
- VIERNES: Pruebas de conectividad (ping/ssh) y documentación de la topología.
- **OBJETIVO CLAVE 1:** Entorno de virtualización estable y red configurada.

SEMANA 2: Preparación del sistema operativo (26 – 30 enero)

- LUNES: Desactivación de memoria Swap y ajuste de módulos del kernel para Kubernetes.
- MARTES: Configuración de reglas de firewall (UFW) para puertos de RKE2.
- MIÉRCOLES: Configuración de NTP y resolución de nombres local en el archivo /etc/hosts.
- JUEVES: Creación del repositorio en GitHub y subida de los archivos de configuración inicial.
- VIERNES: Validación técnica de pre-requisitos antes de la instalación del clúster.

FASE 2: ORQUESTACIÓN Y GESTIÓN WEB

MES: FEBRERO

SEMANA 3: Despliegue del clúster RKE2 (2 – 6 febrero)

- LUNES: Instalación y configuración del servicio rke2-server en el nodo Master.
- MARTES: Extracción del token de registro y verificación de logs del plano de control.
- MIÉRCOLES: Unión del primer nodo Worker al clúster y validación de estado.
- JUEVES: Unión del segundo nodo Worker y verificación global mediante kubectl.
- VIERNES: Instalación de herramientas cliente y obtención del archivo kubeconfig.
- **OBJETIVO CLAVE 2:** Clúster Kubernetes (RKE2) operativo con nodos en estado "Ready".

SEMANA 4: Gestión visual con Rancher (9 – 13 febrero)

- LUNES: Instalación del gestor de paquetes **Helm** en el equipo de administración.
- MARTES: Configuración de repositorios Helm (Rancher y Cert-manager).
- MIÉRCOLES: Instalación de Rancher mediante Helm y configuración de certificados.
- JUEVES: Acceso a la interfaz web de Rancher mediante Port-forwarding.
- VIERNES: Creación de Namespaces de trabajo y validación de RBAC básico.
- **OBJETIVO CLAVE 3:** Panel de gestión visual (Rancher UI) accesible y funcional.

FASE 3: CARGAS DE TRABAJO Y PERSISTENCIA

MES: FEBRERO / MARZO

SEMANA 5: Contenedores y aplicaciones (16 – 20 febrero)

- LUNES: Creación del Dockerfile para el servicio web personalizado (NGINX).
- MARTES: Construcción y etiquetado de la imagen local con Docker.
- MIÉRCOLES: Inicio de sesión y publicación de la imagen en Docker Hub.
- JUEVES: Creación de manifiestos YAML (Deployment y Service) para la aplicación.
- VIERNES: Despliegue de la aplicación y comprobación de acceso vía NodePort.

SEMANA 6: Persistencia de datos (23 – 27 febrero)

- LUNES: Definición técnica de StorageClasses y almacenamiento local.
- MARTES: Creación de objetos **Persistent Volume Claim (PVC)** para la base de datos.
- MIÉRCOLES: Despliegue de MariaDB/MySQL vinculado al almacenamiento persistente.
- JUEVES: Inyección de datos de prueba y validación de persistencia tras reinicio de Pods.
- VIERNES: Documentación técnica del flujo de datos y persistencia.
- **OBJETIVO CLAVE 4:** Aplicaciones propias con persistencia de datos activas.

FASE 4: PROTECCIÓN DE DATOS (NÚCLEO DEL PROYECTO)

MES: MARZO

SEMANA 7: Almacenamiento externo NFS (2 – 6 marzo)

- LUNES: Instalación del servicio NFS-kernel-server en el nodo externo.
- MARTES: Creación del directorio de respaldo y configuración de permisos de red.
- MIÉRCOLES: Montaje de prueba de la unidad NFS en los nodos del clúster.
- JUEVES: Aseguramiento del volumen NFS (políticas de acceso y red).
- VIERNES: Validación del almacenamiento centralizado para los backups.

SEMANA 8: Implementación de Velero (9 – 13 marzo)

- LUNES: Instalación del CLI de Velero y despliegue del servidor Velero en el clúster.
- MARTES: Configuración del repositorio de backup apuntando al destino NFS.
- MIÉRCOLES: Realización de la primera copia de seguridad manual (Full Backup).
- JUEVES: Creación de copias selectivas por Namespace y recursos específicos.
- VIERNES: Programación de tareas automáticas (**Schedules**) para backups periódicos.
- **OBJETIVO CLAVE 5:** Sistema de copias de seguridad automatizado y almacenado en NFS.

SEMANA 9: Pruebas de restauración ante desastres (16 – 20 marzo)

- LUNES: Simulación de fallo 1: Borrado de despliegues y servicios de aplicación.
- MARTES: Ejecución de restauración con Velero y verificación de reconstrucción de objetos.
- MIÉRCOLES: Simulación de fallo 2: Corrupción de datos en la base de datos MariaDB.
- JUEVES: Restauración de volúmenes persistentes (PVC) y validación de datos.
- VIERNES: Informe final de pruebas de resiliencia y resguardo de evidencias.

FASE 5: MONITORIZACIÓN Y SEGURIDAD

MES: ABRIL

SEMANA 10: Observabilidad con Prometheus y Grafana (23 – 27 marzo)

- LUNES: Instalación de Prometheus para la recolección de métricas del clúster.
- MARTES: Instalación de Grafana mediante Helm y configuración de persistencia.
- MIÉRCOLES: Conexión de Grafana con la base de datos de Prometheus.

- JUEVES: Importación de Dashboards técnicos para supervisión de nodos y pods.
- VIERNES: Creación de un panel específico para monitorizar el estado de los backups.

SEMANA 11: Alertas y validación final (30 marzo – 3 abril)

- LUNES: Definición de umbrales críticos para CPU, RAM y almacenamiento.
- MARTES: Configuración de reglas de alerta visuales en Grafana.
- MIÉRCOLES: Pruebas de validación de seguridad (TLS) y certificados en los servicios.
- JUEVES: Auditoría de roles **RBAC** (Administrador vs. Usuario limitado).
- VIERNES: Cierre de la configuración técnica y recopilación de métricas finales.
- **OBJETIVO CLAVE 6:** Infraestructura totalmente monitorizada y securizada.

FASE 6: CIERRE TÉCNICO Y DEFENSA

MES: ABRIL / MAYO

SEMANA 12: Consolidación y memoria técnica (13 – 17 abril)

- LUNES: Revisión global de scripts Bash y manifiestos YAML en GitHub.
- MARTES: Redacción del manual de instalación y configuración para el administrador.
- MIÉRCOLES: Elaboración de la guía de recuperación ante desastres.
- JUEVES: Pruebas finales de integración de todos los componentes.
- VIERNES: Entrega de la versión preliminar de la documentación técnica.

SEMANA FINAL: Preparación de la defensa (4 – 8 mayo)

- LUNES: Revisión final de la memoria del proyecto y corrección de estilo.
- MARTES: Diseño de la presentación visual para la defensa.
- MIÉRCOLES: Ensayo de la exposición oral y control de tiempos.
- JUEVES: Puesta a punto del laboratorio (VirtualBox) para la demostración en vivo.
- VIERNES: **FINALIZACIÓN DEL PROYECTO KUBEBACKUP360**.

8. Análisis de riesgos

En este apartado identificamos los **riesgos principales** que, como grupo de estudiantes, creemos que pueden afectar al desarrollo del proyecto **KubeBackup360**, así como la forma en la que **asumimos y afrontamos** dichas situaciones dentro de un contexto académico de FP.
Al tratarse de un proyecto de laboratorio y no de un entorno productivo real, somos conscientes de que existen **limitaciones técnicas y organizativas**, las cuales se consideran parte del aprendizaje.

8.1 Identificación de riesgos

Riesgo 1: Desarrollo en paralelo en equipos distintos
Creemos que uno de los principales riesgos del proyecto es que cada integrante trabaje en su propio ordenador, sin un servidor o entorno común. Esto puede provocar diferencias de configuración entre los entornos de trabajo.
Para minimizar este riesgo, definimos una configuración base común y compartimos los archivos clave del proyecto (manifiestos YAML, Dockerfile y documentación) mediante un repositorio Git. En caso de problemas, asumimos volver a un punto documentado y repetir el procedimiento para asegurar que el entorno sea reproducible.

Riesgo 2: Problemas técnicos en Kubernetes

Consideramos posible que aparezcan errores durante la instalación o configuración del clúster Kubernetes, especialmente en la unión de nodos o en la comunicación entre ellos.
Ante esta situación, asumimos rehacer la fase afectada, revisar la configuración aplicada y simplificar temporalmente el escenario si es necesario. En caso de bloqueo, contemplamos solicitar orientación al profesorado.

Riesgo 3: Fallos en el sistema de copias de seguridad

Entendemos que pueden producirse problemas en la configuración del almacenamiento NFS o en el funcionamiento de Velero, provocando copias de seguridad incompletas o restauraciones fallidas.
Para reducir este riesgo, definimos pruebas progresivas y validaciones frecuentes. Si se detectan errores, asumimos repetir la configuración y realizar pruebas con servicios simples antes de avanzar.

Riesgo 4: Dependencia de servicios externos

Somos conscientes de que el uso de servicios externos como Docker Hub puede generar incidencias puntuales (limitaciones, errores de acceso o descargas).
En estos casos, asumimos adaptar temporalmente los despliegues y utilizar imágenes públicas conocidas para continuar con el desarrollo del proyecto.

Riesgo 5: Riesgos asumidos por el carácter académico del proyecto

Al tratarse de un anteproyecto académico, asumimos conscientemente una serie de limitaciones:

- No se implementa alta disponibilidad real.
- Los backups se limitan al entorno de laboratorio.
- La automatización es básica.
- La seguridad se implementa de forma sencilla mediante RBAC.

Consideramos que estas decisiones son coherentes con el nivel del ciclo formativo y no comprometen el objetivo principal del proyecto.

Conclusión del análisis de riesgos

Como grupo, consideramos que la identificación de estos riesgos nos permite anticiparnos a posibles problemas y actuar de forma organizada. Asumimos que, ante cualquier incidencia, la estrategia será volver a un punto documentado, rehacer la fase afectada, simplificar el alcance si es necesario y solicitar orientación al profesorado.

Este enfoque nos permite garantizar que el proyecto **KubeBackup360** sea funcional, realista y defendible dentro del marco académico de FP.

8.2 Valoración y respuesta

En este apartado valoramos cada uno de los riesgos identificados en función de su **probabilidad** y **impacto**, y definimos el **plan de prevención o contingencia** que, como grupo de estudiantes, aplicaremos en caso de que el riesgo se materialice.

Riesgo	Probabilidad	Impacto	Plan de prevención o contingencia
Trabajo en paralelo en equipos distintos	Media	Media	Definimos una configuración base común, compartimos los archivos clave mediante Gitlab y documentamos los pasos críticos. Si aparecen diferencias, volvemos a un punto documentado y repetimos el procedimiento para igualar entornos.
Problemas técnicos en Kubernetes (RKE2)	Media	Alta	Probamos cada fase de forma progresiva y documentamos los comandos utilizados. Si surge un error, asumimos rehacer la fase afectada o simplificar el escenario. En caso de bloqueo, solicitamos orientación al profesor.
Fallos en el sistema de copias de seguridad (Velero + NFS)	Media	Alta	Realizamos pruebas controladas de backup y restore con servicios simples antes de avanzar. Si una restauración falla, repetimos la configuración y validamos paso a paso hasta obtener un resultado reproducible.
Dependencia de servicios externos (Docker Hub)	Baja	Media	Utilizamos imágenes públicas y sencillas y verificamos previamente el acceso. Si hay incidencias, asumimos adaptar temporalmente los despliegues usando imágenes alternativas para continuar el proyecto.
Limitaciones asumidas por el carácter académico del proyecto	Alta	Media	Aceptamos conscientemente estas limitaciones (sin alta disponibilidad real, automatización básica y seguridad simple). Definimos claramente el alcance del proyecto y priorizamos demostrar correctamente la funcionalidad principal.

9. Validación y criterios de éxito

El objetivo de este apartado es definir **cómo validaremos que el proyecto KubeBackup360 funciona correctamente** y establecer los **criterios que permitirán considerar el proyecto como exitoso**.

La validación se basa en comprobar que el sistema cumple los objetivos definidos, dentro de las limitaciones propias de un entorno académico y de laboratorio.

9.1 Criterios de aceptación

Consideramos que el proyecto cumple los requisitos y puede darse por válido si se cumplen los siguientes criterios de aceptación:

- El clúster Kubernetes se despliega correctamente y permanece operativo.
- Los nodos master y worker se encuentran en estado funcional.
- Es posible desplegar al menos un servicio contenedorizado (NGINX) en el clúster.
- El sistema de copias de seguridad permite realizar backups del entorno Kubernetes.
- Es posible restaurar servicios y configuraciones tras un fallo simulado.
- El acceso y la gestión del sistema se realizan mediante las interfaces definidas (Rancher y CLI).
- Se implementa un control de accesos básico por roles, diferenciando al administrador de un usuario con permisos limitados.
- El sistema puede validarse y reproducirse en un entorno de laboratorio.

9.2 Pruebas previstas

Para comprobar el cumplimiento de los criterios de aceptación, se realizarán las siguientes pruebas:

Pruebas funcionales

- Verificación del estado del clúster Kubernetes tras su despliegue.
- Despliegue y acceso correcto a servicios contenedorizados.
- Ejecución manual de copias de seguridad mediante Velero.
- Restauración del sistema tras la eliminación intencionada de recursos.
- Comprobación de la persistencia de datos en los servicios que la utilicen.

Pruebas de usuario

- Acceso del administrador al clúster mediante Rancher.
- Gestión básica de recursos desde la interfaz web.
- Verificación de las restricciones de acceso para usuarios con permisos limitados.
- Uso de la CLI para tareas de administración y validación.

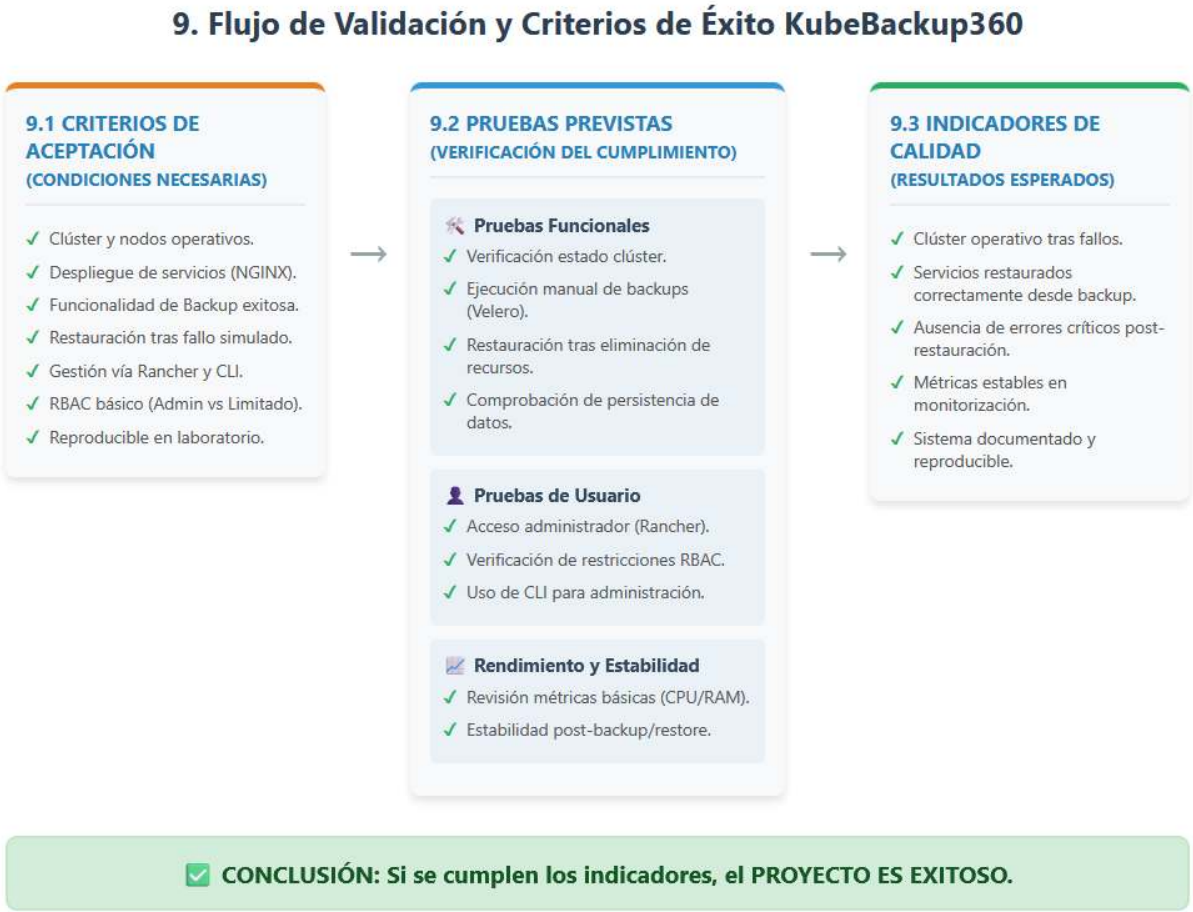
Pruebas de rendimiento y estabilidad

- Comprobación del funcionamiento estable del sistema durante periodos de uso.
- Revisión de métricas básicas de CPU y memoria mediante herramientas de monitorización.
- Validación del comportamiento del sistema tras operaciones de backup y restauración.

9.3 Indicadores de calidad y resultados esperados

Los siguientes indicadores permitirán confirmar que el proyecto ha sido desarrollado con éxito:

- El clúster Kubernetes se mantiene operativo tras fallos simulados.
- Los servicios desplegados se restauran correctamente a partir de una copia de seguridad.
- No se detectan errores críticos tras la restauración del sistema.
- Las métricas de monitorización reflejan un comportamiento estable del entorno.
- El sistema es comprensible, documentado y reproducible por los integrantes del grupo.
- El proyecto cumple su objetivo principal: **demostrar un sistema funcional de copias de seguridad y recuperación en Kubernetes**.



11. Síntesis esquemática del proyecto y guía de desarrollo por fases

Este apartado presenta una **visión estructurada y operativa del proyecto KubeBackup360**, organizada en **fases de desarrollo**.

Para cada fase se indican:

- Las **herramientas principales** que se utilizarán.
- Las **actividades a realizar** (en forma de checklist).
- El **resultado esperado** al finalizar la fase.

Este esquema servirá como **guía práctica** durante la ejecución del proyecto, permitiéndonos avanzar de forma ordenada y comprobar en todo momento el estado del desarrollo.

FASE 1 – Preparación del entorno de laboratorio

Herramientas utilizadas:

- VirtualBox
- OVAs de Ubuntu Server / Desktop
- SSH

Actividades:

- Definir la arquitectura del laboratorio (master y workers).
- Importar las OVAs necesarias en VirtualBox.
- Configurar la red interna entre las máquinas virtuales.
- Configurar accesos SSH y usuarios administrativos.
- Verificar conectividad entre todas las máquinas.
- Documentar la configuración base del entorno.

Resultado esperado:

Entorno de laboratorio operativo, con todas las máquinas accesibles y comunicándose correctamente.

FASE 2 – Despliegue del clúster Kubernetes (RKE2)

Herramientas utilizadas:

- RKE2
- kubectl

Actividades:

- Preparar los nodos para Kubernetes.
- Instalar RKE2 en el nodo master.
- Unir los nodos worker al clúster.
- Verificar el estado de los nodos (Ready).
- Comprobar el funcionamiento básico del clúster.
- Documentar la estructura del clúster.

Resultado esperado:

Clúster Kubernetes funcional con todos los nodos correctamente integrados.

FASE 3 – Gestión del clúster con Rancher

Herramientas utilizadas:

- Helm
- Rancher
- Port-forward

Actividades:

- Instalar Rancher mediante Helm.
- Acceder a la interfaz web usando port-forward.
- Visualizar nodos, pods y namespaces.
- Realizar operaciones básicas de gestión desde la UI.
- Verificar el acceso administrativo.
- Documentar el uso de la interfaz.

Resultado esperado:
Clúster gestionable visualmente mediante Rancher.

FASE 3.5 (opcional) – Contenerización con Docker

Herramientas utilizadas:

- Docker
- Docker Hub

Actividades:

- Crear Dockerfile para el servicio web (NGINX).
- Construir la imagen Docker.
- Publicar la imagen en Docker Hub.
- Verificar la disponibilidad de la imagen.
- Documentar el proceso de construcción y publicación.

Resultado esperado:
Imagen Docker propia disponible en Docker Hub y lista para ser usada en Kubernetes.

FASE 4 – Despliegue de servicios en Kubernetes

Herramientas utilizadas:

- Kubernetes (YAML)
- kubectl

Actividades:

- Crear manifiestos YAML para NGINX.
- Desplegar el servicio en el clúster.
- Verificar el acceso al servicio.
- (Opcional) Desplegar MariaDB.
- Configurar almacenamiento persistente (PVC/PV).
- Documentar los despliegues realizados.

Resultado esperado:
Servicios funcionando correctamente dentro del clúster Kubernetes.

FASE 5 – Copias de seguridad y restauración (núcleo del proyecto)

Herramientas utilizadas:

- Velero
- NFS
- Helm

Actividades:

- Configurar el servidor NFS.
- Instalar Velero mediante Helm.
- Ejecutar copias de seguridad manuales.
- Simular la eliminación de recursos.
- Restaurar el sistema desde una copia.
- Validar el resultado del restore.
- Documentar pruebas y evidencias.

Resultado esperado:
Sistema capaz de realizar backups y restaurar el clúster tras fallos simulados.

FASE 6 – Monitorización del sistema

Herramientas utilizadas:

- Prometheus
- Grafana
- Helm

Actividades:

- Instalar Prometheus.
- Instalar Grafana.
- Acceder a los dashboards mediante port-forward.
- Visualizar métricas de nodos y servicios.
- Comprobar el estado del sistema tras restores.
- Documentar capturas y resultados.

Resultado esperado:
Sistema monitorizado con métricas visibles y comprensibles.

FASE 7 – Seguridad básica y control de accesos

Herramientas utilizadas:

- Kubernetes RBAC
- Rancher

Actividades:

- Definir roles básicos en Kubernetes.
- Crear un usuario con permisos limitados.
- Verificar restricciones de acceso.
- Validar accesos desde Rancher y CLI.
- Documentar la configuración de seguridad.

Resultado esperado:
Control de accesos básico implementado y validado.

FASE 8 – Validación final y preparación de la defensa

Herramientas utilizadas:

- Todas las herramientas del proyecto
- Documentación

Actividades:

- Ejecutar pruebas finales del sistema completo.
- Validar todos los criterios de aceptación.
- Revisar y cerrar la documentación.
- Preparar el guion de la defensa.
- Realizar ensayos de presentación.

Resultado esperado:

Proyecto completo, validado y preparado para su defensa.