DPTO. INFORMÁTICA – I.E.S. LA MARISMA
MÓDULO PROYECTO
C.F.G.S.

ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED – GRADO SUPERIOR

PROCESO DE DESPLIGUE E INTEGRACIÓN CONTINUA (CI/CD)

David Faustino Benítez

19/03/2025 – 09/06/2025

Profesor responsable del seguimiento:

Gonzalo Cañadillas Rueda

Tutora del centro de trabajo:

Carmen Pérez Lema

ÍNDICE

INTRODUCCIÓN

El presente documento expone el desarrollo y ejecución de un proyecto de integración continua y despliegue continuo (CI/CD) aplicado al entorno profesional simulado del Ciclo Formativo de Grado Superior en Administración de Sistemas Informáticos en Red (ASIR). Este trabajo tiene como finalidad implementar un sistema completo y automatizado de despliegue de aplicaciones utilizando tecnologías clave del mundo DevOps: GitHub Actions, Docker, Kubernetes (k3s) y Ansible.

A lo largo del proyecto se aborda el proceso de construcción, entrega y despliegue de una aplicación PHP sencilla, consistente en una agenda de contactos que gestiona su información mediante un archivo JSON como sistema de almacenamiento. Esta decisión tecnológica, más ligera que el uso de una base de datos tradicional, ha permitido centrar el proyecto en el ciclo de automatización y despliegue, dejando como mejora futura la incorporación de un sistema relacional como MySQL.

La infraestructura base parte de una red local con un servidor CI/CD (que actúa como runner de GitHub Actions y servidor Ansible), un nodo Kubernetes independiente aprovisionado desde cero mediante Ansible, y uno o más clientes que acceden a las distintas versiones de la aplicación desplegada. El despliegue de cada versión se realiza automáticamente tras una actualización del código fuente, lo que permite validar los cambios de forma aislada sin interferencias entre usuarios, cada uno accediendo a su propio pod generado a partir de una imagen Docker única.

El sistema CI/CD se apoya en un workflow desarrollado en GitHub Actions que detecta los cambios realizados en la rama desarrollo, genera una imagen Docker con el código actualizado, la sube a Docker Hub, y produce automáticamente un manifiesto Kubernetes personalizado. Este manifiesto es aplicado directamente al clúster, desplegando la nueva versión como un nuevo pod accesible desde el exterior a través de un puerto NodePort.

El entorno Kubernetes es previamente aprovisionado por Ansible desde el servidor CI/CD. El nodo arranca con un sistema limpio (Ubuntu Server 22.04) y Ansible se encarga de instalar y configurar k3s, habilitar el acceso por kubectl, y dejar la infraestructura lista para recibir los despliegues automatizados desde los workflows.

Finalmente, cabe destacar que toda la información recogida en el presente documento tiene como objetivo servir de base para la mejora continua y evolución de la infraestructura de despliegue automatizado aquí desarrollada. Si bien este proyecto se centra en una aplicación PHP de agenda de contactos como ejemplo funcional, la arquitectura y metodología empleadas están diseñadas para ser reutilizadas y ampliadas en futuros desarrollos, como podrían ser páginas web, aplicaciones más complejas o incluso sistemas de comercio electrónico. Por tanto, esta documentación no solo representa una solución concreta, sino también una guía adaptable para fases futuras de proyectos similares en entornos reales.

Requisitos funcionales:

A continuación, se describen los requisitos funcionales que debe cumplir el sistema propuesto, entendidos como aquellas capacidades técnicas mínimas que aseguran su correcto funcionamiento dentro del contexto de automatización y despliegue continuo:

- Automatización del despliegue de servicios web
  El sistema debe permitir desplegar automáticamente una aplicación, directamente desde el repositorio de GitHub tras un cambio de código.

- Almacenamiento local de datos en formato JSON
  La aplicación debe gestionar de forma autónoma sus datos mediante un archivo JSON como sistema de almacenamiento. Esta elección elimina la necesidad de un sistema gestor de bases de datos externo y simplifica el despliegue.

- Versionado de despliegues mediante pods diferenciados
  Cada versión de la aplicación generada a partir de una subida debe ser desplegada en un pod independiente dentro del clúster Kubernetes, accesible por un puerto único.

- Gestión de contenedores a través de Kubernetes
  Los servicios deben ser desplegados y gestionados como contenedores, orquestados mediante Kubernetes (k3s) en un nodo previamente aprovisionado.

- Automatización del aprovisionamiento con Ansible
  El nodo Kubernetes debe ser configurado automáticamente desde cero mediante Ansible, instalando k3s, habilitando el acceso, y preparando posibles despliegues externos.

- Desencadenado automático desde GitHub Actions
  El proceso completo de construcción de la imagen, subida a Docker Hub, generación de manifiesto Kubernetes y despliegue en el clúster, automatizado desde GitHub Actions.

Requisitos no funcionales:

Las cualidades del sistema aseguran que el sistema sea mantenible y adaptable en el tiempo:

- Escalabilidad
  La arquitectura debe permitir la creación dinámica de nuevos pods para futuras versiones de la aplicación sin reconfiguración manual, así como la posibilidad de eliminar pods cuando ya no sean necesarios.

- Trazabilidad mediante control de versiones
  Todos los cambios en el código fuente deben estar controlados mediante Git, lo que permite rastrear el origen de cualquier despliegue, vincularlo a un commit específico y mantener un historial completo de versiones.

- Modularidad y reutilización de infraestructura como código
  El código de automatización (Ansible) debe estar estructurado en roles y tareas reutilizables para facilitar futuras modificaciones, migraciones o ampliaciones del sistema.

- Seguridad de acceso y configuración
  El acceso a las máquinas debe realizarse mediante claves SSH seguras. Los secretos como tokens de Docker Hub o claves SSH del runner se deben gestionar exclusivamente mediante GitHub Secrets y no deben almacenarse en texto plano en ningún momento.

ORIGEN Y CONTEXTUALIZACIÓN DEL PROYECTO

El presente proyecto se enmarca dentro del módulo de Proyecto Integrado del segundo curso del Ciclo Formativo de Grado Superior en Administración de Sistemas Informáticos en Red (ASIR). Este módulo tiene como objetivo principal integrar y aplicar de forma práctica los conocimientos adquiridos durante la formación, promoviendo la resolución de situaciones reales mediante el uso de tecnologías actuales y buenas prácticas del sector.

En los últimos años, los métodos tradicionales de despliegue de aplicaciones han sido reemplazados en el entorno empresarial por metodologías DevOps, centradas en la automatización, integración continua y entrega rápida. Estas metodologías permiten mejorar la eficiencia, reducir errores humanos y acelerar la entrega de software, factores fundamentales en entornos donde la agilidad y la escalabilidad son clave.

Dentro de este contexto, surge la necesidad de desarrollar un sistema automatizado de despliegue que simule de forma realista cómo se gestionan y versionan aplicaciones en producción. El proyecto parte de la implementación de una solución funcional que, si bien está basada en una aplicación sencilla escrita en PHP y respaldada por un sistema de almacenamiento en formato JSON, está pensada para reflejar fielmente los flujos de trabajo modernos que se utilizan en empresas reales.

Gracias a la Formación en Centros de Trabajo (FCT), en la que he recibido formación específica orientada al campo de la automatización y la filosofía DevOps, y motivado además por mi interés personal por estas tecnologías, esta idea de proyecto adquirió mayor solidez y sentido práctico. El contacto con metodologías reales y entornos automatizados reforzó la viabilidad del enfoque adoptado, permitiéndome planificar un trabajo que fuera no solo viable técnicamente, sino también útil y cercano a la realidad del sector profesional.

El despliegue de la aplicación se realiza mediante una combinación de herramientas actuales como Docker, Kubernetes, GitHub Actions y Ansible, las cuales permiten construir un sistema CI/CD totalmente automatizado. A través de GitHub, cada nueva versión del código desencadena un proceso que construye una imagen Docker, la sube a un registro remoto y la despliega en un pod dentro de un clúster Kubernetes previamente aprovisionado mediante Ansible. Cada pod es accesible por un cliente de forma independiente, permitiendo el testeo simultáneo de versiones sin conflictos.

Este enfoque refleja de manera práctica los retos que enfrentan los equipos DevOps al implementar sistemas escalables, reproducibles y controlados. Además, permite al alumno poner en práctica conocimientos de redes, automatización, contenedores, sistemas operativos, scripting y administración de servicios, cumpliendo así con el objetivo de integración multidisciplinar que persigue el módulo.

Por tanto, este proyecto no solo responde a un ejercicio académico, sino que constituye una simulación coherente y profesional del tipo de tareas que se espera que desempeñe un técnico de sistemas en entornos empresariales reales, especialmente aquellos enfocados a la automatización de infraestructuras, despliegues en la nube o administración de entornos contenedorizados.

OBJETIVO GENERAL DEL PROYECTO

El objetivo principal de este proyecto es diseñar, desarrollar e implementar un sistema completo de integración y despliegue continuo (CI/CD) que permita automatizar el ciclo de vida de una aplicación web basada en PHP, utilizando tecnologías modernas ampliamente adoptadas en entornos DevOps como GitHub Actions, Docker, Kubernetes (k3s) y Ansible. El proyecto busca proporcionar una solución funcional, escalable y fácilmente replicable que simule un entorno de despliegue profesional y automatizado, alineado con las necesidades reales de la industria.

A través de este sistema, se pretende automatizar todos los pasos necesarios para la entrega de nuevas versiones de una aplicación PHP que utiliza un archivo JSON como almacenamiento de datos. Desde el momento en que se realiza un cambio en el repositorio, GitHub Actions se encarga de construir una imagen Docker personalizada, etiquetarla, subirla a un registro remoto (Docker Hub), generar un manifiesto Kubernetes específico y desplegarla automáticamente como un pod independiente en un clúster previamente aprovisionado mediante Ansible.

El uso de Ansible como herramienta de aprovisionamiento asegura que el entorno donde se desplegarán los pods (el nodo Kubernetes) esté correctamente configurado desde cero, sin necesidad de intervención manual. Esta infraestructura como código permite aplicar configuraciones repetibles y fácilmente escalables, lo que contribuye a mantener la coherencia del sistema y facilita su mantenimiento a largo plazo.

El despliegue de cada versión como un pod aislado, accesible desde clientes remotos en la red interna, garantiza un entorno seguro para pruebas sin que una versión interfiera con otra. Esto resulta especialmente útil para procesos de validación o revisión en equipos de desarrollo.

El sistema resultante no solo simula una situación de trabajo profesional, sino que también permite aplicar de manera integrada conocimientos adquiridos a lo largo del ciclo: administración de sistemas, redes, automatización, scripting, virtualización, contenedores y servicios web. Además, deja sentadas las bases para una posible evolución futura del proyecto, como el uso de bases de datos relacionales o la incorporación de sistemas de autenticación, balanceo de carga, monitoreo o alta disponibilidad.

En definitiva, el proyecto tiene como fin demostrar la viabilidad, utilidad y eficiencia de un sistema CI/CD automatizado basado en tecnologías abiertas, orientado a despliegues controlados y consistentes en entornos productivos simulados, replicando las dinámicas reales de trabajo en el sector de la informática actual.

OBJETIVOS ESPECÍFICOS

A continuación, se enumeran los objetivos específicos que han guiado el desarrollo de este proyecto y que han sido alcanzados de forma progresiva durante su ejecución:

- Configurar una infraestructura de red funcional entre los distintos nodos
  Establecer una red interna entre los equipos participantes del sistema: el servidor CI/CD (que actúa como runner y orquestador), el nodo Kubernetes y los dispositivos cliente, simulando un entorno de pruebas profesional.

- Automatizar el aprovisionamiento del clúster Kubernetes mediante Ansible
  Emplear Ansible para instalar y configurar desde cero el nodo Kubernetes (k3s), asegurando que esté preparado para recibir despliegues sin intervención manual posterior.

- Desarrollar un sistema CI/CD con GitHub Actions
  Crear workflows en GitHub que permitan construir automáticamente imágenes Docker a partir del código fuente, subirlas a Docker Hub y generar manifiestos Kubernetes personalizados por versión.

- Desplegar automáticamente versiones de la aplicación como pods individuales
  Asegurar que cada subida relevante de la aplicación PHP genere un nuevo pod en el clúster, accesible de forma aislada a través de un puerto NodePort único, permitiendo pruebas simultáneas y sin conflictos.

- Eliminar la dependencia de bases de datos externas en la fase actual
  Utilizar un archivo JSON como mecanismo de almacenamiento ligero y funcional, con la previsión de sustituirlo en el futuro por un sistema como MySQL si el entorno lo requiere.

- Validar el acceso desde clientes remotos
  Comprobar que los dispositivos cliente pueden acceder correctamente a cada versión desplegada de la aplicación, verificando funcionalidad y accesibilidad a través de la red interna.

- Documentar el proceso completo del sistema automatizado
  Redactar una guía técnica que recoja la instalación, configuración, ejecución, pruebas y mantenimiento del sistema, con el objetivo de facilitar su reproducción y evolución.

- Evaluar el comportamiento del sistema y su escalabilidad
  Realizar pruebas de carga y despliegue simultáneo, analizando el rendimiento del clúster y su capacidad para gestionar múltiples versiones activas sin degradar el servicio.

TAREAS

En este apartado se describen las distintas fases de desarrollo que componen el proyecto. Cada tarea está orientada a cumplir uno de los objetivos clave del sistema, desde la preparación del entorno hasta las pruebas finales y la documentación. Las tareas se estructuran de forma secuencial, aunque algunas pueden solaparse parcialmente. El enfoque del proyecto combina automatización con Ansible, contenedores Docker, despliegue orquestado con Kubernetes y flujos CI/CD definidos mediante GitHub Actions.

Tarea 1: Preparación del entorno

Esta fase sienta las bases necesarias para el desarrollo del sistema. Incluye el análisis inicial de la red, el aprovisionamiento de las máquinas virtuales o físicas, la instalación del sistema operativo y la verificación de conectividad entre los nodos participantes.

- ***Subtarea 1.1: Análisis de requisitos***

  Identificación de necesidades técnicas para ejecutar el sistema: capacidad de CPU, RAM, red, acceso remoto, y conectividad entre el servidor CI/CD, el clúster y los clientes.

- ***Subtarea 1.2: Selección de tecnologías***

  Elección de herramientas a utilizar: Docker como sistema de contenedores, Kubernetes (k3s) como orquestador, GitHub Actions para CI/CD y Ansible como herramienta de aprovisionamiento.

- ***Subtarea 1.3: Provisionamiento inicial***

  Instalación y configuración de Ubuntu Server en los distintos equipos implicados (servidor CI/CD, nodo Kubernetes, clientes), así como configuración básica de red y acceso SSH.

- ***Subtarea 1.4: Verificación de conectividad***

  Comprobación del acceso entre nodos, verificación de resolución DNS si procede, accesos por clave SSH y visibilidad desde los clientes al clúster.

Tarea 2: Configuración del CI/CD

Esta tarea se centra en el desarrollo del flujo automatizado de despliegue utilizando GitHub Actions. Desde la subida en el repositorio, se desencadena una acción que construye la imagen Docker de la aplicación PHP, la etiqueta, la publica en Docker Hub y genera un manifiesto Kubernetes con la versión correspondiente.

- ***Subtarea 2.1: Desarrollo del workflow de GitHub Actions***

  Creación de un pipeline que automatice la construcción, publicación y despliegue de la aplicación. Se definen los pasos del workflow y los triggers adecuados (pull request o push sobre ramas).

- ***Subtarea 2.2: Construcción y etiquetado de imágenes***

  Generación automática de imágenes Docker basadas en el código fuente. Cada imagen se etiqueta usando el hash del commit, lo que facilita la trazabilidad.

- ***Subtarea 2.3: Docker Hub***

  Configuración del acceso seguro a Docker Hub mediante GitHub Secrets y publicación de las imágenes generadas en el repositorio del usuario.

- ***Subtarea 2.4: Manifiestos de Kubernetes***

  Generación dinámica de manifiestos YAML con el nombre de la imagen y el puerto asignado, para desplegar automáticamente pods aislados dentro del clúster.

Tarea 3: Aprovisionamiento y mantenimiento con Ansible

Esta parte garantiza que el clúster Kubernetes esté configurado desde una instalación limpia de Ubuntu. Se emplea Ansible para automatizar la instalación de k3s, dejar configurado el entorno y permitir el control remoto desde el servidor CI/CD.

- ***Subtarea 3.1: Estructura de roles***

  Organización de los playbooks de Ansible en roles reutilizables que faciliten el mantenimiento del proyecto y su posible adaptación a otros nodos o entornos.

- ***Subtarea 3.2: Desarrollo de los Playbooks***

  Redacción de tareas Ansible que permitan instalar y configurar k3s, preparar las herramientas necesarias y dejar el entorno listo para recibir manifiestos.

- ***Subtarea 3.3: Acceso al clúster***

  Configuración del acceso desde el servidor CI/CD mediante kubectl, permitiendo a GitHub Actions aplicar directamente los manifiestos desde el runner local.

Tarea 4: Pruebas y documentación final

Combina pruebas exhaustivas con la entrega de documentación completa.

- ***Subtarea 4.1: Pruebas de despliegue automatizado***

  Comprobación de que los cambios en el repositorio generan nuevos pods funcionales en el clúster, y que estos responden correctamente a las peticiones de los clientes.

- ***Subtarea 4.2: Pruebas de acceso***

  Acceso desde los dispositivos cliente a las versiones desplegadas de la aplicación mediante navegador o herramientas de red, validando su aislamiento y funcionamiento individual.

- ***Subtarea 4.3: Redacción de un manual técnico***

  Elaboración de documentación detallada que incluya instrucciones de instalación, uso de Ansible, estructura de los workflows, pruebas realizadas y consideraciones para mantenimiento futuro.

- ***Subtarea 4.4: Revisión y entrega***

  Revisión general del proyecto, inclusión de anexos, limpieza del código y documentación y preparación de la presentación y defensa final.

	Duración (horas)	**Fecha inicio**	**Fecha fin**	**Responsable**
Tarea 1	12	19/03/2025	25/03/2025	David Faustino
Subtarea 1.1	3			David Faustino
Subtarea 1.2	5			David Faustino
Subtarea 1.3	2			David Faustino
Subtarea 1.4	2			David Faustino
Tarea 2	19	02/04/2025	28/04/2025	David Faustino
Subtarea 2.1	4			David Faustino
Subtarea 2.2	4			David Faustino
Subtarea 2.3	6			David Faustino
Subtarea 2.4	5			David Faustino
Tarea 3	9	03/05/2025	18/05/2025	David Faustino
Subtarea 3.1	1			David Faustino
Subtarea 3.2	6			David Faustino
Subtarea 3.3	2			David Faustino
Tarea 4	13	18/03/2025	09/06/2025	David Faustino
Subtarea 4.1	2			David Faustino
Subtarea 4.2	1			David Faustino
Subtarea 4.3	8			David Faustino
Subtarea 4.4	2			David Faustino

TOTAL HORAS: 44 ~

RECURSOS HUMANOS

El desarrollo de este proyecto ha sido realizado de forma individual por el autor, asumiendo todas las responsabilidades técnicas y documentales necesarias para su implementación. No obstante, si este sistema fuera a desarrollarse o mantenerse en un entorno profesional o empresarial real, se requerirían distintos perfiles especializados para asegurar una ejecución óptima, mantenible y escalable del sistema.

A continuación, se detallan los perfiles profesionales implicados, sus funciones principales y la cantidad mínima recomendada para un equipo funcional:

Perfil Profesional	Funciones	Cantidad
Administrador del sistema / DevOps	• Instalación y configuración del sistema. • Gestión de contenedores Docker. • Automatización con Ansible. • Desarrollo de playbooks de Ansible. • Despliegue de Kubernetes. • Construcción proceso CI/CD.	1
Tester / Quality Assurance	• Comprobación de la funcionalidad. • Pruebas de estrés. • Validación de la automatización.	1
Redactor / Documentador	• Manuales técnicos • Diagramas de red e infraestructura • Documentación de soporte • Revisión de la documentación	1

RECURSOS MATERIALES

A continuación, se detallan los recursos materiales necesarios para la ejecución del proyecto. Estos recursos permiten simular un entorno profesional de automatización y despliegue de aplicaciones web, tanto en un entorno virtualizado como en su equivalente en infraestructura real.

Infraestructura:

Equipos desplegados:

- Máquina 1 – Servidor CI/CD, actúa como runner y servidor de Ansible principalmente.
- Máquina 2 – Nodo de Kubernetes, desde este equipo se accede al despliegue.
- Máquina Cliente – Dentro de la red un equipo cliente para comprobar funcionamiento.

Tecnologías:

- Ubuntu Server 22.04 LTS.
- Ubuntu 22.04 (Cliente).
- Docker para la construcción de imágenes de la aplicación PHP.
- Kubernetes para la orquestación de contenedores.
- GitHub Actions para la automatización del proceso CI/CD.
- Ansible, como medio de aprovisionamiento y mantenimiento de máquinas en red.
- Docker Hub, sirviendo como repositorio donde almacenar las imágenes generadas.
- Git, indispensable en el trabajo con repositorios de GitHub.

Arquitectura lógica:

- Se modifica la aplicación de PHP
- La subida en el repositorio crea una automatización que conteneriza la aplicación.
- La imagen se almacena en el repositorio de DockerHub.
- El servidor desde la red interna descarga la imagen y la despliega, mediante playbooks de Ansible, sobre el clúster ya configurado de Kubernetes.
- El cliente puede acceder desde la red interna al nodo y visualizar el programa.

CRONOGRAMA

- Inicio: Marzo 2025
- Fin estimado: Junio 2025
- Duración total: 4 meses
- Fases del proyecto:

    - Revisión de requisitos técnicos y diseño de arquitectura
      Identificación de recursos necesarios, diseño de la red, elección de tecnologías y planificación de tareas.

    - Preparación del entorno virtualizado de pruebas
      Instalación de Ubuntu Server en todas las máquinas virtuales, configuración de red, acceso SSH y pruebas de conectividad.

    - Aprovisionamiento automatizado del clúster Kubernetes con Ansible
      Desarrollo de playbooks que instalan k3s y configuran el nodo para recibir despliegues desde GitHub Actions.

    - Configuración del flujo CI/CD con GitHub Actions
      Desarrollo del workflow que automatiza la construcción de imágenes Docker, su subida a Docker Hub y el despliegue de pods con manifiestos generados dinámicamente.

    - Creación y prueba de la imagen Docker de la aplicación PHP
      Dockerización de la app, configuración del entorno de ejecución y verificación del uso correcto del almacenamiento en JSON.

    - Despliegue automatizado de versiones por subida
      Validación de que cada versión genera un pod único y accesible a través de NodePort, sin conflictos entre clientes.

    - Pruebas de funcionamiento y revisión desde los clientes
      Acceso desde clientes Ubuntu, simulación de errores, validación de conectividad y funcionalidad de los despliegues.

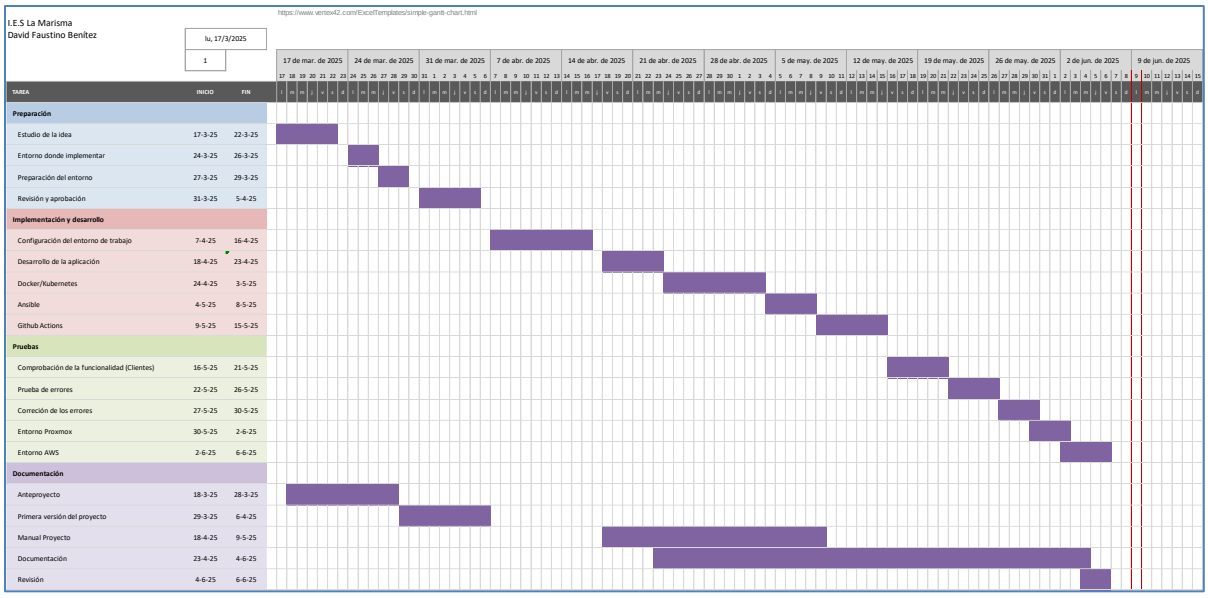
- Redacción del manual técnico y documentación del sistema
  Elaboración de guías de instalación, uso de Ansible, estructura del workflow CI/CD y esquema general del sistema.

- Revisión final, anexos y entrega del proyecto integrado
  Validación completa, ajuste de contenidos, preparación para la presentación y cierre del proyecto.

Gráfico de Gantt

Para facilitar la visualización de la planificación temporal del proyecto y la relación entre las distintas tareas, se ha elaborado un gráfico de Gantt, el cual permite representar de forma clara la distribución de las fases a lo largo del tiempo. Este tipo de herramienta es comúnmente utilizado en la gestión de proyectos para controlar el progreso de las actividades, identificar solapamientos, dependencias y posibles desviaciones con respecto al cronograma inicial.

En el gráfico de Gantt se incluyen las tareas principales del proyecto, divididas en subtareas, así como la duración estimada de cada una de ellas y su distribución a lo largo de las semanas comprendidas entre marzo y junio de 2025. Esto proporciona una visión clara del flujo de trabajo y permite comprender la secuencia lógica de ejecución del proyecto. El uso de este recurso gráfico refuerza la planificación estructurada del trabajo realizado y demuestra una correcta organización y gestión del tiempo, junto a un aspecto técnico de carácter profesional.



PRESUPUESTO

Escenario 1: Entorno de pruebas con Proxmox

Este escenario corresponde a la implementación del proyecto en un entorno de pruebas local, como el desarrollado durante este trabajo. La infraestructura se basa en una máquina servidor capaz de alojar varias máquinas virtuales mediante el sistema de virtualización Proxmox VE. Este tipo de entorno permite simular un escenario realista sin incurrir en grandes costes, resultando ideal para entornos académicos, laboratorios o fases iniciales de desarrollo.

A continuación, se detalla los recursos físicos estimados, junto a un coste aproximado:

Infraestructura (entorno de pruebas)

Recurso	Coste estimado	Descripción
Servidor físico (host Proxmox)	700 € – 1.200 €	Equipo con CPU multinúcleo, 16–32 GB RAM, SSD o disco NVMe.
Almacenamiento adicional (opcional)	100 € – 200 €	Discos extras para un correcto manejo del entorno y permitiendo capacidad y funcionalidad de los servicios del entorno.
Electricidad y mantenimiento	50 € – 100 €/año	Estimación para uso de pruebas.

En este proyecto se ha hecho uso exclusivamente de tecnologías de código abierto y herramientas libres como Ubuntu Server, Docker, Ansible, Kubernetes (k3s) y GitHub Actions, por lo que no se han generado costes asociados a licencias o suscripciones. Este enfoque no solo permite reducir el presupuesto, sino también seguir buenas prácticas de sostenibilidad económica en entornos educativos.

Escenario 2: Entorno simulado empresarial (infraestructura física y licencias)

Nota: La conversión de este proyecto a un entorno empresarial real puede variar considerablemente en función de la infraestructura existente, el número de usuarios/clientes concurrentes, y las políticas de seguridad, soporte y escalabilidad específicas de cada organización.

Este escenario representa una simulación de cómo se desplegaría el proyecto en un entorno empresarial real. A diferencia del entorno de pruebas, en este caso se prescinde del uso de virtualización (como Proxmox) y se asume el uso de equipos físicos dedicados, orientados a producción, con capacidad para atender a un número indefinido de usuarios o clientes en red.

La infraestructura mantiene la misma distribución: un servidor para CI/CD (donde se encuentra Ansible y el runner de GitHub Actions), un nodo dedicado para el clúster Kubernetes (encargado de ejecutar los pods con las versiones de la aplicación) y dispositivos cliente que simulan el acceso y testeo del producto desplegado. Esta conversión, aunque estructuralmente parecida, implica un incremento de costes, especialmente en lo relativo al hardware dedicado, consumo eléctrico, mantenimiento, así como licencias comerciales necesarias para entornos empresariales.

A continuación, se presenta una estimación aproximada de los costes asociados a la infraestructura física y las licencias de software necesarias para su implementación en un entorno empresarial.

Infraestructura física

Recurso	Coste estimado	Descripción
Servidor principal	2.000 € – 2.500 €	Equipo físico con soporte profesional, orientado a automatización y orquestación del despliegue.
Clúster de Kubernetes	2.000 € - 2.500 €	Servidor físico independiente que aloja el clúster y gestiona los pods de la aplicación.
Clientes físicos	Variable (mín. 1.200 € por unidad)	Estaciones de trabajo desde donde se accede a los pods desplegados. El número depende del volumen de testeo.
Elementos de red	800 € – 1.200 €	Infraestructura de red física, protección eléctrica y segmentación de seguridad.

Licencias empresariales y software

Software	Tipo de licencia	Coste aproximado
Docker Business	Anual por usuario	250 € por usuario y año
Ansible Automation Platform	Por nodo (Red Hat)	1.500 € / año
Nginx Plus	Comercial	2.500 € / año
GitHub (plan empresarial)	Plan con runners dedicados y gestión avanzada	500 € / año

Nota: La aplicación PHP no requiere licencias y el sistema de almacenamiento en JSON evita costes adicionales por bases de datos. En caso de evolución futura hacia un sistema como MySQL Enterprise Edition, habría que considerar un coste adicional de ~5.000 € anuales.

Otros costes estimados

- Electricidad, refrigeración y mantenimiento anual: **200 € – 300 €**
- Formación externa y documentación adicional: **500 € aprox.**
- Auditoría o consultoría técnica externa (opcional): variable según necesidades

Esta simulación busca reflejar cómo se transformaría el sistema desarrollado en el proyecto académico a un entorno profesional realista, manteniendo la lógica de automatización y despliegue, pero con mayor robustez, soporte técnico y cumplimiento de estándares empresariales. Los costes pueden variar considerablemente en función de la escala, los acuerdos de licenciamiento, el número de usuarios y los requisitos concretos de cada organización.

Total estimado del proyecto en entorno físico profesional:

El coste global estimado para desplegar este sistema en un entorno empresarial, considerando el uso de hardware físico dedicado, licencias comerciales y servicios básicos de soporte, se sitúa entre **11.500 € y 14.000 €.**

Se considera:

- Equipos físicos: servidor CI/CD, nodo Kubernetes y al menos un cliente de acceso.
- Red empresarial básica: SAI, switches, firewall y cableado.
- Licencias anuales para software en versión comercial.
- Mantenimiento físico estimado (electricidad, refrigeración y soporte técnico).
- Formación técnica del personal o documentación interna personalizada.

Categoría	**Rango de costes estimado**
Hardware e infraestructura	5.000 € – 7.500 €
Licencias y suscripciones anuales	4.500 € – 5.000 €
Otros costes operativos	500 € – 1.000 €
Total estimado	11.500 € – 14.000 €

Tabla de costos por tipo de recursos

Tarea / Subtarea	RH (€)	R. Materiales (€)	Total (€)
Tarea 1: Preparación del entorno			
Subtarea 1.1: Análisis de requisitos	200	0	200
Subtarea 1.2: Selección de tecnologías	150	0	150
Subtarea 1.3: Provisionamiento inicial	300	2.000 (servidor CI/CD)	2.300
Subtarea 1.4: Verificación de red	200	100 (switch/firewall)	300
Tarea 2: Configuración del CI/CD			
Subtarea 2.1: Workflow GitHub Actions	400	500 (plan empresarial	900
Subtarea 2.2: Construcción de imágenes	300	250 (Docker Business)	550
Subtarea 2.3: Publicación Docker Hub	100	incluido arriba	100
Subtarea 2.4: Manifiestos Kubernetes	300	0	300
Tarea 3: Aprovisionamiento con Ansible			
Subtarea 3.1: Estructura de roles	250	0	250
Subtarea 3.2: Desarrollo de playbooks	400	1.500 (Ansible	1.900
Subtarea 3.3: Acceso al clúster	200	2.000 (nodo Kubernetes)	2.200
Tarea 4: Pruebas y documentación final			
Subtarea 4.1: Pruebas de despliegue	250	0	250
Subtarea 4.2: Pruebas de acceso	250	1.200 (cliente físico)	1.450
Subtarea 4.3: Manual técnico	300	0	300
Subtarea 4.4: Revisión y entrega	200	0	200
Totales	**3.950 €**	**7.550 €**	**11.500 €**

MEJORAS FUTURAS

Aunque el sistema implementado cumple con los objetivos establecidos, existen diversas posibilidades de mejora y ampliación que podrían aplicarse en fases posteriores para dotar al proyecto de mayor robustez, funcionalidad y adaptabilidad a entornos productivos más exigentes. Estas posibles mejoras permitirían una evolución progresiva hacia una solución aún más profesional y escalable:

- Sustitución del almacenamiento en JSON por un sistema de base de datos relacional
  La aplicación PHP actual almacena datos en un archivo JSON, lo cual es útil para pruebas y simplicidad. Sin embargo, en entornos reales sería más apropiado utilizar un sistema como MySQL o PostgreSQL, mejorando la persistencia, integridad de los datos y escalabilidad.

- Implementación de balanceo de carga
  Se podría incorporar un balanceador de carga (por ejemplo, Nginx o HAProxy) delante del clúster para distribuir tráfico entre múltiples instancias de la aplicación y garantizar alta disponibilidad.

- Sistema de monitorización y alertas
  Añadir herramientas como Prometheus, Grafana o ELK Stack permitiría tener visibilidad sobre el estado del clúster, el uso de recursos y posibles incidencias en tiempo real.

- Seguridad avanzada y control de acceso
  Implementar mecanismos como HTTPS con certificados válidos, autenticación por tokens, control de acceso por roles (RBAC) en Kubernetes y escaneo de imágenes Docker para mejorar la seguridad general del entorno.

- Despliegue multinodo en Kubernetes
  Ampliar el clúster de Kubernetes a múltiples nodos para simular entornos de producción reales y permitir la distribución de pods entre diferentes servidores físicos o virtuales.

- Integración con herramientas de testing automático
  Incorporar pruebas automáticas (unitarias, de integración o de aceptación) que se ejecuten en cada pull request antes del despliegue automático en el clúster.

- Paso del pipeline a un entorno cloud
  El pipeline CI/CD actualmente se ejecuta desde un runner local. Se podría migrar completamente el proceso a un entorno cloud (como GitHub-hosted runners, GitLab CI, AWS CodePipeline, etc.) para facilitar la portabilidad y disponibilidad global.

ANEXOS

Diagrama de la estructura de red

Diagrama del flujo CI/CD

Manual de usuario

Repositorios utilizados:

Repositorio del proyecto

https://github.com/Dfauben/Proyecto-Integrado-ASIR

Repositorio de simulación del entorno

https://github.com/Dfauben/CI-CD-Simulacion-PHP

BIBLIOGRAFÍA

Documentación oficial y manuales técnicos

- Docker Documentation. Docker Inc.
  https://docs.docker.com/
- Ansible Documentation. Red Hat
  https://docs.ansible.com/
- GitHub Actions Documentation. GitHub
  https://docs.github.com/es/actions
- Kubernetes Documentation. Cloud Native Computing Foundation
  https://kubernetes.io/docs/home/
- Docker Hub Reference
  https://hub.docker.com/
- k3s - Lightweight Kubernetes
  https://rancher.com/docs/k3s/latest/en/
- GitHub CLI y autenticación mediante tokens
  https://cli.github.com/manual/

Guías de buenas prácticas y recursos comunitarios

- Stack Overflow – Preguntas y respuestas sobre automatización y despliegue
  https://stackoverflow.com/
- GitHub Community Discussions – Repositorios, ejemplos y foros de DevOps
  https://github.com/community
- Dev.to – Comunidad de desarrolladores sobre herramientas CI/CD
  https://dev.to/t/ci
- Awesome Ansible – Recolección de recursos útiles sobre Ansible
  https://github.com/ansible-community/awesome-ansible
- Awesome Kubernetes – Herramientas y ejemplos para Kubernetes
  https://github.com/ramitsurana/awesome-kubernetes
- Medium – Artículos técnicos sobre CI/CD con GitHub Actions y Docker
  https://medium.com/tag/github-actions

Recursos académicos y formativos

- Pautas para la elaboración del Proyecto Integrado. IES La Marisma (curso 2024/2025)
- Manual de Administración de Sistemas GNU/Linux – Fundación GNU/Linux
  https://www.escomposlinux.org/manuales/
- Curso DevOps Essentials – Red Hat Learning Community
  https://learn.redhat.com/
- Udemy – Plataforma de formación online. Cursos de Docker, Kubernetes y DevOps
  https://www.udemy.com/

CONCLUSIÓN

El presente proyecto ha sido una experiencia técnica, académica y personal de gran valor, en la que se ha logrado integrar conocimientos adquiridos a lo largo del ciclo formativo en Administración de Sistemas Informáticos en Red (ASIR), combinando prácticas de automatización, despliegue de aplicaciones, administración de sistemas y desarrollo de infraestructuras reproducibles.

El objetivo principal era diseñar un sistema funcional de integración y despliegue continuo (CI/CD) capaz de construir, versionar y desplegar automáticamente una aplicación web PHP contenedorizada, utilizando herramientas modernas. Todo ello debía desarrollarse en una infraestructura de red controlada, partiendo desde cero, simulando un entorno profesional que pudiera escalar en futuras fases o adaptarse a necesidades empresariales reales.

Durante el proceso se ha configurado una red interna que conecta distintos nodos: un servidor que actúa como centro de operaciones (incluyendo Ansible y el runner de GitHub Actions), un nodo independiente con Kubernetes (k3s) configurado y aprovisionado de forma completamente automatizada, y varios clientes que acceden a los pods desplegados para validar nuevas versiones. El sistema permite que cada versión de la aplicación generada desde una rama o pull request se ejecute en un contenedor aislado, accesible desde la red, sin interferir con otras versiones ya desplegadas. Uno de los aspectos más valiosos ha sido la implementación de Ansible para configurar desde cero el nodo Kubernetes. Esto ha demostrado la potencia de la automatización en tareas de infraestructura, facilitando la reproducibilidad del sistema, la coherencia entre entornos y la posibilidad de escalar horizontalmente de manera más eficiente. Asimismo, el uso de GitHub Actions como motor del pipeline de integración y despliegue ha aportado dinamismo y control a todo el ciclo de vida de la aplicación. Se ha demostrado que es posible generar imágenes Docker personalizadas con cada versión del código, subirlas a un registro (Docker Hub), generar automáticamente los manifiestos de despliegue para Kubernetes y aplicar los cambios de forma remota desde un entorno completamente automatizado. El sistema, aunque orientado a entornos educativos y de pruebas, ha sido planteado con visión profesional, y puede evolucionar hacia una solución real para entornos empresariales. De hecho, durante el análisis del presupuesto y la simulación del entorno profesional, se ha reflexionado sobre los costes y requerimientos reales que supondría llevar este modelo a una infraestructura física, incluyendo licencias comerciales, servidores dedicados y medidas de seguridad avanzadas.

En lo personal, este proyecto ha sido una oportunidad para desarrollar habilidades técnicas clave, resolver problemas de forma autónoma, aplicar metodologías reales del mundo IT y comprender mejor los procesos de automatización.

Por último, me gustaría agradecer sinceramente al IES La Marisma por su acompañamiento a lo largo de estos dos años de formación. A todo el profesorado del ciclo ASIR, por su dedicación, apoyo y exigencia, que han contribuido a desarrollar mi perfil técnico y profesional. Y especialmente, al centro de trabajo STEMDO, donde he tenido la oportunidad de adentrarme de forma práctica en el campo de la automatización, DevOps y la gestión de sistemas, y cuyo entorno profesional me ha inspirado directamente en la idea y ejecución de este proyecto.