

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

# **AUTOMATIZACIÓN DE LA ENTREGA Y DESPLIEGUE**




Grado en Ingeniería Informática – Ingeniería del Software  
Evolución y Gestión de la Configuración

Curso 2024 – 2025

Proyecto: [innosoft-diplomas-1](#)

<b>Autores</b>
Aragón Sánchez, Álvaro
Chico Castellano, Álvaro
Guillén Fernández, David
Jiménez Osuna, Álvaro
Linares Barrera, Jaime
López Oliva, Ángela

	Evaluación y Gestión de la Configuración Automatización de la entrega y despliegue
	<b>Control de Versiones</b>

### **Control de versiones**

Fecha	Versión	Descripción
11/11/24	1.0.	Creación y elaboración del documento
07/12/24	1.1.	Última corrección del documento



## Índice de contenido

1. Introducción y objetivos.....	2
2. Herramientas utilizadas.....	2
3. Implementación .....	3
4. Conclusiones y resultados esperados .....	6



## 1. Introducción y objetivos

El presente documento tiene como finalidad establecer de manera clara y precisa los procedimientos y directrices que guiarán la **automatización de la entrega y despliegue** del proyecto **innosoft-diplomas-1**. Este enfoque busca asegurar que la aplicación sea **estable y confiable** en los entornos de prueba y producción, permitiendo una entrega continua de nuevas funcionalidades y correcciones de errores de manera eficiente y sin interrupciones.

## 2. Herramientas utilizadas

Para el despliegue automatizado y el desarrollo del proyecto **innosoft-diplomas-1**, se han seleccionado las siguientes herramientas:

### GitHub Actions

- Plataforma para **Integración Continua (CI)** que ejecuta pruebas y valida cambios de forma automática en cada **push** o **pull request** al repositorio.
- Asegura que el código pase todas las pruebas antes de proceder al despliegue en el entorno de preproducción.

### Render

- Plataforma de **despliegue en la nube** que gestiona la implementación continua de la aplicación sin intervención manual (script de integración continua para que en el momento en el que se añada código a la rama main se despliegue en Render).
- Asegura que la aplicación esté accesible en un entorno remoto, simulando el entorno de producción.

### Docker

- Se utiliza Docker en este proyecto para **empaquetar la aplicación** y sus dependencias en **contenedores**, asegurando que la misma se ejecute de manera consistente en diferentes entornos de desarrollo y producción mejorando la eficiencia y estabilidad del sistema.

- Además, se ha configurado un sistema de integración continua que, con cada cambio en la rama principal (main), construye una nueva imagen de Docker y la publica automáticamente en Docker Hub para su posterior despliegue.

### **Máquinas virtuales (Vagrant y Ansible)**

- En este proyecto también se ha utilizado **Vagrant y Ansible** para el despliegue de máquinas virtuales. Vagrant se emplea para crear entornos de desarrollo consistentes y replicables, mientras que Ansible automatiza la configuración y gestión de las máquinas virtuales, asegurando que el entorno de producción esté configurado correctamente. Con este enfoque, conseguimos una infraestructura flexible y reproducible para el despliegue de la aplicación.

## **3. Implementación**

Para el despliegue automatizado y el desarrollo del proyecto **innosoft-diplomas-1**, se han seleccionado varias herramientas que permiten un flujo de trabajo ágil y controlado. A continuación, se detalla cómo se ha implementado cada una de ellas:

### **Despliegue local**

#### **Entorno Virtual**

Para el desarrollo local, se utiliza un entorno virtual, que permite aislar las dependencias del proyecto. Este entorno asegura que cada miembro del equipo trabaje con las mismas versiones de librerías y herramientas, evitando conflictos de dependencias.

- **Requisitos:** Se usa un archivo requirements.txt para instalar las dependencias necesarias con un solo comando (pip install -r requirements.txt).
- **Propósito:** Garantizar que todos los desarrolladores trabajen en un entorno consistente, reduciendo problemas al integrar el código en el repositorio.

#### **Pruebas Locales**

Antes de realizar un commit, los desarrolladores ejecutan pruebas automatizadas localmente. Estas pruebas son esenciales para asegurarse de que los cambios realizados no introduzcan errores.

- **Objetivo:** Validar que solo el código funcional llegue al repositorio y se minimicen los errores de integración.

### **Despliegue en remoto en Render**

#### **Integración Continua con GitHub Actions**

El proceso de despliegue remoto comienza cuando el código pasa todas las pruebas de integración continua configuradas en GitHub Actions.

- **Proceso:** Cada vez que se realiza un push o pull request en la rama main o preproduction, GitHub Actions ejecuta un pipeline que incluye:
  - Instalación de dependencias.
  - Ejecución de pruebas unitarias e integrales.
  - Validación de la calidad del código.
- **Despliegue Automático:** Si todas las pruebas son exitosas, se procede al despliegue en Render.

#### **Despliegue en Render**

Render se utiliza como plataforma de despliegue en la nube para gestionar la implementación continua de la aplicación sin intervención manual.

- **Automatización:** A través de un script de integración continua, en cuanto el código se añade a la rama main, Render despliega automáticamente la aplicación.
- **Entorno Remoto:** Render asegura que la aplicación esté accesible en un entorno remoto que replica el entorno de producción, permitiendo realizar pruebas finales antes de lanzar al público.

### **Contenerización con Docker**

#### **Uso de Docker**

En este proyecto se utiliza Docker, lo que garantiza que el entorno de desarrollo y producción sean consistentes.

- **Contenerización:** Docker permite empaquetar la aplicación junto con todas sus dependencias dentro de un contenedor, asegurando que se ejecute de manera idéntica en todos los entornos.

- **Beneficios:** Esto mejora la eficiencia, la estabilidad del sistema y facilita el despliegue al eliminar las diferencias entre los entornos de desarrollo, pruebas y producción.
- **Integración Continua:** Además, se ha configurado un sistema de integración continua que, con cada cambio en la rama main, construye una nueva imagen de Docker y la publica automáticamente en Docker Hub. Esta imagen es luego utilizada para el despliegue en los entornos de producción.

### Despliegue de máquinas virtuales con Vagrant y Ansible

#### **Uso de Vagrant y Ansible**

Además de Docker, se ha utilizado **Vagrant** y **Ansible** para el despliegue de máquinas virtuales, permitiendo la creación de entornos replicables tanto para desarrollo como para producción.

- **Vagrant:** Se utiliza para crear entornos virtuales consistentes. Vagrant permite configurar y gestionar máquinas virtuales mediante un archivo de configuración (Vagrantfile), que describe el entorno deseado. Con ello, se garantiza que todos los desarrolladores y el equipo de operaciones utilicen configuraciones idénticas.
- **Ansible:** Ansible se encarga de la automatización de la configuración de las máquinas virtuales. Esto incluye la instalación y configuración de software, servicios y la aplicación. Ansible garantiza que todas las máquinas virtuales estén configuradas de manera correcta y coherente sin intervención manual.
- **Objetivo:** Esta combinación de Vagrant y Ansible facilita la creación y mantenimiento de entornos virtuales replicables y flexibles, asegurando que el entorno de producción esté configurado de manera correcta y consistente, mejorando la estabilidad de la aplicación.

#### 4. Conclusiones y resultados esperados

El proyecto **innosoft-diplomas-1** ha logrado integrar una serie de herramientas y procesos que aseguran un ciclo de desarrollo eficiente y confiable, con un despliegue automatizado que minimiza los riesgos de errores y mejora la estabilidad general de la aplicación. A través de la utilización de **GitHub Actions**, **Render**, **Docker**, **Vagrant** y **Ansible**, se ha conseguido un flujo de trabajo ágil y controlado que no solo optimiza el tiempo de desarrollo, sino que también facilita la escalabilidad y flexibilidad en el despliegue tanto en entornos locales como remotos. A continuación, se resumen las conclusiones clave y los resultados esperados:

##### Conclusiones

1. **Automatización Completa del Despliegue:** La integración de herramientas como **GitHub Actions** y **Render** ha permitido automatizar por completo el proceso de integración y despliegue continuo, lo que reduce significativamente la intervención manual, aumentando la eficiencia y reduciendo la posibilidad de errores humanos durante el despliegue.
2. **Consistencia de Entornos:** El uso de **Docker** garantiza que la aplicación se ejecute de manera consistente en todos los entornos, desde el desarrollo local hasta la producción. La creación de imágenes Docker automáticamente con cada cambio en la rama principal asegura que el entorno de producción se mantenga alineado con el desarrollo y las pruebas.
3. **Escalabilidad y Flexibilidad:** La combinación de **Vagrant** y **Ansible** para el despliegue de máquinas virtuales ofrece una infraestructura altamente escalable y replicable, facilitando la creación de entornos de desarrollo y producción que son consistentes y fácilmente gestionables.
4. **Mejora de la Calidad del Código:** La integración de pruebas automatizadas en el pipeline de **GitHub Actions** asegura que solo el código funcional y validado llegue al entorno de producción, lo que mejora la calidad general del sistema y reduce los fallos en producción.
5. **Despliegue Remoto Simplificado:** Con **Render**, el proceso de despliegue en la nube es simplificado, permitiendo que la aplicación sea accesible en un



entorno remoto que simula la producción. Esto facilita las pruebas finales y reduce el riesgo de problemas al mover la aplicación entre diferentes entornos.

### **Resultados Esperados**

1. **Mayor Eficiencia en el Desarrollo:** Se espera que el equipo de desarrollo se beneficie de un flujo de trabajo más eficiente y controlado, donde las tareas repetitivas de configuración y despliegue sean automatizadas, permitiendo que los desarrolladores se concentren en mejorar las funcionalidades del sistema.
2. **Despliegue Continuo y Sin Interrupciones:** Gracias a la automatización de la entrega y despliegue, se prevé una entrega continua de nuevas funcionalidades y correcciones de errores, sin que esto implique interrupciones en el servicio, lo cual es fundamental para aplicaciones en producción que requieren alta disponibilidad.
3. **Mayor Estabilidad y Menos Errores:** Con el uso de herramientas de integración continua y contenedores, se espera que el sistema sea más estable, con un menor número de errores y fallos en producción, ya que las pruebas automatizadas y el empaquetado del entorno aseguran una mayor consistencia entre desarrollo y producción.

### **Impacto General**

En resumen, la implementación de estas herramientas y procesos no solo optimiza el ciclo de vida del desarrollo de software, sino que también ofrece un marco robusto y fiable para la gestión de la infraestructura. Este enfoque proporciona la agilidad necesaria para responder rápidamente a los cambios en el proyecto, al mismo tiempo que minimiza el riesgo de fallos en producción, lo que asegura una experiencia de usuario final confiable y eficiente.