

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

Automatización de la Entrega y Despliegue (Release & Deployment)




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

Curso 2024 – 2025

Fecha	Versión
11/11/2024	1.0.

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

Equipo de trabajo
Aragón Sánchez, Alejandro
Chico Castellano, Álvaro
Guillén Fernández, David
Jiménez Osuna, Álvaro
Linares Barrera, Jaime
López Oliva, Angela

	Automatización de la Entrega y Despliegue
	Control de Versiones

Control de Versiones

Fecha	Versión	Descripción
11/11/2024	1.0.	Creación, elaboración y firma del documento



Automatización de la Entrega y Despliegue

Índice de contenido

1. Introducción y objetivos	2
2. Herramientas utilizadas.....	2
3. Implementación.....	3
4. Resultados Esperado	5



Automatización de la Entrega y Despliegue

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:

1. Python Virtual Environment (venv)

- Utilizado para crear un **entorno de desarrollo aislado**. Permite instalar dependencias específicas del proyecto sin interferir con otras configuraciones del sistema.
- Asegura que todos los desarrolladores utilicen un entorno uniforme durante el desarrollo local.

2. requirements.txt

- Archivo que contiene una lista de todas las **dependencias del proyecto**. Facilita la instalación y configuración del entorno tanto en desarrollo local como en entornos remotos.

3. 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.

4. Render

- Plataforma de **despliegue en la nube** que gestiona la implementación continua de la aplicación sin intervención manual.
- Asegura que la aplicación esté accesible en un entorno remoto, replicando el entorno de **producción** para pruebas finales.

5. No uso de Docker

- **Docker no se utiliza** en este proyecto debido a su simplicidad y al uso de Render, que ya gestiona la infraestructura necesaria para el despliegue y ejecución de la aplicación.



3. Implementación

El proyecto **innosoft-diplomas-1** sigue un enfoque automatizado para el despliegue tanto en entornos locales como remotos, asegurando un flujo de trabajo eficiente y controlado. A continuación se detalla la implementación:

1. Despliegue Local

- **Entorno Virtual (venv)**
Para el desarrollo local, los desarrolladores configuran un entorno virtual utilizando venv. Esto permite **aislar las dependencias del proyecto** y asegurar que todos trabajen en un entorno consistente.
 - Se utiliza el archivo **requirements.txt** para instalar todas las dependencias necesarias con un solo comando (`pip install -r requirements.txt`).
 - Este entorno permite a los desarrolladores probar todas las funcionalidades de la aplicación **localmente** antes de realizar un **push** al repositorio.
- **Pruebas Locales**
Antes de realizar un commit, los desarrolladores ejecutan **pruebas automatizadas** localmente para validar que los cambios no introduzcan errores. Esto garantiza que solo código funcional llegue al repositorio.

2. Despliegue Remoto en Render

- **Integración Continua con GitHub Actions**
El proceso de despliegue remoto comienza una vez que el código ha pasado todas las pruebas de **integración continua** configuradas en **GitHub Actions**.
 - Cada vez que se realiza un **push** o un **pull request** a 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.
 - Si todas las pruebas son exitosas, se procede al despliegue automático en Render.
- **Despliegue en Render**
Render es utilizado como plataforma de **despliegue en la nube** para gestionar la implementación continua de la aplicación sin intervención manual.
 - Render garantiza que la aplicación sea accesible en un **entorno remoto**, replicando fielmente el entorno de producción para **pruebas finales** antes del lanzamiento.
 - La infraestructura y el despliegue son gestionados automáticamente por Render, lo que simplifica el proceso de implementación y reduce el riesgo de errores.



Automatización de la Entrega y Despliegue

3. Contenerización

Uso de Docker

Debido a la necesidad de asegurar un entorno consistente, Docker se utiliza en este proyecto para contenerizar la aplicación.

- Docker proporciona un entorno controlado y reproducible tanto para el desarrollo local como para el despliegue en producción.
- Facilita la portabilidad del proyecto y asegura que la aplicación se ejecute de la misma forma en todos los entornos.
- Simplifica el proceso de despliegue al reducir las diferencias entre entornos de desarrollo y producción, mejorando la eficiencia y estabilidad del sistema.



4. Resultados Esperado

La implementación de un enfoque automatizado para la **entrega y despliegue** en el proyecto **innosoft-diplomas-1** tiene como objetivo alcanzar los siguientes **resultados esperados**:

1. Reducción de Riesgos en el Despliegue

- Minimizar los **errores manuales** al automatizar el proceso de integración y despliegue, lo que asegura que cada actualización sea consistente y libre de fallos.
- Garantizar que el código que se despliega ha pasado por un proceso de **validación exhaustivo** mediante pruebas automatizadas.

2. Asegurar la Consistencia entre Entornos

- Replicar fielmente el **entorno de producción** en los entornos de prueba y preproducción, lo que permite identificar y corregir problemas antes del despliegue final.
- Asegurar que el entorno de **producción sea una réplica exacta** del entorno de desarrollo y pruebas, evitando discrepancias que puedan causar errores en producción.

3. Mejora en la Velocidad y Eficiencia del Despliegue

- Facilitar un proceso de **despliegue ágil y continuo**, permitiendo al equipo entregar nuevas funcionalidades y correcciones de manera rápida.
- Reducir el tiempo de inactividad durante los despliegues, garantizando que la aplicación esté siempre disponible para los usuarios.

4. Optimización del Ciclo de Desarrollo

- **Automatizar las pruebas** y el despliegue libera tiempo a los desarrolladores, permitiéndoles enfocarse en la creación de nuevas funcionalidades en lugar de tareas manuales repetitivas.
- Aumentar la **eficiencia del equipo** al contar con un proceso automatizado que valida el código antes de su integración en producción.

5. Mejor Experiencia de Usuario

- Asegurar que todas las actualizaciones sean **estables y sin errores**, lo que mejora la experiencia del usuario final.
- Minimizar la posibilidad de fallos en producción al realizar una **validación exhaustiva** antes del despliegue.

6. Facilitar la Entrega Continua (Continuous Delivery)

- Mantener un **flujo de entrega continua** para que nuevas funcionalidades, mejoras y correcciones de errores puedan ser desplegadas de forma regular y segura.
- Permitir un **despliegue controlado** que se ejecute solo cuando el código haya sido completamente validado y aprobado.