David Recio Arnés

Entornos Integración y Entrega Continua

desarrollo de pipeline de Jenkins

Contenido

[Introducción 2](#_Toc201506660)

[Entorno de Trabajo 2](#_Toc201506661)

[Configuración de Jenkins 4](#_Toc201506662)

[Jenkinsfile explicado 5](#_Toc201506663)

[Resultados 7](#_Toc201506664)

[Bibliografía 9](#_Toc201506665)

# Introducción

En esta práctica se ha desarrollado un pipeline en Jenkins con el objetivo de automatizar tareas dentro del proceso de integración continua, abarcando etapas como la construcción del proyecto, la ejecución de pruebas unitarias, pruebas de API y pruebas de extremo a extremo (E2E). Para ello, se ha tomado como base el repositorio público srayuso/unir-cicd, el cual ha sido ampliado para incorporar nuevas funcionalidades. Entre las mejoras implementadas se encuentran la inclusión de etapas específicas para la ejecución de pruebas de API y E2E, el archivado automático de los resultados en formato XML, la visualización de informes generados por dichas pruebas y la configuración de un sistema de notificaciones por correo electrónico, el cual se activa únicamente cuando la ejecución del *pipeline* finaliza con errores, incluyendo en el mensaje el nombre del trabajo y el número de ejecución correspondiente.

# Entorno de Trabajo

Para la elaboración de la práctica se ha creado un proyecto sencillo que permite desplegar y configurar un servidor Jenkins utilizando Docker. Es importante destacar que se ha empleado una imagen personalizada de Jenkins, ya que la imagen oficial no incluye todos los componentes necesarios para el desarrollo completo de la actividad. Para levantar el entorno, se ha utilizado la herramienta Docker Desktop como plataforma de virtualización y gestión de contenedores.

Texto

El contenido generado por IA puede ser incorrecto.

Tal como se observa en la imagen, se instalan una serie de dependencias necesarias para el proceso de *build*, además de copiar la configuración inicial de Jenkins para asegurar que el entorno esté correctamente preparado desde el primer arranque.

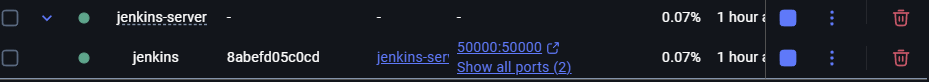
Texto

El contenido generado por IA puede ser incorrecto.

En esta imagen se muestra la configuración de una cuenta de usuario predefinida para acceder al panel de Jenkins, facilitando así el inicio de sesión sin necesidad de crear credenciales manualmente tras el despliegue.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.



Cabe mencionar que, debido a que en la entrega únicamente se permite incluir el Jenkinsfile y esta documentación, en el apartado de bibliografía se proporcionarán todos los enlaces a los repositorios públicos utilizados, tanto el correspondiente al servidor Jenkins como el del proyecto desarrollado para la práctica.

# Configuración de Jenkins

Una vez dentro del panel de Jenkins, se procedió a la instalación de los plugins sugeridos por defecto. Tras completar este paso, se creó un *job* de tipo *Pipeline* llamado **“práctica\_despliegue”**. Esta *pipeline* está configurada para obtener su definición desde un sistema de control de versiones (SCM), seleccionando la opción *Pipeline script from SCM* y utilizando Git como sistema de origen, ya que estará asociado a un repositorio público. Es importante señalar que, en caso de tratarse de un repositorio privado, sería necesario configurar las credenciales adecuadas para permitir el acceso. Se especifica la rama desde la cual se leerá el archivo Jenkinsfile, en este caso "\*/main", y finalmente se indica el nombre del fichero que debe ejecutarse: jenkinsfile.pract.groovy.

Interfaz de usuario gráfica, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

# Jenkinsfile explicado

Para la elaboración de l fichero de jenkis se ha hecho uso de la plantilla establecida en la practica de la cual se ha completado en base a lo requerido en la misma.

Texto

El contenido generado por IA puede ser incorrecto.

Respecto a los workers se ha establecido a cualquier agente haciendo uso de agent any.

Después se han establecido las siguientes etapas:

* Source: en esta etapa se realiza el clonado del repositorio del código
* Build: en esta etapa se construye el aplicativo
* Unit test: en esta etapa se realizan las pruebas unitarias
* Api Test: en esta etapa se realizan las pruebas de aplicación
* E2E test: en esta etapa se realizan las pruebas end to end

Texto

El contenido generado por IA puede ser incorrecto.

Por último, se define una sección de post-ejecución que se activa una vez finalizadas las fases principales del pipeline. En esta etapa, se realiza la publicación de los resultados de las pruebas ejecutadas y, en caso de que la ejecución haya fallado, se envía automáticamente un correo electrónico notificando el fallo. Dicho correo incluye el nombre del job y el número de ejecución que ha generado el error, facilitando así la identificación y el seguimiento del problema.

# Resultados

En esta sección se mostrarán las imágenes que corroboran que se realizo correctamente la practica atendiendo a los objetivos de esta.

Texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

# Bibliografía

* Recio, D. [@davidRecio]. (s.f.). *labJenkins* [Repositorio GitHub]. GitHub. <https://github.com/davidRecio/labJenkins>
* Afraileni, J. [@jafraileni]. (s.f.). *unir-cicd* [Repositorio GitHub]. GitHub. <https://github.com/jafraileni/unir-cicd>
* Recio, D. [@davidRecio]. (s.f.). *servidor\_jenkin* [Repositorio GitHub]. GitHub. <https://github.com/davidRecio/servidor_jenkin>