



Jenkins

■ INTEGRACIÓN CONTINUA CON JENKINS

1

INTRODUCCIÓN A JENKINS



CI/CD INTRODUCCIÓN

- **La integración continua** (*Continuous Integration* o **CI**, por sus siglas en inglés) es una práctica de desarrollo de software que consiste en integrar frecuentemente los cambios de código en un repositorio compartido, idealmente varias veces al día.
- Cada integración es verificada automáticamente mediante scripts de compilación y pruebas automatizadas para detectar errores lo antes posible.
- **Objetivos principales:**
 - Detectar errores rápidamente.
 - Reducir conflictos de integración.
 - Automatizar pruebas y validaciones.
 - Acelerar el desarrollo y entrega de software.

- Elementos clave de la integración continua:
 - Repositorio de código centralizado (como GitHub o GitLab).
 - Automatización de builds (compilaciones del software).
 - Pruebas automatizadas (unitarias, de integración, etc.).
 - Notificaciones de fallos (para que los desarrolladores actúen de inmediato).
 - Herramientas de CI como Jenkins, GitHub Actions, GitLab CI/CD, CircleCI, Travis CI, entre otras.

- Ejemplo simple del flujo:
 - Un desarrollador hace un cambio en el código.
 - Sube esos cambios (push) al repositorio.
 - El sistema de CI se activa automáticamente.
 - Se ejecutan scripts para compilar el proyecto y correr pruebas.
 - Si todo pasa, los cambios pueden integrarse sin problemas.
 - Si falla algo, se notifica para corregirlo de inmediato.

- **La entrega continua** (*Continuous Delivery* o **CD**, por sus siglas en inglés) es una práctica dentro del desarrollo ágil y DevOps que implica mantener el software en un estado donde siempre esté listo para ser desplegado en producción.
- Su objetivo principal es automatizar y acelerar el proceso de entrega, asegurando que cada cambio en el código pase por pruebas automáticas y otras validaciones antes de ser implementado.
- A veces también significa **Despliegue Continuo** (*Continuous Deployment*). Es el siguiente nivel después de la entrega continua: cada cambio que pasa las pruebas automáticas se despliega automáticamente en producción sin intervención manual.

INTRODUCCIÓN A JENKINS



- Enlaces:

- <https://about.gitlab.com/es/topics/ci-cd/>
- <https://unity.com/es/topics/what-is-ci-cd>

INTRODUCCIÓN A JENKINS



INTRODUCCIÓN A JENKINS Y CARACTERÍSTICAS

INTRODUCCIÓN A JENKINS



- Jenkins es una herramienta de automatización de código abierto muy utilizada para la integración continua (CI) y la entrega continua (CD) en el desarrollo de software. Está escrita en Java y permite automatizar diferentes partes del proceso de desarrollo, como compilación, pruebas, empaquetado y despliegue de aplicaciones.
- Jenkins ayuda a los equipos de desarrollo a:
 - Automatizar tareas repetitivas en el ciclo de vida del desarrollo.
 - Detectar errores de forma temprana mediante pruebas automáticas.
 - Integrar continuamente cambios de código en un repositorio compartido.
 - Entregar software más rápidamente y con mayor confiabilidad.
- Se basa en un sistema de jobs o pipelines, que son conjuntos de instrucciones que Jenkins ejecuta de forma automática al detectar un evento, como un cambio de código en un repositorio.

INTRODUCCIÓN A JENKINS



- Características principales de Jenkins
 - **Automatización de procesos CI/CD.** Ejecuta tareas automáticamente cuando se detectan cambios en el código (push a Git, por ejemplo).
 - **Soporte de plugins extensivo.** Cuenta con más de 1,800 plugins para integrar herramientas como Git, Maven, Docker, Kubernetes, Slack, SonarQube, entre muchas otras.
 - **Pipelines declarativos y scriptados.** Permite definir flujos de trabajo como código (Jenkinsfile), lo que facilita la trazabilidad y control de versiones.
 - **Interfaz web amigable.** Ofrece una interfaz gráfica para gestionar trabajos, visualizar logs, resultados de pruebas y configuraciones.
 - **Gestión de nodos (agentes esclavos).** Puede distribuir la carga de trabajo entre múltiples servidores/agentes para mejorar el rendimiento.
 - **Integración con sistemas de control de versiones.** Compatible con Git, Subversion, Mercurial, etc., para detectar automáticamente cambios en el código.
 - **Notificaciones y reportes.** Envía alertas por correo electrónico, Slack, entre otros canales, y muestra reportes de pruebas, cobertura de código, etc.
 - **Código abierto y comunidad activa.** Es gratuito y cuenta con una comunidad grande y activa que colabora constantemente en su desarrollo y mantenimiento.

INTRODUCCIÓN A JENKINS



HOLA MUNDO JENKINS

INTRODUCCIÓN A JENKINS



- **Objetivo:**
 - Validar, compilar y desplegar una aplicación Spring REST en un contenedor Docker remoto.
- **Requisitos:**
 - Repositorio de GitHub con tu aplicación Spring Boot.
 - Cuenta en Render.com y un servicio Web configurado para tu app.
 - Servidor Jenkins accesible (local o en la nube).

INTRODUCCIÓN A JENKINS



■ Generar token en GitHub:

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

token-demo-jenkins

What's this token for?

Expiration

📅 30 days (Jun 11, 2025) ▾

The token will expire on the selected date

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

INTRODUCCIÓN A JENKINS



Administrar Jenkins - Jenkins x +

localhost:8080/manage/

Jenkins

Panel de Control > Administrar Jenkins

+ Nueva Tarea

Historial de trabajos

Administrar Jenkins

Mis vistas

Trabajos en la cola

No hay trabajos en la cola

Estado del ejecutor de construcciones 0/2

Administrar Jenkins

System Configuration

- System**
Configurar variables globales y rutas.
- Tools**
Configure tools, their locations, automatic installers.
- Clouds**
Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**
Configure the look and feel of J

Security

- Security**
Seguridad en Jenkins. Define quién tiene acceso al sistema (autenticación) y qué puede hacer (autorización)
- Credentials**
Configure credentials

INTRODUCCIÓN A JENKINS



Credentials

T P Store ↓ Domain

Stores scoped to Jenkins

P Store ↓

System

Icono: S M L

Domains

(global) ▼

Add credentials

Credenciales
GitHub

Panel de Control > Administrar Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

fpanaguajava

The username.

☐ Treat username as secret ?

Password ?

.....

ID ?

git-creds-id

Description ?

Credenciales de para GIT

Create

INTRODUCCIÓN A JENKINS



demo_rest_ws_jenkins · Web Se x +

dashboard.render.com/web/srv-d0gpvj3uibrs73frsqag/settings

My Workspace demo_rest_ws_jenkins Search + K + New Upgrade ? F

demo_rest_ws_jenkins Edit

Deploy Hook
Your private URL to trigger a deploy for this server. Remember to keep this a secret.

.....

Regenerate hook

Custom Domains
Your service is always available at <https://demo-rest-ws-jenkins.onrender.com>
You can also point [custom domains](#) you own to this service. See [DNS configuration instructions](#).

+ Add Custom Domain

Obtener el
deploy Hook de
Render

INTRODUCCIÓN A JENKINS



Credentials

T	P	Store	Domain
---	---	-------	--------

Stores scoped to Jenkins

PStore

System

Domains

(global)

Add credentials

Icono: SML

Credenciales
Render

Deploy hook
de Render

Panel de Control > Administrar Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

render-deploy-hook

This ID is already in use

Description ?

Deploy hook de Render

Create

INTRODUCCIÓN A JENKINS



The screenshot shows the Jenkins administration page in a web browser. The browser's address bar displays 'localhost:8080/manage/'. The Jenkins logo and name are at the top left, and a search bar and user profile 'Fernando Paniagua' are at the top right. A 'Desconectar' button is also present. Below the header, a breadcrumb trail shows 'Panel de Control > Administrar Jenkins'. On the left sidebar, the 'Nueva Tarea' button is highlighted with a red rectangle. Below it are links for 'Historial de trabajos', 'Administrar Jenkins' (selected), and 'Mis vistas'. A 'Trabajos en la cola' section shows 'No hay trabajos en la cola'. A 'Estado del ejecutor de construcciones' section shows '0/2'. The main content area is titled 'Administrar Jenkins' and contains a 'Search settings' bar. It is divided into two sections: 'System Configuration' and 'Security'. The 'System Configuration' section includes links for 'System' (Configurar variables globales y rutas.), 'Tools' (Configure tools, their locations and automatic installers.), 'Plugins' (Añadir, borrar, desactivar y activar plugins que extienden la funcionalidad de Jenkins.), 'Nodes' (Añadir, borrar, gestionar y monitorizar los nodos sobre los que Jenkins ejecuta tareas.), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand.), and 'Appearance' (Configure the look and feel of Jenkins). The 'Security' section includes links for 'Security' and 'Credentials'.

INTRODUCCIÓN A JENKINS



Nuevo Tarea - Jenkins

localhost:8080/newJob

Jenkins


Panel de Control > Nuevo Tarea


Nuevo Tarea


Enter an item name


demo-task

Select an item type

 Crear un proyecto de estilo libre
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

 Pipeline
Gestiona actividades de larga duración que pueden abarcar varios agentes de construcción. Apropiado para construir pipelines (conocidas anteriormente como workflows) y/o para la organización de actividades complejas que no se pueden articular fácilmente con tareas de tipo freestyle.

 Crear un proyecto multi-configuración
Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en múltiples entornos, ejecutar sobre plataformas concretas, etc.

 Folder

OK

INTRODUCCIÓN A JENKINS



■ Crear el script del Pipeline

Panel de Control > demo-task > Configuration

☐ Lanzar ejecuciones remotas (ejem: desde 'scripts') ?

Configure

- General
- Triggers
- Pipeline**
- Advanced

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
1 pipeline {
2   agent any
3   stages {
4     stage('Checkout') {
5       steps {
6         git branch: 'main', url: 'https://github.com'
7       }
8     }
9     stage('Build') {
10      steps {
11        bat './mvnw clean package -DskipTests=true'
12      }
13    }
14    stage('Test') {
15      steps {
```

The script is already approved

INTRODUCCIÓN A JENKINS



■ Script:

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        git branch: 'main', url: 'https://github.com/fpaniaguajava/demo_rest_ws_jenkins.git'
      }
    }
    stage('Build') {
      steps {
        bat './mvnw clean package -DskipTests=true'
      }
    }
    stage('Test') {
      steps {
        bat './mvnw test'
      }
    }
  }
}
```

INTRODUCCIÓN A JENKINS



■ Script:

```
stage('Deploy to Render') {
    steps {
        withCredentials([string(credentialsId: 'render-deploy-hook', variable: 'RENDER_DEPLOY_HOOK')]) {
            bat 'curl -X POST %RENDER_DEPLOY_HOOK%'
        }
    }
}
stage('Push to Git (Render monitored branch)') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'git-creds-id', usernameVariable: 'GIT_USER',
passwordVariable: 'GIT_PASS')]) {
            bat """
            git config user.name "%GIT_USER%"
            git config user.email "fernando.paniagua.java@gmail.com"
            git add .
            git commit -m "Deploy from Jenkins"
            git push https://$GIT_USER:$GIT_PASS@github.com/fpaniaguajava/demo_rest_ws_jenkins HEAD:main
            """
        }
    }
}
}
```

INTRODUCCIÓN A JENKINS



- Estado de la tarea:

The screenshot displays the Jenkins web interface. At the top, there's a black header with the Jenkins logo and name. Below it, a breadcrumb trail shows 'Panel de Control' followed by 'demo-task'. The main content area is divided into two columns. The left column contains a sidebar with various actions: 'Status' (highlighted), 'Changes', 'Construir ahora', 'Configurar', 'Borrar Pipeline', 'Stages', 'Rename', and 'Pipeline Syntax'. The right column shows the 'demo-task' status with a green checkmark icon, the text 'Despliegue en Render.', and a section titled 'Enlaces permanentes' containing a list of links to specific build executions. At the bottom, there's a 'Builds' section with a list of build entries.

Jenkins

Panel de Control > demo-task >

Status

- </> Changes
- ▶ Construir ahora
- ⚙ Configurar
- 🗑 Borrar Pipeline
- 📄 Stages
- ✎ Rename
- ❓ Pipeline Syntax

demo-task

Despliegue en Render.

Enlaces permanentes

- "Última ejecución (#36) hace 15 Min"
- "Última ejecución estable (#36) hace 15 Min"
- "Última ejecución correcta (#36) hace 15 Min"
- "Última ejecución fallida (#29) hace 53 Min"
- "Última ejecución fallida (#29) hace 53 Min"
- "Last completed build (#36) hace 15 Min"

Builds

INTRODUCCIÓN A JENKINS



- Historial de cambios:

Jenkins

Panel de Control > demo-task >

- Status
- </> Changes**
- ▶ Construir ahora
- ⚙ Configurar
- 🗑 Borrar Pipeline
- 📁 Stages
- ✎ Rename
- ❓ Pipeline Syntax

Changes

- #36 (13 may 2025, 8:27:16)
 - 1. Deploy from Jenkins — [fernando.paniagua.java](#) / [githubweb](#)
- #35 (13 may 2025, 8:25:07)
 - 1. Deploy from Jenkins — [fernando.paniagua.java](#) / [githubweb](#)
- #34 (13 may 2025, 8:23:27)
 - 1. Update DemoController.java — [fernando.paniagua.java](#) / [githubweb](#)
- #33 (13 may 2025, 8:04:24)
 - 1. Update DemoController.java — [fernando.paniagua.java](#) / [githubweb](#)

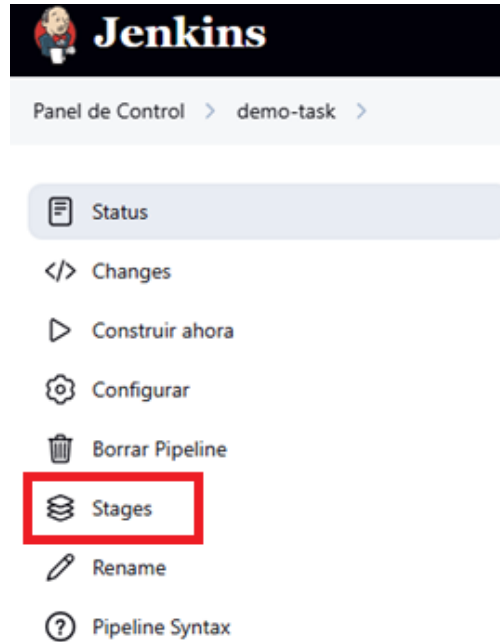
Builds

INTRODUCCIÓN A JENKINS



Jenkins

■ Stages:



Panel de Control > demo-task > Stages

Stages

Stages

13 de mayo de 2025

✓ #37	08:44 - 21 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✓ #36	08:27 - 22 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✓ #35	08:25 - 22 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✓ #34	08:23 - 23 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✓ #33	08:04 - 20 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✓ #32	07:58 - 21 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✓ #31	07:54 - 18 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✓ #30	07:49 - 18 Seg	○ - ✓ - ✓ - ✓ - ✓ - ✓ - ○
✗ #29	07:48 - 0,2 Seg	○ - ○
✗ #28	07:44 - 17 Seg	○ - ✓ - ✓ - ✓ - ✗ - >> - ○

INTRODUCCIÓN A JENKINS



■ Acceso al detalle de los builds:

Panel de Control > demo-task >

Status

</> Changes

▶ Construir ahora

⚙ Configurar

🗑 Borrar Pipeline

📁 Stages

✎ Rename

❓ Pipeline Syntax

Builds

🔍 Filter

Today

- ✅ #37 8:44
- ✅ #36 8:27
- ✅ #35 8:25
- ✅ #34 8:23
- ✅ #33 8:04
- ✅ #32 7:58
- ✅ #31 7:54



Jenkins

Panel de Control > demo-task > #37

Status

</> Changes

📄 Console Output

📝 Edit Build Information

🗑 Delete build '#37'

🕒 Timings

🔍 Git Build Data

🔗 Pipeline Overview

🔄 Restart from Stage

🔄 Replay

📋 Pipeline Steps

📁 Workspaces

← Previous Build

✅ #37 (13 may 2025, 8:44:21)

🕒 Iniciado por el usuario [Fernando Paniagua](#)

🕒 This run spent:

- 6 Ms waiting;
- 21 Seg build duration;
- 21 Seg total from scheduled to completion.

🔍 git

Revision: 8c5534a3d00e0cd40e5cf6297088449acdd93884

Repository: https://github.com/fpaniguajava/demo_rest_ws_jenkins.git

- refs/remotes/origin/main

</> Sin cambios

INTRODUCCIÓN A JENKINS



JENKINS CORE

- El núcleo de Jenkins (Jenkins Core) es el sistema base que proporciona las funcionalidades esenciales de Jenkins, tales como:
 - Interfaz web de usuario (UI)
 - Programación y ejecución de tareas (builds)
 - Gestión básica de usuarios y permisos
 - Integración con sistemas de control de versiones (como Git)
 - Soporte para extensiones mediante plugins
- La mayoría de las funcionalidades avanzadas se añaden a través de plugins

- Agentes (antes llamados *Slaves*):
 - Los agentes son máquinas (físicas o virtuales) que Jenkins utiliza para ejecutar tareas.
 - El servidor principal (Master o ahora llamado Controller) coordina todo, pero puede delegar la ejecución de trabajos a los agentes.
 - Los agentes permiten distribuir la carga y ejecutar trabajos en diferentes entornos (por ejemplo, distintos sistemas operativos o configuraciones).

- Trabajos (Jobs)
 - Un trabajo (job) en Jenkins es una unidad de trabajo que define qué se debe hacer:
 - Compilar código
 - Ejecutar pruebas
 - Desplegar aplicaciones
 - Ejecutar scripts
 - Existen varios tipos. Los principales son:
 - Freestyle Project: configuraciones simples y directas
 - Pipeline: trabajos definidos como código (archivo Jenkinsfile), más flexibles y potentes
 - Multibranch Pipeline: para manejar múltiples ramas automáticamente desde un repositorio
 - Folder: para organizar trabajos

- Plugins
 - Los plugins son extensiones que permiten ampliar las capacidades de Jenkins.
 - Jenkins es altamente modular y su ecosistema de plugins cubre áreas como:
 - Integración con herramientas de CI/CD (Docker, GitHub, Maven, etc.)
 - Notificaciones (Slack, correo, etc.)
 - SeguridadVisualización de resultadosSoporte para nuevas tecnologías (Kubernetes, Terraform, etc.)

INTRODUCCIÓN A JENKINS



JENKINS JOBS

INTRODUCCIÓN A JENKINS



▪ JOBS: Tipos:

▪ **Estilo libre**

- Tipo de trabajo clásico y de propósito general que obtiene código desde un único sistema de control de versiones (SCM), ejecuta pasos de construcción de forma secuencial, seguido por pasos posteriores como archivado de artefactos y envío de notificaciones por correo electrónico.

▪ **Pipeline**

- Gestiona actividades de larga duración que pueden abarcar varios agentes de construcción. Apropiado para construir pipelines (conocidas anteriormente como workflows) y/o para la organización de actividades complejas que no se pueden articular fácilmente con tareas de tipo freestyle.

▪ **Crear un proyecto multi-configuración**

- Adecuado para proyectos que requieren un gran número de configuraciones diferentes, como testear en múltiples entornos, ejecutar sobre plataformas concretas, etc.

▪ **Carpeta (Folder)**

- Crea un contenedor que almacena elementos anidados en su interior. Es útil para agrupar cosas relacionadas. A diferencia de una vista, que solo es un filtro, una carpeta crea un espacio de nombres separado, por lo que puedes tener elementos con el mismo nombre siempre que estén en diferentes carpetas.

▪ **Multibranch Pipeline**

- Crea un conjunto de proyectos Pipeline según las ramas detectadas en un único repositorio SCM.

▪ **Organización Folder**

- Crea un conjunto de subcarpetas de proyectos Multibranch escaneando repositorios por organizaciones.

INTRODUCCIÓN A JENKINS



JENKINS PLUGINS

INTRODUCCIÓN A JENKINS



- Los plug-ins de Jenkins son extensiones que amplían la funcionalidad del núcleo de Jenkins.
- Por defecto, se pueden instalar los más habituales (ver sección de plug-ins).
- Algunos tipos:
- **Proveedores de nube (Cloud providers):**
 - Permiten que Jenkins se integre con diferentes plataformas de cómputo en la nube para:
 - Provisionar agentes (nodos esclavos) dinámicamente.
 - Ejecutar trabajos en entornos escalables y bajo demanda.
 - Ejemplos:
 - Docker/Mesos/Kubernetes: Inician contenedores o pods como agentes.
 - EC2/OpenStack/Azure: Lanzan máquinas virtuales según la carga de trabajo.
- **Estadísticas de la nube (Cloud statistics):**
 - Muestran estadísticas relacionadas con el uso de los agentes en la nube:
 - Número de nodos activos.
 - Historial de ejecuciones.
 - Tiempos de provisión de recursos.
- **Hito (Milestone):**
 - Controlan el avance dentro de pipelines para:
 - Evitar que etapas anteriores se ejecuten si una etapa más avanzada ya ha sido alcanzada.
 - Prevenir condiciones de carrera o redundancias en pipelines paralelos.
 - Útil en entornos donde múltiples ejecuciones pueden superponerse (como despliegues frecuentes).

- **Recursos bloqueables (Lockable resources):**

- Gestionan recursos compartidos (como bases de datos, hardware físico o licencias) para:
 - Evitar conflictos entre trabajos que los usen simultáneamente.
 - Asegurar exclusividad durante su uso.

- **Métricas (Metrics):**

- Ofrecen métricas internas de Jenkins, como:
 - Tiempos de respuesta.
 - Consumo de memoria/CPU.
 - Tiempo de ejecución de jobs.
- Se puede integrar con herramientas externas como Prometheus o Grafana para monitoreo avanzado.

- **Notificaciones (Notifications):**

- Permiten enviar alertas o actualizaciones sobre el estado de los jobs o pipelines.
 - Mail: Notifica por correo electrónico.
 - Slack: Envía mensajes a canales de Slack, útil para equipos ágiles.

- **Control de versiones (SCM - Source Control Management):**
 - Conectan Jenkins con sistemas de control de versiones para:
 - Obtener código fuente.
 - Disparar builds cuando hay cambios.
 - Integrar ramas y commits con pipelines.
 - Ejemplos: Git, Mercurial o Sistema de archivos (Para proyectos locales sin repositorio remoto).
- **Ecosistema de pipelines (Pipeline ecosystem):**
 - Permiten construir y gestionar pipelines de CI/CD con Jenkins.
- **BlueOcean:**
 - Una interfaz visual moderna para Jenkins:
 - Hace que los pipelines sean más intuitivos y fáciles de entender.
 - Muestra visualmente las etapas, errores, duración y resultados.

INTRODUCCIÓN A JENKINS



PIPELINES

- Los pipelines en Jenkins son flujos de trabajo automatizados definidos como código (generalmente en un archivo Jenkinsfile). Permiten modelar procesos completos de CI/CD (Integración y Entrega Continua) en etapas, como: compilar, testear, validar, desplegar, etc.
- Jenkins Pipeline es un conjunto de complementos que permite implementar e integrar pipelines de entrega continua en Jenkins. Pipeline proporciona un conjunto extensible de herramientas para modelar pipelines de entrega, desde simples hasta complejos.

- Un Pipeline se puede crear de las siguientes maneras:
 - Mediante Blue Ocean.
 - <https://www.jenkins.io/doc/book/blueocean/getting-started/>
 - Mediante la UI clásica.
 - En el SCM – escribiendo el fichero Jenkinsfile manualmente, del que se podrá hacer commit al repositorio de control del código fuente del proyecto.

■ Estructura del Pipeline:

```
pipeline {  
    agent any    // Define en qué agente (nodo) se ejecutará el pipeline  
  
    environment {  
        // Variables de entorno, si se necesitan  
        VAR1 = 'valor1'  
        VAR2 = 'valor2'  
    }  
  
    options {  
        // Opciones del pipeline (ej. timeout, timestamps, etc.)  
        timeout(time: 1, unit: 'HOURS')  
        timestamps()  
    }  
  
    parameters {  
        // Parámetros para ejecuciones manuales  
        string(name: 'BRANCH', defaultValue: 'main', description: 'Rama del repositorio')  
        booleanParam(name: 'RUN_TESTS', defaultValue: true, description: '¿Ejecutar pruebas?')  
    }  
}
```

■ Estructura del Pipeline:

```
stages {
    stage('Preparación') {
        steps {
            echo 'Clonando repositorio...'
            git branch: "${params.BRANCH}", url: 'https://tu-repo.git'
        }
    }

    stage('Compilación') {
        steps {
            echo 'Compilando aplicación...'
            // comandos de build, ej: mvn clean package
            sh './build.sh'
        }
    }

    stage('Pruebas') {
        when {
            expression { return params.RUN_TESTS }
        }
        steps {
            echo 'Ejecutando pruebas unitarias...'
            sh './run-tests.sh'
        }
    }

    stage('Despliegue') {
        steps {
            echo 'Desplegando aplicación...'
            sh './deploy.sh'
        }
    }
}
```

INTRODUCCIÓN A JENKINS



■ Estructura del Pipeline:

```
post {  
    always {  
        echo 'Pipeline finalizado.'  
    }  
    success {  
        echo 'Pipeline ejecutado con éxito.'  
    }  
    failure {  
        echo 'El pipeline falló.'  
    }  
}  
}
```

INTRODUCCIÓN A JENKINS



ENVÍO DE EMAIL

INTRODUCCIÓN A JENKINS



- Añadir el envío de email en el pipeline.
- Comprobar plugins:

Nombre ↓	
Email Extension Plugin 1876.v28d8d38315b_d	This plugin is a replacement for Jenkins's email publisher. It allows to configure every aspect of email notifications Report an issue with this plugin
Git plugin 5.7.0	This plugin integrates Git with Jenkins. Report an issue with this plugin
Jakarta Mail API 2.1.3-2	This plugin provides the Jakarta Mail API for other plugins. Report an issue with this plugin
LDAP Plugin 780.vcb_33c9a_e4332	Adds LDAP authentication to Jenkins Report an issue with this plugin
Mailer Plugin 489.vd4b_25144138f	This plugin allows you to configure email notifications for build results. Report an issue with this plugin

- Añadir el envío de email en el pipeline.
- Configurar Jenkins con la cuenta de correo:
 - Administrar Jenkins → Credentials
 - En Google hay que habilitar el doble factor de autenticación y generar un token en:
 - <https://security.google.com/settings/security/apppasswords>
 - Administrar Jenkins → System → Notificación por correo electrónico

- **Añadir el envío de email en el pipeline.**
- Ejemplo de configuración en Jenkins:
- Sección “Extended e-mail-notification”
 - Plugin requerido: Email Extension Plugin (más completo).
 - En Pipelines: Se usa con emailext
- Sección “E-mail-notification”
 - Plugin requerido: Mailer Plugin (básico).
 - Se usa con la directiva mail en el pipeline.

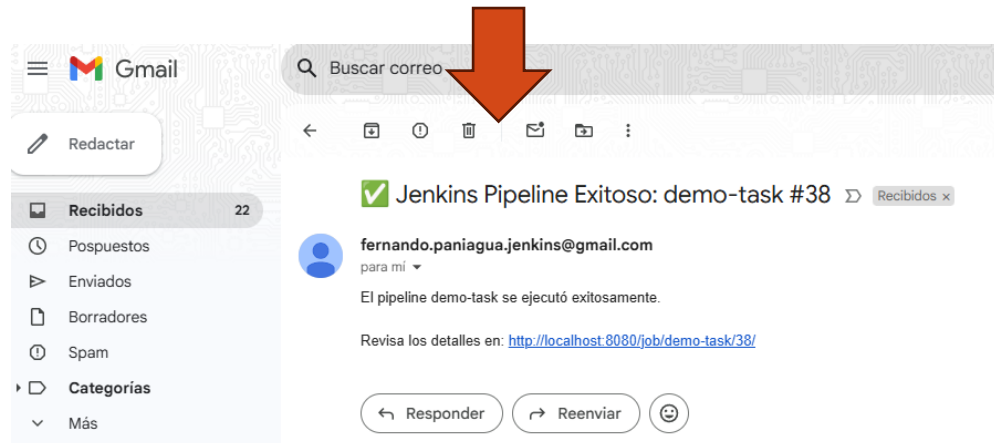
- Añadir el envío de email en el pipeline.
- Sección “E-mail-notification”
 - SMTP server → smtp.gmail.com
 - Use SMTP Authentication
 - Username → tuusuario@gmail.com
 - Password → (la contraseña de aplicación)
 - Use TLS (STARTTLS)
 - SMTP Port → 587
 - Reply-To Address → tuusuario@gmail.com
 - Charset → UTF-8

INTRODUCCIÓN A JENKINS



- Añadir el envío de email en el pipeline.
- Añadir la fase post en el descriptor del pipeline:

```
post {  
    success {  
        mail to: 'fernando.paniagua@gmail.com',  
              subject: "✅ Jenkins Pipeline Exitoso: ${env.JOB_NAME} #${env.BUILD_NUMBER}",  
              body: "El pipeline ${env.JOB_NAME} se ejecutó exitosamente.\n\nRevisa los detalles en: ${env.BUILD_URL}"  
    }  
    failure {  
        mail to: 'fernando.paniagua@gmail.com',  
              subject: "❌ Jenkins Pipeline Fallido: ${env.JOB_NAME} #${env.BUILD_NUMBER}",  
              body: "El pipeline ${env.JOB_NAME} falló.\n\nRevisa los detalles en: ${env.BUILD_URL}"  
    }  
}
```



INTRODUCCIÓN A JENKINS



■ Enlaces:

- Jenkins: <https://www.jenkins.io/>
- Jenkins download: <https://www.jenkins.io/download/>
- Jenkins Handbook: <https://www.jenkins.io/doc/book/getting-started/>
- Instalación de Jenkins: <https://www.jenkins.io/doc/book/installing/>