



Jenkins

Tutorial Básico de Jenkins

Jenkins es una herramienta destinada a la automatización en el ciclo de vida del desarrollo de software. Jenkins actúa como el motor central para los flujos de trabajo de Integración Continua (CI) y Entrega Continua (CD) que son etapas fundamentales en las metodologías DevOps, el enfoque de desarrollo de software que busca integrar los equipos de desarrollo (Dev) y operaciones (Ops) para automatizar y mejorar el proceso de entrega de aplicaciones.

DSSD

¿Qué es Jenkins?



Servidor de Automatización

Jenkins es un servidor de automatización de código abierto, desarrollado en Java, diseñado para orquestar cada etapa del ciclo de vida del desarrollo de software.



Integración Continua (CI)

Habilita la integración continua al automatizar y monitorear los procesos de construcción (build) y prueba (test) de código, asegurando la calidad desde fases tempranas.



Entrega Continua (CD)

Facilita la entrega continua al automatizar los despliegues de software a diversos entornos, desde desarrollo hasta producción, de manera rápida y fiable.



Extensible y Versátil

Es altamente extensible ya que permite instalar una gran variedad de plugins disponibles y ofrece soporte para múltiples sistemas operativos como Linux, Windows, macOS y Docker.

En el contexto del desarrollo e implementación de sistemas se hace necesaria la automatización de diferentes actividades dentro del ciclo de vida del desarrollo, como por ejemplo actualizar código, correr scripts de bd, correr determinadas pruebas, actualizar componentes externos, etc.

Instalación de Jenkins

Requisitos Esenciales

Jenkins se puede instalar a través de paquetes de sistema nativos, Docker o incluso ejecutarse de forma independiente mediante cualquier entorno de ejecución Java.

Opciones de Instalación

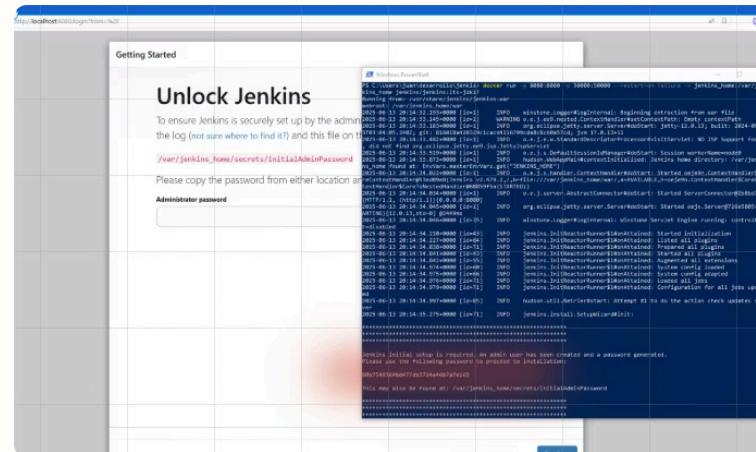
- Archivo .war:** Ideal para despliegues en servidores de aplicaciones existentes como Tomcat, ofreciendo flexibilidad.
- Docker:** La imagen oficial jenkins/jenkins:lts permite una instalación rápida y aislada, accesible en el puerto 8080.
- Paquetes Nativos:** Métodos específicos para sistemas operativos, como apt para Debian/Ubuntu y yum para Red Hat/CentOS, facilitan la integración con el sistema.

Instalación con Docker

```
docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v  
jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
```

Primer Acceso y Desbloqueo

Una vez instalado, Jenkins es accesible en su navegador a través de <http://localhost:8080> (puerto por defecto). Para el primer acceso, se requiere una "initial admin password" que se encuentra en los logs de instalación o en un archivo específico, asegurando la configuración inicial del sistema.



ⓘ Al correr el contenedor docker se mostrará la "initial admin password" para continuar con la configuración.

Configuración Inicial



Creación de Usuario



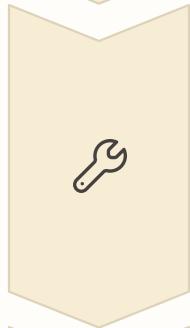
El primer paso tras desbloquear Jenkins es crear un usuario administrador. Este usuario será la base para gestionar la instancia y sus proyectos.



Instalación de Plugins



Jenkins es altamente personalizable mediante plugins. Se recomienda instalar los "plugins sugeridos" para obtener funcionalidades básicas como Git y Pipeline. Posteriormente, se pueden añadir otros según las necesidades del proyecto (Mail, GitHub, Metrics).



Configuración de Herramientas Globales



Es importante configurar las rutas de herramientas como JDK, Maven o Gradle, si se van a utilizar. Además, la gestión de credenciales (ej. SSH keys o tokens para repositorios de código) es vital para la seguridad y el acceso a recursos externos. (No será visto en este tutorial)



Gestión de Nodos



Para proyectos más grandes, se pueden añadir agentes remotos o nodos. Esto permite distribuir las cargas de trabajo de las construcciones, mejorando la eficiencia y el rendimiento de Jenkins al ejecutar tareas en paralelo.

- ⓘ Luego de la creación del usuario administrador, podrás acceder a la pantalla inicial de Jenkins que te mostrará todos los trabajos disponibles para construir.

Plugins Esenciales de Jenkins

Git Plugin

Fundamental para la integración con sistemas de control de versiones como GitHub, GitLab o Bitbucket, permitiendo a Jenkins clonar y monitorear repositorios para disparar builds.

Pipeline Plugin

Permite definir y gestionar pipelines complejos directamente como código (`Jenkinsfile`) dentro de tus repositorios, facilitando la automatización de flujos CI/CD completos y versionados.

Maven Integration Plugin

Diseñado para proyectos Java, este plugin simplifica la configuración y ejecución de builds Maven, integrando sus comandos y fases directamente en tus jobs.

Blue Ocean

Ofrece una interfaz de usuario moderna y visualmente atractiva para Jenkins, optimizada para la creación y monitoreo de pipelines, con una representación gráfica de cada etapa.

Email Extension Plugin

Vital para la comunicación, permite configurar notificaciones por correo electrónico personalizadas sobre el estado de los builds, asegurando que el equipo esté siempre informado.

JUnit Plugin

Este plugin es clave para la calidad del software, ya que publica los resultados de los tests unitarios ejecutados, mostrando reportes detallados y tendencias de fallos.

La vasta biblioteca de plugins es uno de los mayores fortalezas de Jenkins, permitiendo a los usuarios extender su funcionalidad para adaptarse a prácticamente cualquier necesidad de automatización. Elegir los plugins correctos puede optimizar significativamente el rendimiento y la eficiencia de tus flujos de trabajo.

Creando tu Primer Job (Conceptos Iniciales)

Jobs

Un **Job**, o proyecto, en Jenkins es una **tarea configurable** que Jenkins ejecuta. Imaginen una receta: el Job define qué pasos seguir. Desde el dashboard de Jenkins, selecciona "Nuevo Item". Elige "Proyecto de estilo libre" para un proyecto sencillo. Asígnale un nombre descriptivo a tu job.

Workspace

Es el directorio de trabajo donde el Job opera. Todos los archivos del proyecto que se necesiten o se obtengan podrás ubicarlos allí. En general se encuentra en /var/jenkins_home del servidor.

SCM (Source Code Management)

Dónde reside el código fuente (ej. Git). El Job se conecta a este repositorio

Build

Es la ejecución de un Job. Cada vez que corremos la "receta", obtenemos un "build". Por cada corrida jenkins retorna información acerca de su estado, tiempo de ejecución, etc.

Console Output

La salida de texto detallada de un "build". Es como el log de la ejecución.

Trigger

Evento que inicia un Job (ej. push a Git, horario programado).

Tipos de Jobs en Jenkins

Proyecto de Estilo Libre

Es el tipo de job más flexible y versátil. Permite configurar pasos manuales para construir, probar o desplegar código.

Pipeline

Define el flujo de integración y despliegue continuo (CI/CD) como código. Es versionable y altamente escalable para automatización compleja.

Proyecto Maven/Gradle

Optimizado para proyectos Java que utilizan herramientas como Maven o Gradle. Simplifica la configuración y gestión de dependencias automáticamente.

Carpeta (Folder)

Permite organizar jobs, vistas y otros elementos de Jenkins en una estructura jerárquica, mejorando la gestión de proyectos grandes.

Carpeta de Organización (Organization Folder)

Ideal para integrar Jenkins con sistemas de control de versiones como GitHub o Bitbucket, detectando automáticamente y creando jobs para repositorios o ramas que cumplen ciertos criterios.

Ejemplo Básico de Job con Jenkins

Sigue estos pasos para crear y ejecutar tu primer Job simple en Jenkins, generando una salida básica de "Hola Mundo".



The screenshot shows the Jenkins dashboard with the 'Nuevo Tarea' (New Item) option highlighted under the 'Nuevo Tarea' (New Item) section.

1. Crear Nuevo Item

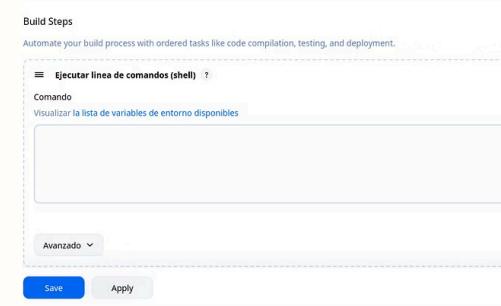
Desde el dashboard de Jenkins, selecciona "Nuevo Item" para empezar un nuevo proyecto.



The screenshot shows the 'Nuevo Tarea' configuration page. The item name is set to 'miproyecto'. The item type is selected as 'Crear un proyecto de estilo libre' (Create a free-style project). A brief description is provided: 'Classic, general-purpose job type that checks out from up to one post-build steps like archiving artifacts and sending email notifications'. Save and Apply buttons are at the bottom.

2. Configurar Proyecto Libre

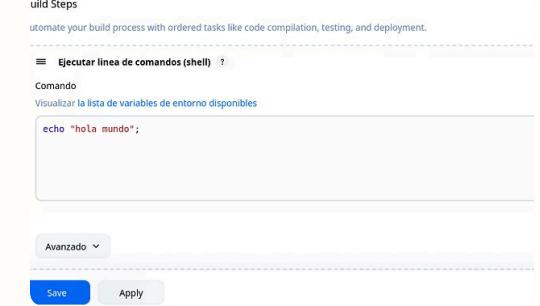
Elige "Proyecto de estilo libre", asignale un nombre descriptivo y haz clic en "OK" para continuar.



The screenshot shows the 'Build Steps' configuration page for the 'miproyecto' job. It contains a single step: 'Ejecutar línea de comandos (shell)' (Execute shell). The command is set to 'echo "Hola Mundo"'. Save and Apply buttons are at the bottom.

3. Añadir Paso de Construcción

En la sección "Build", selecciona la opción "Execute shell" (o "Execute Windows batch command" si es en Windows).



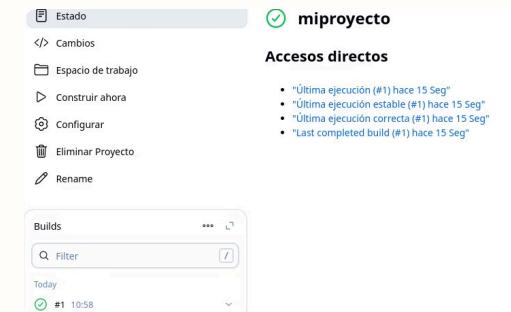
The screenshot shows the 'Build Steps' configuration page for the 'miproyecto' job. The command field contains 'echo "Hola Mundo"'. Save and Apply buttons are at the bottom.

4. Ejecutar Script Simple

En el área de texto, escribe el comando **'echo 'Hola Mundo'**. Este comando generará la salida deseada en la consola.



The screenshot shows the Jenkins dashboard with the 'miproyecto' job selected. The left sidebar shows options like Estado, Cambios, Espacio de trabajo, Construir ahora, Configurar, Eliminar Proyecto, and Rename. The right panel shows the job details: Estado (Estado), Accesos directos (Access directos), and Builds (Builds). The last build was completed successfully at 10:58 AM today.



The screenshot shows the Jenkins job configuration page for 'miproyecto'. It includes sections for Estado, Accesos directos (Access directos), and Builds. The 'Builds' section shows the last build was successful at 10:58 AM.



The screenshot shows the Jenkins job console output for the '#1' build of 'miproyecto'. The output shows the command 'echo "Hola Mundo"' was executed and the output 'Hola Mundo' was printed. The status is 'Finished: SUCCESS'.

5. Guardar y Ejecutar

Guarda tu Job y luego haz clic en "Build Now" (Construir Ahora) para iniciar la ejecución. Podrás ver "Hola Mundo" en la consola de salida.

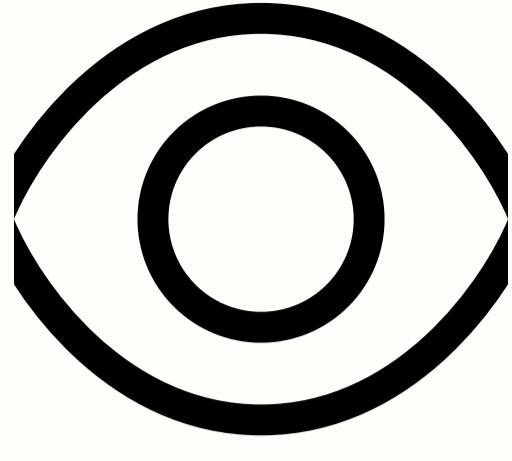
Resultado

Puedes apreciar el resultado de la construcción de tu Job (correcto, error, etc.), el número de instancia de ejecución y el horario en que se realizó.

Información de la construcción

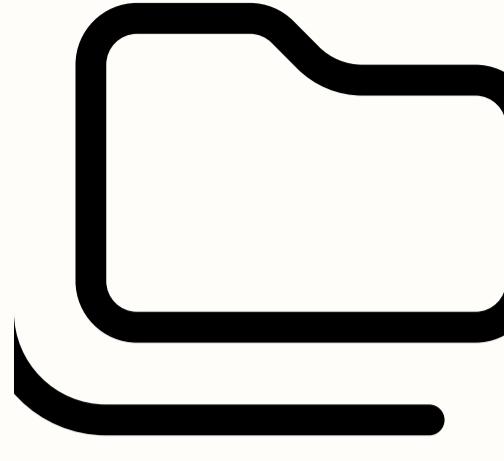
Se puede visualizar la salida por consola del resultado obtenido.

Conceptos Avanzados



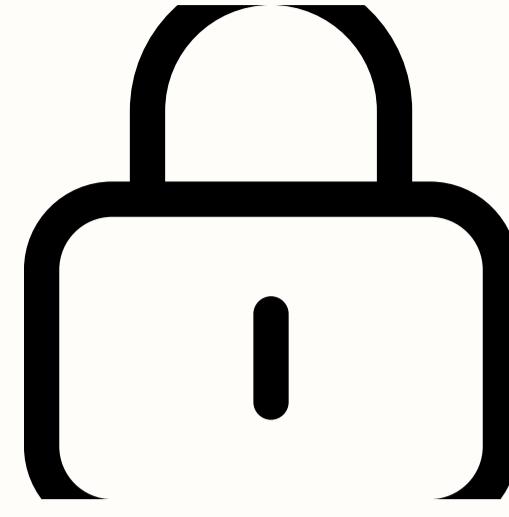
Vistas

Permite realizar una vista de todos los jobs filtrados bajo algún criterio específico. Permite organizar y filtrar los jobs de Jenkins para una mejor visualización y gestión.



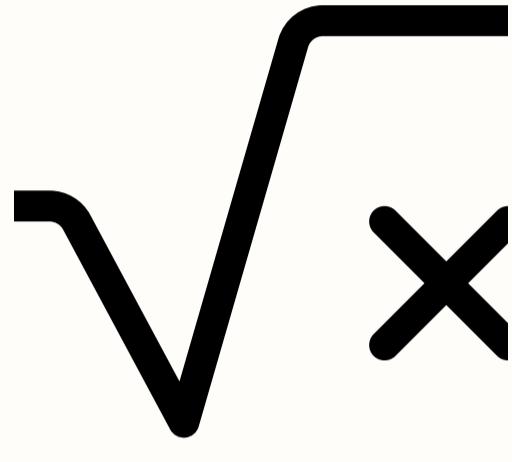
Folders

Los folders permiten organizar jobs jerárquicamente, creando una estructura de carpetas similar a un sistema de archivos.



Credenciales

Jenkins proporciona un sistema seguro para almacenar y gestionar credenciales como passwords, tokens, certificados y claves SSH.



Variables de Entorno

Existe un conjunto de Variables Built-in de Jenkins.

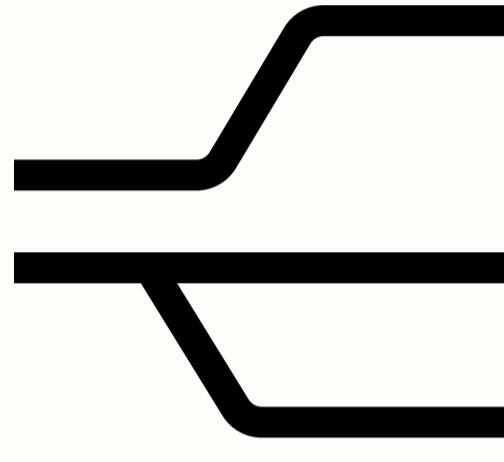
- BUILD_NUMBER
- JOB_NAME
- WORKSPACE
- BUILD_URL

Tienen un alcance global, por pipeline o step.

Podemos acceder al listado de variables de entorno incluso de los plugins que tenemos instalados (ver documentación del plugin). Para crearlas: Configuración>Sistema..

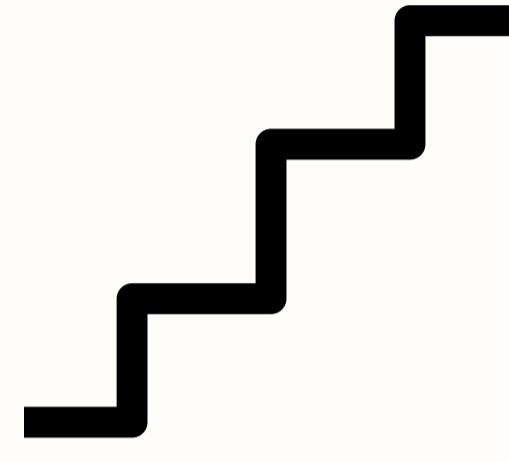
TODAS LAS ENV SON VISIBLES PÚBLICAMENTE (no asignar datos sensibles)

<http://.../env-vars.html>



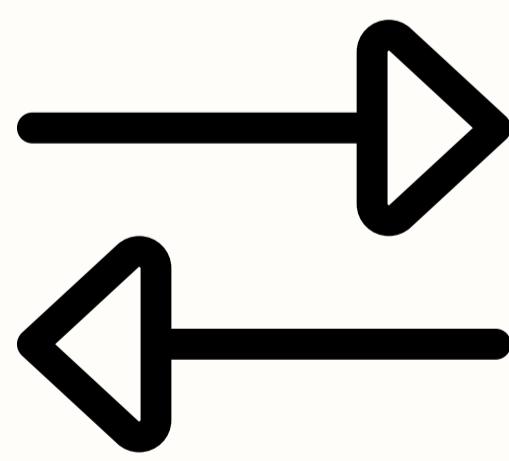
Pipeline

Los pipelines permiten definir todo el proceso de CI/CD como código, versionable y reproducible. Gestiona actividades de larga duración que pueden abarcar varios agentes de construcción. Apropiado para construir pipelines trabajos que involucran tareas complejas que no se pueden articular fácilmente con tareas de tipo freestyle.



Step

Los steps son las unidades básicas de ejecución en un pipeline. Cada step ejecuta una acción específica.

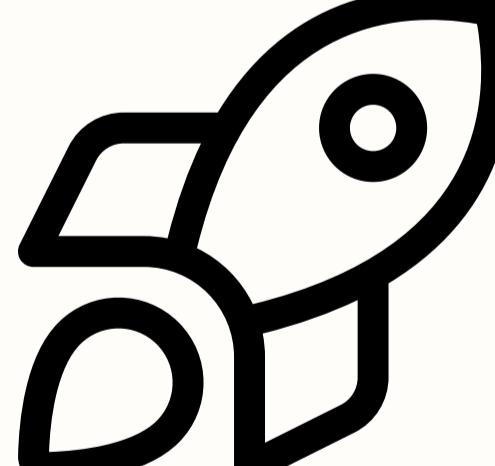


Parámetros

Los parámetros permiten que el usuario proporcione valores de entrada al ejecutar un pipeline, haciendo el pipeline configurable y reutilizable.

Tipos de Parámetros

- string
- choice
- booleanParam
- password
- etc.



Trigger Job

En el caso de necesitar segmentar nuestro trabajo podremos realizar llamadas a diferentes jobs dentro de un mismo pipeline para que sean ejecutados dentro del mismo trabajo.

Pipeline - Clonando un repositorio git

A continuación daremos un ejemplo de actualización y despliegue de código en repositorio git. Generaremos un job de tipo pipeline donde incluiremos.

- **parametros**
- **variables de entorno**
- **credenciales**
- **steps de actualización**

eline script

Script ?

```
1~ pipeline {  
2   agent any  
3~   parameters{ //SE DEFINEN PARÁMETROS  
4     string(name:"FINAL_MESSAGE",description:"defina el men  
5     string(name:"NUM_RELEASE",description:"defina la versi  
6     text(name:"COMMENT_RELEASE",description:"defina el com  
7   }  
8~   environment{//SE DEFINEN VARIABLES DE ENTORNO  
9     FILENAME=“nuevo.txt”  
10  }  
11~  stages {  
12~    stage('INIT') {  
13~      steps {  
14        echo "A CONTINUACIÓN SE DESPLEGARÁ LA ACTUALIZ  
15        //FORMA DE ESCRIBIR UN PARÁMETRO
```

Use Groovy Sandbox ?

Pipeline Syntax

anced

anzado ▾

Save

Apply

```

1 pipeline {
2     agent any
3
4     parameters {
5         string(
6             name: 'VERSION',
7             defaultValue: '1.0.0',
8             description: 'Versión a desplegar'
9         )
10        booleanParam(
11            name: 'RUN_TESTS',
12            defaultValue: true,
13            description: 'Ejecutar tests antes del deploy'
14        )
15    }
16
17    stages {
18        stage('Deploy') {
19            steps {
20                echo "Desplegando versión ${params.VERSION}"
21
22                script {
23                    if (params.RUN_TESTS) {
24                        echo "Ejecutando tests..."
25                    }
26                }
27            }
28        }
29    }
30 }

```

Parámetros

Los parámetros en Jenkins son variables configurables que permiten personalizar la ejecución de un job sin modificar su configuración base. Funcionan como argumentos de entrada que el usuario puede definir al momento de ejecutar el build, proporcionando flexibilidad y reutilización del código.

Tipos de Parámetros Principales

- **String Parameter:** Texto libre
- **Choice Parameter:** Lista desplegable con opciones predefinidas
- **Boolean Parameter:** Checkbox verdadero/falso
- **File Parameter:** Permite subir archivos

```

33 pipeline {
34     agent any
35
36     environment {
37         // Variables personalizadas
38         APP_NAME = 'mi-aplicacion'
39         DOCKER_REGISTRY = 'registry.company.com'
40     }
41
42     stages {
43         stage('Build Info') {
44             steps {
45                 echo "==== INFORMACIÓN DEL BUILD ==="
46                 echo "Job: ${env.JOB_NAME}" //variable de entorno global
47                 echo "Build #: ${env.BUILD_NUMBER}"
48                 echo "Workspace: ${env.WORKSPACE}"
49             }
50         }
51
52         stage('Docker Build') {
53             steps {
54                 script {
55                     def imageTag = "${env.APP_NAME}:${env.BUILD_NUMBER}"
56                     echo "Construyendo imagen: ${env.DOCKER_REGISTRY}/${imageTag}"
57
58                     // Simulación del comando docker
59                     sh "echo 'docker build -t ${env.DOCKER_REGISTRY}/${imageTag} .'"
60                 }
61             }
62         }
63     }
64 }

```

Variables de Entrno

Las variables de entorno en Jenkins son valores dinámicos que contienen información sobre el contexto de ejecución del job. Se crean automáticamente durante el build y proporcionan metadatos esenciales como información del workspace, build number, etc.

Acceso a Variables de Entorno

- En Pipeline Script:
\${env.VARIABLE_NAME} o
env.VARIABLE_NAME
- En Shell Commands:
\$VARIABLE_NAME
- Listado completo: Disponible en
\${JENKINS_URL}/env-vars.html

```

70 stage('Database Migration') {
71     steps {
72         // Uso de credenciales Username/Password
73         withCredentials([usernamePassword(
74             credentialsId: 'db-credentials',
75             usernameVariable: 'DB_USER',
76             passwordVariable: 'DB_PASS'
77         )]) {
78             sh '''
79                 echo "Conectando a base de datos como: $DB_USER"
80                 mysql -u $DB_USER -p$DB_PASS -e "SELECT VERSION();"
81             '''
82         }
83     }
84 }

```

Credenciales

Las credenciales en Jenkins permiten el almacenamiento y gestión de información sensible como contraseñas, tokens de API, claves SSH y certificados. Permiten que los pipelines accedan a recursos protegidos sin exponer datos confidenciales en el código fuente.

Las credenciales se configuran en "Manage Jenkins > Credentials" y se referencian en los pipelines mediante su ID único, nunca exponiendo el valor real.

- **Credenciales Globales:**
Credenciales accesibles por todos los jobs y usuarios del sistema Jenkins, con permisos de lectura para la mayoría de roles. Ej: Tokens de APIs externas (GitHub, Docker Registry), credenciales de bases de datos de aplicación, etc.
- **Credenciales de Sistema:**
Credenciales restringidas exclusivamente para uso interno de Jenkins y sus componentes del sistema. Ej: Claves SSH para nodos, Tokens de servicios internos de Jenkins.

Esta separación permite aplicar el **principio de menor privilegio**: las credenciales de sistema mantienen la infraestructura segura mientras que las globales facilitan el trabajo colaborativo sin comprometer la seguridad del sistema.

```
1 pipeline {
2     agent any
3     parameters {
4         string(name:"FINAL_MESSAGE",description:"defina el mensaje final al terminar la construcción")
5         string(name:"NUM_RELEASE",description:"defina la versión de despliegue que quiere informar")
6         text(name:"COMMENT_RELEASE",description:"defina el comentario de la versión despliegue")
7     }
8     environment {
9         FILENAME=$newfile
10    }
11    stages {
12        stage('INIT') {
13            steps {
14                echo "A CONTINUACIÓN SE DESPLEGARÁ LA ACTUALIZACIÓN DEL PROYECTO"
15                echo "FORMA DE ESCRIBIR UN PARAMETRO"
16                echo "VERSIÓN ${params.NUM_RELEASE}"
17                echo "Comentarios de la versión"
18                //OTRA FORMA DE ESCRIBIR UN PARAMETRO
19                echo "${COMMENT_RELEASE}"
20            }
21        }
22        stage('BUILD') {
23            steps {
24                echo "MUESTRA UNA VARIABLE DE ENTORNO DEL SISTEMA"
25                echo "pasa por la construcción nro: ${BUILD_NUMBER}"
26                //MUESTRA UNA VARIABLE DE ENTORNO DEFINIDA EN EL PROYECTO
27                echo "En el caso de requerir más memoria hacerlo en un archivo denominado ${FILENAME}"
28                withCredentials([usernamePassword(credentialsId: '...', usernameVariable: '----_USERNAME', passwordVariable: '----_PASSWORD')]){
29                    echo "usuario gitlab ${----_USERNAME} ${----_PASSWORD}"
30                    sh 'git clone https://${----_USERNAME}:${----_PASSWORD}@gitlab.com/----group/----project.git'
31                }
32                /*script{
33                    /sh "git clone https://gitlab.com/jadevlaplatabricks-group/jadevlaplatabricks-project.git"
34                }*/
35            }
36        }
37        stage('FINAL'){
38            steps {
39                echo "ingresa al stage final"
40            }
41            post {
42                success {
43                    echo "${FINAL_MESSAGE}"
44                }
45            }
46        }
47    }
48 }
```

Pasos para la generación del archivo jenkinsfile (pipeline)

A continuación presentaremos los pasos básicos para generar el trabajo y el código del archivo resultante.

Ejemplo Básico de Job con Jenkins

Sigue estos pasos para crear y ejecutar tu primer Job simple en Jenkins, generando una salida básica de "Hola Mundo".



The dashboard shows the Jenkins logo and navigation links: 'Nuevo Tarea' (selected), 'Historial de Construcción', and 'Ayuda'.

1. Crear Nuevo Item
Desde el dashboard de Jenkins, selecciona "Nuevo Item" para empezar un nuevo proyecto.



The 'Nuevo Tarea' configuration page shows the item name 'miproyecto' entered in the 'Enter an item name' field. The 'Select an item type' dropdown is set to 'Crear un proyecto de estilo libre' (Free-style project).

2. Configurar Proyecto Libre
Elige "Pipeline", asígnale un nombre descriptivo y haz clic en "OK" para continuar.



The 'Build Steps' configuration page shows a single step: 'Ejecutar línea de comandos (shell)'. The command field contains 'Comando' and 'Visualizar la lista de variables de entorno disponibles'. Buttons for 'Save' and 'Apply' are at the bottom.

3. Añadir el código del pipeline
En la sección "Pipeline - Definition", selecciona la opción Pipeline Script



The Jenkins dashboard shows the job 'miproyecto' under the 'proyectogit' workspace. It includes links for 'Status', 'Changes', 'Build with Parameters', and 'Configurar'.

4. Construir el trabajo.

Ejemplo de código pipeline

```
pipeline {
    agent any
    parameters {
        string(name:"FINAL_MESSAGE",description:"Mensaje final")
        string(name:"NUM_RELEASE",description:"Versión de despliegue")
        text(name:"COMMENT_RELEASE",description:"Comentarios de la versión")
    }
    environment {
        FILENAMEX="nuevo.txt"
    }
    stages {
        stage('INIT') {
            steps {
                echo "Desplegando actualización"
                echo "Versión ${params.NUM_RELEASE}"
                echo "${COMMENT_RELEASE}"
            }
        }
        stage('BUILD'){
            steps {
                echo "Construcción #${BUILD_NUMBER}"
                echo "Archivo ${FILENAMEX}"
                withCredentials([usernamePassword(credentialsId: '----', usernameVariable: '----_USERNAME', passwordVariable: '----_PASSWORD')]){
                    echo "Usuario: ${----_USERNAME} ${----_PASSWORD}"
                    sh "git clone https://${----_USERNAME}:${----_PASSWORD}@gitlab.com/----group/----project.git"
                }
            }
        }
        stage('FINAL'){
            steps {
                echo "Etapa final"
            }
            post {
                success {
                    echo "${FINAL_MESSAGE}"
                }
            }
        }
    }
}
```

Propuesta de Trabajo

Deberá entregar un archivo jenkinsfile de tipo pipeline en el que se incluya:

- Parámetros
- Variables de Entorno
- Clonación de repositorio git público
- Comandos necesarios para la el despliegue exitoso
- Generación de archivo de texto con detalles del despliegue.



Conclusión

1

Herramienta Fundamental

Jenkins es una pieza clave en la automatización de CI/CD, indispensable para equipos de desarrollo modernos.

2

Automatización y Eficiencia

Permite automatizar tareas repetitivas, minimizando errores y mejorando la consistencia de los procesos de desarrollo.

3

Calidad y Velocidad

Contribuye significativamente a la mejora de la calidad del software y a la aceleración en la entrega de nuevas funcionalidades.

4

Comunidad y Soporte

Cuenta con una comunidad global activa, extensa documentación y miles de plugins que garantizan su adaptabilidad y evolución constante.