


Creación de una pipeline de Jenkins: proyecto Maven


Crear un trabajo ◇ Haga clic en Nuevo elemento y proporcione el nombre del proyecto ◇ Seleccione el tipo de pipeline

Enter an item name


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**


Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Multibranch Pipeline

Describe el proyecto en la sección Descripción

Description

Build a Maven Pipeline

[Plain text] [Preview](#)

- ☐ Discard old builds [?](#)
 - ☐ Do not allow concurrent builds
 - ☐ Do not allow the pipeline to resume if the controller restarts
 - ☐ GitHub project
 - ☐ Pipeline speed/durability override [?](#)
 - ☐ Preserve stashes from completed builds [?](#)
 - ☐ This project is parameterized [?](#)
 - ☐ Throttle builds [?](#)
-

Build Triggers

- ☐ Build after other projects are built [?](#)
- ☐ Build periodically [?](#)

En el proceso, agregue la cuenta de GitHub en SCM y la ruta del script como Jenkinsfile que ya está presente en el repositorio de GitHub.

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/NanditaSahu03/maven-jenkins-pipeline

Credentials ?

- none -

+ Add

Advanced...

Add Repository

Branches to build ?

Archivo Jenkins

```

pipeline {
    agent any
    tools {
        maven "Maven 3.9.5"
    }

    stages {
        stage('Build Artifact') {
            steps {
                sh "rm -rf maven-simple"
                sh "git clone https://github.com/pabloariasora/maven-simple/"
                sh "mvn package -f maven-simple/pom.xml"
            }
        }
    }
}

```

En este archivo Jenkins, estamos creando una canalización declarativa en la que utilizamos cualquier agente. En la sección de herramientas, estamos definiendo herramientas para instalar automáticamente y colocar el archivo `PATH`. Aquí hemos agregado el mismo nombre "Maven 3.9.6" que hemos definido en Configuración global en Administrar Jenkins.

En la primera etapa, estamos construyendo Artifact a partir de Maven Build. En el primer paso, limpiamos el paquete y compilamos el proyecto maven y omitimos todos los casos de prueba usando - `DskipTest=true` y luego archivamos la compilación en `target/*.jar`. archivos en el espacio de trabajo de Jenkins.

Aquí podemos ver que la compilación se ejecutó correctamente y los archivos `.jar` se crearon en la carpeta de destino.



Build #7 (Sep 27, 2022, 6:56:37 PM)



Build Artifacts



[numeric-0.0.1.jar](#) 16.75 MB  [view](#)



Changes

1. Added jenkins pipeline ([details](#) / [githubweb](#))
2. Added jenkins pipeline ([details](#) / [githubweb](#))



Started by user [Nandita Sahu](#)



Revision: 53478d646aa75afbc2089da50c9bc8ceefbaa53c

Repository: <https://github.com/NanditaSahu03/maven-jenkins-pipeline>

- refs/remotes/origin/main



[Test Result](#) (no failures)

En la segunda etapa, estamos probando la compilación de Maven usando "mvn test" y verificando los casos de prueba de JUnit en el paso posterior donde generamos un informe .xml en el directorio target/surefire-reports/ en el espacio de trabajo de Jenkins.

```
stage('Test Maven - JUnit') {  
    steps {  
        sh "mvn test"  
    }  
    post{  
        always{  
            junit 'target/surefire-reports/*.xml'  
        }  
    }  
}
```

Aquí puede ver que se pasan todos los casos de prueba de Junit.