Jenkins

1.- CREAR UN JOB 2

2.- INTEGRACIÓN CON GIT 3

1. Instalar plugin Git en Jenkins 3
2. Crear un nuevo Job con Git 3
3. Configurar el repositorio Git 3
4. Configurar disparadores de construcción automática 3
5. Agregar pasos de construcción 4
6. Guardar y probar 4
7. Configurar webhook en GitHub (opcional pero recomendado) 4

3.- INTEGRACIÓN CON GIT Y MAVEN 5

1. Prerrequisitos 5
2. Configurar Maven en Jenkins 5
3. Crear nuevo Job en Jenkins 5
4. Configurar integración con Git 5
5. Configurar pasos de construcción para Maven 5
6. Configurar disparadores (opcional) 6
7. Guardar y ejecutar 6

4.- INTEGRACIÓN CON PRUEBAS 7

1. Configura Jenkins para usar Maven (como antes) 7
2. Ejecutar pruebas automáticamente 7
3. Ver reportes de pruebas 7
4. Integración con pipelines (opcional avanzado) 8

5.- CREAR PIPELINES 9

Paso 1: Crear un nuevo pipeline 9
Paso 2: Definir el pipeline 9
Paso 3: Ejemplo básico de Jenkinsfile declarativo 9
Paso 4: Guardar y ejecutar 10

1.- Crear un Job

1. Acceder a Jenkins: Abre tu navegador y entra a Jenkins en `http://localhost:8080` o la URL donde esté corriendo.
2. Iniciar sesión: Ingresa con tu usuario y contraseña.
3. Crear un nuevo job:
    - En la página principal, haz clic en "Nuevo Item" o "New Item".
    - Escribe un nombre para el job (por ejemplo: `MiPrimerJob`).
    - Selecciona el tipo de job:
        - Proyecto estilo libre (Freestyle project): El más común y sencillo para empezar.
        - Pipeline: Para definir trabajos con múltiples etapas en código.
        - Otros tipos: Multibranch Pipeline, Folder, etc.
    - Haz clic en Aceptar o OK.
4. Configurar el job:
    - En la pantalla de configuración, puedes definir:
        - Descripción del job.
        - Gestión del código fuente: Por ejemplo, integrar con un repositorio Git.
        - Disparadores de ejecución: Como ejecutar al hacer commit o con periodicidad.
        - Construir: Aquí defines los pasos, por ejemplo, ejecutar comandos, scripts, pruebas.
        - Post-construcción: Acciones tras la ejecución, como publicar resultados o notificaciones.
5. Guardar:
    - Haz clic en Guardar o Apply para conservar la configuración.
6. Ejecutar el job:
    - En la página del job, haz clic en "Construir ahora" o "Build Now".
    - Verás el progreso y puedes revisar los logs de ejecución.

2.- Integración con Git

1. Instalar plugin Git en Jenkins

- Accede a Jenkins como administrador.
- Ve a "Administrar Jenkins" > "Administrar plugins".
- En la pestaña Disponible, busca Git plugin.
- Selecciónalo y haz clic en Instalar sin reiniciar (o con reinicio si prefieres).

---

2. Crear un nuevo Job con Git

- Desde la página principal de Jenkins, haz clic en Nuevo Item.
- Escribe un nombre para el job.
- Selecciona Proyecto estilo libre y da clic en Aceptar.

---

3. Configurar el repositorio Git

- En la configuración del job, encuentra la sección Gestión del código fuente.
- Selecciona Git.
- En el campo Repository URL, escribe la URL del repositorio Git (SSH o HTTPS), por ejemplo:

  ```
  text
  https://github.com/usuario/repositorio.git
  ```

- Si el repositorio es privado, configura las credenciales:
  - Al lado de URL, haz clic en Agregar para añadir las credenciales (usuario/contraseña o clave SSH).
- También puedes especificar la rama que quieres construir (por ejemplo: `main` o `master`).

---

4. Configurar disparadores de construcción automática

- En la sección Disparadores de construcción, selecciona una o varias de las siguientes opciones:
  - Construir periódicamente: para construir en un horario (usa formato cron).

- Github hook trigger for GITScm polling: para ejecutar el job cuando hay cambios en GitHub (requiere configurar webhook en el repositorio GitHub).
- Poll SCM: Jenkins revisará periódicamente cambios en el repositorio.

---

5. Agregar pasos de construcción

- En la sección Construir, agrega, por ejemplo, un Ejecutar comando shell (Linux/Mac) o Ejecutar batch Windows para compilar o ejecutar scripts.
- Ejemplo simple:

```
bash
echo "Compilando proyecto..."
./gradlew build   # o cualquier comando de construcción
```

---

6. Guardar y probar

- Haz clic en Guardar.
- Ejecuta el job con Construir ahora y verifica que Jenkins haga la clonación del repositorio y ejecute los pasos correctamente.

---

7. Configurar webhook en GitHub (opcional pero recomendado)

- Ve a tu repositorio remoto en GitHub.
- En Configuración > Webhooks, agrega un nuevo webhook:
  - Payload URL: `http://[tu-jenkins]/github-webhook/`
  - Tipo de contenido: `application/json`
  - Escoge eventos: Pushes, Pull Requests, etc.
- Esto hará que GitHub notifique a Jenkins inmediatamente de cambios.

3.- Integración con GIT y Maven

1. Prerrequisitos

- Jenkins instalado y corriendo.
- Plugin Git instalado en Jenkins (como en pasos anteriores).
- Maven instalado en la máquina donde corre Jenkins o configurado en Jenkins.
- Tener un proyecto Java con archivo `pom.xml` en un repositorio Git.

---

2. Configurar Maven en Jenkins

- Ve a Administrar Jenkins > Configuración global de herramientas.
- Busca la sección Maven.
- Añade una instalación de Maven, puedes:
  - Marcar para instalar automáticamente la última versión.
  - O indicar una ruta de Maven ya instalada.
- Guarda la configuración.

---

3. Crear nuevo Job en Jenkins

- Desde la interfaz principal, haz clic en Nuevo Item.
- Nombre del job, selecciona Proyecto estilo libre y OK.

---

4. Configurar integración con Git

- En la sección Gestión del código fuente, selecciona Git.
- Escribe la URL del repositorio Git donde está tu proyecto Maven.
- Agrega credenciales si el repositorio es privado.
- Configura la rama (por ejemplo, `main`).

---

5. Configurar pasos de construcción para Maven

- En la sección Construir, haz clic en Agregar paso y selecciona Construir con Maven (o Invoke top-level Maven targets).
- En el campo Objetivos, escribe por ejemplo:

```
text
```

```
clean install
```

- Si tienes un `pom.xml` ubicado en una subcarpeta usa el campo POM para indicarlo (por ejemplo: `subcarpeta/pom.xml`).

---

6. Configurar disparadores (opcional)

- Elige disparadores como:
  - Poll SCM para revisar cambios en Git periódicamente.
  - Configura un webhook en tu GitHub para disparar automáticamente las builds.

---

7. Guardar y ejecutar

- Guarda el job.
- Haz clic en Construir ahora para probar la integración.
- Revisa la consola para verificar que Maven ejecuta la compilación y pruebas exitosamente.

4.- Integración con Pruebas

1. Configura Jenkins para usar Maven (como antes)

- Maven configurado en Jenkins (herramientas globales).
- Repositorio Git integrado en el job.

---

2. Ejecutar pruebas automáticamente

En la sección Construir del job en Jenkins:

- Añade un paso de construcción con Maven.
- En el campo Objetivos escribe:

```
text
clean test
```

o, si quieres hacer construcción y pruebas a la vez:

```
text
clean install
```

El ciclo de Maven ejecutará las pruebas unitarias definidas en el proyecto (normalmente con JUnit o TestNG).

---

3. Ver reportes de pruebas

- Maven genera reportes de pruebas automáticamente en el directorio `target/surefire-reports`.
- Para visualizar estos reportes en Jenkins, puedes:
  - Instalar el plugin JUnit en Jenkins (usualmente viene instalado).
  - En la configuración del job, agregar un post-paso de Publicar resultados de pruebas JUnit.
  - En el campo de reportes pon el path a los archivos XML de resultados, comúnmente:

```
text
**/target/surefire-reports/*.xml
```

Esto mostrará en Jenkins métricas visuales de pruebas ejecutadas, fallos, tiempos, y más.

---

4. Integración con pipelines (opcional avanzado)

Si usas un pipeline Jenkinsfile, el bloque para pruebas se vería así en Groovy:

```
groovy
stage('Test') {
   steps {
      sh 'mvn clean test'
      junit '**/target/surefire-reports/*.xml'
   }
}
```

5.- Crear Pipelines

Paso 1: Crear un nuevo pipeline

- En la interfaz web de Jenkins, haz clic en Nuevo Item.
- Ingresa un nombre para el pipeline.
- Selecciona el tipo Pipeline y haz clic en OK.

Paso 2: Definir el pipeline

En la sección de configuración:

- En Definición, elige:
  - Pipeline script: escribe el script directamente en Jenkins.
  - Pipeline script from SCM: carga el `Jenkinsfile` desde un repositorio Git.

Paso 3: Ejemplo básico de Jenkinsfile declarativo

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Compilando...'
                sh 'mvn clean install'
            }
        }
        stage('Test') {
            steps {
                echo 'Ejecutando pruebas...'
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Desplegando aplicación...'
                // comandos de despliegue aquí
            }
        }
    }
}
```

Paso 4: Guardar y ejecutar

- Guarda la configuración.
- Ejecuta el pipeline con Construir ahora.
- Puedes ver en tiempo real cada etapa y su salida.

6.- Integrar SonarQube en Jenkins con Maven

1. Tener SonarQube corriendo

- Puedes instalar SonarQube localmente o usar un servidor remoto.
- SonarQube expone una URL donde se accede a los reportes.

2. Instalar plugin SonarQube Scanner en Jenkins

- En Jenkins, ve a Administrar Jenkins > Administrar plugins.
- Busca e instala SonarQube Scanner.

3. Configurar SonarQube en Jenkins

- Ve a Administrar Jenkins > Configuración global > sección SonarQube servers.
- Añade el servidor con su URL y token de autenticación (desde SonarQube).

4. Configurar Maven para SonarQube

En tu `pom.xml` asegúrate que tienes configurado el plugin sonar-maven-plugin (opcional porque puede usarse scanner externo).

5. Modificar Jenkinsfile para análisis con SonarQube

Ejemplo sencillo:

```
pipeline {
  agent any
  tools {
    maven 'Maven'
  }
  stages {
    stage('Checkout') {
      steps {
        git 'https://github.com/usuario/repositorio.git'
      }
    }
    stage('Build') {
      steps {
        sh 'mvn clean install'
      }
    }
    stage('SonarQube analysis') {
      environment {
        scannerHome = tool 'SonarQubeScanner' // Nombre dado en Jenkins
      }
      steps {
        sh "${scannerHome}/bin/sonar-scanner"
      }
```

```
    }
  }
  post {
    always {
      junit '**/target/surefire-reports/*.xml'
    }
  }
}
```

```
pipeline {
    agent any
    tools {
        maven 'Maven'
    }
    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/usuario/repositorio.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn clean install'
            }
        }
        stage('SonarQube analysis') {
            environment {
                scannerHome = tool 'SonarQubeScanner' // Nombre dado en Jenkins
            }
            steps {
                sh "${scannerHome}/bin/sonar-scanner"
            }
        }
    }
    post {
        always {
            junit '**/target/surefire-reports/*.xml'
        }
    }
}
```

6. Ejecuta el pipeline y ve los reportes

- Cuando se complete, accede al dashboard de SonarQube para analizar la calidad del último análisis.