

Manual de despliegue con Jenkins

A continuación se detalla el proceso para montar un ambiente Jenkins utilizando docker, y proceder al despliegue de un pequeño back

Paso 1: Configuración del contenedor Jenkins

Abra docker desktop, luego, guarde los siguientes archivos en una carpeta y por medio del CMD, en la ruta de la carpeta ejecute `docker compose up -d --build`.

- `docker-compose.yml`:

```
version: '3.8'

services:
  jenkins:
    build: .
    container_name: jenkins
    ports:
      - "8080:8080"
      - "50000:50000"
    volumes:
      - jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      - DOCKER_HOST=unix:///var/run/docker.sock
    restart: unless-stopped
    extra_hosts:
      - "host.docker.internal:host-gateway"
    entrypoint: >
      bash -c "
        sudo chmod 666 /var/run/docker.sock || true
        /usr/local/bin/jenkins.sh
      "

volumes:
  jenkins_home:
```

- `docker-socket-permission.sh`:

```
#!/bin/bash
# docker-socket-permission.sh
# Este script ajusta los permisos del socket de Docker al iniciar

# Asegurar que el socket es accesible
if [ -e /var/run/docker.sock ]; then
  echo "Ajustando permisos para docker.sock"
  sudo chmod 666 /var/run/docker.sock
```

```
else
    echo "No se encontró el socket de Docker en /var/run/docker.sock"
fi

# Mostrar información útil para diagnóstico
echo "Usuario actual: $(whoami)"
echo "Grupos del usuario jenkins: $(groups jenkins)"
echo "Permisos del socket Docker: $(ls -la /var/run/docker.sock)"

# Verificar la conectividad a Docker
echo "Probando conexión a Docker:"
docker info || echo "Error al conectar con Docker"
```

- Dockerfile:

```
# Dockerfile modificado
FROM jenkins/jenkins:lts

USER root

# Instalación de dependencias necesarias
RUN apt-get update && \
    apt-get install -y apt-transport-https ca-certificates curl gnupg lsb-release
sudo

# Instalación de Docker CLI
RUN curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg && \
    echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" |
tee /etc/apt/sources.list.d/docker.list > /dev/null && \
    apt-get update && \
    apt-get install -y docker-ce-cli

# Crear grupo docker con GID 999 (evitamos conflictos con el GID 1000 existente)
RUN groupadd -g 999 docker

# Añadir usuario jenkins al grupo docker y sudo
RUN usermod -aG docker jenkins && \
    usermod -aG sudo jenkins && \
    echo "jenkins ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

# Instalar herramientas de diagnóstico
RUN apt-get install -y procps net-tools

# Script para ajustar los permisos del socket al iniciar
COPY docker-socket-permission.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/docker-socket-permission.sh

USER jenkins

# Instalar plugins de Jenkins
```

```
COPY plugins.txt /usr/share/jenkins/ref/plugins.txt
RUN jenkins-plugin-cli --plugin-file /usr/share/jenkins/ref/plugins.txt
```

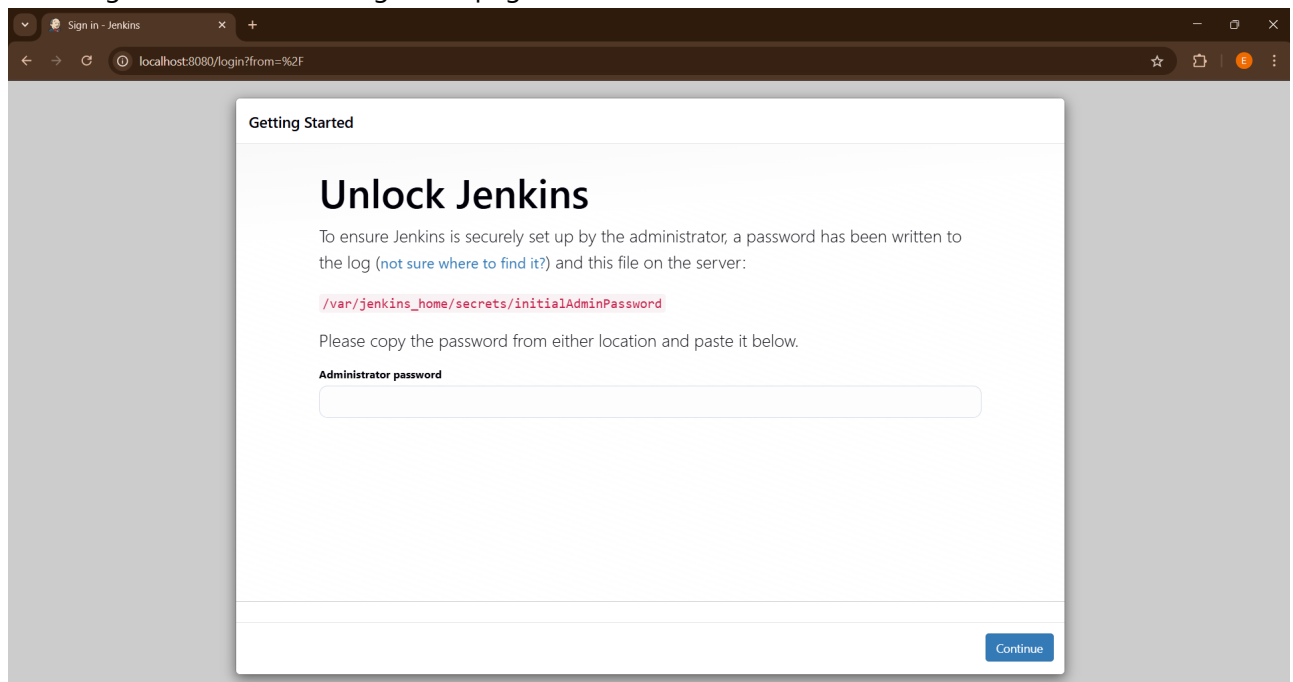
- `plugins.txt`:

```
workflow-aggregator
docker-workflow
git
blueocean
pipeline-utility-steps
credentials-binding
```

Paso 2: Iniciar sesión en la interfaz de Jenkins

2.1 A continuación diríjase a su navegador e ingrese la url `http://localhost:8080`.

- Su navegador lo llevará a la siguiente página.



2.2 Ahora, en la misma terminal ejecute el comando `docker compose logs`, esto generará en las últimas líneas la contraseña que necesita para poder ingresar

- En este caso la contraseña es `b4bbf76e9903484a8a8253c6162949cb`. Copie y pegue en el navegador y presione `continue`.

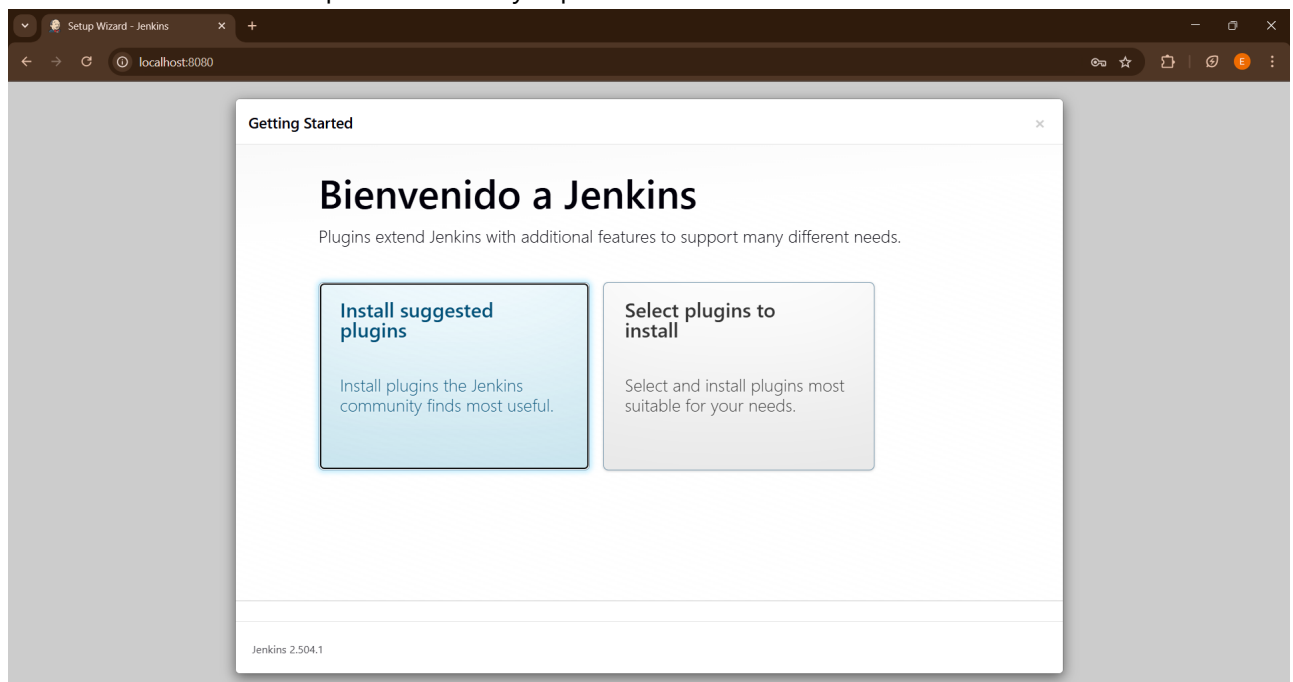
```
C:\Windows\System32\cmd.e |  +  v
```

```
jenkins | 2025-05-04 20:55:50.690+0000 [id=57] INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
jenkins | 2025-05-04 20:55:50.720+0000 [id=73] INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
jenkins | 2025-05-04 20:55:51.182+0000 [id=40] INFO    jenkins.install.SetupWizard#init:
jenkins | *****
jenkins | *****
jenkins | *****
jenkins | *****
jenkins | Jenkins initial setup is required. An admin user has been created and a password generated.
jenkins | Please use the following password to proceed to installation:
jenkins | b4bbf76e9903484a8a8253c6162949cb
jenkins | This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
jenkins | *****
jenkins | *****
jenkins | *****
jenkins | *****
jenkins | 2025-05-04 20:55:56.064+0000 [id=40] INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
jenkins | 2025-05-04 20:55:56.101+0000 [id=25] INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
jenkins | 2025-05-04 20:55:56.851+0000 [id=73] INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
jenkins | 2025-05-04 20:55:56.852+0000 [id=73] INFO    hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
```

```
D:\ELKIN\COSAS\jenkins-docker>
```

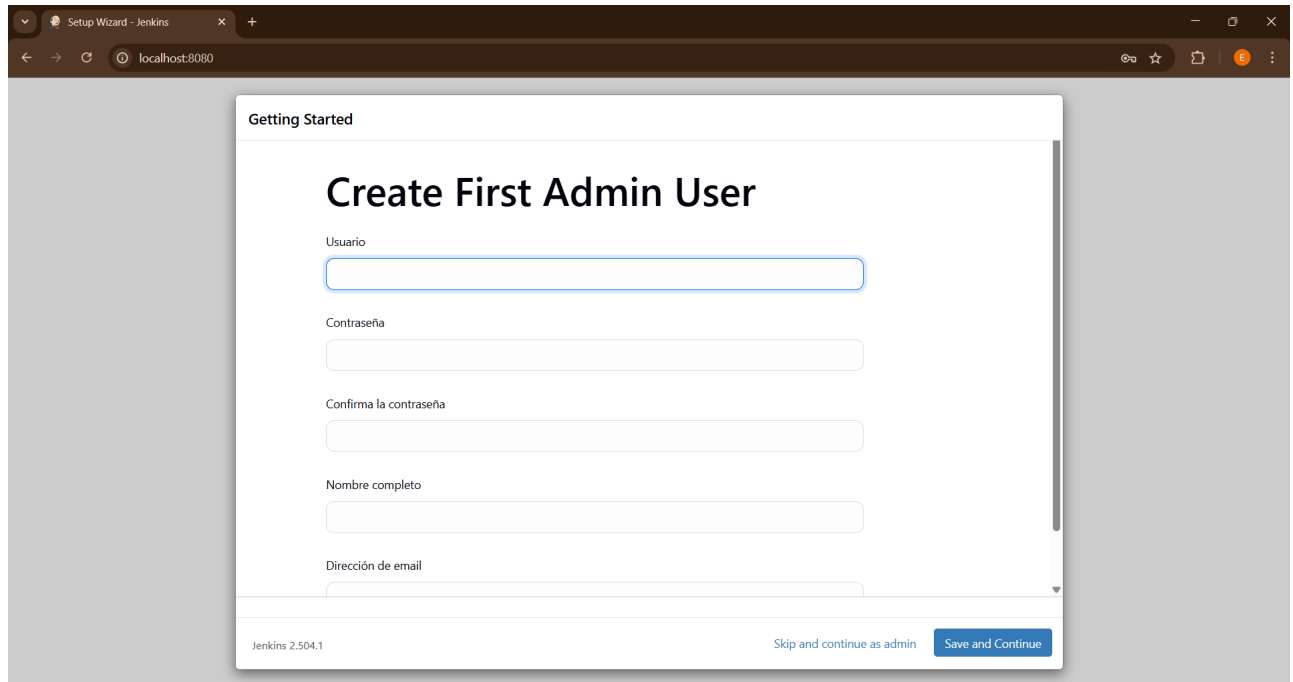
2.3 Como siguiente paso Jenkins necesita instalar diferentes plugins para su configuración inicial.

- Presione el recuadro azul para continuar y espere.



2.4 Una vez se haya instalado todo lo necesario, será redirigido al login para que ingrese las credenciales con las que se registrará.

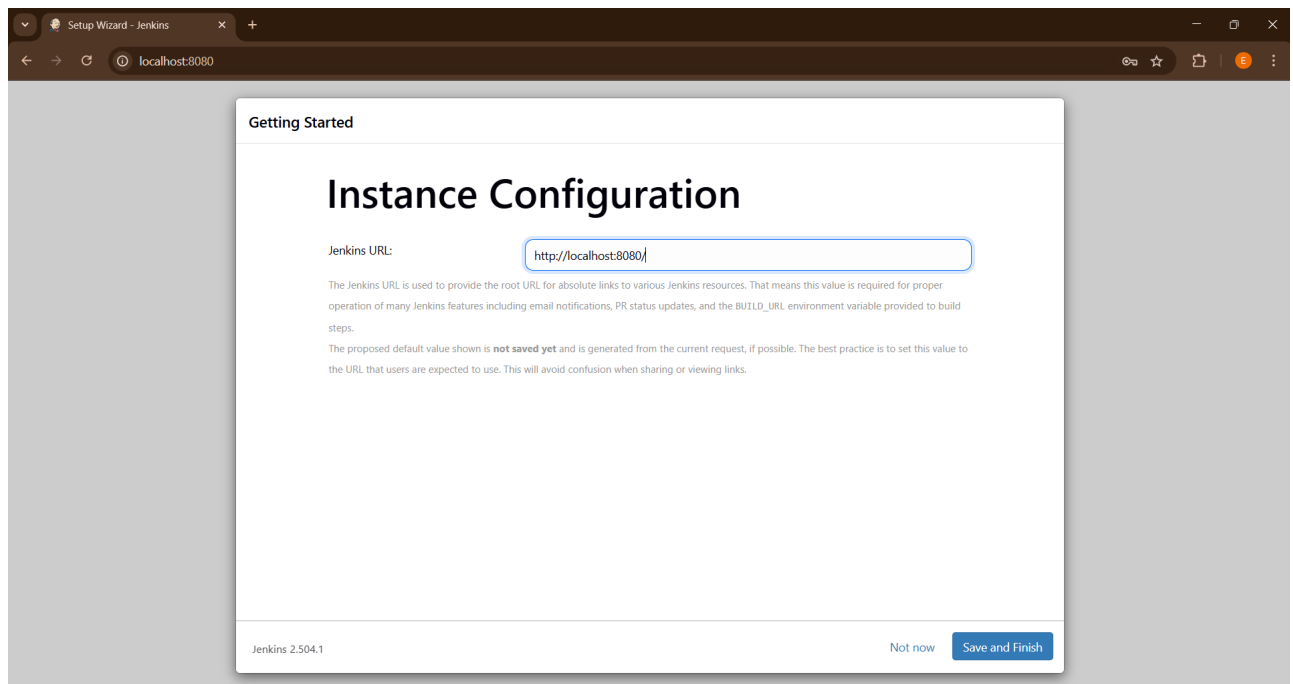
- Tenga en cuenta su nombre de usuario y contraseña que luego serán necesarios para iniciar la sesión. Ingrese los datos y presione **Save and Continue**.



The screenshot shows the 'Getting Started' section of the Jenkins Setup Wizard. The main heading is 'Create First Admin User'. Below this, there are five input fields: 'Usuario', 'Contraseña', 'Confirma la contraseña', 'Nombre completo', and 'Dirección de email'. At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version 'Jenkins 2.504.1' is displayed at the bottom left.

2.5 Confirme la ruta.

- Presione **Save and Finish**



The screenshot shows the 'Getting Started' section of the Jenkins Setup Wizard. The main heading is 'Instance Configuration'. Below this, there is a 'Jenkins URL:' label and an input field containing 'http://localhost:8080/'. Below the input field, there is a paragraph of text explaining the Jenkins URL and its importance. At the bottom right, there are two buttons: 'Not now' and 'Save and Finish'. The Jenkins version 'Jenkins 2.504.1' is displayed at the bottom left.

Paso 3: Crear y montar el repositorio git hub con el proyecto backend.

3.1 Vamos primero a descargar el proyecto desde Spring Initializr, con la siguiente configuración.

The screenshot shows the Spring Initializr web application interface. The browser tabs include 'Panel de control - Jenkins' and 'Spring Initializr'. The address bar shows 'start.spring.io'. The interface is divided into several sections:

- Project:**
 - Language: ☒ Java, ☐ Kotlin, ☐ Groovy
 - Framework: ☐ Gradle - Groovy, ☐ Gradle - Kotlin, ☒ Maven
- Spring Boot:**
 - Version: ☐ 3.5.0 (SNAPSHOT), ☐ 3.5.0 (RC1), ☐ 3.4.6 (SNAPSHOT), ☒ 3.4.5
 - Other versions: ☐ 3.3.12 (SNAPSHOT), ☐ 3.3.11
- Project Metadata:**
 - Group:
 - Artifact:
 - Name:
 - Description:
 - Package name:
 - Packaging: ☒ Jar, ☐ War
 - Java: ☐ 24, ☐ 21, ☒ 17
- Dependencies:**
 - Spring Boot DevTools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Configuration Processor** (DEVELOPER TOOLS): Generate metadata for developers to offer contextual help and "code completion" when working with custom configuration keys (ex.application.properties/.yaml files).
 - PostgreSQL Driver** (SQL): A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

At the bottom, there are buttons for **GENERATE** (CTRL + G), **EXPLORE** (CTRL + SPACE), and a menu icon (three dots).

3.2 Creamos un repositorio en git hub con el nombre **mi-app**, lo clonamos, pegamos el proyecto spring descargado, y antes de hacer el push hacia el repo remoto, creamos dentro del proyecto, en la carpeta raíz los siguientes archivos:

docker-compose.yml:

```
# docker-compose.yml corregido
version: '3.8'
services:
  app:
    image: miapp
    # 0 alternativamente, si necesitas construir la imagen desde aquí:
    # build: ./demo
    ports:
      - "8090:8080" # Cambiado a 8090 para evitar conflicto con Jenkins
    depends_on:
      - db
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/mydb
      SPRING_DATASOURCE_USERNAME: user
      SPRING_DATASOURCE_PASSWORD: password

  db:
    image: postgres:14
    environment:
      POSTGRES_DB: mydb
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
    ports:
      - "5432:5432"
    volumes:
```

```
- postgres_data:/var/lib/postgresql/data
```

```
volumes:
  postgres_data:
```

Dockerfile:

```
# Dockerfile corregido
# Usa una versión actualizada y disponible de Maven con JDK 17
FROM maven:3.9-eclipse-temurin-17 AS build
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests

# Runtime stage con una imagen JDK disponible
FROM eclipse-temurin:17-jdk
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Jenkinsfile:

```
//Jenkinsfile corregido
pipeline {
    agent any
    stages {
        stage('Clonar') {
            steps {
                git branch: 'main', url: 'https://github.com/ElkinContrerasR/mi-
app.git'
            }
        }
        stage('Compilar') {
            steps {
                dir('demo') {
                    // Asegúrate de que mvnw tenga permisos de ejecución
                    sh 'chmod +x mvnw'
                    sh './mvnw clean package -DskipTests'
                }
            }
        }
        stage('Docker Build') {
            steps {
                // Verificar que Docker está funcionando
                sh 'docker --version'

                // Mostrar imágenes disponibles (para diagnóstico)
                sh 'docker images'
```

```

// Prueba a usar una imagen de hello-world para verificar conexión
a Docker Hub
sh 'docker pull hello-world'

// Construir la imagen con el Dockerfile en el directorio correcto
dir('demo') {
    sh 'docker build -t miapp .'
}
}
stage('Desplegar') {
    steps {
        dir('demo') {
            // Verifica si existe el archivo docker-compose.yml
            sh 'ls -la'

            // Si docker-compose.yml está en este directorio:
            sh 'docker compose down || true'
            sh 'docker compose up -d --build'

            // Alternativa si el archivo está en otro lugar
            // sh 'docker compose -f ../docker-compose.yml down || true'
            // sh 'docker compose -f ../docker-compose.yml up -d --build'
        }
    }
}
}
post {
    always {
        // Limpieza y registro del estado
        echo 'Pipeline completado'
        sh 'docker ps'
    }
    failure {
        echo 'El pipeline ha fallado'
    }
}
}

```

y en la carpeta donde se encuentra el `MiApplication.java` creamos un controlador con un endpoint.

`HelloController.java`:

```

package com.ejemplo.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {
    @GetMapping("/")
    public String hello() {

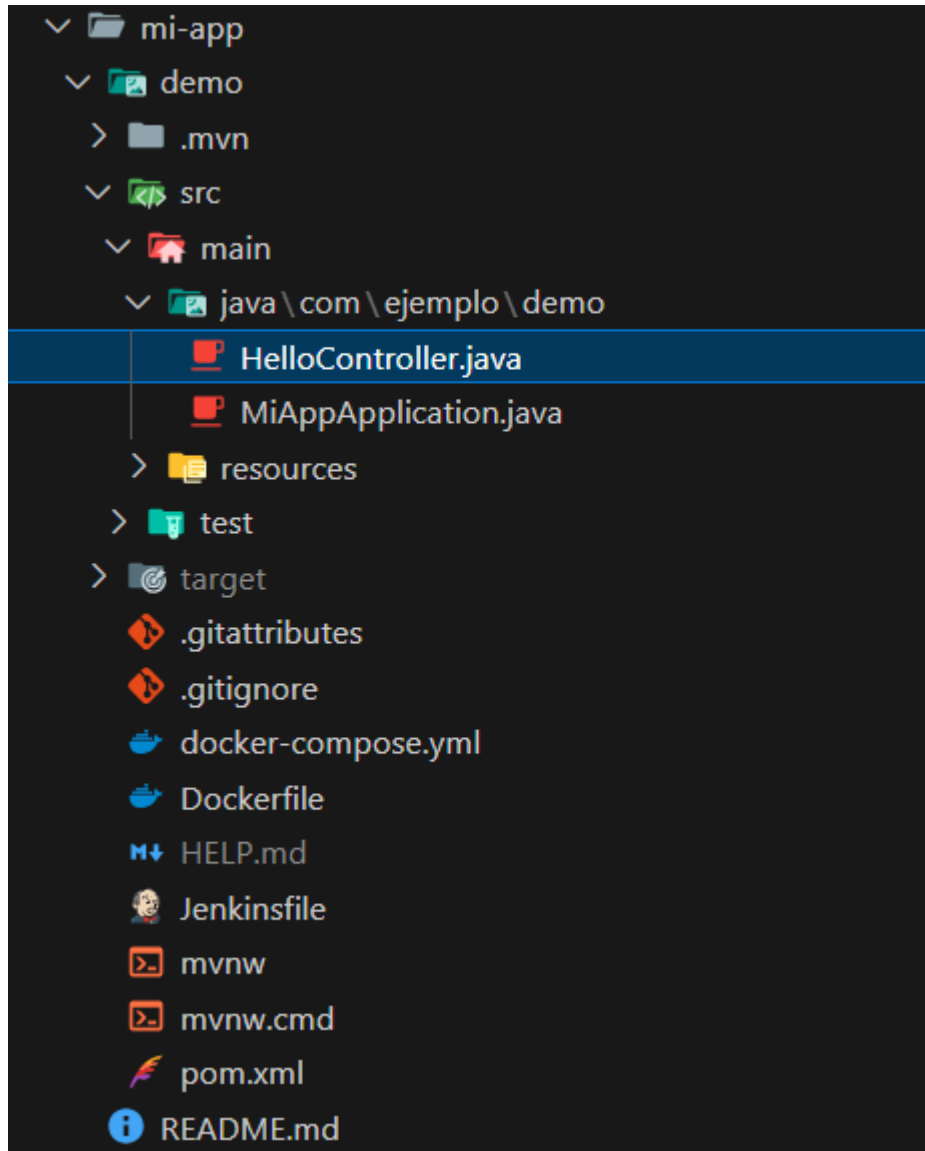
```



```
        return " Hola desde Spring Boot!";  
    }  
}
```

De esta manera deben quedar los archivos creados en el proyecto spring boot.

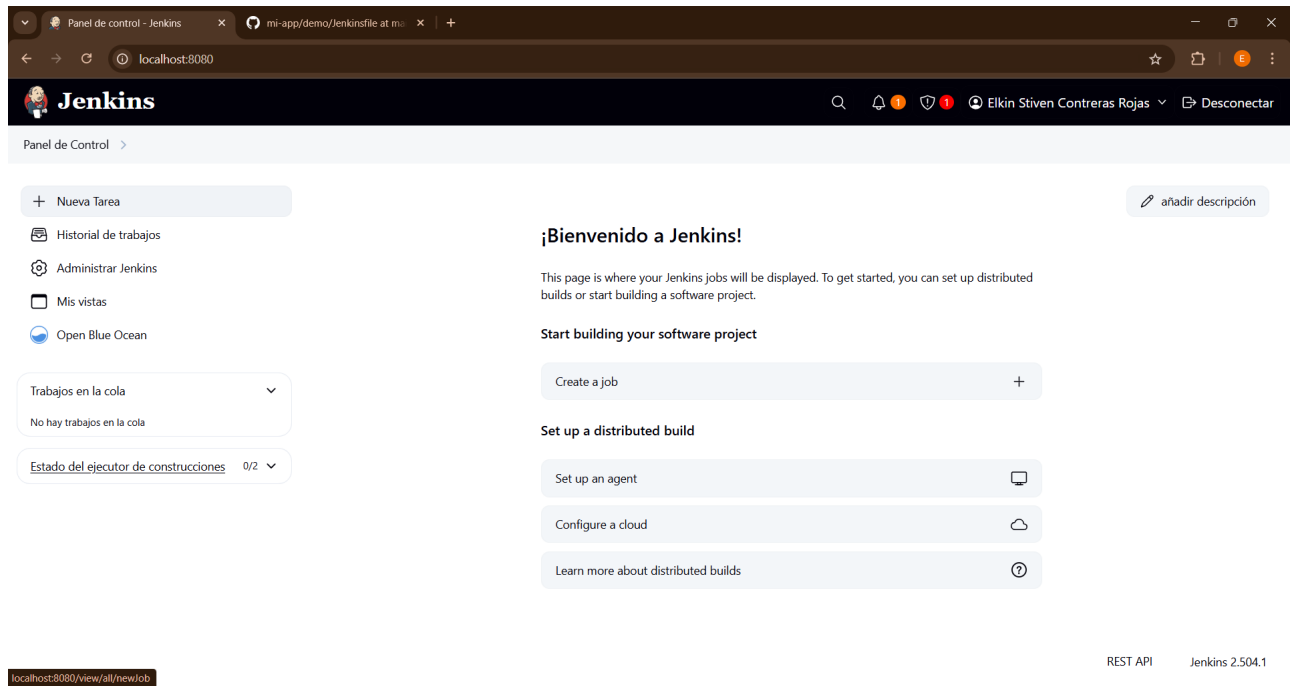
- Ya podemos hacer el push para que se suban los cambios.



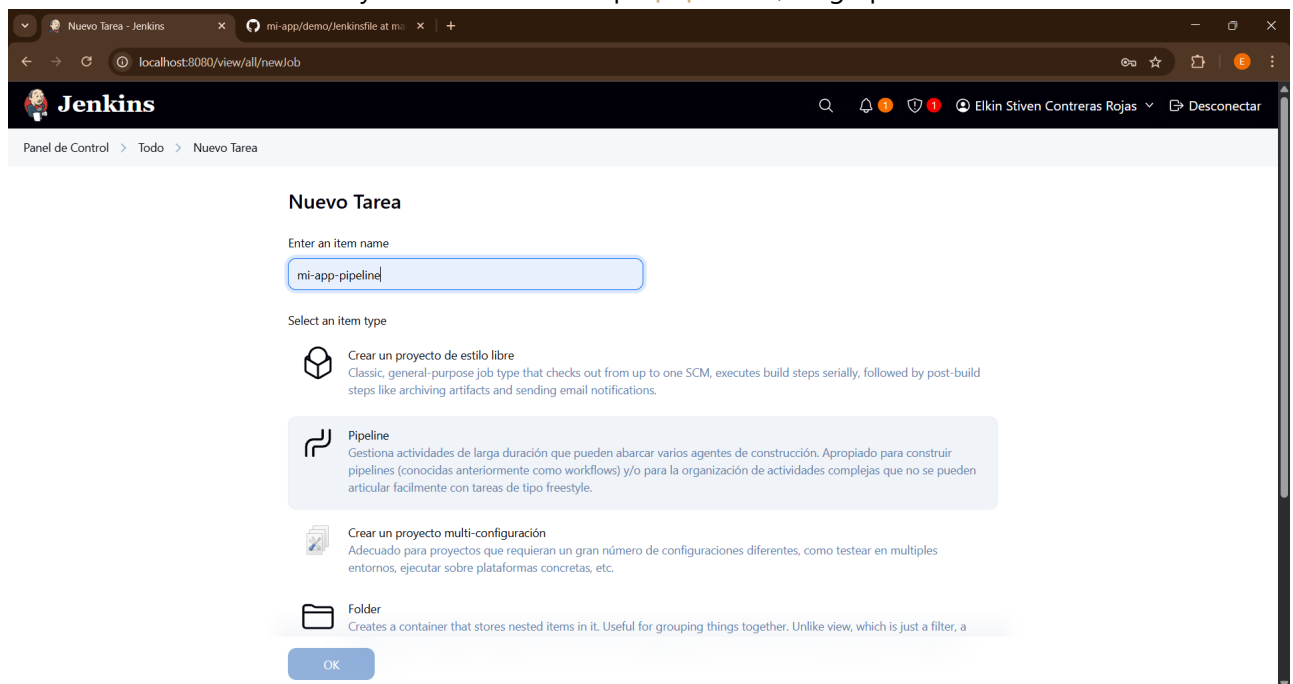
Paso 4: Crear el pipeline para la conexión y despliegue del proyecto en el repositorio.

4.1 Volvemos a la interfaz de Jenkins y creamos un pipeline

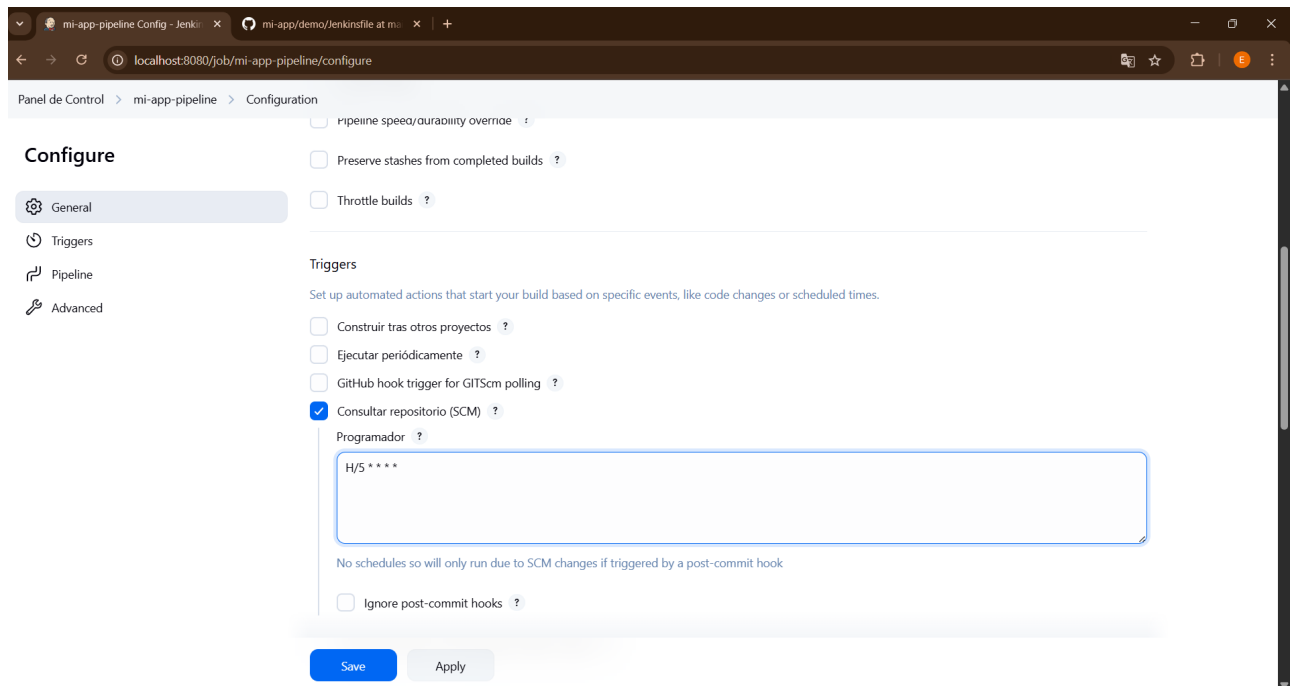
- Presionamos donde dice **Nueva Tarea**



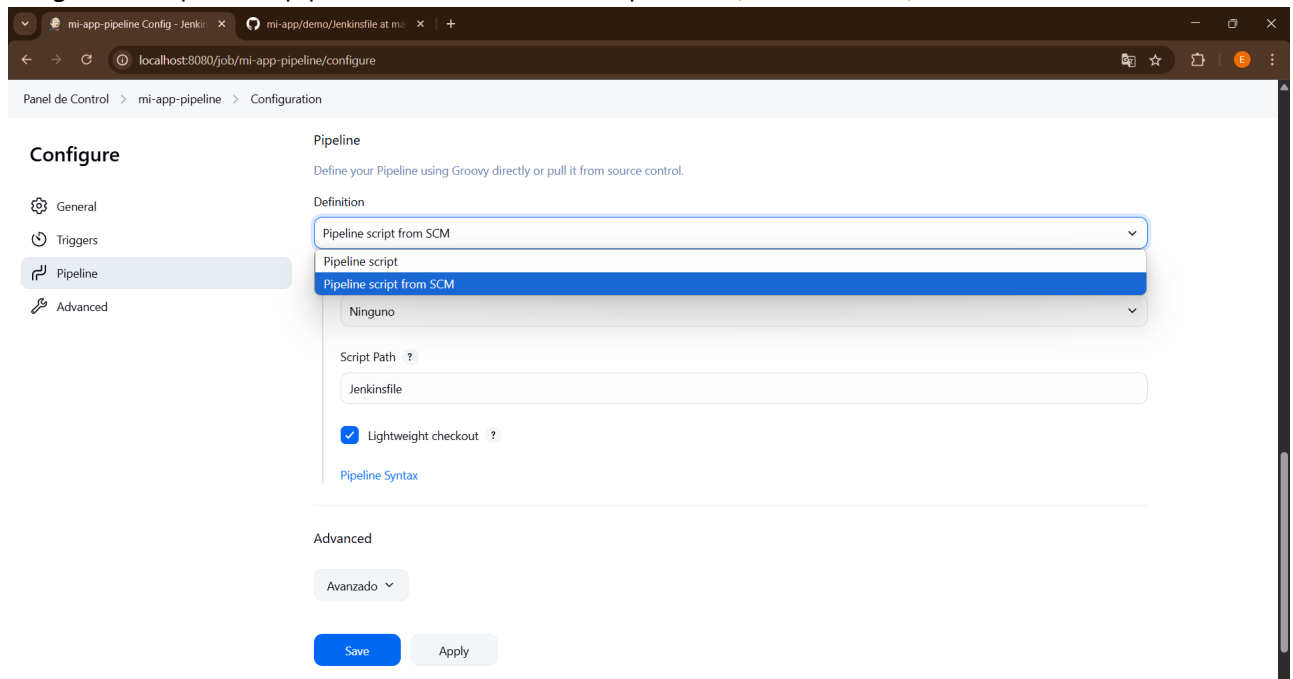
- Le damos nombre a la tarea y seleccionamos el tipo **pipeline**, luego presionamos OK.



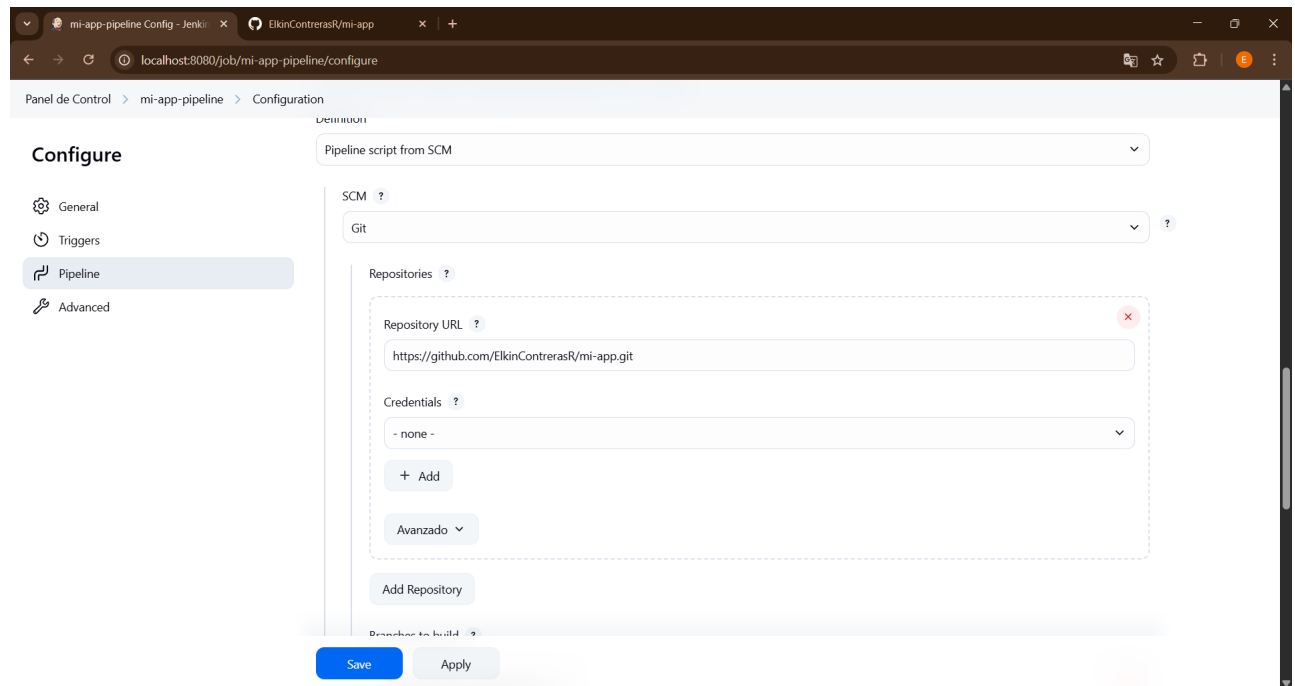
- Ingresamos de esta manera a la configuración del pipeline y lo primero es configurar el SCM en el apartado de **Triggers**, con el intervalo **H/5 * * * ***.



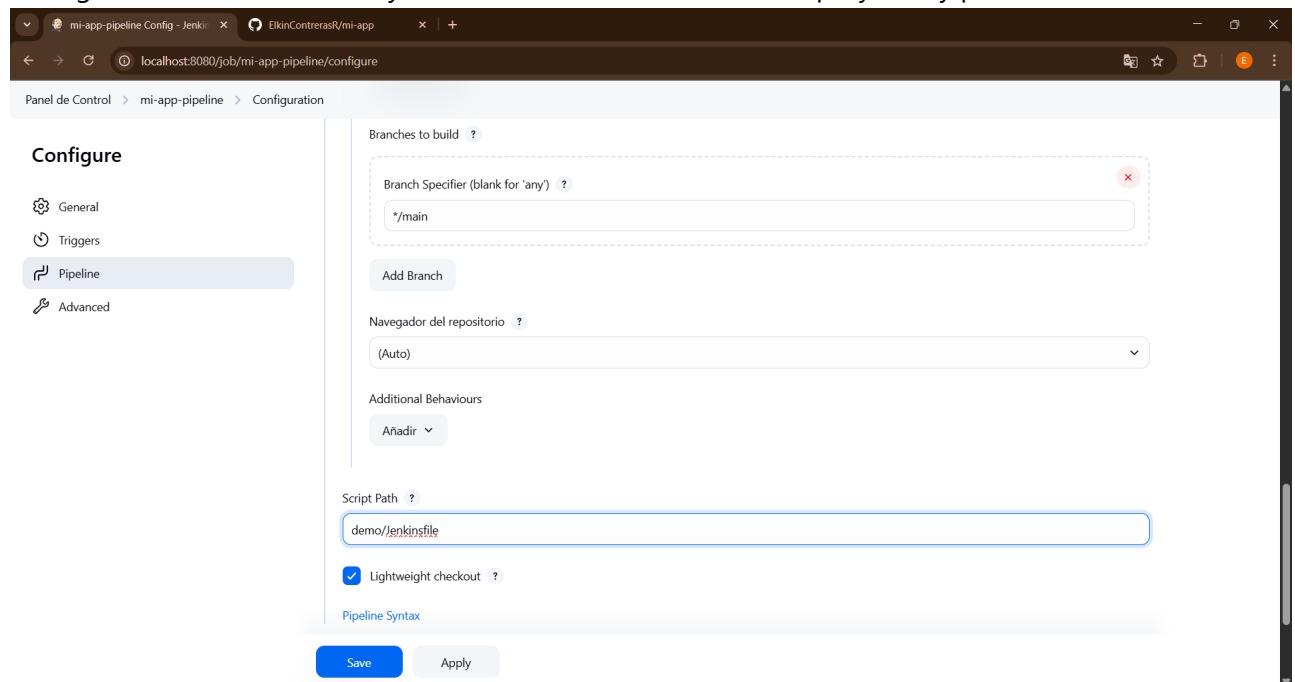
- Luego, en el apartado pipeline, seleccionamos la opción **Pipeline script from SCM**



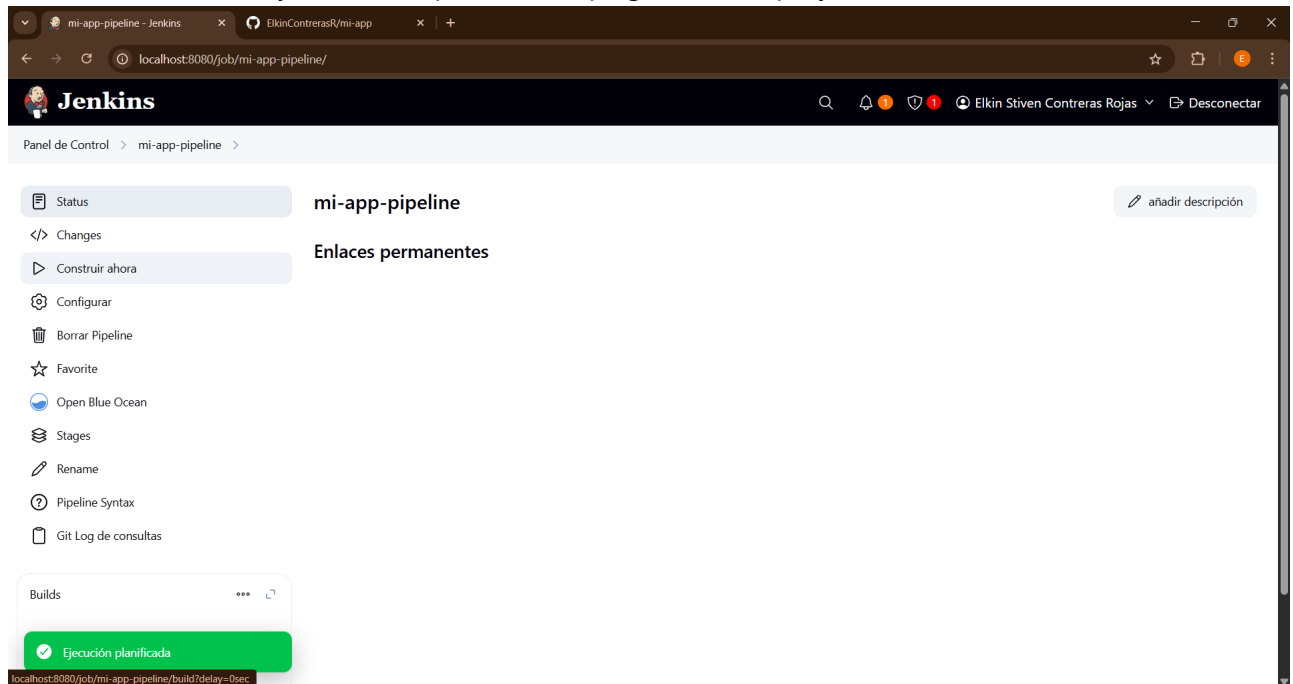
- Ponemos el SCM tipo GIT, y en **Repository URL** ponemos la URL de nuestro git creado, <https://github.com/@Usuario/mi-app.git> para conectar Jenkins con el repositorio



- Configuramos a la rama **main**, y la ruta del archivo Jenkinsfile del proyecto, y presionamos **Save**.



- Presionamos construir y Jenkins empezará a desplegar nuestro proyecto



Ya podrás probar tu proyecto llendo a la web con la ruta <http://localhost:8090/> para hacer la petición al servidor

