Let's break down the process of setting up a Jenkins freestyle job to build a Maven project from a GitHub repository on a Windows machine, providing detailed steps and practical examples.

Prerequisites (Ensure these are installed and configured):

* Windows: A Windows machine with a user account.
* Git: Download and install Git for Windows. After installation, open Git Bash and configure your username and email:

Bash

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

* JDK 11: Download and install JDK 11. Set the JAVA\_HOME environment variable:
  + Search for "Environment Variables" in Windows.
  + Click "Edit the system environment variables".
  + Click "Environment Variables...".
  + Under "System variables", click "New..." and add:
    - Variable name: JAVA\_HOME
    - Variable value: C:\Program Files\Java\jdk-11.0.x (adjust to your JDK installation path).
  + Edit the Path variable and add %JAVA\_HOME%\bin.
* Maven: Download and extract Apache Maven. Set the M2\_HOME and add to PATH:
  + Extract the Maven archive (e.g., apache-maven-3.9.x-bin.zip) to a directory (e.g., C:\apache-maven-3.9.x).
  + Set M2\_HOME:
    - Variable name: M2\_HOME
    - Variable value: C:\apache-maven-3.9.x (adjust to your Maven directory).
  + Edit the Path variable and add %M2\_HOME%\bin.
  + Open a new command prompt and run mvn -v to verify Maven is installed.
* Jenkins: Download and install Jenkins for Windows. After installation, Jenkins runs as a Windows service. Access the Jenkins web UI at http://localhost:8080.
* GitHub Account: A GitHub account.

Steps:

1. Create a Maven Project:

Open a command prompt or PowerShell:

Bash

mvn archetype:generate -DgroupId=com.example -DartifactId=my-jenkins-project -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

cd my-jenkins-project

This generates a basic Maven project in a folder named my-jenkins-project.

1. Create a GitHub Repository:
   * Go to GitHub and create a new repository (e.g., "my-jenkins-project"). *Do not* initialize it with a README, license, or .gitignore yet.
2. Push the Project to GitHub:

Bash

cd my-jenkins-project

git init

git add .

git commit -m "Initial commit"

git remote add origin <your\_github\_repo\_url> # Replace with your GitHub repo URL (e.g., https://github.com/yourusername/my-jenkins-project.git)

git branch -M main # Renames the local branch to main

git push -u origin main

1. Configure Jenkins:
   * Install Plugins:
     + Go to "Manage Jenkins" -> "Manage Plugins".
     + On the "Available" tab, search for and install the "Git plugin" and "Maven Integration plugin". Restart Jenkins if prompted.
   * Configure JDK and Maven:
     + Go to "Manage Jenkins" -> "Global Tool Configuration".
     + JDK: Click "Add JDK". Uncheck "Install automatically". Set "Name" (e.g., "JDK 11") and "JAVA\_HOME" to your JDK path (e.g., C:\Program Files\Java\jdk-11.0.x).
     + Maven: Click "Add Maven". Uncheck "Install automatically". Set "Name" (e.g., "Maven 3.9.x") and "MAVEN\_HOME" to your Maven path (e.g., C:\apache-maven-3.9.x).
2. Create a Jenkins Freestyle Job:
   * On the Jenkins dashboard, click "New Item".
   * Enter a name (e.g., "Maven-GitHub-Build").
   * Select "Freestyle project" and click "OK".
3. Configure the Freestyle Job:
   * Source Code Management:
     + Select "Git".
     + Repository URL: Paste your GitHub repository URL (e.g., https://github.com/yourusername/my-jenkins-project.git).
     + Credentials: If your repo is private:
       - Click "Add" -> "Jenkins".
       - Choose "Username with password". Enter your GitHub username and password or a personal access token (recommended for security).
       - Click "Add".
     + Branch Specifier: Leave as \*/main to build the main branch.
   * Build:
     + Click "Add build step" -> "Invoke top-level Maven targets".
     + Maven Version: Select the Maven installation you configured (e.g., "Maven 3.9.x").
     + Goals: Enter clean install. This is the standard Maven build lifecycle:
       - clean: Removes the target directory (where build artifacts are stored).
       - install: Compiles the code, runs tests, packages the application, and installs it in your local Maven repository (.m2 folder).
   * Post-build Actions (Optional but Recommended):
     + Click "Add post-build action" -> "Archive the artifacts".
     + Files to archive: Enter target/\*.jar (or target/\*.war if it's a web application). This will save the built JAR/WAR file in Jenkins.
   * Click "Save".
4. Build the Job:
   * On the job page, click "Build Now".
5. View the Results:
   * Click on the build number (e.g., "#1") in the "Build History".
   * Click "Console Output" to see the build logs. Check for "BUILD SUCCESS" at the end.
   * If you configured artifact archiving, you'll see a "Last Successful Artifacts" section on the build page where you can download the JAR/WAR file.

Example pom.xml (After project creation):

XML

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>my-jenkins-project</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>my-jenkins-project</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

This comprehensive1 guide should help you set up your Jenkins build process successfully. Remember to replace placeholder values (URLs, paths, etc.) with your actual values. If you encounter errors, carefully review the console output and Jenkins logs for specific error messages.

**Troubleshooting:**

* **Git Errors:** Double-check your repository URL and credentials. Make sure Git is correctly installed and configured on your Windows machine.
* **Maven Errors:** Verify that your JAVA\_HOME, M2\_HOME, and M2 environment variables are correctly set. Test Maven from the command line by running mvn -v.
* **Jenkins Errors:** Check the Jenkins logs for more detailed error messages. You can find the logs in the Jenkins home directory (usually C:\Users\<YourUser>\.jenkins).