TASK 8 — Run a Simple Java Maven Build Job in Jenkins

Objective: Learn how to use Jenkins to build a simple Java application with Maven.

Purpose of This Project: Run a Simple Java Maven Build in Jenkins

🔹 **1. Goal / Objective**

The main purpose is to **learn how Continuous Integration (CI)** works — by making Jenkins automatically build a simple Java application using **Maven**.

This project teaches:

- How Jenkins automates a Java build (no manual javac or mvn in terminal).
- How Maven manages dependencies and compiles your app.
- How Jenkins pipelines or jobs can detect, build, and verify code.

Deliverables

- A basic Java HelloWorld app (src/main/java/HelloWorld.java) and pom.xml
- A Jenkins Freestyle job configured to build the project
- Screenshot of successful build console output

Prerequisites

- Docker (optional) or a local Jenkins installation
- Java JDK 8 or 11 installed on the Jenkins agent (or controller if running builds there)
- Maven (will be configured in Jenkins Global Tool Configuration)
- Git (optional: if you store the repo in Git)

Repository layout (sample):

```
hello-java-maven/
 ├– app/
   └–pom.xml
└– src/
└– main/java/
          └– HelloWorld.java
```

HelloWorld.java

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello, Jenkins + Maven!");
}
}
```

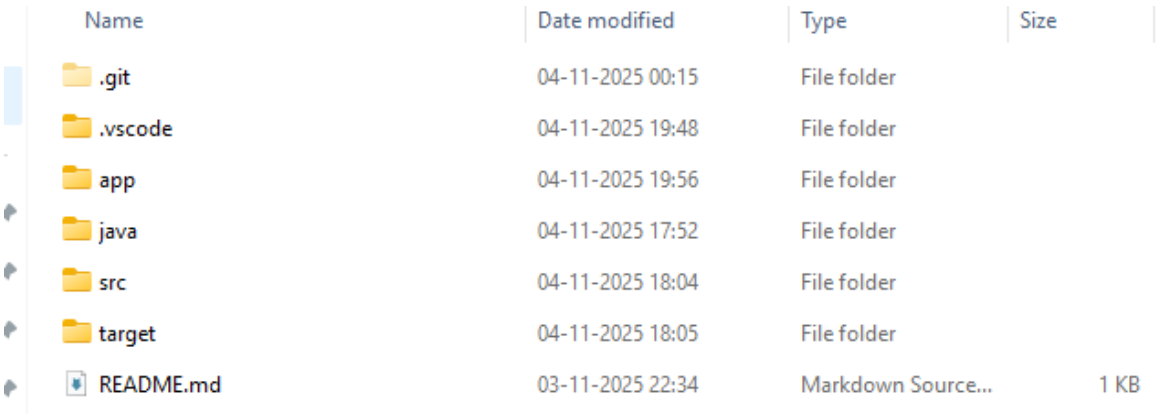
pom.xml

```
<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>hello</artifactId>
<version>1.0</version>
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.1</version>
<configuration>
<source>1.8</source>
<target>1.8</target>
</configuration>
</plugin>
</plugins>
</build>
</project>
```

Step-by-step Guide :

1) Prepare the project

- Create the folder structure above or clone the sample repo hello-java-maven.
- Make sure HelloWorld.java and pom.xml are at the correct paths.



2) (Optional) Run locally to verify

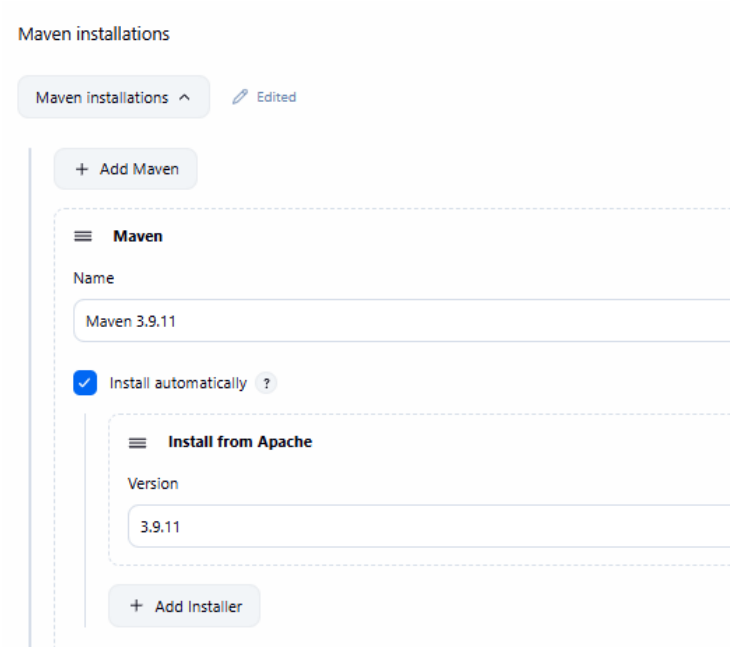
- From project root run:
  mvn -v
  mvn clean package

  The compiled .class files will be generated inside .class file:

```
PS D:\Priti_Data\elevatelab\Task8> cd java
PS D:\Priti_Data\elevatelab\Task8\java> javac HelloWorld.java
PS D:\Priti_Data\elevatelab\Task8\java> java HelloWorld
Hello Priti, Welcome to Jenkins + Maven Program!
PS D:\Priti_Data\elevatelab\Task8\java>
```

HelloWorld.class	04-11-2025 17:52	CLASS File	1 KB
HelloWorld.java	04-11-2025 00:16	Java Source File	1 KB

Go to Manage Jenkins → Global Tool Configuration → Add Maven (e.g., Maven 3.8.6)
Create a new Freestyle project



Step 1: Make sure Docker is installed & running

Start Jenkins (use Docker if needed): docker run -p 8080:8080 jenkins/jenkins:lts how to run this

```
PS D:\Priti_Data\elevatelab\Task8> docker --version
Docker version 28.5.1, build e180ab8
```

Step 2: Pull Jenkins image (optional but recommended)

```
PS D:\Priti_Data\elevatelab\Task8> docker pull jenkins/jenkins:lts
lts: Pulling from jenkins/jenkins
Digest: sha256:f2519b99350faeaaeef30e3b8695cd6261a5d571c859ec37c7ce47e5a241458d
Status: Image is up to date for jenkins/jenkins:lts
docker.io/jenkins/jenkins:lts
```

Step 3: Run Jenkins Container

```
Run 'docker run --help' for more information
PS D:\Priti_Data\elevatelab\Task8> taskkill /PID 8140 /F
ERROR: The process "8140" not found.
PS D:\Priti_Data\elevatelab\Task8> docker run -p 9090:8080 jenkins/jenkins:lts
Running from: /usr/share/jenkins/jenkins.war
webroot: /var/jenkins_home/war
2025-11-04 13:06:26.531+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war f
2025-11-04 13:06:27.627+0000 [id=1]     WARNING o.e.j.ee9.nested.ContextHandler#setContextPath: Empty contex
2025-11-04 13:06:27.674+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-12.0.25; buil
1T23:52:37.219Z; git: a862b76d8372e24205765182d9ae1d1d333ce2ea; jvm 21.0.8+9-LTS
2025-11-04 13:06:28.093+0000 [id=1]     INFO    o.e.j.e.w.StandardDescriptorProcessor#visitServlet: NO JSP S
, did not find org.eclipse.jetty.ee9.jsp.JettyJspServlet
2025-11-04 13:06:28.145+0000 [id=1]     INFO    o.e.j.s.DefaultSessionIdManager#doStart: Session workerName=
2025-11-04 13:06:28.577+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory
ns_home found at: EnvVars.masterEnvVars.get("JENKINS_HOME")
2025-11-04 13:06:28.763+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started oeje9n.Conte
```

Step 4: Alternative — change Jenkins port (recommended if 8080 is busy)

Instead of killing that process, you can **just start Jenkins on another port** (super safe and easy):

Run it from bash:

It creates a container as jenkins-lts

docker run -d --name jenkins-lts -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts

Then open:

http://localhost:8080

Note if***Jenkins will work fine on port 9090 instead of 8080.

Step 1. Identify your Jenkins container name

```
PS D:\Priti_Data\elevatelab\Task8> docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS         PORTS
                NAMES
06c18926c52c   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   4 minutes ago   Up 4 minutes   0.0.0.0:8080->8080/tcp, [:
]:8080->8080/tcp   funny_ardinghelli
PS D:\Priti_Data\elevatelab\Task8>
```

Step-by-step in the browser UI

1. Open the Jenkins dashboard

Open: http://localhost:8080 → login with your admin user.

2. Create a new Freestyle project

1. On the left, click **New Item**.
2. In **Enter an item name** type: hello-java-maven-build (or any name).
3. Select the **Freestyle project**.
4. Click **OK**.

Step 2: Update Jenkins job (if needed)

1. In Jenkins, open your project (**hello-java-maven-build**).
2. Click **Configure** (left sidebar).
3. Scroll down to the **Build → Invoke top-level Maven targets** section.
4. In **POM**, enter:
    - pom.xml → if pom.xml is in root
    - app/pom.xml → if it's inside app/ folder
5. Save **→ Click Build Now**

Step 3: What You Learn

- How Jenkins works
- How to integrate Maven with Jenkins
- How Jenkins uses pom.xml to build a Java app
- How CI tools automate repetitive developer tasks

In simple words:

This project teaches you how to use Jenkins to automatically build a Java project using Maven — your first practical step into the world of Continuous Integration and DevOps automation.

That command shows Jenkins is running Maven correctly 👏

Now, since it still says:

mvn.cmd -f app/pom.xml clean package

that means Jenkins is trying to build your project from app/pom.xml, but it might not exist at that path.

```
PS D:\Priti_Data\elevatelab\Task8> mvn.cmd -f app/pom.xml clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.example:hello-priti >-----------------------
[INFO] Building hello-priti 1.0
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/p
```

```
fire-shared-utils-3.2.5.jar (2.4 MB at 1.9 MB/s)
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ hello-priti ---
Downloading from central: https://repo.maven.apache.org/mav
ment-3.1.0.pom
```

Step 3: Expected output

If everything is correct, the console log should end with:

BUILD SUCCESS

```
s-1.26.1.jar (1.1 MB at 140 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/github/luben/zstd
6.8 MB at 773 kB/s)
[WARNING] JAR will be empty - no content was marked for inclusion!
[INFO] Building jar: D:\Priti_Data\elevatelab\Task8\app\target\hello-priti-1.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  23.867 s
[INFO] Finished at: 2025-11-04T19:56:46+05:30
[INFO] ------------------------------------------------------------------------
PS D:\Priti_Data\elevatelab\Task8>
```