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

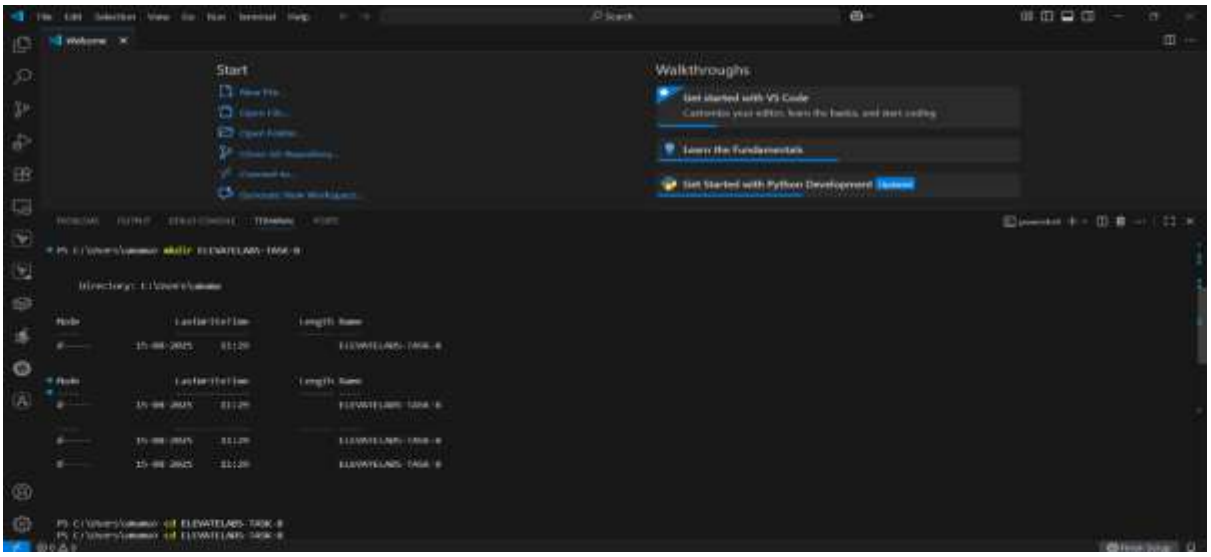
Objective: Learn how to use Jenkins to build a simple Java application using Maven a first step into CI/CD.

Tools Used

- Jenkins (via Docker)
- Java JDK
- Maven
- Git & GitHub

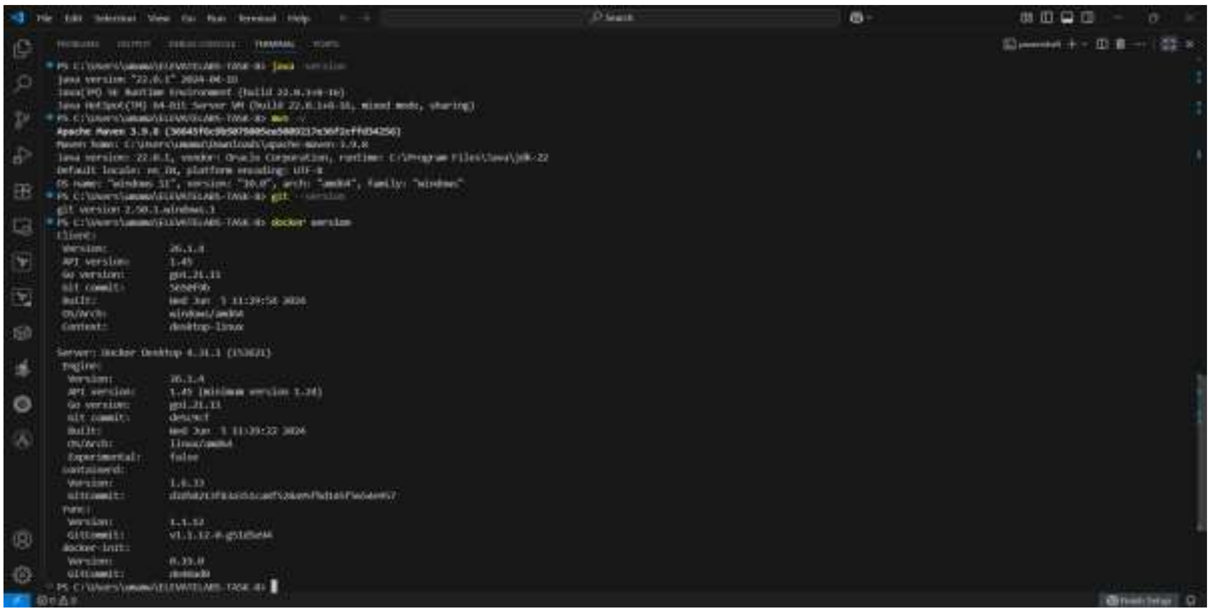
Steps Followed:

Created the project folder locally



```
PS C:\Users\[illegible]> mkdir ELEVATELABS-TASK-8

    Directory: C:\Users\[illegible]

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          15-08-2025     11:20                ELEVATELABS-TASK-8

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          15-08-2025     11:20                ELEVATELABS-TASK-8

d----          15-08-2025     11:20                ELEVATELABS-TASK-8
d----          15-08-2025     11:20                ELEVATELABS-TASK-8

PS C:\Users\[illegible]> cd ELEVATELABS-TASK-8
PS C:\Users\[illegible]> cd ELEVATELABS-TASK-8
```

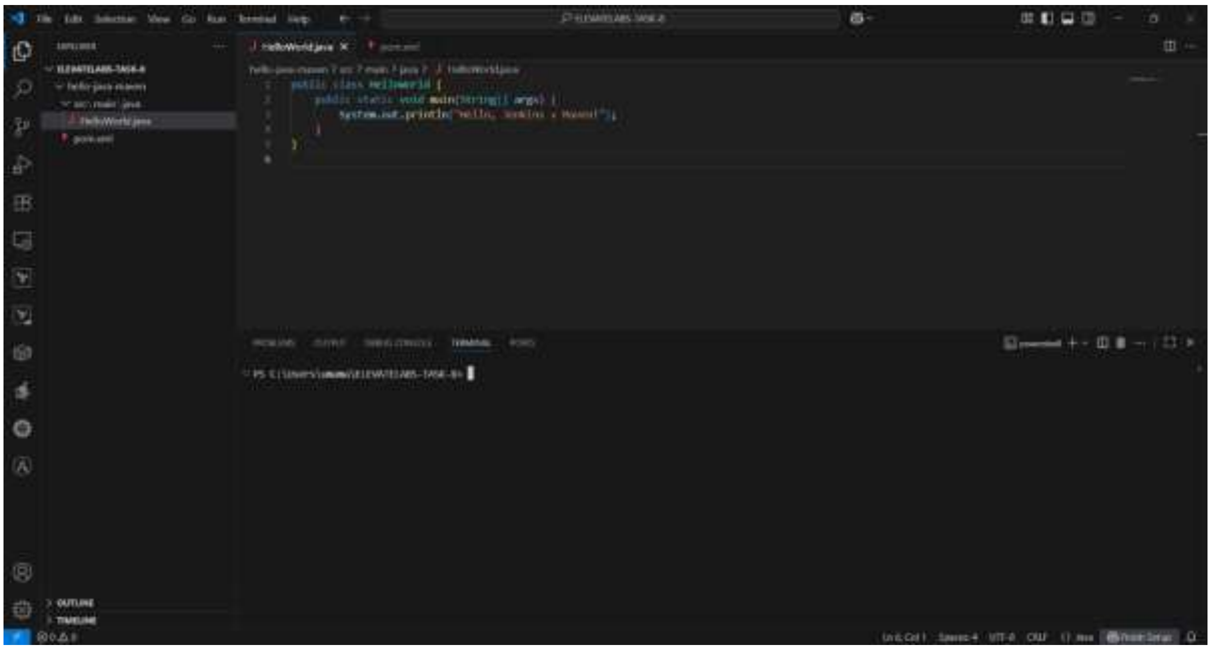
Install needed tools already I have I am checking its versions



```
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> java -version
java version "22.0.1" 2024-04-16
Java(TM) SE Runtime Environment (build 22.0.1+8-16)
Java HotSpot(TM) 64-Bit Server VM (build 22.0.1+8-16, mixed mode, sharing)
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> mvn -v
Apache Maven 3.9.8 ([illegible])
Maven home: C:\Users\[illegible]\Downloads\apache-maven-3.9.8
Java version: 22.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-22
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> git --version
git version 2.50.1.windows.1
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> docker version
Client:
 Version:           26.1.4
 API version:       1.45
 Go version:        go1.21.11
 Git commit:        5650f9b
 Built:             Wed Jun  5 11:29:58 2024
 OS/Arch:           windows/amd64
 Context:           desktop-linux

Server: Docker Desktop 4.31.1 (153621)
 Engine:
  Version:          26.1.4
  API version:      1.45 (minimum version 1.24)
  Go version:       go1.21.11
  Git commit:       de5c9cf
  Built:            Wed Jun  5 11:29:22 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.33
  GitCommit:        [illegible]
 runc:
  Version:          1.1.12
  GitCommit:        v1.1.12-0-g51d5e94
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
PS C:\Users\[illegible]\ELEVATELABS-TASK-8>
```

Create Java App: File: src/main/java/HelloWorld.java

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



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

```
PS C:\Users\[illegible]\ELEVATELABS-TASK-8>
```

Explanation:

package com.example; – Defines the namespace for the Java class.

public class HelloWorld – Declares a public class named HelloWorld.

public static void main(String[] args) – Entry point for the program; this method runs when the Java application starts.

System.out.println(...) – Prints the message to the console output.

Create pom.xml

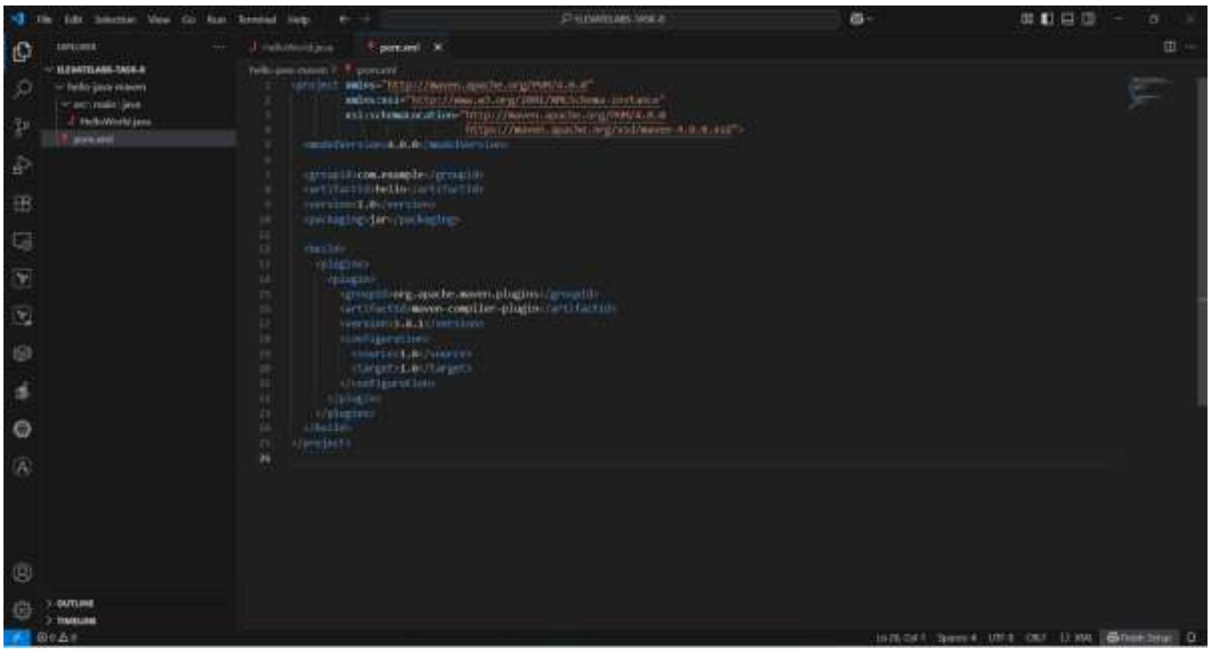
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                    https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
```

```
<groupId>com.example</groupId>
<artifactId>hello</artifactId>
<version>1.0</version>
<packaging>jar</packaging>
<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>
```

POM File Code Explanation:

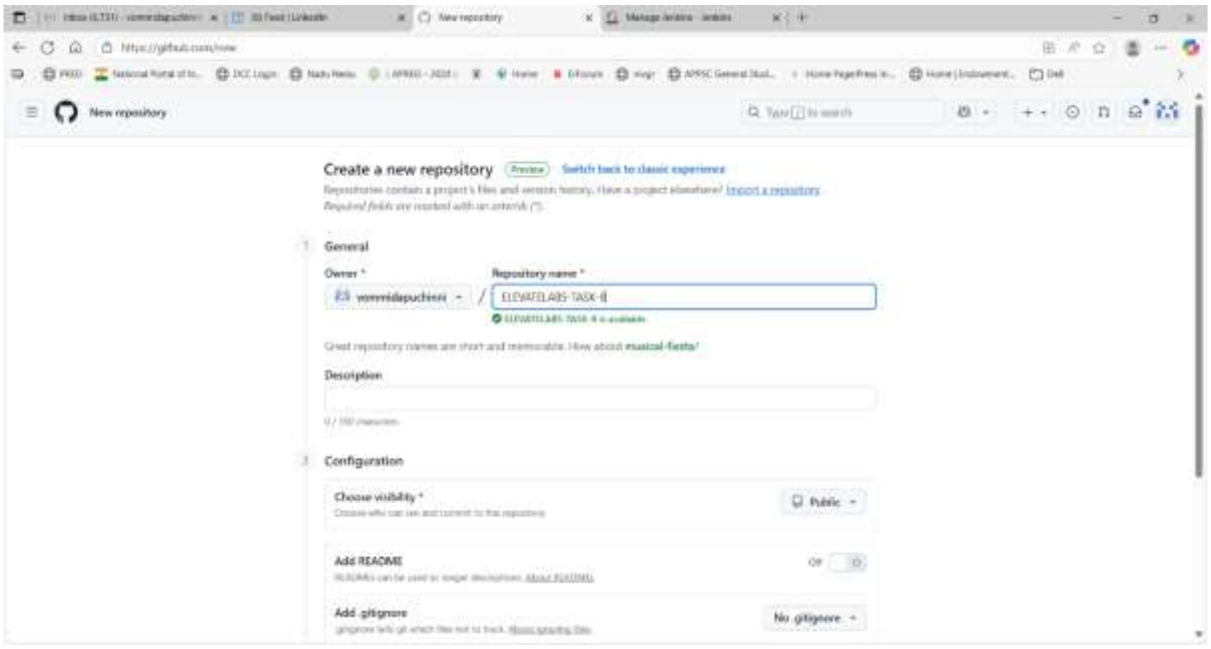
- <project> – Root element of the Maven Project Object Model (POM) file.
- <modelVersion> – Maven POM model version used (here 4.0.0).
- <groupId> – Unique ID for the project's group or organization (com.example).
- <artifactId> – The project's name or module (hello).
- <version> – Version of the project (1.0).
- <packaging> – Specifies output type (jar).
- <build> – Section containing build-related configurations.
- <plugins> – List of Maven plugins used during build.
- <plugin> – Defines a specific plugin configuration.
    - <groupId> – Group ID for the plugin (org.apache.maven.plugins).

- <artifactId> – Plugin name (maven-compiler-plugin).
- <version> – Plugin version (3.8.1).
- <configuration> – Settings for the plugin.
  - <source> – Java source version (1.8).
  - <target> – Java bytecode version (1.8).



```
<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
                             https://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>
  <packaging>jar</packaging>

  <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>
```

Create git repo



Create a new repository

General

Owner *: vommidapuchinni / Repository name *: ELEVATELABS-TASK-8

Configuration

Choose visibility *: Public

Add README: Off

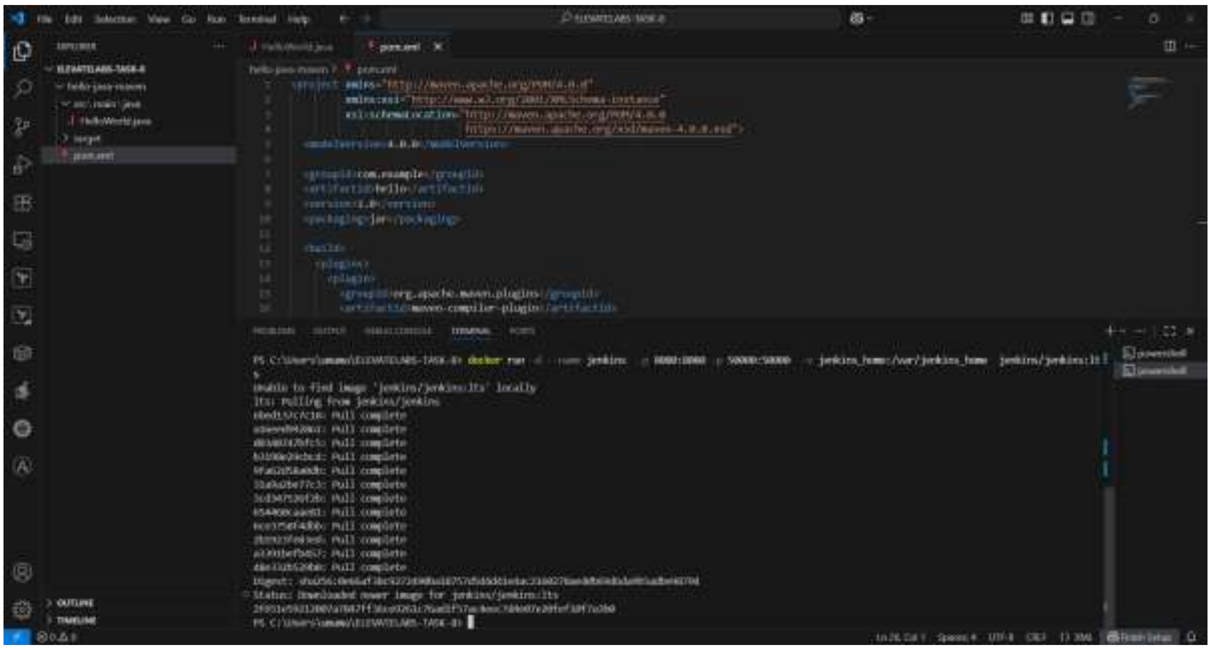
Add .gitignore: No .gitignore

Push the files to repo

Run Jenkins in Docker:

docker run --name jenkins -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts

- **8080** → Jenkins Web Interface
- **50000** → Jenkins Agent communication

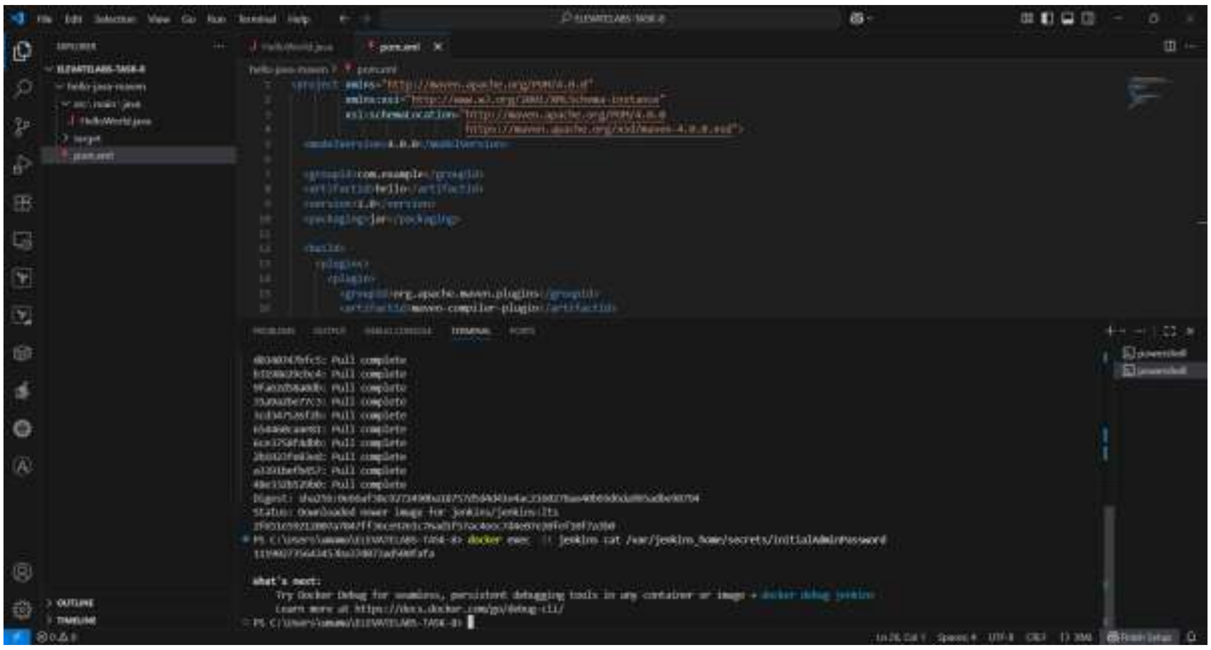


```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>hello</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> docker run -d --name jenkins -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/jenkins:lt
s
Unable to find image 'jenkins/jenkins:lts' locally
lts: Pulling from jenkins/jenkins
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
Digest: sha256:[illegible]
Status: Downloaded newer image for jenkins/jenkins:lts
[illegible]
PS C:\Users\[illegible]\ELEVATELABS-TASK-8>
```

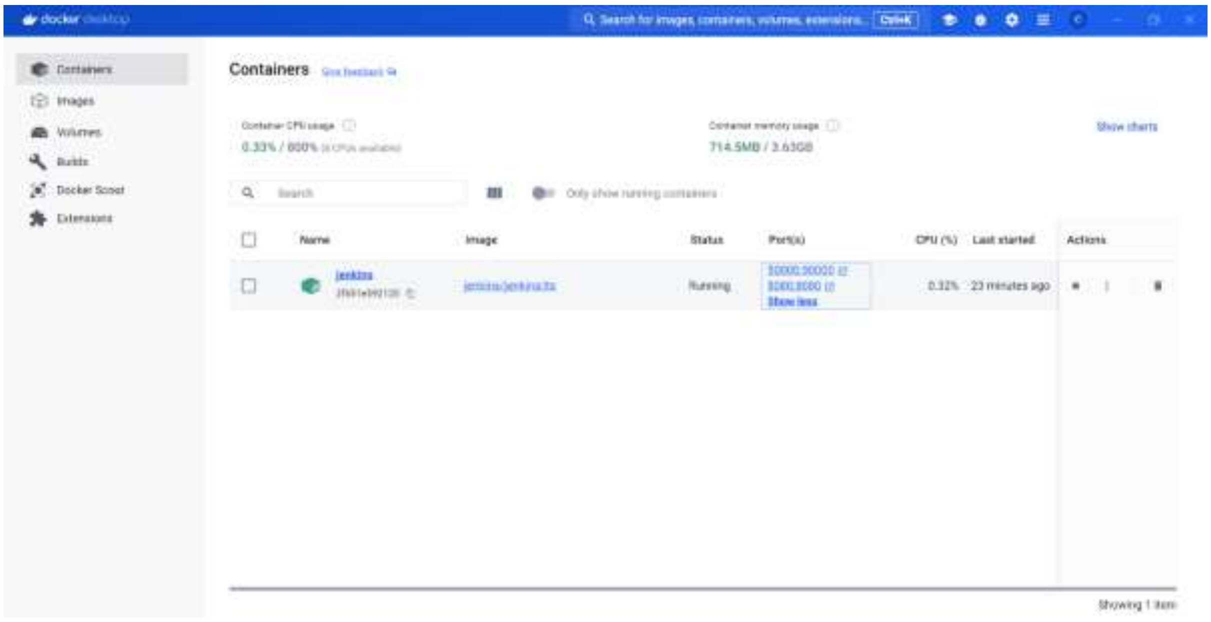
Unlock Jenkins:

docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword



```
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
[illegible]: Pull complete
Digest: sha256:[illegible]
Status: Downloaded newer image for jenkins/jenkins:lts
[illegible]
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
[illegible]

What's next:
    Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug jenkins
    Learn more at https://docs.docker.com/go/debug-cli/
PS C:\Users\[illegible]\ELEVATELABS-TASK-8>
```

In docker desktop we see Jenkins container and image



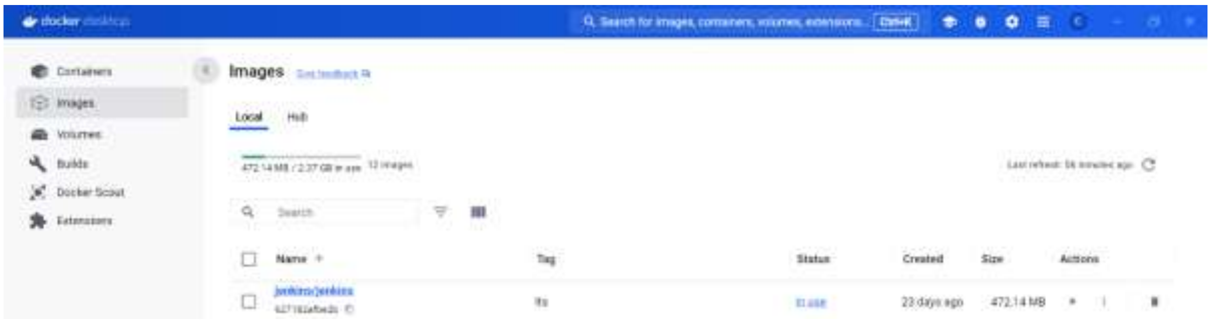
Containers

Container CPU usage: 0.33% / 800% (8 CPUs available)

Container memory usage: 714.5MB / 3.63GB

Name	Image	Status	Port(s)	CPU (%)	Last started
jenkins	jenkins/jenkins:lts	Running	50000:50000, 8080:8080	0.32%	23 minutes ago

Showing 1 item

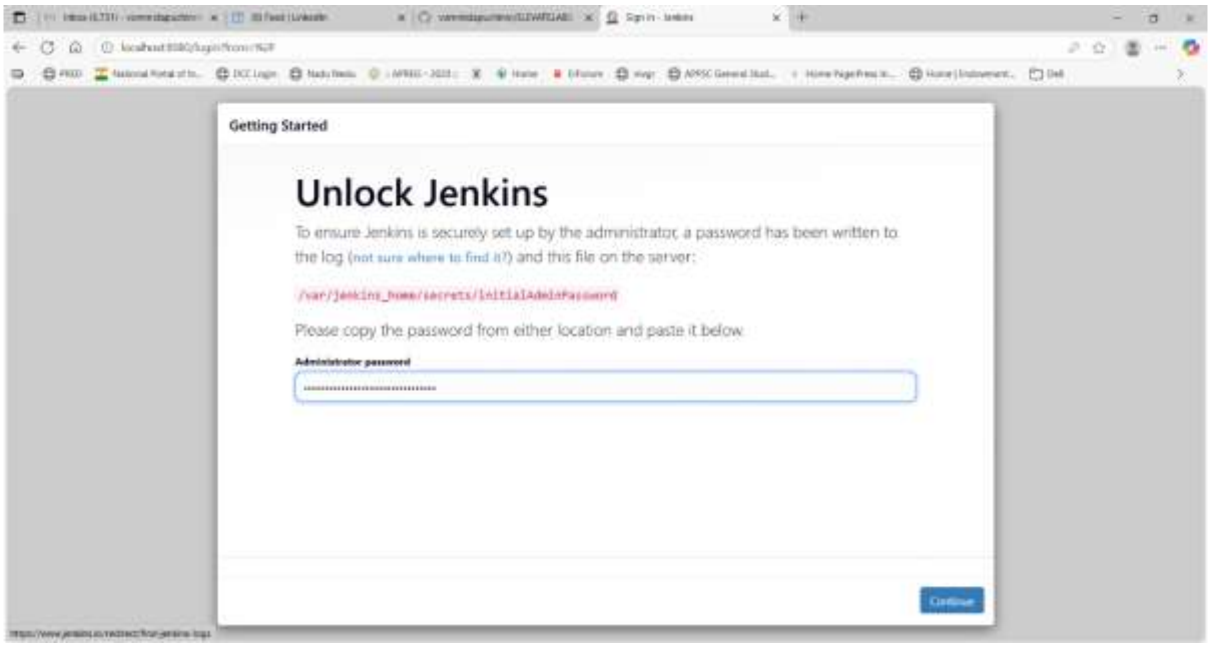


Images

472.14 MB / 2.37 GB in use 12 images

Name	Tag	Status	Created	Size
jenkins/jenkins	lts	In use	23 days ago	472.14 MB

Access Jenkins dashboard with http://localhost:8080



Getting Started

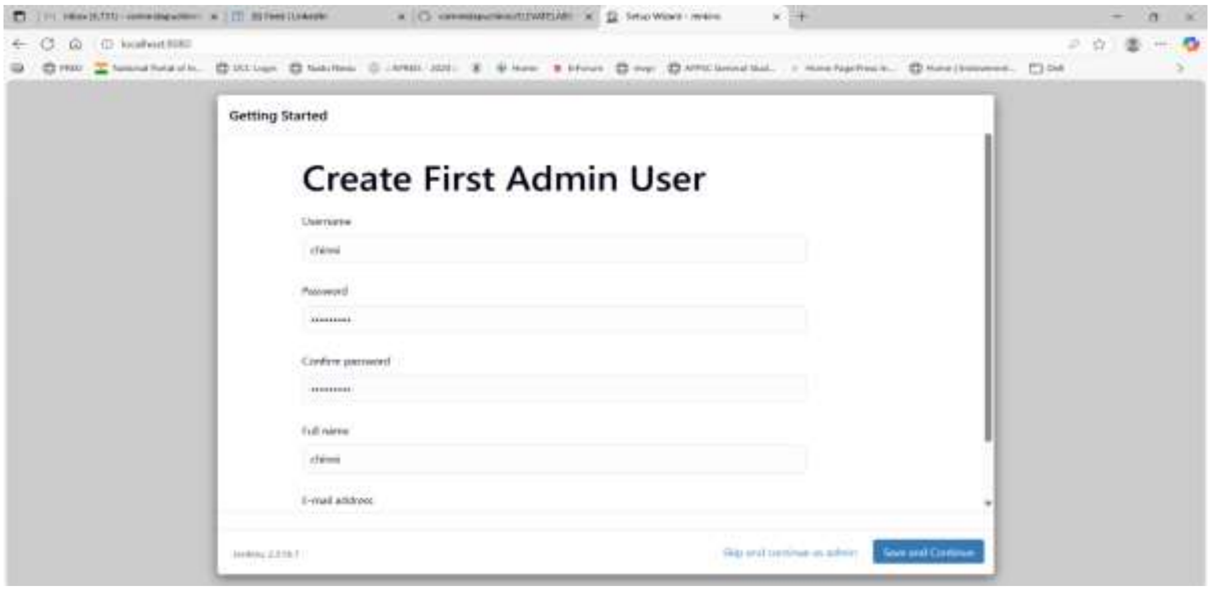
Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password


Getting Started

Create First Admin User

Username: chinni

Password: ••••••••••

Confirm password: ••••••••••

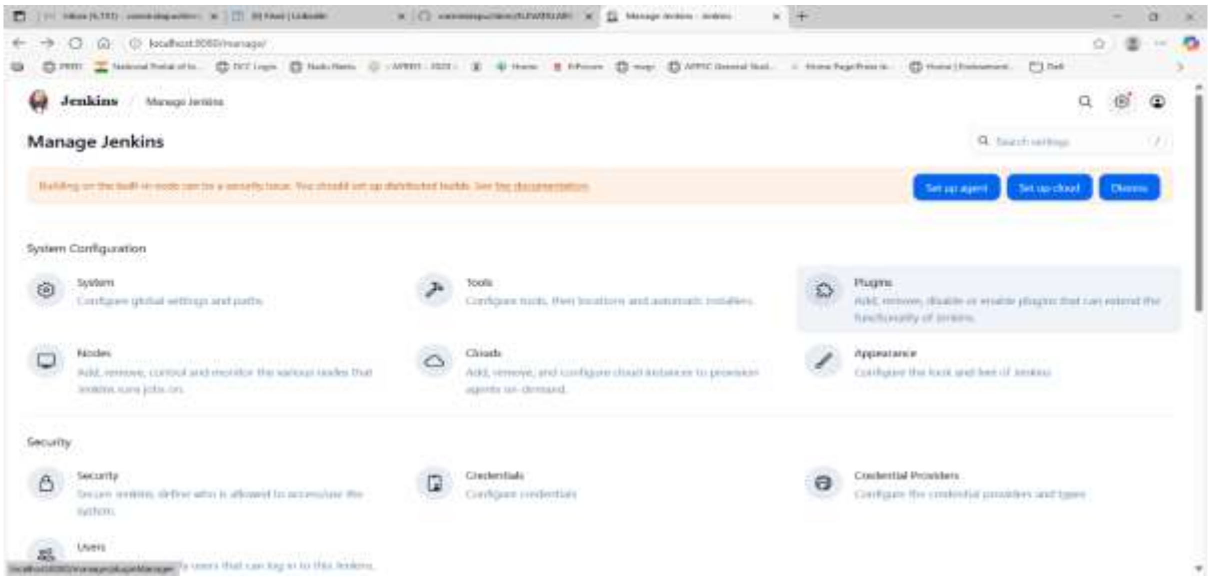
Full name: chinni

E-mail address

Jenkins 2.516.1

Skip and continue as admin | Save and Continue

To integrate we have to install maven plugin for better


localhost:8080/manage/

Jenkins / Manage Jenkins

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

Set up agent | Set up cloud | Dismiss

System Configuration

System — Configure global settings and paths.

Tools — Configure tools, their locations and automatic installers.

Plugins — Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes — Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds — Add, remove, and configure cloud instances to provision agents on-demand.

Appearance — Configure the look and feel of Jenkins

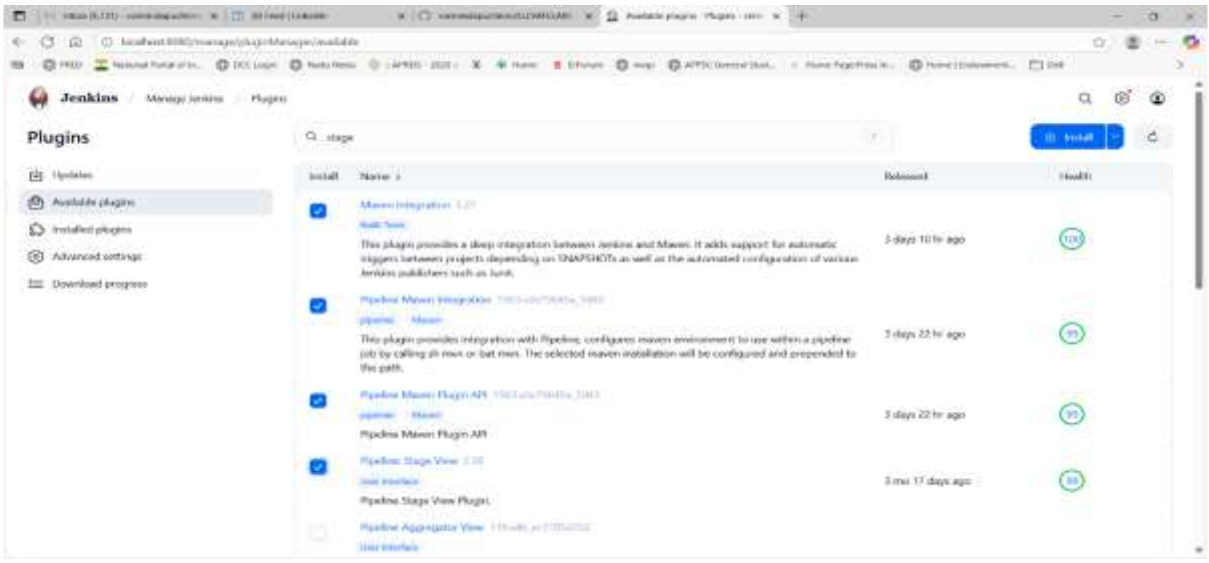
Security

Security — Secure Jenkins; define who is allowed to access/use the system.

Credentials — Configure credentials

Credential Providers — Configure the credential providers and types

Users — users that can log in to this Jenkins


localhost:8080/manage/pluginManager/available

Jenkins / Manage Jenkins / Plugins

Plugins

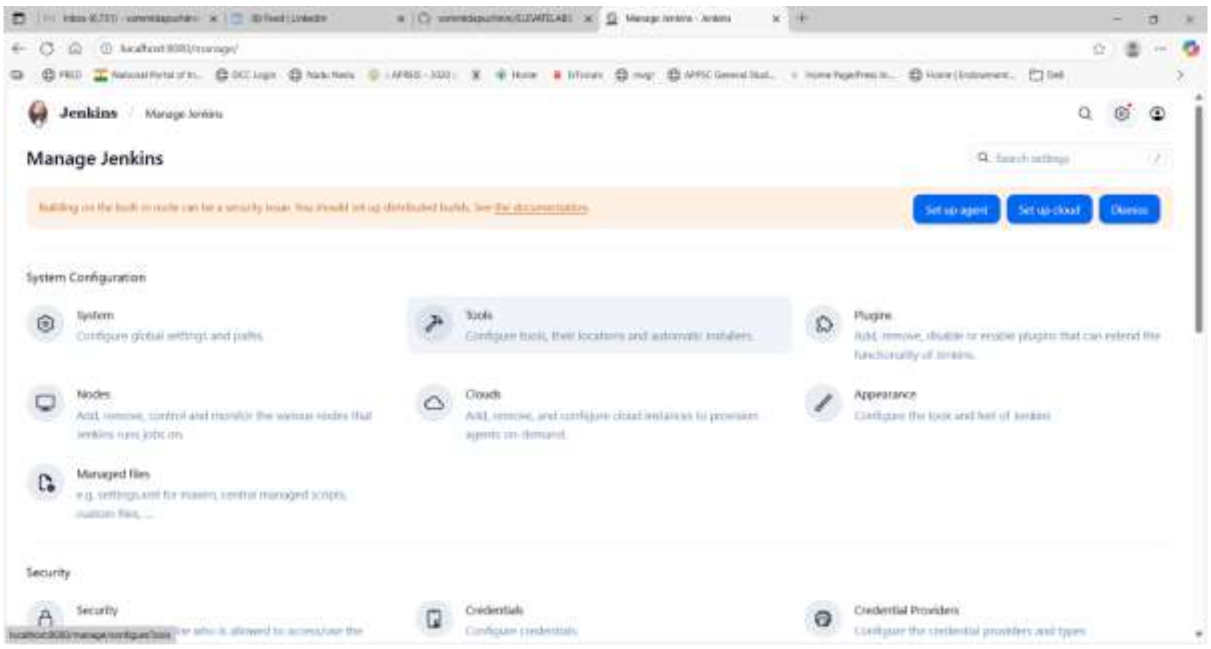
Updates | Available plugins | Installed plugins | Advanced settings | Download progress

Search: stage

Install	Name ↓	Released	Health
✓	Maven Integration 3.27 Build Tools This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.	3 days 10 hr ago	100
✓	Pipeline Maven Integration [illegible] pipeline Maven This plugin provides integration with Pipeline, configures maven environment to use within a pipeline job by calling sh mvn or bat mvn. The selected maven installation will be configured and prepended to the path.	3 days 22 hr ago	99
✓	Pipeline Maven Plugin API [illegible] pipeline Maven Pipeline Maven Plugin API	3 days 22 hr ago	99
✓	Pipeline: Stage View 2.38 User interface Pipeline Stage View Plugin.	3 mo 17 days ago	99
	Pipeline Aggregator View [illegible] User interface		

Configure Java in Jenkins:

Click on tools in manage Jenkins



Jenkins / Manage Jenkins

Manage Jenkins

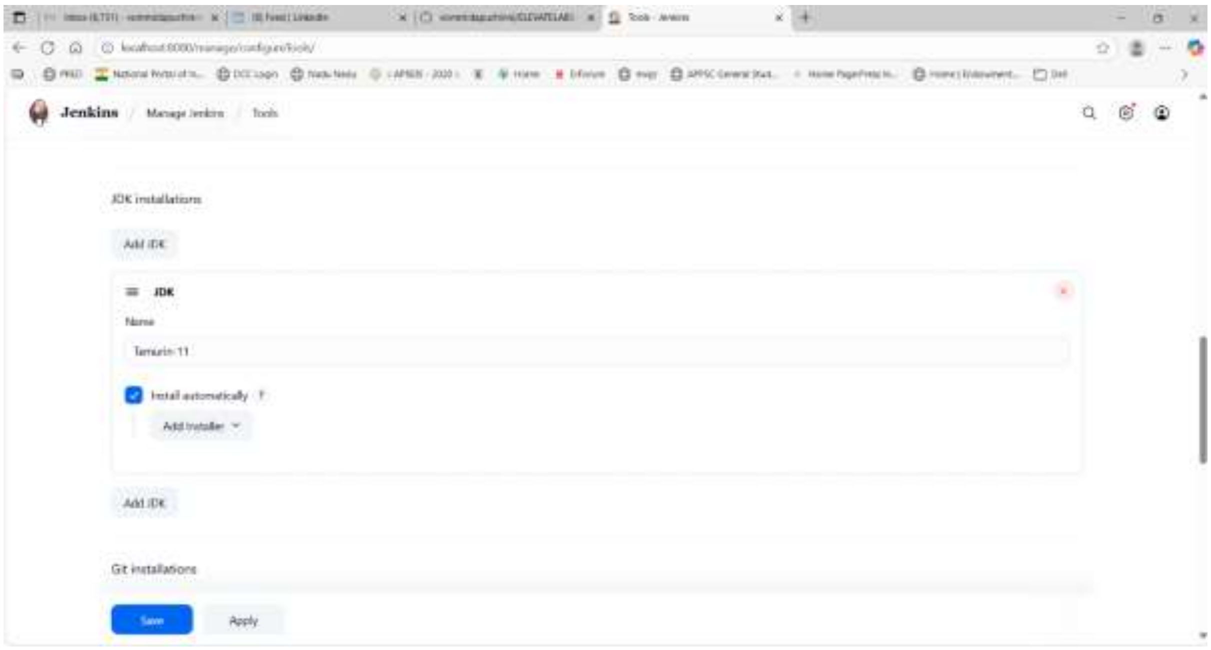
Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

- System — Configure global settings and paths.
- Tools — Configure tools, their locations and automatic installers.
- Plugins — Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes — Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds — Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance — Configure the look and feel of Jenkins
- Managed files — e.g. settings.xml for maven, central managed scripts, custom files, ...

Security

- Security
- Credentials — Configure credentials
- Credential Providers — Configure the credential providers and types



Jenkins / Manage Jenkins / Tools

JDK installations

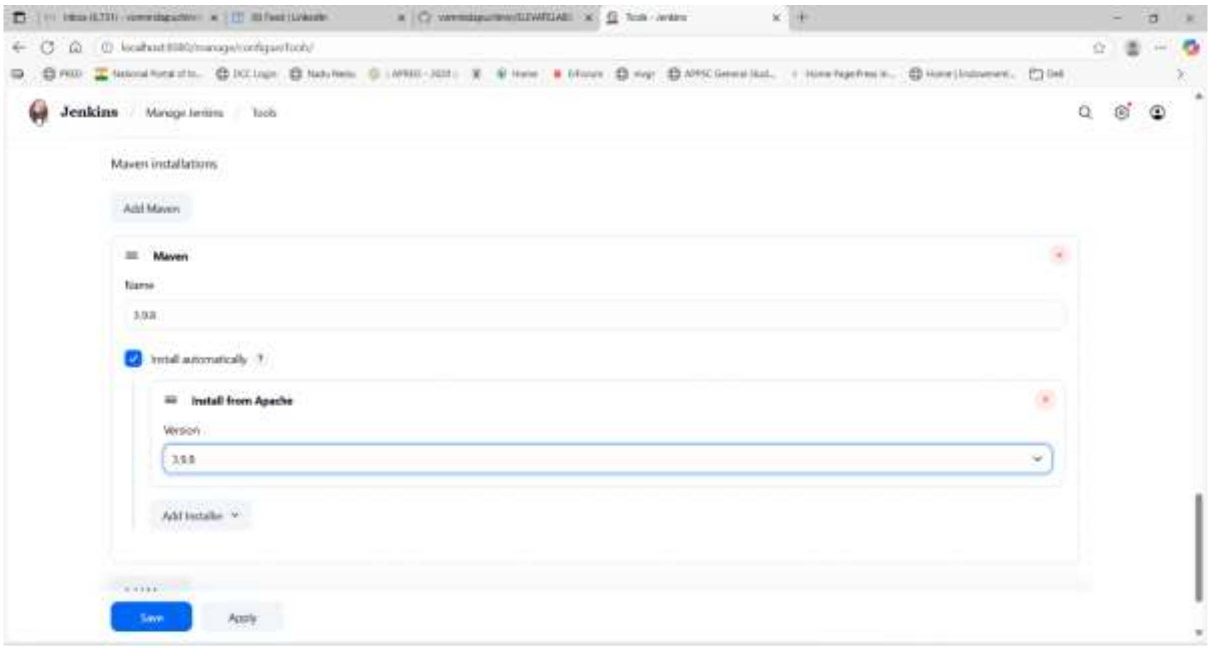
JDK

Name: Temurin 11

[✓] Install automatically

Git installations

- Add Maven → Name: 3.9.8
- Tick Install Automatically


localhost:8080/manage/configureTools/

Jenkins / Manage Jenkins / Tools

Maven installations

Add Maven

Maven

Name

3.9.8

[✓] Install automatically

Install from Apache

Version

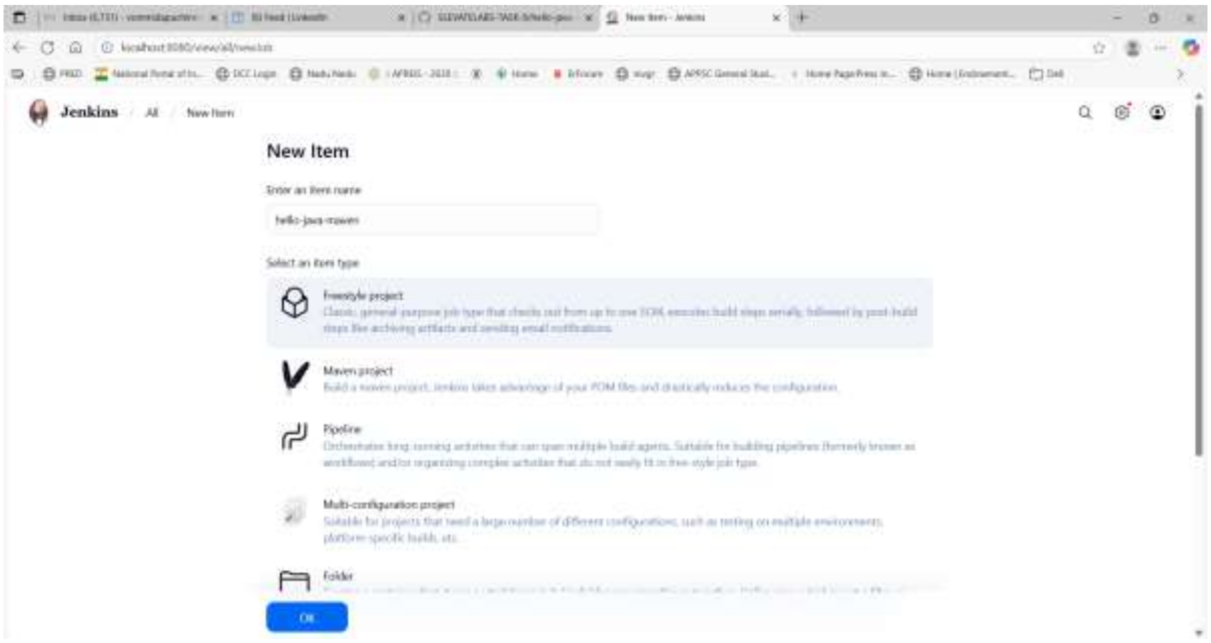
3.9.8

Add Installer

Save   Apply

Create Freestyle Job

- New Item → give name → select free style → click ok


localhost:8080/view/all/newJob

Jenkins / All / New Item

New Item

Enter an item name

hello-java-maven

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.

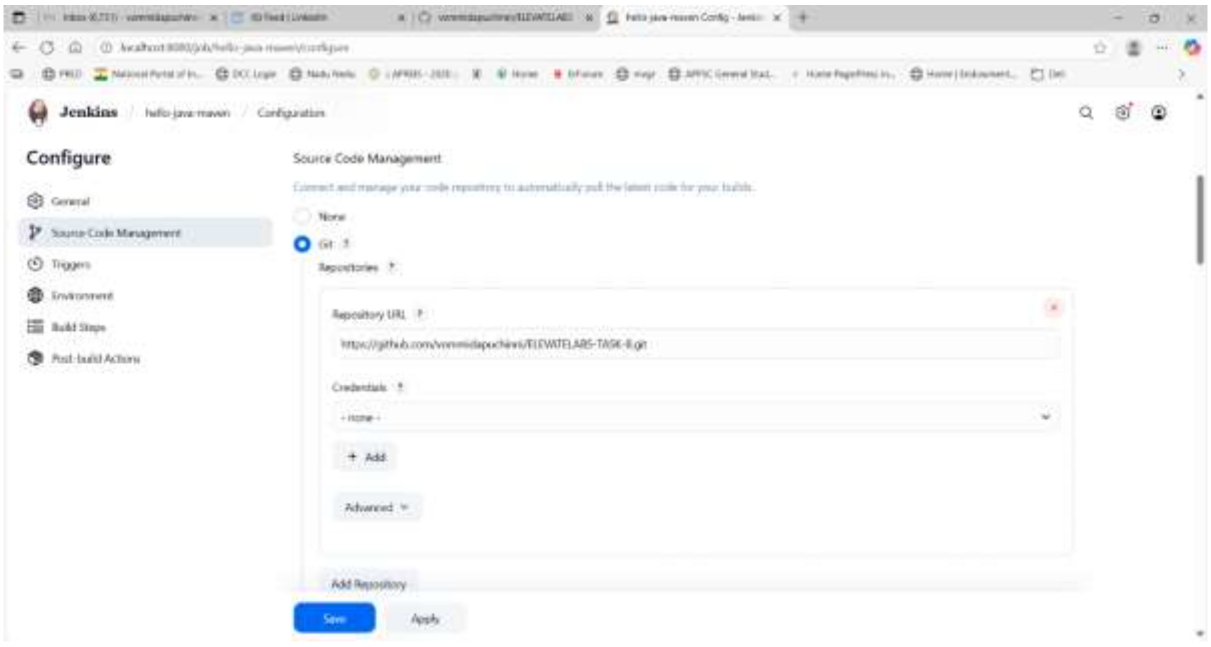
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

OK

- Source Code Management: GitHub repo URL

Select git → give repo url → its public repo no need of credentials


Jenkins / hello-java-maven / Configuration

Configure

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

- None
- [✓ Git]

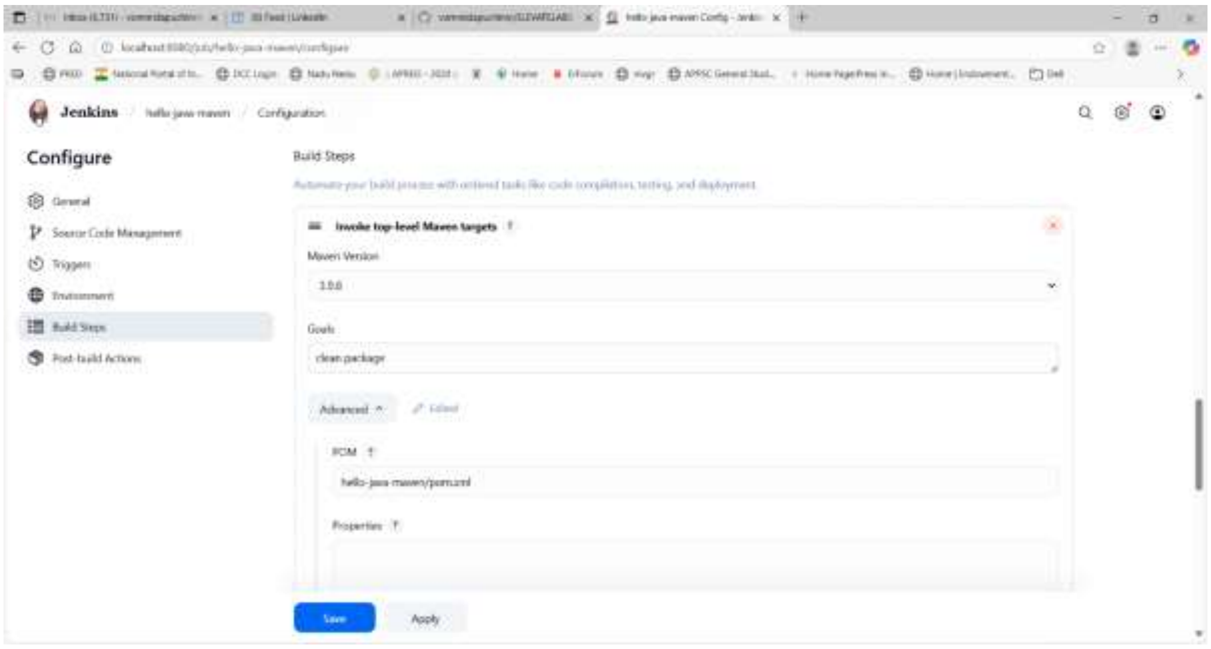
Repositories

Repository URL: https://github.com/vommidapuchinni/ELEWATELABS-TASK-8.git

Credentials: - none -

- Build Step: Invoke top-level Maven targets

Keep maven version → goals keep as clean package → mentioning where is our pom file


Jenkins / hello-java-maven / Configuration

Configure

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Invoke top-level Maven targets

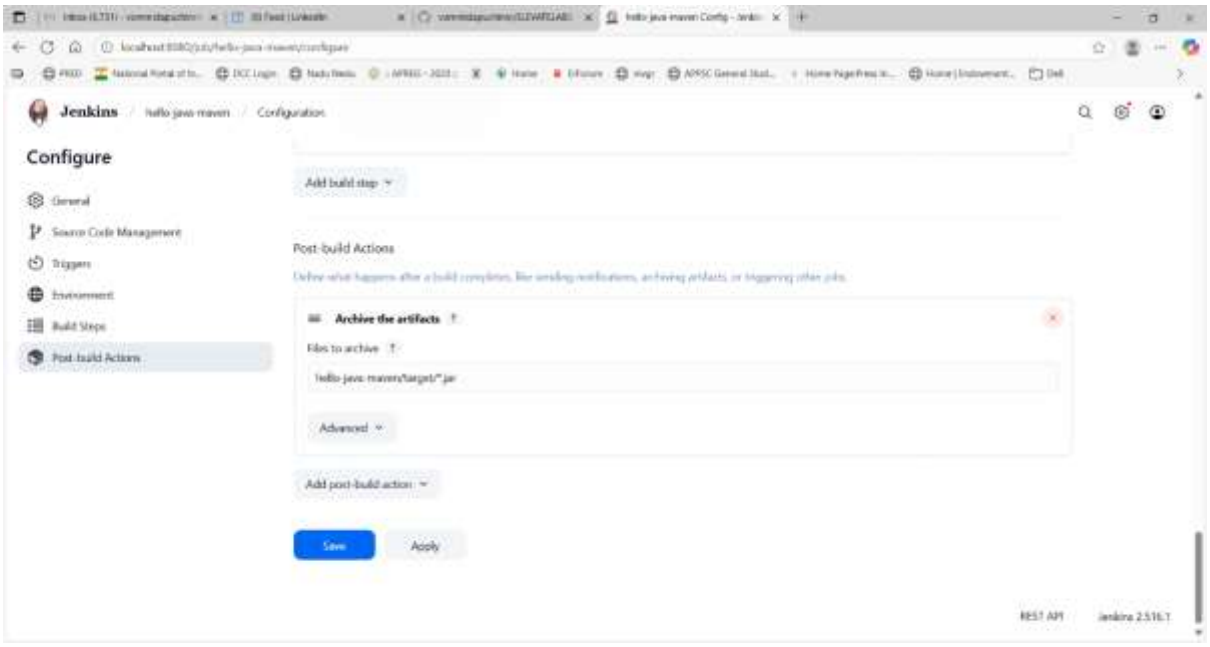
Maven Version: 3.9.6

Goals: clean package

POM: hello-java-maven/pom.xml

Properties:

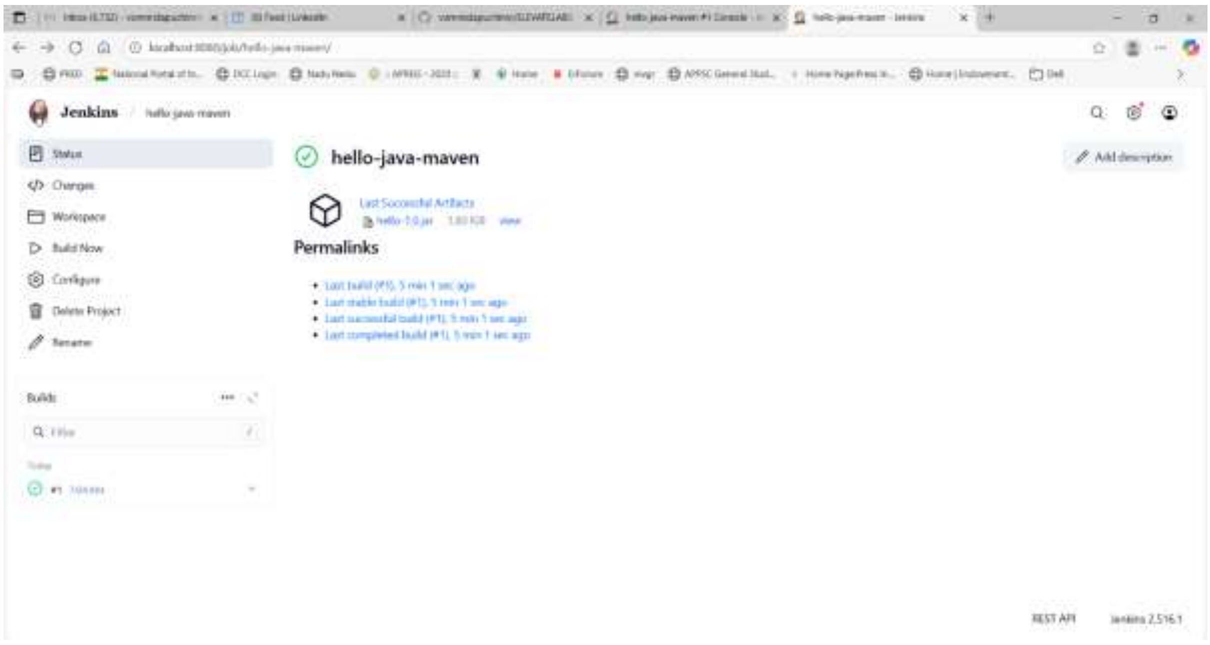
- Goals: clean package
- Added post build steps also artifacts



Click on save

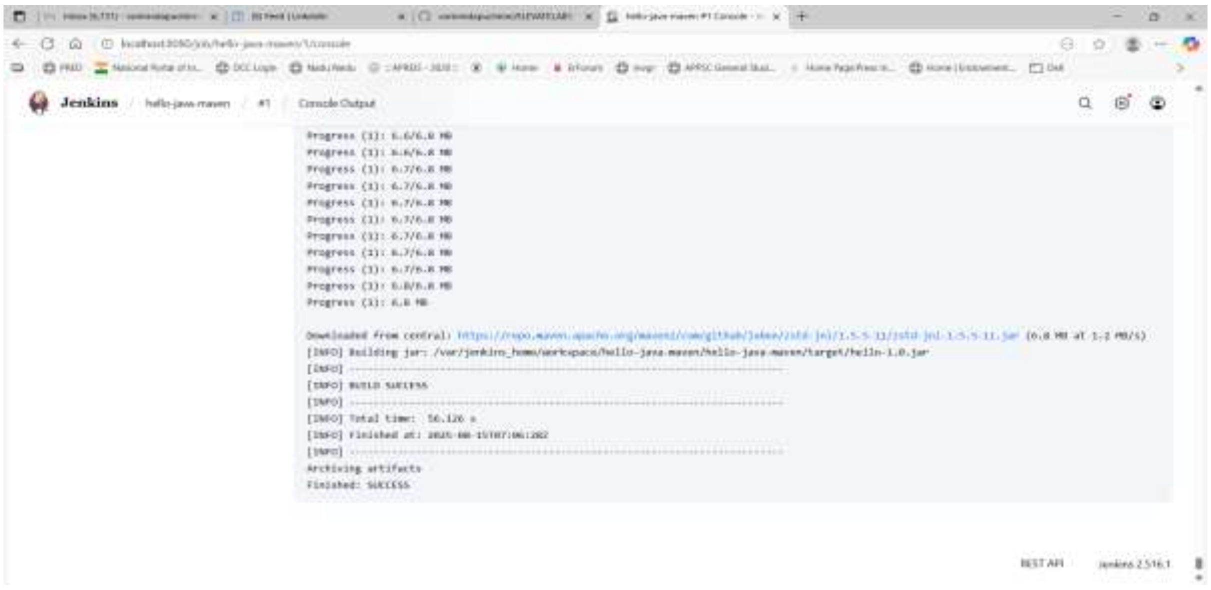
Click on build now

After building we see like this


Jenkins
hello-java-maven
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename
hello-java-maven
Add description
Last Successful Artifacts
hello-1.0.jar
view
Permalinks
Last build (#1), 5 min 1 sec ago
Last stable build (#1), 5 min 1 sec ago
Last successful build (#1), 5 min 1 sec ago
Last completed build (#1), 5 min 1 sec ago
Builds
Filter
#1
REST API
Jenkins 2.516.1

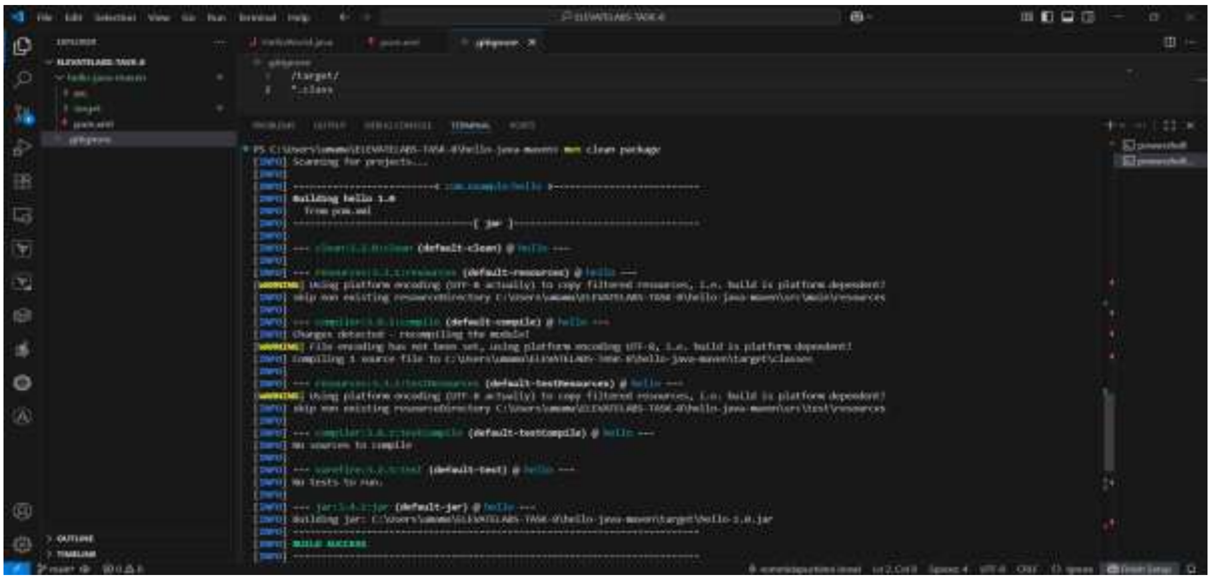
Verify Build

Click on the small icon in build section to see console output



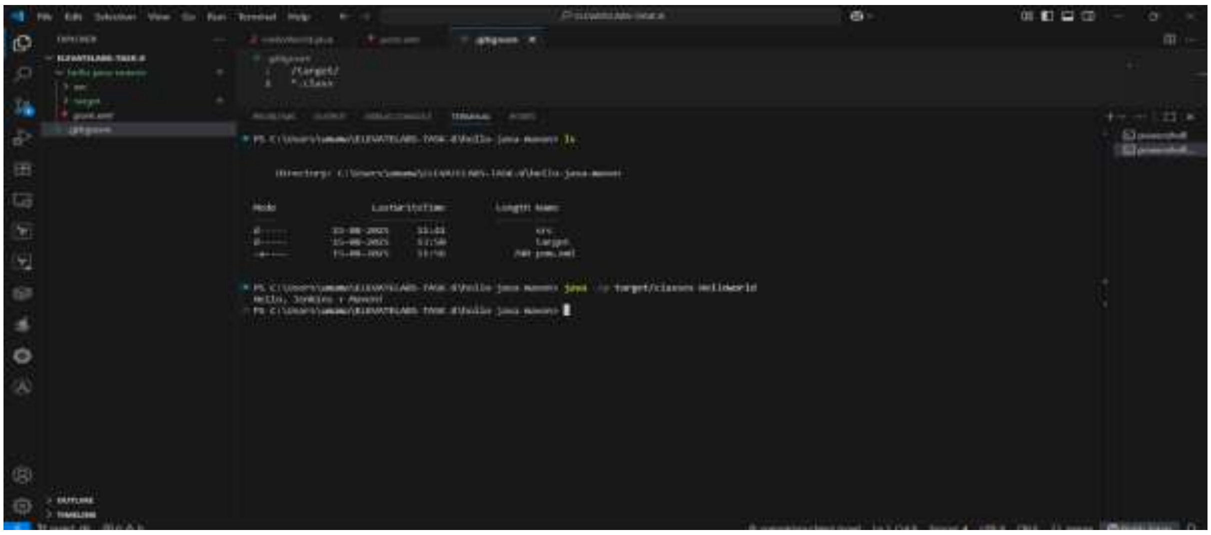
```
Progress (1): 6.6/6.8 MB
Progress (1): 6.6/6.8 MB
Progress (1): 6.7/6.8 MB
Progress (1): 6.7/6.8 MB
Progress (1): 6.7/6.8 MB
Progress (1): 6.7/6.8 MB
Progress (1): 6.7/6.8 MB
Progress (1): 6.7/6.8 MB
Progress (1): 6.7/6.8 MB
Progress (1): 6.8/6.8 MB
Progress (1): 6.8 MB

Downloaded from central: https://repo.maven.apache.org/maven2/com/github/luben/zstd-jni/1.5.5-11/zstd-jni-1.5.5-11.jar (6.8 MB at 1.2 MB/s)
[INFO] Building jar: /var/jenkins_home/workspace/hello-java-maven/hello-java-maven/target/hello-1.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  56.126 s
[INFO] Finished at: 2025-08-15T07:06:28Z
[INFO] ------------------------------------------------------------------------
Archiving artifacts
Finished: SUCCESS
```

I ran locally



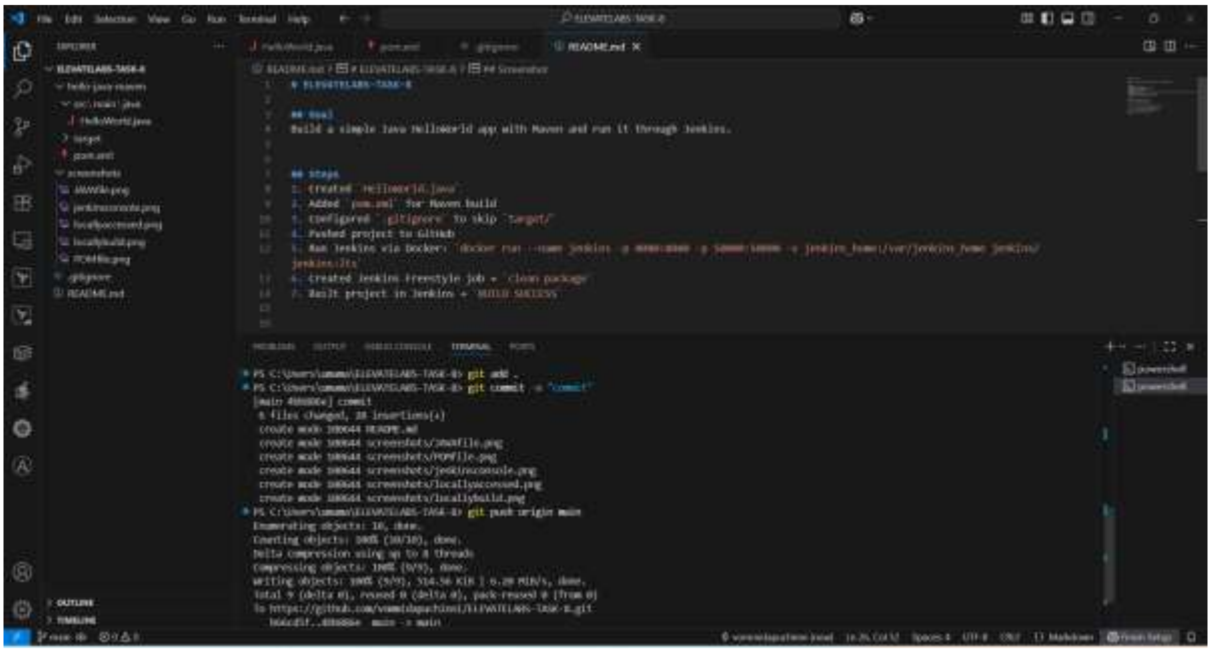
```
PS C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven> mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< com.example:hello >------------------------
[INFO] Building hello 1.0
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ hello ---
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ hello ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven\src\main\resources
[INFO]
[INFO] --- compiler:3.x.x:compile (default-compile) @ hello ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven\target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ hello ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven\src\test\resources
[INFO]
[INFO] --- compiler:3.x.x:testCompile (default-testCompile) @ hello ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ hello ---
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ hello ---
[INFO] Building jar: C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven\target\hello-1.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Tagert file came and we can see the output also



```
PS C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven> ls

    Directory: C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        15-08-2025     11:43                src
d-----        15-08-2025     11:58                target
-a----        15-08-2025     11:58            [illegible] pom.xml

PS C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven> java -cp target/classes HelloWorld
Hello, Jenkins + Maven!
PS C:\Users\[illegible]\ELEVATELABS-TASK-4\hello-java-maven>
```

Add readme file and screenshots and keep targets in .gitignore file



```
# ELEVATELABS-TASK-8

## Goal
Build a simple Java HelloWorld app with Maven and run it through Jenkins.


## Steps
1. Created `HelloWorld.java`
2. Added `pom.xml` for Maven build
3. Configured `.gitignore` to skip `target/`
4. Pushed project to GitHub
5. Ran Jenkins via Docker: `docker run --name jenkins -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/
jenkins:lts`
6. Created Jenkins Freestyle job → `clean package`
7. Built project in Jenkins → `BUILD SUCCESS`
```

```
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> git add .
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> git commit -m "commit"
[main [illegible]] commit
 6 files changed, 28 insertions(+)
 create mode 100644 README.md
 create mode 100644 screenshots/JAVAfile.png
 create mode 100644 screenshots/POMfile.png
 create mode 100644 screenshots/jenkinsconsole.png
 create mode 100644 screenshots/locallyaccessed.png
 create mode 100644 screenshots/locallybuild.png
PS C:\Users\[illegible]\ELEVATELABS-TASK-8> git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 314.56 KiB | 6.20 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/[illegible]/ELEVATELABS-TASK-8.git
   [illegible]..[illegible]  main -> main
```

key concepts

- Maven is used to build and manage the Java project.
- pom.xml defines project dependencies, plugins, and build configuration.
- src/main/java contains the source code (HelloWorld.java).
- target/ stores compiled .class files and the generated .jar file.
- .gitignore is used to exclude unnecessary files from Git (like target/).
- Jenkins automates build and deployment using a pipeline.

Conclusion:

This project demonstrates building a simple Java application using Maven, automating the build with Jenkins, and running the generated .jar file locally. It covers source code organization, dependency management, artifact creation, and version control best practices using .gitignore