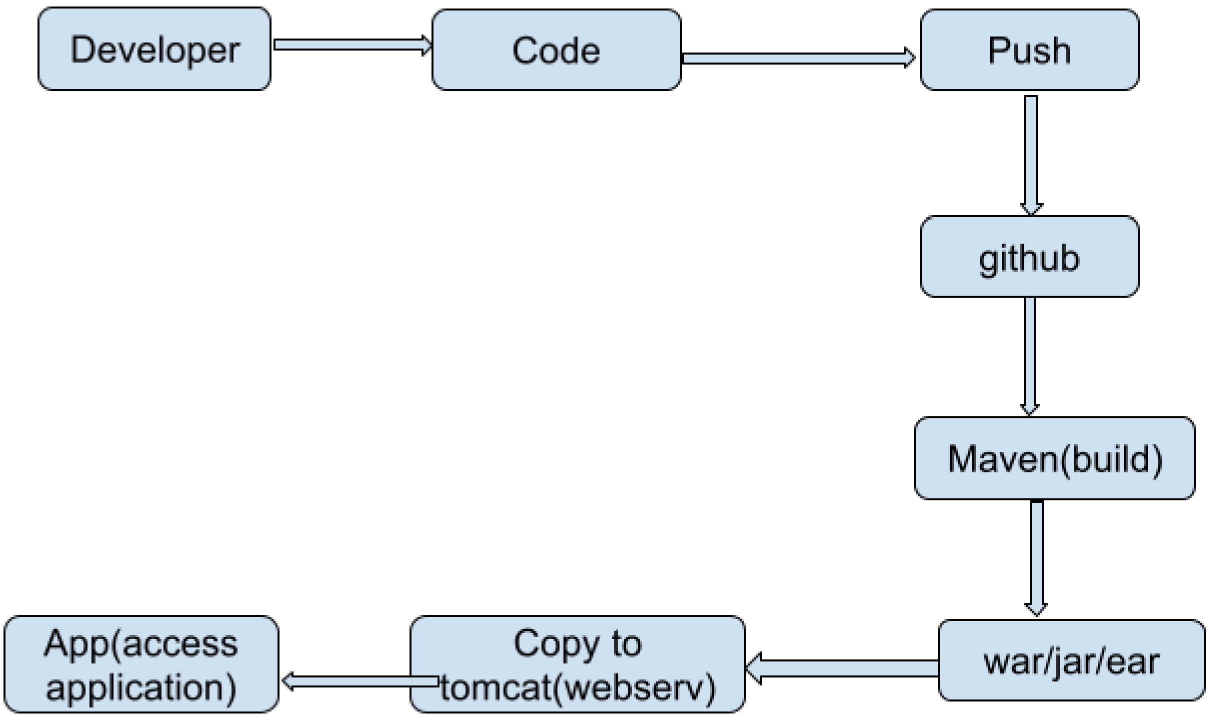
Today’s Documentation

27-03-2025
Suneetha.V

1. **Workflow of maven in diagrammatic representation - - - - - - - - - - - -**



Or

Developers - - - -> code - - - -> Push - - - -> github(remote repository) - - - -> Build(Maven) - - - -> packages - - - -> ear/war/jar - - - - > copy the packages on to the web server(Tomcat) - - - -> App (Access the application)

Define ear/war/jar files:- - - - - - - - - - - - - - - - - - - - - - - - - - - - -

ear - - - - -> enterprise application archive

ear: A larger container for enterprise-level applications, which can include multiple modules, such as war files, EJB (Enterprise

JavaBeans), and others.

war - - - – -> web application

war: Contains web application files (like HTML, CSS, and Java code for web apps).

jar - - – - - ->java application

jar: A standard format for packaging Java code into a single compressed file.

2. Process Of Maven: - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Stages:

Default: It fetches the source code from developers and performs few functions.

1. Compile: Compiles the entire source code
2. Validate: Validates the compiles code
3. Test: Tests the source code
4. Package: Generates the package for Source code
5. Instal: Install all the packages generated by Package
6. Verify: It will verify the generated Package

Clean: Performed before compilation

1.Pre-clean: Check for jar/war/ear

2.Clean: delete the older ear/jar/war files

3. Post-Clean:the new generated war/ear/jar files will be saved

Site: It's like a folder where we will deploy our applications

1.Pre-Site: It receives the post clean files

2.Site:it receives the pre-site files

3.Post-site: it receives the files from site

4. Site - Deploy: To where (address Of Server) we need to copy the files.

It generates comprehensive project documentation test reports and metric(exact values)

For better understanding and collaboration between team members

3. Steps: In maven -----------------------------------

1. Validate

- **Purpose**: Validates that the project is correctly configured.
- **Step**: Checks if the project's configuration (in `pom.xml`) is valid and if all required files are present.
- **Example** : Validate dependencies and plugins.

2. Compile

- **Purpose**: Compiles the source code of the project.
- **Step**: Takes all Java source files and compiles them into `.class` files (bytecode).
- **Example** : Compile all the source code in the `src/main/java` directory.

3. Test

- **Purpose**: Runs unit tests to verify the correctness of the code.
- **Step**: Runs the unit tests from `src/test/java` using testing frameworks like JUnit or TestNG.
- **Example** : Execute test cases to check the functionality of your code.

4. Package

- **Purpose**: Packages the compiled code into a distributable format.
- **Step**: Packages the code into an artifact (like a JAR, WAR, or EAR file).
- **Example** : Create a JAR file from the compiled classes.

5. Verify

- **Purpose**: Verifies that the artifact is valid and meets quality standards.
- **Step**: This step runs any checks that you want to apply to the packaged code (e.g., code quality checks, static analysis).
- **Example** : Perform additional validation of the final artifact.

6. Install

- **Purpose**: Installs the artifact into the local repository.
- **Step**: Once the artifact is created, this phase installs it into your local Maven repository so it can be reused by other projects.
- **Example** : Install the artifact to your local repository.

7. Deploy

- **Purpose**: Deploys the artifact to a remote repository.

- **Step**: Deploys the artifact to a remote repository (e.g., Maven Central or your company's Nexus repository) for sharing or release.
- **Example** : Upload the artifact to a remote server or repository.

4. Fields Present in POM.XML file: -

1.group id: Clent name or project; e.g: Axis, ICICI, SBI.

2.Artifact id : feature name; e.g: Personal loans, gold loans, Credit cards, debit cards, internet banking.

3. Version id: e.g: 8.2.0

Consists of '3' fields:

1. Major release: if 8 changes to 9 - - -> project got released
2. Minor release: if 2 changes to 3 - - - - >new feature is added

Hot fix/bug release: developers are still developing the code.

5. Maven Goals or Maven Commands: -

Every Maven commands starts with 'mvn':

1. **Maven compile**
2. **Maven clean**
3. **Maven test**
4. **Maven validate**

6. Define Releases: - - - - - - - - - - — - - - - - - - - - - - - - - - - - - - - -

Releases:It is a final version of that build which will not change

1. Looks like 8.2.0 - - - -> only numerical values are present
2. Releases maintain versions.

In another terms we can it as:

A **Release** version refers to a **final, stable version** of a project that is ready for production use. Once a version is marked as a release, it is **not changed** (unless there's a critical issue that requires a new version). Releases are typically used for production environments.

- Example: `1.0.0`, `2.3.4`, etc.
- Maven stores releases in a specific repository (e.g., `releases` repository).
- Once a version is released, it won't be overwritten, and its artifacts remain unchanged.

7. Define Snapshots: - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Snapshots:It is under development version build compile/artifacts can change:

- - - -> looks like-8.2.0 -SNAPSHOT
- - - - > doesn’t maintain any versions.

In other terms we can say it as:

A **Snapshot** version refers to a **development version** that is still being worked on and may change. Snapshots are typically used during the development cycle, as they represent the "work in progress" of a project. These versions are **mutable** and can be updated frequently with new changes.

- Example: `1.0.0-SNAPSHOT`, `2.3.5-SNAPSHOT`, etc.

- Maven stores snapshots in a separate repository (e.g., `snapshots` repository).
- A snapshot version can change over time, meaning when you download it, you might get a different version each time, depending on updates.

8.- - -> maven Installation steps: - - - - - - - - - - - - - - - - - - - - - - - - - - - -

1. Take 1 Ec2 instance and Login
2. Connect to Linux environment
3. Install java
4. Install git
5. Clone the git url
6. Cd filename(pom.xml)
7. Maven clean package - - - >jar/war/ear - - - ->stored in target directory

1. sudo yum install java-11-amazon-corretto.x86_64
2. sudo alternatives --config java
3. yum install -y git
4. https://github.com/SuneethaVemula1234/myweb
5. java installation link

   sudo yum install java-11-amazon-corretto.x86_64

6. sudo wget

http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo
-O

/etc/yum.repos.d/epel-apache-maven.repo

7.sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

8. sudo yum install -y apache-maven

9. mvn clean package

now am getting build Issue Error

10.go to project Folder (Myweb or new project name) and ls

11."mvn clean"

check target folder is present or not if present delete

12. rm -rf target

13. mvn clean install

14. If i am getting same error

same processer am applying here

check target folder is present or not if present delete

rm -rf /home/ec2-user/myweb/target

15. "mvn clean
(project with target location) rm -rf target or (home or any location) rm -rf

/home/ec2-user/myweb/target"

adding pom.xml some line or anything

cmd line

16. sed -i '/<properties>/a\
<maven.compiler.source>11</maven.compiler.source>\n

<maven.compiler.target>11</maven.compiler.target>' pom.xml

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

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

19. Verify the Changes
cat pom.xml | grep -A 3 '<properties>'
20. Build the Project Again
21. mvn clean install
    our project build
now i got build success msg