11/03/2025

**MAVEN:**

Maven is a build automation and project management tool used for java applications. Maven simplifies the process of building, managing dependencies, and deploying applications

Maven was developed by Apache Company

Maven is used for only java projects

Maven is a open source project

1. **Project management:** Uses a POM.xml file to define a project-dependencies, build configurations and plugins

Project Object Model is an XML file that contains information about the project and configuration details used by maven to build the project

1. **Build automation:** compiles, tests, packages and deploys applications with a single command
2. **Dependency management:** Automatically downloads required libraries and dependencies from remote repositories
3. **Plugins:** Offers a wide range of plugins for compiling, testing and packaging projects
4. **Integration with CI/CD:** Works with Jenkins, GitHub actions
5. **Standard directory layout:** Maven encourages a standard project directory structure which helps in maintaining consistency across projects. This standardization simplifies the build process and makes it easier for developers to understabd and navigate the project

POM:

Maven uses an xml file called POM.xml to describe and manage its configuration. The POM file includes information about the project, its dependencies, build instructions and other details

**What is a Build tool in DevOps?**

It is a software program that automates the process of transforming the source code into a deployable and executable format

As a build automation tool, it automates the source code compilation and dependency management assembles binary codes into packages and executes the test scripts

**Build tools:**

Java: Apache maven

Python: Pybuilder

.NET: MSbuild

Node js: Gulp, Gradle, Grunt

**Workflow of maven:**

Developers 🡪 code 🡪 push🡪GitHub🡪Maven (Build)🡪packages🡪ear/jar/war🡪copy the packages on to the web server (tomcat)🡪EU (access the application)

Ear: Enterprise Application Archive

War: Web Application archive

Jar: Java archive

How does maven works?

Initially we having the POM.xml

POM.xml🡪build🡪war/jar/ear🡪packages(deployable format)🡪copy tomcat(web server)

1. **Maven works for POM.xml/POM.xml2:**

POM stands for project object model

* POM.xml used for managing the java appliacations
* Developers will write the POM.xml
* POM.xml is also known as super POM or Parent POM
* It contains all the dependency libraries
* Dependency libraries are nothing but internal or external features
* POM.xml should be unique
* Each project contains one POM.xml file

1. **Plugins:**

* Installing the dependencies (external features) is known as plugins
* Dependencies🡪external features adding to our project
* We have 2 types of plugins 1. In built plugins (one time use) 2. Added plugins(reusable)
* Maven is extensible through plugins, which add functionality to the build process.
* There are plugins for compiling code, running tests, generating the documents, creating project reports

1. **Repositories:**

* **Maven** uses repositories to manage project dependencies and plugins
* Central repository: Online Repository GitHub

The central repository is the default location from which maven downloads dependencies

* Remote repository: Organization Repository
* Local repository: Local machine/ own laptops/ systems

**Maven in DevOps should be utilised in 3 Scenarios**

* If the initiative has number of significant dependencies
* If the dependencies versions needs to be upgraded frequently
* The tasks involves rapid documentation, compilation and building of source code as jar or zip files

**ANT:**

* It is an older version of maven
* Developed by Apache Company
* ANT can build any kind of projects
* Has no life cycle
* Developers write the build.xml
* Junit (java unit) test cases not there in ANT
* Scripts are not reusable

3 steps process of a maven:

1. Default: It takes the code from developers and performs few functions

* Compile: compiles the entire source code
* Validate: validates the compiled code
* Test: tests the source code
* Package: generates the package for source code
* Install: Install all the packages generates by packages
* Verify: It will verify the generated package

1. **Clean:**

Performed before compilation

* Pre clean: clean for jar/war/ear files
* Clean: delete the folder jar/war/ear files
* Post clean: the new generated jar/war/ear files will be saved

1. Site:

Its like a folder where we will deploy our applications

* Pre site: It receives the post clean files
* Site: It receives the pre site files
* Post site: it receives the files from the site
* Site deploy: to where (address of the server) we need to copy the files

**Life cycle of Maven:**

**Phases of maven life cycle:**

**Validate:** confirm that the project information is correct and available

**Compile:** compile the source code

**Test:** Test the code using a testing framework

**Package:** package the code into a jar file

**Verify:** confirm that quality assurance criteria are met

**Install:** Install packages into a local repository

**Compile/Build:**

Compilation is the process of converting source code written in a high level programming language into machine readable code(machine readable format) using a compiler

Javac is a compiler to compile the java code

**Why we use Maven and why javac cant support couple of major features:**

* Dependency management
* Build life cycle management

No standard lifecycle: javac only combines java files

It doesn’t provide a standardized build life cycle (compile, test, package, deploy)

* **Task automation:**

Tasks like running tests, creating jar/war files and deploying to servers are not automated and must be handled separately

* Integration with other tools(CI/CD)

Difference between ANT and MAVEN

ANT:

* ANT has no life cycle
* Developers write the build.xml file
* No built in support
* No standard directory structure
* Limited plugin support
* Manual configurations required
* ANT requires custom scripting

Maven:

* Maven has life cycle
* Developers write the pom.xml file
* It has dependency management
* It has standard structure layout
* It supports the plugins
* Standardized and repeatable builds
* Easily integrates with Jenkins, GitHub actions and AWS code pipeline

**12/03/2025**

**Install maven in AWS Linux instances**

Step1: Open AWS console and create instance

Step2: Connect the server with Gitbash

Step3: install Java, run the below command

**sudo yum install java-17-amazon-corretto -y** # Install Java 17 (latest)

**Set JAVA\_HOME:**

export JAVA\_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86\_64

export PATH=$JAVA\_HOME/bin:$PATH

**Make it permanent:**

echo "export JAVA\_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86\_64" | sudo tee -a /etc/profile.d/java.sh

echo "export PATH=\$JAVA\_HOME/bin:\$PATH" | sudo tee -a /etc/profile.d/java.sh

source /etc/profile.d/java.sh

**step4: install maven**

cd /opt

sudo wget <https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.tar.gz>

sudo tar -xvzf apache-maven-3.9.9-bin.tar.gz

sudo ln -sfn /opt/apache-maven-3.9.9 /opt/maven

sudo nano /etc/profile.d/maven.sh

export M2\_HOME=/opt/maven

export PATH=$M2\_HOME/bin:$PATH

source /etc/profile.d/maven.sh

mvn -version

**To remove the maven**

sudo rm -rf /opt/apache-maven-\* /opt/maven

**step6: Install git**

sudo yum install git

step7: git clone url

step8: cd directory name

step9: mvn clean package