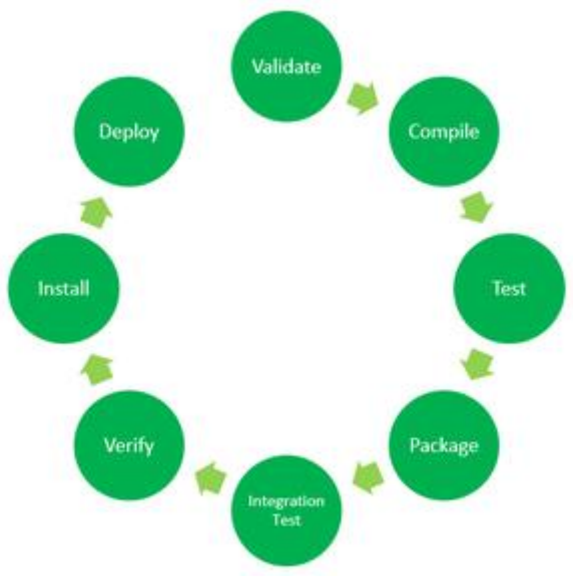
Maven Lifecycle

Maven is a powerful project management tool that is based on POM (project object model), used for projects build, dependency and documentation. It is a tool that can be used for building and managing any Java-based project. Maven makes the day-to-day work of Java developers easier and helps with the building and running of any Java-based project.



The default Maven lifecycle consists of 8 major steps or phases for compiling, testing, building and installing a given Java project as specified below:

1. **Validate:** This step validates if the project structure is correct. For example – It checks if all the dependencies have been downloaded and are available in the local repository.
2. **Compile:** It compiles the source code, converts the .java files to .class and stores the classes in target/classes folder.
3. **Test:** It runs unit tests for the project.
4. **Package:** This step packages the compiled code in distributable format like JAR or WAR.
5. **Integration test:** It runs the integration tests for the project.
6. **Verify:** This step runs checks to verify that the project is valid and meets the quality standards.
7. **Install:** This step installs the packaged code to the local Maven repository.
8. **Deploy:** It copies the packaged code to the remote repository for sharing it with other developers.

What is a Maven Repository?

In Maven terminology, a repository is a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

Maven repository are of three types. The following illustration will give an idea regarding these three types.

- local
- central
- remote

Local Repository

Maven local repository is a folder location on your machine. It gets created when you run any maven command for the first time.

Central Repository

Maven central repository is repository provided by Maven community. It contains a large number of commonly used libraries.

Remote Repository

Sometimes, Maven does not find a mentioned dependency in central repository as well. It then stops the build process and output error message to console. To prevent such situation, Maven provides concept of **Remote Repository**, which is developer's own custom repository containing required libraries or other project jars.

Maven Dependency Search Sequence

When we execute Maven build commands, Maven starts looking for dependency libraries in the following sequence −

- **Step 1** − Search dependency in local repository, if not found, move to step 2 else perform the further processing.
- **Step 2** − Search dependency in central repository, if not found and remote repository/repositories is/are mentioned then move to step 4. Else it is downloaded to local repository for future reference.
- **Step 3** − If a remote repository has not been mentioned, Maven simply stops the processing and throws error (Unable to find dependency).
- **Step 4** − Search dependency in remote repository or repositories, if found then it is downloaded to local repository for future reference. Otherwise, Maven stops processing and throws error (Unable to find dependency).