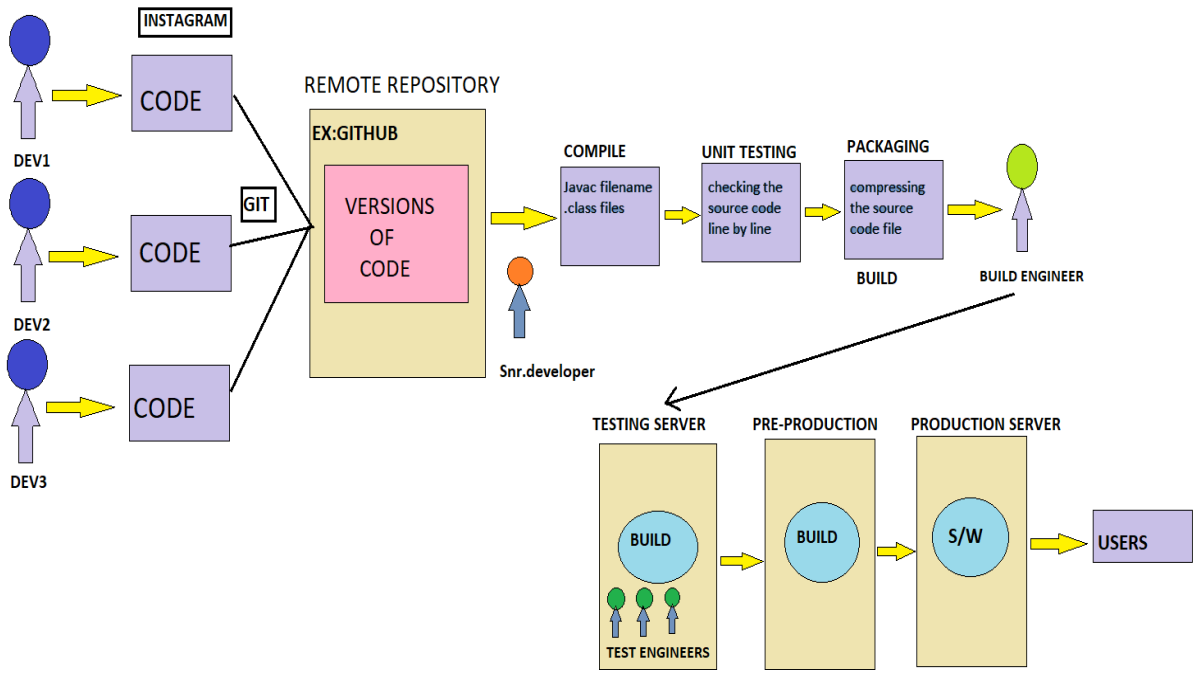
# MAVEN

**BUILD:** Converting the source code files into executable code, by compiling, testing, and packaging is called "Build"

## TRADITIONAL MODEL OF CREATING BUILD




## Drawbacks:

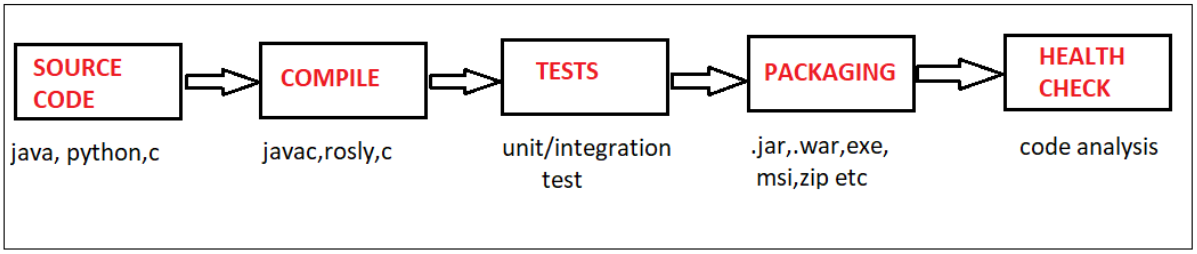
1.Time consuming.

2.Delay in deliverying the software.

3.More manual work.

4.Difficult to rectify errors.

# Build Process:




**Source code:** code is written by developers like java, c, python etc.

**Compile:** check the syntax of source code written by developers like java generate .class file.

**Unit/Integration test:** unit test are the test written by the developers to test their own code this is not software testing.

Unit of code is tested here, if it passed then it is packaged.

**Packaging:** based on the target/Requirement source code will be packaged.

Ex: .war, .jar, exe, msi, zip etc.

**Health check:** it will be done by developers, this process is going to find any upcoming bugs/security vulnerable in code.

Developers along with writing the code, regularly committing the code and also need to regularly do this build process. Which is too much work to do it regularly.

To automate this build process there are build tools.

# Build tools:

1.Maven 3. Gradle

2.Ant 4. MS Build

## Maven:

It is build automation tool.
It is mostly used for java projects.

## Ant:

It is a build automation tool.
It was invented in 2000 by James.
It is used for programming languages like java, C, C++.

## Gradle:

It is a build automation tool.
It was invented in 2008 by Hans dockter.
It is used for java, C, C++, and python projects.

## What is Maven?

Maven is a popular open-source build tool developed by the Apache group to build, publish and deploy several projects at once.

It was invented in the year 13 july 2004 by Jason van zyl.

Maven is written in java and is used to build projects written in C#, Scala, Ruby etc.

This tool is used to build and manage any java-based project. It simplifies the day to day work of java developers and helps them in their projects.

## Why do we need Maven?

Maven is mostly used for java projects. The tool helps in building the code downloading dependencies, the tool is used to build and manage any java-based projects.

It simplifies the day to day work of java developers and helps them in their projects.

To download dependencies it is no more needed to visit the official website of each software. It could now be easily done by visiting "mvnrepository".

## Features of Maven

1.Enhanced dependency support.

2.Follows build lifecycle.

3.Enhanced Plugin support.

4.Makes build process easy

5.Easy to install and update.

## Difference between Maven and Ant

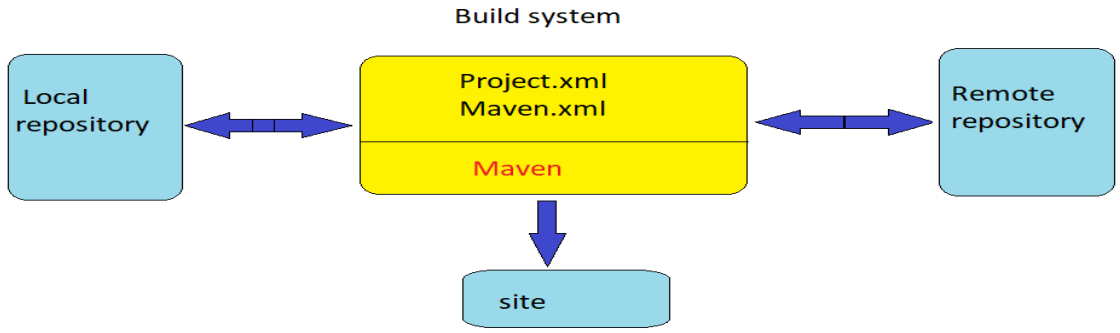
| MAVEN | ANT |
|---|---|
| 1.Maven is project management tool. | 1.Ant is mainly a build tool. |
| 2. Maven scripts are re-usable. | 2.Ant scripts are not re-usable. |
| 3.Maven follows build lifecycle. | 3.It does not have build lifecycle. |
| 4.It is a declarative tool. Everything need to define in pom.xml file. | 4.It is procedural tool. We need to provide information of order of execution. |
| 5.It is more preferred than Ant. | 5. It is less preferred than Maven. |

## Difference between Maven and Gradle

| MAVEN | GRADLE |
|---|---|
| 1.It is project management tool that is primarily used for java projects. | 1.It is build automation tool that uses a groovy based language. |
| 2.It uses XML file for declaring the project dependencies. | 2.It does not use XML file for project configuration. |
| 3.It will not track the tasks so it will not use build cache. | 3.It avoids the work by tracking the tasks and only runs the updated tasks. |
| 4.Performance is slow compared to Gradle. | 4.Performance is high and faster. |
| 5.Maven is dependent on java platform. | 5.Gradle is not dependent on java platform. |
| 6.Compilation is mandatory in maven. | 6.Compilation is not mandatory. |



## Maven Architecture:



*There are various components of maven

*Local repository or the local machine that we work on.

*Remote repository or remote web server.

*When we specify any dependency in the pom.xml the maven will look for that file in the central repository, if the dependency present in the central repository maven will copy that dependency onto our local machine. But, if it is not present maven will fetch it from the remote web server or remote repository using internet. So, internet is very much mandatory for using maven.

*Site is used to create project site documentation.

## Pom.xml:

*The core component of any maven project.

*Maven project consists of one configurable file called pom.xml Which stands for Project Object Model.

*This pom.xml always be located in the root directory of any Maven project.

*pom.xml contains all the necessary information about the configuration details, dependencies included and plugins included in the project.

### Why we need Pom.xml?

*To create a build in structured process.

*To add the dependencies to the build related to the project.

*To make the build process easy and automate.

## Maven Repositories:

Maven repository is a place where are going to store the information and the dependencies of our project

### 3 types of Maven repositories:

1. Local repository
2. Central repository
3. Remote repository

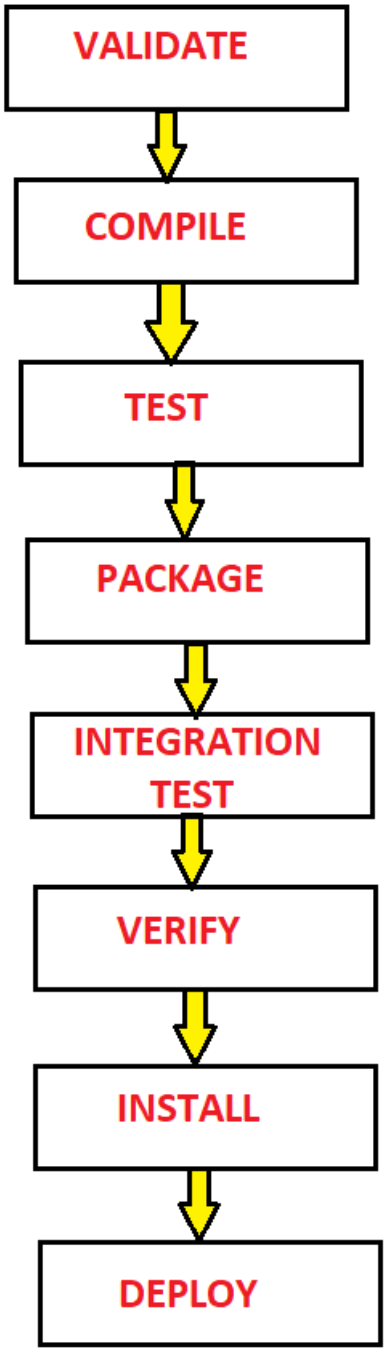
1. **Local repository:** It is a repository which is present locally in our system. In this repository we are going to store the dependencies and the documentation of a particular project.

2. **Central repository:** Central repository is managed by Apache. It is located in web, we need internet to take the dependencies from central repository.

3. **Remote repository:** Remote repository is present in the web, if we need any dependencies to our project we can get it from remote repository of maven. It is hosted by Apache.

## Maven Phases:

The maven lifecycle consists of 8 major phases, which defines the order of execution for goals.




**Validate:** Validate the project is correct and all necessary information is available.

Command: mvn validate

**Compile:** compile the source code of the project.

Command: mvn compile

**Test:** Test the compiled source code using suitable unit testing framework. These tests should not require the code be packaged or deployed.
Command: mvn test

**Package:** Take the compiled code and package it in its distributed format such as .jar, .war, exe, msi, zip etc.
Command: mvn package

**Integration test:** This testing is done by using suitable testing framework.

**Verify:** Run any checks on results of integration tests to ensure quality criteria are met.

**Install:** Install the package into the local repository for use as a dependency in other projects locally.
Command: mvn install

**Deploy:** Done in the build environment, copies the final package to the remote repository for deploying it in the production.
Command: mvn deploy
**mvn clean:** To delete the target directory

## Maven Commands:

1.mvn validate

2.mvn compile

3.mvn test

4.mvn package

5.mvn clean

6.mvn install

7.mvn deploy

## Pre-requisites for Maven

Install the java in the system

Website: https://www.oracle.com

Install Maven

Website: https://maven.apache.org