maven

General:

Waterfall Model:

- In this model first they study the requirement, analyse the requirement, develop the full code, test the code and finally deploy the code.
- SDLC minimum 6 months.
- Lot of time waste.
- We can't find issues/bugs at early stage.

Agile Model:

- In this Method incremental development of the code and testing will be done,once all are success it will be deployed.
- In this method IOC(Infrastructure as Code) and Configuration Management[CM] are not applicable[Means not automated]
- SDLC is minimum 2-3 months
- Compared to Waterfall time should be save

DevOps Method: DevOps is the combination of cultural philosophies, best practices, and tools that increases an organization's ability to deliver applications and services at high velocity.

- In this method Continuous Integration and Continuous deployment/delivery should be done.
- In this method IOC(Infrastructure as Code) and Configuration Management[CM] are fully automated.
- SDLC IS 1 week.
- We can find issues/bugs at early stage.
- No waste of time.

Continuous Integration(CI): Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently(like Git), preferably several times a day. Each integration can then be verified by an automated build and automated tests. In this following steps are should be done.

- Validate
- Compile
- Test cases(Junit/Integration)
- Code Analysis(SonarQube)
- Creating Package(Jar/war/ear/Docker Image)

Use/Advantage of CI:

- We can fix issues/bugs as fast as possible at early stage
- Time Saving

Continuous Deployment:

- It is next level of continuous Integration
- Continuous Deployment is a software development practice in which every code change goes through the entire pipeline and is put into production, automatically, resulting in many production deployments every day.

Difference between Continuous deployment and Continuous delivery:

- In Continuous deployment there no manual approve between UAT and PROD Env.
- In Continuous delivery there manual approve between UAT and PROD Env.

Apache Maven(From Apache):

- Maven is a powerful build & dependency management tool for Java software projects.
- Maven is primarily known for build and dependency management however it can do more than that.

What Maven do as a build tool?

- Organise project specific files
- Validate the Code
- Compile the Code
- Test cases(Junit/Integration)
- Creating Package(Jar/war/ear/Docker Image)

What is Junit Test Cases?

- Junit is a framework for automating unit testing, whenever we add new features to the software we need to retested with all functionalities
- Developers write Junit test cases.

What is integration testing?

- Integration test cases are written by QA team
- Integrations testing is testing end to end flow by using modules..
- QA teams use tools like selenium, QAT for automating integration testing.

JAR vs WAR vs EAR:

- JAR(Java Archive) is group of dart class files.
- WAR(Web Archive) is combination of JSP, Servlets, HTML, CSS and XML files(For Web Applications).
- EAR(Enterprise Archive)is combination of Servlets, JSPs and JMS Components etc...

Dependency:

- We our project wants to use a framework, frameworks come as a jar file, this jar is our project dependency.

Transitive Dependency:

- A dependency on which our dependency depends on 1.jar is our dependency and 2.jar is our transitive dependency
- Maven automatically manages dependencies and transitive dependencies by downloading them from maven repositories.

Types of 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 Repo
- Central Repo
- Remote Repo

Central Repository(Universal):

Maven central repository is repository provided by Maven community. It contains a large number of commonly used libraries, Here any one can download.

When Maven does not find any dependency in local repository, it starts searching in central repository.

Remote Repository(Organizations specific):

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.

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.

Maven local repository keeps your project's all dependencies (library jars, plugin jars etc.). When you run a Maven build, then Maven automatically downloads all the dependency jars into the local repository. It helps to avoid references to dependencies stored on remote machine every time a project is build.

maven

Path of local repository:

- **{usr_home}/.m2/repository**
- Change path of Local repository in settings.xml
- Settings.xml location should be **{MAVEN_HOME} /conf/settings.xml**

Maven build lifecycle:

- Validate
- Compile
- Test cases(Junit test cases written by the development team)
- Verify (Run integration test cases written by the QA team(Testing Team))
- Package (Jar/war/Ear)(artefacts)/Docker Images
- Install (stores the package into the local repo)
- Deploy(Upload the artefacts to remote repo)(It may be Sonatype Nexus, Docker hub or any private repo)

Installation of Maven:

Maven depends on java, make sure **java(JDK)** is installed.

Installing java:

sudo yum install java-1.8.0-openjdk-devel -y

If u Want update alternatives java:

sudo update-alternatives --config java→ select option beside 1.8.0

If u Want update alternatives javac:

sudo update-alternatives --config javac→ select option beside 1.8.0

JDK vs JRE vs JVM:

- To develop and run java applications required environment is JDK(Java Development Kit).
- To run java applications required environment is JRE(Java Runtime Environment).
- In JRE, JVM is responsible to run the application(JVM is interpreter).

JDK = JRE + Development Tools

JRE = JVM + Library Classes

Installation of Maven on Linux:

1. *sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo*
2. *sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo*
3. *sudo yum install -y apache-maven*
4. *mvn --version*

maven

Maven Commands:

- *mvn package*
- *mvn install*
- *mvn deploy*
- *mvn clean*
- *mvn clean package*

mvn package -DskipTests → If you run this command without executing test cases maven can build the package**..**