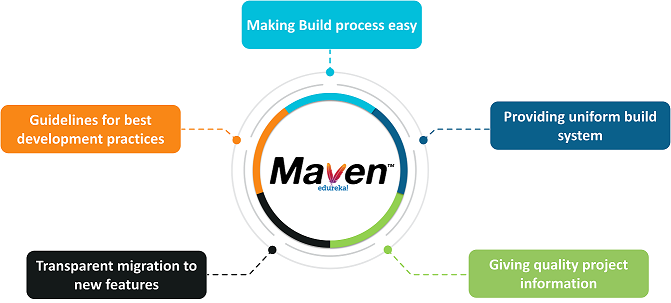
**What is Maven**

* Maven is a powerful build automation tool that is primarily used for Java-based projects. Maven helps you tackle two critical aspects of building software –
  + It describes how software is built
  + It describes the dependencies.
* Maven prefers convention over configuration. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven Central Repository and stores them in a local cache.
* The artifacts of the local projects can also be updated with this local cache. Maven can also help you build and manage projects written in C#, Ruby, Scala, and other languages.
* Project Object Model(POM) file is an XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals, etc. used by Maven to build the project.
* When you execute a maven command you give maven a POM file to execute the commands. Maven reads the pom.xml file to accomplish its configuration and operations.

**Maven Objectives**



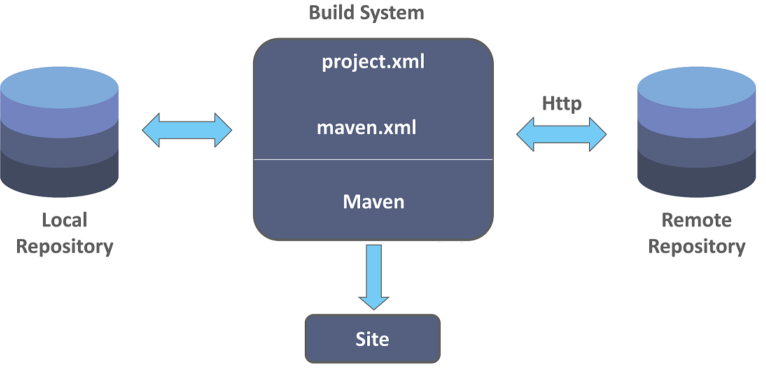
**Why do we need Maven**

* If you are working on Java projects then most of the time you need dependencies.
* You need to download and add them manually. Also, the task of upgrading the software stack for your project was done manually before Maven.
* Repeatable Builds.
  + We can recreate our build for any environment
* Transitive dependencies
  + Downloading a dependency with also pull other items it needs
* Works with your IDE, but also standalone
* The preferred choice for working with build tools like Jenkins or Cruise Control

**When should someone use Maven**

* If there are too many dependencies for the project.
* When the dependency version update frequently.
* Continuous builds, integration, and testing can be easily handled by using maven.
* When one needs an easy way to generate documentation from the source code, compiling the source code, packaging compiled code into JAR files or ZIP files.

**Maven Architecture**



# **Maven Installation on Windows:**

* Open Windows Powershell 🡪 Run as Administrator
  + **choco install maven**
  + Sometimes Choco automatically sets the environment variable if not
    - Set environment variable
      * MAVEN\_HOME=<Maven-Install-Directory>
  + To see the version of Maven installed, execute
    - mvn –version (JAVA\_HOME Variable should be available)
  + if Java is not installed already then follow the below steps
    - **choco install jdk8**
    - Set environment variable
      * JAVA\_HOME=<Java-Install-Directory>
  + If mvn –version did not give you the Maven installation details then add Maven installation directory (upto bin folder) to path environment variable
    - Ex: C:\ProgramData\chocolatey\lib\maven\apache-maven-3.6.3\bin

**OR (To Install Maven without Choco)**

* Download the Maven package from
  + [Maven – Download Apache Maven](http://maven.apache.org/download.cgi)
* Then follow the instructions
  + [Maven – Installing Apache Maven](http://maven.apache.org/install.html)

# **Maven Installation on MacOS:**

* **To Install Java 8**
  + brew tap adoptopenjdk/openjdk
  + brew cask install adoptopenjdk8
* **To Install Maven**
  + brew install maven
  + mvn -v 🡪 To check the Version

# **POM.xml**

**(File gets created when developer creating the Maven Project and provides below information)**

**Group Id** – Id of your Organization (Org.apache, com.google etc)

**artifactId** – name of your project (Project folder name)

**modelVersion** – Pom version (POM Schema version

**packaging** jar/war

**version** 1.0-SNAPSHOT (Until you have stable version)

* We can also import the existing Maven Project which already will have POM file

# **Maven Commands**

* mvn clean --- Cleans the Workspace
* mvn compile --- mvn clean + Compiles the source code
* mvn test -- mvn compile + executes the automation test cases
* mvn package -- mvn test + creates the executable package
* mvn install – mvn package + will copy the package into local repository
* mvn deploy – mvn install + copies the package into remote location as well

# **Build the Sample Project with Maven**

* **Get the Spring-petclinic java project** 
  + **Github URL:** [**https://github.com/spring-projects/spring-petclinic.git**](https://github.com/spring-projects/spring-petclinic.git)
* **Create folder in your Local system**
* **git clone** [**https://github.com/spring-projects/spring-petclinic.git**](https://github.com/spring-projects/spring-petclinic.git)
* **cd spring-petclinic**
* **mvn package**
* **Go To target folder with in your project**
* **Open git bash or command line**
* **Java -jar <package-name>.jar**
* **The application should start on port no 8080**
* **Open the browser and type** [**http://localhost:8080**](http://localhost:8080)
* **Then you should be able to see your application running**