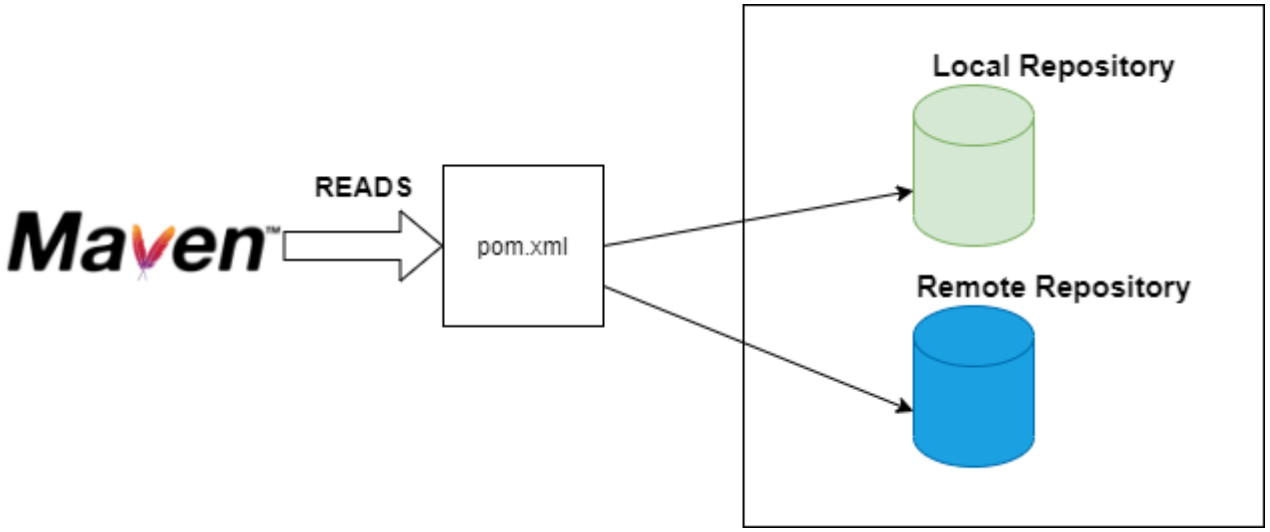
Welcome to Maven

What is Maven

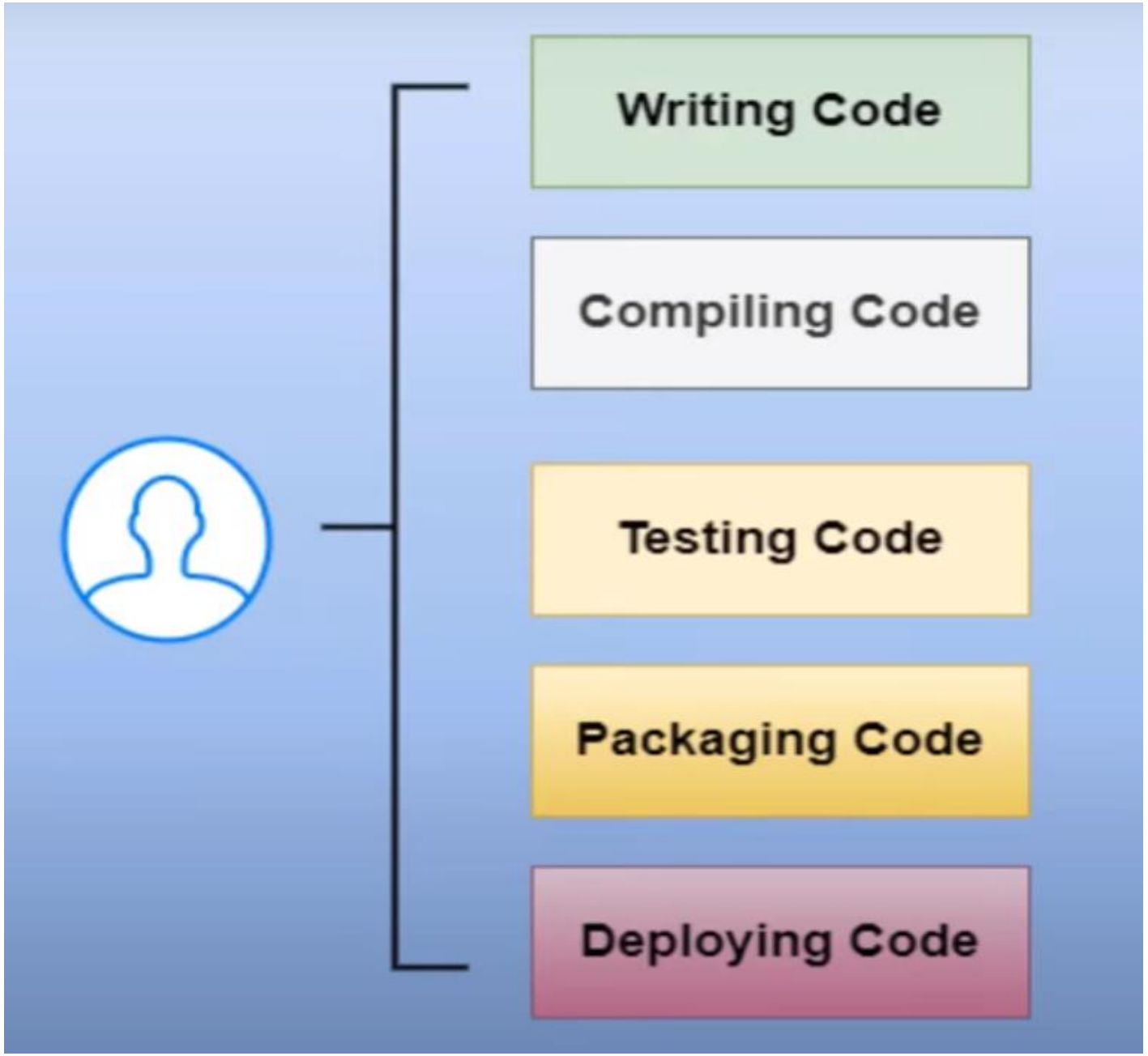
- Maven is a powerful build automation tool that is primarily used for Java-based projects. Maven helps you tackle to perform the below major tasks
  - Building Source Code
  - Testing
  - Packaging into JAR, WAR or EAR
  - Manage Dependencies
  - Also called as Build and Dependency Management Tool
- 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.
- Maven can also help you build and manage projects written in C#, Ruby, Scala, and other languages.
- Maven uses a Project Object Model (POM) file to manage the build process and its dependencies. The POM file contains information about the project, such as the project's dependencies, build settings, and plugins. Maven can be used from the command line or integrated into integrated development environments (IDEs) such as Eclipse and IntelliJ IDEA.

What is Maven

- 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.
- Local Repository
  - Folder inside the machine running Maven
  - <User-Home>/.m2
- Remote Repository
  - Remote Website where we can download dependnecies
  - Ex: Maven Website, Artifactory etc.



Roles of Developer



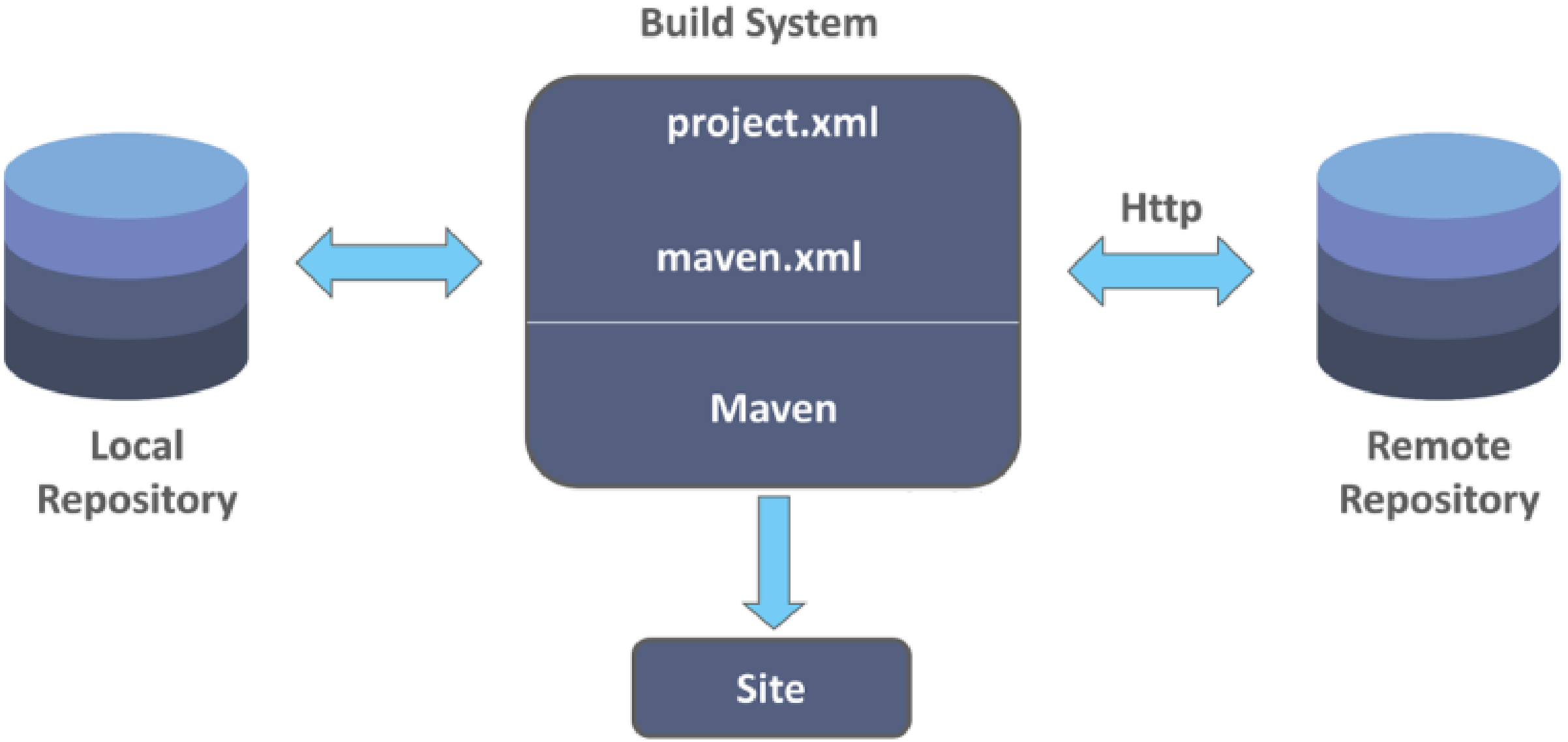
Why do we need Maven

- If you are working on 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 compiling the source code, packaging compiled code into JAR files or ZIP files.

Maven Architecture



Maven Life Cycle

Clean Lifecyle
pre-clean
clean
post-clean

Default Lifecyle	
validate	test-compile
initialize	process-test-classes
generate-sources	test
process-sources	prepare-package
generate-resources	**package**
process-resources	pre-integration-test
compile	integration-test
process-classes	post-integration-test
generate-test-sources	verify
process-test-sources	**install**
generate-test-resources	**deploy**
processs-test-resources	

Maven Lifecycle

- Each Lifecycle is independent of each other and they can be executed together.
- The **default** lifecycle is the main life cycle of Maven as it is responsible for project deployment and is divided into different phases like below:
  - **compile** – compiles the source code inside the project
  - **test** – runs unit-tests inside the project
  - **package** – packages the source code into an artifact (ZIP, JAR, WAR or EAR)
  - **integration-test**– executes tests marked as Integration Tests
  - **verify** – checks whether the created package is valid or not.
  - **install** – installs the created package into our Local Repository
  - **deploy** – deploys the created package to the Remote Repository
- The **clean** lifecycle is mainly responsible to clean the .class and meta-data generated by the above build phases.

- A Maven phase is nothing but a stage in the Maven build life cycle. Each phase executes a specific task.

Maven Installation

- Java is a pre-requisite to install Maven
- If Java is not already installed, please download and install from https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html

- Download the Maven from
  Maven – Download Apache Maven
- Once Downloaded, Extract it and set
  - JAVA_HOME
  - MAVEN_HOME
  - Path

- mvn –version
  - This confirms if Maven installed successfully and path is set
- java –version – to verify if Java is installed

Maven Installation Usning Chocolatey

- Java is a pre-requisite to install Maven
- If Java is not already installed,

choco install openjdk17

Choco install maven

- mvn –version
  - This confirms if Maven installed successfully and path is set
- java –version – to verify if Java is installed

Maven Project

- Get the Spring-petclinic java project
  - Github URL: https://github.com/spring-projects/spring-petclinic.git
- Create folder in your Local system
- git clone https://github.com/spring-projects/spring-petclinic.git
- cd spring-petclinic
- mvn package
- Once the Build is success
  - Cd target
  - Java –jar springpetclinic*.jar
- Once the application has started then access from browser with below URL
- http://localhost:8080

Maven Project

- mvn clean
- mvn compile
- mvn test
- mvn package
- mvn deploy
- cd spring-petclinic
- mvn package
- http://localhost:8080 OR
- Import the Existing Maven Project into Eclipse IDE
- Build Package (Run As → Maven Install)
- Java –jar <jarfilename.jar>

Q & A