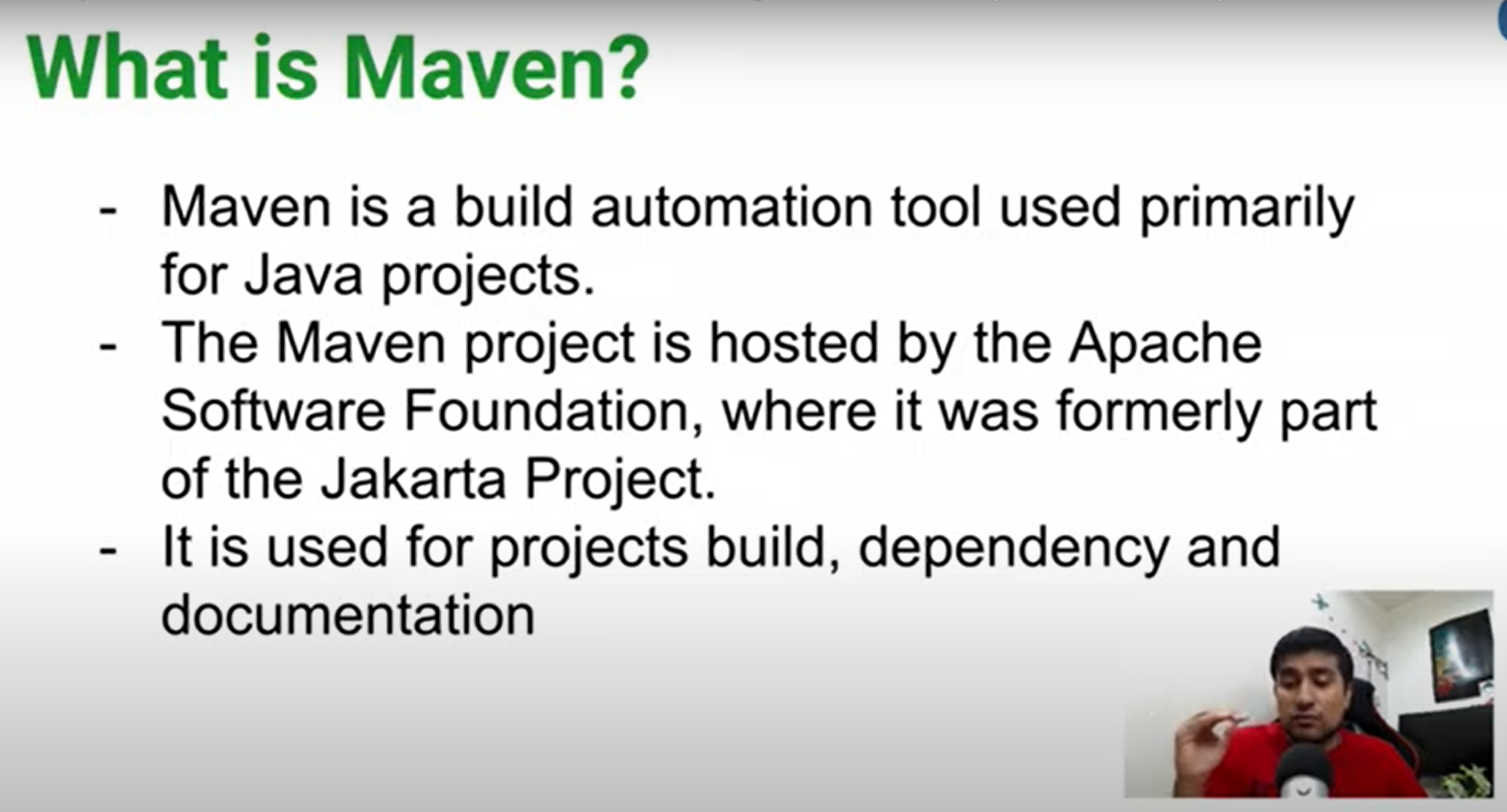
Maven:

It is a build tool.

It has a lifecycle.



MVN install places the jar at m2 repo so in case it don’t need to install it again from internet and it acts as a cache for it like search in m2 if there is then fetch it and use it in our project.

C:\Users\talha\_atif\.m2\repository\co

<https://stackoverflow.com/questions/38723833/what-is-the-purpose-of-mvnw-and-mvnw-cmd-files>

**short answer**: to run Maven and Gradle in the terminal without following manual installation processes.

Gradle example:

./gradlew clean build

./gradlew bootRun

Maven example:

./mvnw clean install

./mvnw spring-boot:run

"*The****recommended way to execute any Gradle build is with the help of the Gradle Wrapper****(in short just “Wrapper”). The Wrapper is a script that invokes a declared version of Gradle, downloading it beforehand if necessary. As a result, developers can get up and running with a Gradle project quickly without having to follow manual installation processes****saving your company time and money****.*"

<https://www.baeldung.com/maven-wrapper>

Note: we use the executable *mvnw* in place of *mvn,*which stands now as the Maven command line program.

Now that we know what Maven wrapper is, let’s answer one of the common questions: should *mvnw* files be added to our projects?

The short answer is no. ***mvnw* files are not necessarily part of our projects**. However, including them could be beneficial. For example, it will allow anyone cloning our project to build it without installing Maven.

Yes, that's correct! Maven Wrapper (mvnw) is a script included in your project that ensures Maven is available, even if it's not installed globally on the system. Here's how it works:

**What Maven Wrapper (mvnw) Does**

1. **Bootstraps Maven Installation:**
   * The Maven Wrapper downloads a specific version of Maven from the internet (based on the maven-wrapper.properties file in the wrapper directory).
   * It installs the downloaded Maven version in your local Maven repository under the .m2/wrapper directory.
2. **Ensures Consistency:**
   * It ensures that everyone working on the project uses the same Maven version, avoiding version conflicts.
   * This is particularly useful in CI/CD pipelines or shared team environments.
3. **Uses Project-Specific Maven:**
   * When you run ./mvnw <goal>, the wrapper:
     + Checks if the required Maven version is already downloaded.
     + If not, downloads it and stores it under the .m2/repository (or .m2/wrapper for the wrapper-specific installation).
4. **Wrapper Files in a Project:** A Maven project with a wrapper includes these files:
   * **mvnw.cmd** (for Windows)
   * **mvnw** (for Unix/Linux/Mac)
   * **wrapper/maven-wrapper.jar**: The main wrapper Java library.
   * **wrapper/maven-wrapper.properties**: Specifies the Maven version to download.

**How It Works in Your Project**

1. **First Run:** When you execute ./mvnw clean validate, the wrapper script:
   * Reads the maven-wrapper.properties file to determine the required Maven version.
   * Downloads Maven if it's not already present in the .m2 directory.
   * Executes the specified Maven goal (clean validate in this case) using the downloaded Maven.
2. **Subsequent Runs:** If the required Maven version is already available in .m2, the wrapper script directly uses it without downloading.

**Advantages of Maven Wrapper**

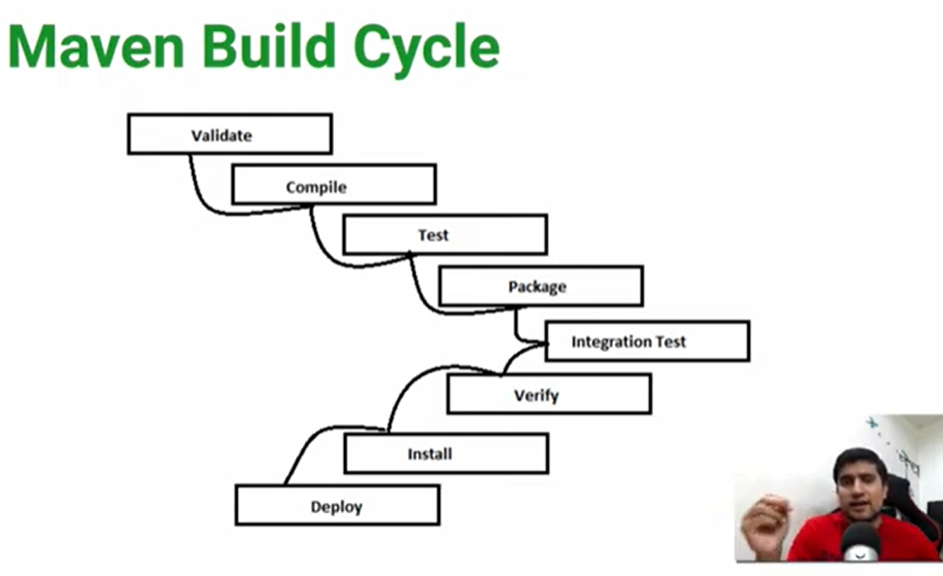
1. **Portable Builds:**
   * No need to manually install Maven on every machine.
   * Ensures a consistent Maven version across all environments.
2. **CI/CD Friendly:**
   * The Maven Wrapper is ideal for CI/CD systems where installing Maven globally might not be feasible.
3. **Version Management:**
   * Projects can specify their own Maven version without affecting other projects or global installations.

If you're using the Maven Wrapper and it isn't working, ensure the mvnw files are in your project root, and the execution policy allows running scripts.

Main -> Core functionality

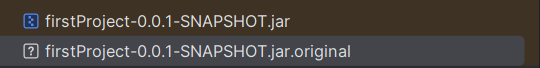
Test -> Test you run on your code

Application.properties for adding configuration related to mongodb or any other modules.



**Jar file:**

The jar file we make for our project will contain also dependencies for our project needed to compile.



Original -> Only my compiled code

Simple Jar -> It is self-contained jar includes other dependencies and also my code and also it has self-contained tom cat server (**FAT JAR**)

Size of original is less than first simple jar.