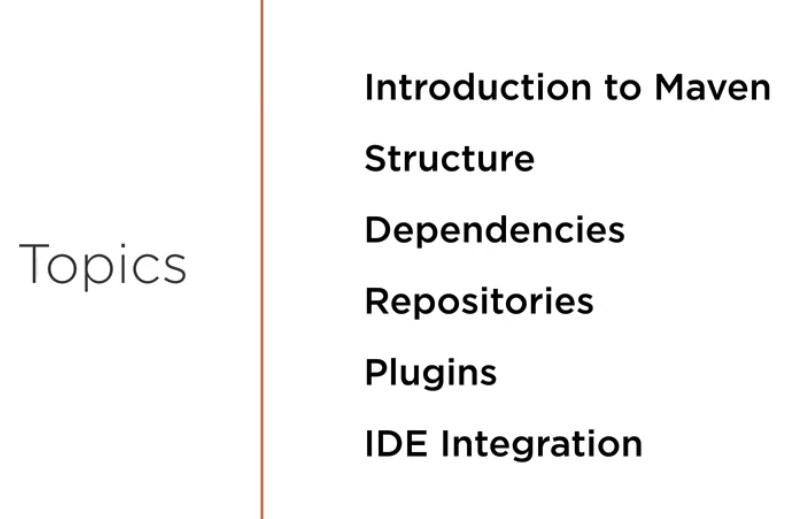
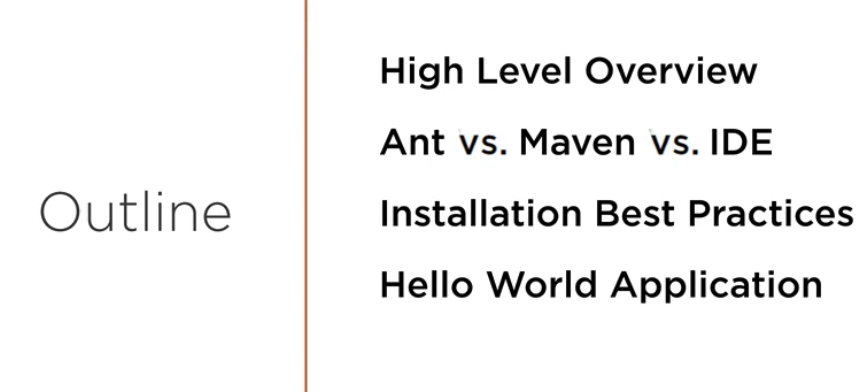
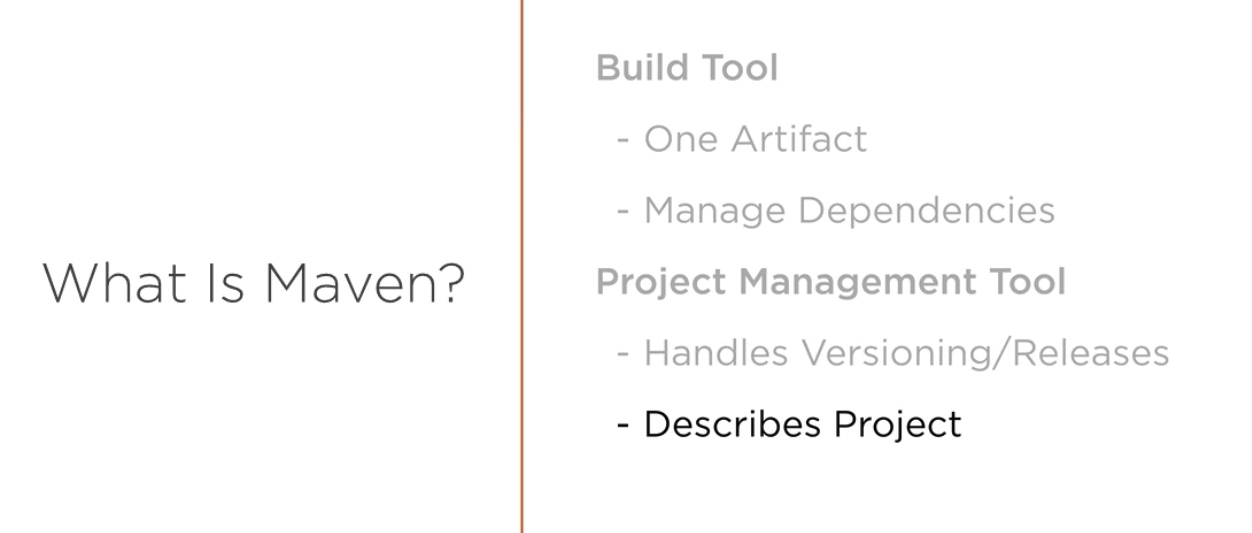
**MAVEN FUNDAMENTALS**

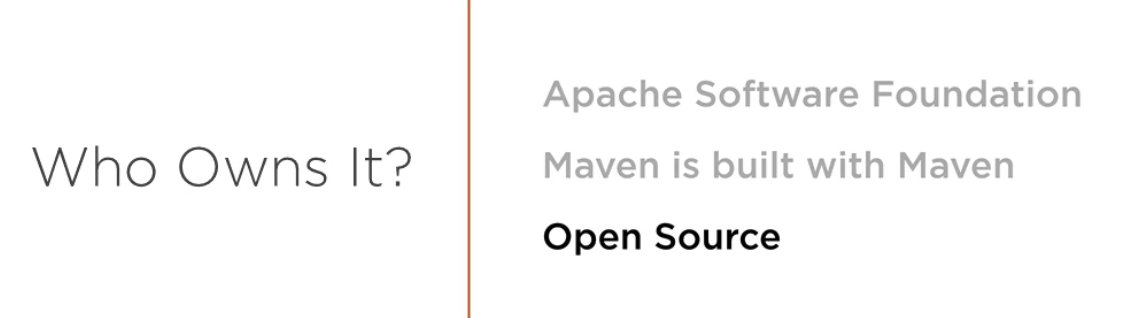




**Maven is a build tool which is used to produce artifact (output of the source code)**

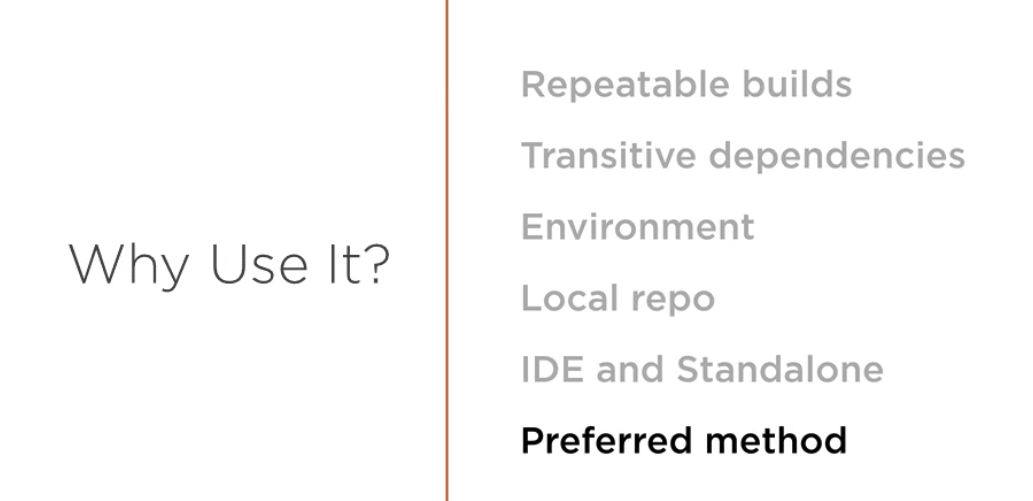


* Maven always produce one artifact (think of as a jar or zip file).
* Dependency management is one of the biggest reasons we use maven. It downloads all the **transitive dependencies** also.
* It can also be used as project management tool. Handles versioning of our code. It has version number associated with each artifact.
* It has some meta info tied to the project. Produce java docs , jar files etc.



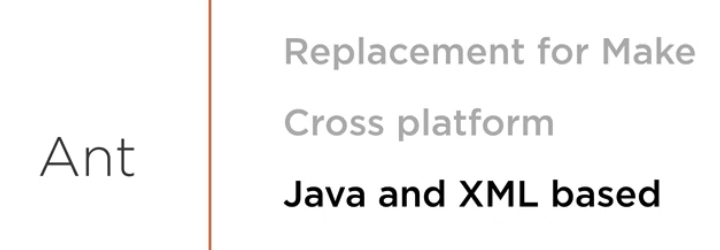
**WHY USE IT?**

* **REPEATABLE BUILD** – Ability to recreate our build for any env and not have to change our settings for each env. Maven lets us externalize our settings so that we can develop on windows, test on linux box and deploy to different flavor of linux or unix box. Docker also helps us to do that.
* **Transitive dependency** – Downloading dependency will pull other item it needs.
* **Local Repo**- Let suppose we need a jar for multiple projects, historically we use to download each jar and kept in the project structure. Maven helps us to download it once and use it as a reference in other projects. Saves disks space and overhead.



**ANT VS MAVEN**

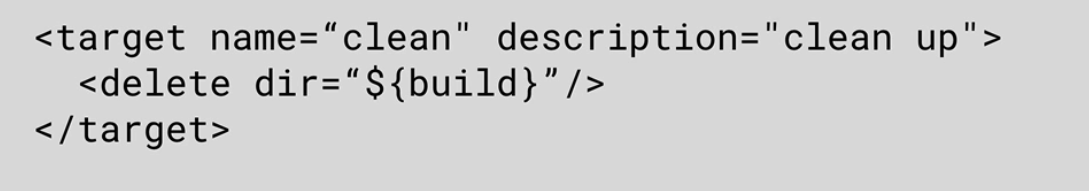
**ANT –**



* Ant was built for the replacement of build tool call **Make** which was not cross-platform. Make was built on unix and was very brittle.
* Ant was built on top of java and xml. Both tools are **cross platform**.

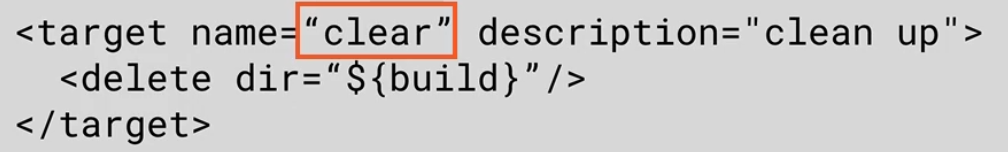
**Problems with ANT:**

* With ant we have to explicitly code everything.



Just to do a clean we need to call our target set our goals.

**This can lead to problems as it is not standard**. What if someone likes the word “clear” instead of clean.

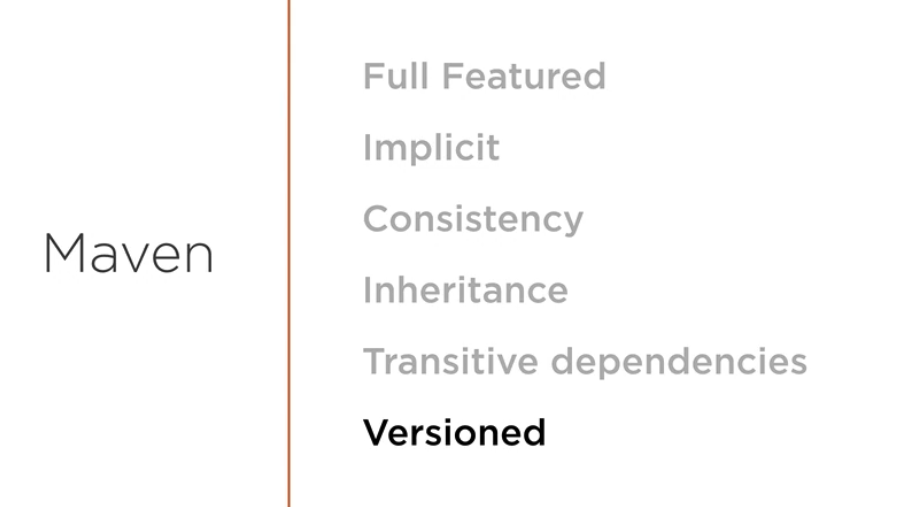


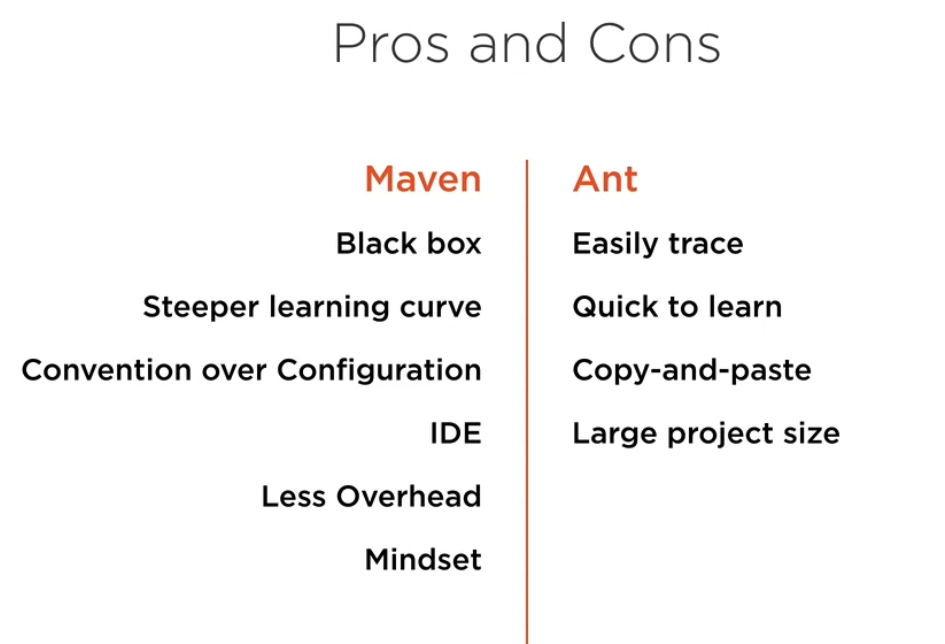
It can lead to n number of variations.

* Each organization our team does it differently and there is no standard. Each org has a large repo of scripts.
* Not much re-use.

**MAVEN**

* It is a full featured build tool more than a scripting tool.
* Maven clean is maven clean, it is standard.
* Consistency.
* Inheritance from parent pom using composition





**Basic ANT script**

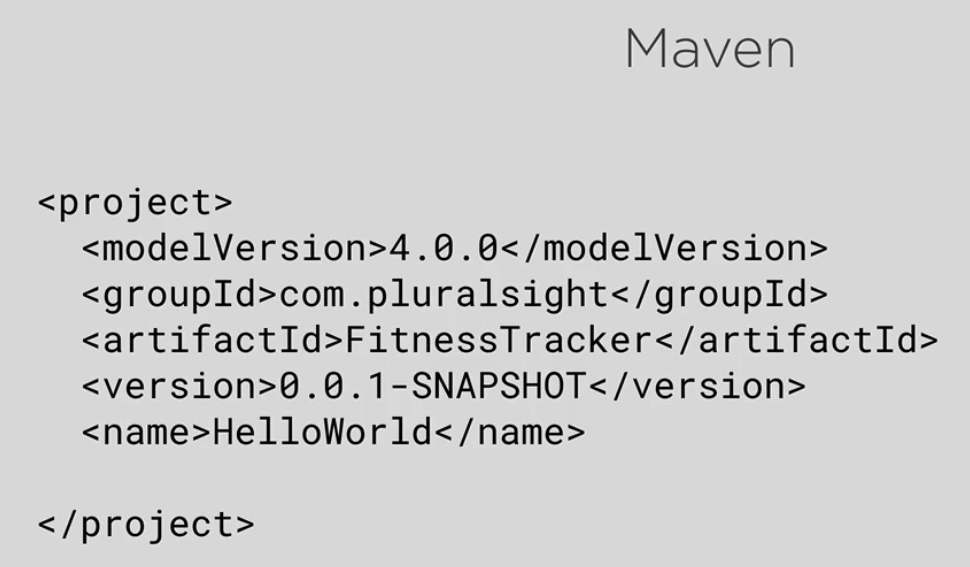


**Very straight-forward and clear** but we need tribal knowledge and run in specific order.

It is **brittle**.

We can skip steps.

**BASIC MAVEN POM FILE**

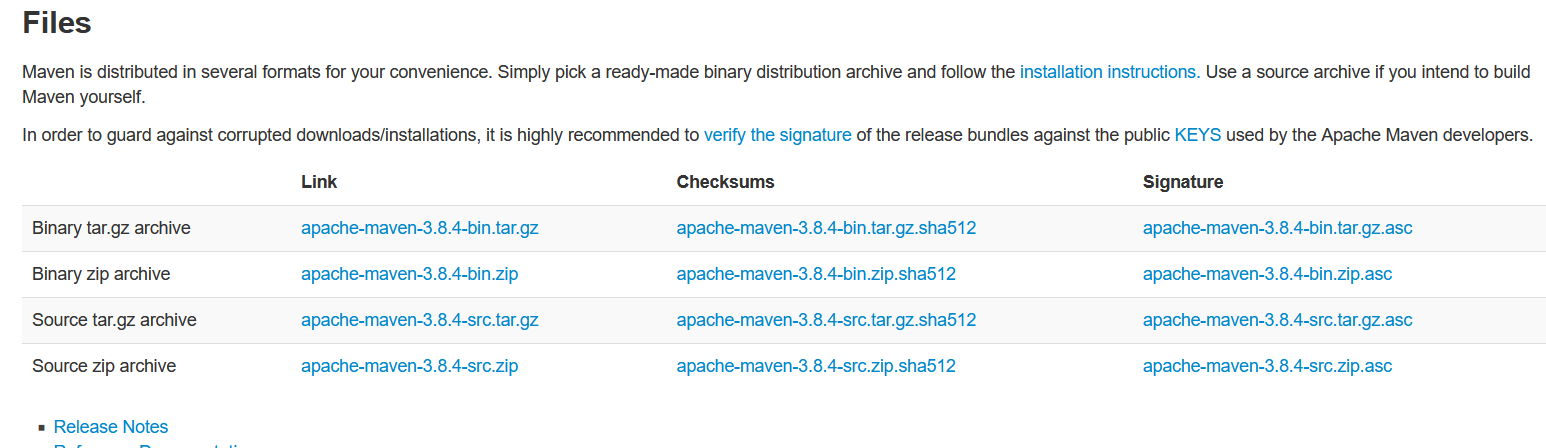


Once we run pom file, maven will do clean, compile and package the project. It is kind of black box.

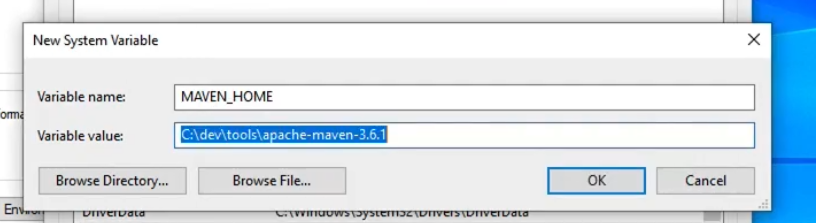
**Simpler but confusing as it is not descriptive like ANT.**

**INSTALLATION BEST PRACTICE FOR MAVEN.**

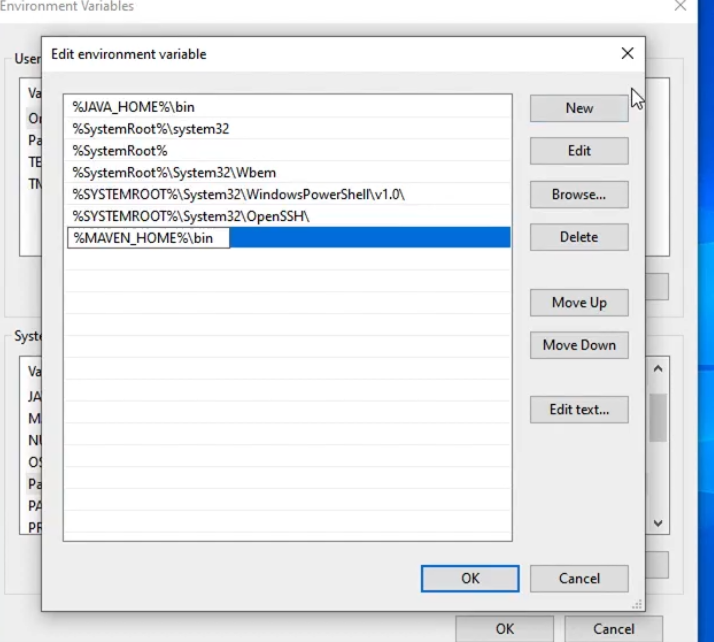
* Go to <https://maven.apache.org/>
* Go downloads and FILEs and download the **binary zip archieve.**



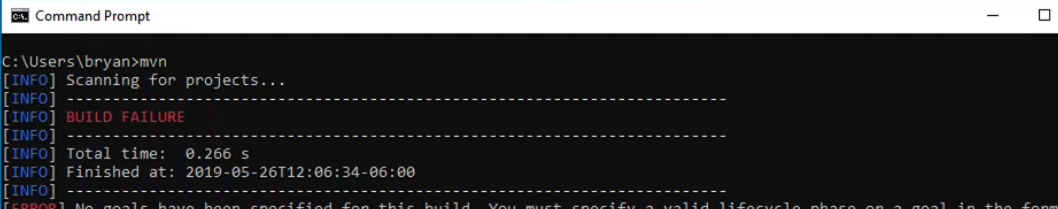
* **Go to system environment variable**



* **EDIT PATH ALSO**



**Go to cmd and type mvn**

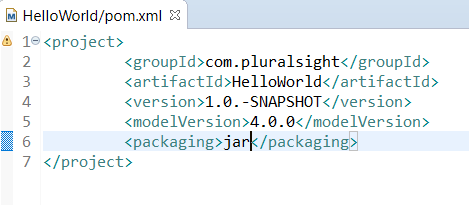


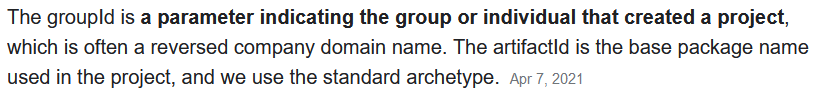
**mvn -version will give us all info.**



**BASIC HELLOWORLD FROM START:**

1. File->new ->general->project
2. File->new->pom.xml
3. **Now add 5 things in pom**

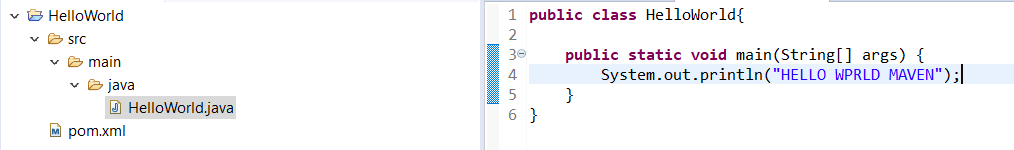




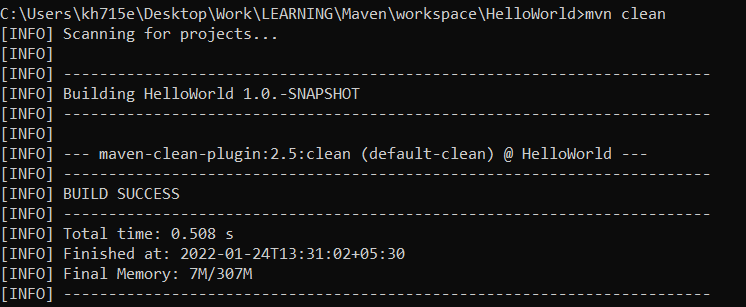
**ArtifactId is the name of the jar without version**. If you created it then you can choose whatever name you want with lowercase letters and no strange symbols. If it's a third party jar you have to take the name of the jar as it's distributed. eg. maven, commons-math

**GroupId will identify your project uniquely across all projects**, so we need to enforce a naming schema. It has to follow the package name rules, what means that has to be at least as a domain name you control, and you can create as many subgroups as you want. Look at More information about package names. eg. org.apache.maven,

1. Create folder and java file like below



1. Now do cmd to the location inside HelloWorld project ,basically it will look for pom file and do **mvn clean** – It will download some dependency and plugin from maven repo.

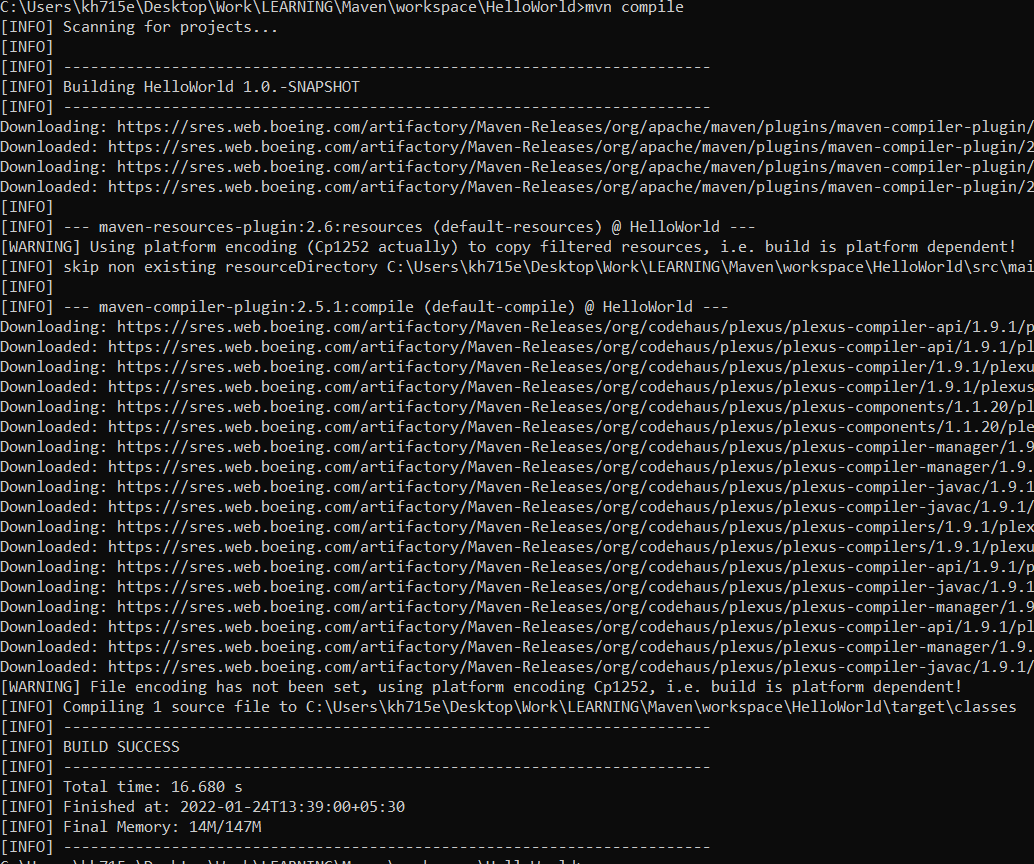


**Mvn clean will initialize your directory structure and delete existing build file.**

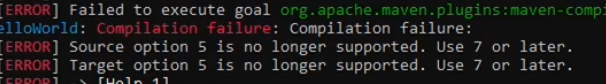
1. **Now do mvn compile –** it will again download plugins and compile our source code.

**In Boeing it downloads from** [**https://sres.web.boeing.com**](https://sres.web.boeing.com)**....**

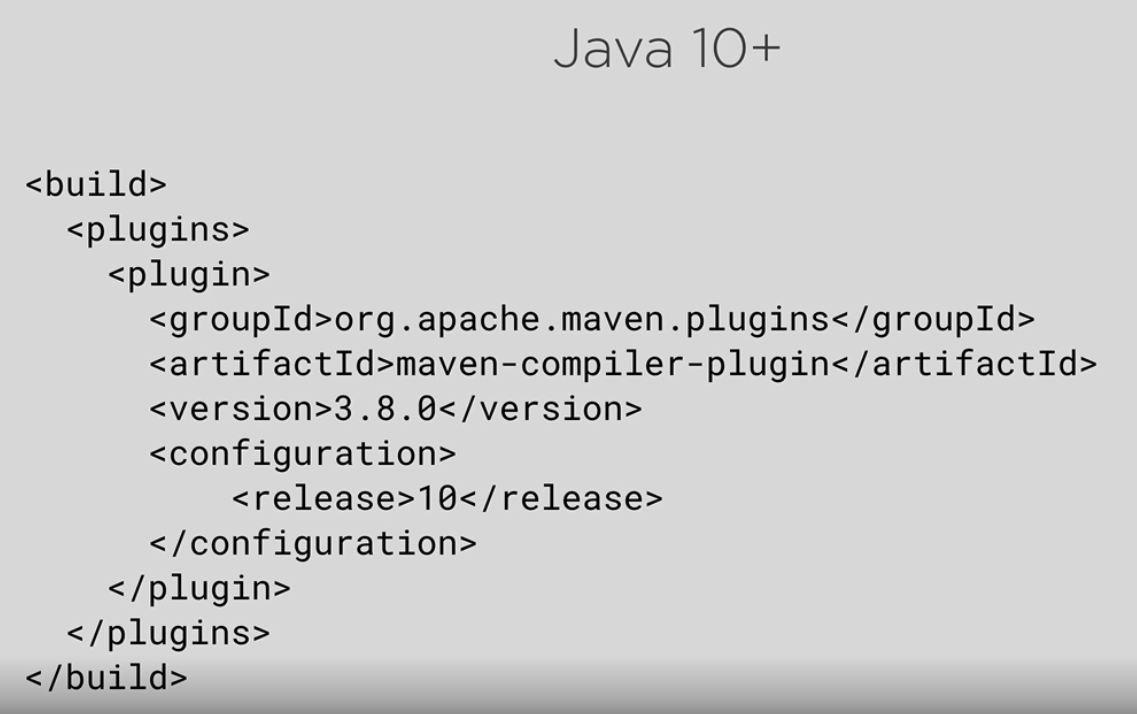
**Normal central maven repo: https://repo.maven.apache.org...**

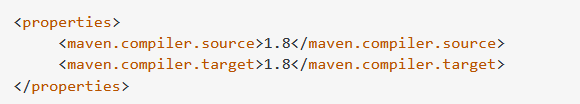


1. If we use JAVA 10+ then during compile issue can come source and target option 5 not supported so we need to include properties in pom.



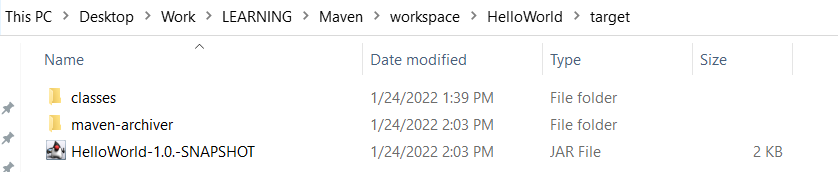
**SOLUTION we have 2 option:**





1. Once compilation is successful we can go to target classes and do **java HelloWorld** to run it.
2. Now we can go to project location and do **mvn package** and it will create the artifact jar for us.

**JAR name is** **artifact+version+.jar**

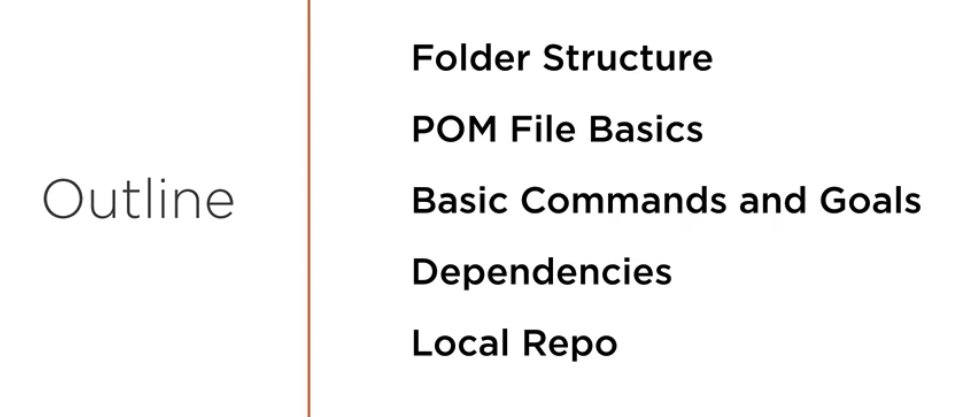




MAVEN IS CONVENTION OVER CONFIGURATION:

If we do naming convention then we don’t need to do scripting configuration like ant.

**STRUCTURE**

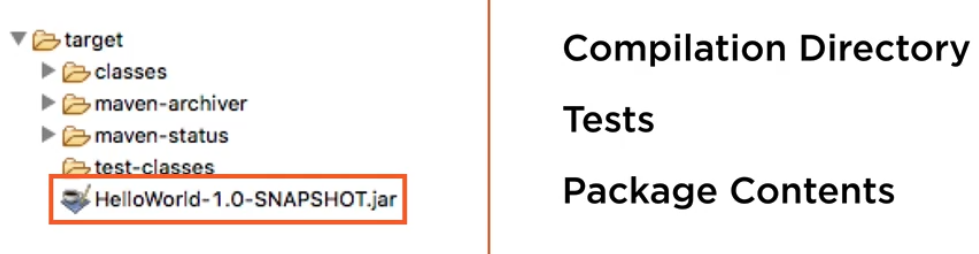


**FOLDER STRUCTURE**



**Src , target and pom file are all at same level (top level)**

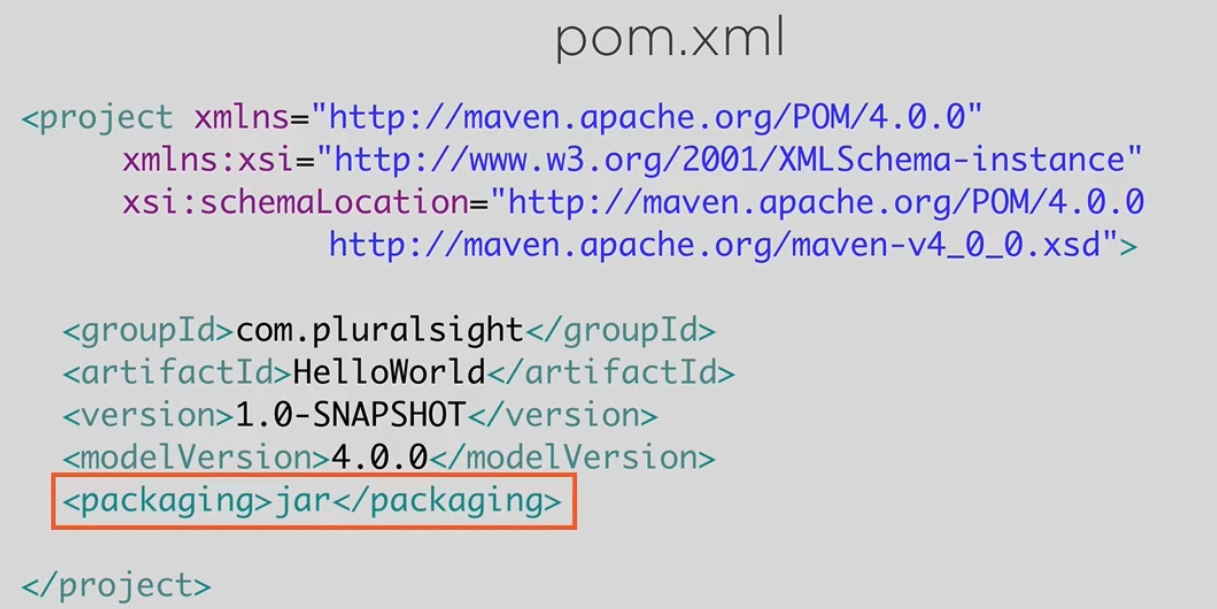




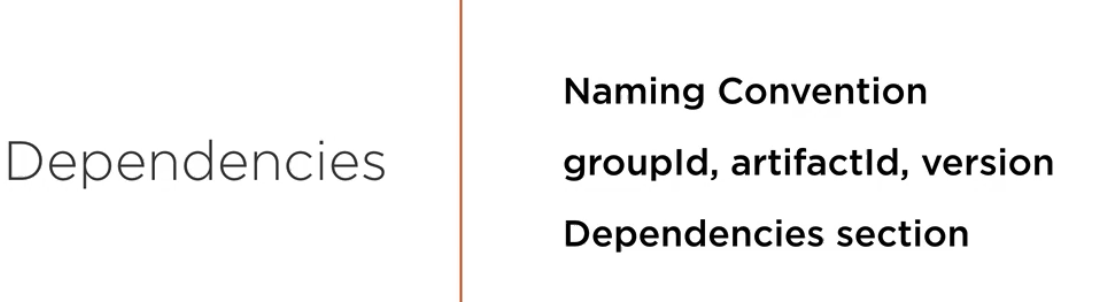
**pom.xml file**

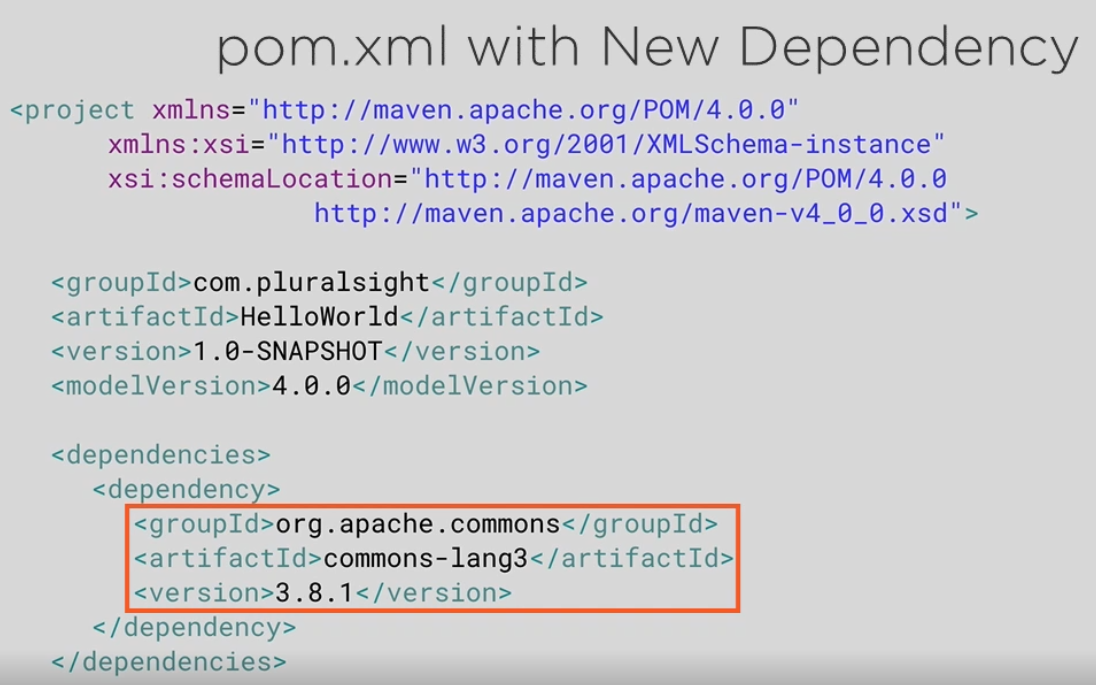
**4 parts:**

* **Project**
* **groupId – package name (business name)**
* **artifactId – final package or jar is created**
* **version – what we want to call our jar**
* **packaging – how we distribute our project.**



**DEPENDENCIES**



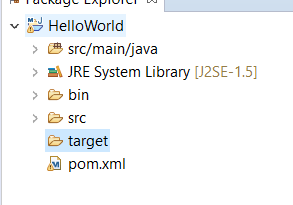


**GOALS**

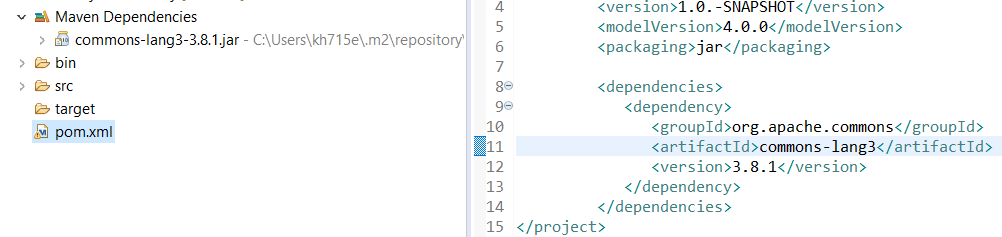
* **Clean –** deletes all the target folder and any of generated sources.
* **Compile-** compile all source code and place in classes directory
* **Package –** run compile first, run the unit test and package that up
* **Install-** run package command and install in **local repo**. Home/.m2
* **Deploy –** runs install command and deploy to **remote repo**

**Now we convert our normal project above into Maven project by using configure command in STS.**

Structure will change



**Once we add any dependency in pom, it will automatically download that and create a maven dependencies folder and add those library there.**



Now got to cmd and in folder structure, do **mvn clean** and **mvn install**

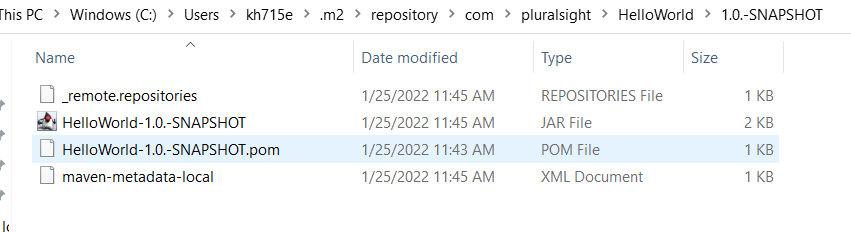
Mvn install will do compile, it will package and finally install the jar in .m2 location

**FROM**



**TO**





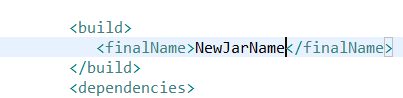
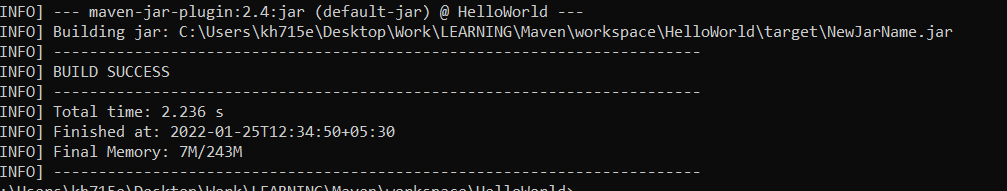
**DEFAULTS**

How do we override those defaults that maven provide.

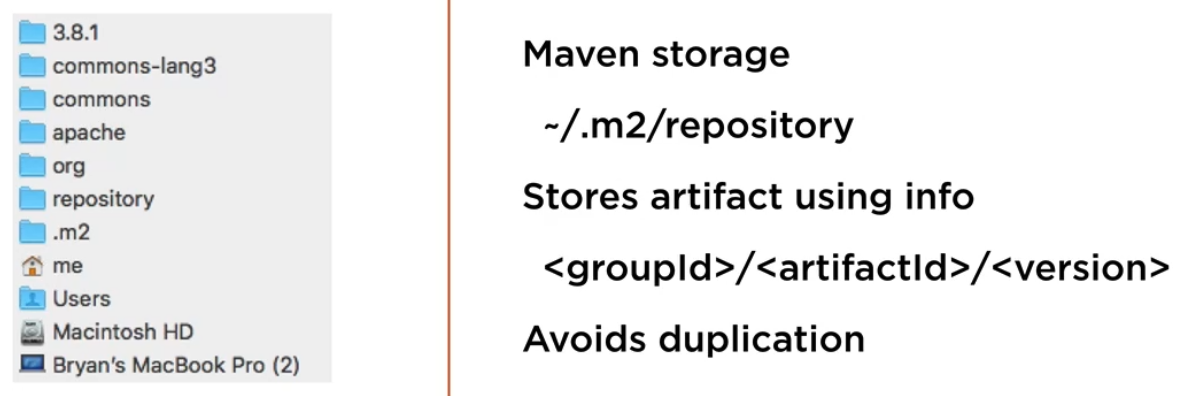
We can override it in **Build** section.



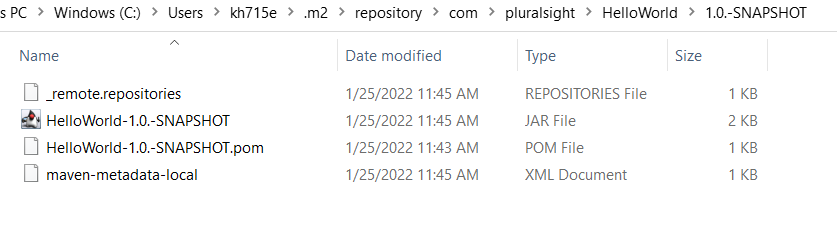
Now if we want to change the final jar that is created then we can do so in build section.



**Local repo Structure .m2 location.**



**It stores everything as groupId/artifact/version**



**Groupid - com/pluralsight**

**Artifact -Helloworld**

**Version – 1.0-SNAPSHOT**