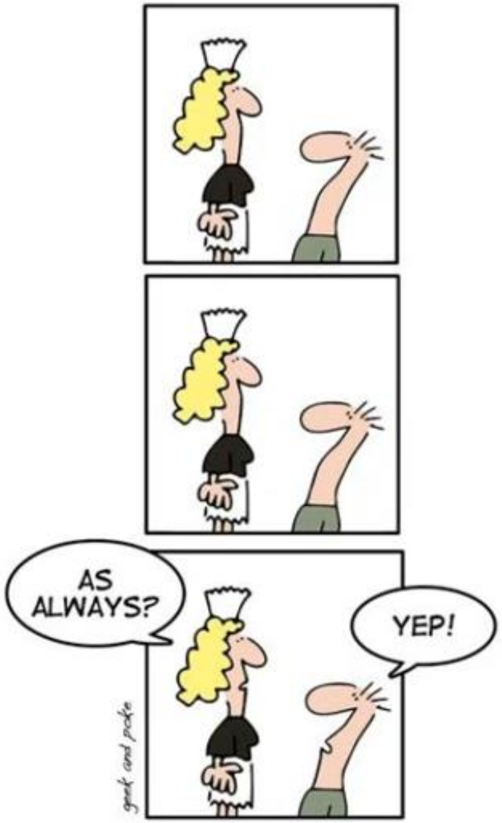
Maven

Agenda

WHAT IS MAVEN

“Maven is a project management tool which encompasses a project object model, a set of standards, a project lifecycle, a dependency management system, and logic for executing plugin goals at defined phases in a lifecycle. When you use Maven, you describe your project object model, Maven can then apply cross-cutting logic from a set of shared (or custom) plugins.”
From http://maven.apache.org/

Convention Over Configuration



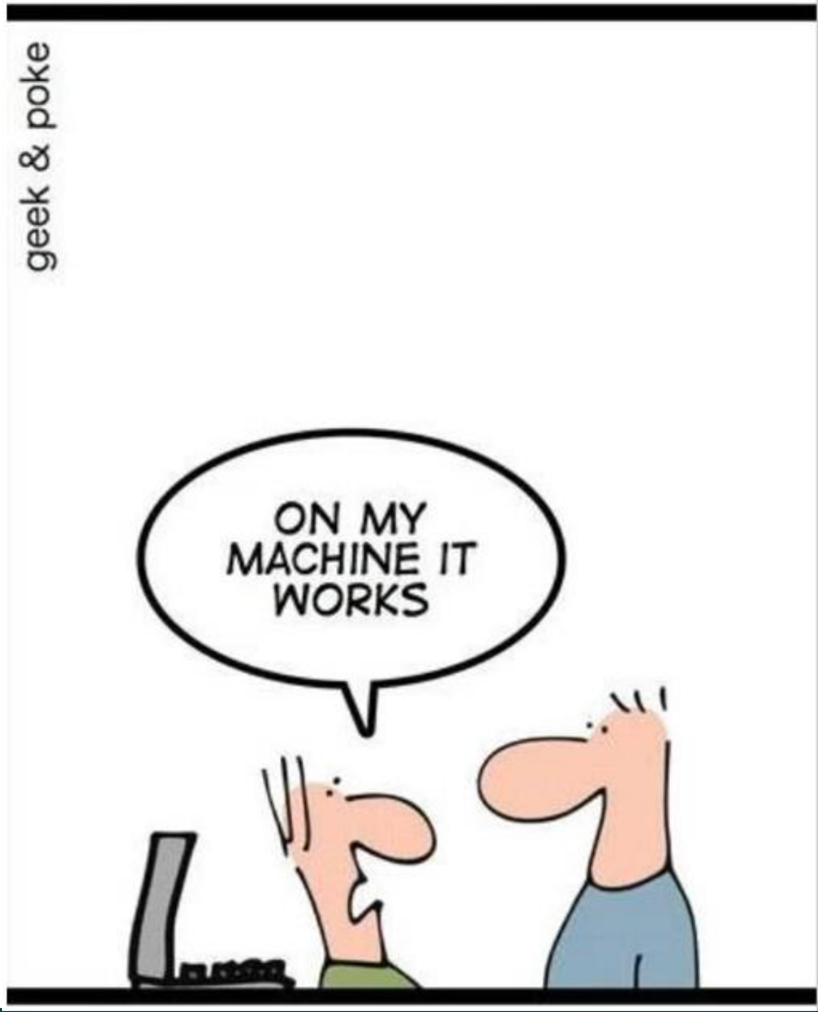
Motivation

The challenges

- How do you compile your java code?
- How do you package your code into a JAR file?
- How do you run unit tests?
- What does the project need to build?
- What libraries do I need to download?

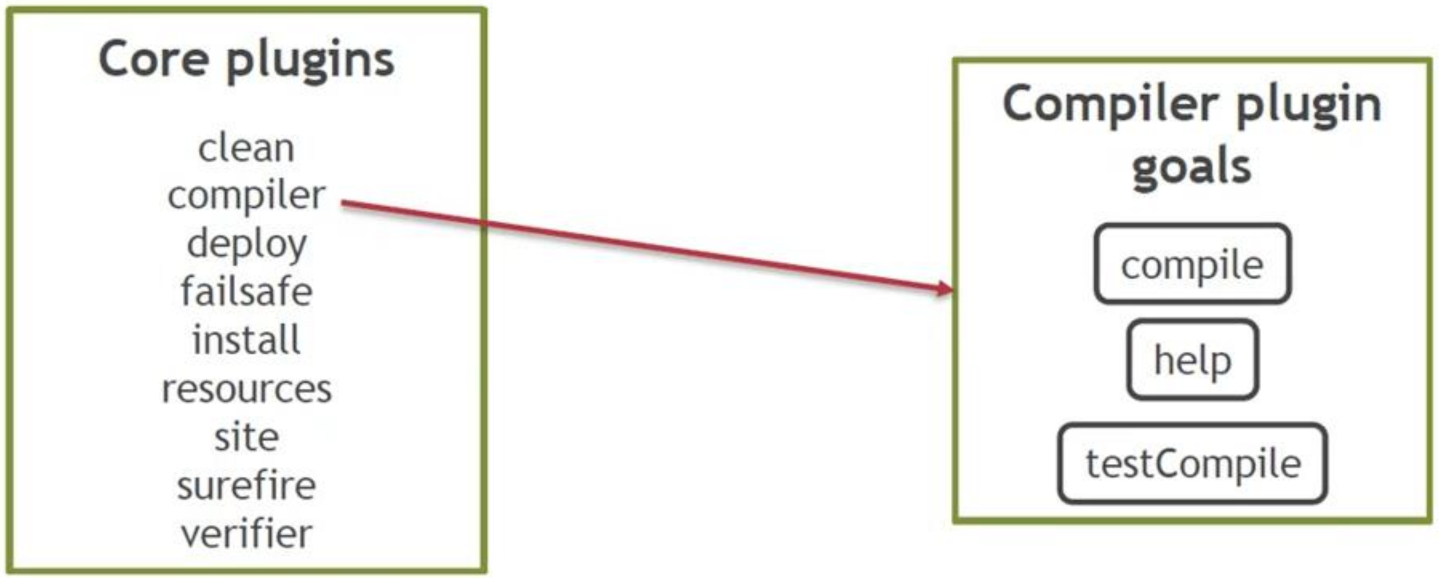
The solution

- Easy as: *mvn install*



Maven plugins

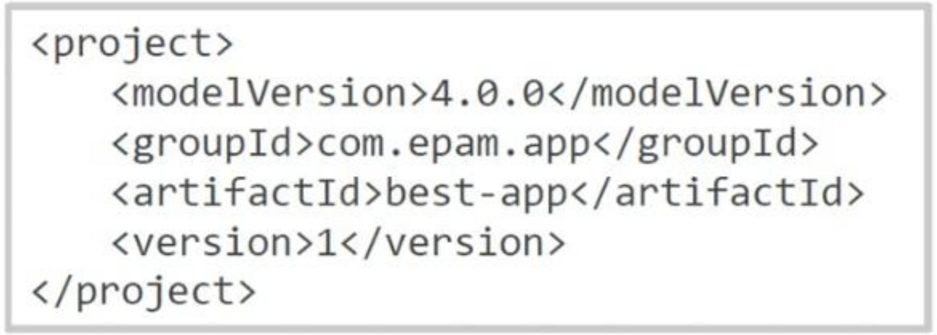
DELEGATE MOST RESPONSIBILITY TO SET OF MAVEN PLUGINS



HOW TO USE MAVEN

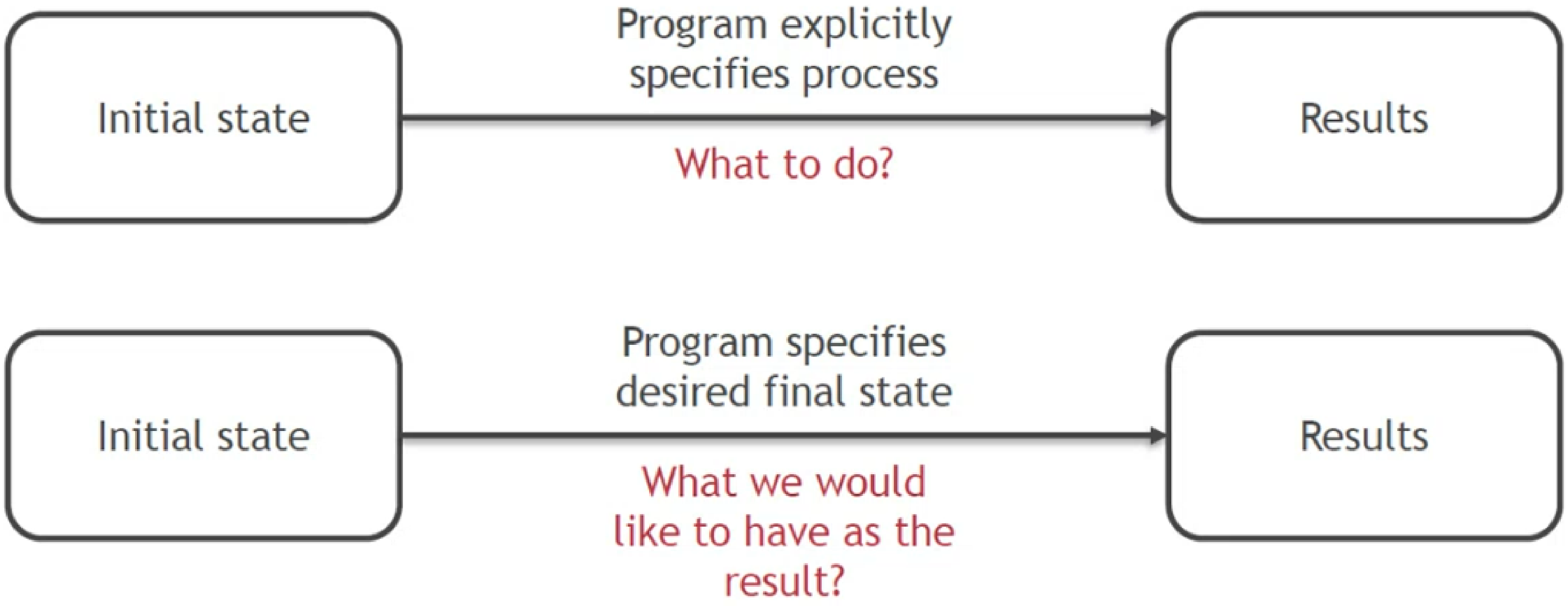
Conceptual Model of a “Project”

- Description of a software project
- Attributes of the project
- Project coordinates
- Enables features such as:
  - Dependency management
  - Remote repositories
  - Universal reuse of build logic
  - Tool Portability / Integration
  - Easy Searching and Filtering



```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.epam.app</groupId>
    <artifactId>best-app</artifactId>
    <version>1</version>
</project>
```

Declarative vs. Imperative


Initial state
Program explicitly specifies process
What to do?
Results
Initial state
Program specifies desired final state
What we would like to have as the result?
Results

Lifecycles and Phases

LIFECYCLES

- Default
- Clean
- Site

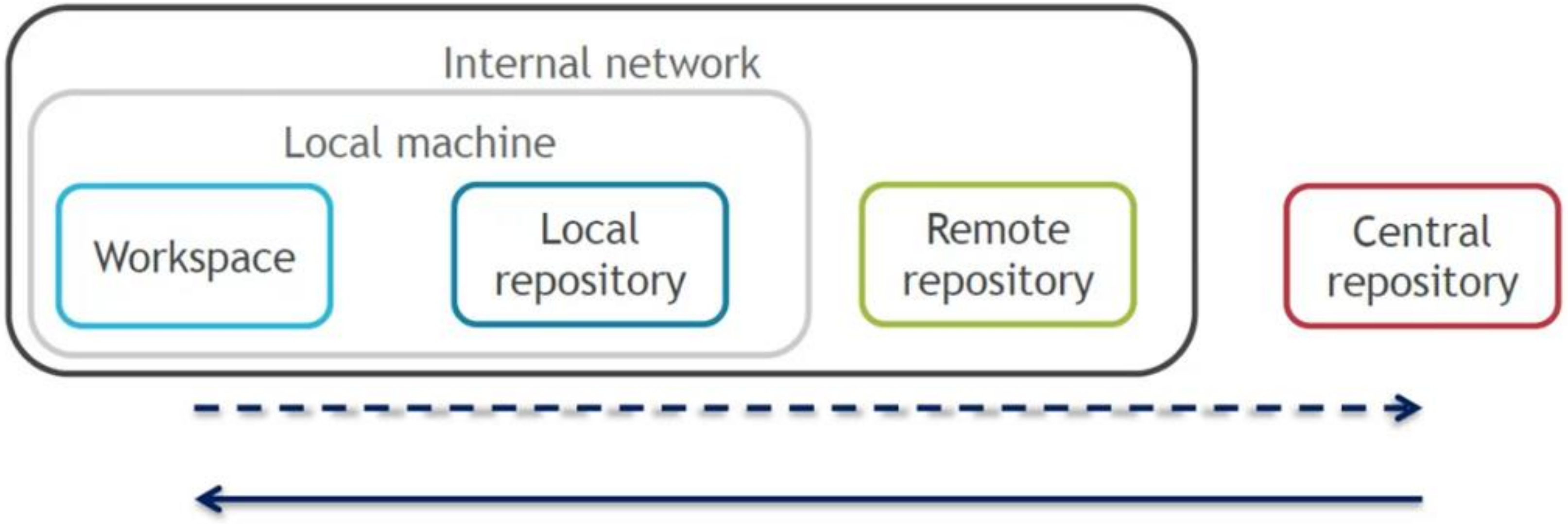
DEFAULT LIFECYCLE PHASES

- **validate** – validate the project is correct and all necessary information is available
- **compile** – compile the source code of the project
- **test** – test the compiled source codeusing a suitable unit testing framework. These tests should not require the code be packaged or deployed
- **package** – take the compiled code and package it in its distributeable format, such as JAR.
- **verify** – run any checks on results of integration tests to ensure quality criteria are met.
- **install** – install the package into the local repository, for use as a dependency in other projects locally
- **deploy** – done in the build environment, copies the final package to the remote repository for sharing with other developers and projects

Dependency resolution

```
<project>
  . . .
  <dependencies>
    <dependency>
      <groupId>test</groupId>
      <artifactId>1.0</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
  . . .
</project>
```

Maven repositories



Maven project folder structure

```
first-project
| - pom.xml
| - src
| --- main
| ---- java
| ----- com
| ------ epam
| ------- training
| -------- App.java
| ---- resources
| --- test
| ---- java
| ----- com
| ------ epam
| ------- training
| -------- AppTest.java
| ---- resources
```

DEMO