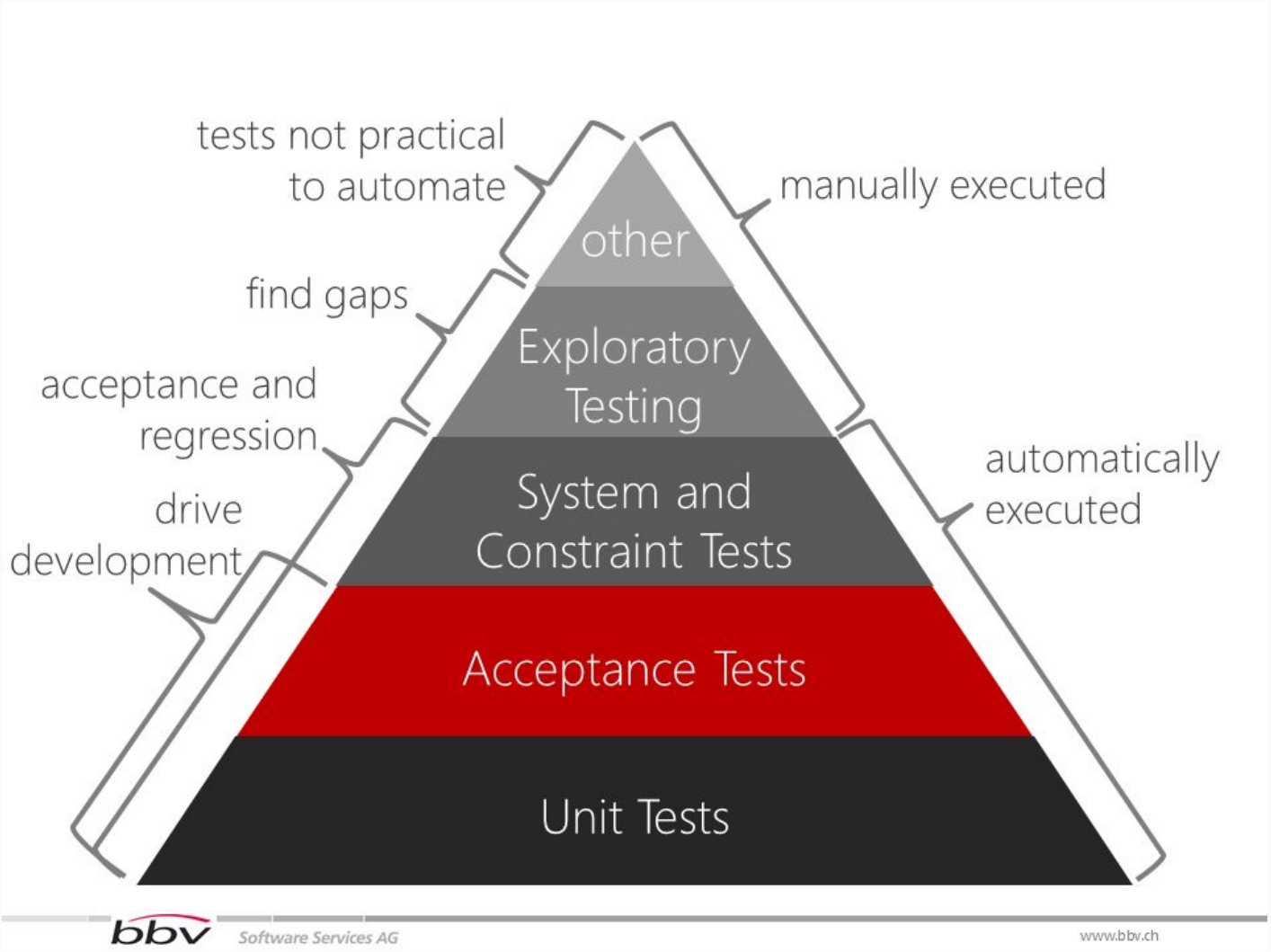
Packaging, automation, Continuous Integration

Based on Simon Urli and Sébastien Mosser courses

18/10/2016

Cécile Camillieri/Clément Duffau

Previously ….


tests not practical
to automate
manually executed
other
find gaps
Exploratory
Testing
acceptance and
regression
automatically
executed
drive
development
System and
Constraint Tests
Acceptance Tests
Unit Tests
bbv
Software Services AG
www.bbv.ch

Development process

Develop features and tests

Launch tests

Develop…

Launch tests…

Release and/or deployment
of a stable version

Development process

Develop features and tests	**code**
	
Launch tests	**commands**
	
Develop…	**code**
	
Launch tests…	**commands**
	
Release and/or deployment of a stable version	**commands**

Development process

Develop features and tests	**code**
	
Launch tests	**commands -> script?**
	
Develop…	**code**
	
Launch tests…	**commands -> script?**
	
Release and/or deployment of a stable version	**commands -> script?**

Compiling during lab sessions

```
azrael:labs mosser$ ls
  Exercice.java Main.java

azrael:labs mosser$ javac *.java

azrael:labs mosser$ java Main
  42
```

Legendary, isn't it?

Sébastion Mosser

How far can you go like this ?

Code in Real-Life™

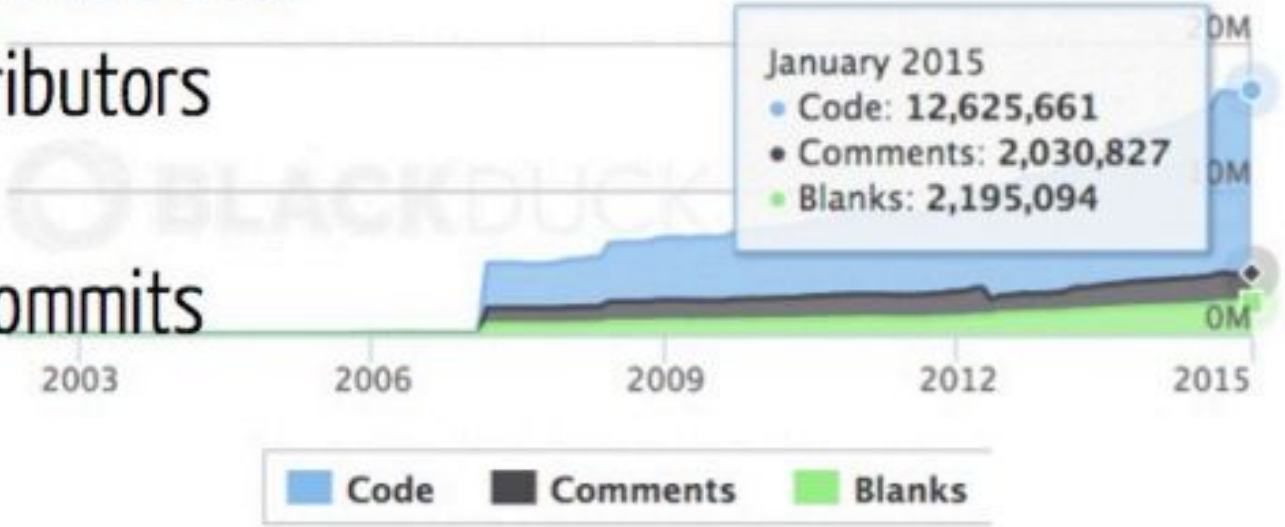
Languages

C++	38%	JavaScript	20%
C	17%	31 Other	25%

Lines of Code

3,254 contributors

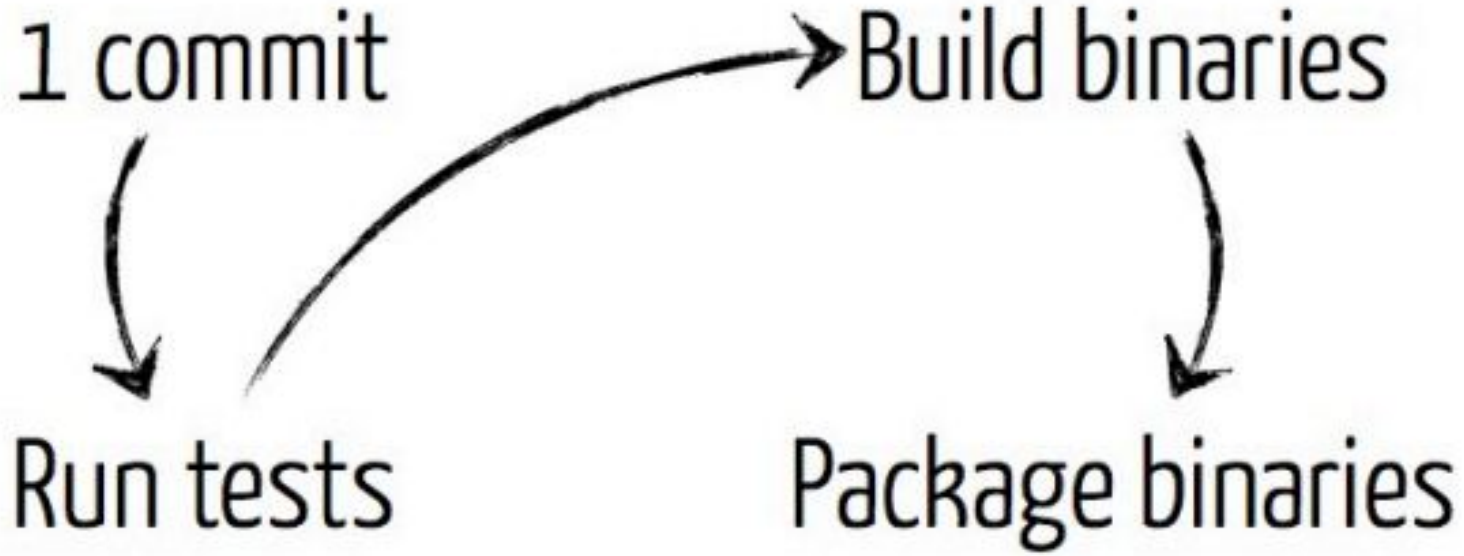
230,442 commits



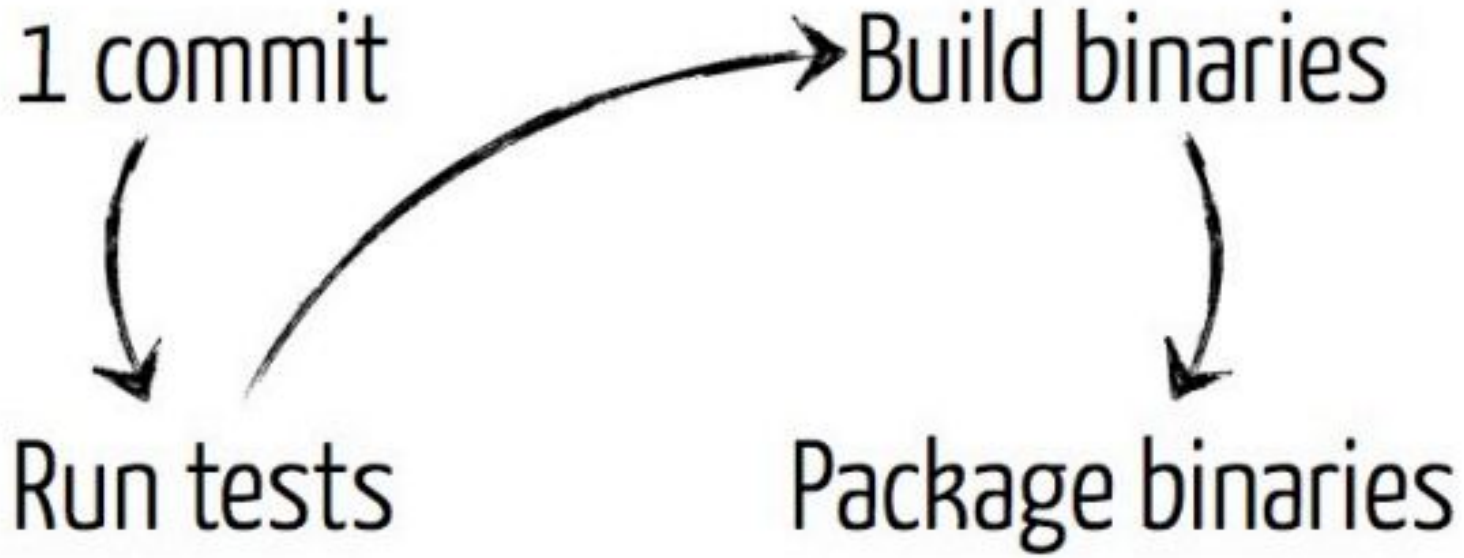
http://www.ohloh.net/p/firefox

4

Ok, That's Impressive... So What?

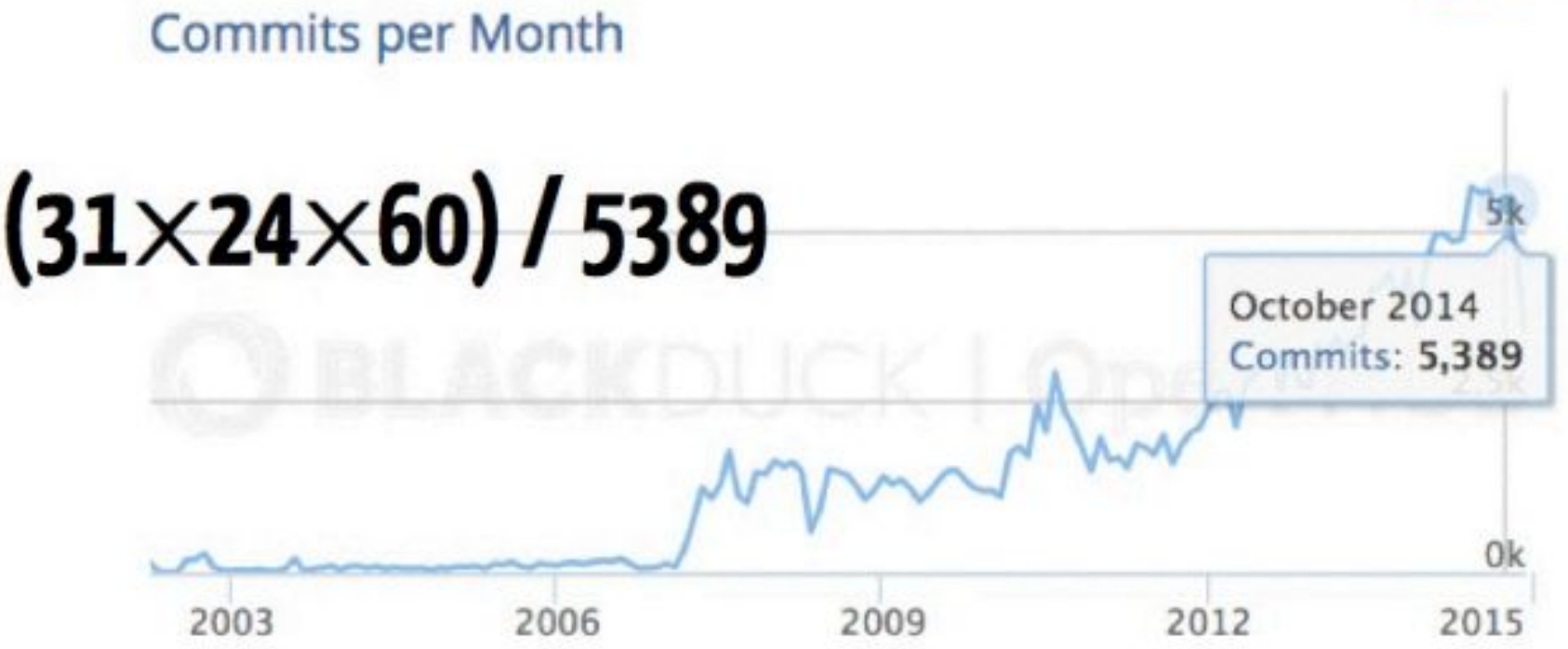


Ok, That's Impressive... So What?



137 hours! (08.2012)

4403 Commits Last Month!



~1commit each 8 minutes

http://www.ohloh.net/p/firefox

6

One **cannot** do it by hand

Towards automation

Automate = **no human intervention**

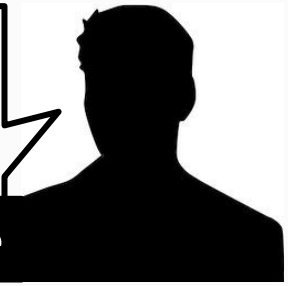
=> Speed

=> Avoid mistakes

=> Concentrate on the code

« Lorsque vous écrivez une commande plus de trois fois, pensez à l'automatiser. »

- Raphaël Marvie



What to automate?

- Creation of a new project: structure, configuration, …
- Creation of a new element: class, resource file, …
- Tests execution
- Publication of tests results
- Packaging: creation of .jar, .zip, …
- Deployment of software
- Documentation creation
- Documentation deployment
- ...

Continuous Integration

Continuous Integration (CI)

Continuous:

“ Forming an unbroken whole; without interruption. ”

Integration:

“ The process of combining two or more things into one, to assemble in order to create an organic whole. ”

Continuous Integration:

The process to assemble without interruption

Continuous Integration

- Automate building of your product
- Have an overview of your system
- Trace the results of these activities

“Building”?

- Source code compilation
- Tests execution
- Software deployment

When to build?

- Clock: scheduled jobs
- Code change: job execution on commit
- Dependencies: after the build of another project

What context for building?

- Different OS: Windows, Mac OS, Linux, ...
- Different architectures: 32/64 bits
- Called nodes

Example scenario

Bob commits and pushes code

Pull the repository

Checkout on master

Compile the source code

Execute tests

Summarize the results

Bob knows if something failed

Example scenario

Bob commits and pushes code

Pull the repository

Checkout on master

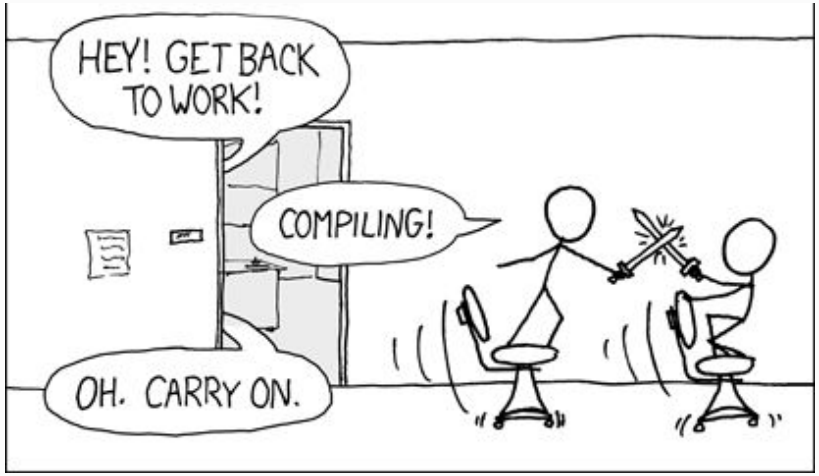
Compile the source code

Execute tests

Summarize the results

Bob knows if something failed

Bob can do something else (hopefully) productive



http://xkcd.com

Tools for CI

Versioning

Build automation

SoapUI Testing

Jenkins

drone.io

Travis CI

Assembly of these tasks: CI

Tools for CI

Versioning
For us: **git**

Build automation
For us: **maven**

maven

SoapUI

JUnit

Testing
For us: **JUnit**

Jenkins

drone.io

Travis CI

Assembly of these tasks: CI
For us: **Jenkins**

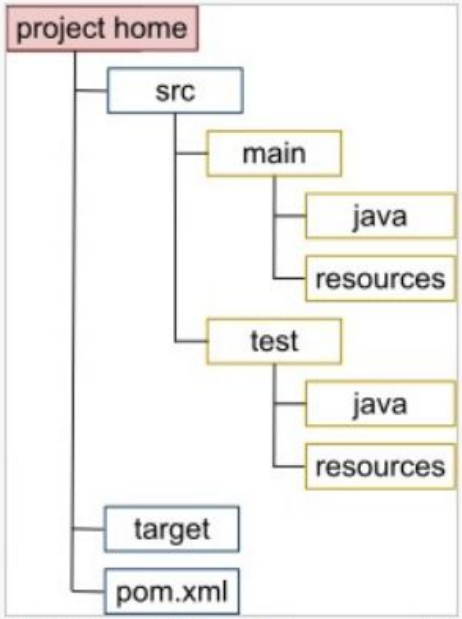
Maven

What's maven?

- Descendant of Make and Ant
- Objectives
  => Handle dependencies
  => Automate tasks (compile, test, …)
- Configuration with a **pom.xml** file
- Javascript equivalent:
  npm for dependencies, grunt for tasks

pom.xml

```xml
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

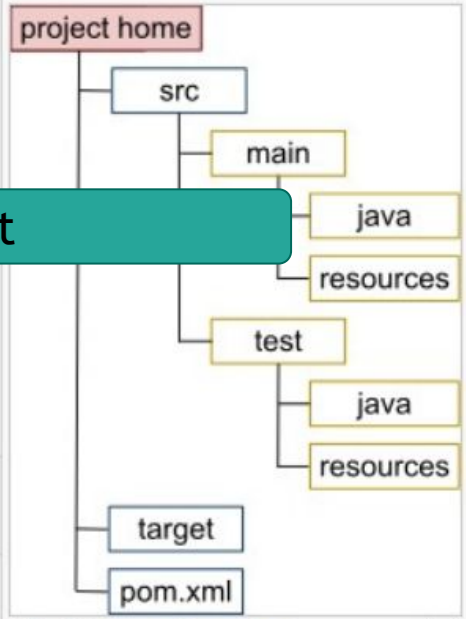


The Maven software tool auto-generated this directory structure for a Java project.

pom.xml

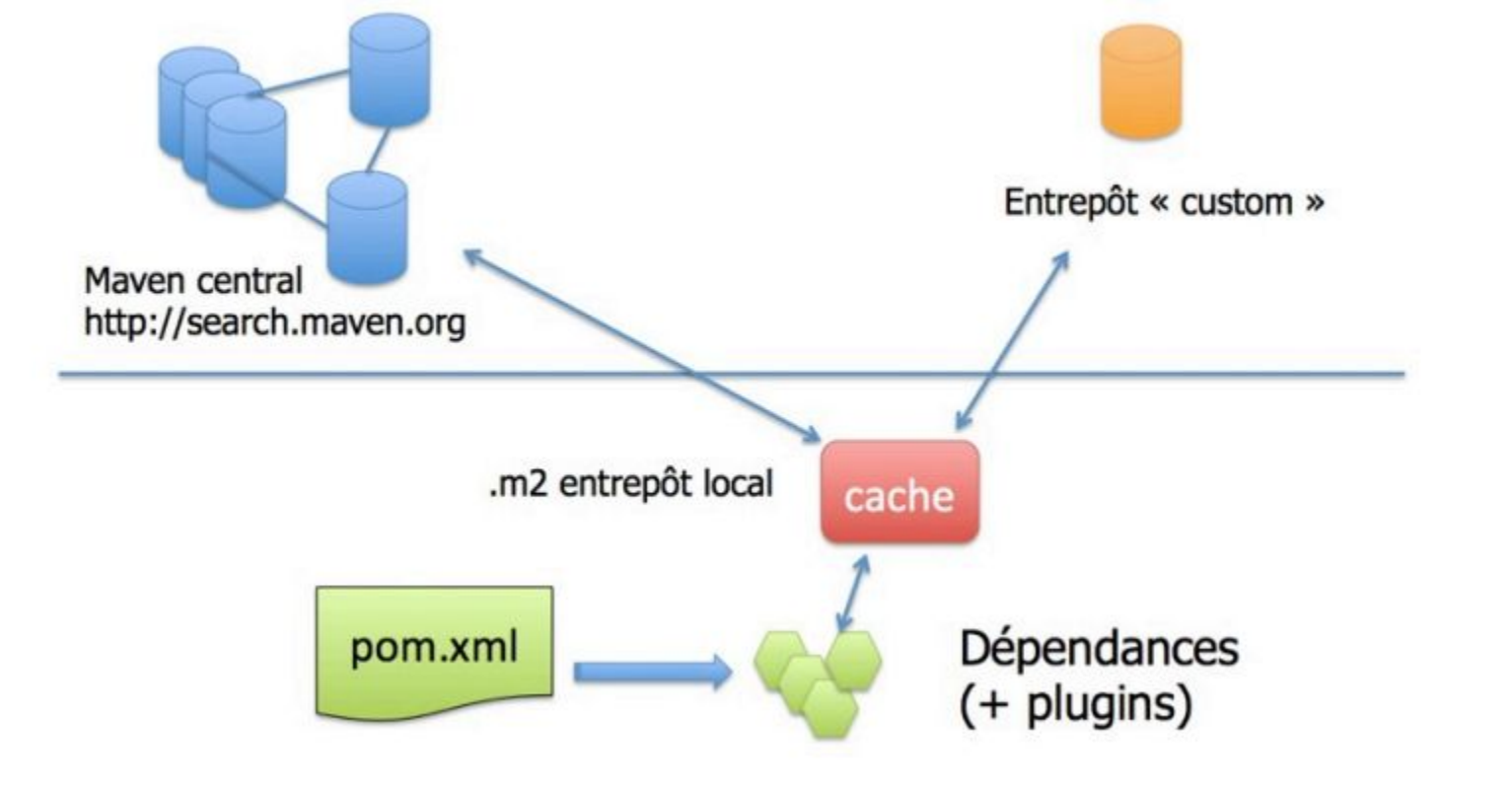
```xml
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates
       uniquely identify this pr

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

Identify uniquely a project

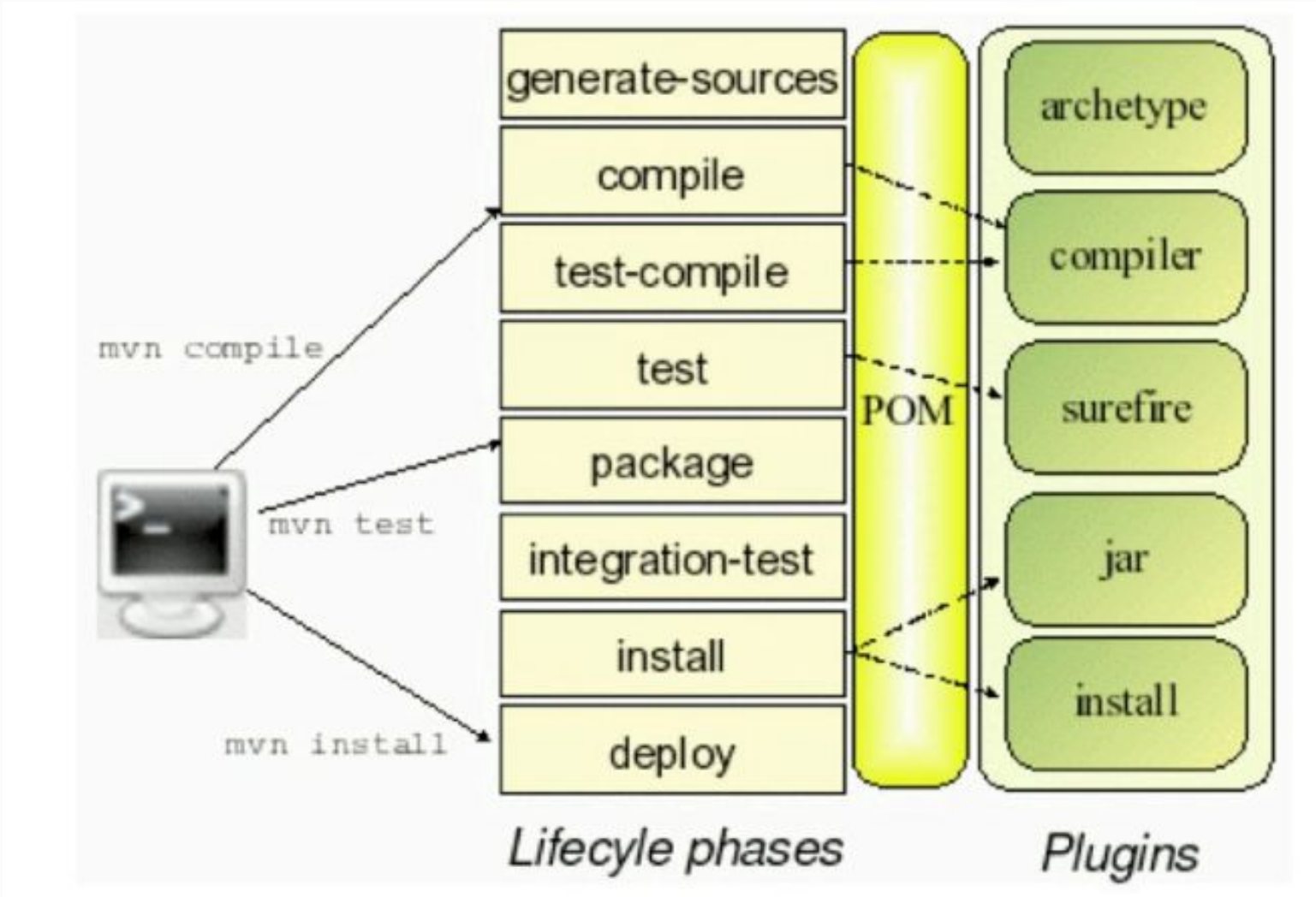


The Maven software tool auto-generated this directory structure for a Java project.

Maven repositories



Default maven lifecycle



Maven goals

- **mvn clean**: clean the project (by default, delete the target folder)
- **mvn compile**: compile the project
- **mvn test**: compile the project and tests and run the tests
- **mvn package**: compile + test then creates a package containing binaries (.jar, .war)
- **mvn install**: package + install the package locally
- **mvn deploy**: install + deploy the package on a remote maven repository

Back to Shapes WS

```
<project>

    <groupId>fr.unice.iut</groupId>
    <artifactId>shapes-ws</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <packaging>war</packaging>

    <dependencies> … </dependencies>

    <build>
        <plugins> … </plugins>
    </build>
</project>
```

Identify uniquely the project

Package into a war (web app)

Required dependencies

External plugins

Back to Shapes WS

```
<dependencies>
    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.1</version>
    </dependency>

    <dependency>
        <groupId>javax.ws.rs</groupId>
        <artifactId>javax.ws.rs-api</artifactId>
        <version>2.0.1</version>
    </dependency>
</dependencies>
```

Serialization to XML/Json

Annotations (@POST, @Consumes, ...

Back to Shapes WS

```
<dependencies>
	<dependency>
		<groupId>org.eclipse.jetty</groupId>
		<artifactId>jetty-servlets</artifactId>
		<version>8.1.10.v20130312</version>
	</dependency>

	<dependency>
		<groupId>com.sun.jersey</groupId>
		<artifactId>jersey-servlet</artifactId>
		<version>1.19</version>
	</dependency>
</dependencies>
```

Handle HTTP communication, deployment (sockets, etc.)

Implementation of the JAX RS interface

Back to Shapes WS

```
<build><plugins>
        <plugin>
            <groupId>org.mortbay.jetty</groupId>
            <artifactId>jetty-maven-plugin</artifactId>
            <version>8.1.10.v20130312</version>

            <configuration>
                <scanIntervalSeconds>10</scanIntervalSeconds>
                <webApp>
                    <contextPath>/</contextPath>
                    <descriptor> [...] web.xml</descriptor>
                </webApp>
            </configuration>
        </plugin>
</plugins></build>
```

Unique identifier of the plugin

Configuration (ports, url, etc.)

Plugins for custom goals

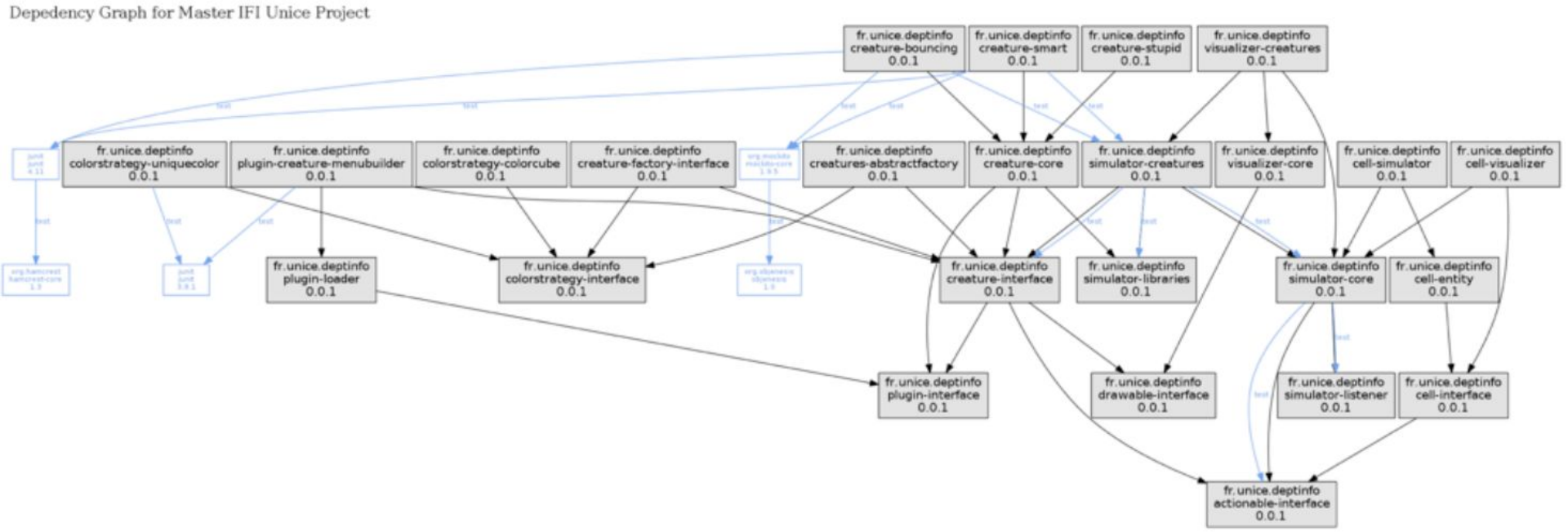
mvn jetty:run

run a jetty (server) instance based on the configuration

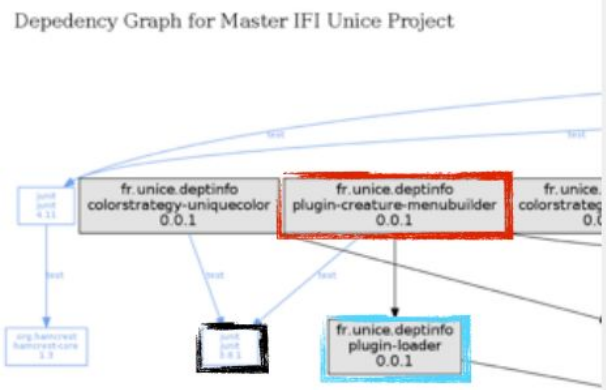
```
<configuration>
    <scanIntervalSeconds>10</scanIntervalSeconds>
    <webApp>
        <contextPath>/</contextPath>
        <descriptor> [...] web.xml</descriptor>
    </webApp>
</configuration>
```

Contains configuration for the web-service

Going further: dependencies



Going further: dependencies



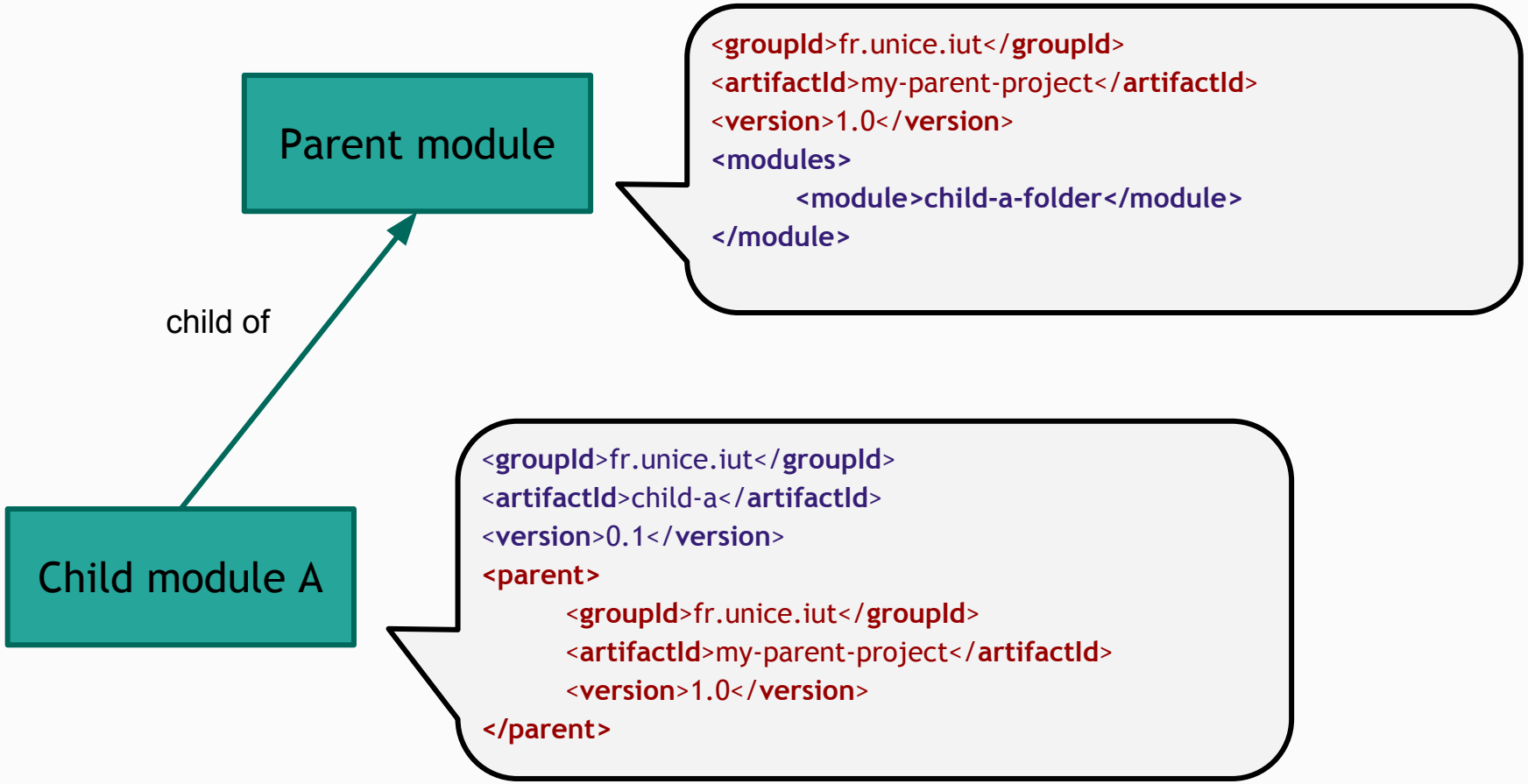

```xml
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>fr.unice.deptinfo</groupId>
    <artifactId>gl-ifi-parent</artifactId>
    <version>0.0.1</version>
  </parent>
  <groupId>fr.unice.deptinfo</groupId>
  <artifactId>plugin-creature-menubuilder</artifactId>
  <version>0.0.1</version>
  <name>plugin-creature-menubuilder</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>fr.unice.deptinfo</groupId>
        <artifactId>creature-interface</artifactId>
        <version>0.0.1</version>
    </dependency>
    <dependency>
        <groupId>fr.unice.deptinfo</groupId>
        <artifactId>plugin-loader</artifactId>
        <version>0.0.1</version>
    </dependency>
  </dependencies>
</project>
```

Going further: modules hierarchy

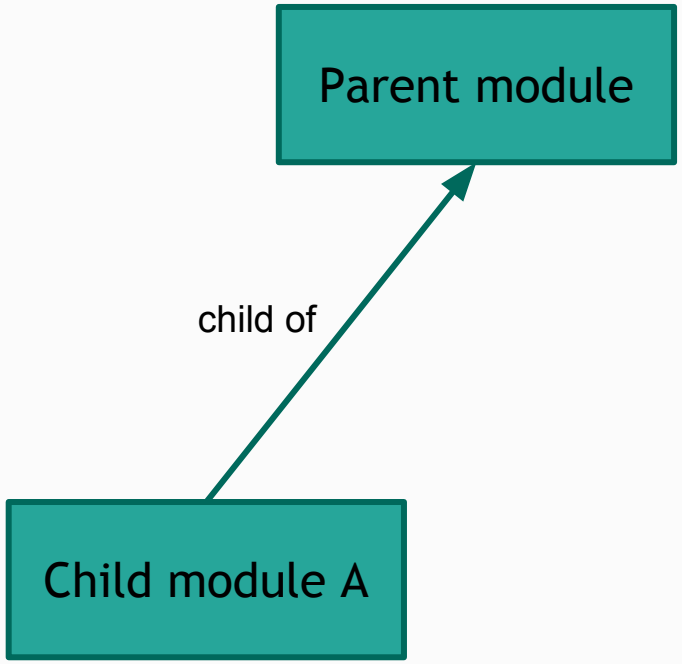
Parent module

```
<groupId>fr.unice.iut</groupId>
<artifactId>my-parent-project</artifactId>
<version>1.0</version>
```

Going further: modules hierarchy



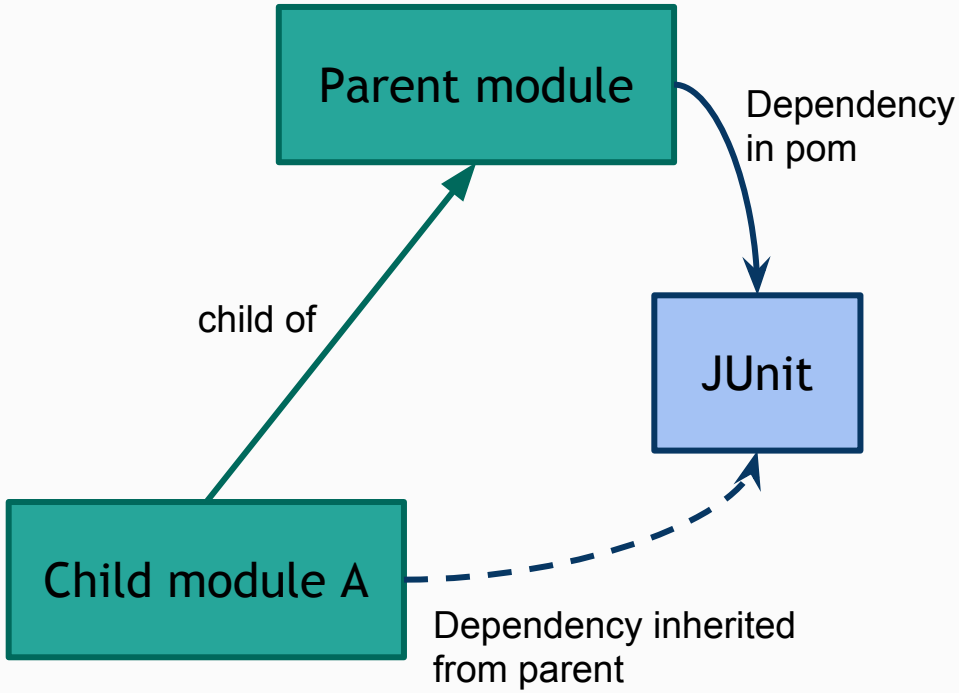
Going further: modules hierarchy



Build parent `mvn install`

Build child `mvn install`

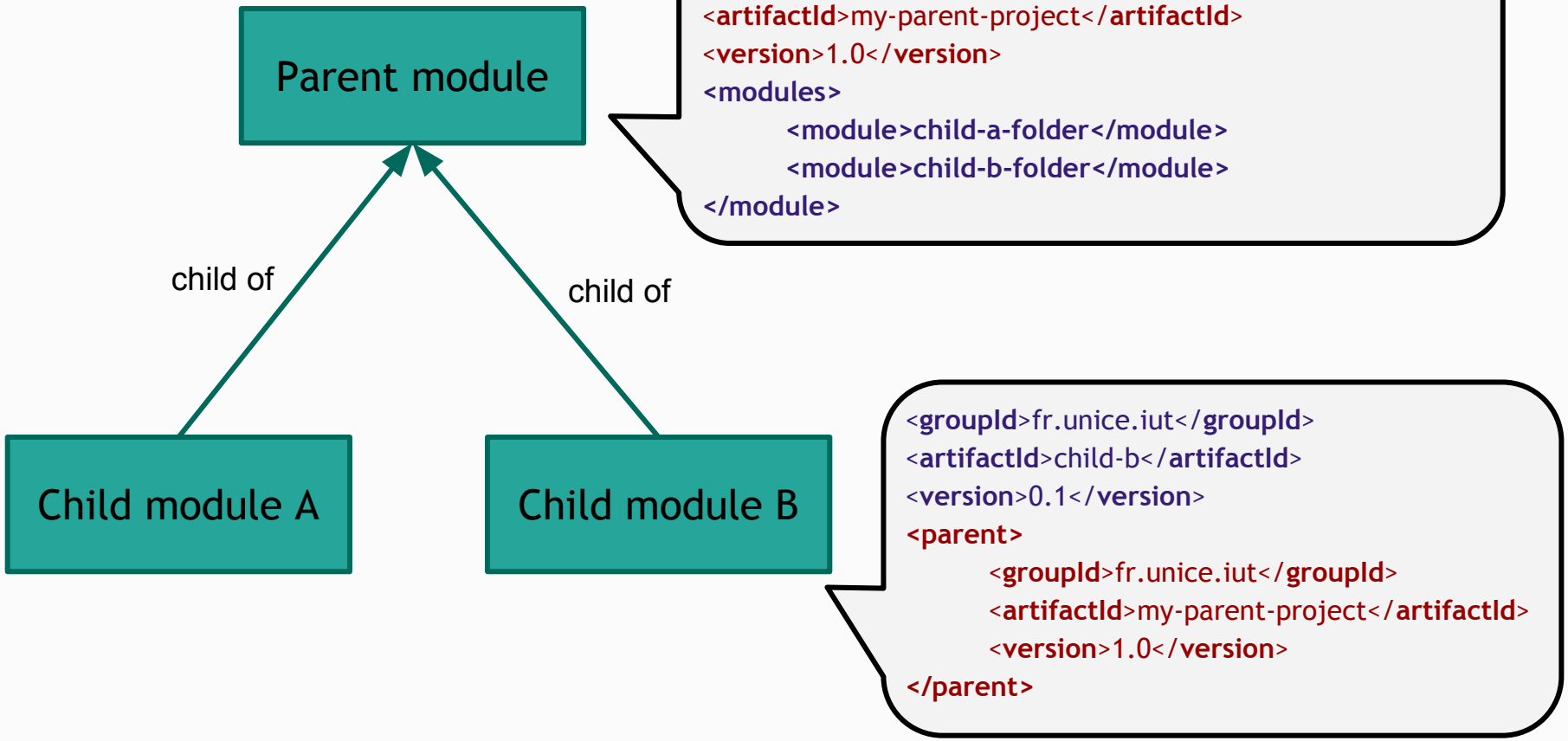
Going further: modules hierarchy



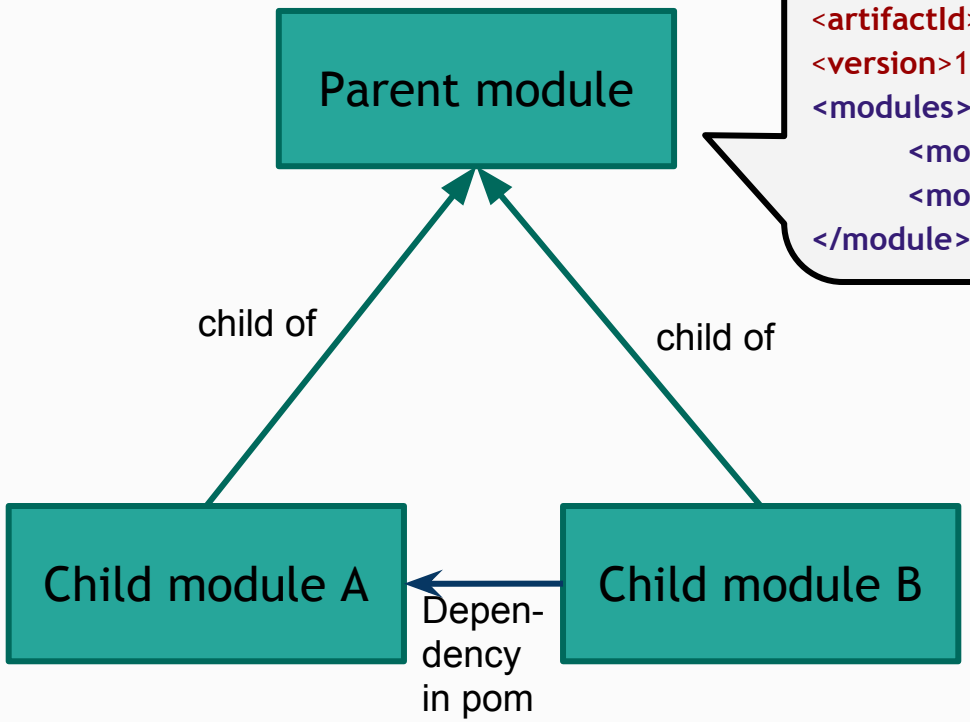
Parent's dependencies

Child's dependencies

Going further: modules hierarchy



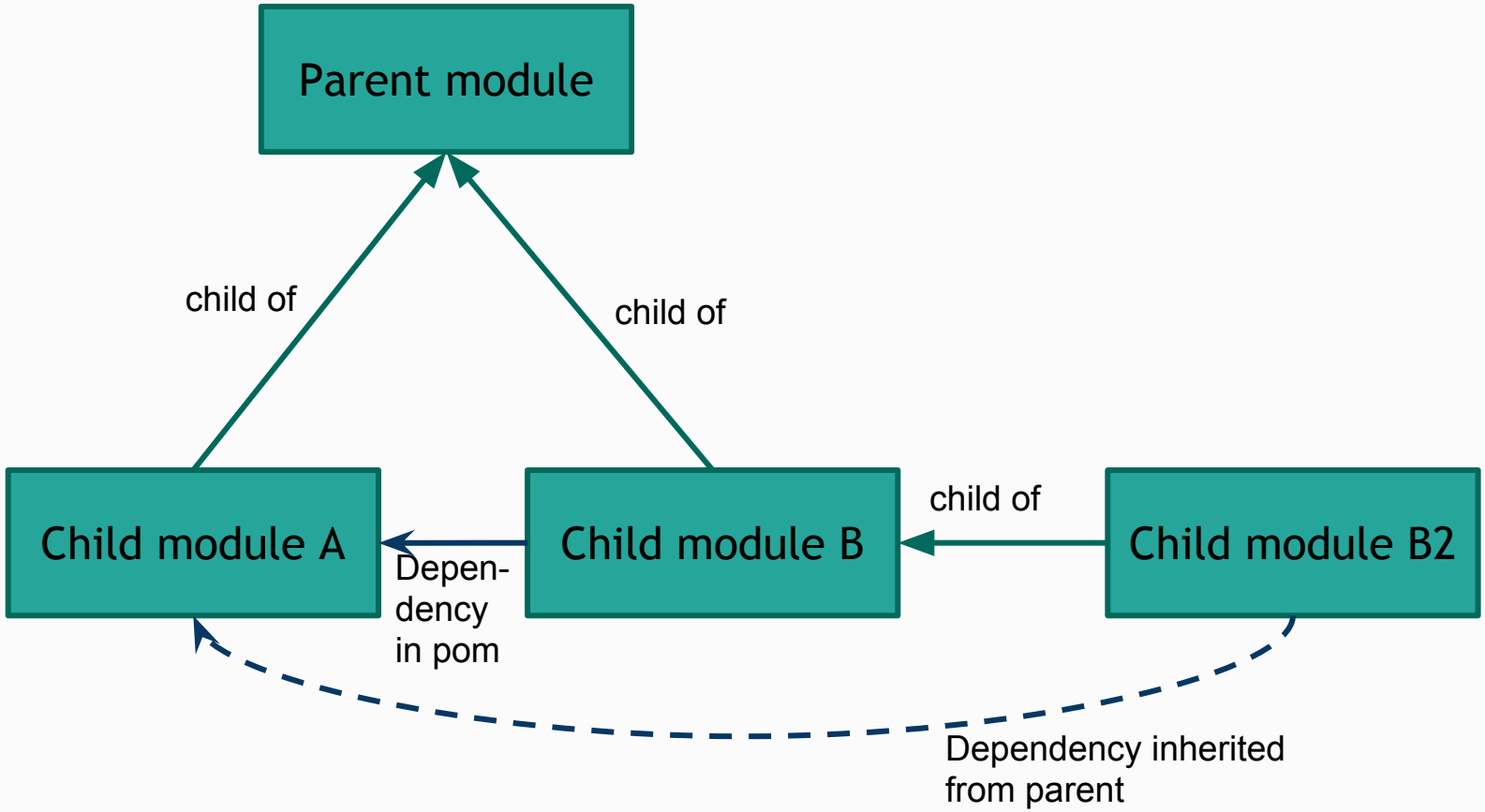
Going further: modules hierarchy




```
<groupId>fr.unice.iut</groupId>
<artifactId>my-parent-project</artifactId>
<version>1.0</version>
<modules>
        <module>child-a-folder</module>
        <module>child-b-folder</module>
</module>
```

```
<groupId>fr.unice.iut</groupId>
<artifactId>child-b</artifactId>
<version>0.1</version>
<parent>
        <groupId>fr.unice.iut</groupId>
        <artifactId>my-parent-project</artifactId>
        <version>1.0</version>
</parent>
<dependencies><dependency>
        <groupId>fr.unice.iut</groupId>
        <artifactId>child-a</artifactId>
        <version>0.1</version>
</dependency></dependencies>
```

Going further: modules hierarchy



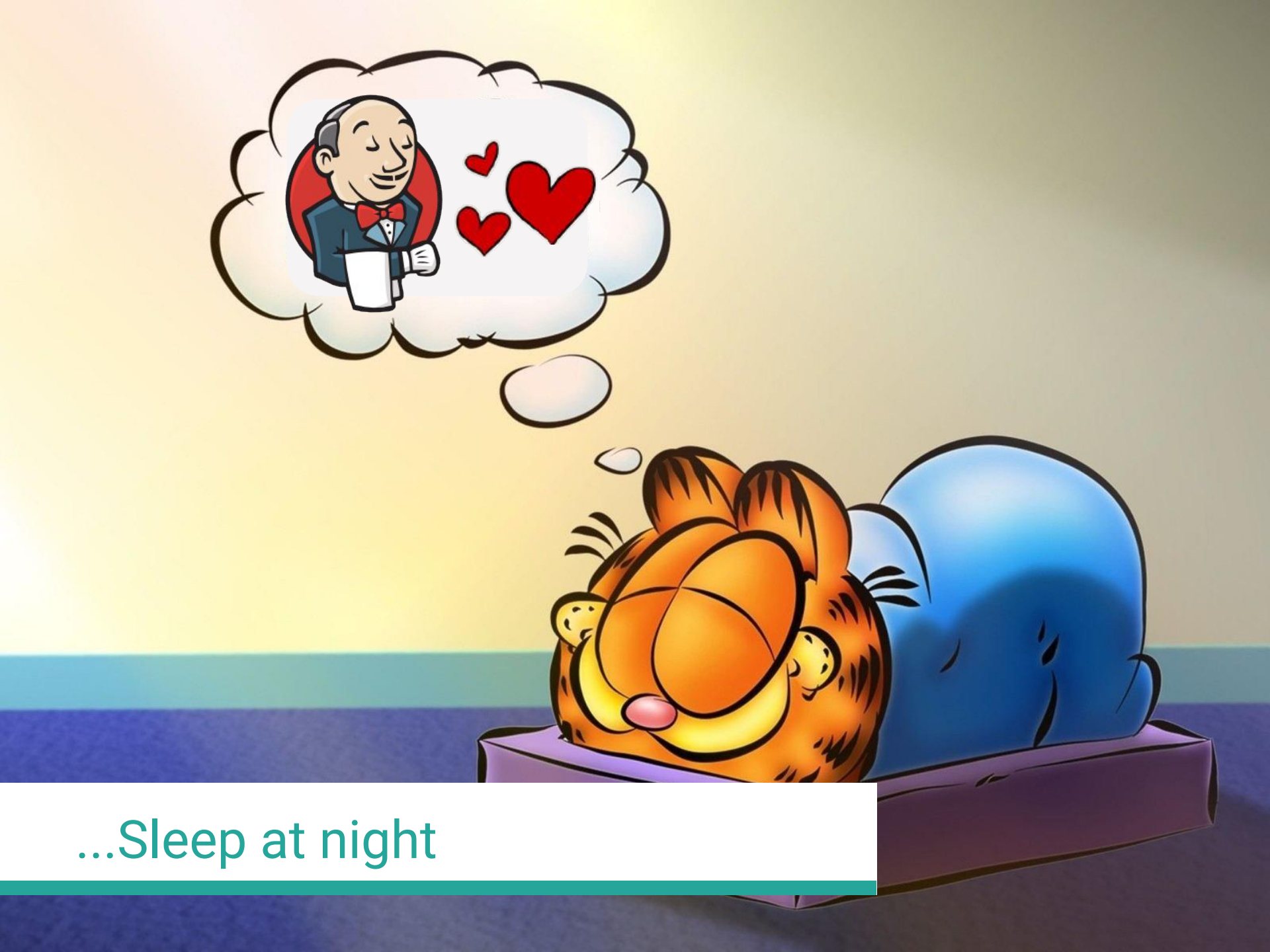
Maven to…

Build…
Test…
Deploy…

CI to…

Automate…

Trace…



...Sleep at night

Jenkins Setup

- http://iut-outils-gl.i3s.unice.fr/jenkins/job/lpidse16-17/job/lpidse16-17/
- Plug-and-play solution with maven goals
- Need a “Jenkinsfile” file at git repository root

The default Jenkinsfile

```
node ('master') {
    checkout scm
    sh 'mvn clean package'
}
```

The node where it will be executed

Keep an eye on multi-branch

The maven goal to apply

The default Jenkinsfile

```
node ('master') {
    checkout scm
    sh 'mvn -f myFolder clean package'
}
```

The server node where it will be executed.
This is not the git branch. Do not change.

Keep an eye on multi-branch

The maven goal to apply
-f to reference the folder
containing the pom.xml file