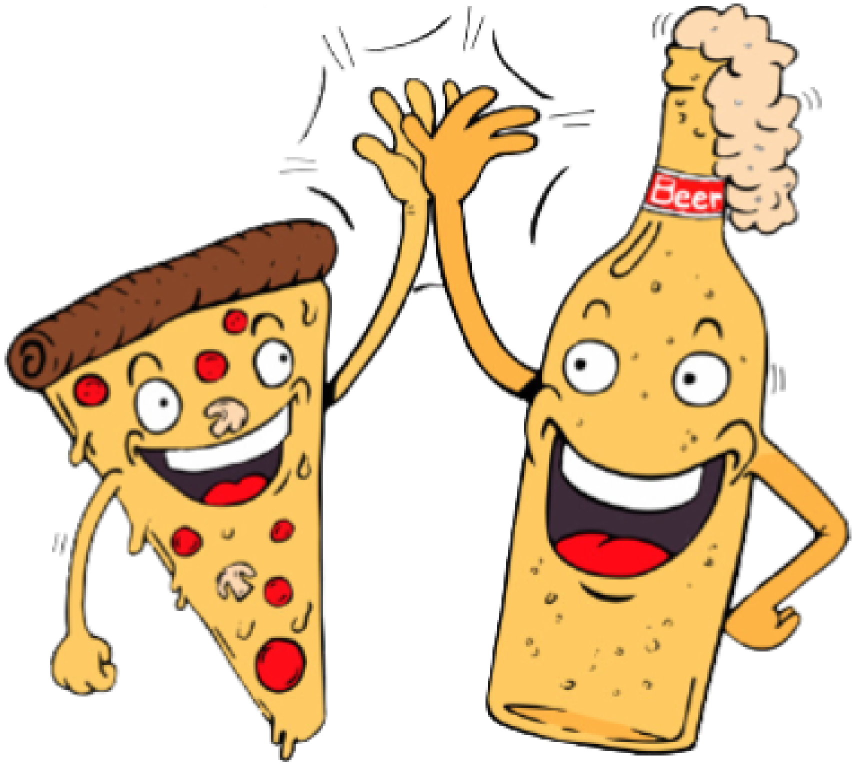
Maven™

OLD BUT GOLD

CONFIGURATION, CUSTOMIZATION, OPTIMIZATION TO SPEED UP BUILD

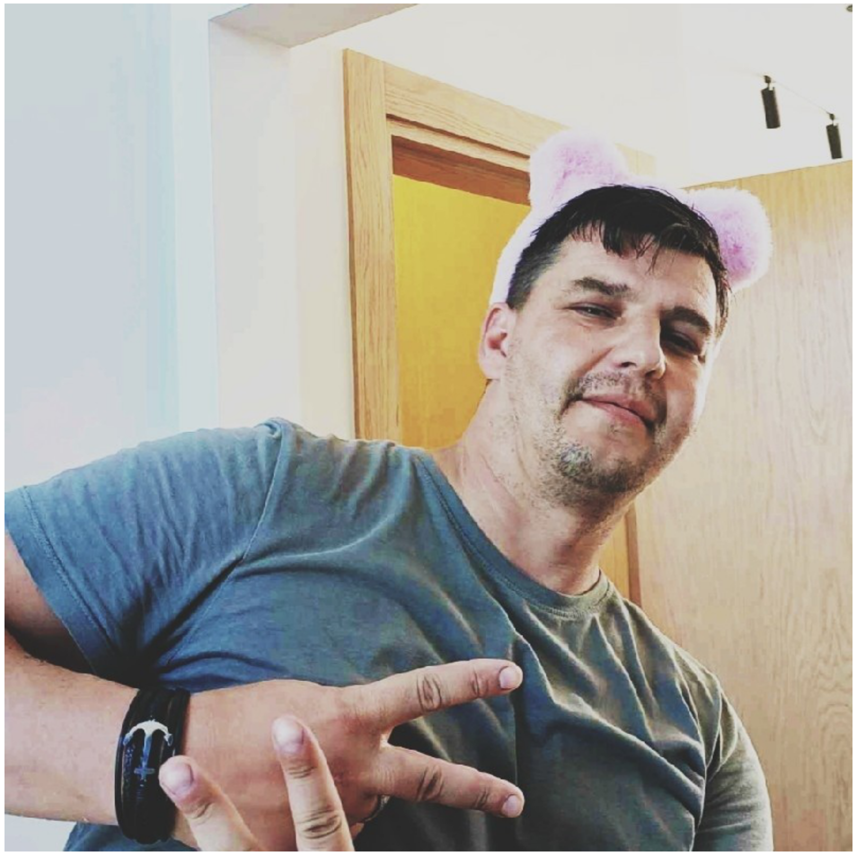
We know why we are here



About presenter

- Java Engineer with a decade of experience in development and architecture.
- Open source contributor to Spring and Apache Camel Frameworks.

- Bouldering enthusiast
- Cycling fan



⌨ https://www.linkedin.com/in/kirilnugmanov/
🖂 kiril.nugmanov@gmail.com

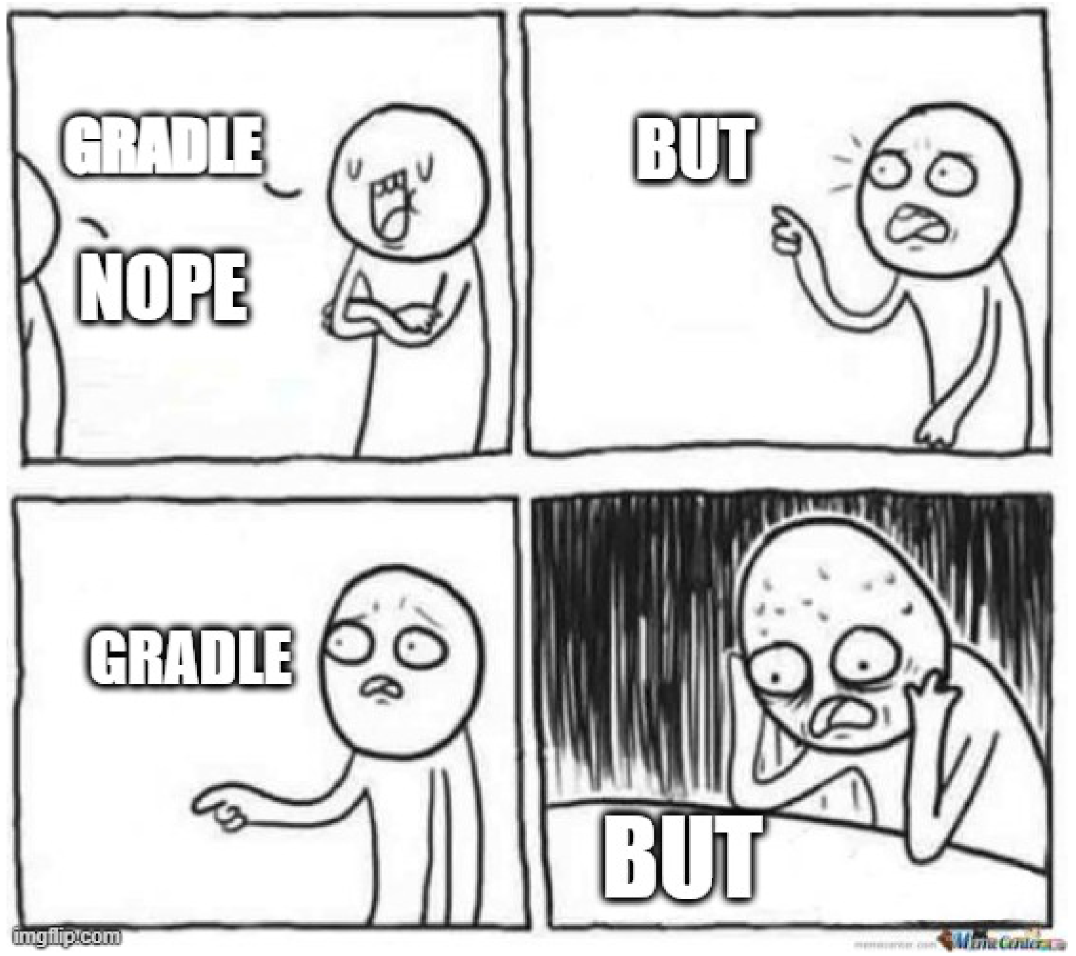
Agenda

- What is Maven
- What is presentation NOT about
- Execution phases, ordering
- Plugin execution configuration
- Disabling execution
- Plugin configuration
- Performance optimization
- Global configurations params
- JVM global configuration
- Bonus

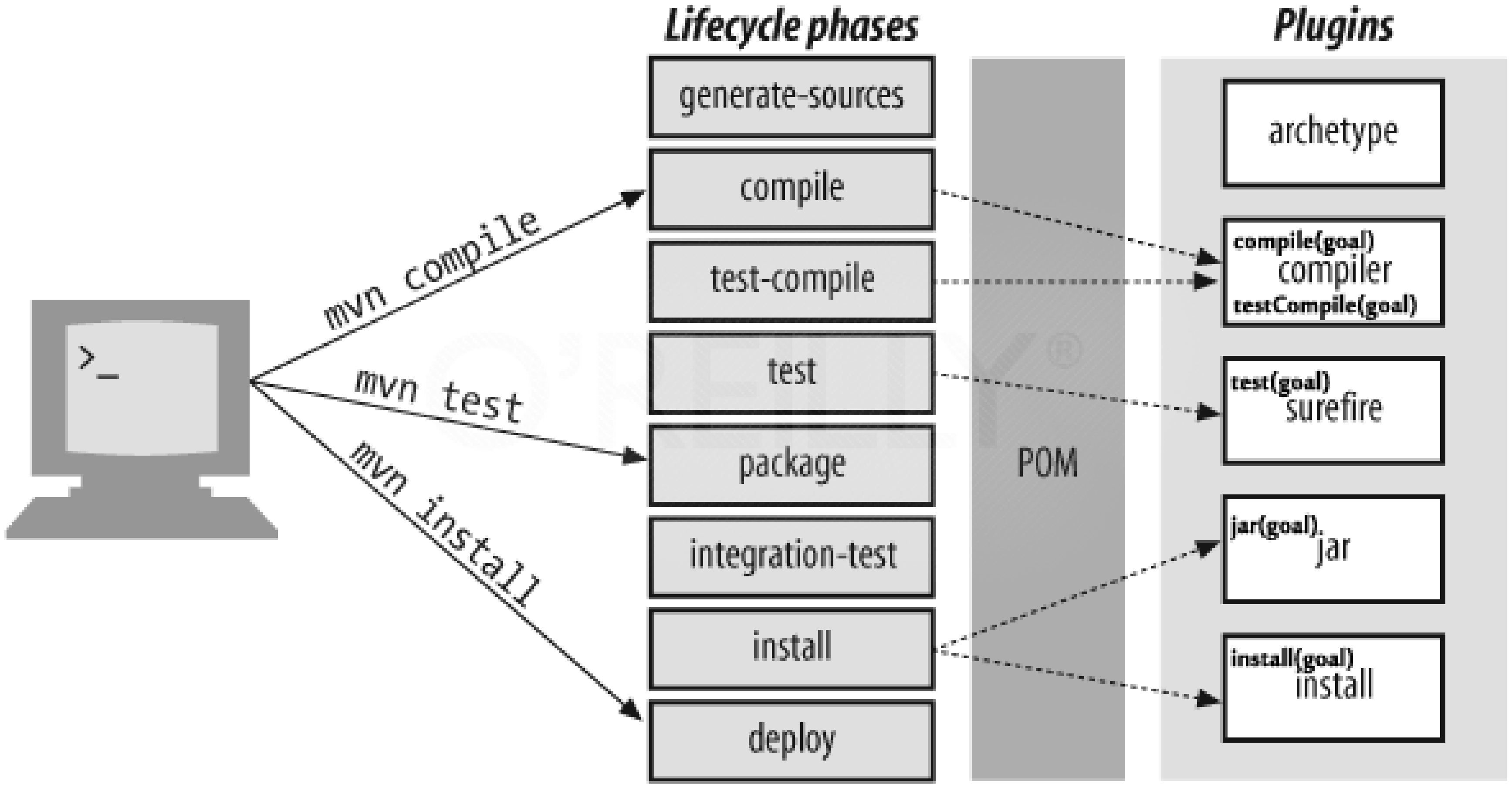
What is Maven

- Maven is a **build automation tool** used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by The Apache Software Foundation, where it was formerly part of the Jakarta Project

What is presentation NOT about



Execution phases, ordering



Execution phases, ordering

- Summary
  - When executing phase - all peviouse phases are **executed in strict order**
  - Developer **can change** execution **phase** of plugin but **not lifecycle**
  - **Parent** defined plugins executed **first**

Plugin execution configuration

- ***<pluginsManagement>*** has **no affect** on execution order
- ***<plugins>*** **defines order** of execution (if phase same)

Plugin execution configuration

```
<build>
  <plugins>
    <plugin>
      <groupId>....</groupId>
      <artifactId>some-plugin</artifactId>
      <version>....</version>
      <executions>
        <execution>
          <id>some-id</id>
          <phase>compile</phase>                    <– Lyfecycle phase of execution
          <goals>
            <goal>plugin-goal</goal>                <– Plugin goal to be executed
          </goals>
        </execution>
      </plugin>
    </plugins>
  </build>
</build>
```

Disabling execution

```
<plugin>
  <groupId>...</groupId>
  <artifactId>some-plugin</artifactId>
  <version>...</version>
  <executions>
    <execution>
      <id>...</id>
      <phase>none</phase>                          <– Define phase as NONE
    </execution>
  </executions>
</plugin>
```

Plugin configuration

```
<build>
   <plugins>
      <plugin>
         <groupId>....</groupId>
         <artifactId>some-plugin</artifactId>
         <version>....</version>
         <configuration XXX=”YYY” >
               …
               …
               ...
         </configuration>
         </plugin>
      </plugins>
   </build>
</build>
```

<– **XXX** attribute may be:

- ***combine.self***
- ***combine.children***

***YYY** value may be:*

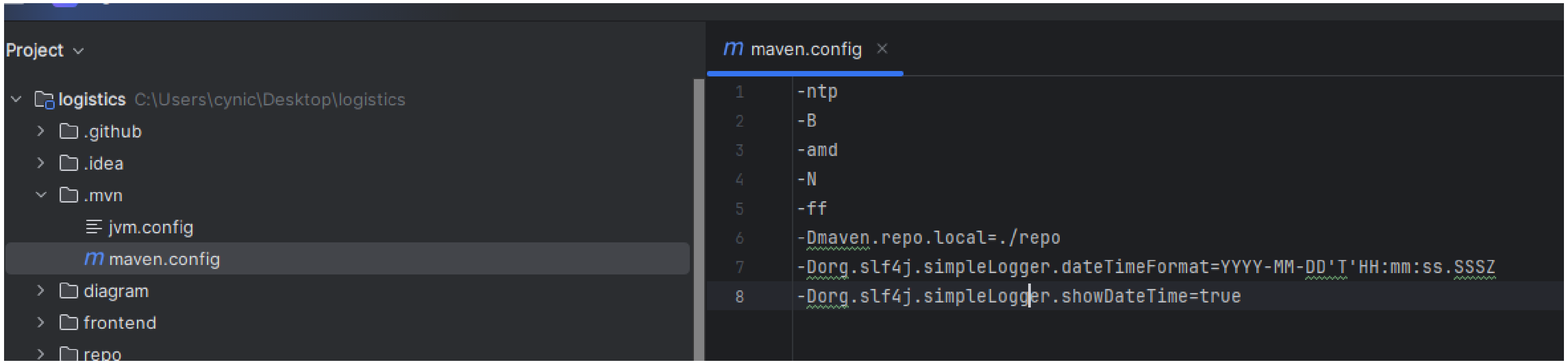
- ***append***
- ***override***

Performance optimization

- Parallel builds: **-T 2C** **//2 threads per Core**
- Disable batch: **-B** **//no prompts**
- Disable terminal prompt: **-ntp** **// no terminal**
- Build dependant projects: **-amd**
- Disable recursive builds: **-N**
- Fail fast: **-ff** **//stops on first test fail**
- Use maven daemon: **mvnd**

Global configurations params

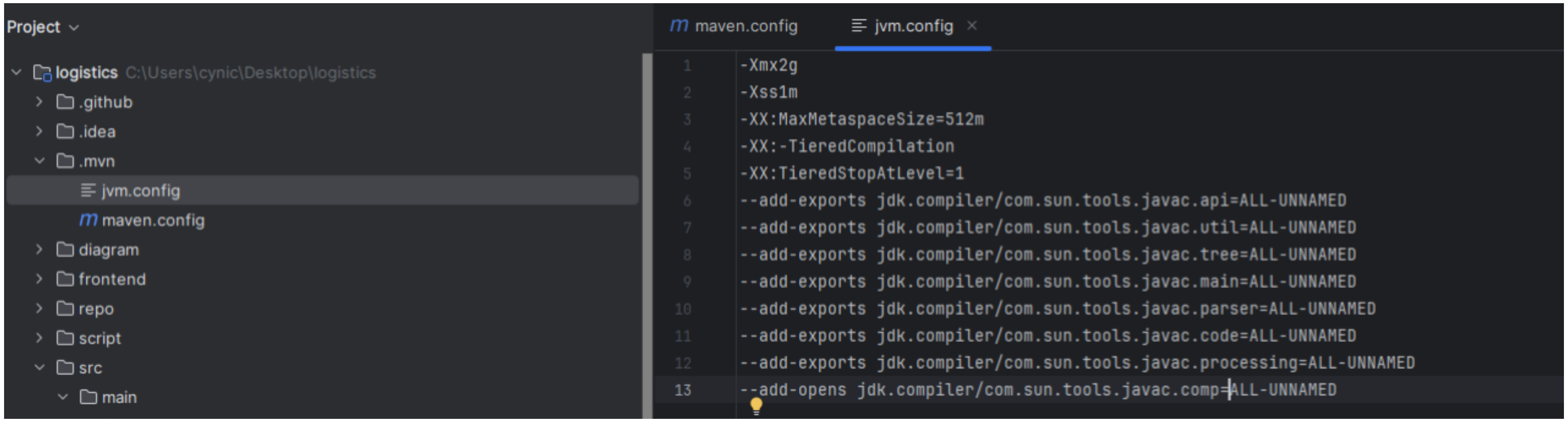
Since Maven 3.6.0: .mvn/maven.config



```
-ntp
-B
-amd
-N
-ff
-Dmaven.repo.local=./repo
-Dorg.slf4j.simpleLogger.dateTimeFormat=YYYY-MM-DD'T'HH:mm:ss.SSSZ
-Dorg.slf4j.simpleLogger.showDateTime=true
```

JVM global configuration

Since Maven 3.8.1: .mvn/jvm.config



```
-Xmx2g
-Xss1m
-XX:MaxMetaspaceSize=512m
-XX:-TieredCompilation
-XX:TieredStopAtLevel=1
--add-exports jdk.compiler/com.sun.tools.javac.api=ALL-UNNAMED
--add-exports jdk.compiler/com.sun.tools.javac.util=ALL-UNNAMED
--add-exports jdk.compiler/com.sun.tools.javac.tree=ALL-UNNAMED
--add-exports jdk.compiler/com.sun.tools.javac.main=ALL-UNNAMED
--add-exports jdk.compiler/com.sun.tools.javac.parser=ALL-UNNAMED
--add-exports jdk.compiler/com.sun.tools.javac.code=ALL-UNNAMED
--add-exports jdk.compiler/com.sun.tools.javac.processing=ALL-UNNAMED
--add-opens jdk.compiler/com.sun.tools.javac.comp=ALL-UNNAMED
```

JVM global configuration

- Did You know that **-opens** and **-exports** can be defined **not only** via JVM **params**

- Define attributes in **Manifest file** (JDK 9+):

  *Add-Exports: <module>/<package>(<module>/<package>)**
  *Add-Opens: <module>/<package>(<module>/<package>)**

https://openjdk.org/jeps/261

Bonus: usefull plugins

- **editorconfig-maven-plugin** - file format and collection of text editor plugins for maintaining **consistent coding styles** between different editors and IDEs.

Bonus: usefull plugins

- **hibernate-enhance-maven-plugin** - Hibernate will **enhance the classes** in an application's domain model in order to add one or more of the following capabilities:
  - **Lazy** state **initialization**
  - **Dirtiness tracking**
  - Automatic bi-directional **association management**
  - Performance optimizations

Bonus: usefull plugins

- **spell-check-maven-plugin** - **Spell checking** of existing code base
- **gitlog-maven-plugin** - **change log generation** based on commit messages

Bonus: usefull plugins

- **maven-pmd-plugin**, **forbiddenapis** - static code analysis in pipelines. **NO stinky code shall pass!**
- **pitest-maven** - Mutational testing of created test. **NO mocking geenfields!**
- **dependency-check-maven** - Dependency **check against OWASP** and othe CVE lists

Bonus: test execution

```
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
        <groups>unit</groups>                    <– Actualy is a @Tag("unit") on tests
        …
        …
    </configuration>
</plugin>
```

```
<plugin>
    <artifactId>maven-failsafe-plugin</artifactId>
    <configuration>
        <groups>it</groups>                      <– Actualy is a @Tag("it") on tests
        …
        …
    </configuration>
</plugin>
```

Bonus: local repository

- To define **local repository** (as in JS projects) use following maven parameter:

```
-Dmaven.repo.local=./repo
```

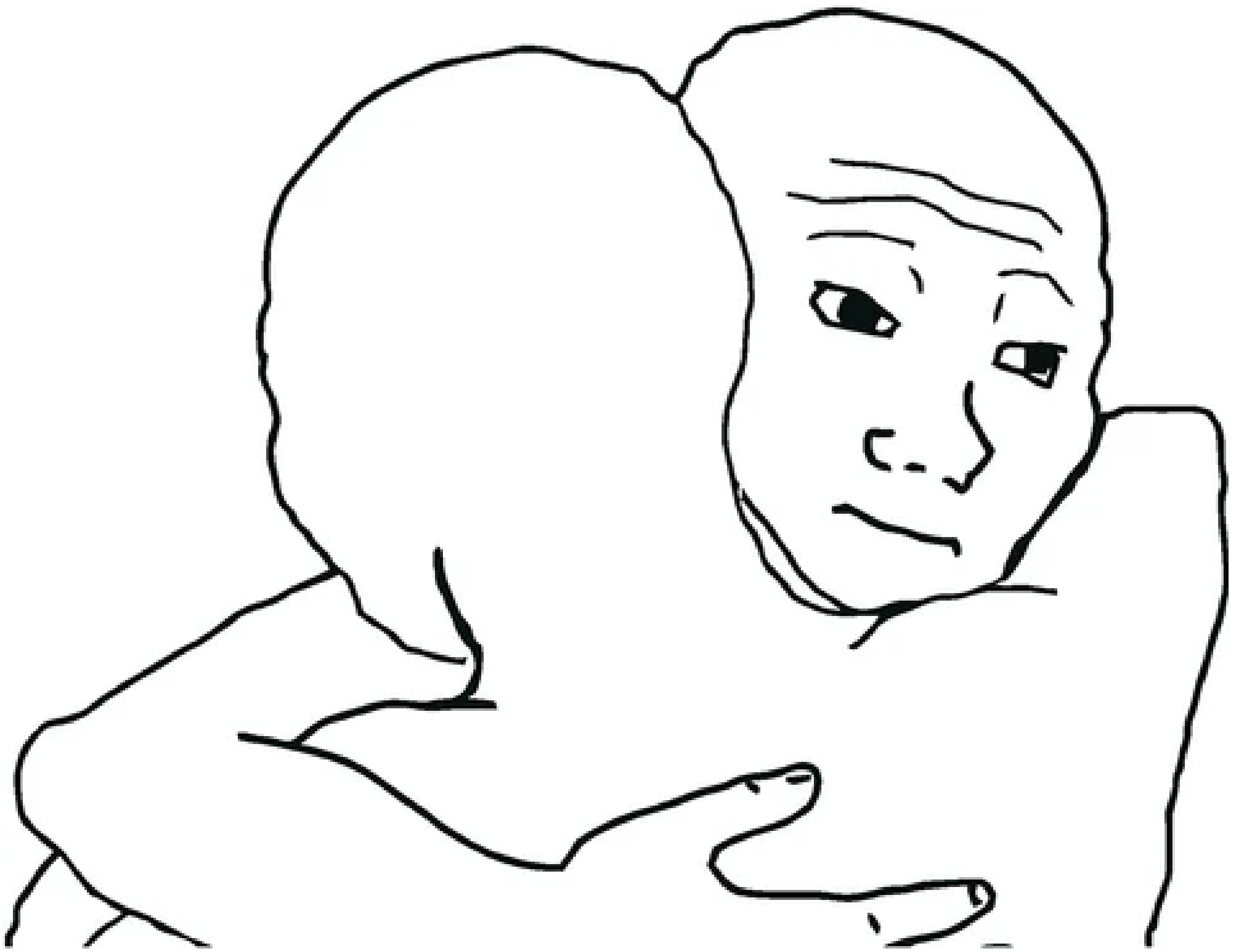
Bonus: execution timestamp

- To **see timestamps** on each line of building process use following parameter:

*-Dorg.slf4j.simpleLogger.dateTimeFormat=**<FMT>***
-Dorg.slf4j.simpleLogger.showDateTime=true

<FMT> - format of timestamp like *YYYY-MM-DD'T'HH:mm:ss.SSSZ* or *HH:mm:ss*

TL;DR; Too much, too long...



Questions/answers/discussion