All about dependencies Maven Puzzlers Dependency resolution Ed

@ SG Virtual Conference 2023

Ixchel Ruiz & Andres Almiray

The Liquid Software Company

Maven Puzzlers

@ SG Virtual Conference 2023

Ixchel Ruiz & Andres Almiray

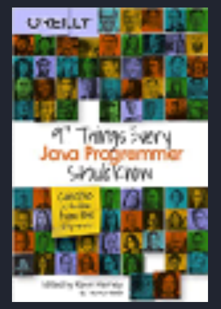
The Liquid Software Company

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

CLOUD NATIVE
COMPUTING FOUNDATION

HACKERGARTEN

O'REILLY
DevOps Tools for Java Developers


O'REILLY
97 Things Every Java Programmer Should Know

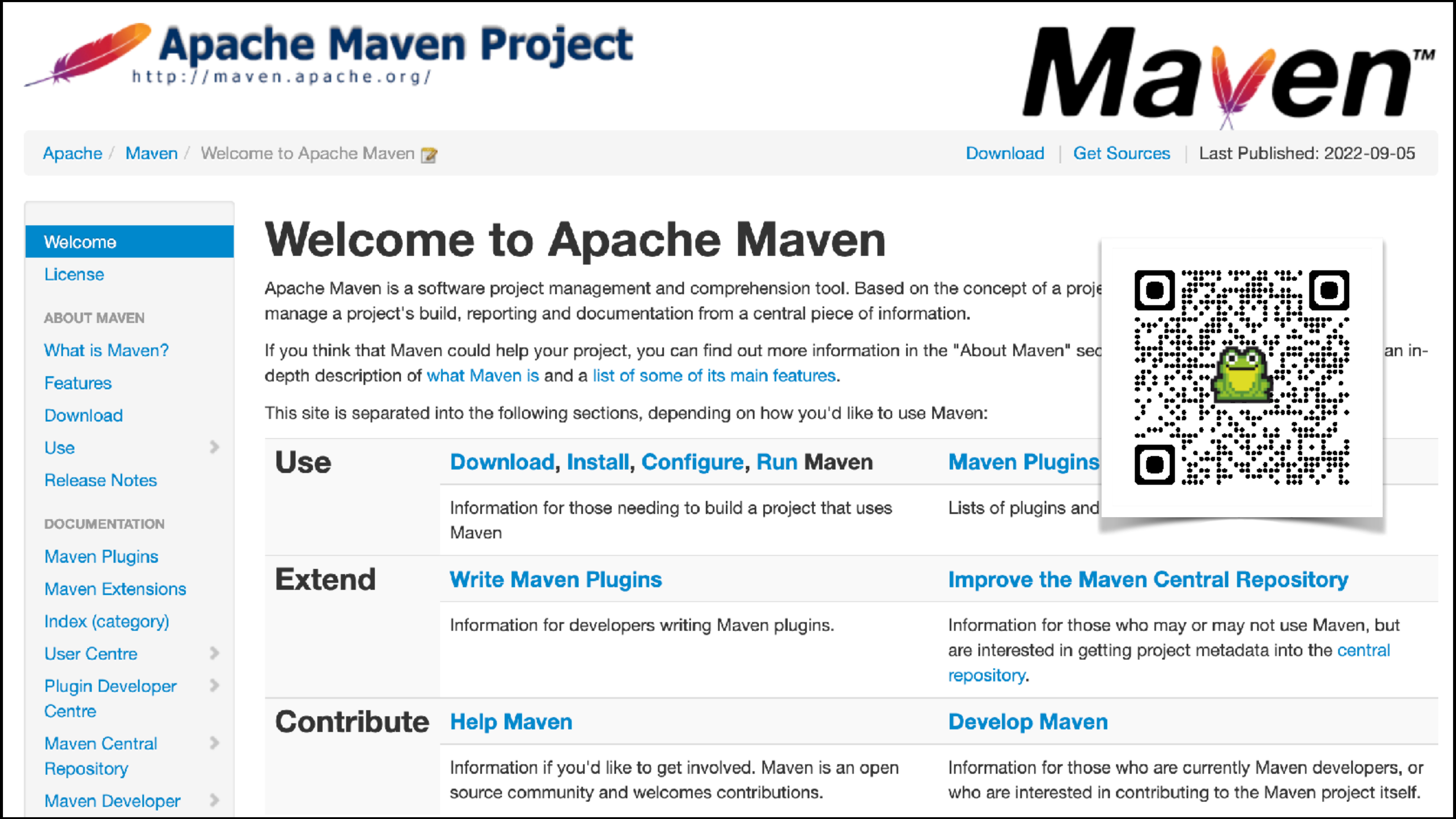
JFrog

JFrog The Liquid Software Company

maven.apache.org

Maven™

The Liquid Software Company



Apache Maven Project
http://maven.apache.org/

Maven™

Apache / Maven / Welcome to Apache Maven

Download | Get Sources | Last Published: 2022-09-05

Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a proje
manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information in the "About Maven" sec an in-depth description of what Maven is and a list of some of its main features.

This site is separated into the following sections, depending on how you'd like to use Maven:

Use	**Download, Install, Configure, Run Maven**	**Maven Plugins**
	Information for those needing to build a project that uses Maven	Lists of plugins and
Extend	**Write Maven Plugins**	**Improve the Maven Central Repository**
	Information for developers writing Maven plugins.	Information for those who may or may not use Maven, but are interested in getting project metadata into the central repository.
Contribute	**Help Maven**	**Develop Maven**
	Information if you'd like to get involved. Maven is an open source community and welcomes contributions.	Information for those who are currently Maven developers, or who are interested in contributing to the Maven project itself.

The
Liquid
Software
Company

Dependencies

Not all are the same.

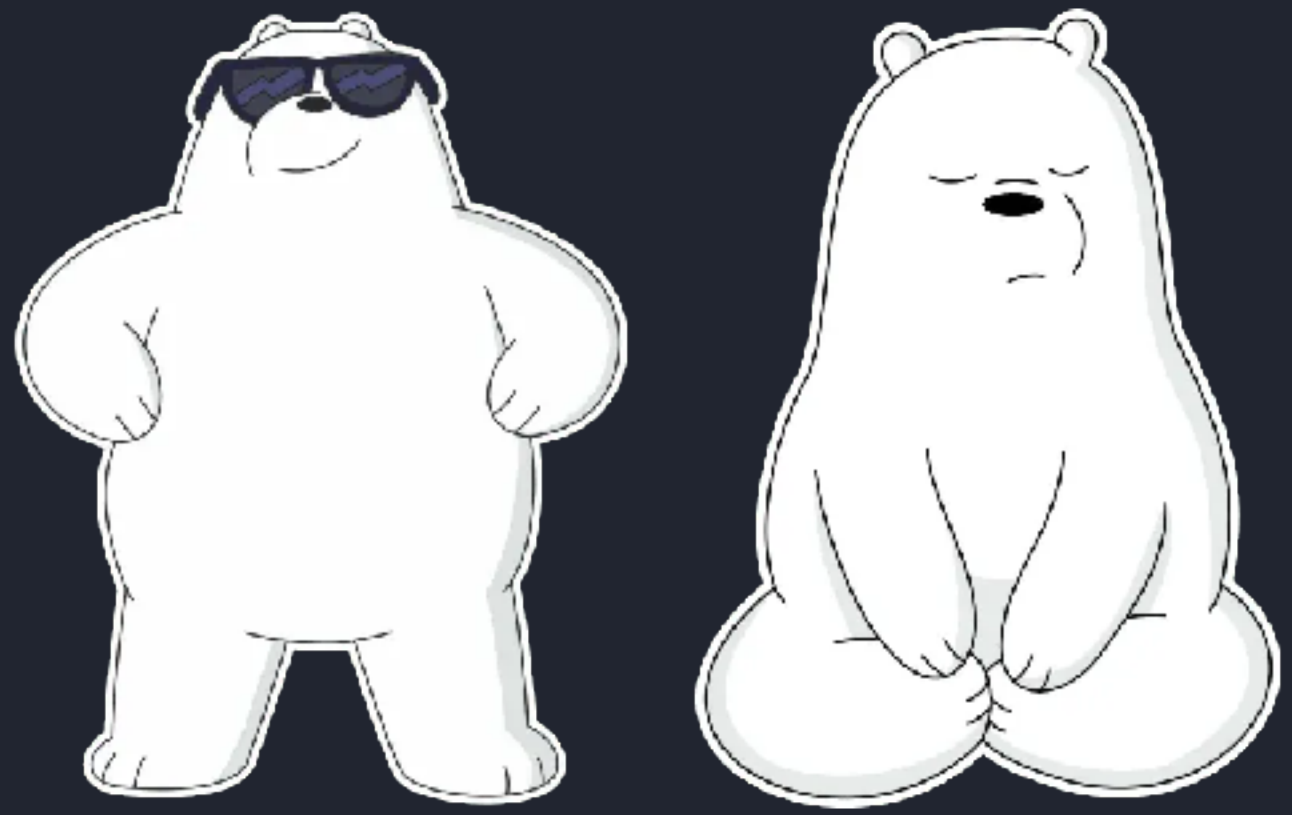
JFrog The Liquid Software Company

The
Liquid
Software
Company

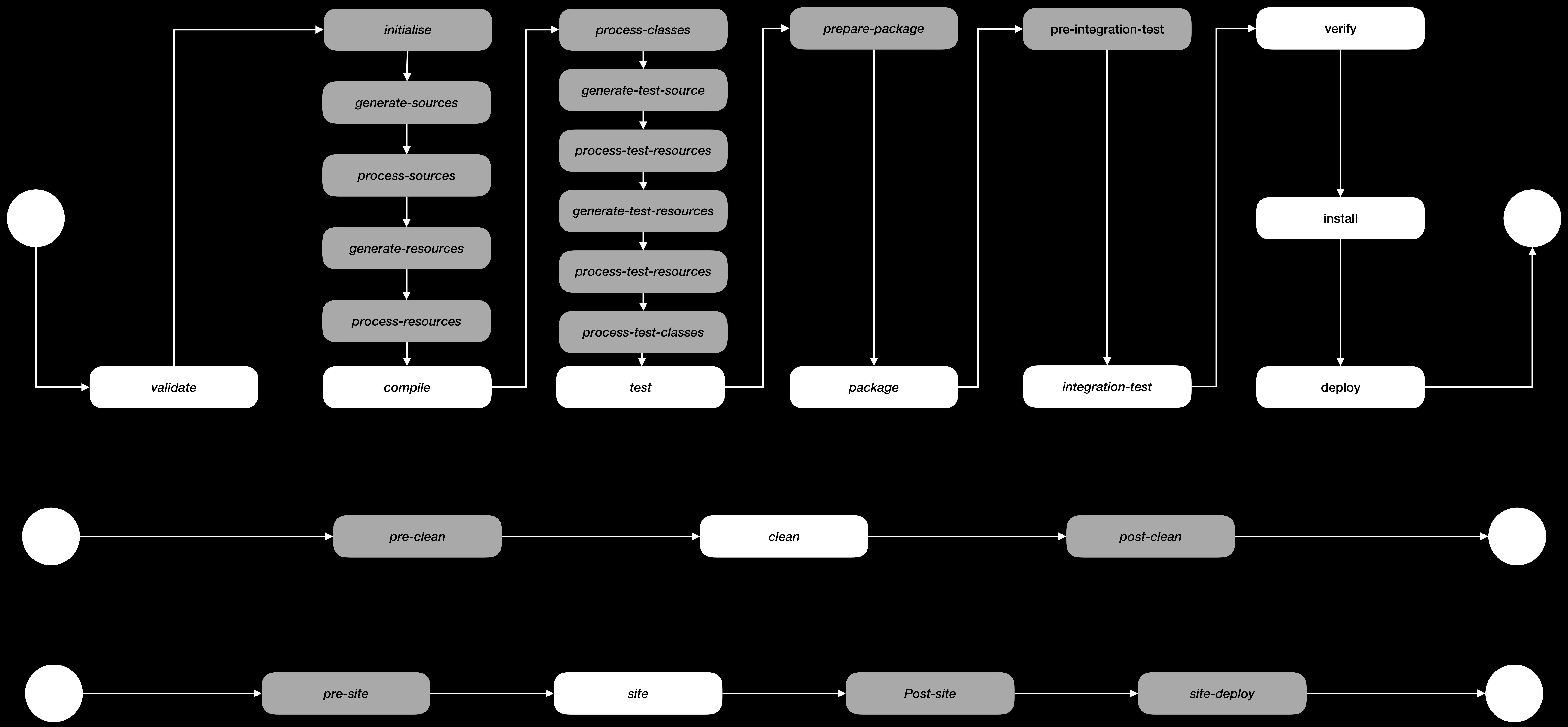
The
Liquid
Software
Company

mvn clean install

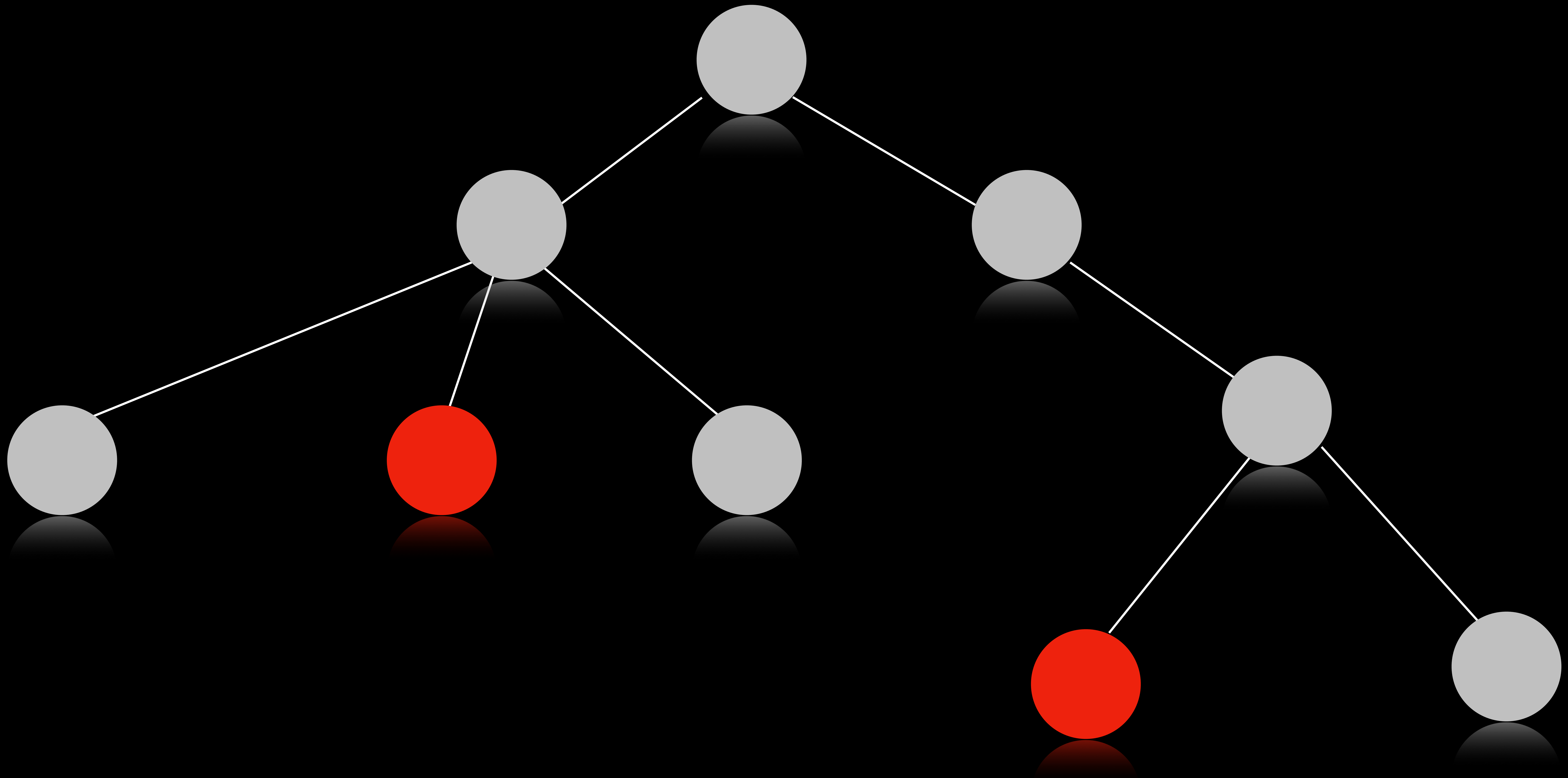
mvn verify

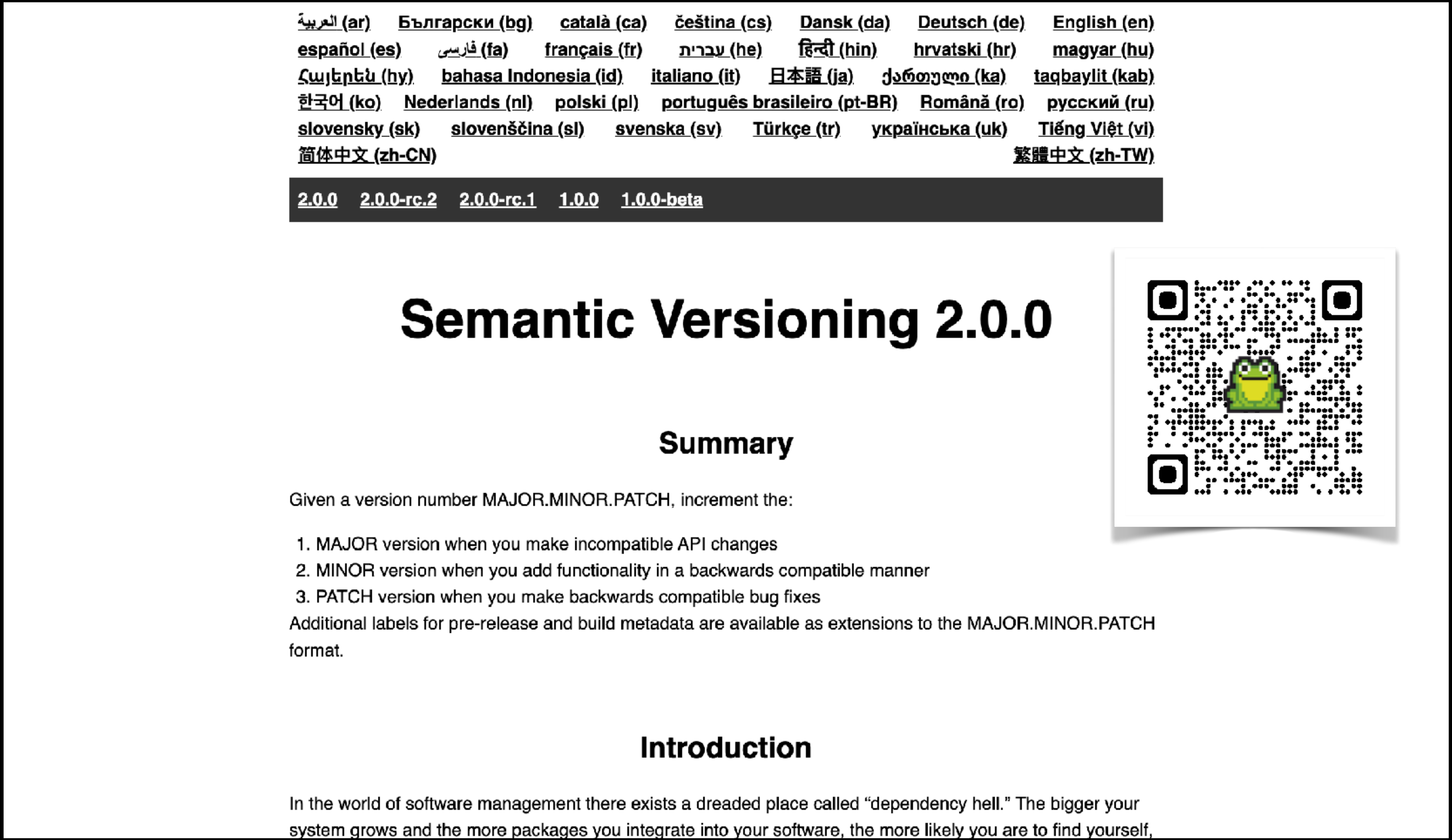


The Liquid Software Company



JFrog The Liquid Software Company





العربية (ar) Български (bg) català (ca) čeština (cs) Dansk (da) Deutsch (de) English (en)
español (es) فارسی (fa) français (fr) עברית (he) हिन्दी (hin) hrvatski (hr) magyar (hu)
Հայերեն (hy) bahasa Indonesia (id) italiano (it) 日本語 (ja) ქართული (ka) taqbaylit (kab)
한국어 (ko) Nederlands (nl) polski (pl) português brasileiro (pt-BR) Română (ro) русский (ru)
slovensky (sk) slovenščina (sl) svenska (sv) Türkçe (tr) українська (uk) Tiếng Việt (vi)
简体中文 (zh-CN) 繁體中文 (zh-TW)

2.0.0 2.0.0-rc.2 2.0.0-rc.1 1.0.0 1.0.0-beta

Semantic Versioning 2.0.0

Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

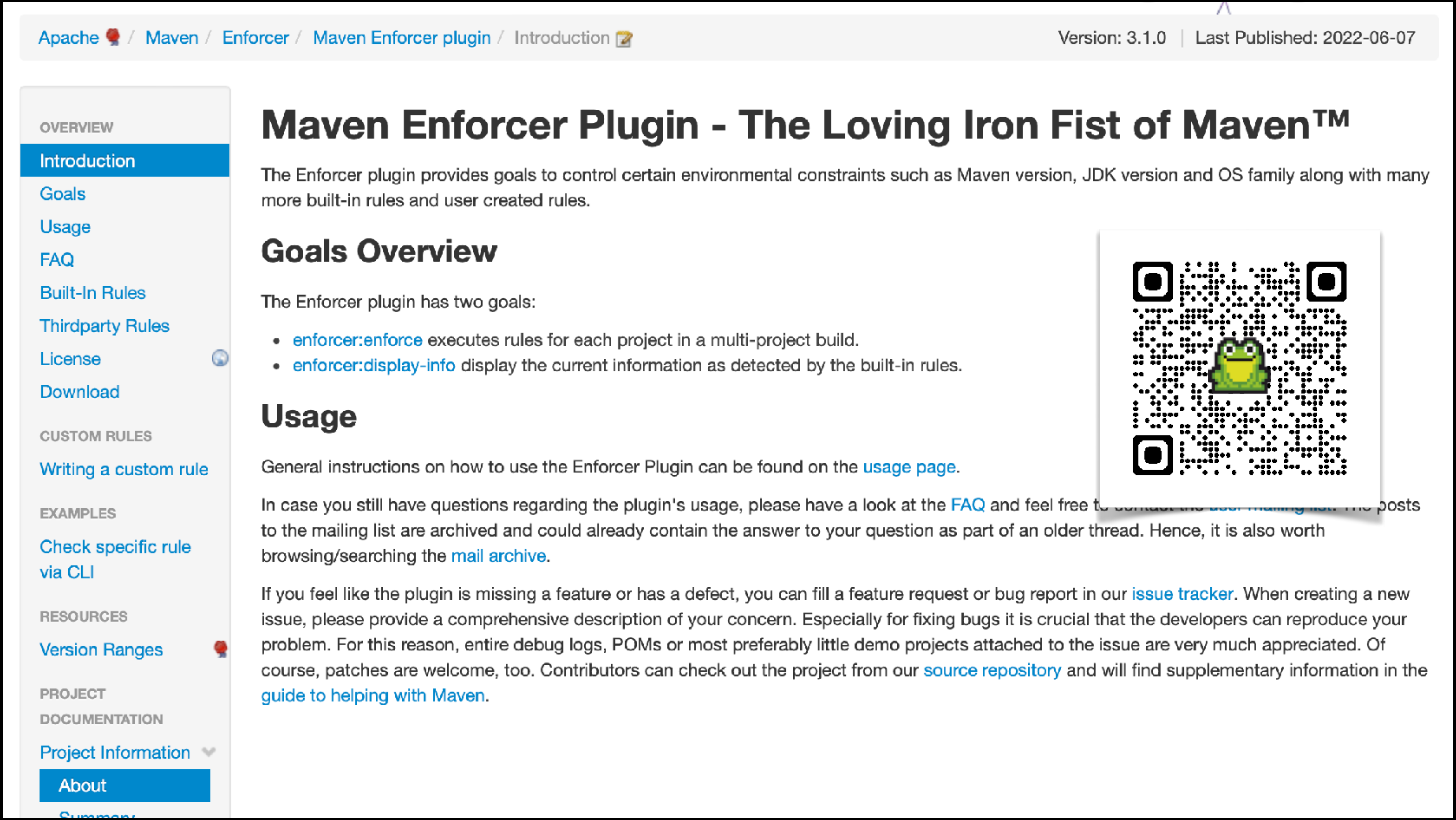
1. MAJOR version when you make incompatible API changes
2. MINOR version when you add functionality in a backwards compatible manner
3. PATCH version when you make backwards compatible bug fixes

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Introduction

In the world of software management there exists a dreaded place called “dependency hell.” The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself,

Max input: 230 V



Maven Enforcer Plugin - The Loving Iron Fist of Maven™

The Enforcer plugin provides goals to control certain environmental constraints such as Maven version, JDK version and OS family along with many more built-in rules and user created rules.

Goals Overview

The Enforcer plugin has two goals:

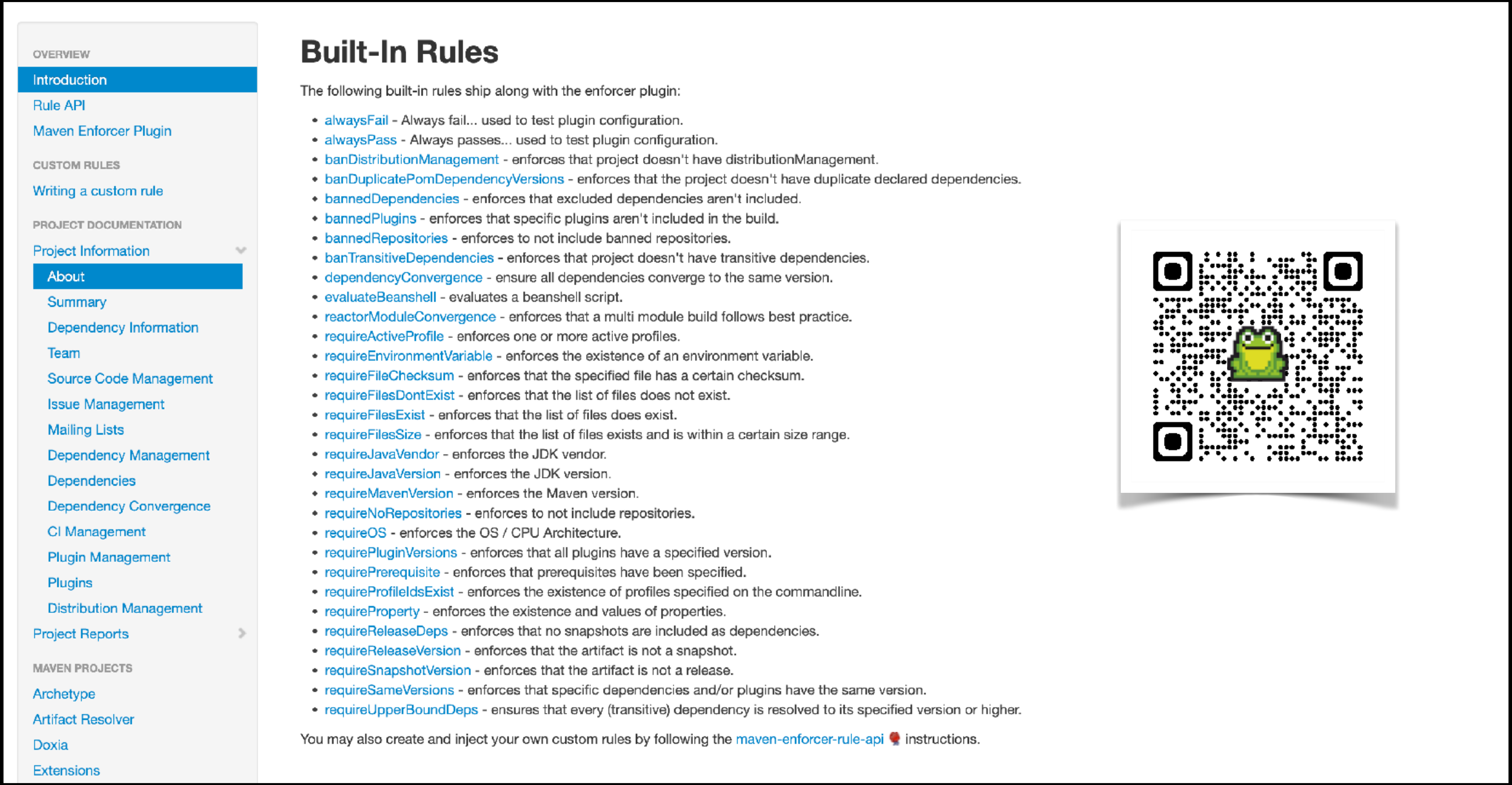
- enforcer:enforce executes rules for each project in a multi-project build.
- enforcer:display-info display the current information as detected by the built-in rules.

Usage

General instructions on how to use the Enforcer Plugin can be found on the usage page.

In case you still have questions regarding the plugin's usage, please have a look at the FAQ and feel free t posts to the mailing list are archived and could already contain the answer to your question as part of an older thread. Hence, it is also worth browsing/searching the mail archive.

If you feel like the plugin is missing a feature or has a defect, you can fill a feature request or bug report in our issue tracker. When creating a new issue, please provide a comprehensive description of your concern. Especially for fixing bugs it is crucial that the developers can reproduce your problem. For this reason, entire debug logs, POMs or most preferably little demo projects attached to the issue are very much appreciated. Of course, patches are welcome, too. Contributors can check out the project from our source repository and will find supplementary information in the guide to helping with Maven.



OVERVIEW

- Introduction
- Rule API
- Maven Enforcer Plugin

CUSTOM RULES

- Writing a custom rule

PROJECT DOCUMENTATION

- Project Information
  - About
  - Summary
  - Dependency Information
  - Team
  - Source Code Management
  - Issue Management
  - Mailing Lists
  - Dependency Management
  - Dependencies
  - Dependency Convergence
  - CI Management
  - Plugin Management
  - Plugins
  - Distribution Management
- Project Reports

MAVEN PROJECTS

- Archetype
- Artifact Resolver
- Doxia
- Extensions

Built-In Rules

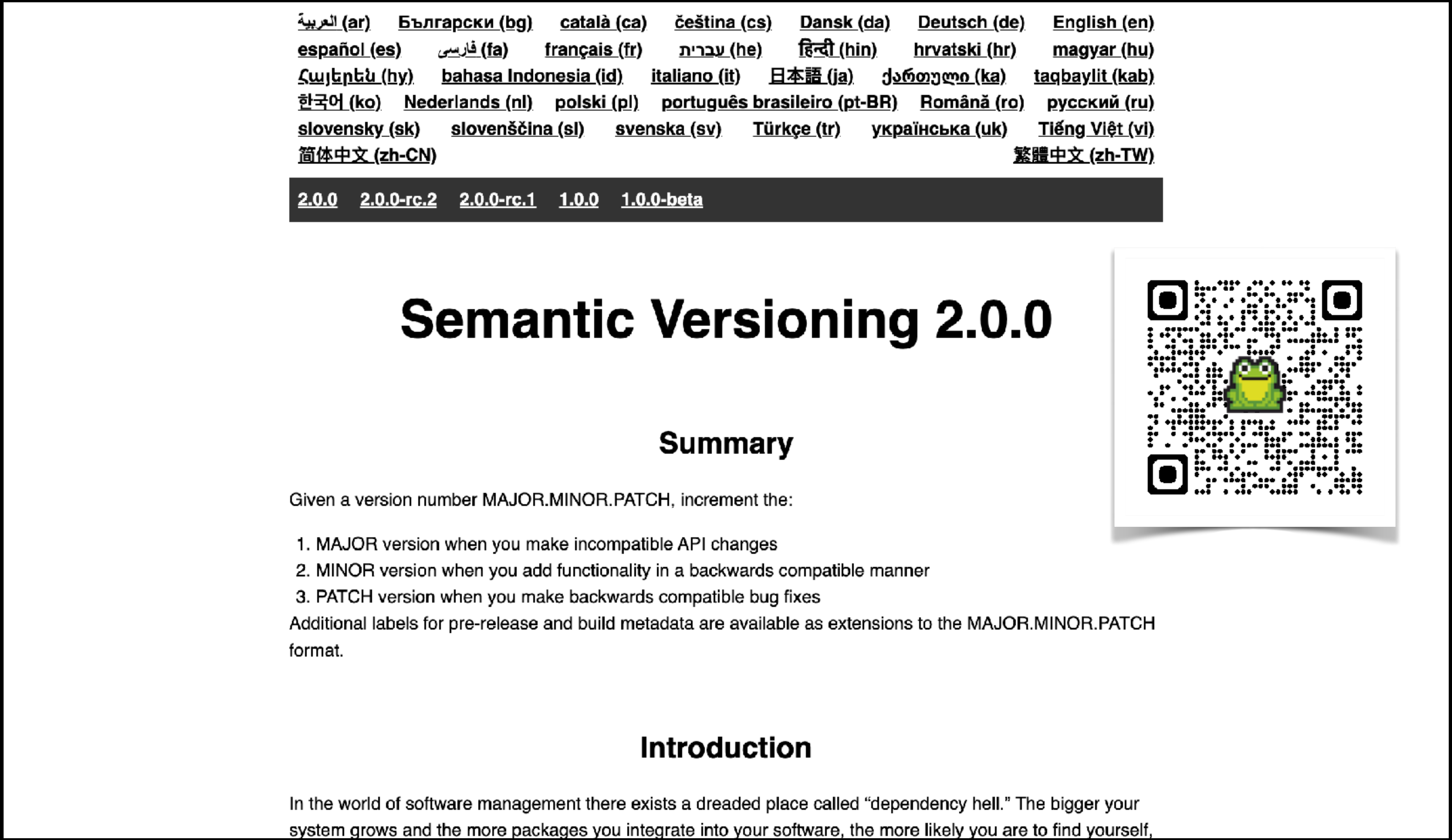
The following built-in rules ship along with the enforcer plugin:

- alwaysFail - Always fail... used to test plugin configuration.
- alwaysPass - Always passes... used to test plugin configuration.
- banDistributionManagement - enforces that project doesn't have distributionManagement.
- banDuplicatePomDependencyVersions - enforces that the project doesn't have duplicate declared dependencies.
- bannedDependencies - enforces that excluded dependencies aren't included.
- bannedPlugins - enforces that specific plugins aren't included in the build.
- bannedRepositories - enforces to not include banned repositories.
- banTransitiveDependencies - enforces that project doesn't have transitive dependencies.
- dependencyConvergence - ensure all dependencies converge to the same version.
- evaluateBeanshell - evaluates a beanshell script.
- reactorModuleConvergence - enforces that a multi module build follows best practice.
- requireActiveProfile - enforces one or more active profiles.
- requireEnvironmentVariable - enforces the existence of an environment variable.
- requireFileChecksum - enforces that the specified file has a certain checksum.
- requireFilesDontExist - enforces that the list of files does not exist.
- requireFilesExist - enforces that the list of files does exist.
- requireFilesSize - enforces that the list of files exists and is within a certain size range.
- requireJavaVendor - enforces the JDK vendor.
- requireJavaVersion - enforces the JDK version.
- requireMavenVersion - enforces the Maven version.
- requireNoRepositories - enforces to not include repositories.
- requireOS - enforces the OS / CPU Architecture.
- requirePluginVersions - enforces that all plugins have a specified version.
- requirePrerequisite - enforces that prerequisites have been specified.
- requireProfileIdsExist - enforces the existence of profiles specified on the commandline.
- requireProperty - enforces the existence and values of properties.
- requireReleaseDeps - enforces that no snapshots are included as dependencies.
- requireReleaseVersion - enforces that the artifact is not a snapshot.
- requireSnapshotVersion - enforces that the artifact is not a release.
- requireSameVersions - enforces that specific dependencies and/or plugins have the same version.
- requireUpperBoundDeps - ensures that every (transitive) dependency is resolved to its specified version or higher.

You may also create and inject your own custom rules by following the maven-enforcer-rule-api instructions.

```xml
    <build>
        <plugins>
            <plugin>
              <groupId>org.apache.maven.plugins</groupId>
              <artifactId>maven-enforcer-plugin</artifactId>
              <version>3.1.0</version>
              <executions>
                  <execution>
                      <id>check</id>
                      <phase>initialize</phase>
                      <goals>
                          <goal>enforce</goal>
                      </goals>
                      <configuration>
                          <rules>
                              <banDuplicatePomDependencyVersions/>
                          </rules>
                      </configuration>
                  </execution>
              </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

```
[INFO] --- maven-enforcer-plugin:3.1.0:enforce (check) @ enforcer-01 ---
[ERROR] Rule 0: org.apache.maven.plugins.enforcer.BanDuplicatePomDependencyVersions failed with message:
Found 1 duplicate dependency declaration in this project:
 - dependencies.dependency[com.google.guava:guava:jar] ( 2 times )
```



العربية (ar) Български (bg) català (ca) čeština (cs) Dansk (da) Deutsch (de) English (en) español (es) فارسی (fa) français (fr) עברית (he) हिन्दी (hin) hrvatski (hr) magyar (hu) Հայերեն (hy) bahasa Indonesia (id) italiano (it) 日本語 (ja) ქართული (ka) taqbaylit (kab) 한국어 (ko) Nederlands (nl) polski (pl) português brasileiro (pt-BR) Română (ro) русский (ru) slovensky (sk) slovenščina (sl) svenska (sv) Türkçe (tr) українська (uk) Tiếng Việt (vi) 简体中文 (zh-CN) 繁體中文 (zh-TW)

2.0.0 2.0.0-rc.2 2.0.0-rc.1 1.0.0 1.0.0-beta

Semantic Versioning 2.0.0

Summary

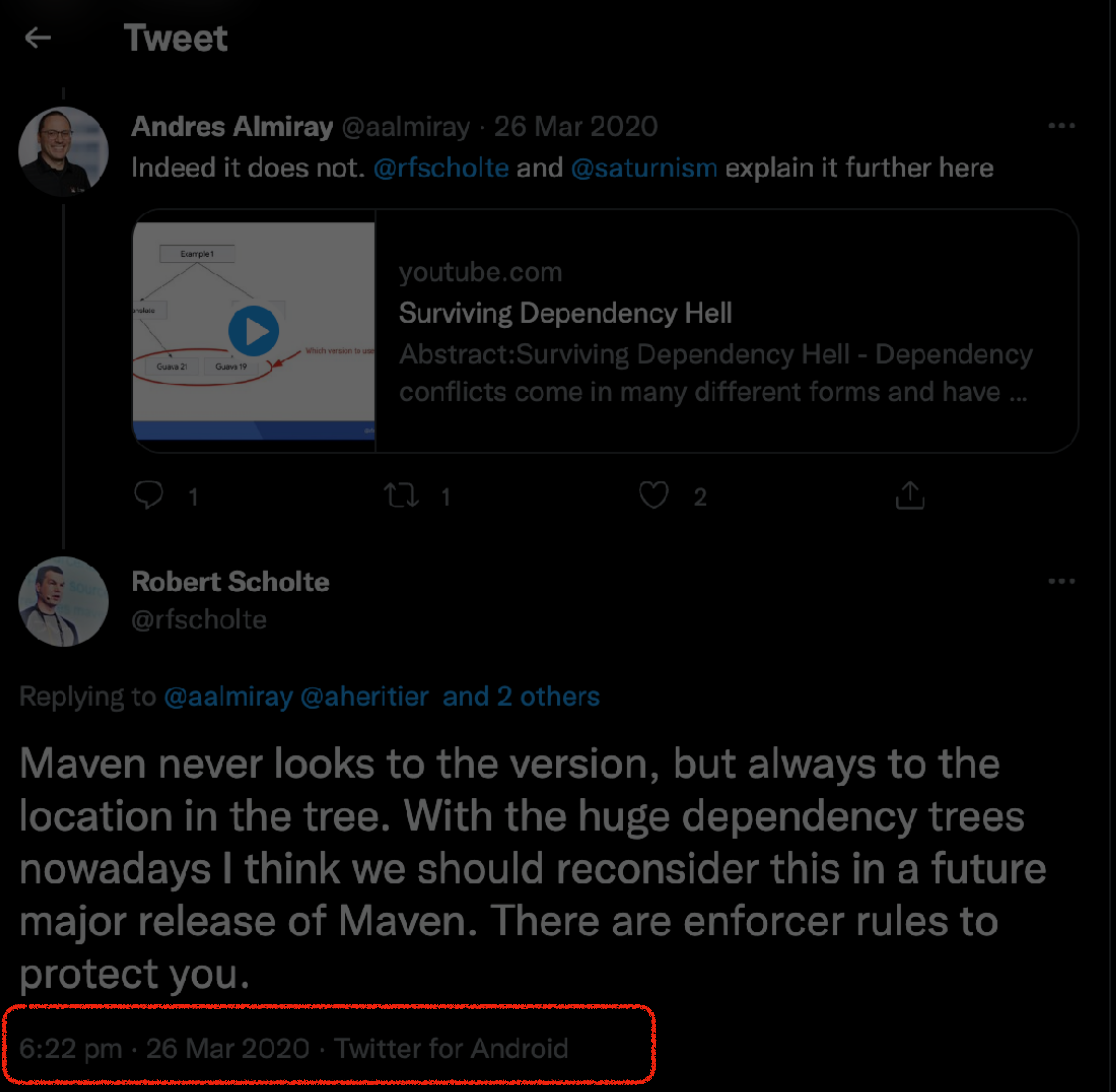
Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes
2. MINOR version when you add functionality in a backwards compatible manner
3. PATCH version when you make backwards compatible bug fixes

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Introduction

In the world of software management there exists a dreaded place called “dependency hell.” The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself,



Tweet

Andres Almiray @aalmiray · 26 Mar 2020

Indeed it does not. @rfscholte and @saturnism explain it further here

youtube.com

Surviving Dependency Hell

Abstract:Surviving Dependency Hell - Dependency conflicts come in many different forms and have ...

1 1 2

Robert Scholte
@rfscholte

Replying to @aalmiray @aheritier and 2 others

Maven never looks to the version, but always to the location in the tree. With the huge dependency trees nowadays I think we should reconsider this in a future major release of Maven. There are enforcer rules to protect you.

6:22 pm · 26 Mar 2020 · Twitter for Android

The Liquid Software Company

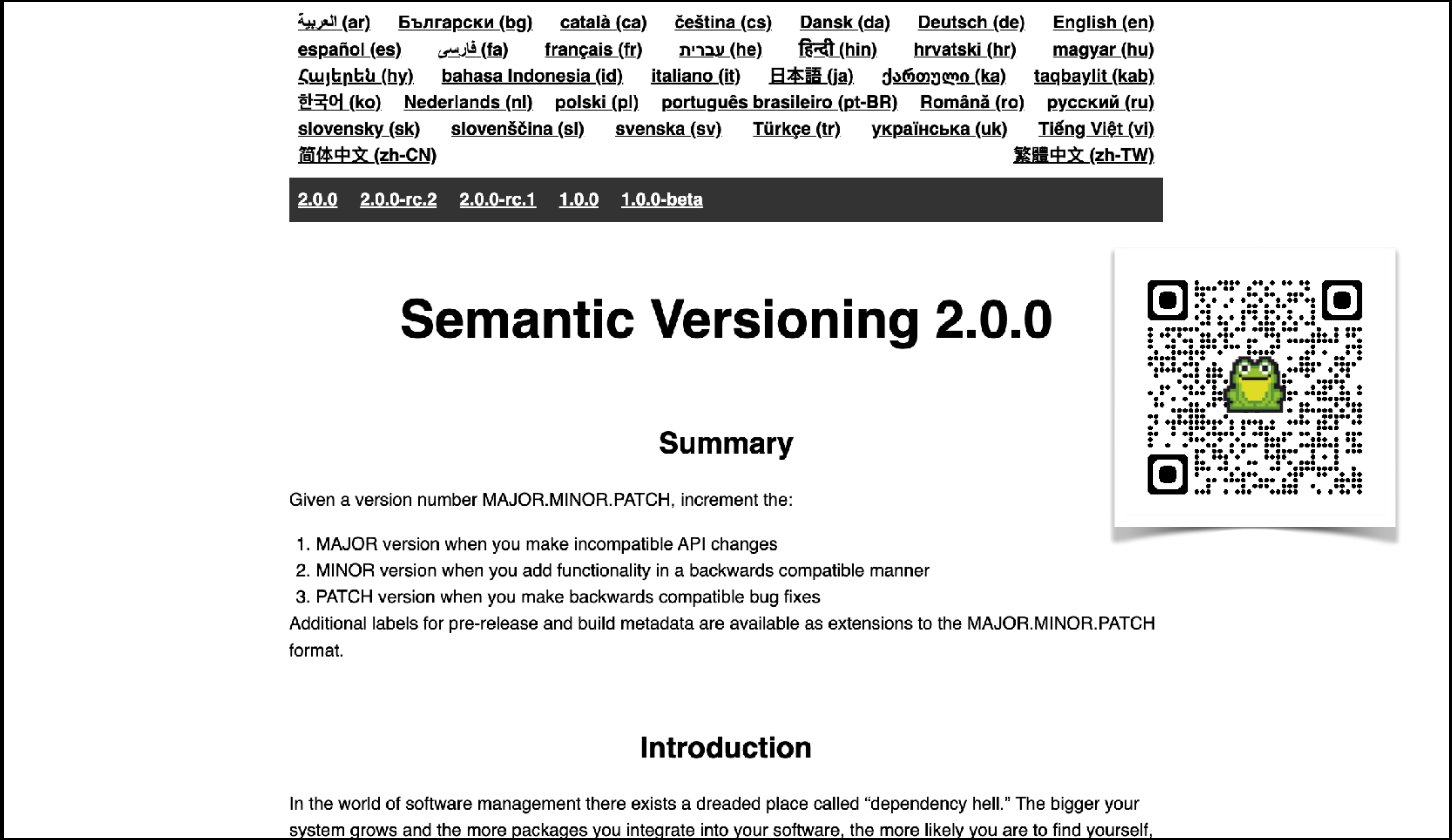
```xml
  <build>
    <plugins>
      <plugin>
       <groupId>org.apache.maven.plugins</groupId>
       <artifactId>maven-enforcer-plugin</artifactId>
       <version>3.1.0</version>
       <executions>
         <execution>
           <id>check</id>
           <phase>initialize</phase>
           <goals>
             <goal>enforce</goal>
           </goals>
           <configuration>
             <rules>
               <dependencyConvergence/>
             </rules>
           </configuration>
         </execution>
       </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

```
[INFO] --- maven-enforcer-plugin:3.1.0:enforce (check) @ enforcer-02 ---
[WARNING]
Dependency convergence error for org.checkerframework:checker-qual:jar:2.5.2:compile paths to dependency are:
+-com.acme:enforcer-02:jar:0.0.0-SNAPSHOT
  +-com.google.truth:truth:jar:1.1.3:compile
    +-com.google.guava:guava:jar:27.0-jre:compile
      +-org.checkerframework:checker-qual:jar:2.5.2:compile
and
+-com.acme:enforcer-02:jar:0.0.0-SNAPSHOT
  +-com.google.truth:truth:jar:1.1.3:compile
    +-org.checkerframework:checker-qual:jar:3.13.0:compile

[WARNING]
Dependency convergence error for com.google.errorprone:error_prone_annotations:jar:2.2.0:compile paths to dependency are:
+-com.acme:enforcer-02:jar:0.0.0-SNAPSHOT
  +-com.google.truth:truth:jar:1.1.3:compile
    +-com.google.guava:guava:jar:27.0-jre:compile
      +-com.google.errorprone:error_prone_annotations:jar:2.2.0:compile
and
+-com.acme:enforcer-02:jar:0.0.0-SNAPSHOT
  +-com.google.truth:truth:jar:1.1.3:compile
    +-com.google.errorprone:error_prone_annotations:jar:2.7.1:compile

[ERROR] Rule 0: org.apache.maven.plugins.enforcer.DependencyConvergence failed with message:
Failed while enforcing releasability. See above detailed error message.
```

```xml
    <build>
        <plugins>
            <plugin>
              <groupId>org.apache.maven.plugins</groupId>
              <artifactId>maven-enforcer-plugin</artifactId>
              <version>3.1.0</version>
              <executions>
                  <execution>
                      <id>check</id>
                      <phase>initialize</phase>
                      <goals>
                          <goal>enforce</goal>
                      </goals>
                      <configuration>
                          <rules>
                              <requireUpperBoundDeps />
                          </rules>
                      </configuration>
                  </execution>
              </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

```
[INFO] --- maven-enforcer-plugin:3.1.0:enforce (check) @ enforcer-03 ---
[ERROR] Rule 0: org.apache.maven.plugins.enforcer.RequireUpperBoundDeps failed with message:
Failed while enforcing RequireUpperBoundDeps. The error(s) are [
Require upper bound dependencies error for com.google.guava:guava:27.0-jre paths to dependency are:
+-com.acme:enforcer-03:0.0.0-SNAPSHOT
  +-com.google.truth:truth:1.1.3
    +-com.google.guava:guava:27.0-jre (managed) <-- com.google.guava:guava:30.1.1-android
]
```

The
Liquid
Software
Company



العربية (ar) Български (bg) català (ca) čeština (cs) Dansk (da) Deutsch (de) English (en) español (es) فارسی (fa) français (fr) עברית (he) हिन्दी (hin) hrvatski (hr) magyar (hu) Հայերեն (hy) bahasa Indonesia (id) italiano (it) 日本語 (ja) ქართული (ka) taqbaylit (kab) 한국어 (ko) Nederlands (nl) polski (pl) português brasileiro (pt-BR) Română (ro) русский (ru) slovensky (sk) slovenščina (sl) svenska (sv) Türkçe (tr) українська (uk) Tiếng Việt (vi) 简体中文 (zh-CN) 繁體中文 (zh-TW)

2.0.0 2.0.0-rc.2 2.0.0-rc.1 1.0.0 1.0.0-beta

Semantic Versioning 2.0.0

Summary

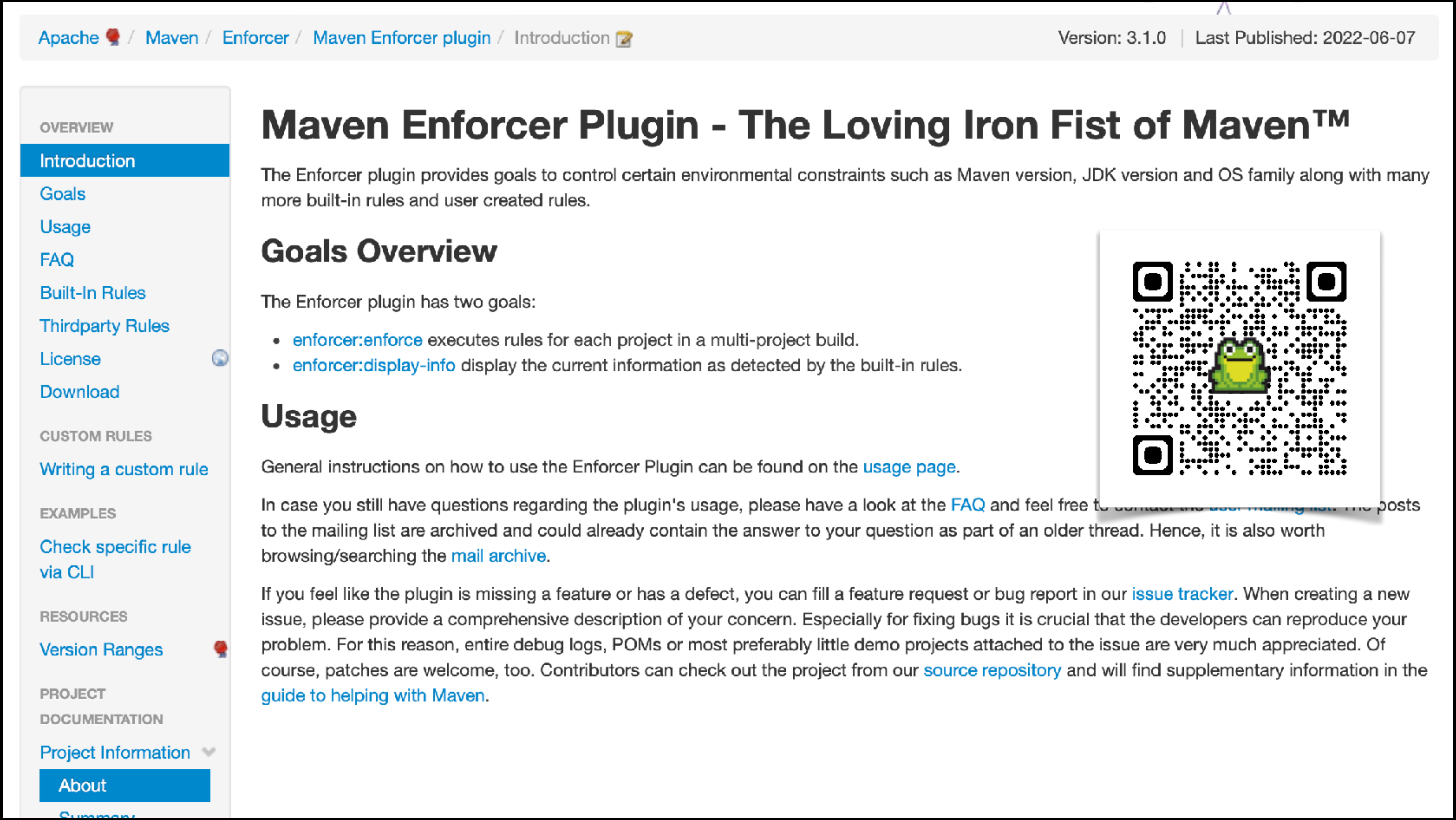
Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes
2. MINOR version when you add functionality in a backwards compatible manner
3. PATCH version when you make backwards compatible bug fixes

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Introduction

In the world of software management there exists a dreaded place called “dependency hell.” The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself,



Apache / Maven / Enforcer / Maven Enforcer plugin / Introduction

Version: 3.1.0 | Last Published: 2022-06-07

OVERVIEW
- Introduction
- Goals
- Usage
- FAQ
- Built-In Rules
- Thirdparty Rules
- License
- Download

CUSTOM RULES
- Writing a custom rule

EXAMPLES
- Check specific rule via CLI

RESOURCES
- Version Ranges

PROJECT DOCUMENTATION
- Project Information
  - About

Maven Enforcer Plugin - The Loving Iron Fist of Maven™

The Enforcer plugin provides goals to control certain environmental constraints such as Maven version, JDK version and OS family along with many more built-in rules and user created rules.

Goals Overview

The Enforcer plugin has two goals:

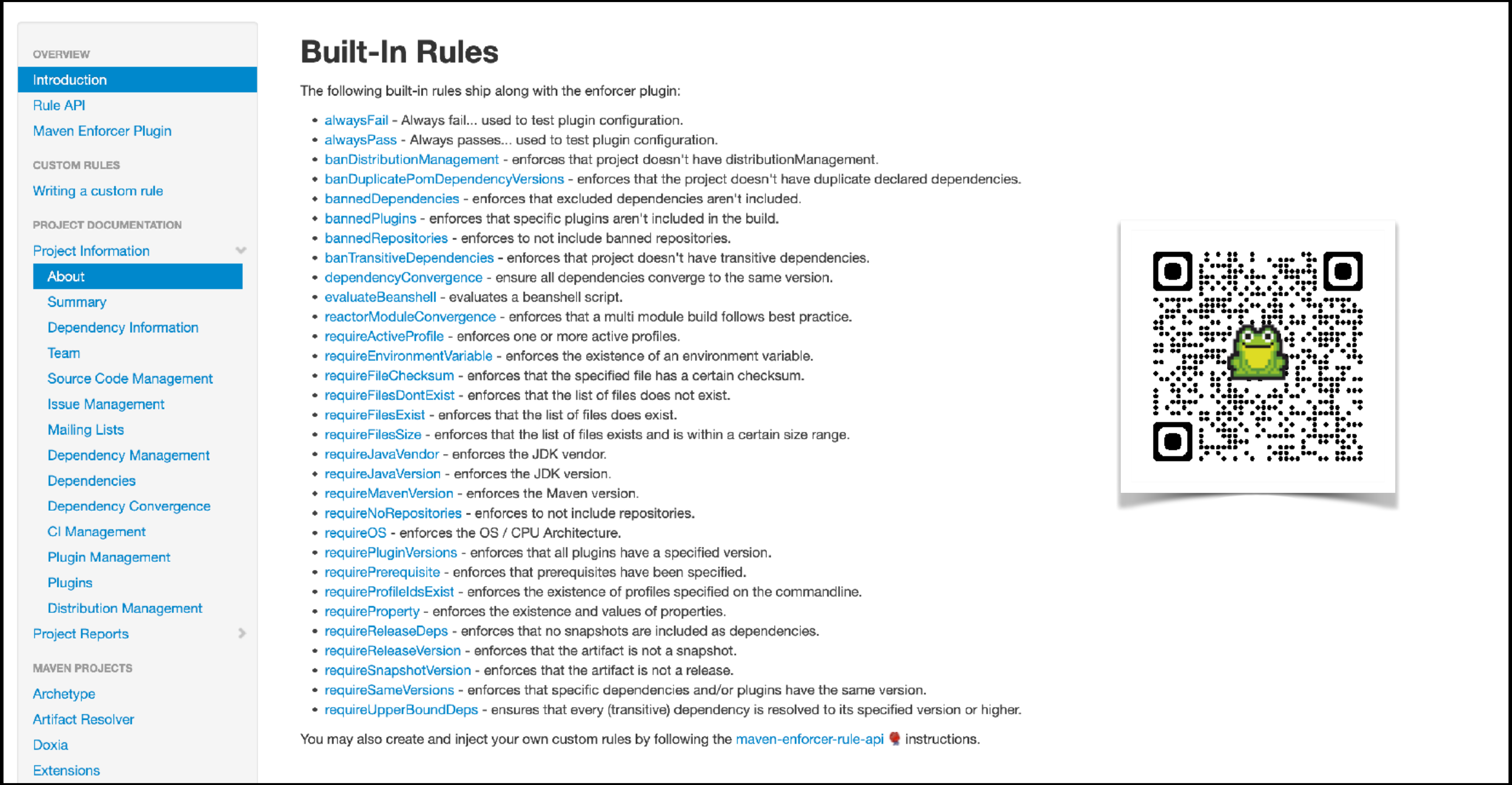
- enforcer:enforce executes rules for each project in a multi-project build.
- enforcer:display-info display the current information as detected by the built-in rules.

Usage

General instructions on how to use the Enforcer Plugin can be found on the usage page.

In case you still have questions regarding the plugin's usage, please have a look at the FAQ and feel free t[illegible] posts to the mailing list are archived and could already contain the answer to your question as part of an older thread. Hence, it is also worth browsing/searching the mail archive.

If you feel like the plugin is missing a feature or has a defect, you can fill a feature request or bug report in our issue tracker. When creating a new issue, please provide a comprehensive description of your concern. Especially for fixing bugs it is crucial that the developers can reproduce your problem. For this reason, entire debug logs, POMs or most preferably little demo projects attached to the issue are very much appreciated. Of course, patches are welcome, too. Contributors can check out the project from our source repository and will find supplementary information in the guide to helping with Maven.



OVERVIEW

Introduction

Rule API

Maven Enforcer Plugin

CUSTOM RULES

Writing a custom rule

PROJECT DOCUMENTATION

Project Information

About

Summary

Dependency Information

Team

Source Code Management

Issue Management

Mailing Lists

Dependency Management

Dependencies

Dependency Convergence

CI Management

Plugin Management

Plugins

Distribution Management

Project Reports

MAVEN PROJECTS

Archetype

Artifact Resolver

Doxia

Extensions

Built-In Rules

The following built-in rules ship along with the enforcer plugin:

- alwaysFail - Always fail... used to test plugin configuration.
- alwaysPass - Always passes... used to test plugin configuration.
- banDistributionManagement - enforces that project doesn't have distributionManagement.
- banDuplicatePomDependencyVersions - enforces that the project doesn't have duplicate declared dependencies.
- bannedDependencies - enforces that excluded dependencies aren't included.
- bannedPlugins - enforces that specific plugins aren't included in the build.
- bannedRepositories - enforces to not include banned repositories.
- banTransitiveDependencies - enforces that project doesn't have transitive dependencies.
- dependencyConvergence - ensure all dependencies converge to the same version.
- evaluateBeanshell - evaluates a beanshell script.
- reactorModuleConvergence - enforces that a multi module build follows best practice.
- requireActiveProfile - enforces one or more active profiles.
- requireEnvironmentVariable - enforces the existence of an environment variable.
- requireFileChecksum - enforces that the specified file has a certain checksum.
- requireFilesDontExist - enforces that the list of files does not exist.
- requireFilesExist - enforces that the list of files does exist.
- requireFilesSize - enforces that the list of files exists and is within a certain size range.
- requireJavaVendor - enforces the JDK vendor.
- requireJavaVersion - enforces the JDK version.
- requireMavenVersion - enforces the Maven version.
- requireNoRepositories - enforces to not include repositories.
- requireOS - enforces the OS / CPU Architecture.
- requirePluginVersions - enforces that all plugins have a specified version.
- requirePrerequisite - enforces that prerequisites have been specified.
- requireProfileIdsExist - enforces the existence of profiles specified on the commandline.
- requireProperty - enforces the existence and values of properties.
- requireReleaseDeps - enforces that no snapshots are included as dependencies.
- requireReleaseVersion - enforces that the artifact is not a snapshot.
- requireSnapshotVersion - enforces that the artifact is not a release.
- requireSameVersions - enforces that specific dependencies and/or plugins have the same version.
- requireUpperBoundDeps - ensures that every (transitive) dependency is resolved to its specified version or higher.

You may also create and inject your own custom rules by following the maven-enforcer-rule-api instructions.

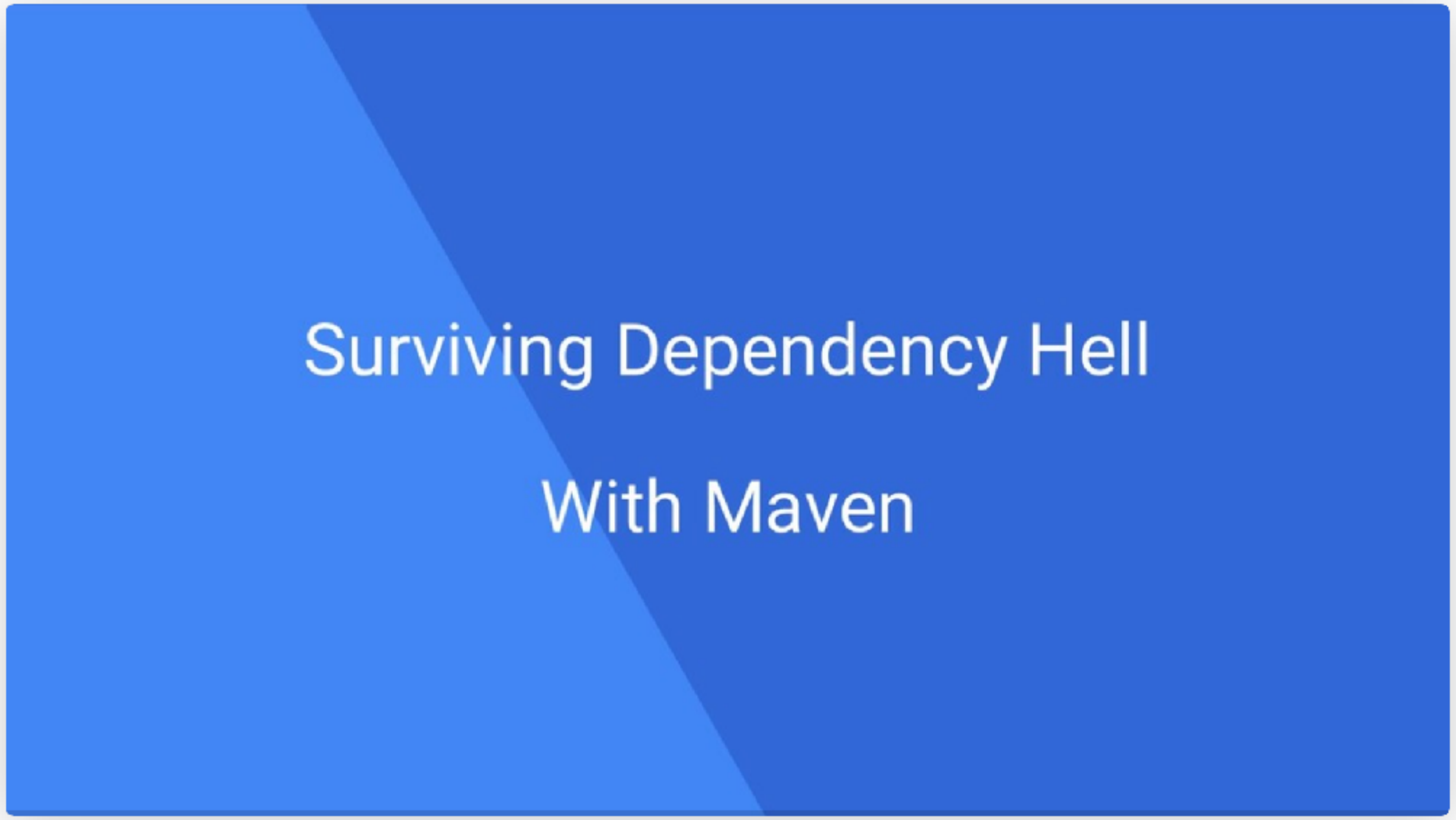
Ray Tsang

Surviving Dependency Hell with Maven

Abstract

As a developer advocate working with customers, Ray has seen all sorts of issues due to dependency conflicts. Dependency conflicts come in many different forms and have different impacts on your applications. This presentation examines common causes of a dependency conflict, how you can mitigate it as a library developer, and how end users can resolve it. It also covers what Google has been documenting in terms of best practices and what tools it has created to help, based on its learnings.

Slides



Surviving Dependency Hell

With Maven

Videos

JFrog

JFrog
ARTIFACTORY

JFrog
ARTIFACTORY EDGE

JFrog
XRAY

Andres Almiray
Seasoned Sourceror, Oracle

@aalmiray@mastodon.social
www.linkedin.com/in/aalmiray

Ixchel Ruiz
Developer Advocate, JFrog

@ixchelruiz@mastodon.social
www.linkedin.com/in/ixchelruiz

JFrog The Liquid Software Company