Versioning with Maven, NodeJS, and Other Build Tools

This lesson discusses how to implement versioning using other build tools such as Maven and NodeJS.

We'll cover the following

Chances are that your applications are not written in Go. Even if some are, that's probably not the case with all of them. You might have applications written in Java, NodeJS, Python, Ruby, or a myriad of other languages. You might be using **Maven**, **Gradle**, or any other build and packaging tool. *What should you do in that case?*

Versioning using Maven

If you are using Maven, everything we explored so far is applicable with a difference that the version is defined in `pom.xml` instead of `Makefile`. Just as we added `VERSION := 1.0.0` to `Makefile`, you'd need to add `<version>` inside `<project>` in `pom.xml`. Everything else stays the same.

Versioning using NodeJS

In the case of NodeJS, `jx-release-version` is not used at all. Instead, pipelines rely on the semantic-release plugin to determine the next version number by fetching information from commit messages.

Using other tools for versioning

What about other build tools? You can still add `Makefile` that would serve only for versioning, or you can control major and minor version increments by creating dummy Git tags. We explored both, so you should have no problem adopting one of those. Or you might choose to adopt versioning through a plugin in your

favorite build tool (e.g., Gradle) or to roll out your own. Finally, you might want to contribute to jx-release-version by adding support for your favorite build tool.

That would be the best option. After all, it is open-source, and it relies on the community of volunteers.

We will wrap up the discussion regarding versioning releases in the next lesson.