Working with Advanced Packaging Problems

Chris B. Behrens
Software Architect

@chrisbbehrens

Versioning and Version Control

If your versions are just labels in version control...

You're on solid ground

Versions

1. 1.0.0-pre-release
2. 2.1.9-alpha.1
3. 3.2.0-c9e62a

A pre-release package that was ready for release

What we expect, anyway

You find a defect and fix it

Sometimes, the choice is clear

Sometimes, there are multiple good options

Suppose the latest version of a popular software library is something like 2.14.3. The development team decides to make significant changes to the API in an upcoming version. This update will include breaking changes, so this will be a new major version: 3.0.0.

https://bit.ly/45Q84Z4

They begin by releasing the first draft of the overhaul with major version prerelease, 3.0.0-alpha.1, which includes the initial implementation of the new API. This prerelease serves as an early preview for consumers to experiment with and provide feedback. Based on that feedback, the maintainers can iterate and refine the initial prerelease with subsequent prereleases like 3.0.0-alpha.2, 3.0.0-beta, or 3.0.0-rc (release candidate).

---

https://bit.ly/45Q84Z4

All Bets Are Off

There could be breaking changes

Version 9.9.9

Repository-per-lifecycle-stage

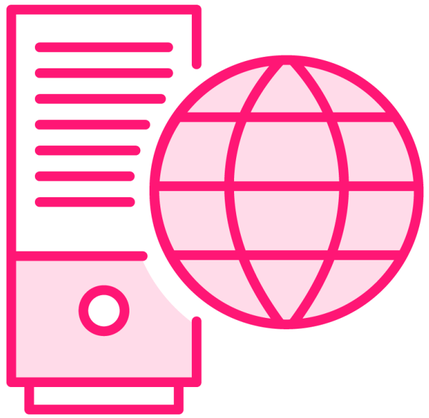
I prefer a single unified feed

Sometimes, pre-release ends in tears

But you can't modify published artifacts

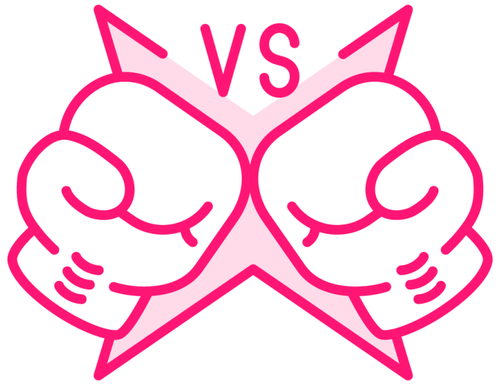
Alternative Version Schemes

Why



Packing web is deprecated

Zip file is the way



Not "Don't talk about SemVer"

Software using Semantic Versioning MUST declare a public API. This API could be declared in the code itself or exist strictly in documentation. However, it is done, it SHOULD be precise and comprehensive.

Then...Why Did We Version the Package?

Mostly because I didn't want to get derailed into this discussion

But also, to tie the builds together

Use the build number

Globomantics.Artifactory.1709

Globomantics.Artifactory-42813e

Some Jenkins plugins have moved to this

Recommendations

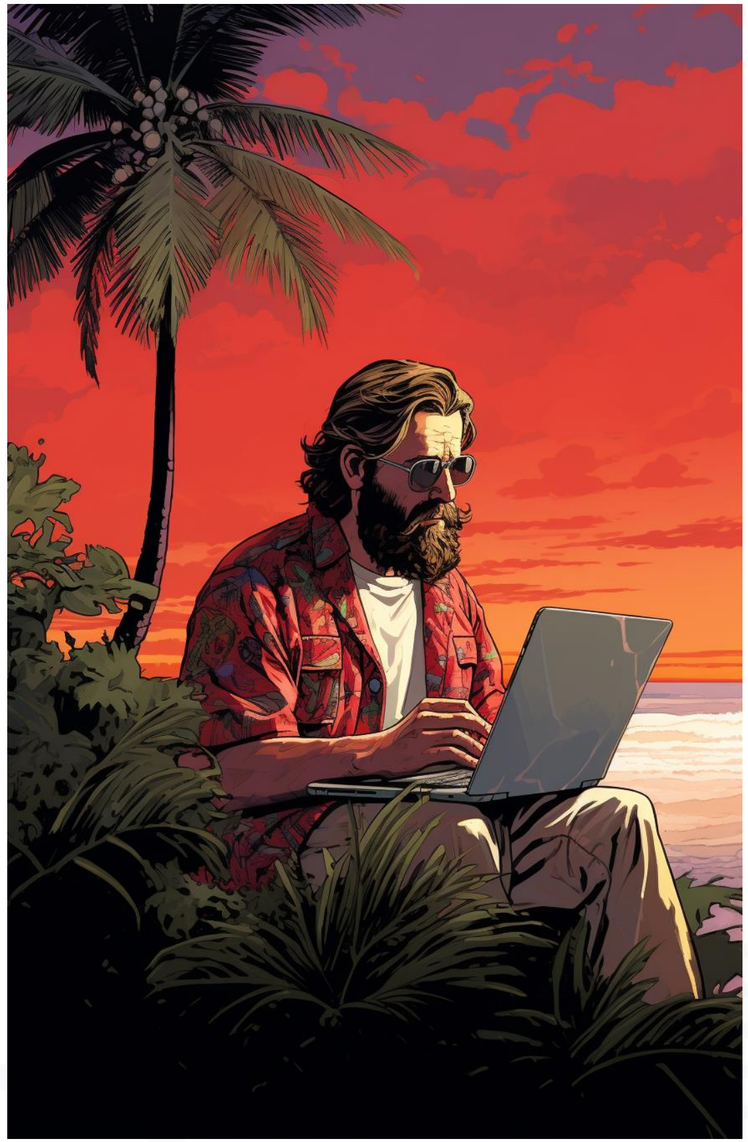
Do it how you please

I want a SemVer from Fenix

SemVer communicates the scope of the changes

Packing Unpackaged Dependencies

Why This Was a Problem



No man is an island, and no software project these days has no dependencies

These installs were decades-old

A major bottleneck to get these installs in place

It was on me to fix it

I installed twice, and then automated

But how?

Demo

Take a second look at a NuGet package

Look at a handful of raw dlls from an install

Package them up into a new package

Validate our package

Install it to a project

Wrap-up

Identify the files that are copied by an installer or other non-automatable process

T

https://bit.ly/3s51osm

Strip the package down

Modify the contents to reflect your new package

Up Next:

An API/CLI Demo

Demo

Using the API

List out all our feeds using the raw API

Using the CLI we'll list out the contents of a repo

Look at the reference material for all this stuff

Course Summary

- Understand what binaries actually are
- Learned Semantic Versioning
- Working with Artifactory in builds
- Working with Docker containers in Artifactory and Jenkins
- Some advanced topics
- Packing problem dependencies
- Working with the Artifactory API and CLI

Thank You for Watching!