Versioning Your Binaries

Chris B. Behrens

Software Architect

@chrisbbehrens

Tripartite Versions

1.17.9

1.19.5

One version, by itself, in isolation is worthless.

Versions are only meaningful in light of other versions.

Installing Something for the First Time

The latest, because why not?

Generally, the best and most complete effort from the enterprise

Your only concern is compatibility

Compatibility

That's the magic word

Nothing relies on anything brand new

Informed decisions about what's changed

Compatibility States

Changes from the old version were minor, and compatibility is not expected to be impacted

Changes from the old version were moderate, but there were no changes to the public interface that are likely
to break compatibility

Changes from the old version were significant, and there’s a good chance that the code will have to change to be compatible with the new version

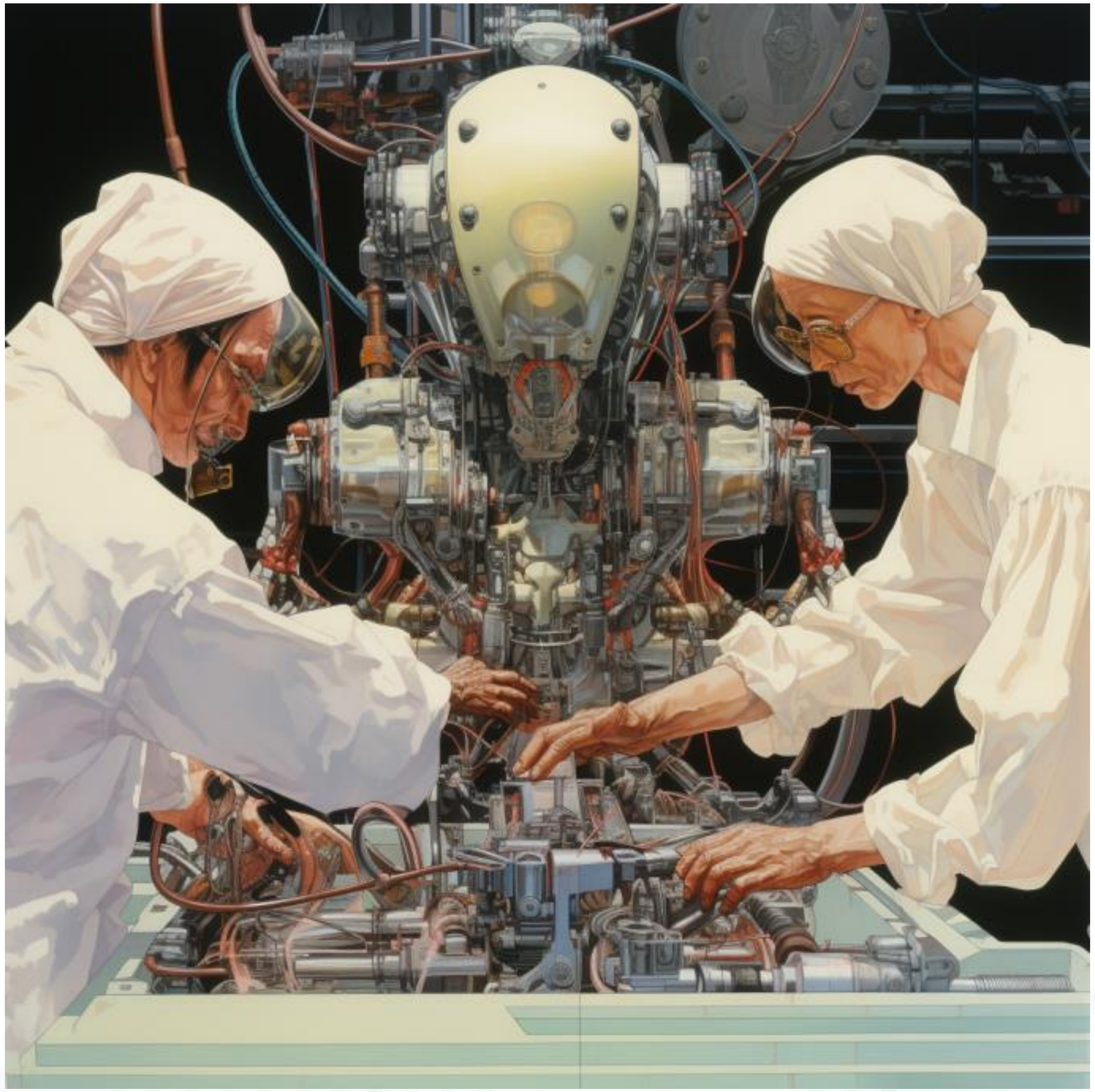
Why does it matter, anyway?

Why create headaches?

Software Composition Analysis (SCA)

Package ID and version form a key for a database of known issues

The first asterisk (why a single version is meaningful)



Introducing Semantic Versioning

What causes each part of the version to change?

A developer decided it should be that way.

Semantic Release and Other Semver Automators

https://bit.ly/45JvHCo

“remove the connection between human emotions and version numbers”

These rely on the format of commit messages

So, we're still driven by those human emotions

The Parts of a SemVer

major.minor.patch

What sort of change does each element of a semantic version represent?

Patch

No impact on compatibility is expected

1.17.9

1.17.14

This should be safe to upgrade

1.0.0 => 1.0.1

Minor

A change, but a backward compatible one

1.17.9

1.18.0

Major bugfixes MAY increment MINOR

1.0.1 => 1.1.0

Major

Breaking compatibility with the previous version

1.17.9

2.0.0

Possible problems, but maybe not

1.1.0 => 2.0.0

Some Problems with SemVer

https://xkcd.com/1172
/

Changes can surprise you

Implementing caching in the background broke my library

Hynek on Semantic Versioning

https://hynek.me/articles/semver-will-not-save-you

SemVer is a TL;DR
of the Changelog

Semantic Versioning doesn’t have to be perfect; it just has to be better than the alternative.

What I Want You to Take from All of This

We need *some* way to communicate changes in compatibility

In an ideal world, a detailed changelog

SemVer is the best worst solution we have

Demo

Flip the order of the SemVer we described

Begin with a major version update

Update the minor version

Perform a bugfix

Increment the patch version

A Judgment Call

Maybe a patch, maybe a minor version bump

The point is to communicate impact

Maybe this is a significant change

The Rest of SemVer

From semver.org

A pre-release version MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version. Identifiers MUST comprise only ASCII alphanumerics and hyphens [0-9A-Za-z-]. Identifiers MUST NOT be empty. Numeric identifiers MUST NOT include leading zeroes.

From semver.org

Pre-release versions have a lower precedence than the associated normal version. A pre-release version indicates that the version is unstable and might not satisfy the intended compatibility requirements as denoted by its associated normal version. Examples: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7, 1.0.0-x.7.z.92, 1.0.0-x-y-z.

What This Tells Us

"It's ready...we think"

It is what it is

This allows you to get a new interface into users' hands

From semver.org

Major version zero (0.y.z) is for initial development. Anything MAY change at any time. The public API SHOULD NOT be considered stable.

0.9.0 and 1.0.0

From semver.org

Once a versioned package has been released, the contents of that version MUST NOT be modified. Any modifications MUST be released as a new version.

This is the way

The Problem with Republishing

Is 4.3.0 before or after 2:13 PM on August 4th, 2013?

This way madness lies

A Particular Versioning Story

A course on Jenkins in Docker

It works through a chain of plugins

One of them broke binary compatibility without bumping the major version

So, we all had to hash it out on a Saturday afternoon

Summary

Semantic Versioning in depth

- Major
- Minor
- Patch

With a demo of each

I need more Artifactory in my Artifactory course