Szemantikus verziószámozás

Jeszenszky Péter

2023.03.06.

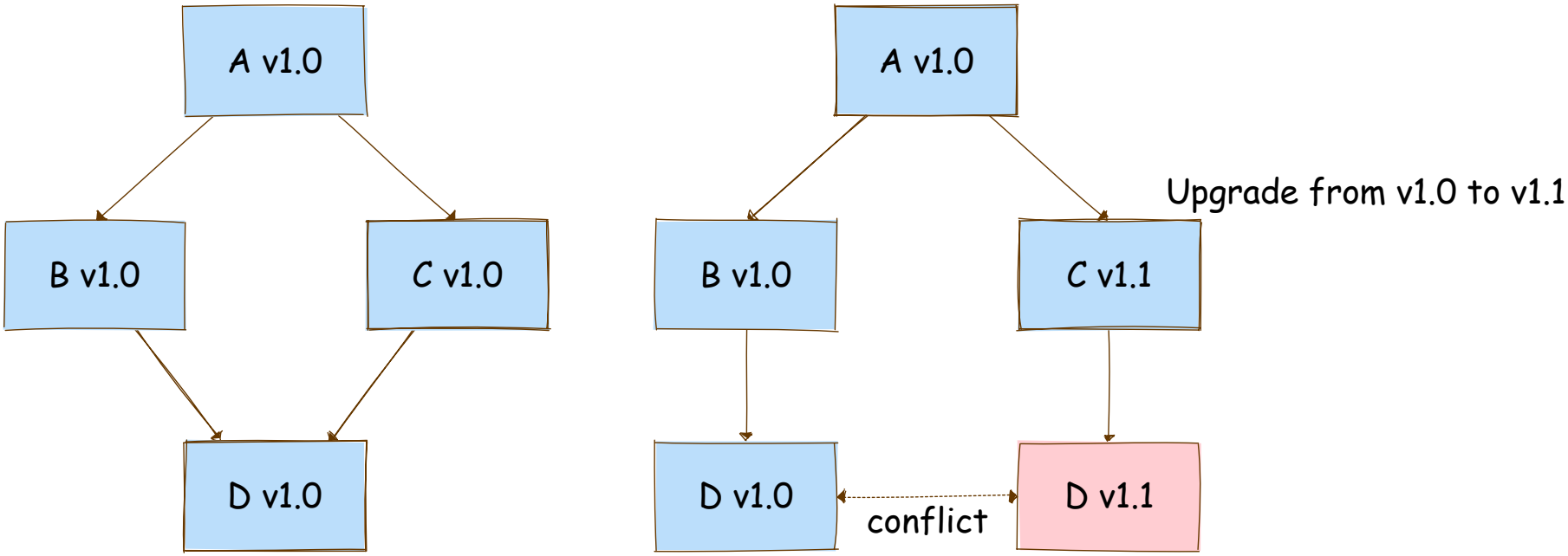
Bevezetés

Miért van szükség a verziószámokhoz egy formális specifikációra?

- A függőségkezeléshez pontosan meghatározott verziószámok szükségesek, melyek világos és rugalmas függőség-specifikációkat tesznek lehetővé.
- Kapcsolódó fogalom: verziópokol (*version hell*), függőségi pokol (*dependency hell*)

Verziópokol

Példa:



Mi a szemantikus verziószámozás?

- Az általános bevett gyakorlaton alapuló egyszerű szabályok és követelmények a verziószámok kiosztásához és növeléséhez.
- Tetszőleges olyan szoftverhez használható, mely nyilvános API-val rendelkezik.
  - Az API változásai a verziószámának növelésével kerülnek kifejezésre.
- Webhely: https://semver.org/
- Tároló: https://github.com/semver/semver

Specifikáció

- Aktuális specifikáció: *Semantic Versioning 2.0.0*
  https://semver.org/spec/v2.0.0.html
  - Számos nyelven elérhető.
  - Magyar fordítás: https://semver.org/lang/hu/spec/v2.0.0.html
- A Szemantikus verziószámozás specifikáció eredeti szerzője Tom Preston-Werner, a GitHub egyik társalapítója.

Normál verziószámok

A normál verziószámok $X.Y.Z$ formájúak, ahol X, Y és Z nemnegatív egész számok:

- X: főverzió (*major version*),
- Y: alverzió (*minor version*),
- Z: *patch* verzió.

Összefoglalás

Növeld

- a főverziót, amikor a korábbi verzióval inkompatibilis módon változik az API;
- az alverziót, amikor a korábbi verzióval kompatibilis módon vezetünk be új funkcionalitást;
- a *patch* verziót, amikor a korábbi verzióval kompatibilis hibajavítások történnek.

Részletek

- Miután kiadásra került egy verziózott csomag, a verzió tartalma nem módosítható. Bármilyen módosítást egy új verzióként kell kiadni.
- A nulla főverzió a kezdeti fejlesztéshez van fenntartva.
  - Bármilyen változás történhet, az API nem tekinthető stabilnak.
- A *patch* verziót 0-ra kell visszaállítani az alverzió növelésekor (például `1.2.3` $\rightarrow$ `1.3.0`).
- A *patch* verziót és az alverziót 0-ra kell visszaállítani a főverzió növelésekor (például `0.9.15` $\rightarrow$ `1.0.0`).

Breaking Changes

Kapcsolódó fogalom:

- ***Breaking change***: nem visszafelé kompatibilis változás egy nyilvános API-ban.
  - Az API kliensei számára fordításidejű, szerkesztésidejű vagy futásidejű hibákat okoz.

Kiadás előtti verziók

Egy kiadás előtti (*pre-release*) verzió $X.Y.Z\text{-}V$ formájú, ahol V alfanumerikus karakterekből és kötőjelekből (`[0-9A-Za-z-]`) álló azonosítók egy pontokkal elválasztott sorozata.

- A kiadás előtti verziók alacsonyabb precedenciájúak, mint a megfelelő normál verziók.
- Egy kiadás előtti verzió azt jelzi, hogy a verzió instabil és lehet, hogy nem elégíti ki a megfelelő normál verzió tervezett kompatibilitási követelményeit.
- Példák: `1.0.0-alpha`, `1.0.0-alpha.1`, `1.0.0-beta.2`

Összeállítási metaadatok

Alfanumerikus karakterekből és kötőjelekből (`[0-9A-Za-z-]`) álló azonosítók egy pontokkal elválasztott sorozata követheti egy plusz jel után a *patch* vagy a kiadás előtti verziót.

- Az ilyen összeállítási metaadatokat figyelmen kívül kell hagyni a verzió precedencia megállapításakor.
- Példák: `1.0.0-alpha+001`, `1.0.0+20230304142500`

Verzió precedencia

Példa:

`1.0.0-alpha` < `1.0.0-alpha.1` < `1.0.0-alpha.beta` < `1.0.0-beta` < `1.0.0-beta.2` < `1.0.0-beta.11` < `1.0.0-rc.1` < `1.0.0`

Felhasználások

Csomag ökoszisztémák és csomagkezelők:

- Node.js: npm
  - Lásd: https://docs.npmjs.com/about-semantic-versioning
- PHP: Composer
  - Lásd: https://getcomposer.org/doc/faqs/which-version-numbering-system-does-composer-itself-use.md
- Rust: Cargo
  - Lásd: https://doc.rust-lang.org/cargo/reference/resolver.html#semver-compatibility
- .NET: NuGet
  - Lásd: https://learn.microsoft.com/en-us/nuget/concepts/package-versioning

Eszközök

- composer/semver (programozási nyelv: PHP; licenc: *MIT License*)
  https://github.com/composer/semver
- GitVersion (programozási nyelv: C#; licenc: *MIT License*)
  https://gitversion.net/ https://github.com/GitTools/GitVersion
- semver (programozási nyelv: JavaScript; licenc: *ISC License*)
  https://www.npmjs.com/package/semver
  https://github.com/npm/node-semver
- semver (programozási nyelv: Python; licenc: *New BSD License*)
  https://github.com/python-semver/python-semver
  https://python-semver.readthedocs.io/en/latest/
- Semver4j (programozási nyelv: Java; licenc: *MIT License*)
  https://github.com/vdurmont/semver4j

Más verziószámozási sémák

- *Calendar Versioning*: https://calver.org/ https://github.com/mahmoud/calver
  - Példa:
    - Ubuntu: The Ubuntu lifecycle and release cadence
- *ZeroVer: 0-based Versioning*: https://0ver.org/ https://github.com/mahmoud/zerover
- Python: PEP 440: Version Identification and Dependency Specification
- TEX: a TEX verziószáma a π-hez konvergál, a legutóbbi verzió a `3.141592653` számú.
  - Lásd: https://www.ctan.org/pkg/tex

Apache Maven és a szemantikus verziószámozás (1)

- A Maven verzió rendezési algoritmusa nem kompatibilis a Szemantikus verziószámozás 2.0.0 specifikációéval.
  - Például a Maven nem kezeli a plusz jelet speciális karakterként.
  - A szemantikus verziószámozástól eltérően a Maven nem tulajdonít jelentést a verziószámoknak.
- Lásd:
  - POM Reference - Version Order Specification
  - `ComparableVersion` (Javadoc)

Apache Maven és a szemantikus verziószámozás (2)

Empirikus megfigyelés a központi Maven tárolóra:

- A könyvtár frissítések 83,4%-a megfelel a szemantikus verziószámozásnak.
- A tanulmányhoz felhasznált eszköz:
  - Maracas (programozási nyelv: Java; licenc: *MIT License*)
    https://crossminer.github.io/maracas/
    https://github.com/crossminer/maracas
- Lásd:
  - Lina Ochoa, Thomas Degueule, Jean-Rémy Falleri, Jurgen Vinju. *Breaking bad? Semantic versioning and impact of breaking changes in Maven Central: An external and differentiated replication study*. Empirical Software Engineering, 2022, 27 (3).
    https://hal.science/hal-03378089