NETMOD YANG Semver Update

NETMOD WG

November 2023

Presenting on behalf of the weekly versioning call attendees:
Joe Clarke

IETF 118

Key Issue #2 (from 117)

- Do we need multiple `revision-label` schemes?

- Initially, the thinking was vendors might want to bring their own

- This adds complexity, yet no one concretely suggested an alternate scheme

  ➢ Recommendation: **NO multiple schemes**. YANG Semver to be *the* scheme, and `revision-label` to be renamed `yang-semver:version`

- Note: It is desirable to keep a label to encode rough information about compatibility
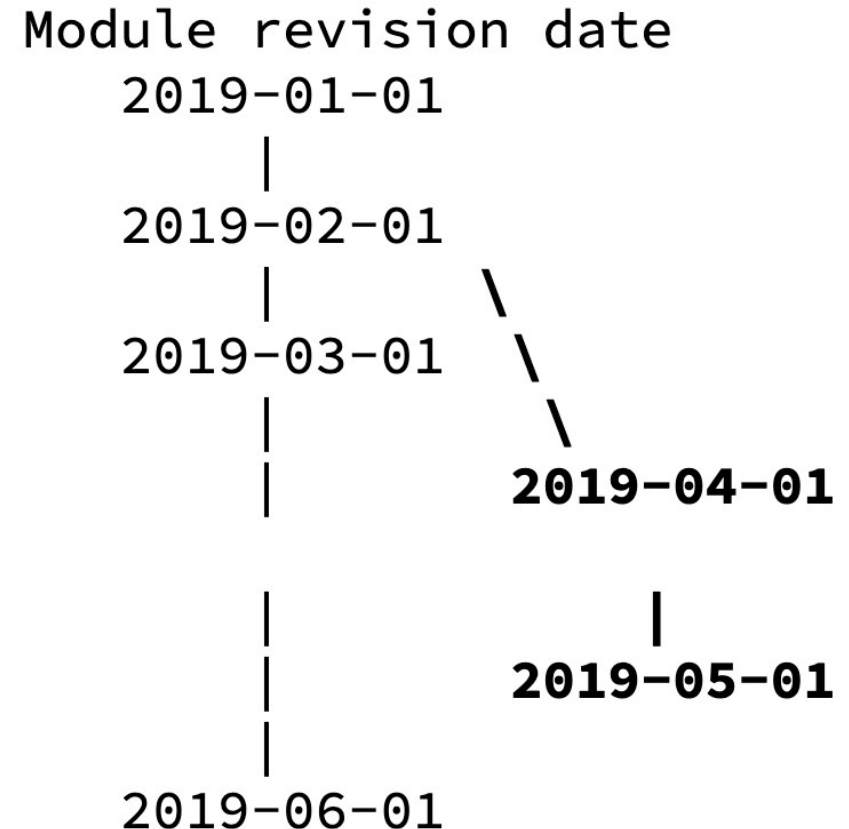
Key Issue #3 (from 117)

Why do we need YANG Semver (vs. SemVer 2.0.0)?

SemVer 2.0.0

- Linear (no branching)
- Simpler in construction
  - Major
  - Minor
  - Patch
- 1.0.0, 1.0.1, 1.1.0, 2.0.0, …
  - If a new feature is needed in 1.0.1, a *2.1.0* would need to be minted that incorporates the features of 1.1.0 and 2.0.0 (creating a 1.2.0 is ambiguous with respect to 2.0.0 in how SemVer is documented)

Module Versioning allows non-linear revision history

- Not actually disallowed by RFC 7950!
- No limit to the amount of branching that could be expressed!
- **Does not imply arbitrary branching is good or recommended**
- Revision history represents the path from that revision to the root of the revision tree
- Dates alone don't tell the story (e.g., someone might assume 2019-06-01 is fully compatible with 2019-05-01)

```
Module revision date
2019-01-01
    |
2019-02-01
    |          \
2019-03-01      \
    |            \
    |         2019-04-01
    |
    |              |
    |         2019-05-01
    |
2019-06-01
```

YANG Semver

- Support for *limited* branching
- Supports SemVer 2.0.0 rules (without branching it's 100% compatible to SemVer)
- MAJOR.MINOR.PATCH_MODIFIER
  - _compatible
  - _non_compatible
- 1.0.0, 1.0.1, 1.1.0, 2.0.0, *1.0.2_compatible*
  - A feature (or an NBC change can be backported)

Recommendation for YANG Semver

- Given that module versioning allows branching, the labeling scheme must also support branching

- YANG Semver is a compromise between power and simplicity
  - Encourage "mostly" single track development with modifiers the exception
  - Retains support for some updates to older versions

- Sufficient for SDOs and vendors