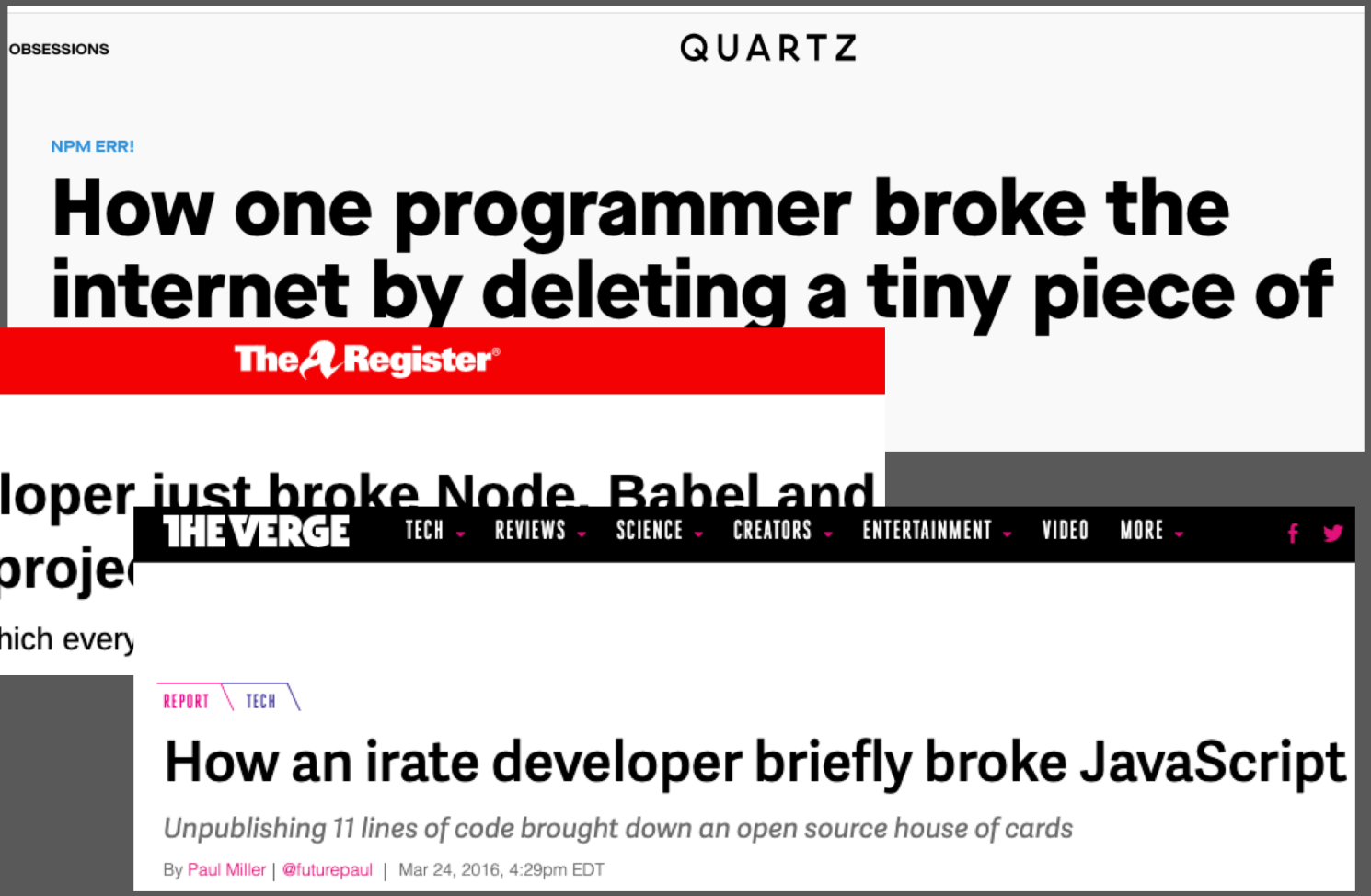
# Dependency Management & Versioning

Michael Hilton and **Rohan Padhye**

institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

# Left-pad (March 22, 2016)



OBSESSIONS

QUARTZ

NPM ERR!

**How one programmer broke the internet by deleting a tiny piece of**

SIGN IN

The Register

{* SOFTWARE *}

**How one developer just broke Node, Babel and thousands of proje**

Code pulled from NPM – which every

THE VERGE

TECH REVIEWS SCIENCE CREATORS ENTERTAINMENT VIDEO MORE

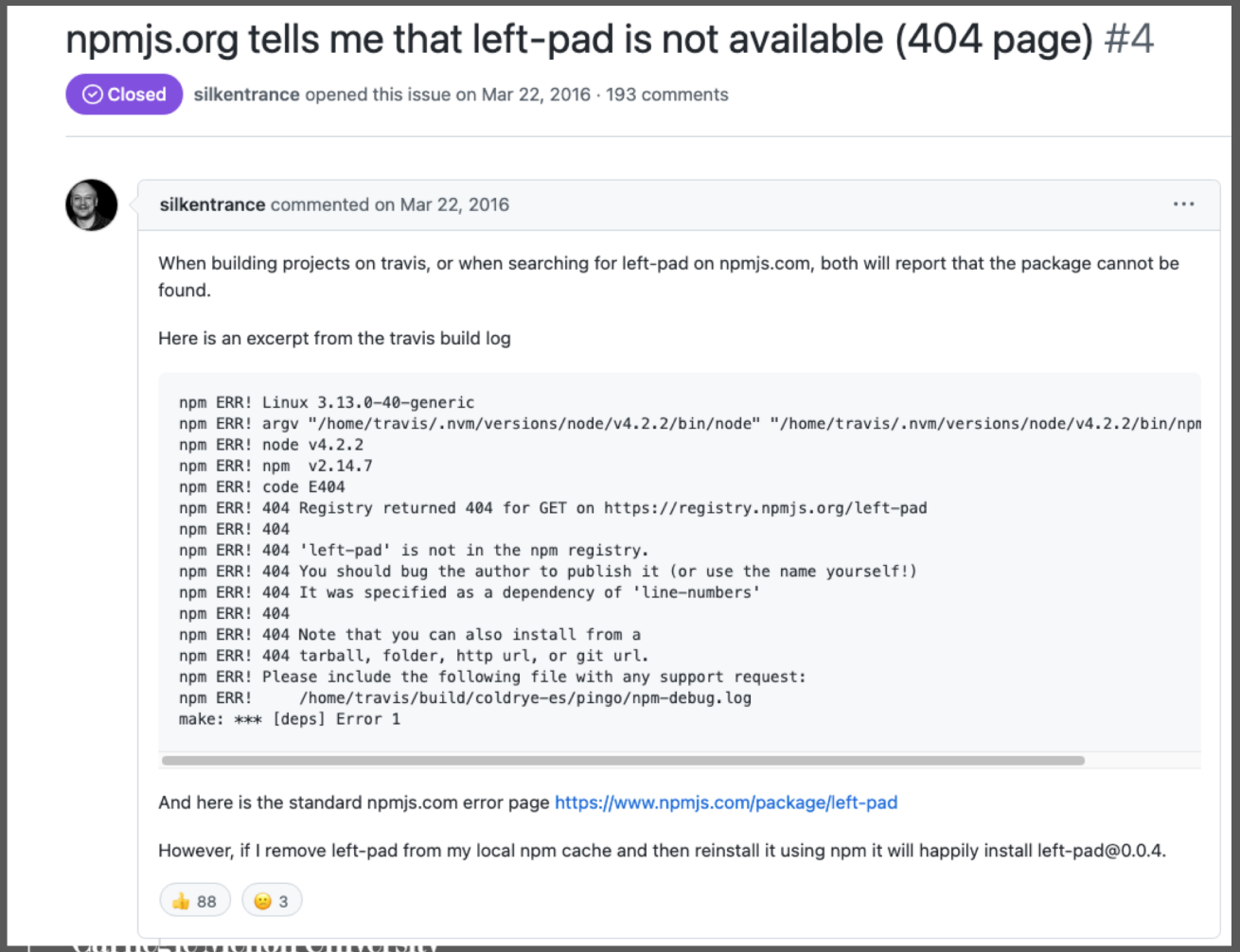
REPORT \ TECH

**How an irate developer briefly broke JavaScript**

*Unpublishing 11 lines of code brought down an open source house of cards*

By Paul Miller | @futurepaul | Mar 24, 2016, 4:29pm EDT

# Left-pad (March 22, 2016)



## npmjs.org tells me that left-pad is not available (404 page) #4

Closed **silkentrance** opened this issue on Mar 22, 2016 · 193 comments

**silkentrance** commented on Mar 22, 2016

When building projects on travis, or when searching for left-pad on npmjs.com, both will report that the package cannot be found.

Here is an excerpt from the travis build log

```
npm ERR! Linux 3.13.0-40-generic
npm ERR! argv "/home/travis/.nvm/versions/node/v4.2.2/bin/node" "/home/travis/.nvm/versions/node/v4.2.2/bin/npm
npm ERR! node v4.2.2
npm ERR! npm  v2.14.7
npm ERR! code E404
npm ERR! 404 Registry returned 404 for GET on https://registry.npmjs.org/left-pad
npm ERR! 404
npm ERR! 404 'left-pad' is not in the npm registry.
npm ERR! 404 You should bug the author to publish it (or use the name yourself!)
npm ERR! 404 It was specified as a dependency of 'line-numbers'
npm ERR! 404
npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.
npm ERR! Please include the following file with any support request:
npm ERR!     /home/travis/build/coldrye-es/pingo/npm-debug.log
make: *** [deps] Error 1
```

And here is the standard npmjs.com error page https://www.npmjs.com/package/left-pad

However, if I remove left-pad from my local npm cache and then reinstall it using npm it will happily install left-pad@0.0.4.

88 3

# Left-pad (Docs)

## left-pad

String left pad

build unknown

## Install

```
$ npm install left-pad
```

## Usage

```
const leftPad = require('left-pad')

leftPad('foo', 5)
// => "  foo"

leftPad('foobar', 6)
// => "foobar"

leftPad(1, 2, '0')
// => "01"

leftPad(17, 5, 0)
// => "00017"
```

Install

```
> npm i left-pad
```

Repository
github.com/stevemao/left-pad

Homepage
github.com/stevemao/left-pad#readme

Weekly Downloads
2,962,641

| Version | License |
|---|---|
| 1.3.0 | WTFPL |

| Unpacked Size | Total Files |
|---|---|
| 9.75 kB | 10 |

| Issues | Pull Requests |
|---|---|
| 3 | 7 |

Last publish
4 years ago

# Left-pad (Source Code)

17 lines (11 sloc) | 222 Bytes

```
1   module.exports = leftpad;
2
3   function leftpad (str, len, ch) {
4     str = String(str);
5
6     var i = -1;
7
8     if (!ch && ch !== 0) ch = ' ';
9
10    len = len - str.length;
11
12    while (++i < len) {
13      str = ch + str;
14    }
15
16    return str;
17  }
```

# See also: isArray

5 lines (4 sloc) | 133 Bytes

```
1  var toString = {}.toString;
2
3  module.exports = Array.isArray || function (arr) {
4    return toString.call(arr) === '[object Array]';
5  };
```

## isarray

`Array#isArray` for older browsers and deprecated Node.js versions.

build passing downloads 227M/month


**Just use Array.isArray directly**, unless you need to support those older versions.

### Usage

```
var isArray = require('isarray');

console.log(isArray([])); // => true
console.log(isArray({})); // => false
```

Install

```
> npm i isarray
```

Repository
github.com/juliangruber/isarray

Homepage
github.com/juliangruber/isarray

Weekly Downloads
50,913,317

Version
2.0.5

License
MIT

Unpacked Size
3.43 kB

Total Files
4

Issues
4

Pull Requests
3

# Dependency Management

- It's hard
- It's mostly a mess (everywhere)
- But it's critical to modern software development

# What is a Dependency?

- Core of what most build systems do
  - “Compile” and “Run Tests” is just a fraction of their job
- Examples: Maven, Gradle, NPM, Bazel, ...
- Foo->Bar: To build Foo, you may need to have a built version of Bar
- Dependency Scopes:
  - Compile: Foo uses classes, functions, etc. defined by Bar
  - Runtime: Foo uses an abstract API whose implementation is provided by Bar (e.g. logging, database, network or other I/O)
  - Test: Foo needs Bar only for tests (e.g. JUnit, mocks)
- Internal vs. External Dependencies
  - Is Bar also built/maintained by your org or is it pulled from elsewhere using a package manager?

# Dependencies: Example

master Mayan-EDMS / requirements / base.txt

Roberto Rosario Workaround swagger-spec-validator bug ...

1 contributor

45 lines (45 sloc) | 893 Bytes

```
1  Pillow==8.3.1
2  PyPDF2==1.26.0
3  PyYAML==5.4.1
4  Whoosh==2.7.4
5  bleach==4.0.0
6  celery==5.1.2
7  django-activity-stream==0.10.0
8  django-celery-beat==2.2.1
9  django-colorful==1.3
10 django-cors-headers==3.8.0
11 django-formtools==2.2
12 django-mathfilters==1.0.0
13 django-model-utils==4.1.1
14 django-mptt==0.12.0
15 django-pure-pagination==0.3.0
16 django-qsstats-magic==1.1.0
17 django-solo==1.1.5
18 django-stronghold==0.4.0
19 django-widget-tweaks==1.4.8
20 djangorestframework==3.12.4
21 djangorestframework-recursive==0.1.2
22 drf-yasg==1.20.0
23 extract-msg==0.28.7
24 flanker==0.9.11
25 flex==6.14.1
26 furl==2.1.2
27 fusepy==3.0.1
28 gevent==21.8.0
29 graphviz==0.17
30 gunicorn==20.1.0
31 jsonschema==3.2.0
32 mock==4.0.3
33 node-semver==0.8.0
34 packaging==21.0
```

Package: git (1:2.17.1-1ubuntu0.9 and others) [**security**]

fast, scalable, distributed revision control system

Other Packages Related to git

● depends ◆ recommends ■ suggests • enhances

- git-man (<< 1:2.17.0-.) [not amd64, i386]
  fast, scalable, distributed revision control system (manual pages)
- git-man (<< 1:2.17.1-.) [amd64, i386]
- git-man (>> 1:2.17.0) [not amd64, i386]
- git-man (>> 1:2.17.1) [amd64, i386]
- libc6 (>= 2.16) [not arm64, ppc64el]
  GNU C Library: Shared libraries
  also a virtual package provided by libc6-udeb
- libc6 (>= 2.17) [arm64, ppc64el]
- libcurl3-gnutls (>= 7.16.2)
  easy-to-use client-side URL transfer library (GnuTLS flavour)
- liberror-perl
  Perl module for error/exception handling in an OO-ish way
- libexpat1 (>= 2.0.1)
  XML parsing C library - runtime library
- libpcre3
  Old Perl 5 Compatible Regular Expression Library - runtime files
- perl
  Larry Wall's Practical Extraction and Report Language
- zlib1g (>= 1:1.2.0)
  compression library - runtime
- ◆ less
  pager program similar to more
- ◆ patch
  Apply a diff file to an original
- ◆ ssh-client
  virtual package provided by openssh-client

Links for git

Ubuntu Resources:

- Bug Reports
- Ubuntu Changelog
- Copyright File

Download Source Package git:

- [git_2.17.1-1ubuntu0.9.dsc]
- [git_2.17.1.orig.tar.xz]
- [git_2.17.1-1ubuntu0.9.debian.tar.xz]

Maintainer:

- Ubuntu Developers (Mail Archive)

Please consider filing a bug or asking a question v
Launchpad before contacting the maintainer
directly.

Original Maintainers (usually fror
Debian):

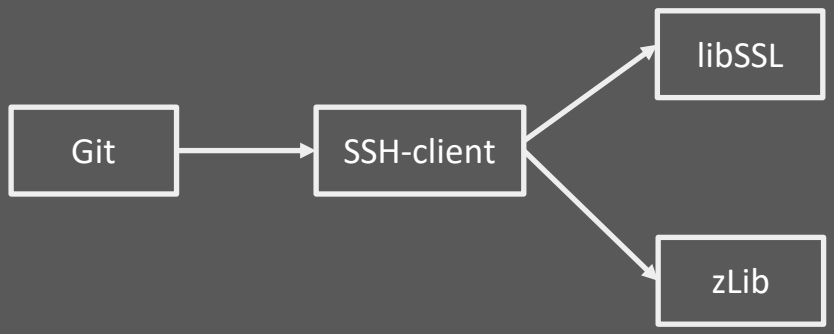
- Gerrit Pape
- Jonathan Nieder
- Anders Kaseorg

It should generally not be necessary for users to
contact the original maintainer.

External Resources:

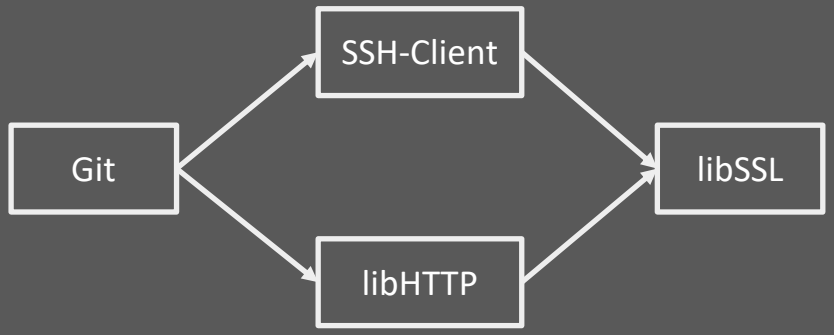
# Transitive Dependencies

- Should Git be able to use exports of libSSL (e.g. certificate management) or zLib (e.g. gzip compression)?

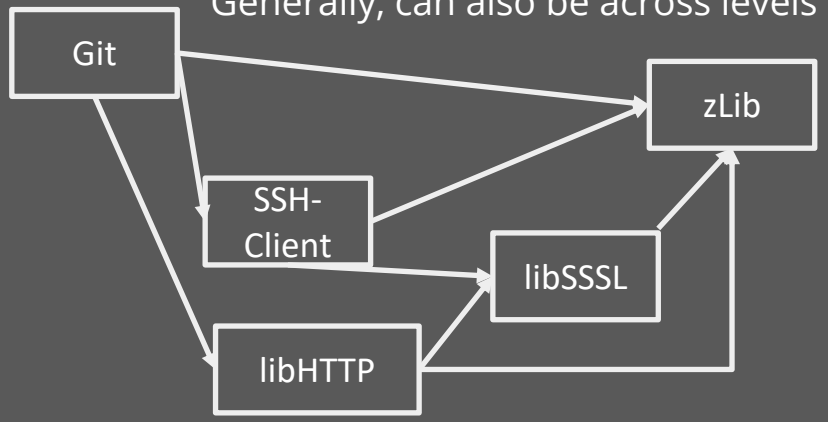


# Diamond Dependencies

- What are some problems when multiple intermediate dependencies have the same transitive dependency?

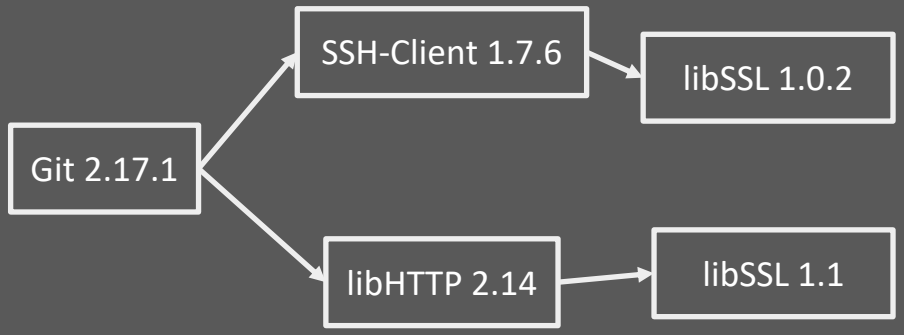



Generally, can also be across levels



# Diamond Dependencies

- What are some problems when multiple intermediate dependencies have the same transitive dependency?



# Resolutions to the Diamond Problem

1. Duplicate it!
   - Doesn't work with static linking (e.g. C/C++), but may be doable with Java (e.g. using ClassLoader hacking or package renaming)
   - Values of types defined by duplicated libraries cannot be exchanged across
2. Ban transitive dependencies; just use a global list with one version for each
   - Challenge: Keeping things in sync with latest
   - Challenge: Deciding which version of transitive deps to keep
3. Newest version (keep everything at latest)
   - Requires ordering semantics
   - Intermediate dependency may break with update to transitive
4. Oldest version (lowest denominator)
   - Also requires ordering semantics
   - Sacrifices new functionality
5. Oldest non-breaking version / Newest non-breaking version
   - Requires faith in tests or semantic versioning contract

# Semantic Versioning

- Widely used convention for versioning releases
  - E.g. 1.2.1, 3.1.0-alpha-1, 3.1.0-alpha-2, 3.1.0-beta-1, 3.1.0-rc1
- Format: {MAJOR} . {MINOR} . {PATCH}
- Each component is ordered (numerically, then lexicographically; release-aware)
  - 1.2.1 < 1.10.1
  - 3.1.0-alpha-1 < 3.1.0-alpha-2 < 3.1.0-beta-1 < 3.1.0-rc1 < 3.1.0
- Contracts:
  - MAJOR updated to indicate breaking changes
    - Same MAJOR version => backward compatibility
  - MINOR updated for additive changes
    - Same MINOR version => API compatibility (important for linking)
  - PATCH updates functionality without new API
    - Ninja edit; usually for bug fixes

# https://semver.org/

2.0.0 2.0.0-rc.2 2.0.0-rc.1 1.0.0 1.0.0-beta

## Semantic Versioning 2.0.0

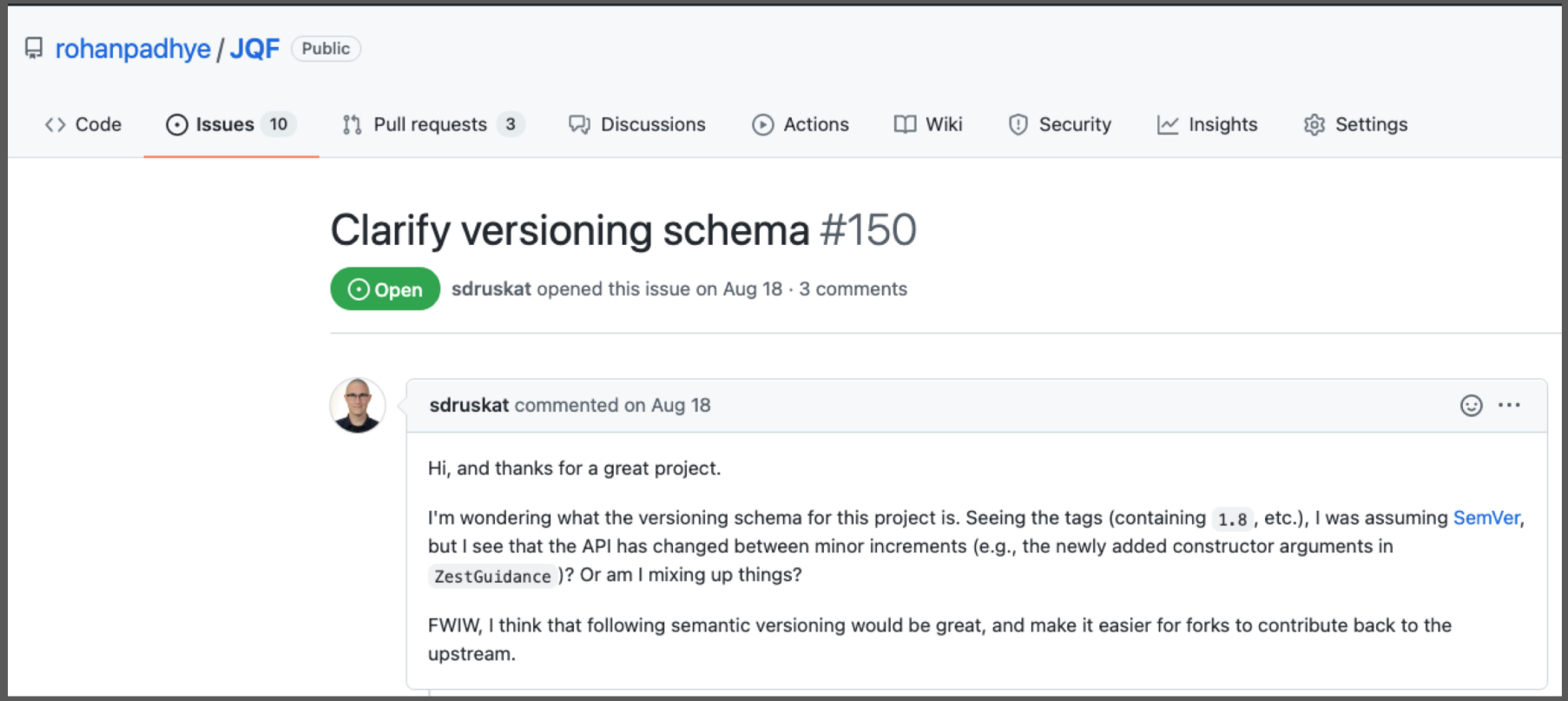
### Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

# People rely on SemVer contracts

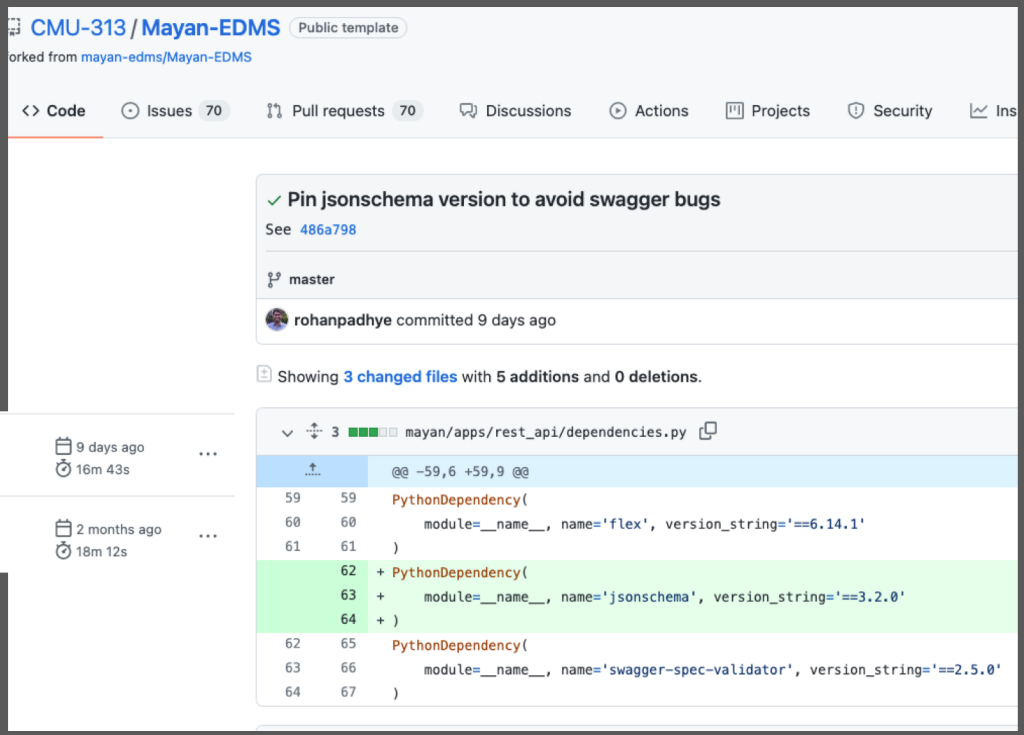


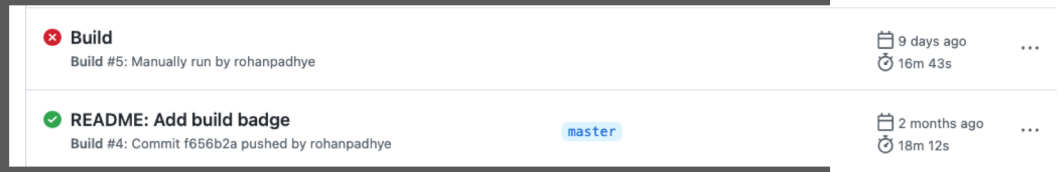
# Dependency Constraints

- E.g. Declare dependency on ”Bar > 2.1”
  - Bar 2.1.0, 2.1.1, 2.2.0, 2.9.0, etc. all match
  - 2.0.x does NOT match
  - 3.0.x does NOT match
- Diamond dependency problem can be resolved using SAT solvers
  - E.g. Foo 1.0.0 depends on “Bar >= 2.1” and “Baz 1.8.x”
    - Bar 2.1.0 depends on “Qux [1.6, 1.7]”
    - Bar 2.1.1 depends on “Qux 1.7.0”
    - Baz 1.8.0 depends on “Qux 1.5.x”
    - Baz 1.8.1 depends on “Qux 1.6.x”
  - Find an assignment such that all dependencies are satisfied
    - Solution: Use Bar 2.1.0, Baz 1.8.1, and Qux 1.6.{latest}

# Semantic Versioning Contracts

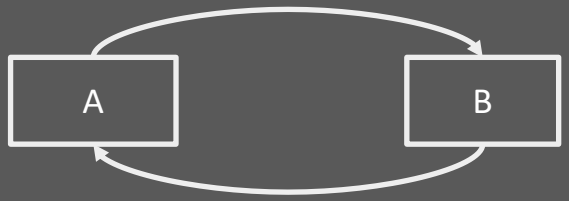
- Largely trusting developers to maintain them
- Constrained/range dependencies can cause unexpected build failures
- Automatic validation of SemVer is hard



# Cyclic Dependencies

- A very bad thing
- Avoid at all costs
- Sometimes unavoidable or intentional
  - E.g. GCC is written in C (needs a C compiler)
  - E.g. Apache Maven uses the Maven build system
  - E.g. JDK tested using JUnit, which requires the JDK to compile



# Cyclic Dependencies

- Bootstrapping: Break cycles over time
- Assume older version exists in binary (pre-built form)
- Step 1: Build A using an older version of B
- Step 2: Build B using new (just built) version of A
- Step 3: Rebuild A using new (just built) version of B
- Now, both A and B have been built with new versions of their dependencies
- Doesn't work if both A and B need new features of each other at the same time (otherwise Step 1 won't work)
  - Assumes incremental dependence on new features
- How was the old version built in the first place? (it's turtles all the way down)
  - Assumption: cycles did not exist in the past
  - Successfully applied in compilers (e.g. GCC is written in C)

# Dependency Reliability

- Availability
  - Remember left-pad?
  - Many orgs will mirror package repositories
- Security
  - Will you let strangers execute arbitrary code on your laptop?
  - Think about this every time you do “pip install” or “npm install” or “apt-get updgrade” or “brew updgrade” or whatever (esp. with sudo)
    - Scary, right? Who are you trusting? Why?
  - Typo squatting
    - “pip install numpi”

# Takeaways

- Dependency management is hard.