

View online

Download PDF

How to breed a good OSS community

Paolo Insogna

Node.js TSC, Principal Engineer @ **Platformatic**

Don't ask if you can't handle the answers!



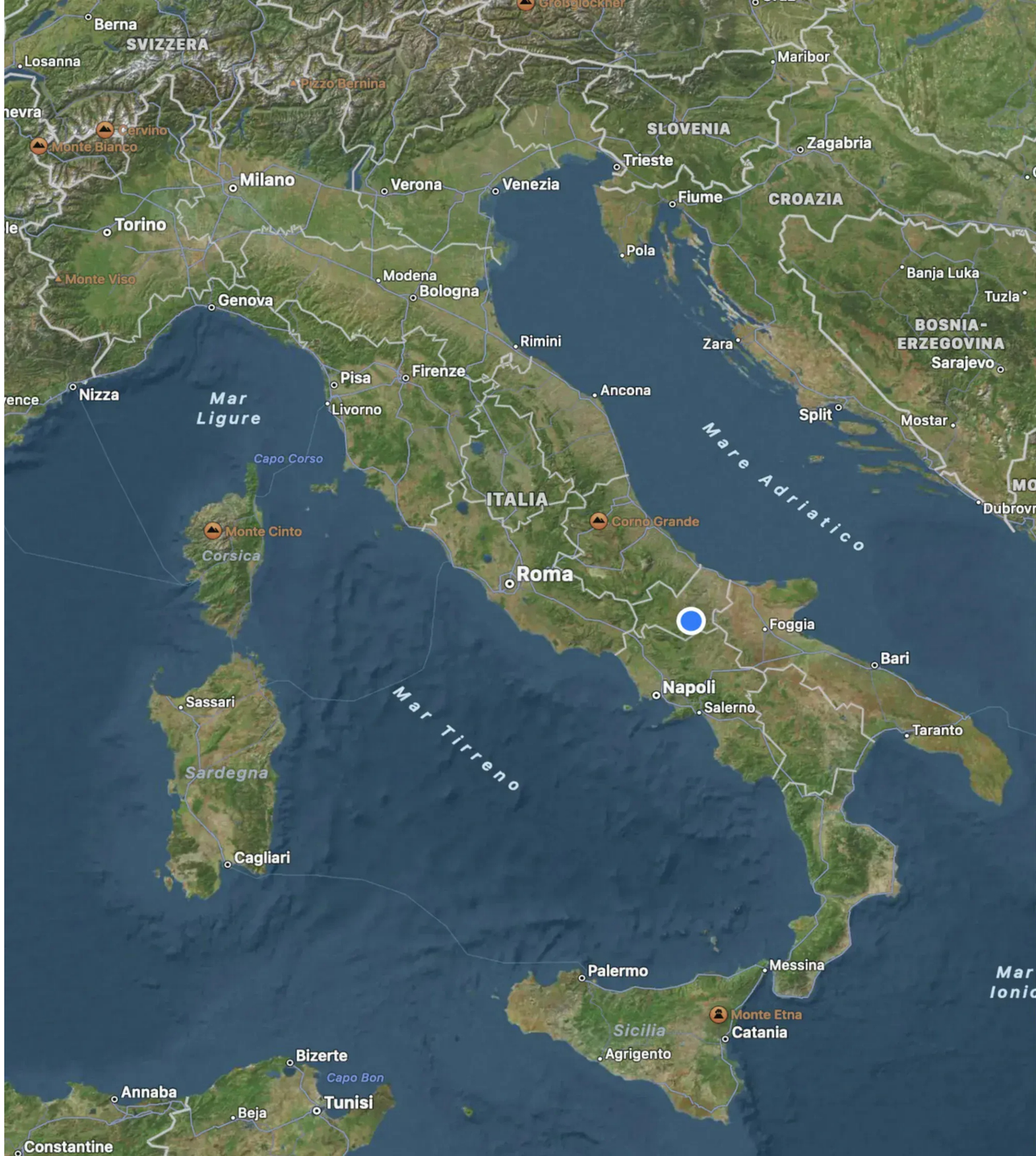
Hello, I'm **Paolo**!



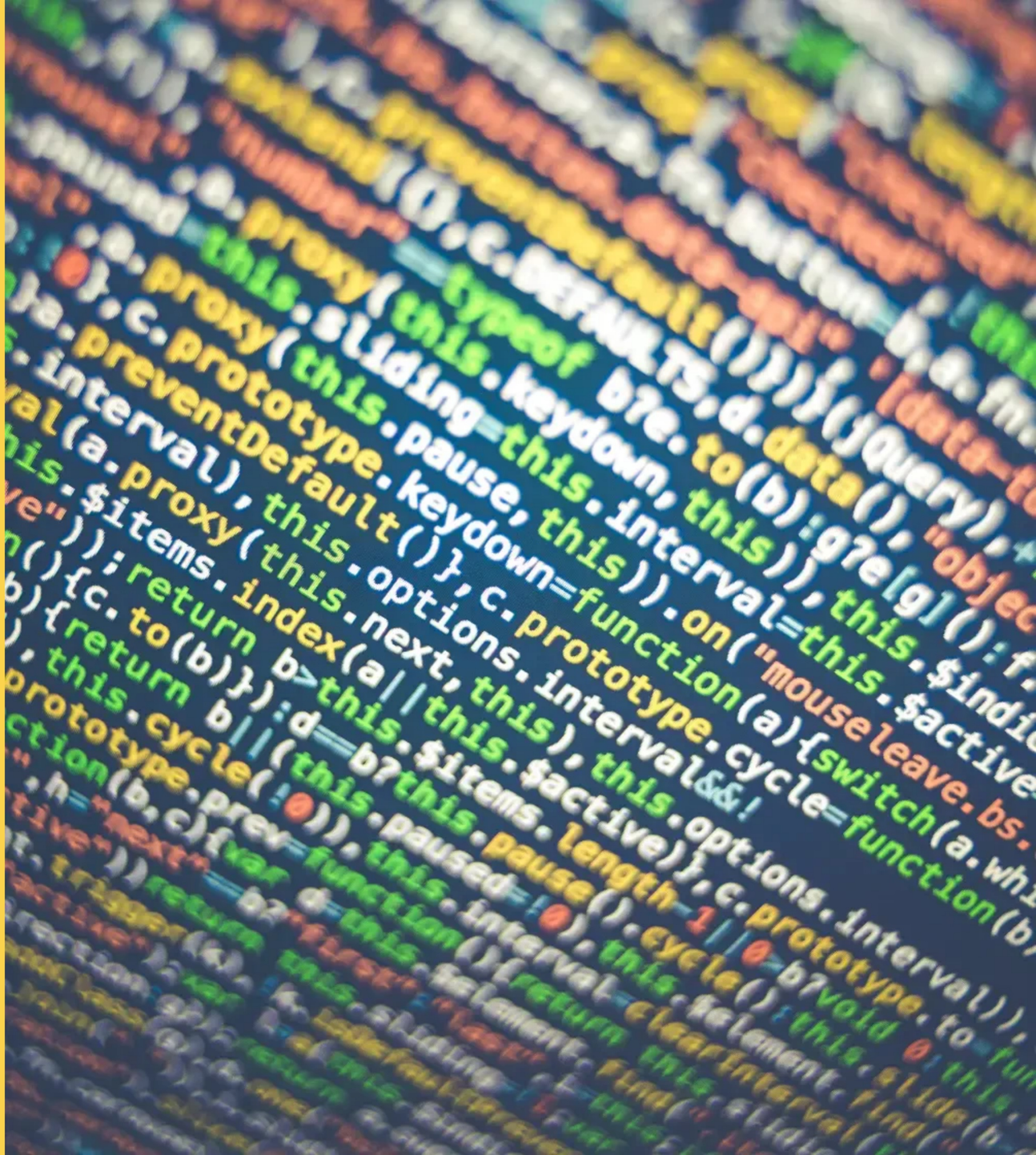
Node.js Technical Steering Committee Member

Platformatic Principal Engineer

`paoloinsogna.dev` `ShogunPanda` `p_insogna` `pinsogna`



Do you want to mess with Open Source?



Let's clarify things



What is it?

A decentralized software development model that **encourages open collaboration**.



How is it possible?

The secret is in the **passion** of the people.



Is it working?

Open Source has proven to be really effective in improving software quality.

How did it start?



Initially, it was free, not open

In the 70s the **free software movement** challenged the for-profit based development.



Free means too much

In English, the word free is ambiguous. **Not paid** was the most understood meaning.

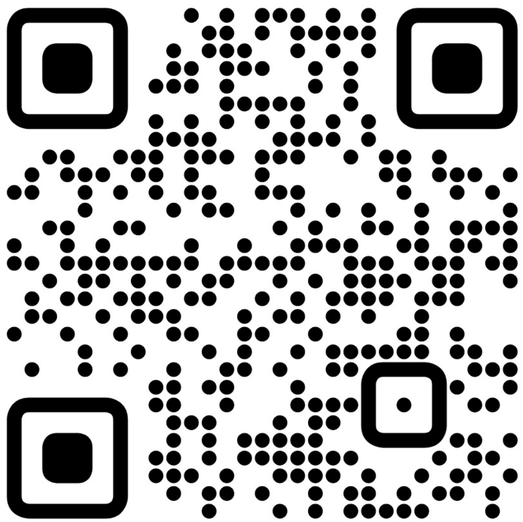


Open Source was branched off the Free Software Movement

OSS software can be used for commercial purposes. **The freedom is retained.**

It might not be OpenSource

To be considered Open Source, a software must use a OSS compatible license.
The **Open Source Initiative** maintains a list of such licenses.



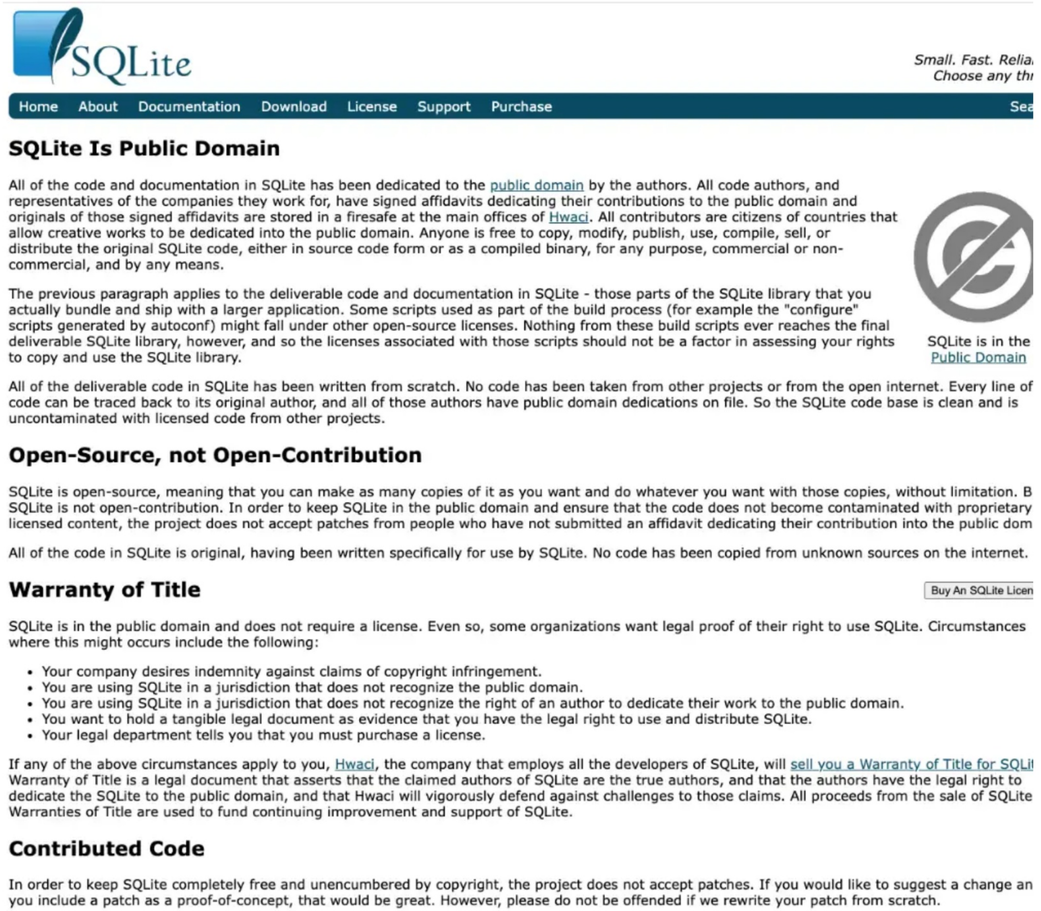
Beware!



The devil is in the details

Even if a software has a OSS license, it might not strictly be OSS.

SQLite is **Open Source but not Open Contribution**.



SQLite

Small. Fast. Relia. Choose any thr

Home About Documentation Download License Support Purchase Sea

SQLite Is Public Domain

All of the code and documentation in SQLite has been dedicated to the public domain by the authors. All code authors, and representatives of the companies they work for, have signed affidavits dedicating their contributions to the public domain and originals of those signed affidavits are stored in a firesafe at the main offices of Hwaci. All contributors are citizens of countries that allow creative works to be dedicated into the public domain. Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

The previous paragraph applies to the deliverable code and documentation in SQLite - those parts of the SQLite library that you actually bundle and ship with a larger application. Some scripts used as part of the build process (for example the "configure" scripts generated by autoconf) might fall under other open-source licenses. Nothing from these build scripts ever reaches the final deliverable SQLite library, however, and so the licenses associated with those scripts should not be a factor in assessing your rights to copy and use the SQLite library.

SQLite is in the Public Domain

All of the deliverable code in SQLite has been written from scratch. No code has been taken from other projects or from the open internet. Every line of code can be traced back to its original author, and all of those authors have public domain dedications on file. So the SQLite code base is clean and is uncontaminated with licensed code from other projects.

Open-Source, not Open-Contribution

SQLite is open-source, meaning that you can make as many copies of it as you want and do whatever you want with those copies, without limitation. B SQLite is not open-contribution. In order to keep SQLite in the public domain and ensure that the code does not become contaminated with proprietary licensed content, the project does not accept patches from people who have not submitted an affidavit dedicating their contribution into the public dom

All of the code in SQLite is original, having been written specifically for use by SQLite. No code has been copied from unknown sources on the internet.

Warranty of Title

Buy An SQLite Licen

SQLite is in the public domain and does not require a license. Even so, some organizations want legal proof of their right to use SQLite. Circumstances where this might occurs include the following:

- Your company desires indemnity against claims of copyright infringement.
- You are using SQLite in a jurisdiction that does not recognize the public domain.
- You are using SQLite in a jurisdiction that does not recognize the right of an author to dedicate their work to the public domain.
- You want to hold a tangible legal document as evidence that you have the legal right to use and distribute SQLite.
- Your legal department tells you that you must purchase a license.

If any of the above circumstances apply to you, Hwaci, the company that employs all the developers of SQLite, will sell you a Warranty of Title for SQLi Warranty of Title is a legal document that asserts that the claimed authors of SQLite are the true authors, and that the authors have the legal right to dedicate the SQLite to the public domain, and that Hwaci will vigorously defend against challenges to those claims. All proceeds from the sale of SQLite Warranties of Title are used to fund continuing improvement and support of SQLite.

Contributed Code

In order to keep SQLite completely free and unencumbered by copyright, the project does not accept patches. If you would like to suggest a change an you include a patch as a proof-of-concept, that would be great. However, please do not be offended if we rewrite your patch from scratch.

Why would somebody want to go Open Source?



The best of the best of the best

You can access a developer base not physically close to you.



Different cultures, backgrounds and capabilities

You get different point of views, which increases quality.



Low maintenance cost

You don't need an office, just a version control server.

Wait a sec...





Ring a bell?

It's similar, but it's not!

The Open Source development model is similar to Remote Working.
The huge difference is that people are driven by **passion**, not money.

You are not their boss!

How to make it similar?

You can create bounty programs to prioritize collaboration on bug fixing or feature development.



BAD!

DON'T!

Diversity and freedom are a huge treasure



Let them follow their path

The Open Source model works because people can freely follow their paths.



It's a hobby

Most collaborators send code in their free time, after a day of boring coding.



Keep the difference

If you make it look like a job, you will lose them.

How can you avoid losing control?

1

Propose, don't impose

You can share your vision and see if people are interested.

2

Follow your path

You can still deliver the features you care about.

3

Be open

Other people's path might be useful for yours.

Trust the community!



The world is nice ...

People generally want to help.

They have interest in what you are doing for their own needs.



… but don't be afraid to be harsh …

Quick and effective interventions about uncomfortable situations are the key.

Avoiding creating a toxic community is always your **first** priority.



… and don't forget to be kind!

You will get people of all knowledges and expertises. Be patient, especially with newcomers.

Basically, it's like **parenting**: endless patience, close to no recognition.



But here we're talking about breeding...



The job NEVER ends



Too big to fail

Once you start a community, you are responsible for maintaining it. Potentially forever.



Don't take it personally

The best way to manage such communities is to avoid a **one-person show**.



Everybody is replaceable

All people are important, but prepare the community to survive if people suddenly disappear.

Nobody wants to be sued!

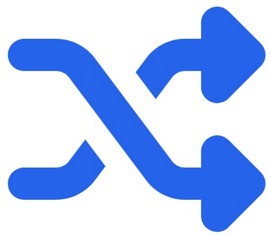


Why do we need legal licenses?

Since Open Source Software is used in every sector, it has legal implications and liability. You need to avoid consequences.

Do not reinvent the wheel, choose an existing license!

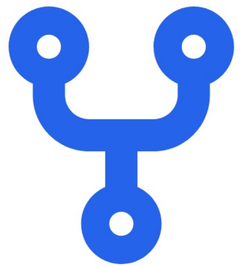
Remember, it's permanent ...



You can change...
With every release, you can pick a new compatible license.



... but not retroactively.
Only new releases use different licenses. Past is unchangeable.



Forking is unavoidable
If people don't like the new license, they can fork the software.

...but you can still profit from it!



Remember how it started?
Open Source doesn't mean free.



Think outside of the box
You don't have to think about profiting directly from the software.



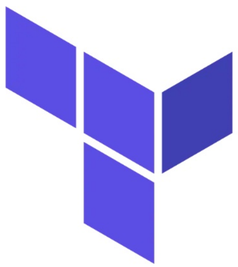
Be around and everywhere
You can profit by offering premium services or a commercial derived product.

Don't try to mess the system

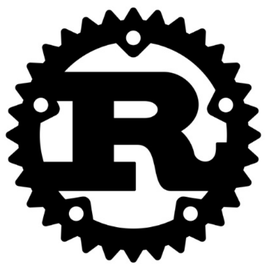
People don't like if you try to take away their beloved Open Source software.



Node.js



Terraform



Rust

Being kind never hurts!



Questions?




Coffee Break

Let's get to the action!



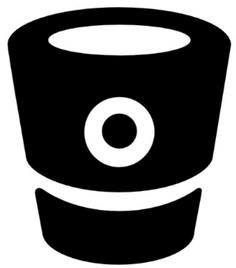
First of all, choose the hosting



GitHub



GitLab



BitBucket

Three reasons not to go self-hosted or exotic

1

Self hosting is hard

Ownership cost will increase.

2

Self hosting is hidden

You will be difficult to find.

3

Git is well known

Do not create entry barriers.

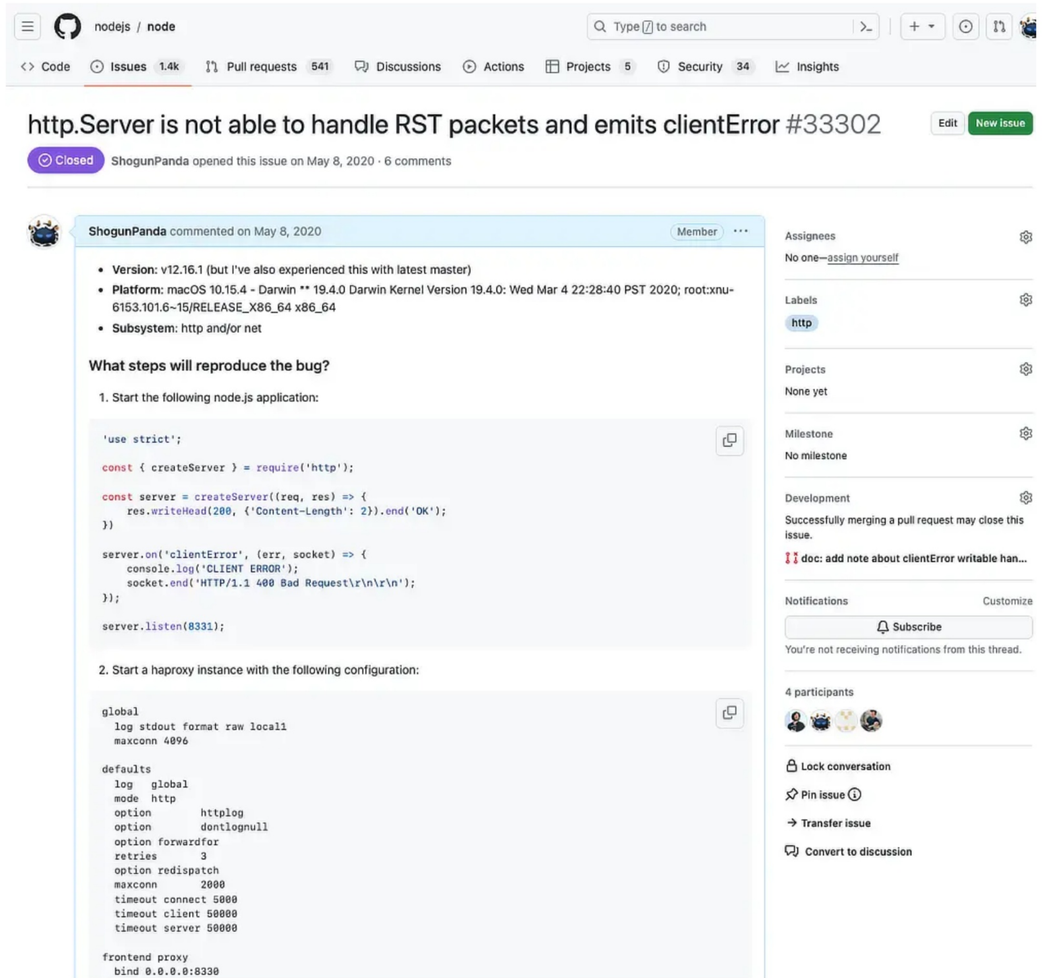
How do we get started?



Everybody has issues

The core idea to attract people is to show you care.

The issue system is where you usually take feedback and bug reports from your user base.



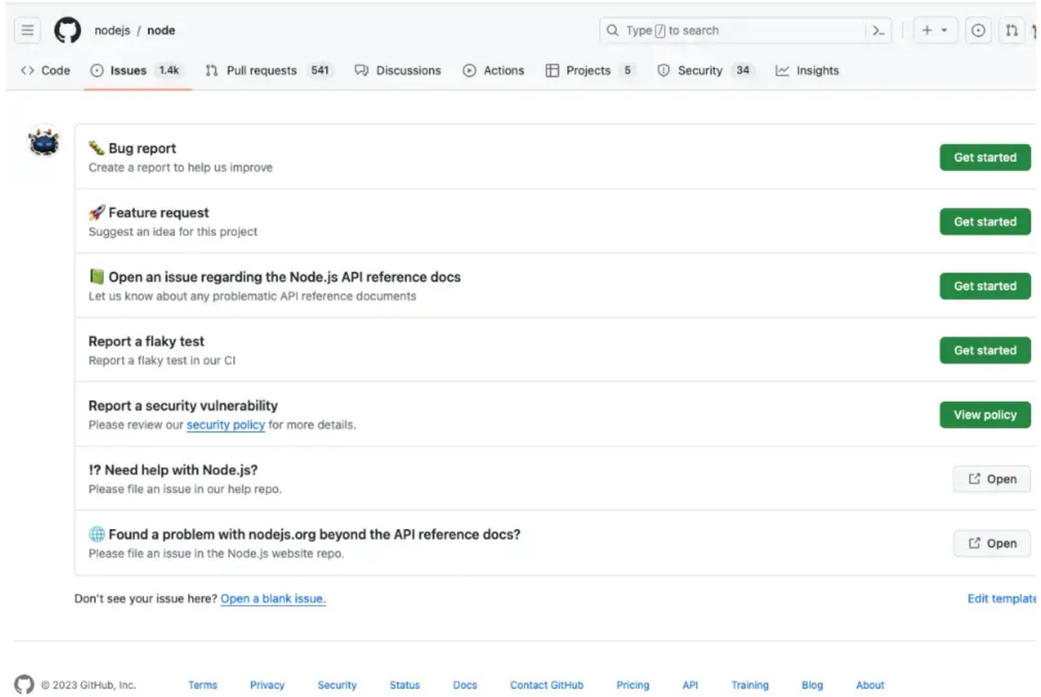
Templates help

It will help you

You can explicitly ask for the info you care the most.

It will help them

Having templates speeds up their reporting time.



Reproducible examples are crucial for bugs



Reduce to the minimum

You want to rule out all irrelevant details.



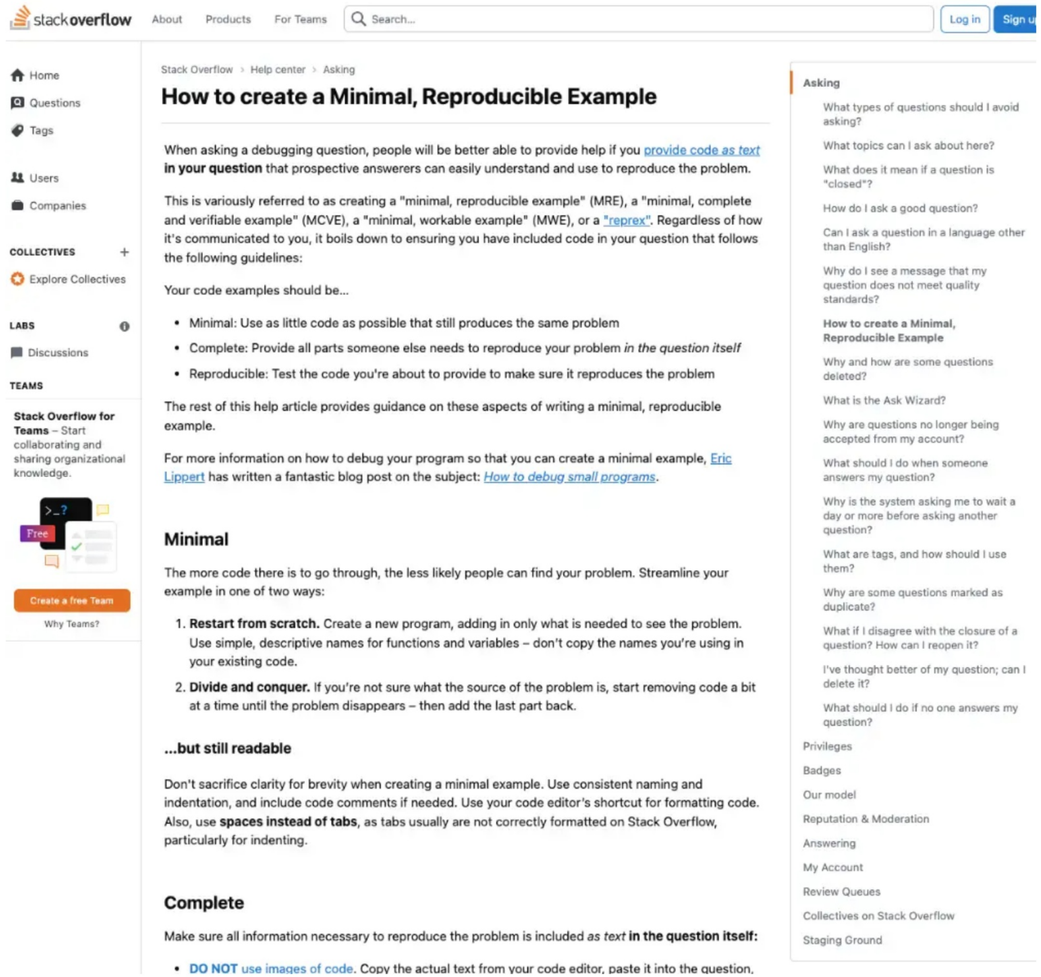
Remove sensitive information

Nobody wants to get in trouble.



Ask for a public repository

It will facilitate reporters to help triagers.



You don't owe anybody anything ...

It is in your rights to close bugs or features without solving them.



… but don't forget kindness!

1

Be respectful

Provide an exhaustive explanation.

2

Be gentle

Use a kind tone.

3

Be reasonable

Be open to change your mind.

In case of doubts

“A component of a system should behave in a way that most users will expect it to behave, and therefore not astonish or surprise users.”

Principle of least surprise

Label everything



Have a process
Following procedures will speed you up.



Divide et impera
Using labels helps categorizing and dividing tasks.

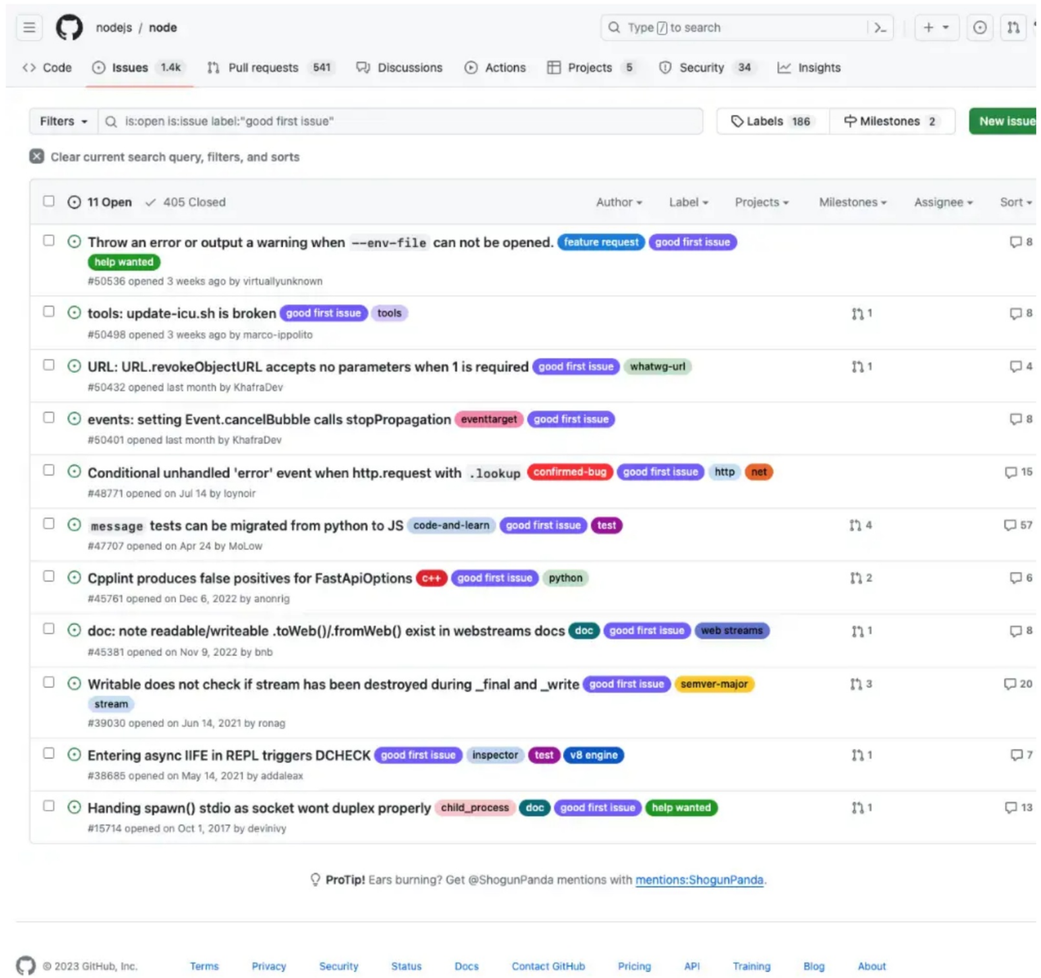


Don't overcommit
Limit the number of labels so that they are still useful.

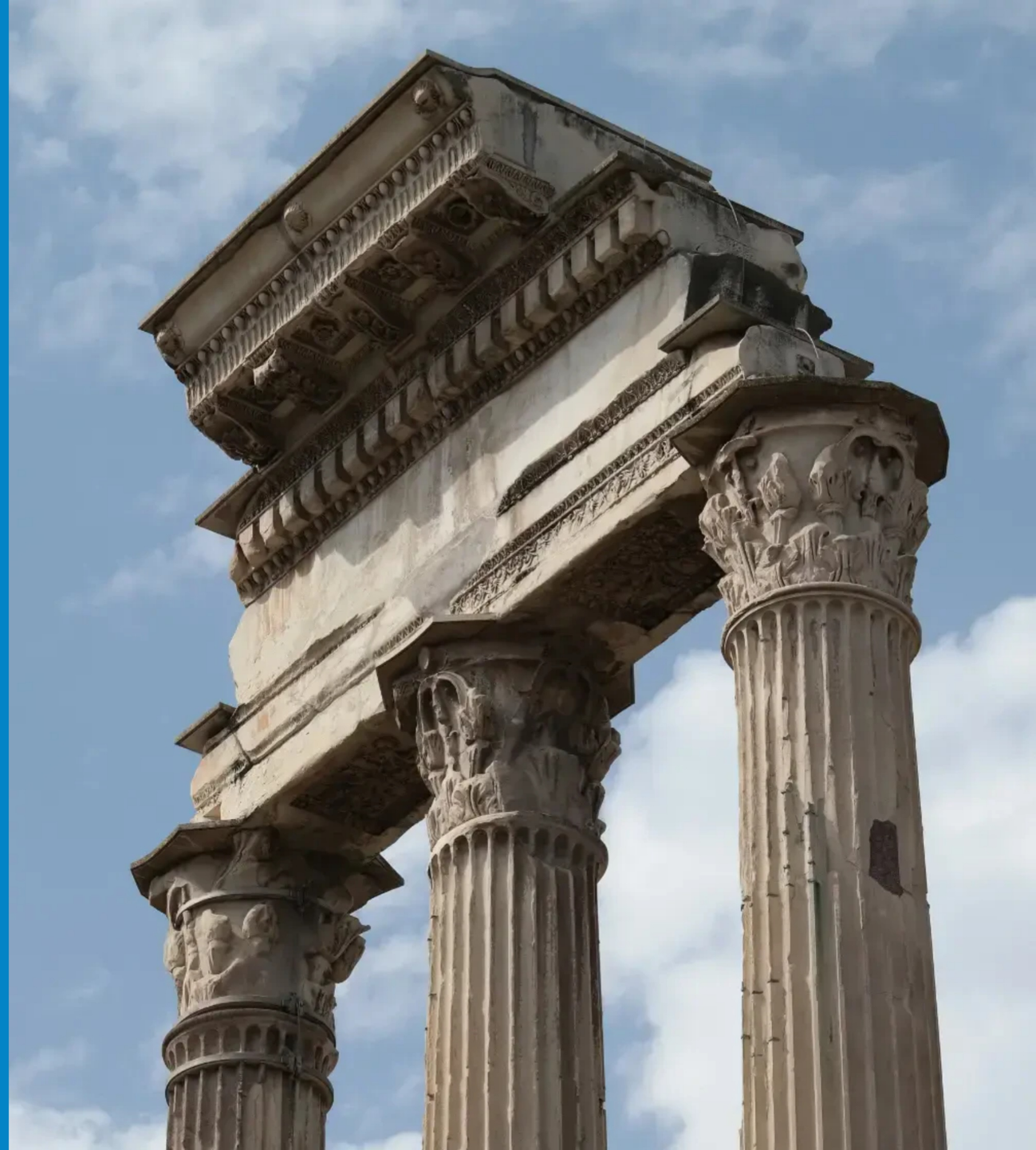
Remember about good first time issues

When people come in your community and want to help, they don't know where to start.

Having **good first time issues** will help them to independently starting being involved.



The next big pillar



Pull Request



The core of the model

This is how collaborators send you code.



The quality will vary

It is your responsibility to maintain the level.



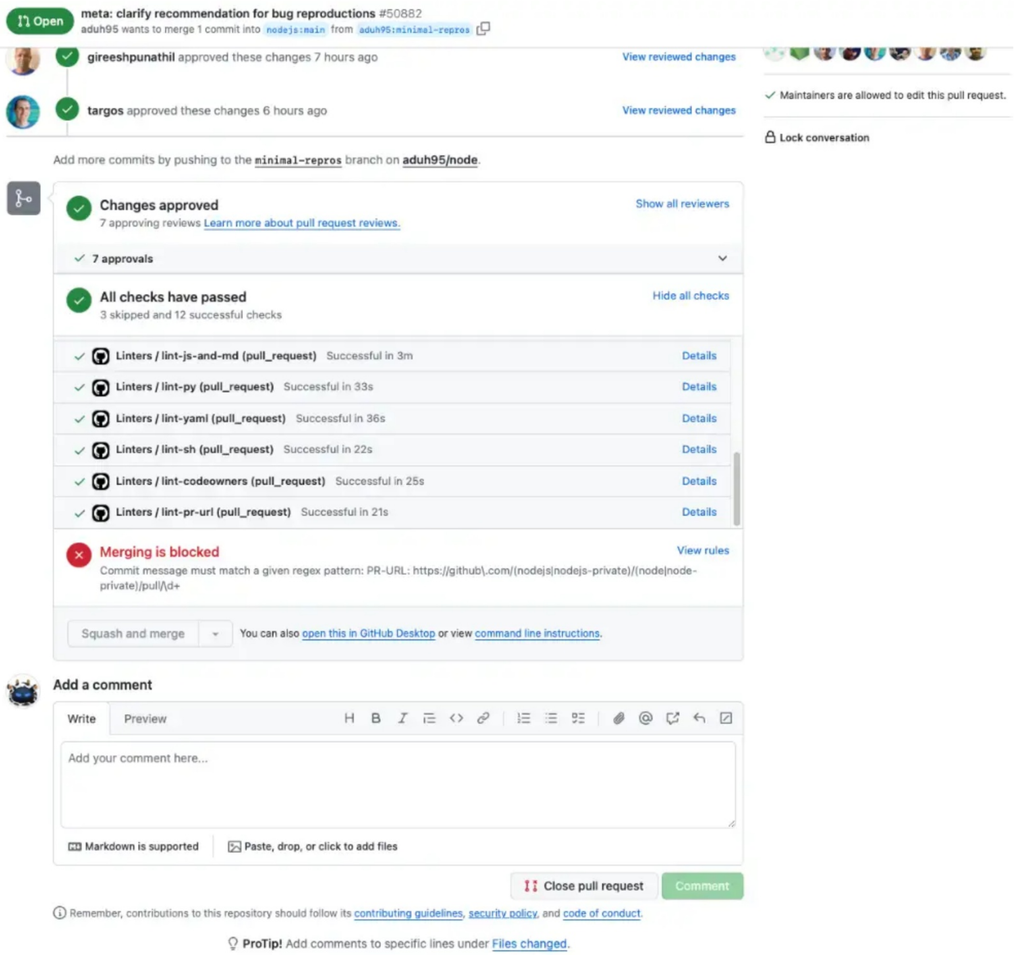
The scope will vary

Same as above, **don't be afraid of changes**.

Always ask for tests

While formal TDD is hard, Pull Requests **must** contain tests when non trivial.

Even if a PR content is very appealing, you have to **resist the temptation to accept it without tests**.



The review process

1. **Behavior and security**
   Make sure the PR does what it claims, and double check security implications.

2. **Performance**
   Nobody likes slow software, do they?

3. **Quality and clarity**
   Poorly (or obscurely) written performant code makes more harm than good.

4. **Enforce the conventions**
   Same as above, **don't be afraid to ask for picky changes**.

Review must be thorough

No one rushes you to merge code. If possible, require for multiple approvals before committing.

Once you give enough time for objection, then you can merge.



CI and workflows



Available everywhere

In modern services you have a CI and workflow system.



It's so easy

Most of these tools are very easy to use.

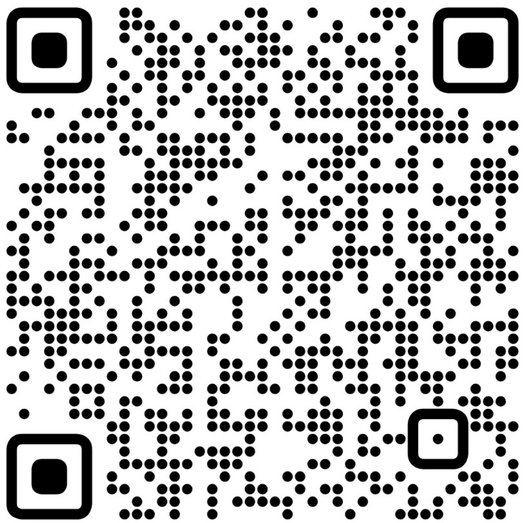


CI is a life saver

Don't merge any PR with a red CI.

Conventional commits

How to add human and machine readable meaning to commit messages.
Check the **official website**!

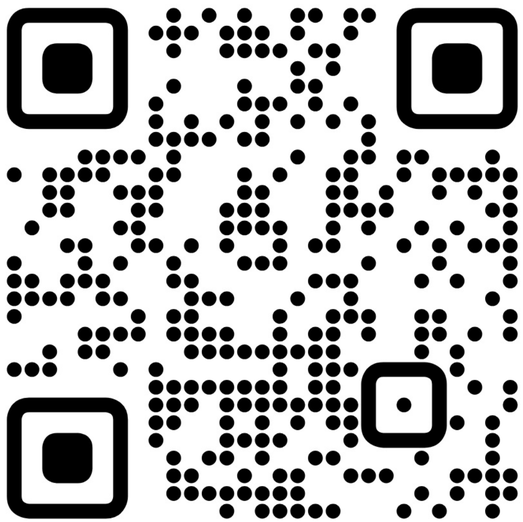


Show me some examples

```
shogun@panda:~/repo$ git commit -m "feat!: send an email to the customer when a product is shipped"
shogun@panda:~/repo$ git commit -m "feat: allow provided config object to extend other configs"
shogun@panda:~/repo$ git commit -m "fix: prevent racing of requests"
shogun@panda:~/repo$ git commit -m "chore!: drop support for Node 6"
shogun@panda:~/repo$ git commit -m "docs: correct spelling of CHANGELOG"
```

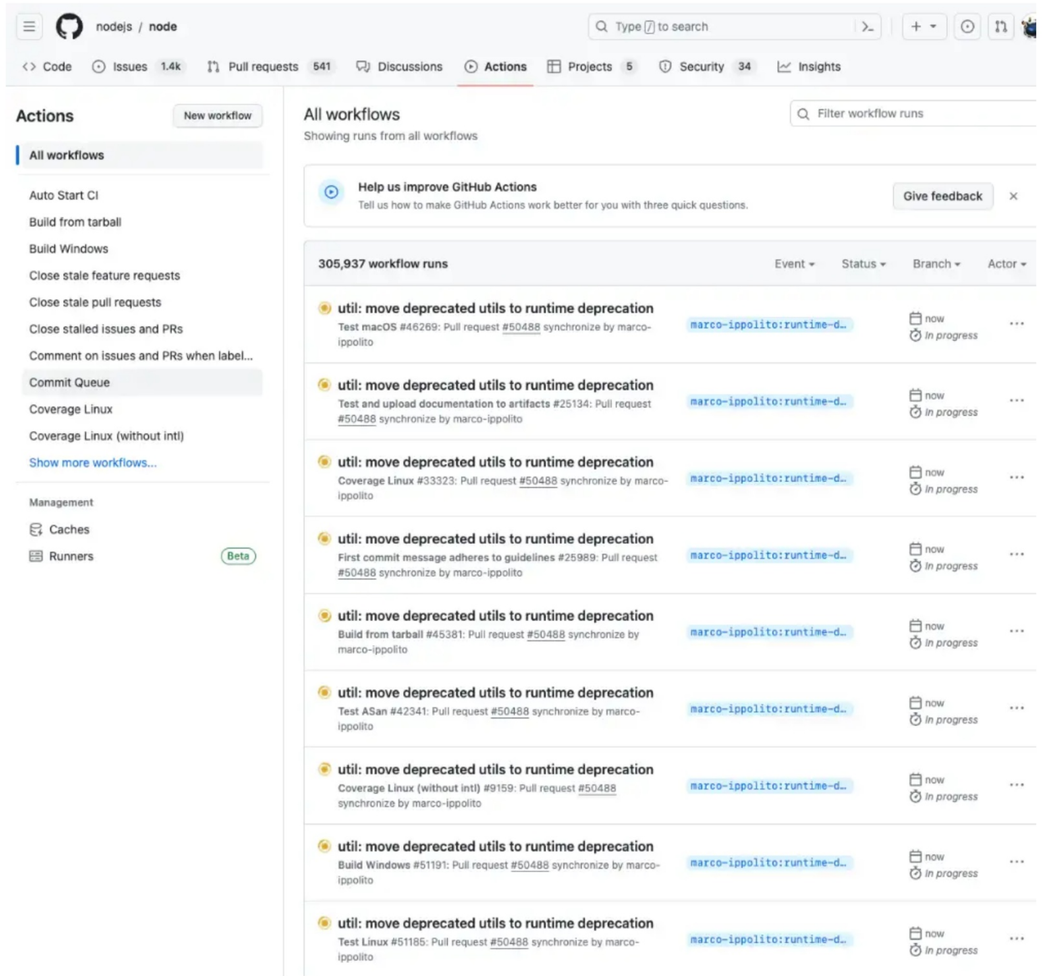
Semantic versioning

A specification for having meaningful software versions.
Check the **official website**!



(Semi) Automated releasing

By leveraging existing tools and the workflow feature you can release new version with just a click.



Questions?




Coffee Break

Humans are social beings



Let's talk about the community

Even if we're developing software, it's always about the people.

A line of code can be replaced at any time with no effort, you can't do the same with people.



Where do you store the code?



It's better to keep things separate
The management will be simplified.



Public, separate organizations are the best
They give you the visibility and ease of use you need.

Which accesses do you want to give?



Keep the accesses to the minimum

This is a general rule that should be followed as much as possible.



Make sure people cannot cut you out

It will be very hard to retake control.

Remember, you don't really own the code



It started with you ...
You own the software as whole.



... but it doesn't end with you ...
People are still owners and responsible for their contributions.



... so don't forget about others!
Contributors should have a say in relevant administrative tasks.

Diversity and Inclusion



Conscious and unconscious bias

We all suffer from these biases, there is simply no escape.

1

Conscious

We all have and use attitudes and heuristics.

2

Unconscious

More subtle because we don't control them.

Why do you want D&I?



Richness

Diversity is a pot of values.



New experiences

All people are having their lives ...



New perspectives

... and different point of views.

How do you enforce it? (1/2)

1

Be pro-active

You can't wait for things to happen to act.

2

Be receptive

Ask for suggestion from affected people.

How do you enforce it? (2/2)

3

Code of Conduct

This must be enforced from day zero.

4

Zero Tolerance

No exception allowed.

Everybody is responsible for moderation!



Last advices

1

Common sense

It might be surprising, but it is very underestimated and **never hurts**.

2

Yield control

You can't and neither want to control everything. **Don't burn out!**

3

Be genuine and honest

People will never fail to notice, appreciate and **love you**.

Questions?




Coffee Break

What about Node.js?



Questions?



One last thing™

“Money is just a way to keep score.
The best people in any field are motivated
by passion. That becomes more true
the higher the skill level gets.”

Eric Raymond



Thank you!

Paolo Insogna

Node.js TSC, Principal Engineer

@p_insogna

paolo.insogna@platformatic.dev

Platformatic