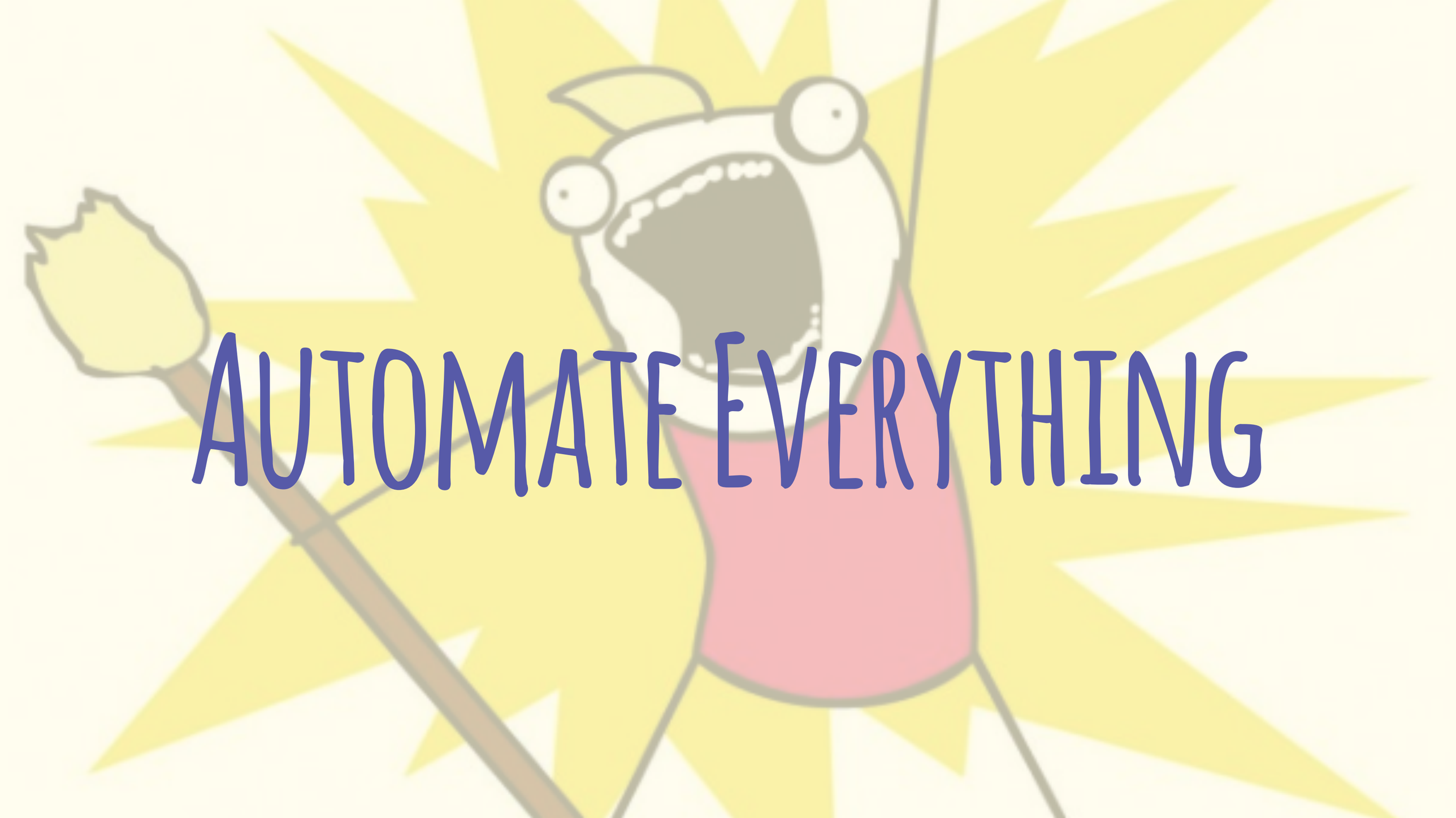

AUTOMATE EVERYTHING

Our tooling should serve us

- Reduce time to market
- Reduce development time
- Minimize toil
- Focus on the important stuff
- Make it easy to know what changed and why
- Reduce human error

What can we do

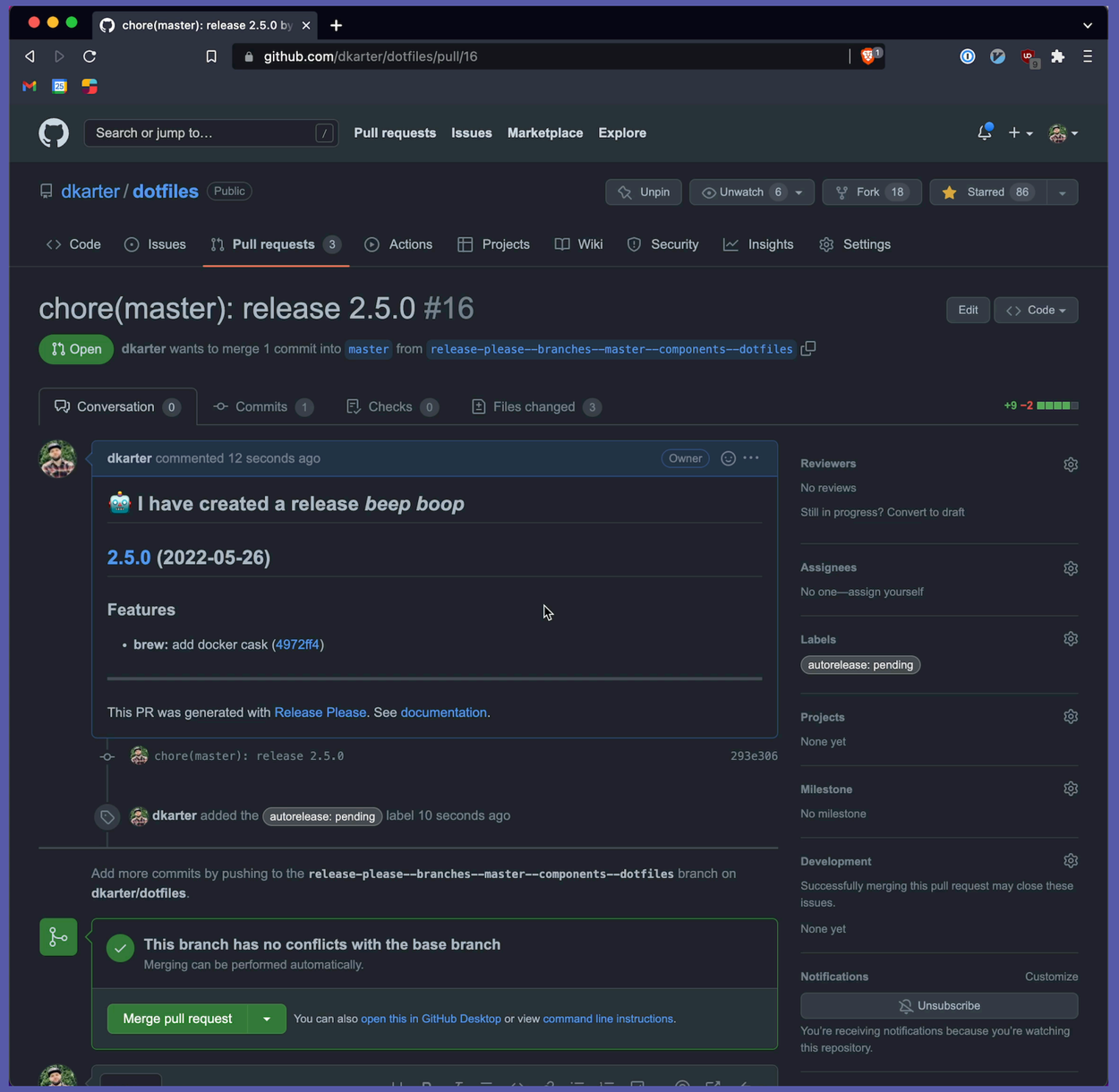
- Automate release branches
- Automate versioning
- Automate changelog updates
- Automate GitHub releases


CARTOON NETWORK
R

Release Please

- Project by Google
- Integrates with Github Actions (minimal config)
- Supports Elixir and JavaScript

Demo Time


chore(master): release 2.5.0 by
github.com/dkarter/dotfiles/pull/16
Search or jump to…
Pull requests
Issues
Marketplace
Explore
dkarter / dotfiles
Public
Unpin
Unwatch 6
Fork 18
Starred 86
Code
Issues
Pull requests 3
Actions
Projects
Wiki
Security
Insights
Settings
chore(master): release 2.5.0 #16
Edit
Code
Open
dkarter wants to merge 1 commit into master from release-please--branches--master--components--dotfiles
Conversation 0
Commits 1
Checks 0
Files changed 3
+9 −2
dkarter commented 12 seconds ago
Owner
I have created a release beep boop
2.5.0 (2022-05-26)
Features
brew: add docker cask (4972ff4)
This PR was generated with Release Please. See documentation.
chore(master): release 2.5.0
293e306
dkarter added the autorelease: pending label 10 seconds ago
Add more commits by pushing to the release-please--branches--master--components--dotfiles branch on dkarter/dotfiles.
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
Reviewers
No reviews
Still in progress? Convert to draft
Assignees
No one—assign yourself
Labels
autorelease: pending
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these issues.
None yet
Notifications
Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

How can we get there

Conventional Commits

The Conventional Commits specification is a lightweight convention on top of commit messages. It provides an easy set of rules for creating an explicit commit history; which makes it easier to write automated tools on top of.

This convention dovetails with SemVer, by describing the features, fixes, and breaking changes made in commit messages.

Commit Message Structure

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

Commit Message Structure

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

Examples

```
feat: adds v4 UUID to crypto

This adds support for v4 UUIDs to the library.

fix(utils): unicode no longer throws exception
  PiperOrigin-RevId: 345559154
  BREAKING-CHANGE: encode method no longer throws.
  Source-Link: googleapis/googleapis@5e0dcb2

feat(utils): update encode to support unicode
  PiperOrigin-RevId: 345559182
  Source-Link: googleapis/googleapis@e5eef86
```

Commit Types

There are multiple commit types, but here are the main ones:

Commit Types

There are multiple commit types, but here are the main ones:

- **fix:** patches a bug -> SemVer PATCH

Commit Types

There are multiple commit types, but here are the main ones:

- **fix:** patches a bug -> SemVer `PATCH`
- **feat:** introduces a new feature -> SemVer `MINOR`

Commit Types

There are multiple commit types, but here are the main ones:

- `fix:` patches a bug -> SemVer `PATCH`
- `feat:` introduces a new feature -> SemVer `MINOR`
- `feat!:` / `fix!:` breaking change -> SemVer `MAJOR`

Other Types

`build:`, `chore:`, `ci:`, `docs:`, `style:`, `refactor:`, `perf:`, `test:`

Don't affect versioning... (unless you add a `!`, e.g. `refactor!:`)

But my commits contain all kinds of stuff

Mindset shift

Git is your friend. Especially when it comes to tracking down a change that introduced a bug.

You want to know:
- What changed
- Why it changed
- What are the implications to users

Small, Atomic Commits

Commits only change one small thing and are easy to revert.

It gets easier with time

The more you do it the easier it becomes and there are tools out there that help make this very easy

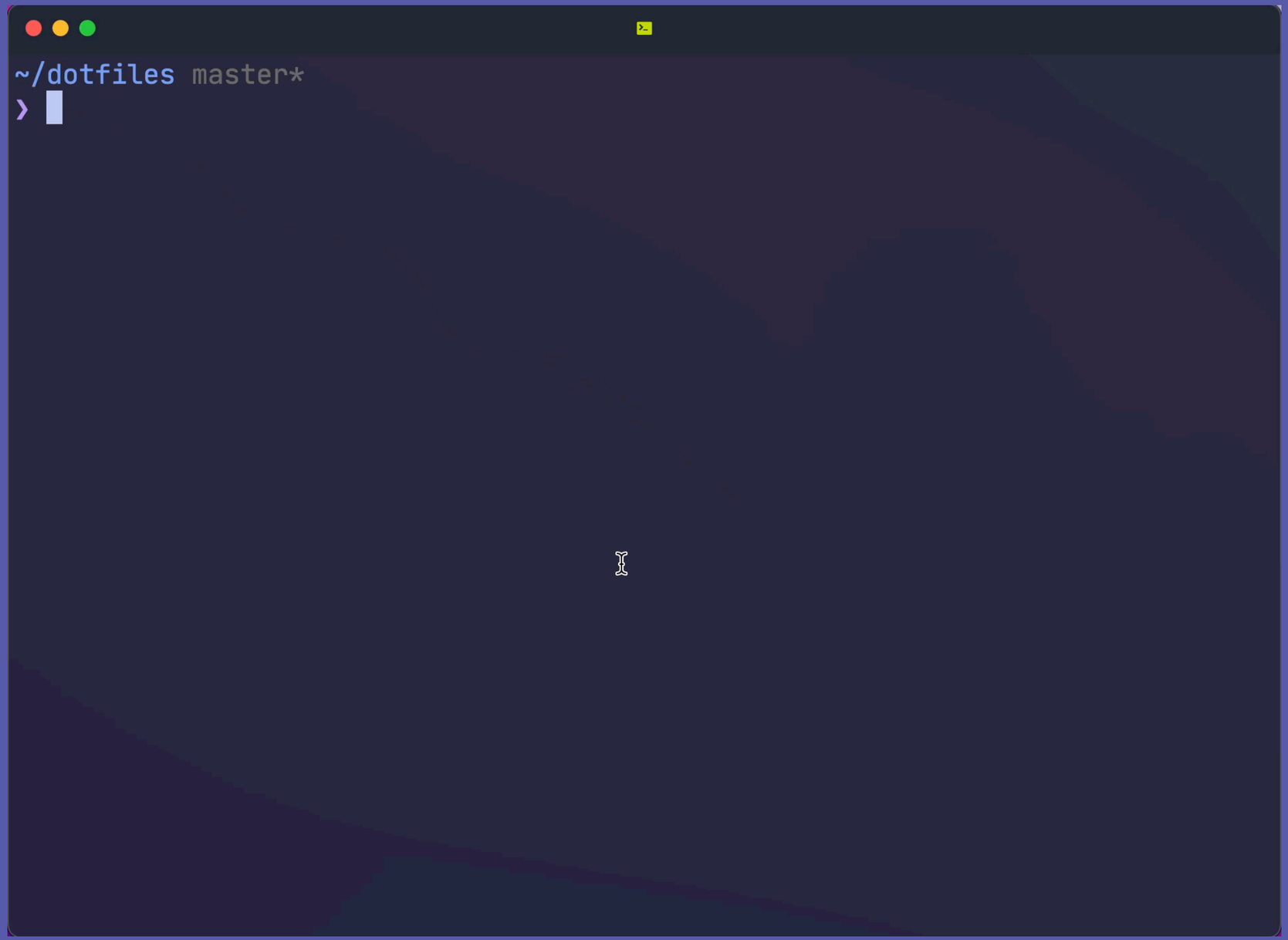
How

❌ Instead of

```
git add --all
```

✅ Use `--patch`

```
git add -p
```


~/dotfiles master*
❯

Husky + Commitlint

Will help integrate conventional commits into our workflow by letting you know after committing if the commit doesn't match the spec.

Resources (links)

- Conventional Commits Spec
- Husky + Commitlint setup
- Release Please release automation
- Release Please GH Action
- git add --patch docs
- conventional commits plugin for VSCode