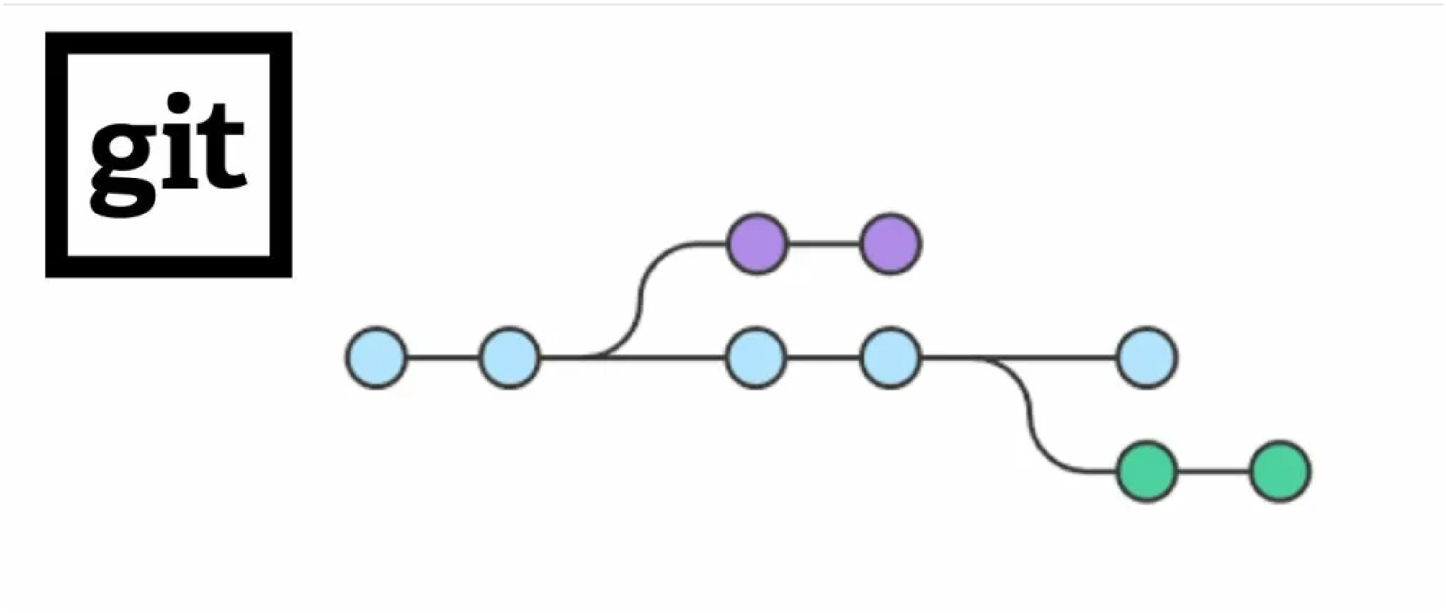

varbSan
Posted on May 19, 2022

A Simplified Convention for Naming Branches and Commits in Git

#git #beginners #programming #github

There are many excellent naming conventions regarding git branches and commits.

But what if you want something very lean and simple?

Here is a proposition.

Branch Naming Convention

The Git Branching Naming Convention article is an excellent base.
However, you can simplify even more.

Category

A git branch should start with a category. Pick one of these: `feature`, `bugfix`, `hotfix`, or `test`.

- `feature` is for adding, refactoring or removing a feature
- `bugfix` is for fixing a bug

- `hotfix` is for changing code with a temporary solution and/or without following the usual process (usually because of an emergency)
- `test` is for experimenting outside of an issue/ticket

Reference

After the category, there should be a " `/` " followed by the reference of the issue/ticket you are working on. If there's no reference, just add `no-ref`.

Description

After the reference, there should be another " `/` " followed by a description which sums up the purpose of this specific branch. This description should be short and "kebab-cased".

By default, you can use the title of the issue/ticket you are working on. Just replace any special character by " `-` ".

To sum up, follow this pattern when branching:

```
git branch <category/reference/description-in-kebab-case>
```

Examples:

- You need to add, refactor or remove a feature: `git branch feature/issue-42/create-new-button-component`
- You need to fix a bug: `git branch bugfix/issue-342/button-overlap-form-on-mobile`
- You need to fix a bug really fast (possibly with a temporary solution): `git branch hotfix/no-ref/registration-form-not-working`
- You need to experiment outside of an issue/ticket: `git branch test/no-ref/refactor-components-with-atomic-design`

Commit Naming Convention

For commits, you can combine and simplify the Angular Commit Message Guideline and the Conventional Commits guideline.

Category

A commit message should start with a category of change. You can pretty much use the following 4 categories for everything: `feat`, `fix`, `refactor`, and `chore`.

- `feat` is for adding a new feature
- `fix` is for fixing a bug
- `refactor` is for changing code for peformance or convenience purpose (e.g. readibility)
- `chore` is for everything else (writing documentation, formatting, adding tests, cleaning useless code etc.)

After the category, there should be a "`:`" announcing the commit description.

Statement(s)

After the colon, the commit description should consist in short statements describing the changes.

Each statement should start with a verb conjugated in an imperative way. Statements should be seperated from themselves with a "`;`".

To sum up, follow this pattern when committing:

```
git commit -m '<category: do something; do some other things>'
```

Examples:

- `git commit -m 'feat: add new button component; add new button components to templates'`
- `git commit -m 'fix: add the stop directive to button component to prevent propagation'`
- `git commit -m 'refactor: rewrite button component in TypeScript'`
- `git commit -m 'chore: write button documentation'`

References

sources

- article: Git Branching Name Convention
- article: Conventional Commits 1.0.0
- article: Commit Message Guideline
- article: A Successful Git Branching Model

Top comments (0)

Code of Conduct • Report abuse

varbSan

Full Stack Developer in Berlin 👨‍💻 Green Tech Industry 🌱 🚀 Learn in public! 👨‍🎓

LOCATION
Berlin

WORK
Full Stack Developer

JOINED
Jul 18, 2021

More from varbSan

What is HTTP? (... for newbies)
#beginners #webdev