

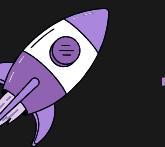


KHARAGPUR OPEN SOURCE SOCIETY

GIT & GITHUB WORKSHOP 2025



WHO ARE WE?



We are a group of Open Source Enthusiasts who share something more fundamental, '**a love for coding**'.

Every year KOSS holds events to familiarize students with Git and GitHub, Google Summer of Code, as well as conducts workshops on topics such as Linux Installation and Cybersecurity.



>> OVERVIEW

- Terminal
-

- Version Control
-

- What is Git?
-

- Git Commands
-

- Branching
-

- What is GitHub?
-

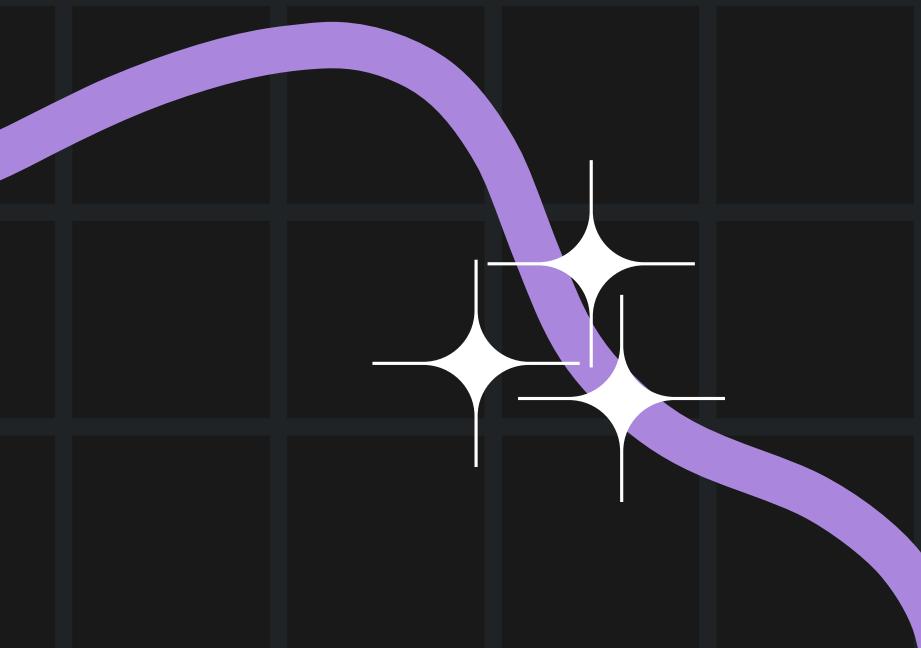
- Pull Requests
-



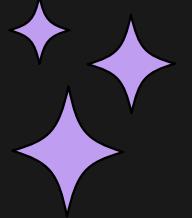
THE TERMINAL

- What is a Terminal?

```
└─⚙️>🏠~ ━━━━━━ ⚙️ impure < 11:48:39 PM IST ━
 fortune | lolcat
timesharing, n:
    An access method whereby one computer abuses many people.
```

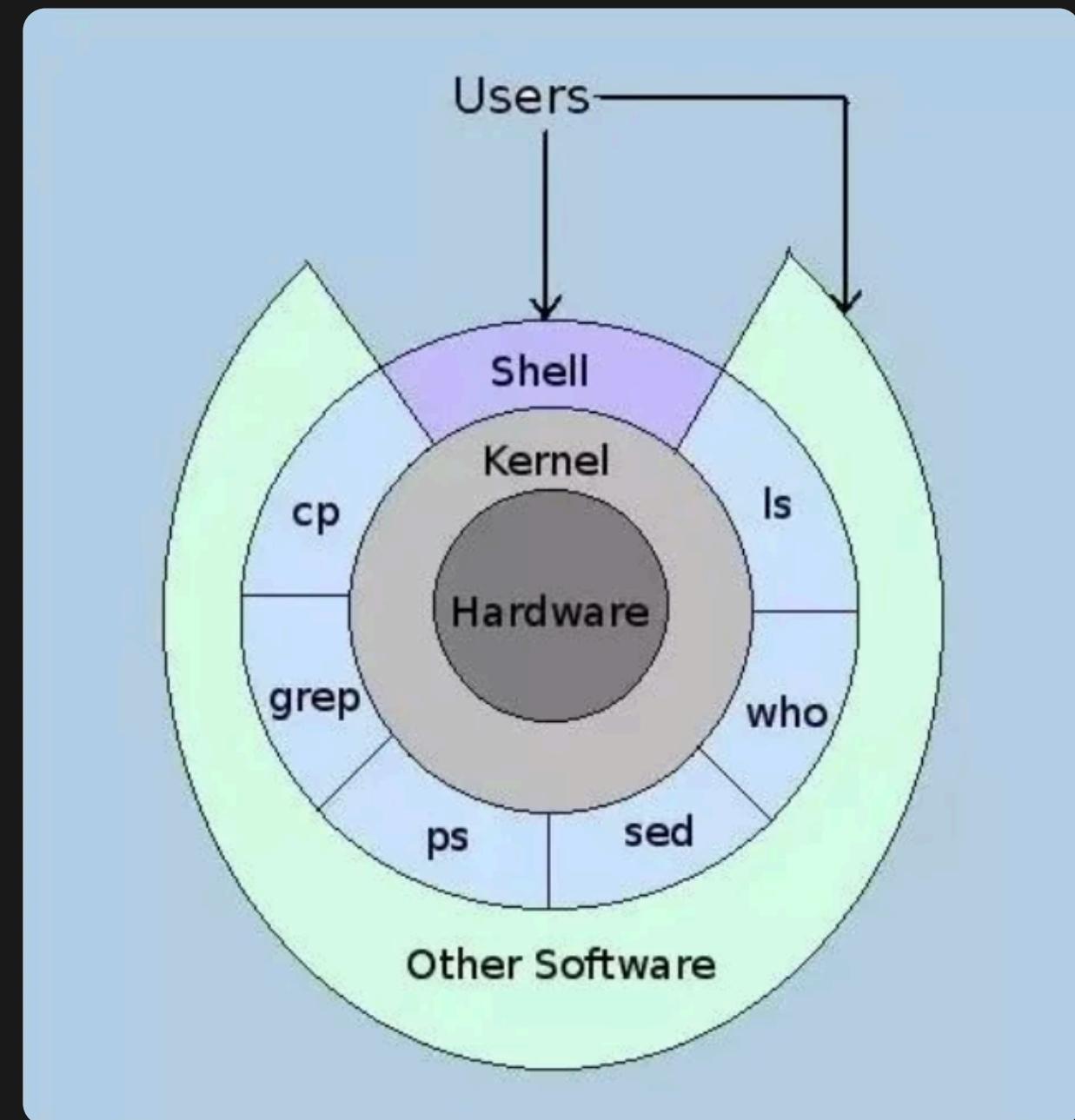


THE TERMINAL



> What is a Terminal?

- A **text-based interface** to give instructions to your computer
- Faster and more powerful than only using GUIs



VERSION CONTROL

➤ *What is Version Control?*

- Version Control or source control is the process of keeping track of **changes** made to software code
- Version Control System tools help software teams **manage and organize** these changes over time

GIT

Git is a **free and open source** version control system, used for source code management.

Git was originally created by **Linus Torvalds**, (who also happens to be the creator of Linux) for version control in the development of the Linux kernel.



In British slang, "git" means an unpleasant person (roughly like "jerk" or "idiot").

According to official docs. GIT can be '*anything*' depending on your mood

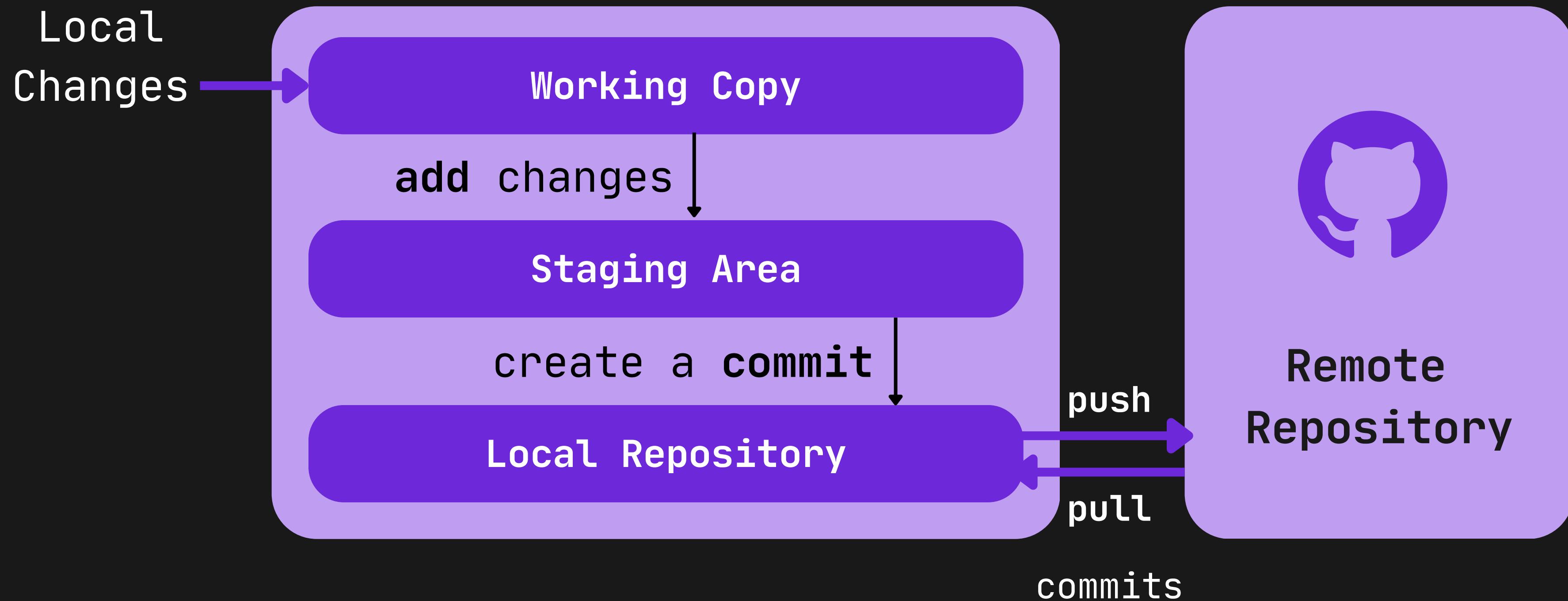
Brain:
"Ah yes,
Global Information
Tracker"



Linus:
"No. It means
I'm a Git."



GIT WORKFLOW



INITIALISING A REPOSITORY

INITIALISATION:

```
$ cd dir-name  
$ git init
```

```
⚙️ ➜ ~/Development/builds-systems/nixos-scripts > git master ↑1  
↳ tree -L 1 .git  
.git  
├── COMMIT_EDITMSG  
├── config  
├── description  
├── HEAD  
└── hooks  
├── index  
├── info  
├── logs  
└── refs
```

GIT COMMANDS

\$ git add <path>

adds the specified files to the staging area

\$ git status

shows files in current working directory and staging area

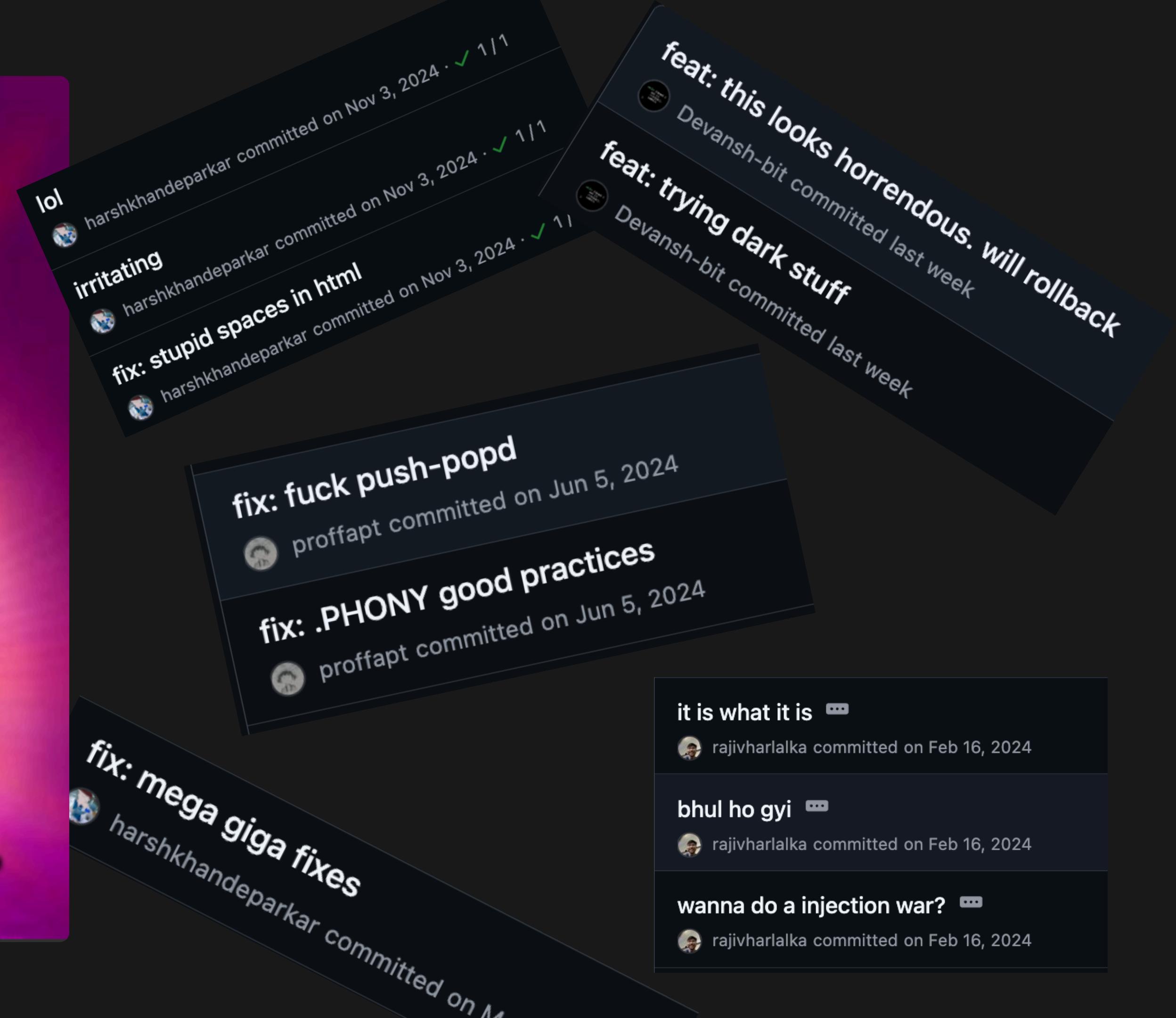
\$ git commit -m "message"

creates a commit object with the added changes

AFTER

+8,731 -16,049

git commit -m "minor changes"



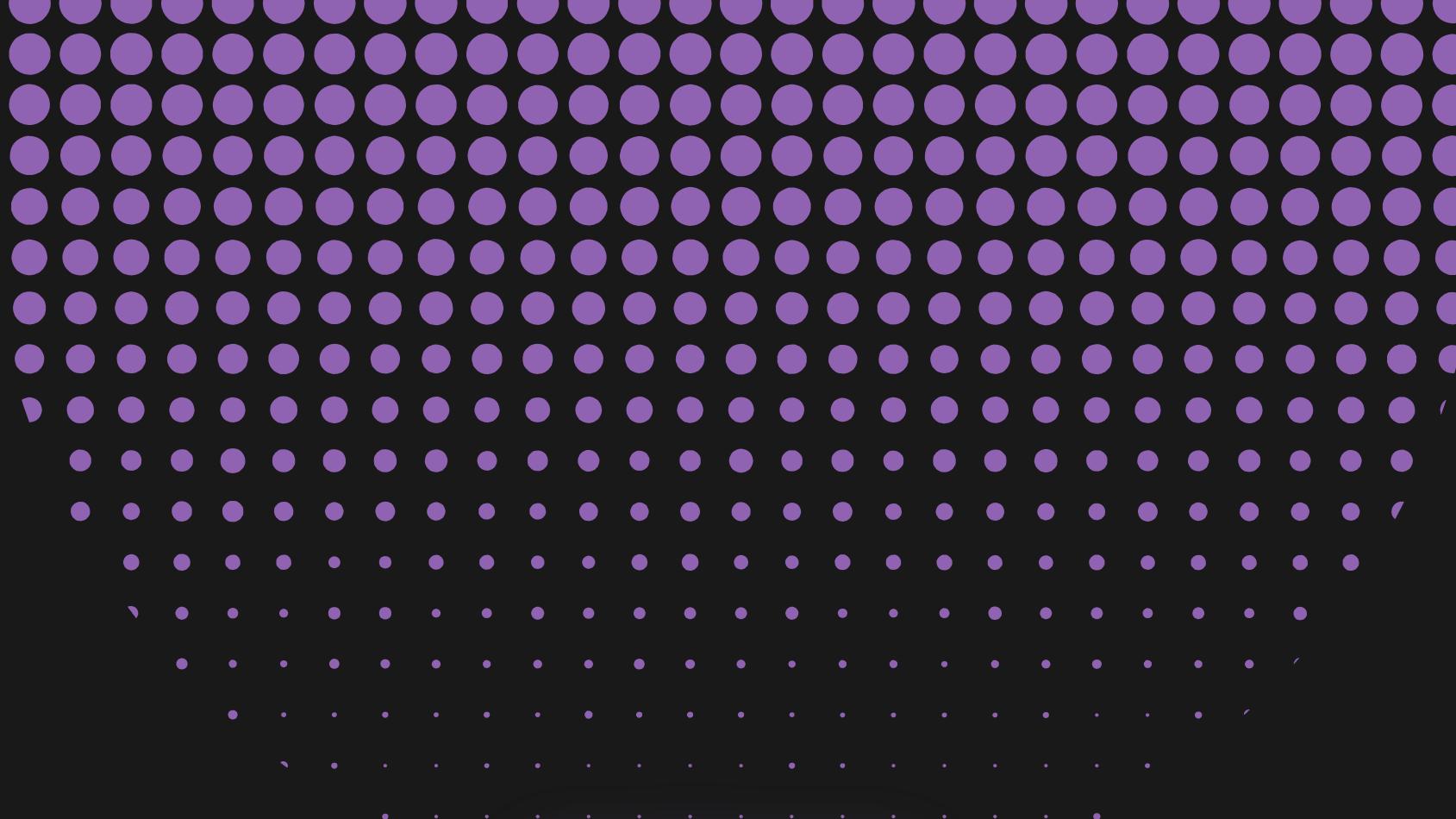
COMMIT MESSAGE

The commit message should succinctly describe (to you and others) what changes have been made in the commit.

Examples:

feat: add user login and sign up

fix: cart is persistent across different tabs



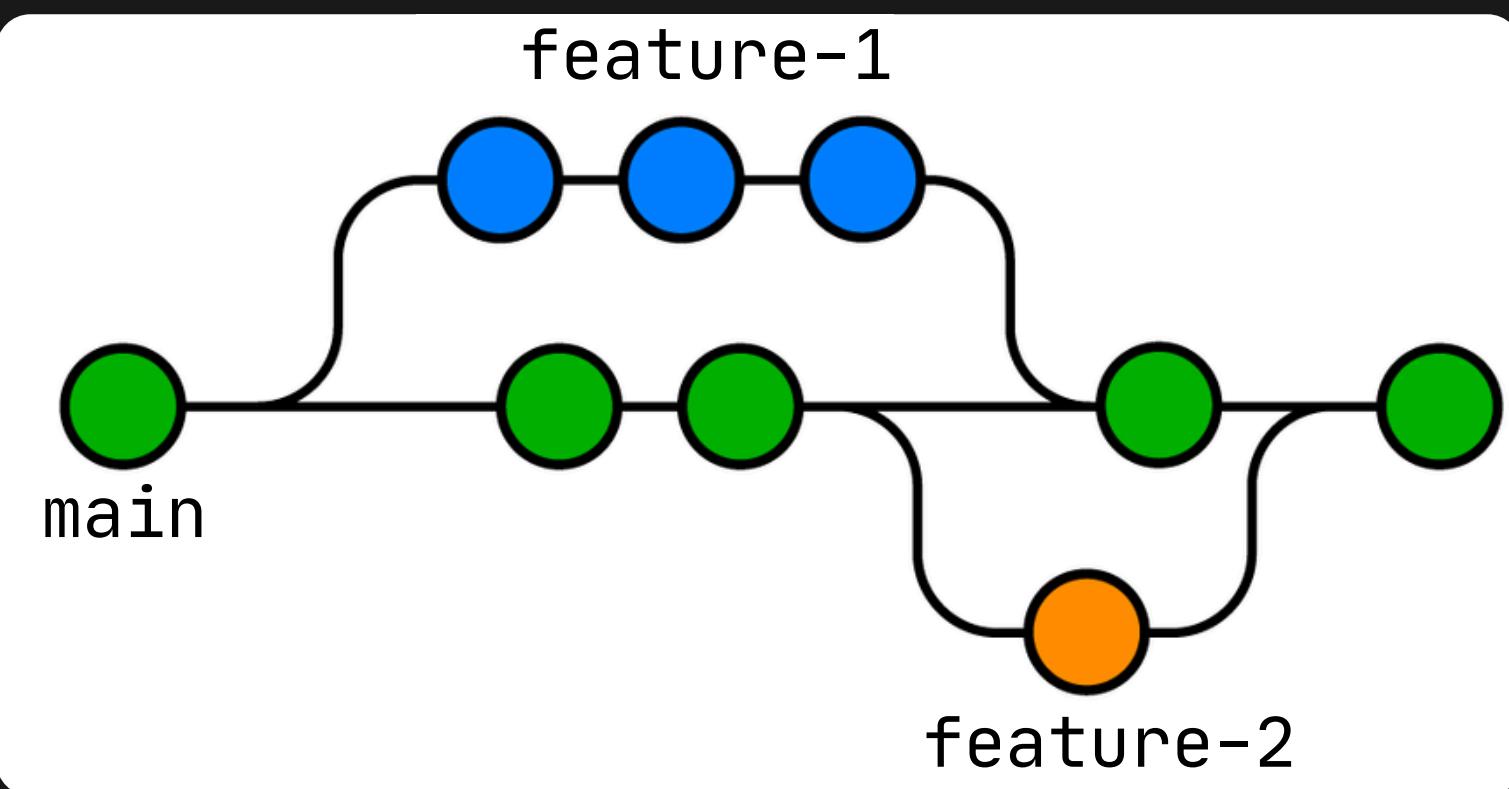
DEMO

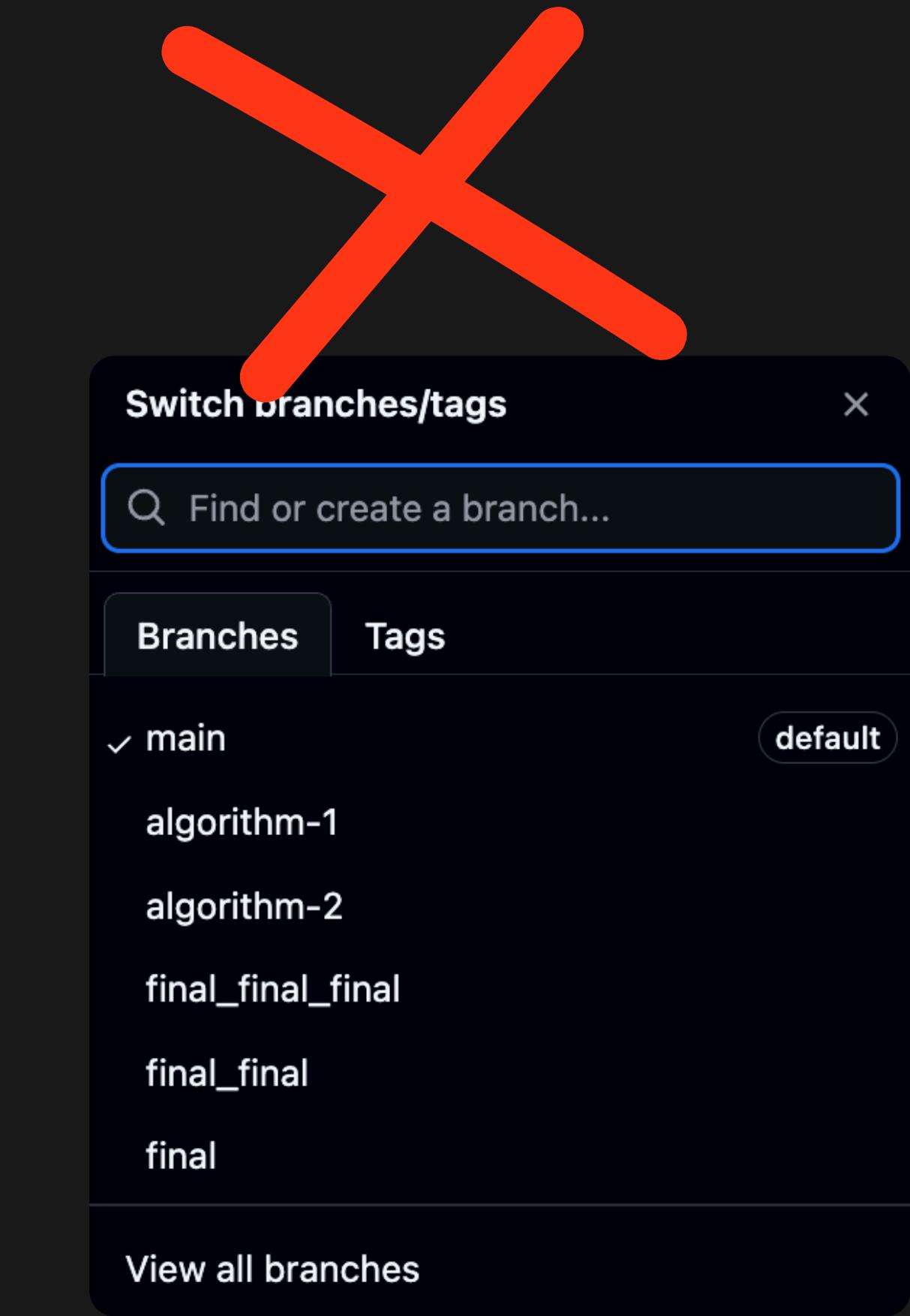
BRANCHING

In a git repository, you can create multiple '**branches**', which are different versions of the source code.

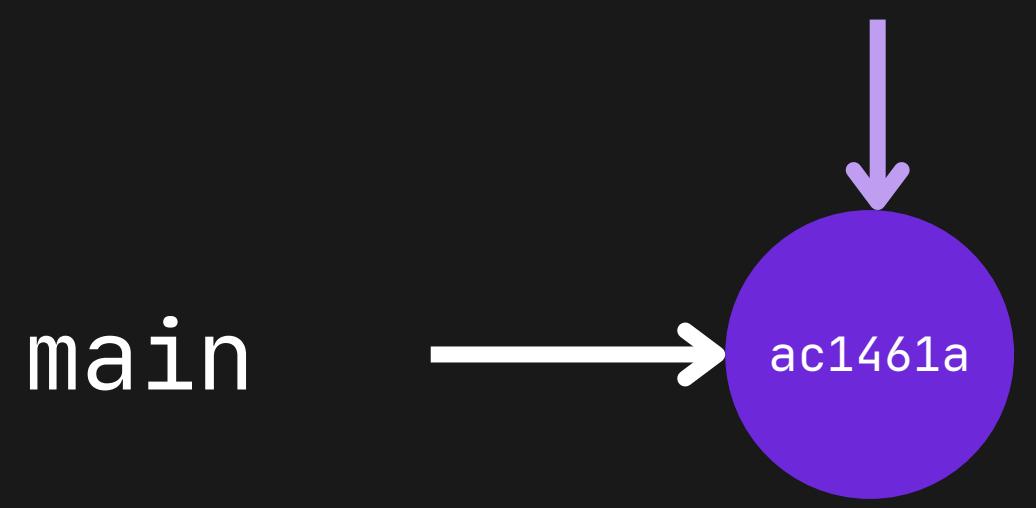
Branches allow you to work on **different tasks** in the repository **separately**, without affecting the main version of the code.

Changes made in the branches can be **combined** with the main code, when it is ready.



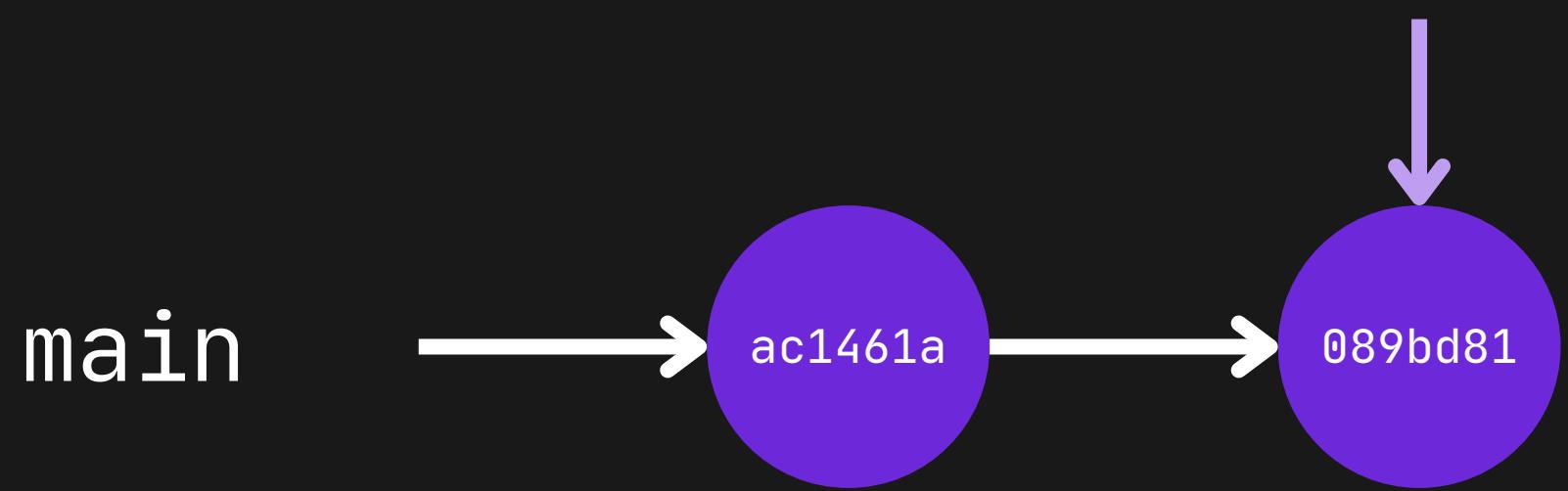


(main) \$

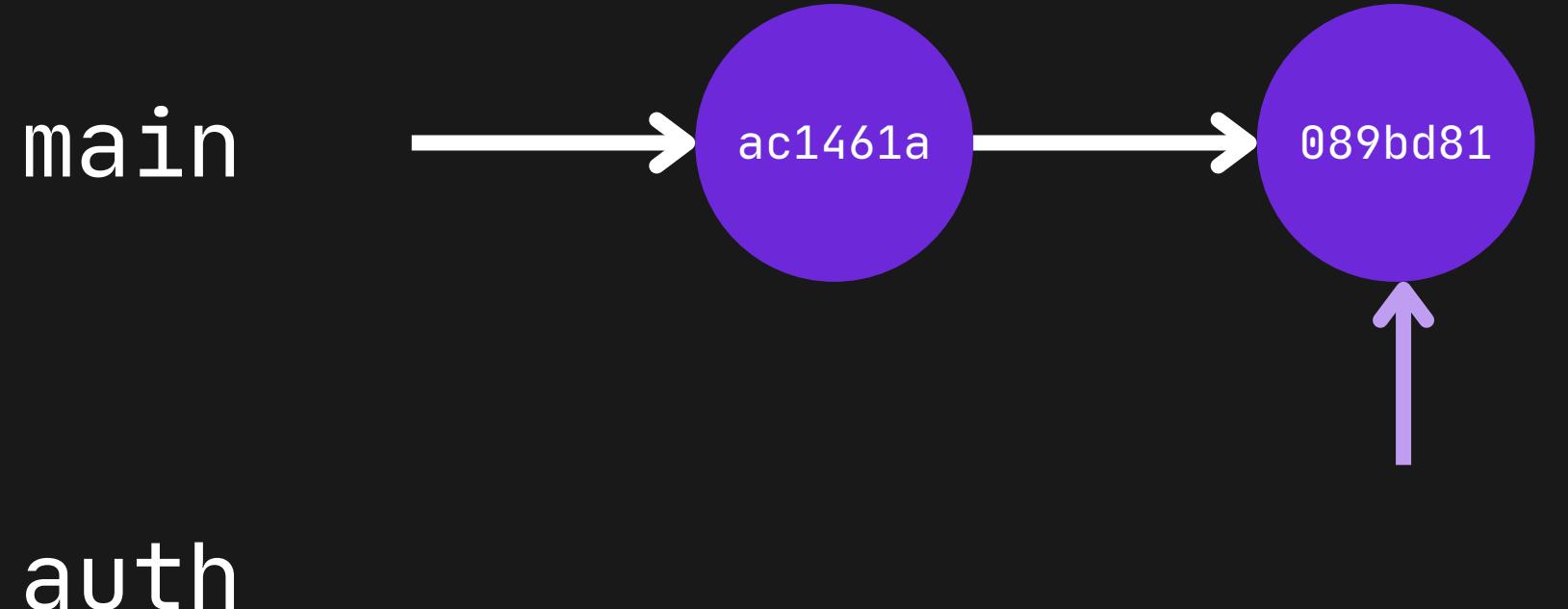


```
(main) $ git add .
```

```
(main) $ git commit -m "feat: add about us page"
```

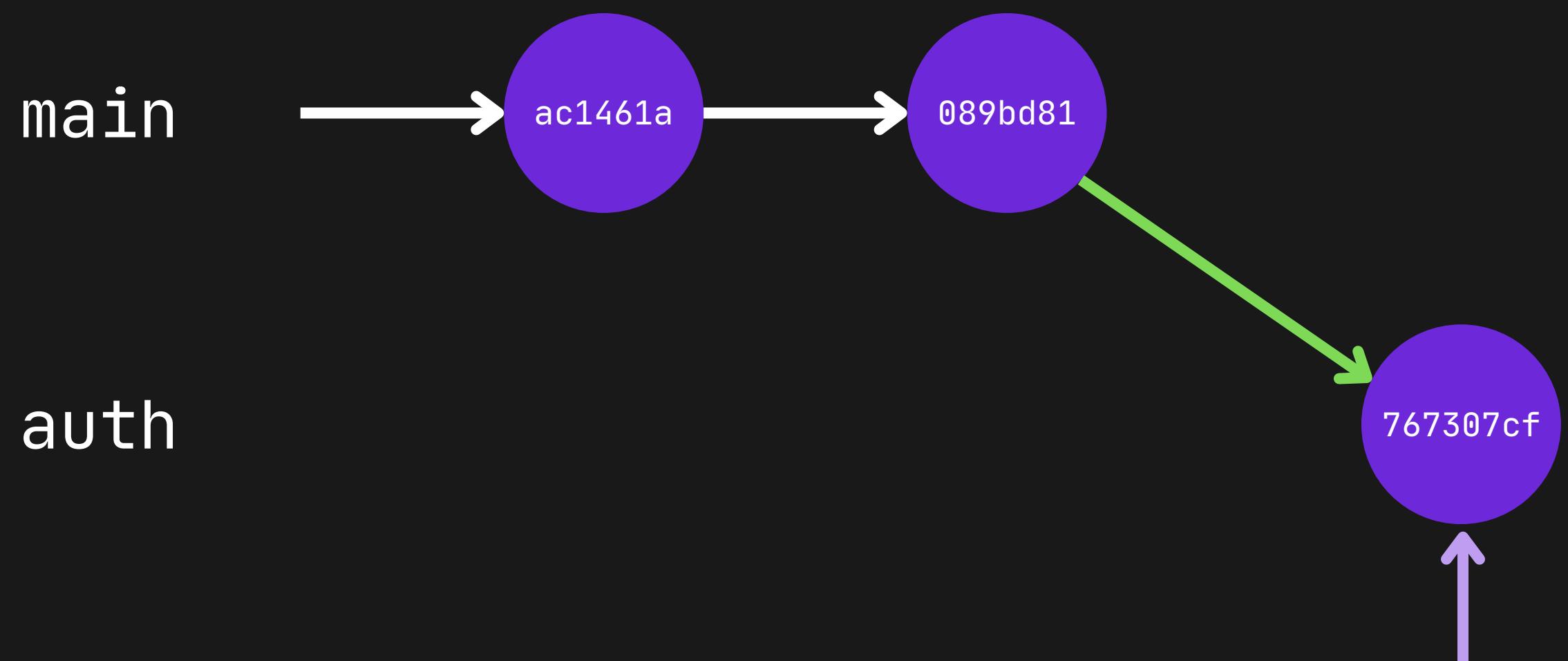


```
(main) $ git branch auth  
(main) $ git checkout auth
```

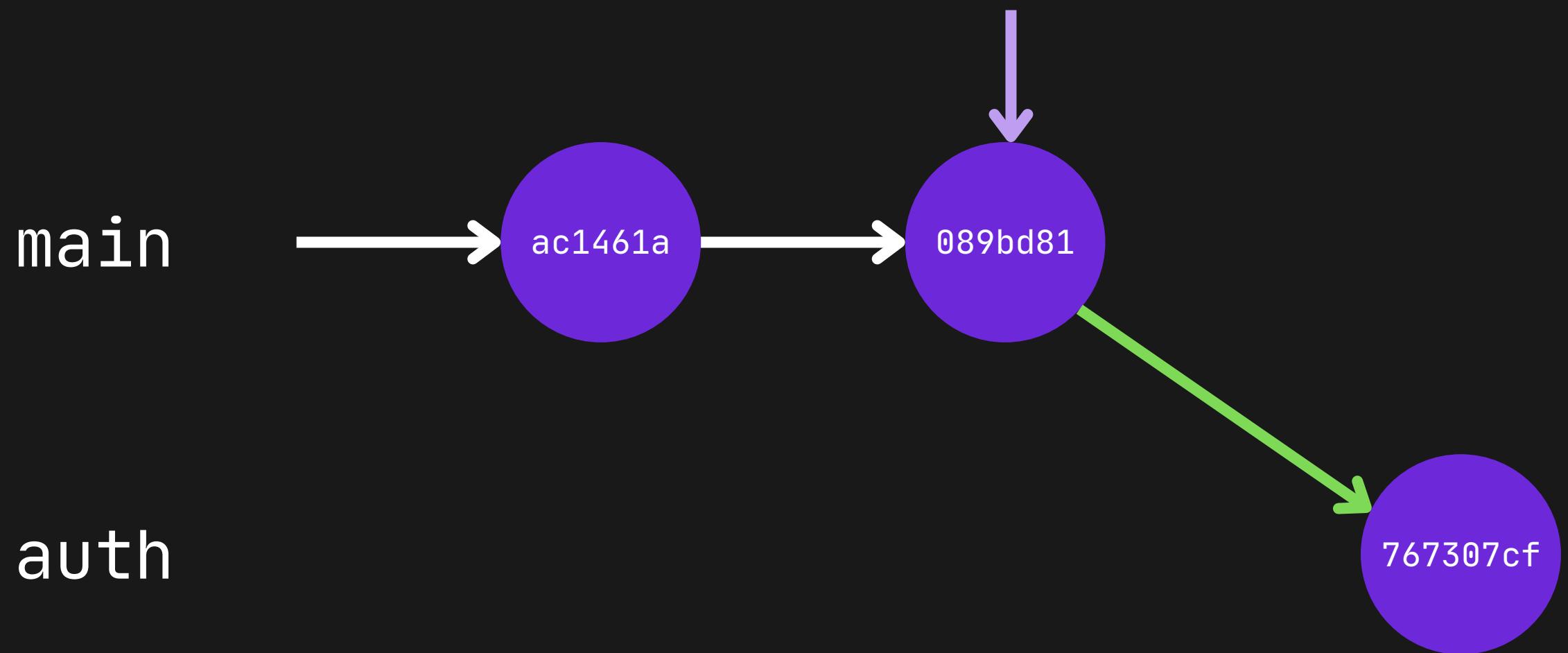


```
(auth) $ git add .
```

```
(auth) $ git commit -m "feat: implement login page"
```

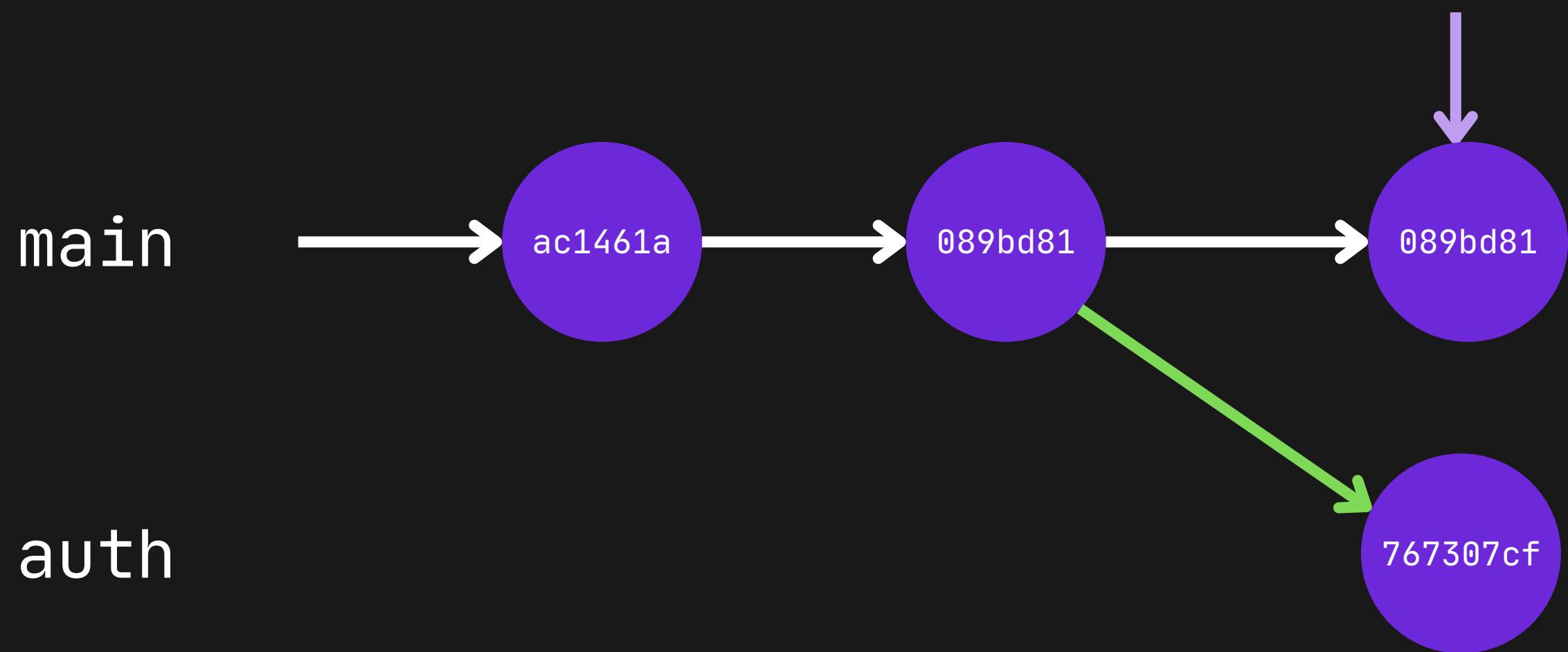


(auth) \$ git checkout main

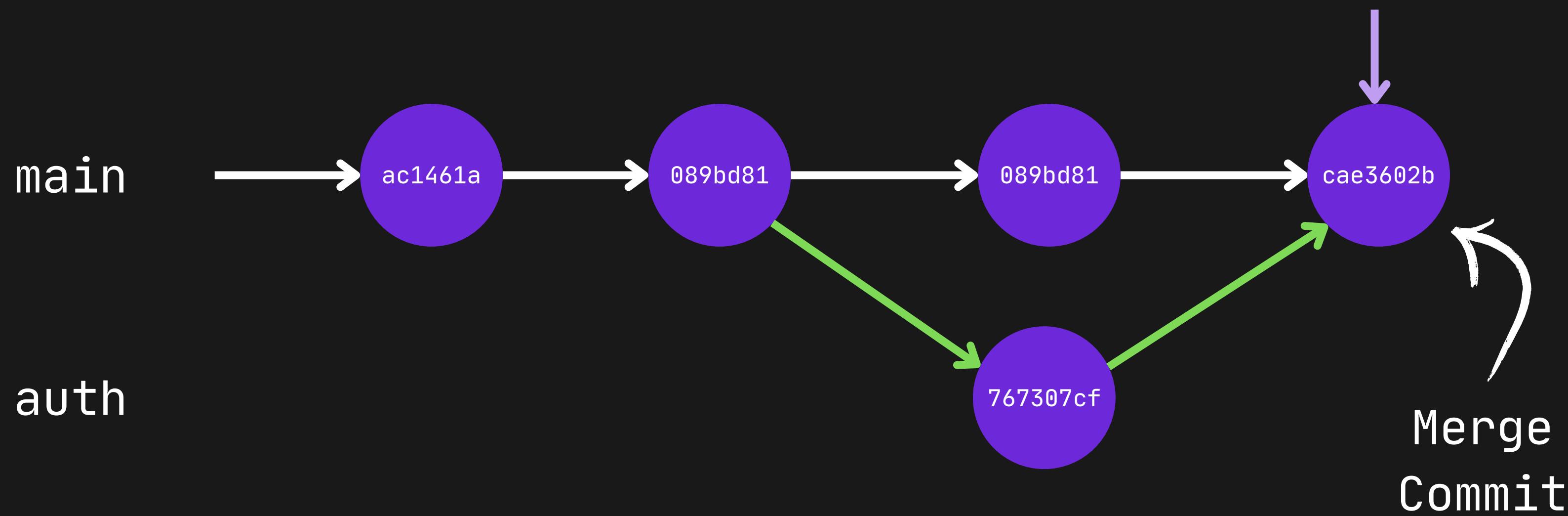


```
(main) $ git add .
```

```
(main) $ git commit -m "fix: typo"
```



(main) \$ git merge auth



Merge Conflicts

While performing the merge, there may be some conflicts in the code, if e.g. the same file has been changed both in the current branch, and in the incoming branch. If this happens, the user has to fix the merge conflicts, before the merge commit can be created.

```
1 Hello, World!
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
2 <<<<< HEAD (Current Change)
3 My favourite fruit is a mango.
4 =====
5 My favourite fruit is an apple.
6 >>>>> superior (Incoming Change)
7 |
```

DOUBTS?

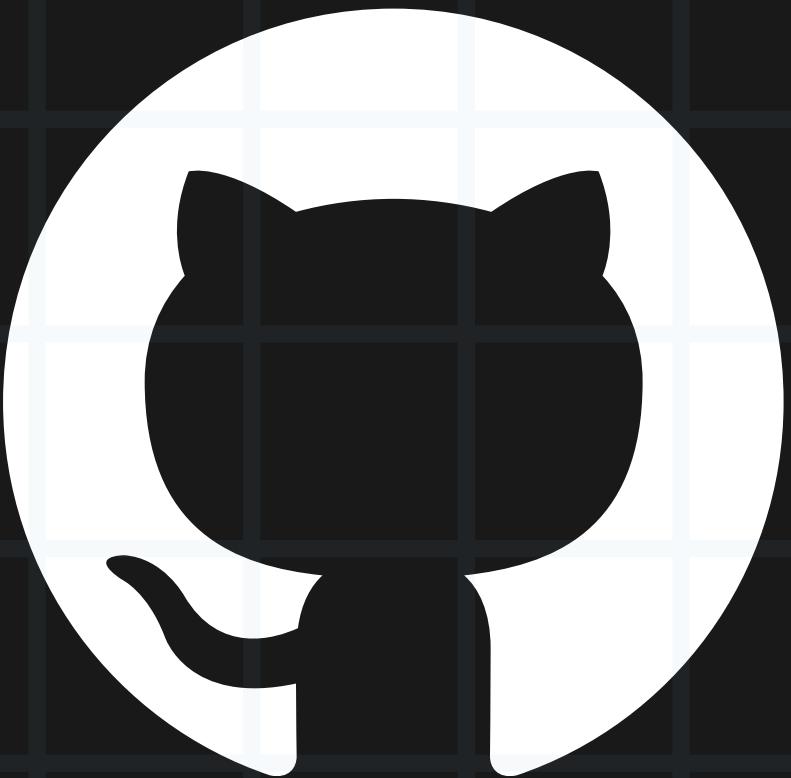


GITHUB

GitHub is a platform that allows users to manage their git repositories.

It is used commonly to host open source projects.

GitHub allows developers to collaborate on a project from anywhere.



CREATING A REPO

The screenshot shows the GitHub 'Create a new repository' interface. At the top, there's a navigation bar with 'Overview', 'Repositories 80', 'Projects', 'Packages', 'Stars 51', a search bar 'Find a repository...', and dropdown menus for 'Type', 'Language', 'Sort'. A green 'New' button is highlighted with a purple arrow. Below the bar, the main form starts with a 'General' section. It has fields for 'Owner *' (set to 'saksham-kumar-14') and 'Repository name *' (containing 'coolrepo'). A note says 'coolrepo is available.' To the right of this field, a purple arrow points down to the text 'Name of your repo'. The 'General' section also includes a 'Description' field with placeholder text 'about project ...' and character count '17 / 350 characters'. Below the general section is a 'Configuration' section with options for 'Choose visibility *' (set to 'Public'), 'Add README' (with a toggle switch set to 'Off'), 'Add .gitignore' (set to 'No .gitignore'), and 'Add license' (set to 'No license'). At the bottom is a large green 'Create repository' button.

Create a new repository Preview Switch back to classic experience

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

1 General

Owner * Repository name *

saksham-kumar-14 / coolrepo

coolrepo is available.

Great repository names are short and memorable. How about [super-duper-happiness?](#)

Description

about project ...

17 / 350 characters

2 Configuration

Choose visibility * Public

Choose who can see and commit to this repository

Add README

READMEs can be used as longer descriptions. [About READMEs](#)

Off

Add .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

No .gitignore

Add license

Licenses explain how others can use your code. [About licenses](#)

No license

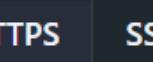
Create repository

Name of your repo

click this!

CREATING A REPO

Quick setup — if you've done this kind of thing before

 Set up in Desktop or   <https://github.com/dipamsen/my-repo.git> 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# my-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/dipamsen/my-repo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/dipamsen/my-repo.git
git branch -M main
git push -u origin main
```



use these commands to push an existing git repo to GitHub

MAKING CHANGES TO A REPO

The beauty of Open Source Software is that anyone can contribute to it!

To contribute to a repo (which you don't have write-access to):

1. **Fork** the Repo to your GitHub account
2. **Clone** the fork locally
3. Make changes to your local repository (on a different branch)
4. Create a **Pull Request** from your branch to the repo.
5. Wait for the maintainers to review and merge your Pull Request! :)

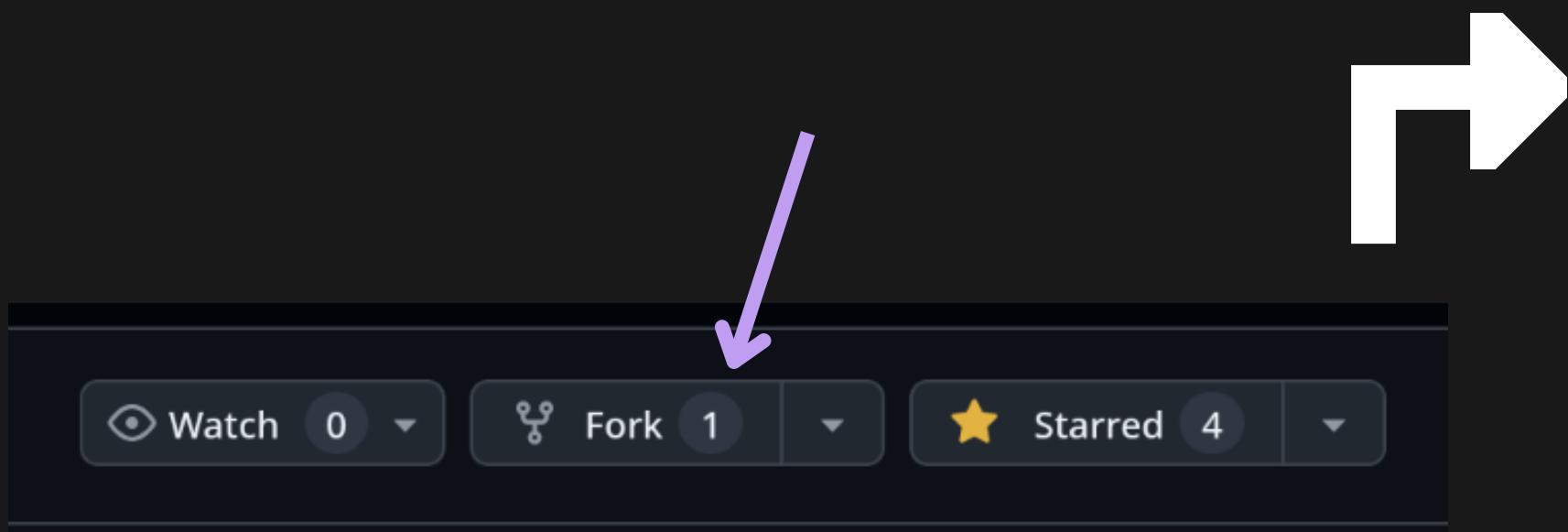
ISSUES



- Issues can represent a wide range of tasks, including bug reports, feature requests, enhancements, documentation needs, or general project tasks.
- Issues can be assigned labels - '*good first issues*', '*bug*', '*enhancement*'.
- Other features -
 - Assignees
 - Comments
 - References



FORKING



Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * **Repository name ***

AnshumaanMishra / reactplusplus
reactplusplus is available.

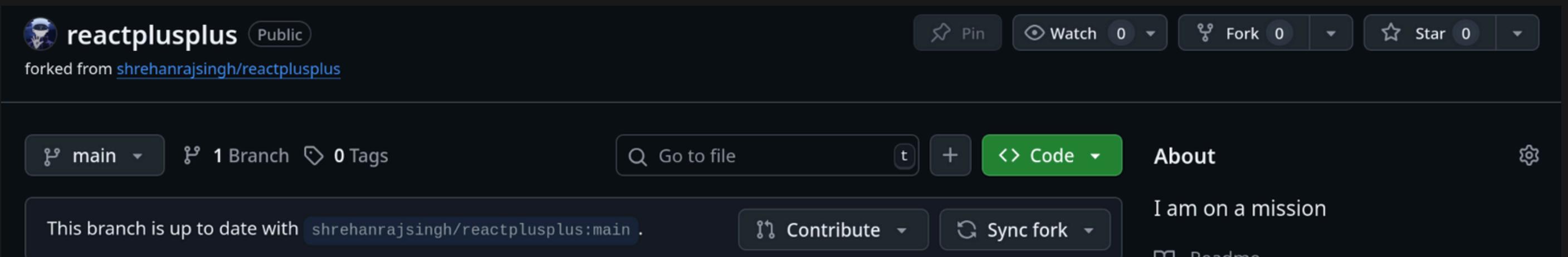
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)
I am on a mission

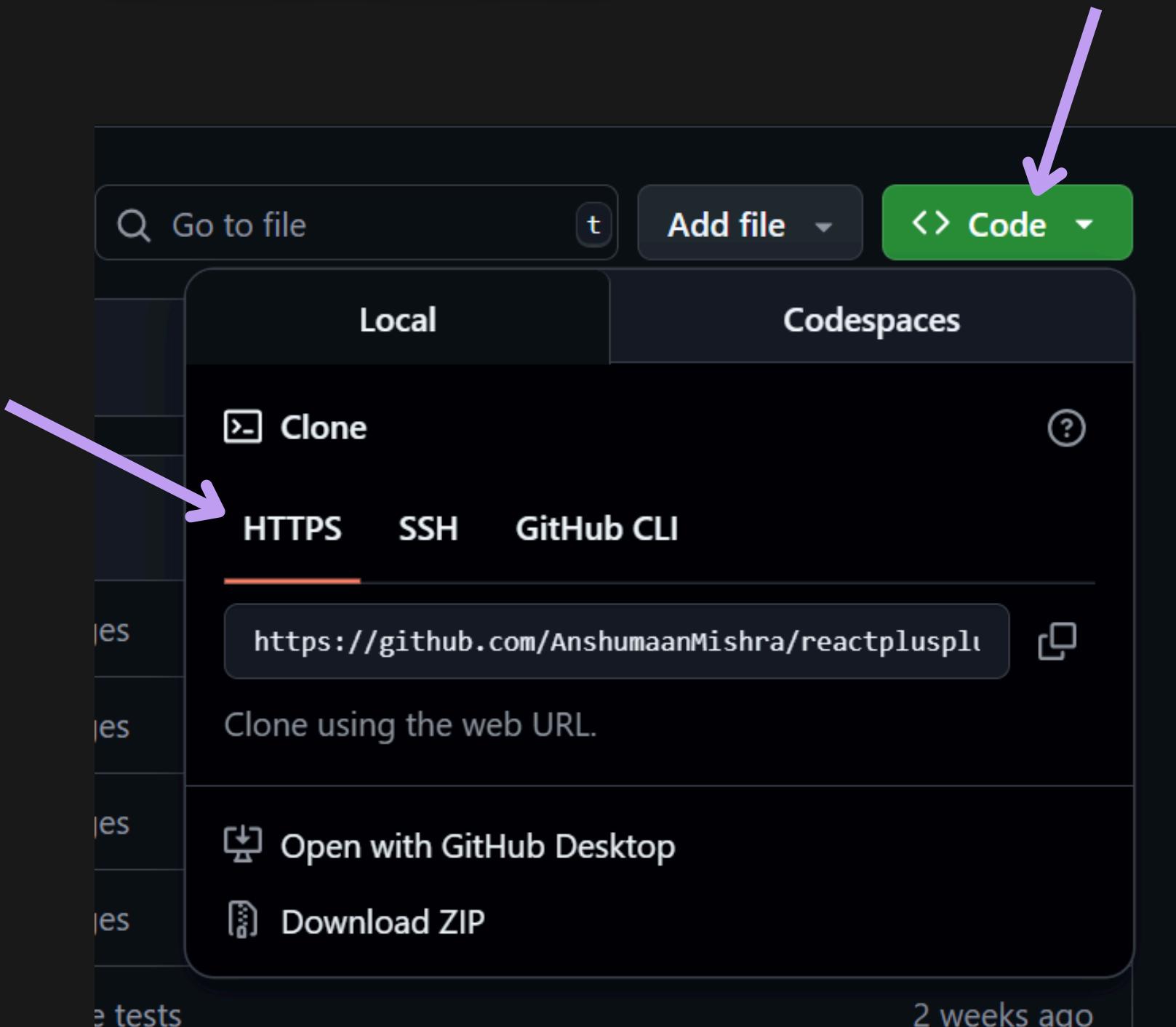
Copy the main branch only
Contribute back to shrehanrajsingh/reactplusplus by adding your own branch. [Learn more](#).

ⓘ You are creating a fork in your personal account.

Create fork



CLONING



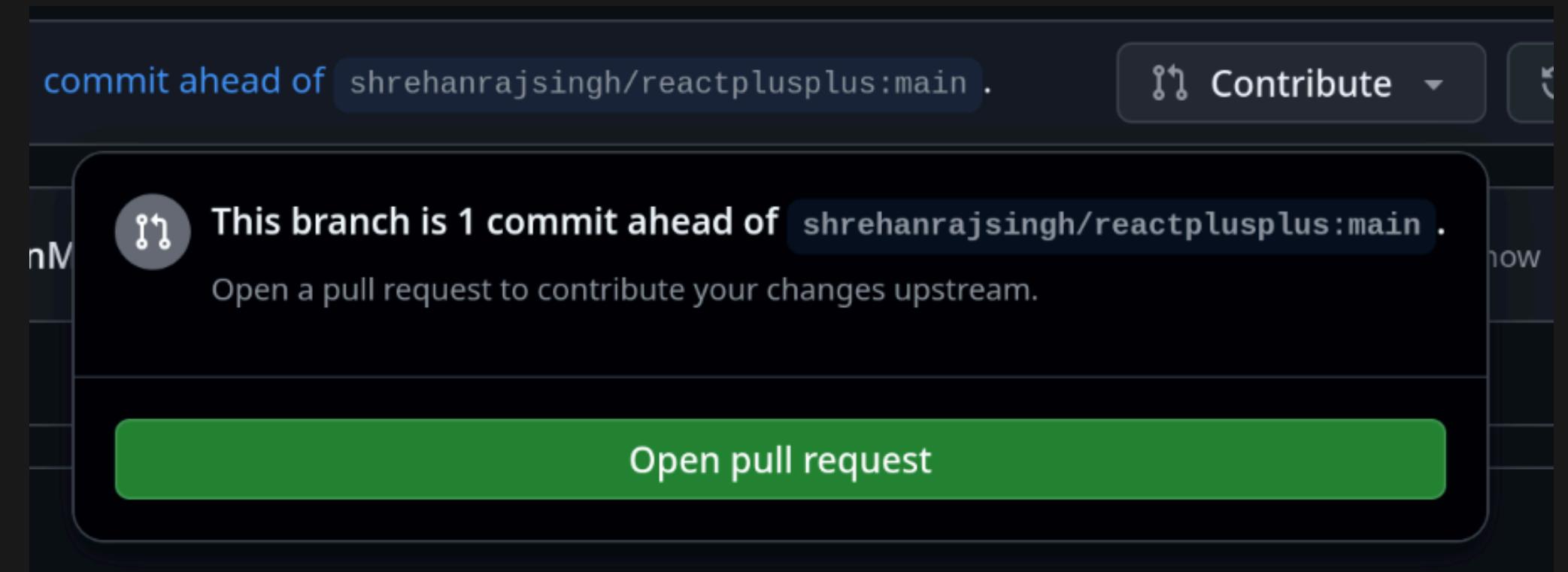
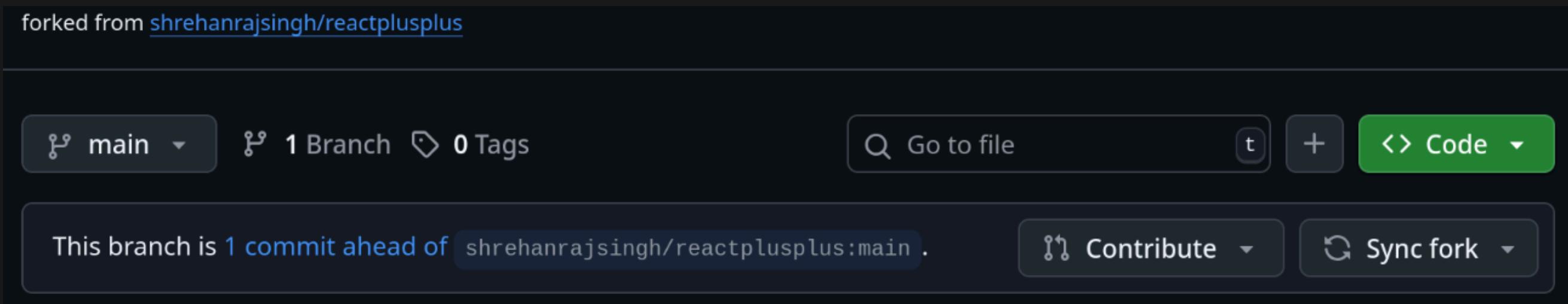
```
$ git clone <url>
```

Clones (downloads) a repository from GitHub to your computer.

A directory will be created with the same name as the repo. So to open the repo in your terminal, use

```
$ cd <repo-name>
```

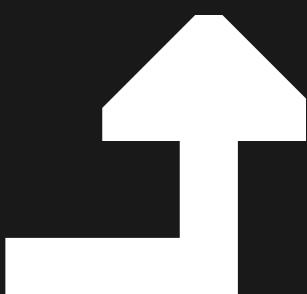
PULL REQUESTS



PULL REQUESTS

Give your PR a descriptive title

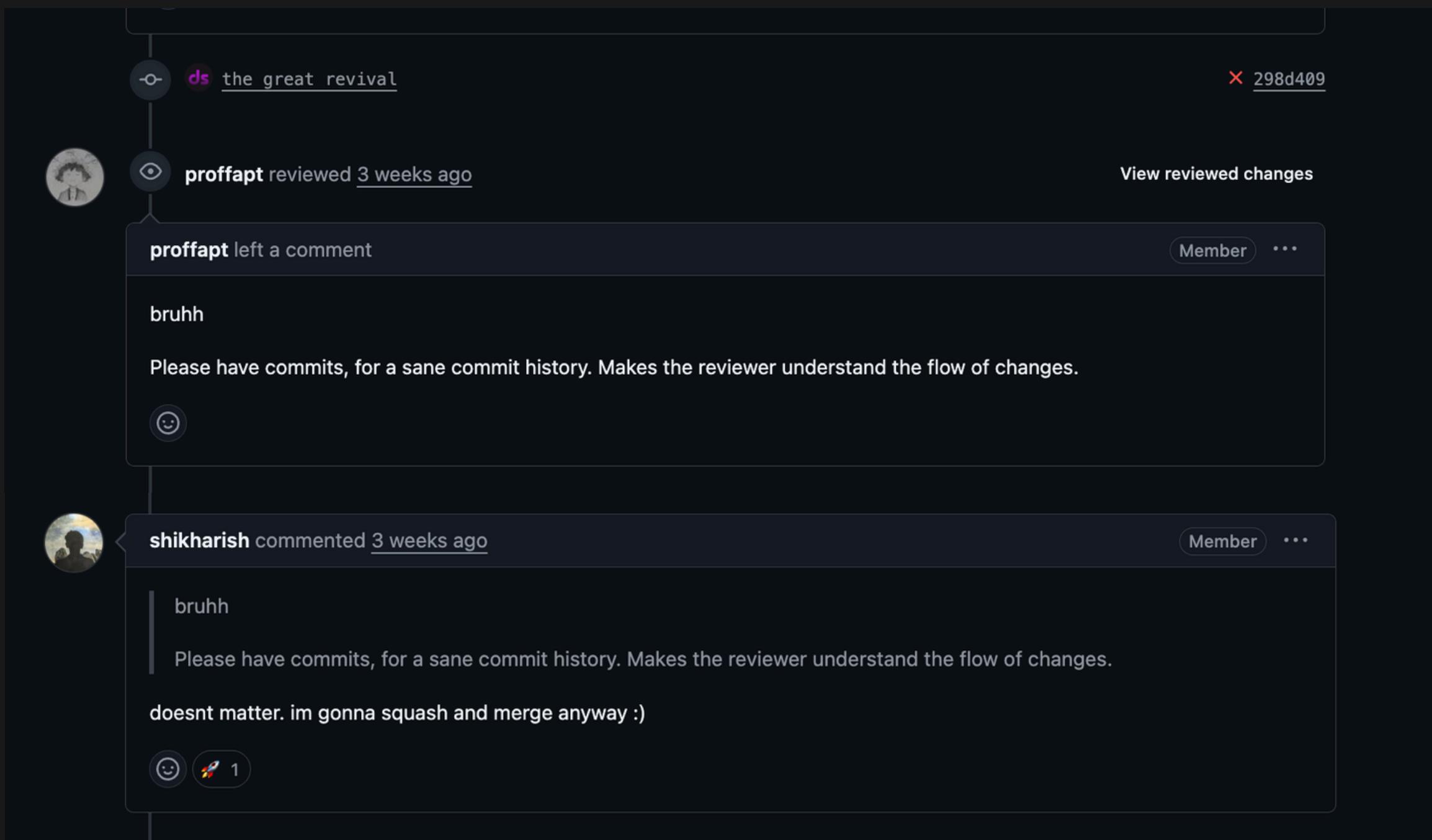
The screenshot shows a GitHub pull request page. The title of the PR is "fix: I am doing the world a favour #1". Below the title, there is a green "Open" button and a message stating "AnshumaanMis... wants to merge 1 commit into shrehanrajsingh:main from AnshumaanMishra:main". The PR has 0 conversations, 1 commit, 0 checks, and 23 files changed. The code review section shows a comment from "AnshumaanMishra" that says "No description provided." There are also options to "Convert to draft" and "Still in progress?". On the right side, there are "Edit" and "Code" buttons.



The screenshot shows the "Comparing changes" interface. It allows users to choose two branches to see what's changed. The base repository is set to "shrehanrajsingh/reactplusplus" and the base branch is "main". The head repository is "AnshumaanMishra/reactplus..." and the compare branch is "main". Below this, there is a "Add a title" field containing "fix: I am doing the world a favour". There is also an "Add a description" field with a rich text editor toolbar at the bottom. A large white arrow points from the "Add a title" field towards the top right of the slide, where the GitHub logo is located.

In the description, describe what your PR does, what issue it fixes, etc.

- Your PR will get discussed in the comments
- Maintainers will review your PR and sometimes request for changes.
- Maintainers will merge your PR once it has approved.



LIST OF COMMANDS

git init	Initialises a git repository in the current directory
git clone <url>	Clones a GitHub repository on your computer
git add .	Adds the changes made to the 'staging area' so they are ready for committing
git commit -m "msg"	Creates a commit with the added changes
git status	Shows the current status of the repo
git push origin <branch>	Pushes the new commits in <branch> to GitHub
git branch	Shows a list of branches and the current branch
git branch <branch>	Creates a new branch called <branch>
git checkout <branch>	Switch to the branch <branch>

DO IT YOURSELF!

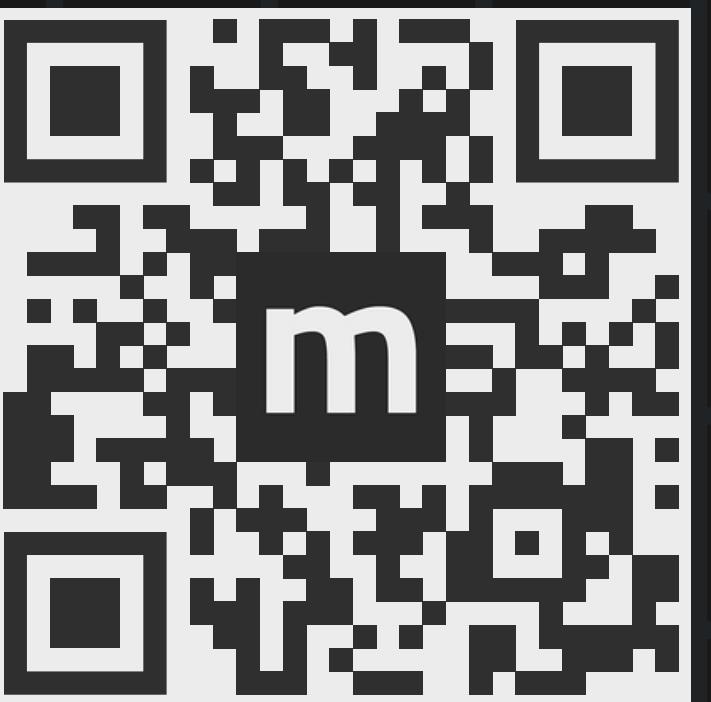
- Search for `kossiitkgp/sandbox` on GitHub
- Fork the repository to your GitHub account
 - Clone your fork to your computer
- Create a new branch named `update-info`, switch to that branch
- Create a file called `<your-github-username>.txt`
 - Write some message in the file
- Add and Commit your changes and push it to GitHub
- Create a Pull Request from your `update-info` branch to the main repo

GIT GOOD :)

- learngitbranching.js.org
- ohmygit.org
- [Git cheatsheet](https://git-cheatsheet.com/)
- [Build your own Git](https://buildyourowngit.com/)

MAKE YOUR FIRST CONTRIBUTION

- [Check out metaKGP repositories](#)
- Start with issues labelled as *Good first issue*
- goodfirstissue.dev
- Participate in KWoC 2025 (Follow KOSS on [socials](#))



WE NEED YOUR FEEDBACK!



tinyurl.com/git-25-feedback

