Git/GitHub Workflows

a.k.a. "Git with Joyce"

RLadies meetup, January 9, 2019

Part 2

Joyce Robbins
Dept. of Statistics
Columbia University

jtr13@columbia.edu
Twitter: @jtrnyc
GitHub: @jtr13

Git/GitHub Workflows

1. GitHub only
2. GitHub + local master branch
3. **GitHub + local master plus additional branches on your repo**
4. Contribute to someone else's repo

3. Create a new branch

Why?

- By working on a branch, we can allow collaborators to review our code before merging to master.
- The keyword is collaborators, but branching is useful regardless.

3. Remote + local master + other branches

From the perspective of the project:

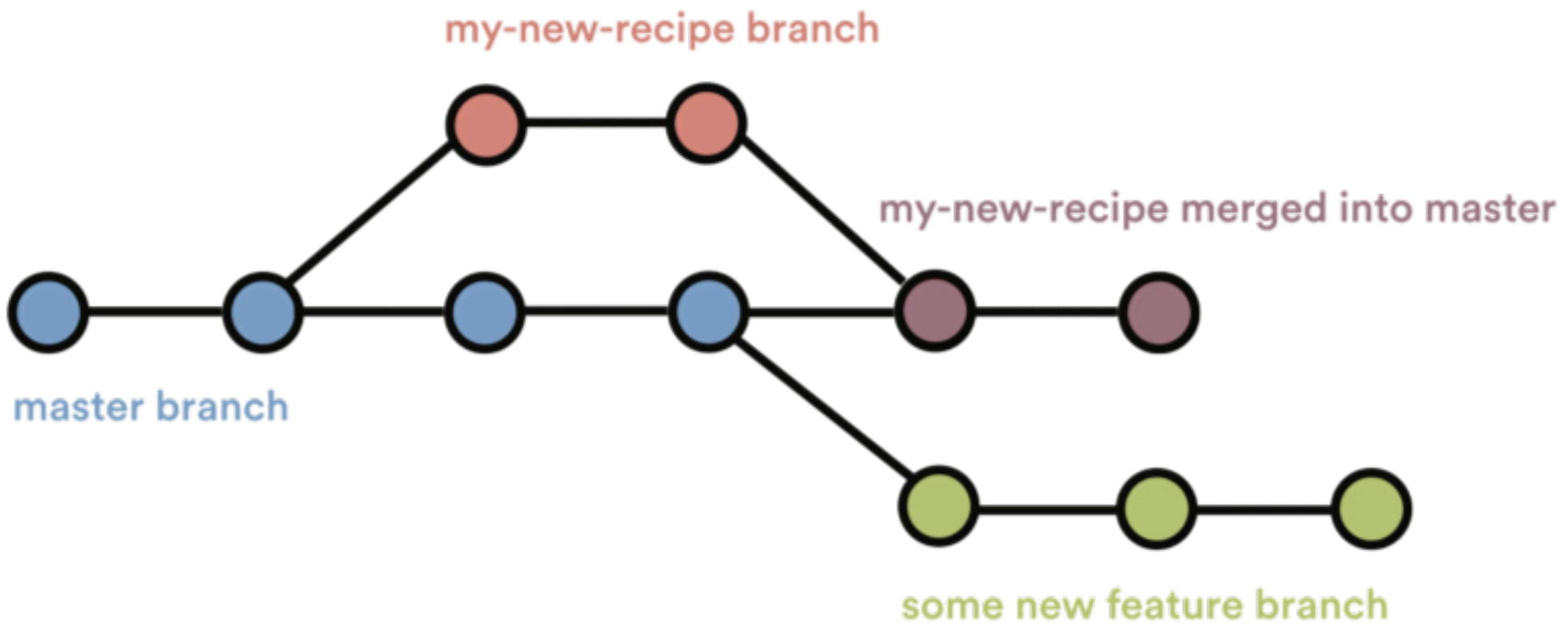


Image source: https://medium.com/designing-atlassian/learn-to-love-git-part-three-collaboration-a778157d3a1c

Your perspective

Start by creating a repo on GitHub

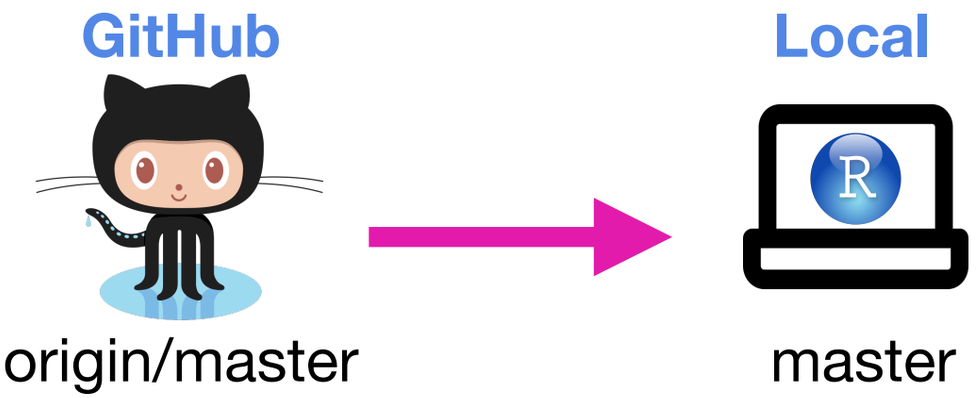
origin/master

Origin/master

Working, shared code lives here

origin/master

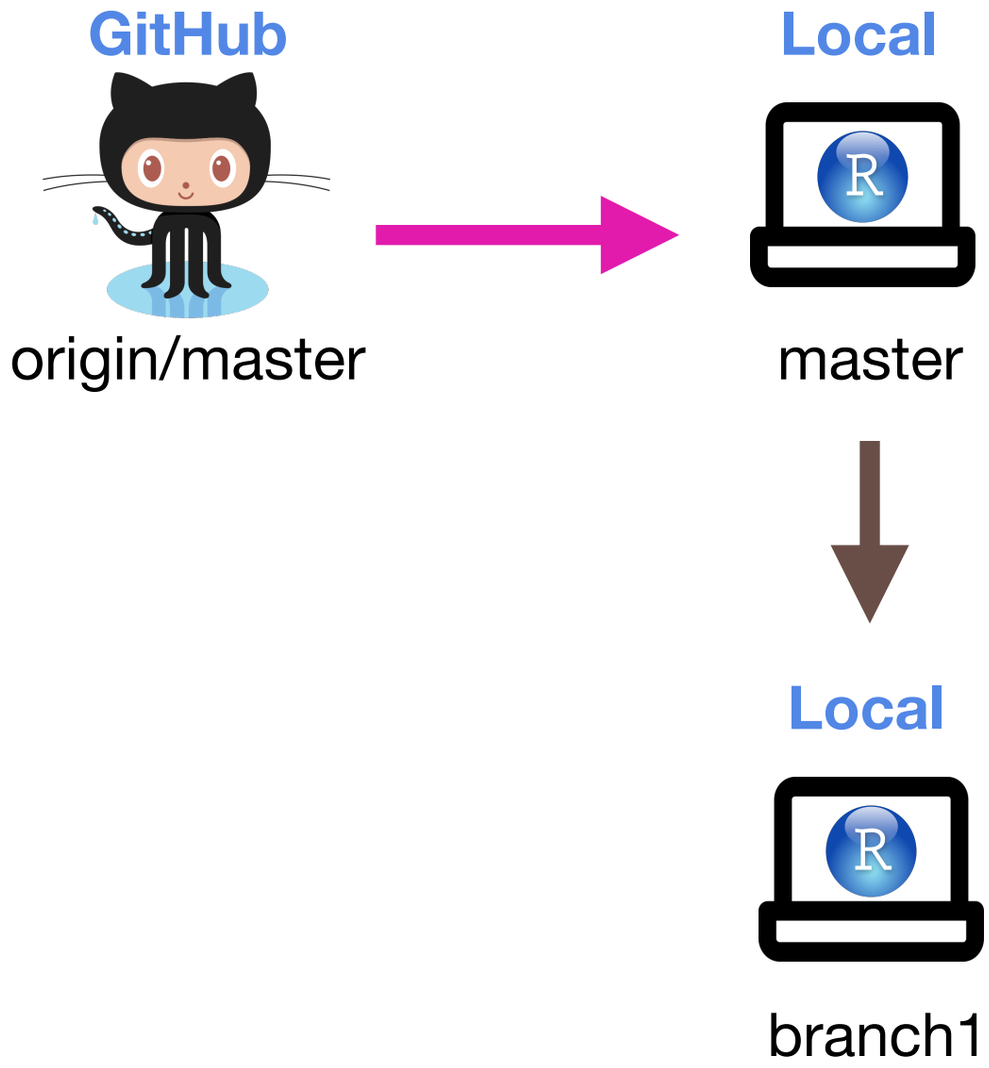
Clone it once



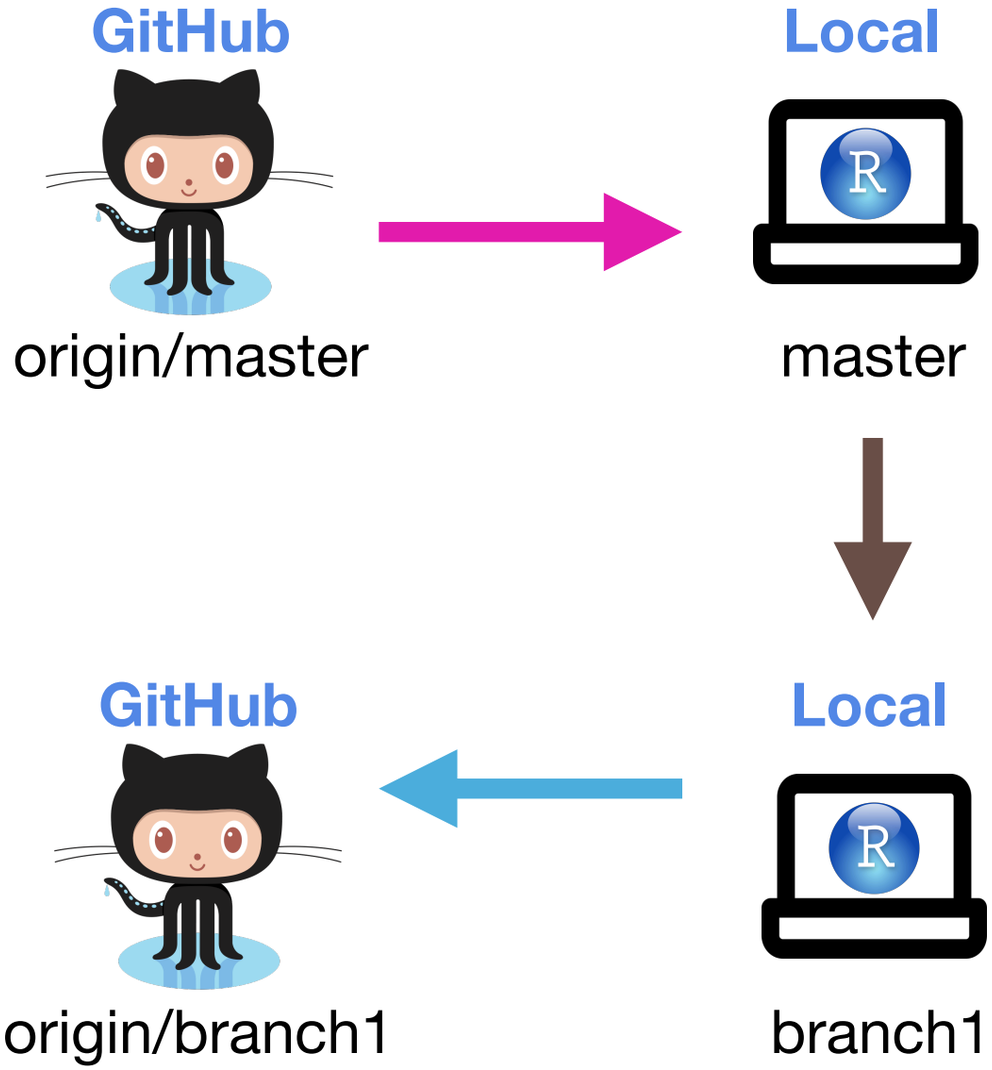
Make a local copy

(Next time, this step will be a PULL)

Create a branch



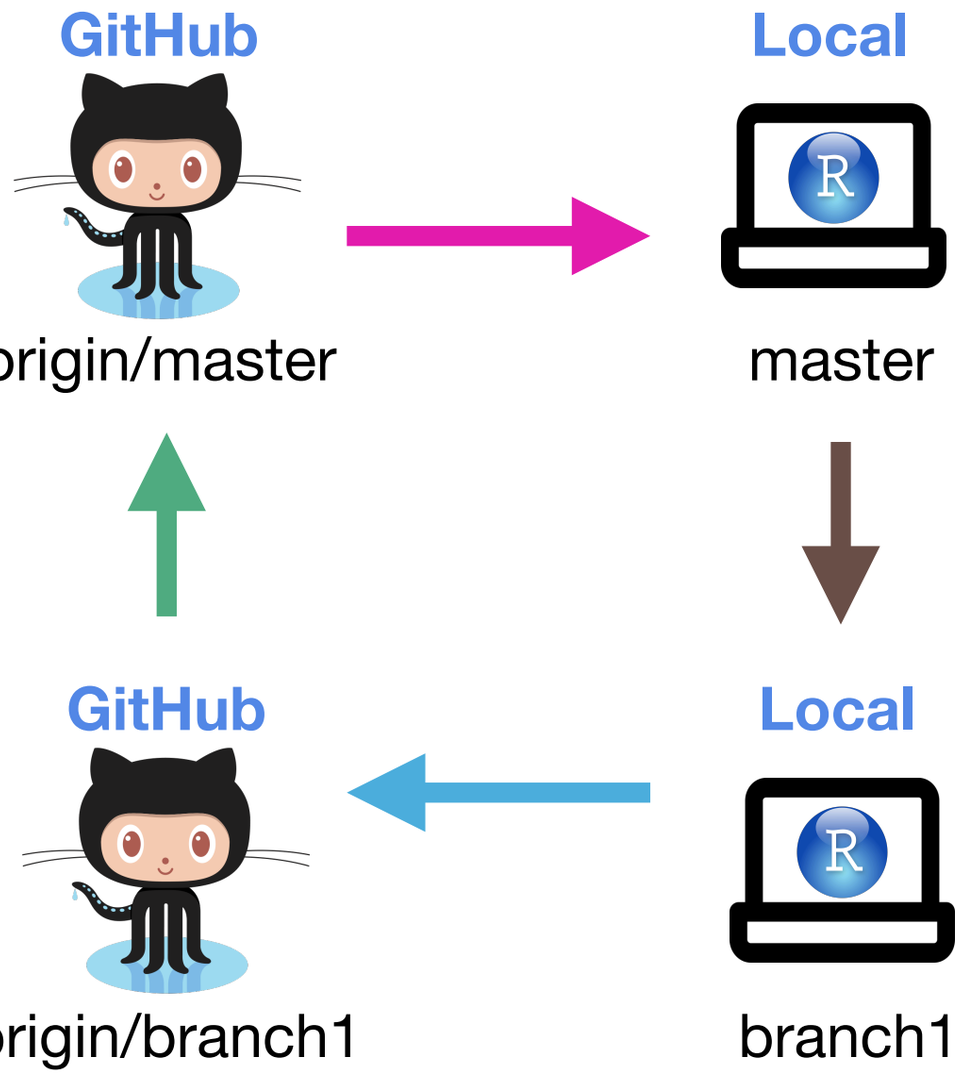
Save / commit / push changes



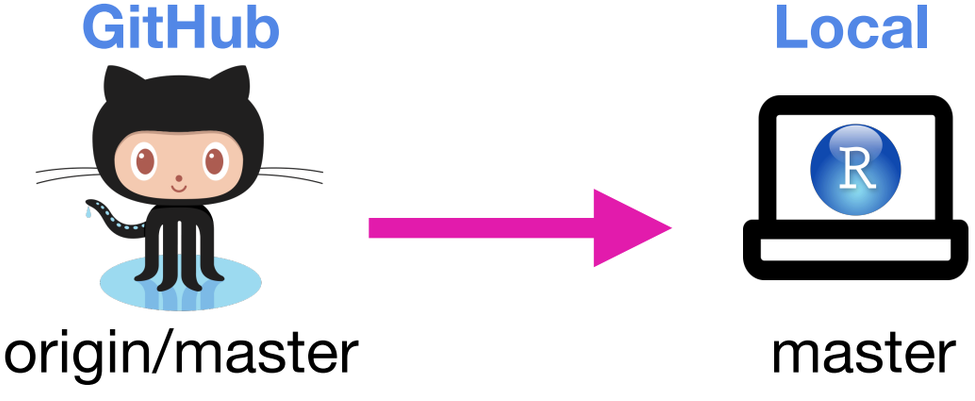
Push the new branch to GitHub to be reviewed

Pull request and merge

Submit a pull request, then someone else merges your changes into master

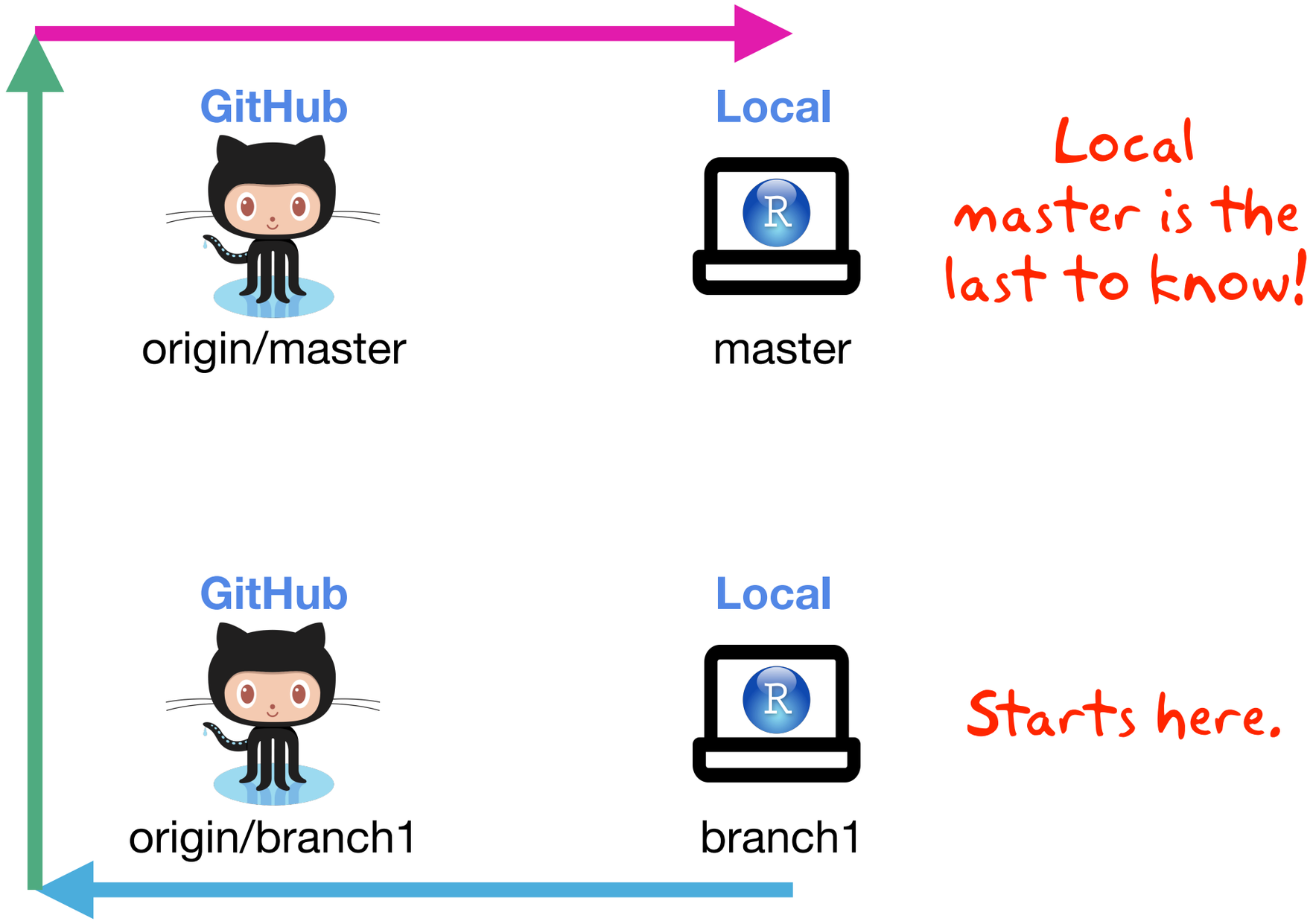


Your perspective

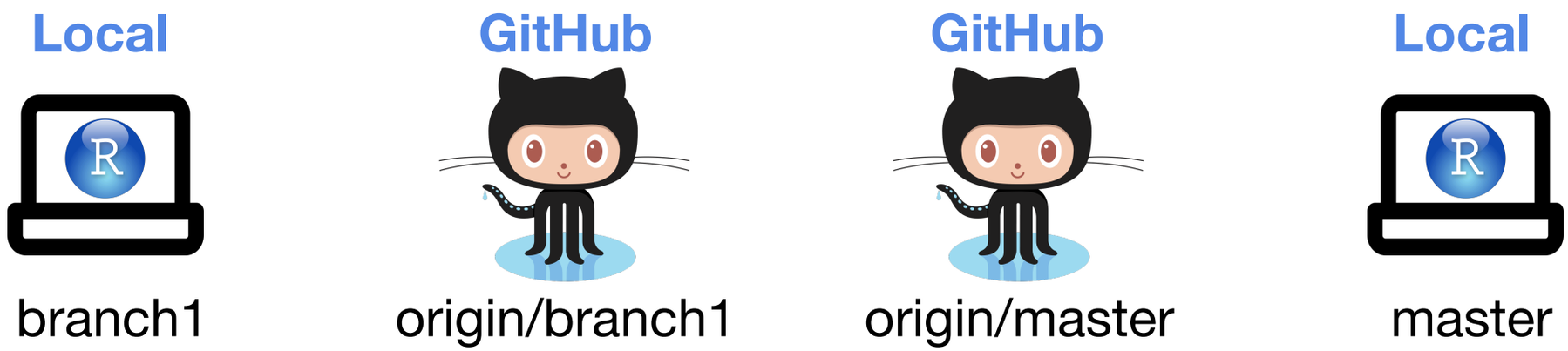


Your branch is deleted and the new stuff is pulled into your copy of the master branch.

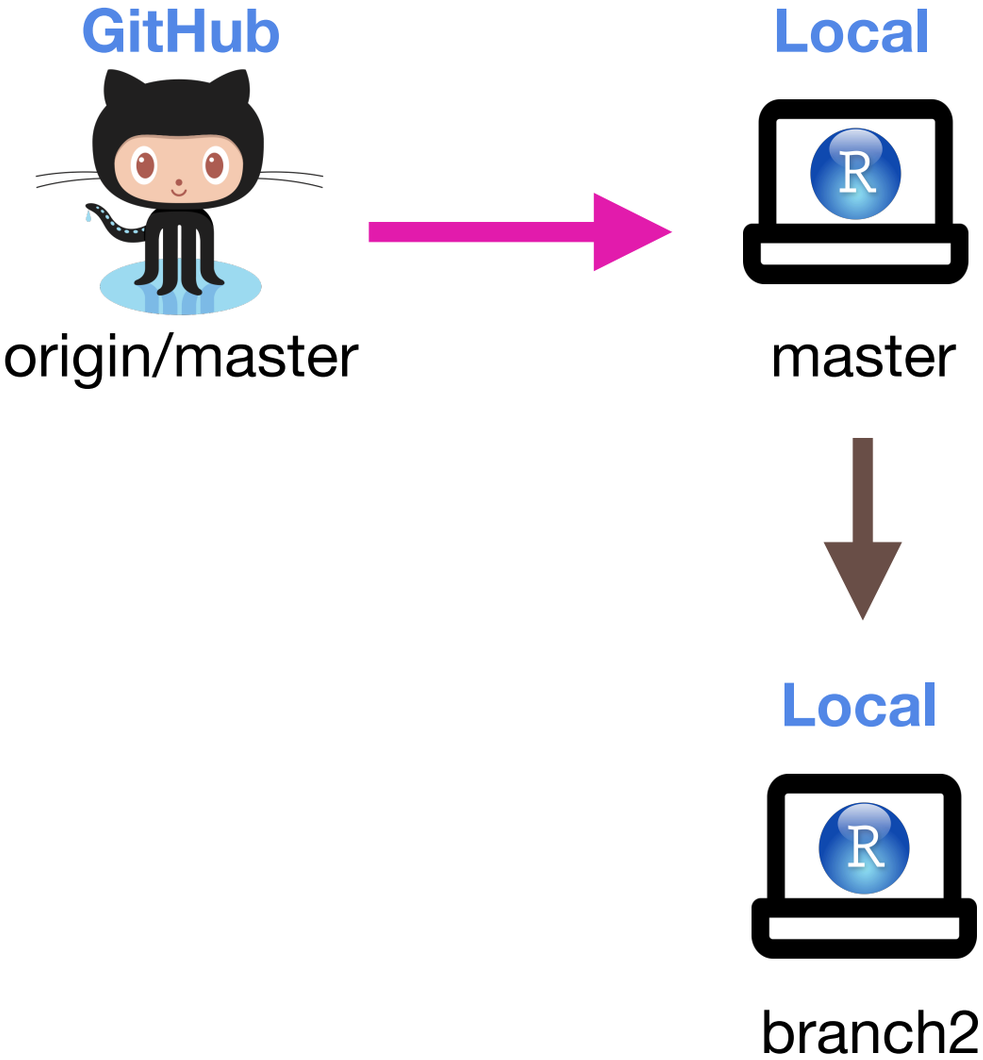
Note the flow of new code



Note the flow of new code

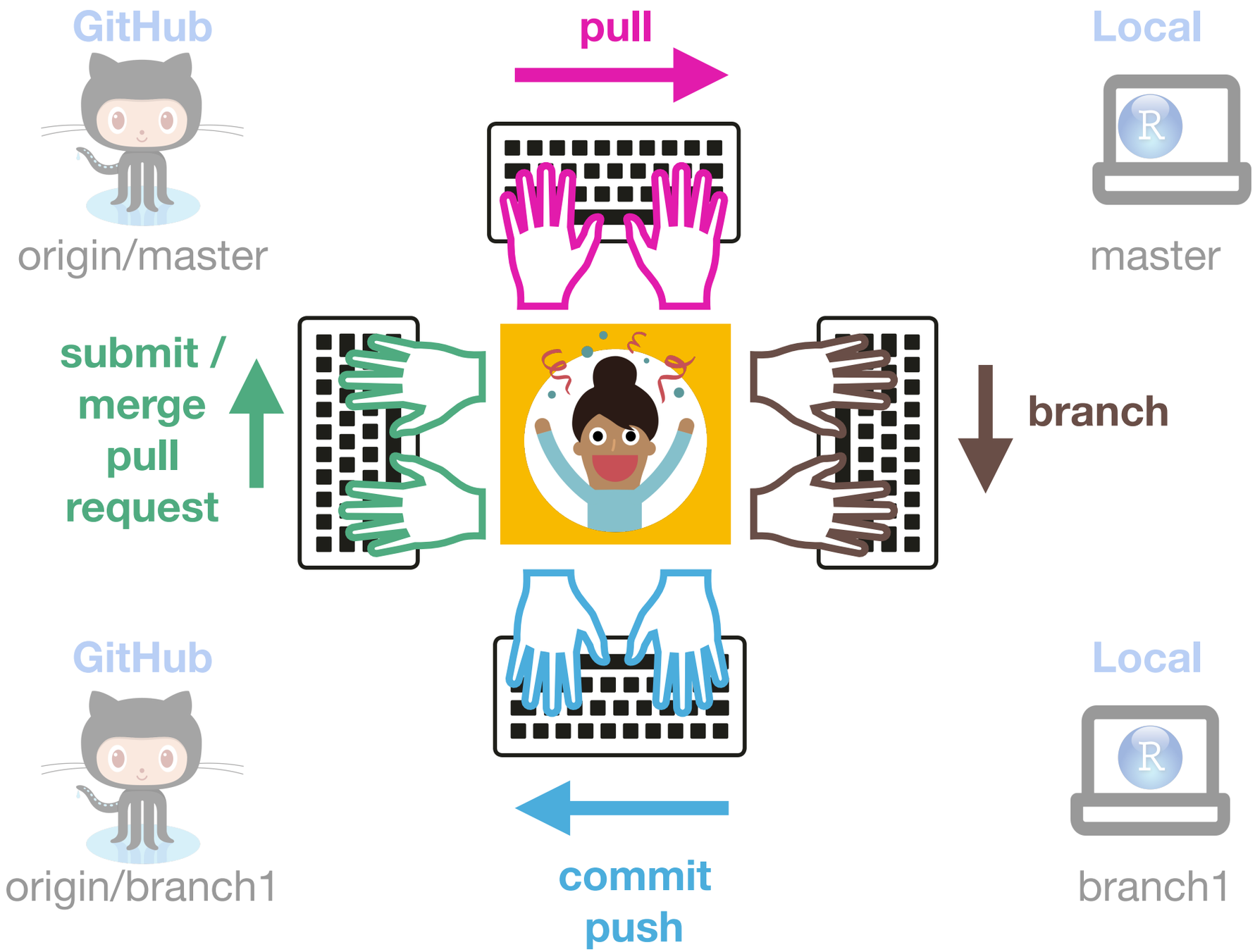


The second change...

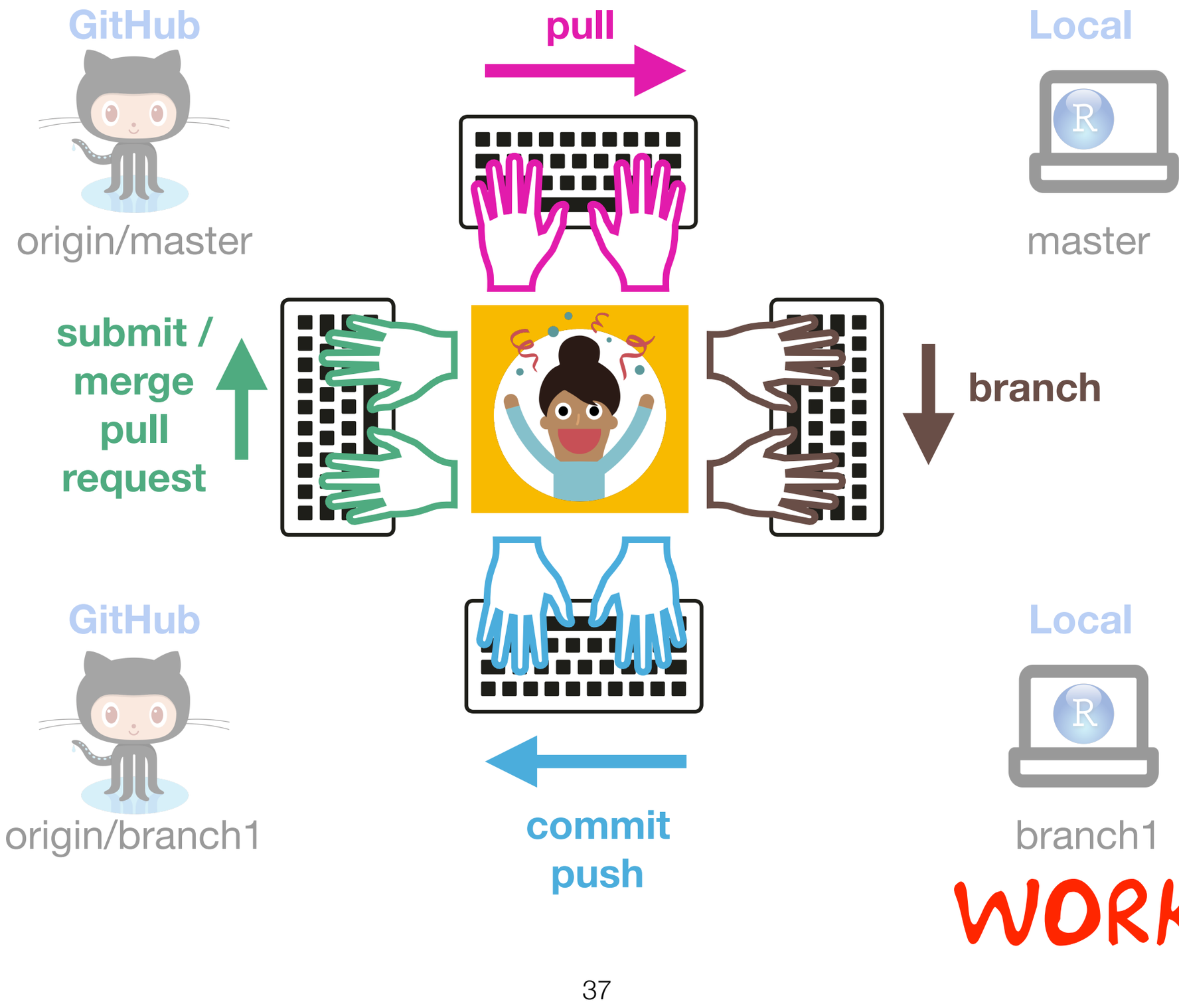


And so on and so on...

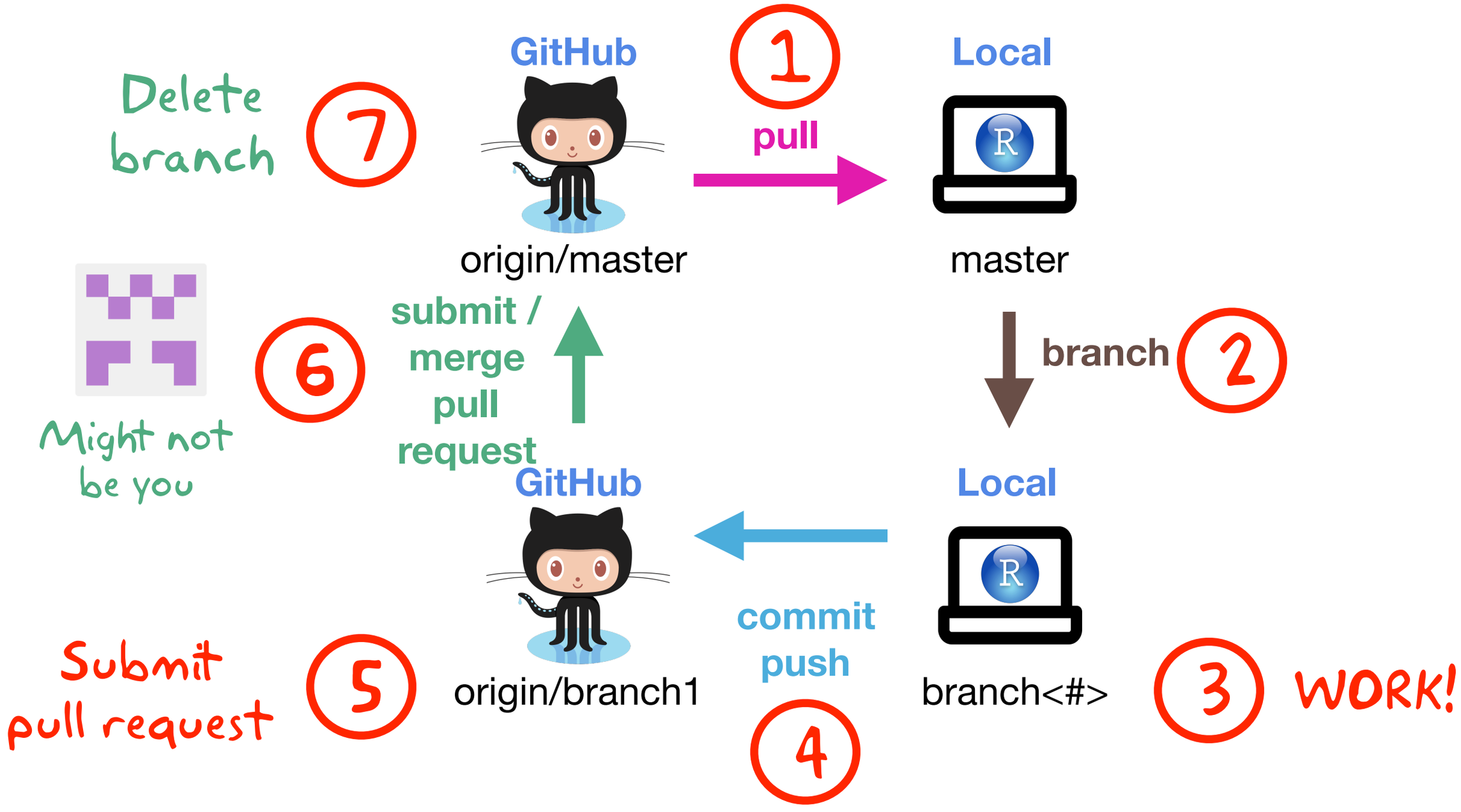
Your perspective



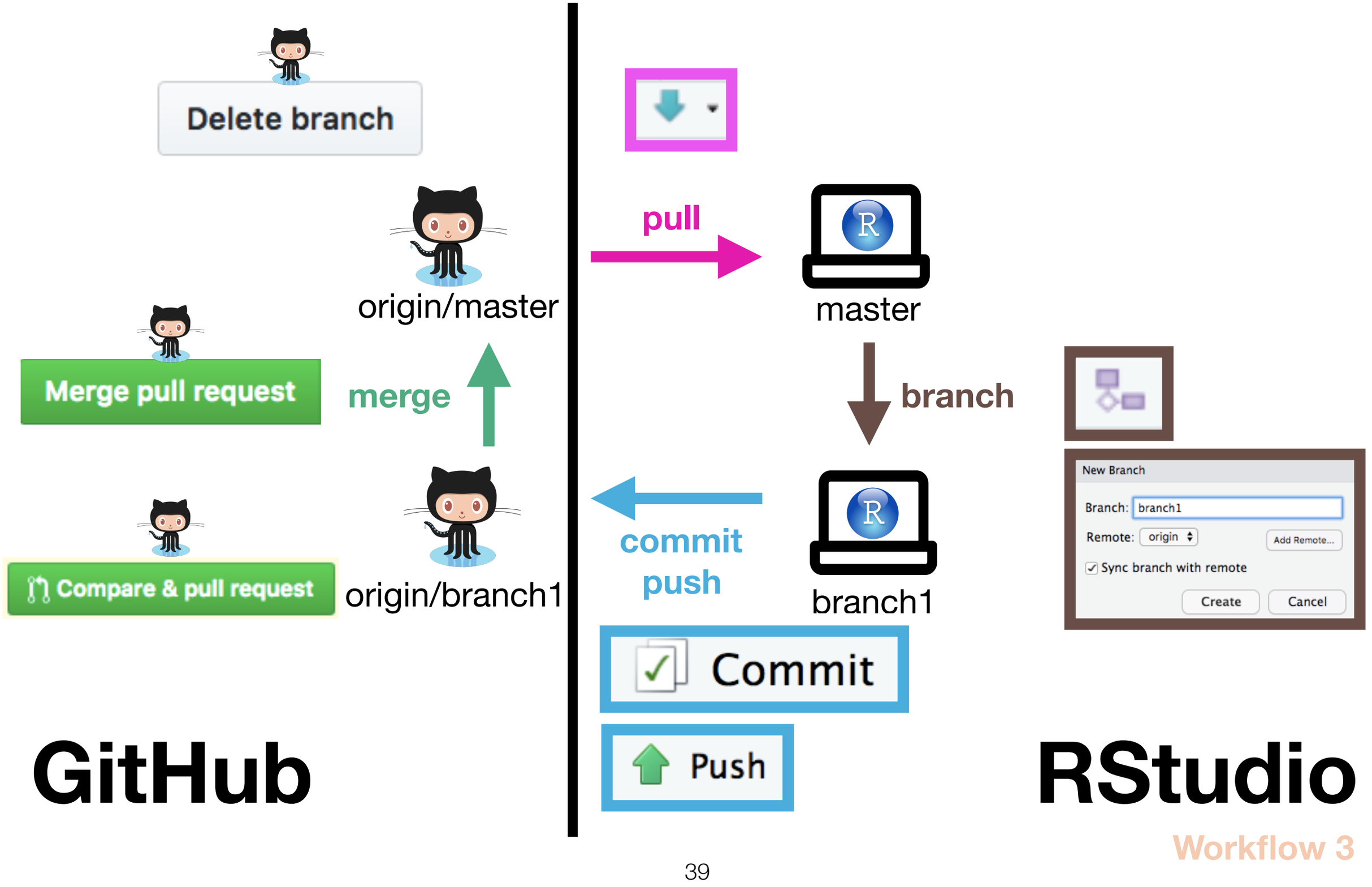
Your perspective



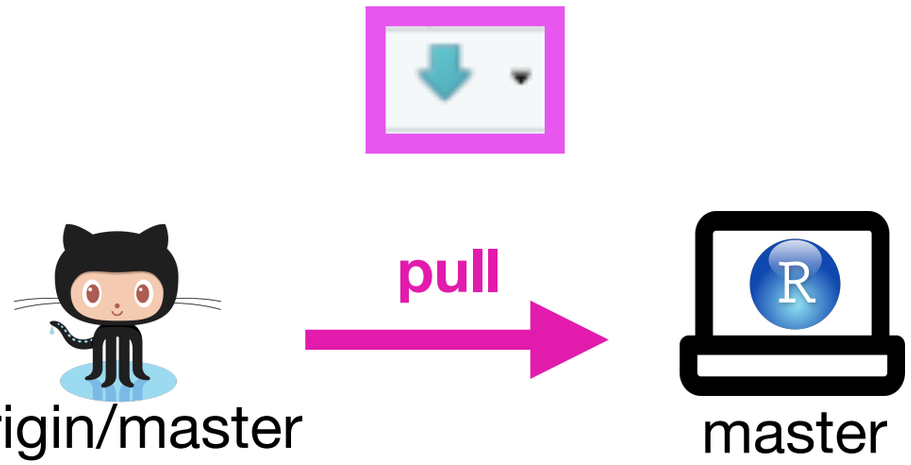
Your workflow



What's happening where

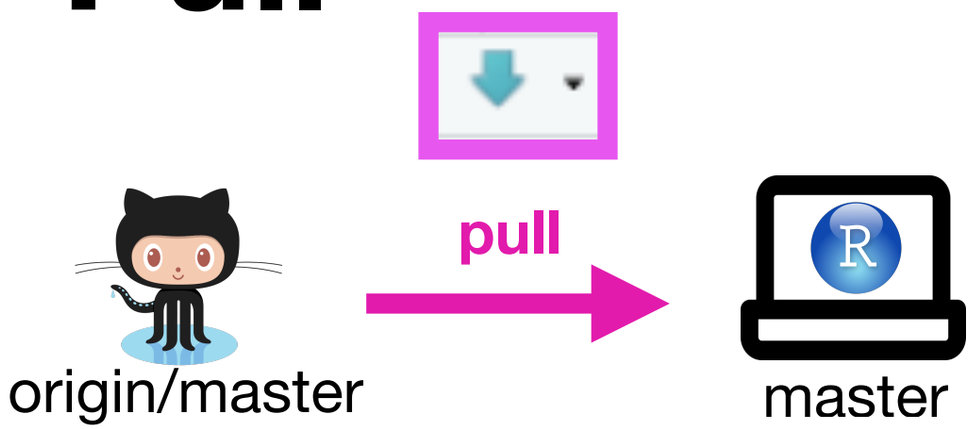


Step 1. Pull



- Every work session should begin with a pull to make sure that we're up-to-date with master (as in the previous workflow).

Step 1. Pull

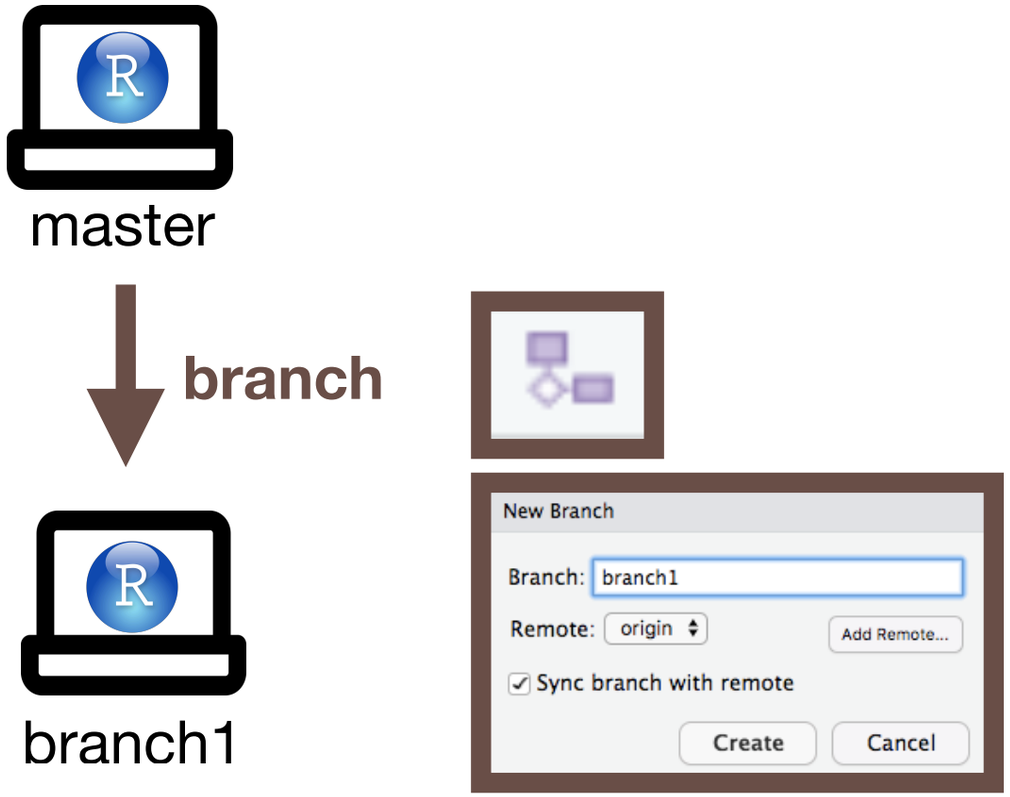


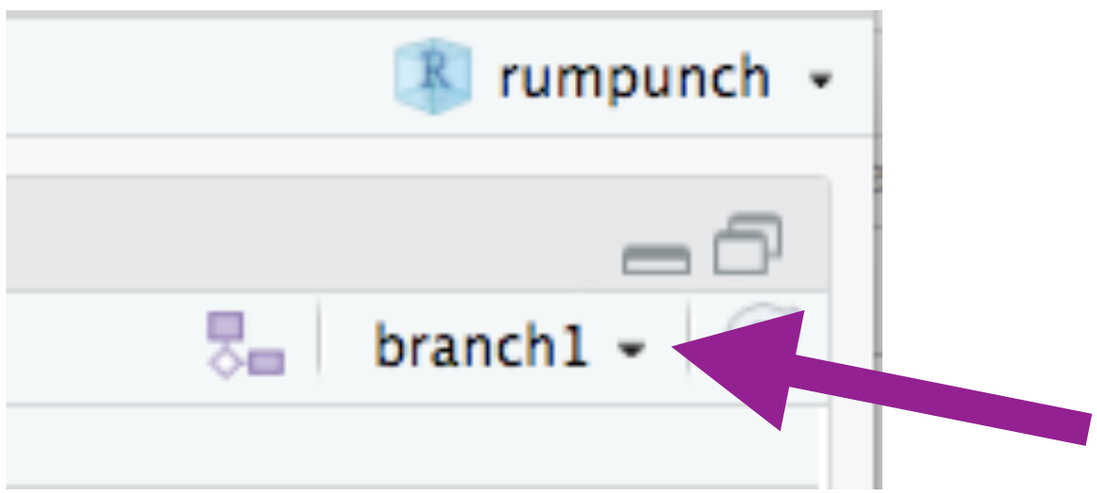
- If all goes well (no conflicts), our copy of master will be updated:

```
>>> git pull
From https://github.com/jtr13/rumpunch
   788e3b0..465857b  master     -> origin/master
Updating 788e3b0..465857b
Fast-forward
 Thanksgiving.R | 3 +++
 1 file changed, 3 insertions(+)
```

Step 2: Create a new branch

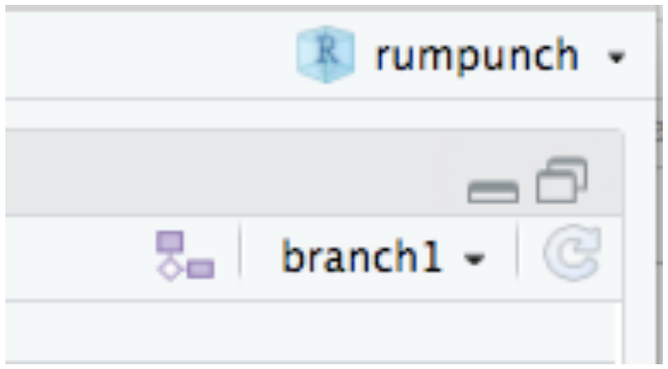
- We'll do our work on this branch.
- Check the top right corner to be sure you're in the right place:





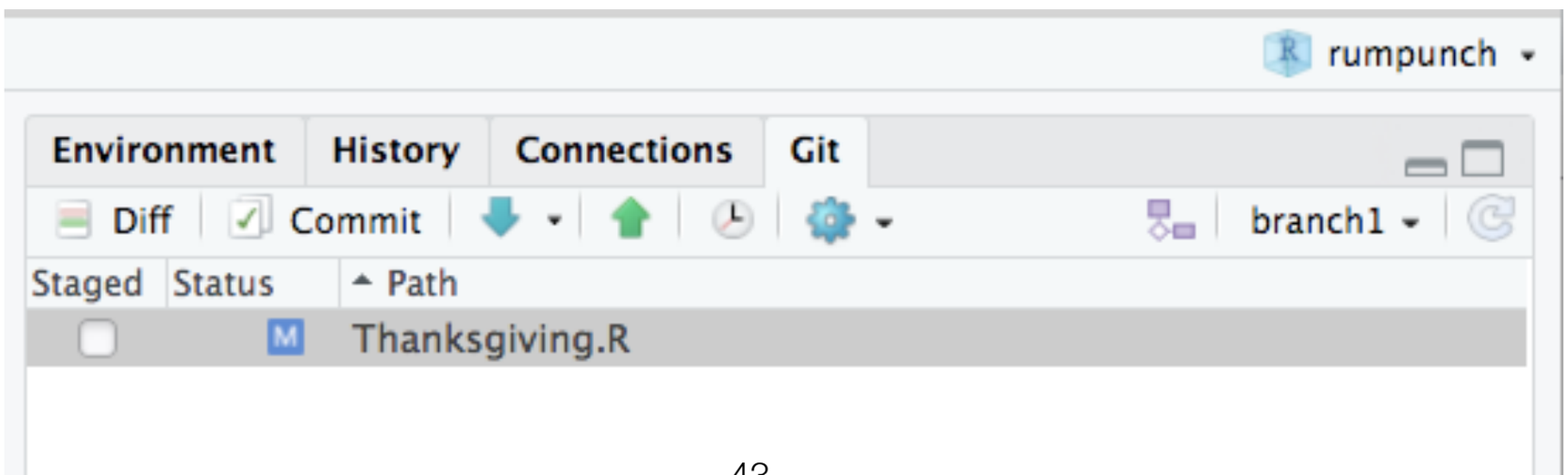
Step 3: Work

```
28
29  ## @param .op Can be a function or a quoted name of a function. If a
30  ##   quoted name, the default environment is the [base
31  ##   environment][rlang::base_env] unless you supply a
32  ##   [quosure][rlang::quo].
33  quo_reduce <- function(..., .op) {
34    stopifnot(is_symbol(.op) || is_function(.op))
35
36    dots <- quos(...)
37    if (length(dots) == 0) {
38      abort("At least one expression must be given")
39    } else if (length(dots) == 1) {
40      return(dots[[1]])
41    }
42
43    op_quo <- as_quosure(.op, base_env())
44    op <- quo_get_expr(op_quo)
45
46    expr <- reduce(dots, function(x, y) expr((!!op)((!!x), (!!y))))
47    new_quosure(expr, quo_get_env(op_quo))
48  }
```

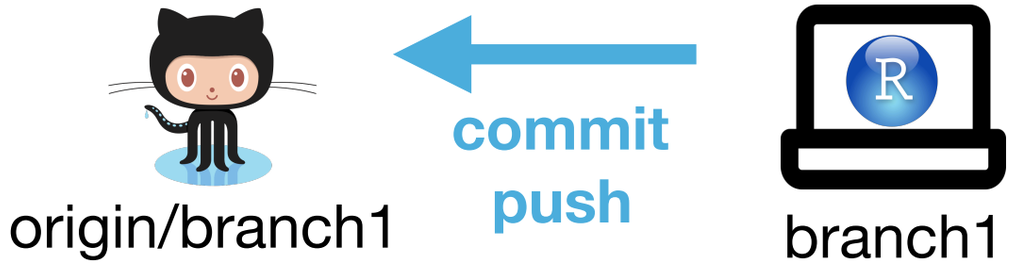


branch1

- Observe changing files in the Git pane:



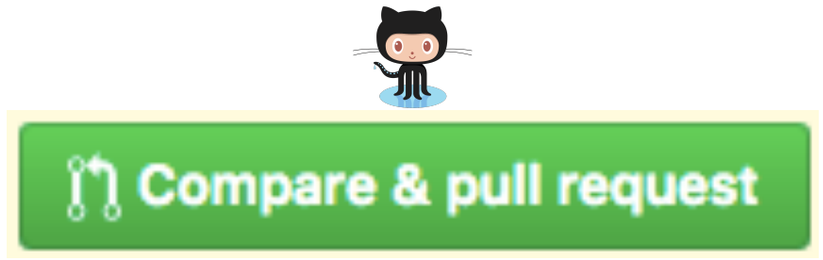
Step 4: Commit and push



- Commit and push files as before.
- If all goes well:

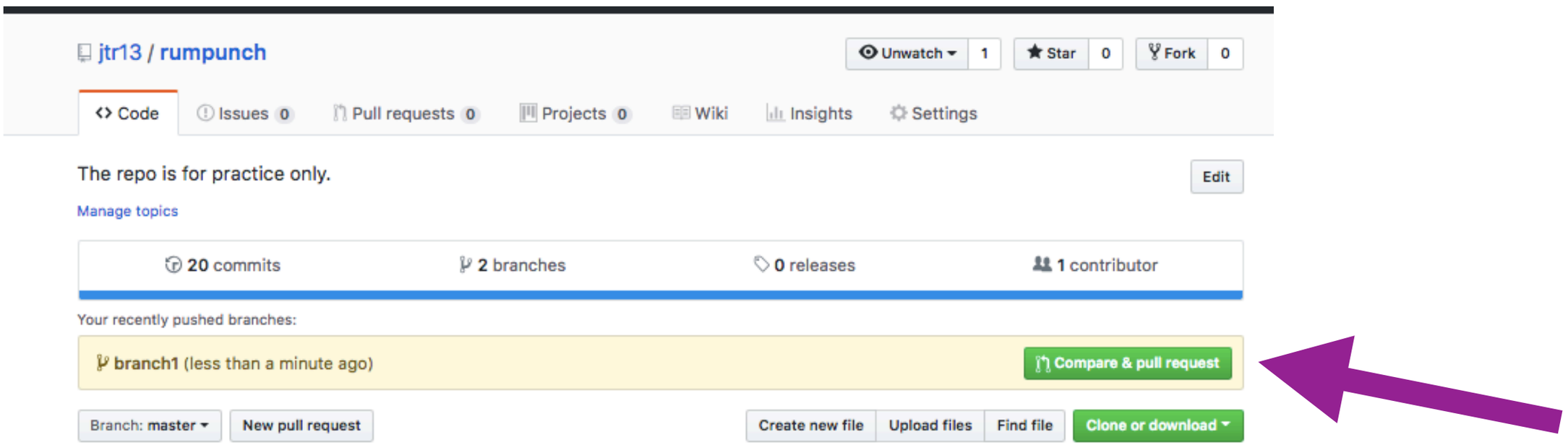
```
>>> git push origin refs/heads/branch1
To https://github.com/jtr13/rumpunch.git
   7424222..6cf5975  branch1 -> branch1
```

Step 5: Submit a pull request



origin/branch1

- GitHub detects a difference between the master branch and branch1:

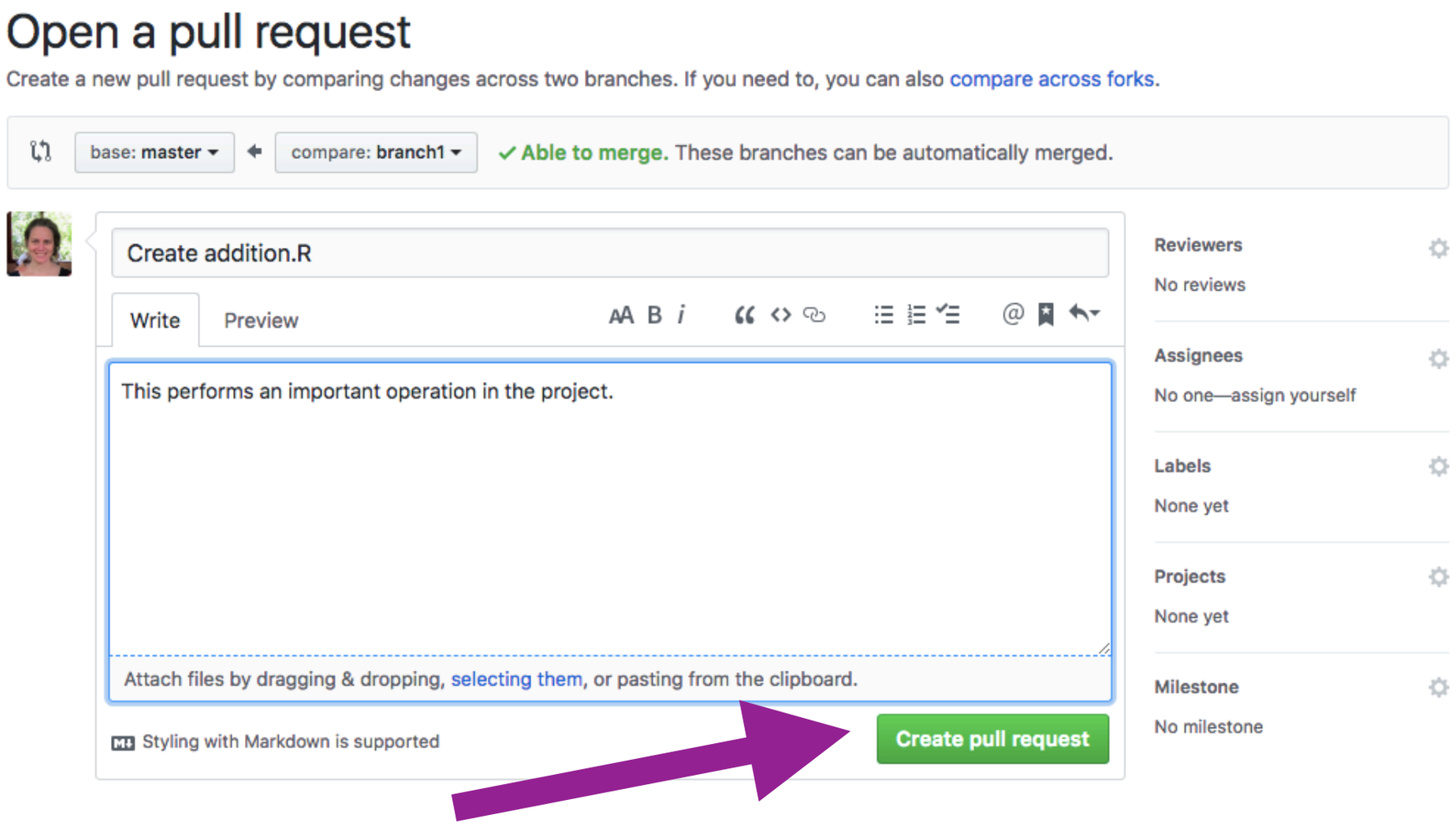


Workflow 3

Step 5: Submit a pull request

- Click: 

- Add a description



- Then click "Create pull request"

Step 6: Merging a pull request

- There are a lot of opinions on who should merge the pull request: the original author (you) or someone else
- What's most important is that you communicate with your collaborators and decide how you're going to manage the pull requests.
- Practice both merging your own pull requests and letting someone else do it.

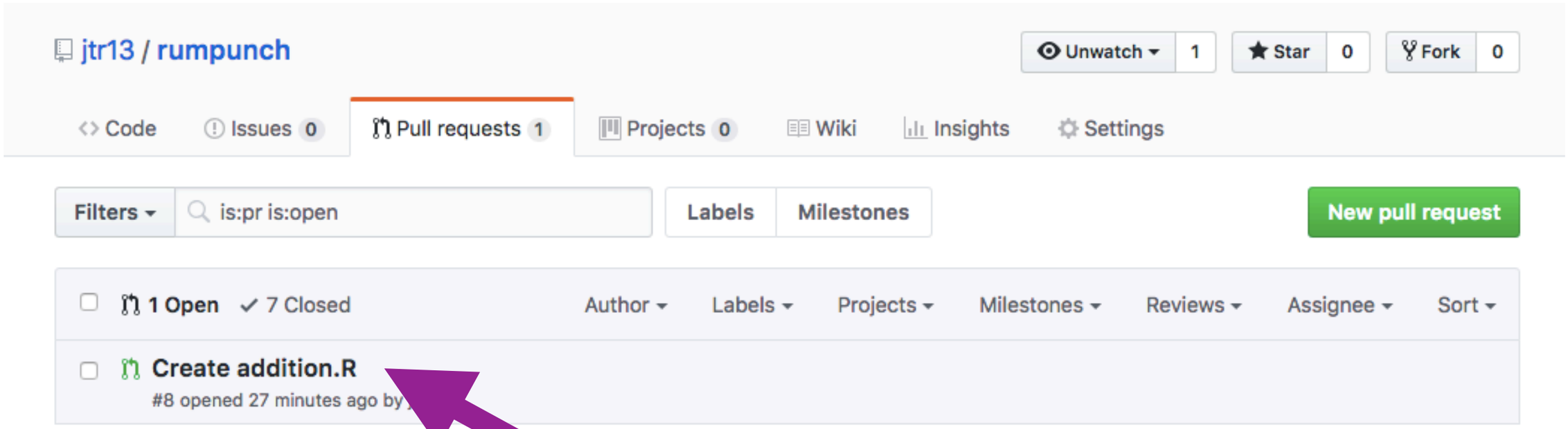
Step 6: Merging a pull request

- Pull requests can either be merged on GitHub, or locally.
- Here we only cover merging pull requests on GitHub.
- To learn how to do it locally, see:

"Explore and extend a pull request", *Happy Git with R* (ch. 25)

Step 6: Merging a pull request

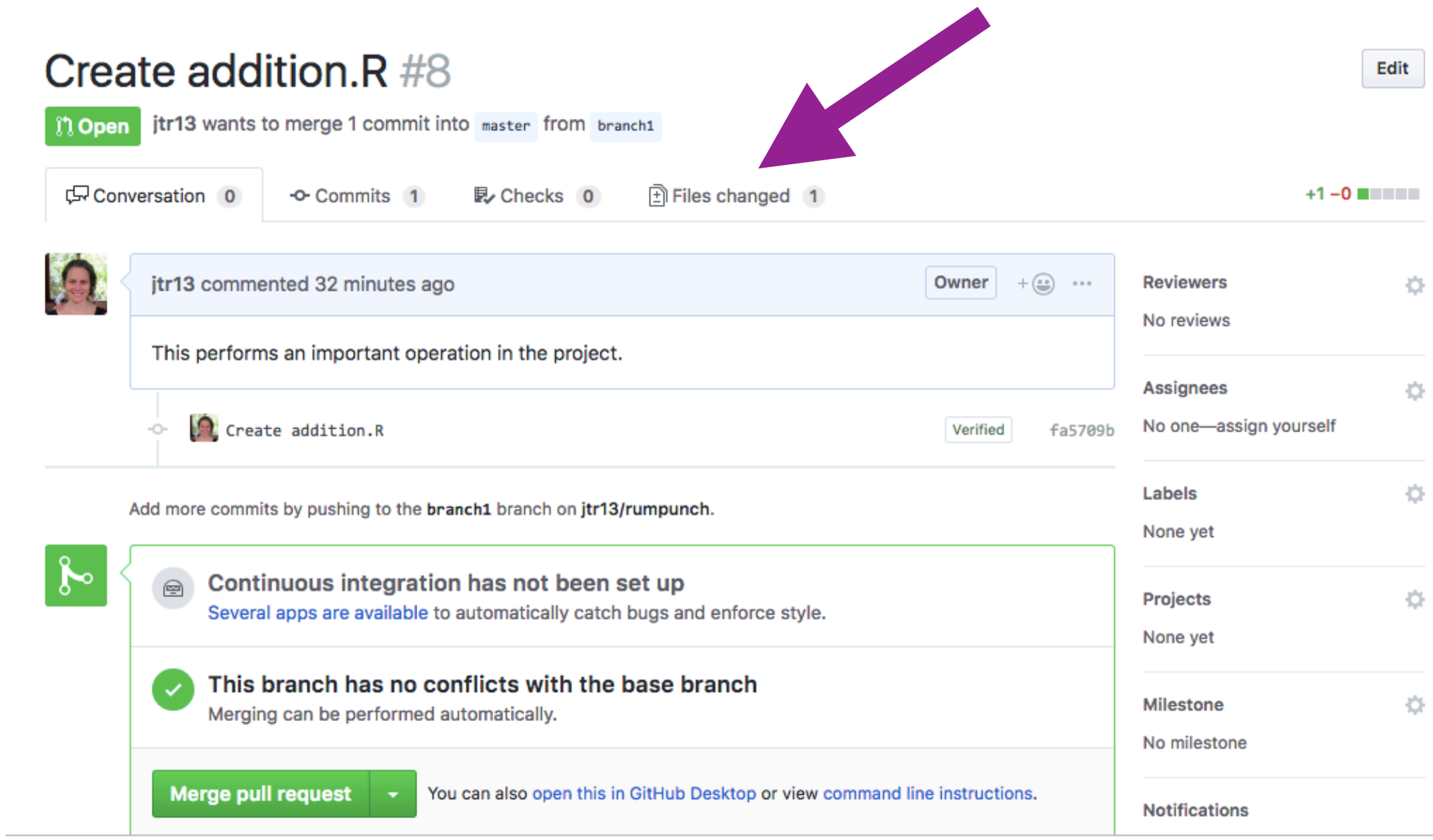
- If you're the one merging the pull request, click the "Pull Requests" tab and you'll see something like this:



- Click the title of the pull request

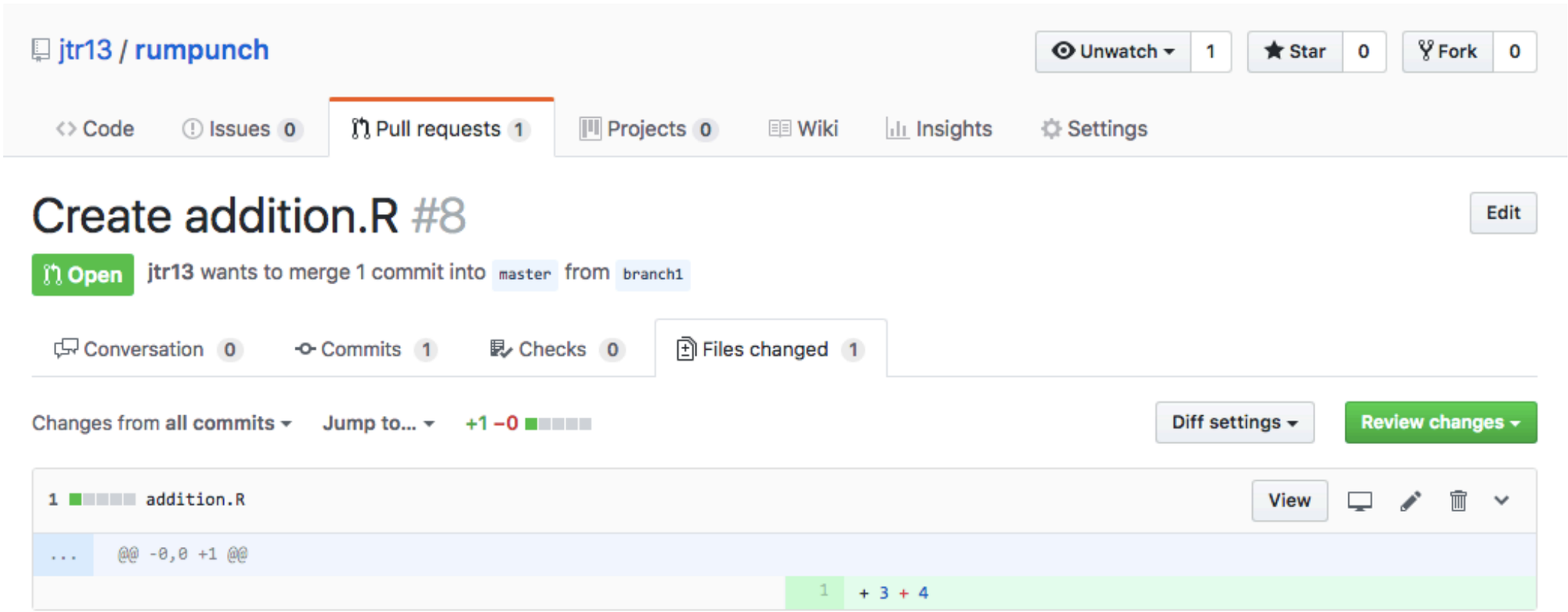
Step 6: Merging a pull request

- Click "Files changed"



Step 6: Merging a pull request

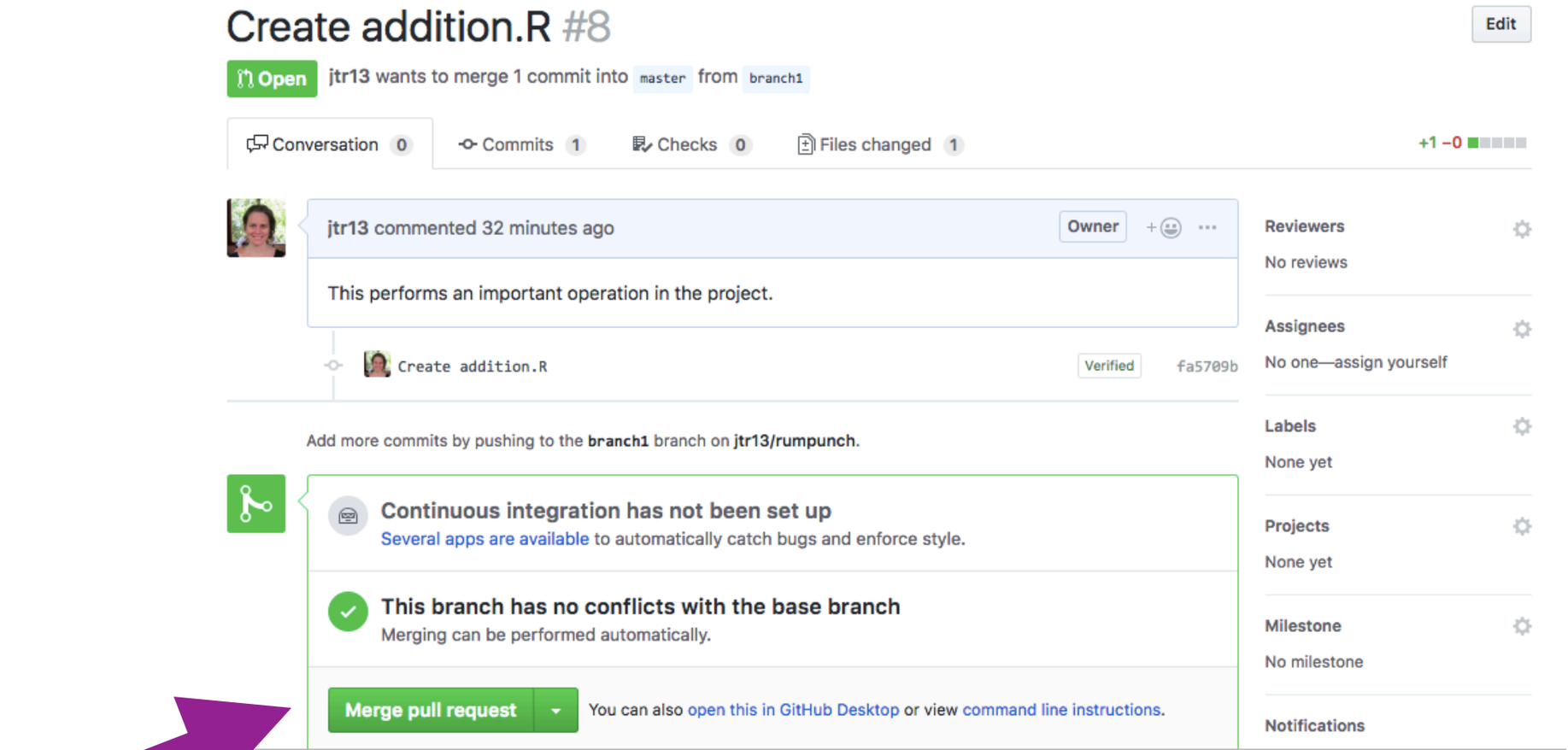
- Review the changes



- Leave comments to the author to make edits (if applicable)

Step 6: Merging a pull request

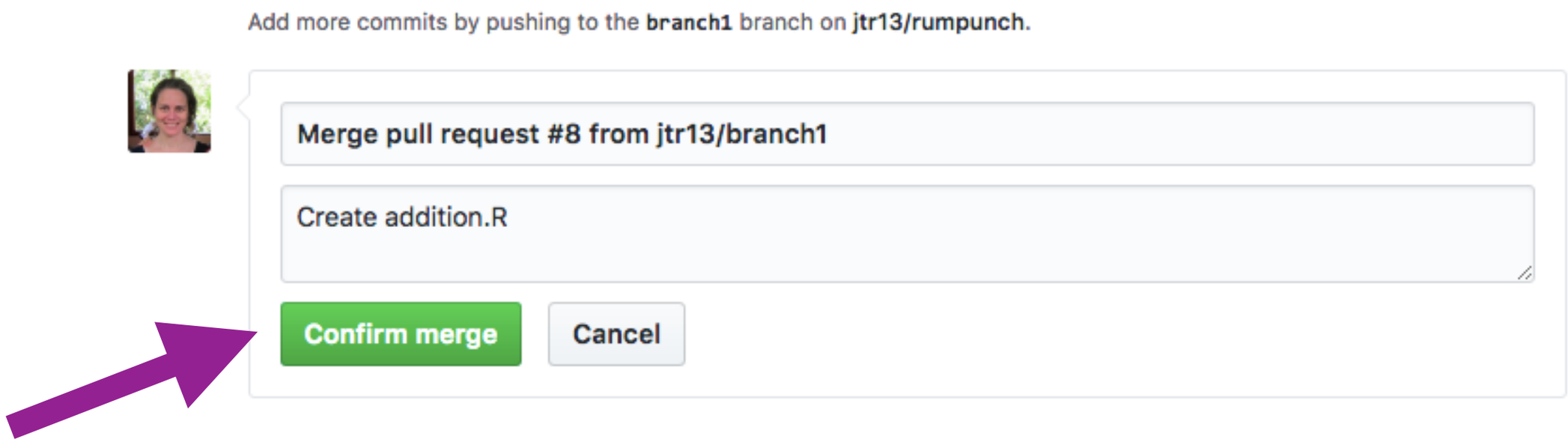
- Click back to return the pull request



- If you're satisfied with the code, click "Merge pull request"

Step 6: Merging a pull request

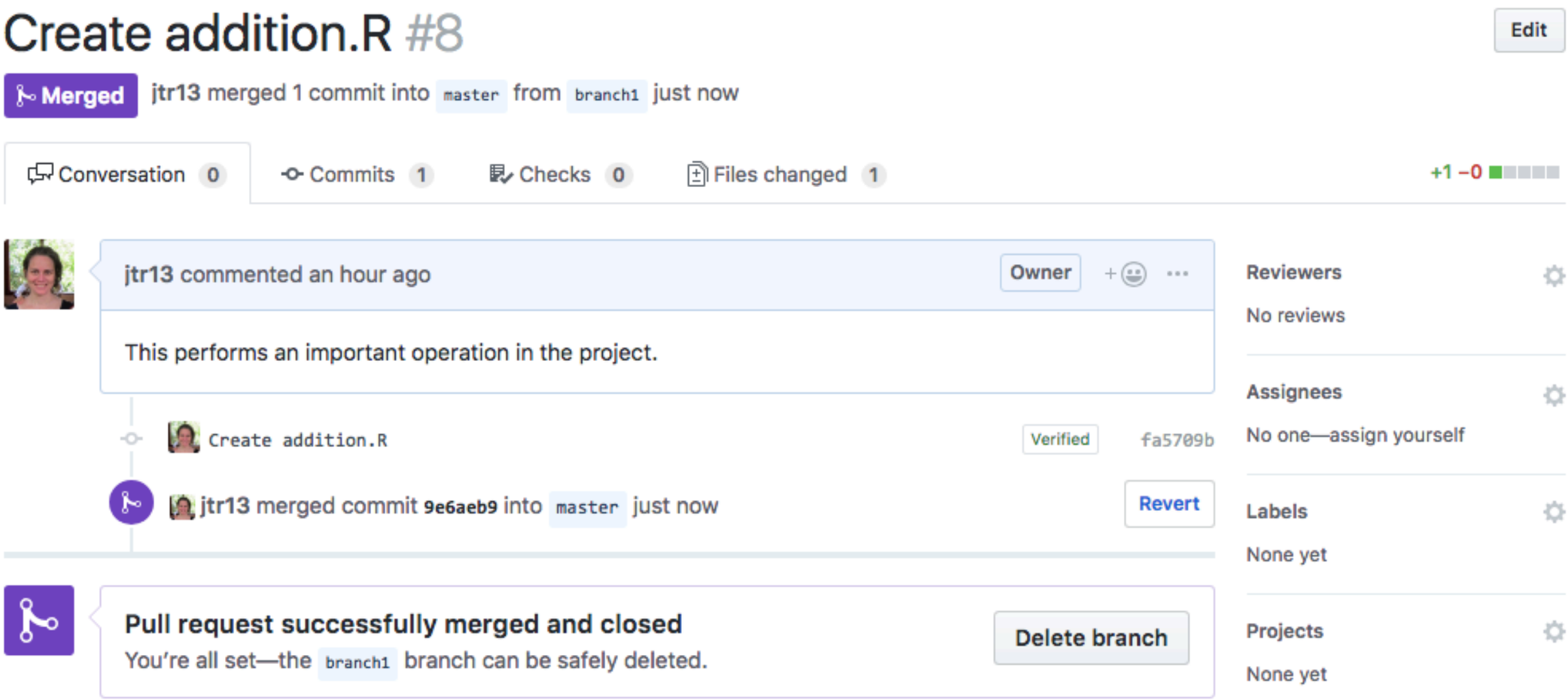
- Almost done...



- And if you really meant it, click "Confirm merge"

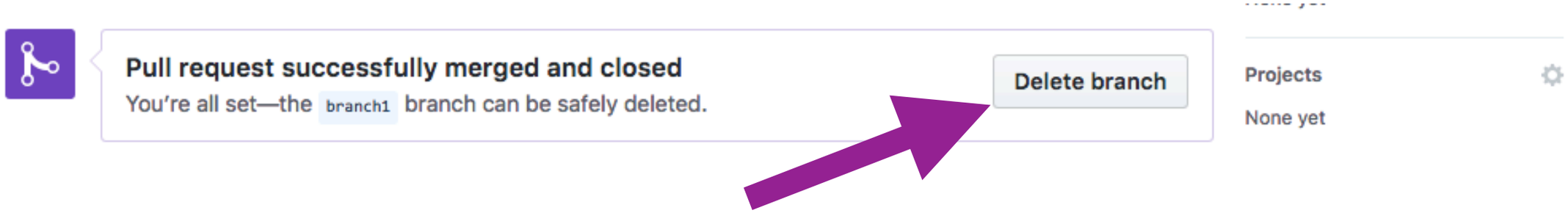
Step 6: Merging a pull request

- Success!



Step 6: Delete the branch

- It's a good idea to delete merged branches. When the merge is complete, you're given the option to delete the branch on GitHub:



Step 7: Delete the branch locally

```
> git branch -d <branchname>
```

Stop tracking remote branch

```
> git fetch -p
```

Git/GitHub Workflows

1. GitHub only
2. GitHub + local master branch
3. GitHub + local master plus additional branches on your repo
4. **Contribute to someone else's repo**

Terminology

Think in terms of *repositories* and *branches*

Types of Repositories (from your perspective)

local repository -- resides on *your* computer

remote repository -- resides somewhere else

> **origin** -- the repo that you created or forked on GitHub
>
> **upstream** -- the original repo of the project that you forked (if you didn't create it)

Note: these are simplified definitions that focus on the way these terms are most commonly used

Contribute to someone else's repo

1. Begin by *forking* another repo on GitHub rather than creating your own.

2. Main challenge: keeping your code up-to-date with upstream

The Workflow

1. **new**: fork repo (once)

2. **new**: configure a remote that points to the upstream repository (once)

https://help.github.com/articles/configuring-a-remote-for-a-fork/

```
> git remote add upstream https://...
```

3. clone repo (once)

4. repeat: pull, branch, work, commit/ push, submit pull request, wait for PR to be merged... delete branch locally

new: sync local master with upstream master:

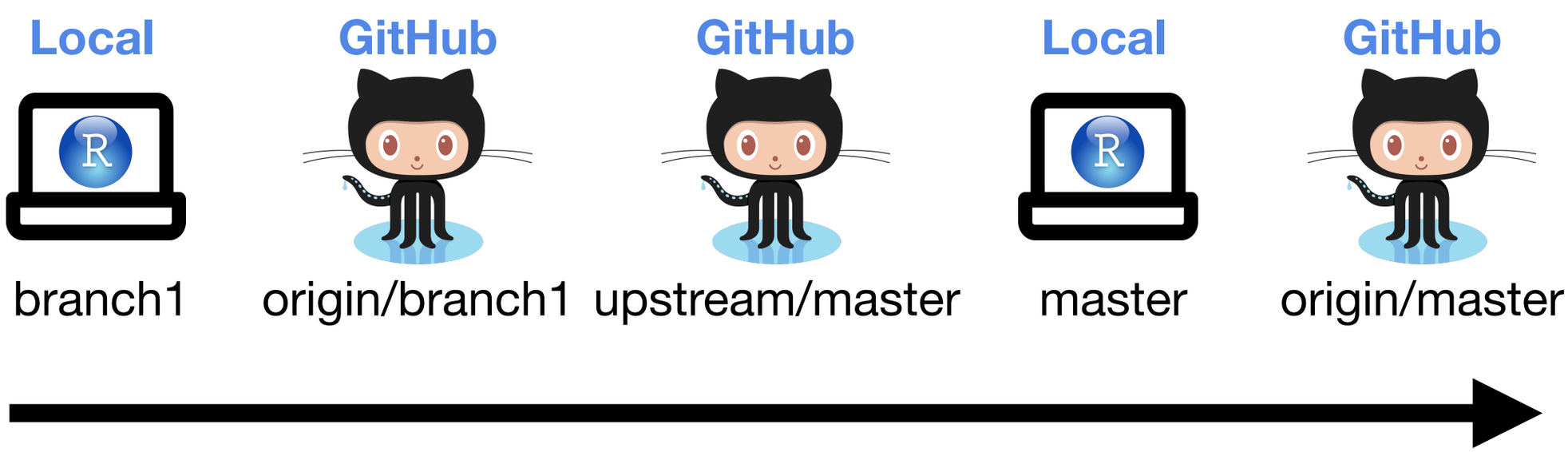
https://help.github.com/articles/syncing-a-fork/

```
> git fetch upstream

> git checkout master

> git merge upstream/master
```

4. (cont.)

new: push changes up to origin/master (GitHub)

Flow of new code:



Yes, it's not what you might expect!