Git/GitHub Workflows

Why?

1. Version Control
2. Sharing
3. Collaboration

Git/GitHub Workflows that will be covered here

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

Git/GitHub Workflows

1. **GitHub only**

https://jtr13.github.io/EDAV/contribute.html

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

1. GitHub only

GitHub only

- Create repos and upload files to share them.
- Other people can copy (fork) the repository, submit pull requests, and/or create issues.
- The best formats for reading material on GitHub without downloading are: **.md** (knit from .Rmd with output: github_document)**, .pdf**

GitHub only

Notes:

- The new repository has one branch and it is called “main”: main
- If you don't provide a commit message when you upload the file, you will get the default "Add files via upload"
- You can create files on GitHub. The default commit message in this case is: "Create <filename>"

GitHub only

Example:

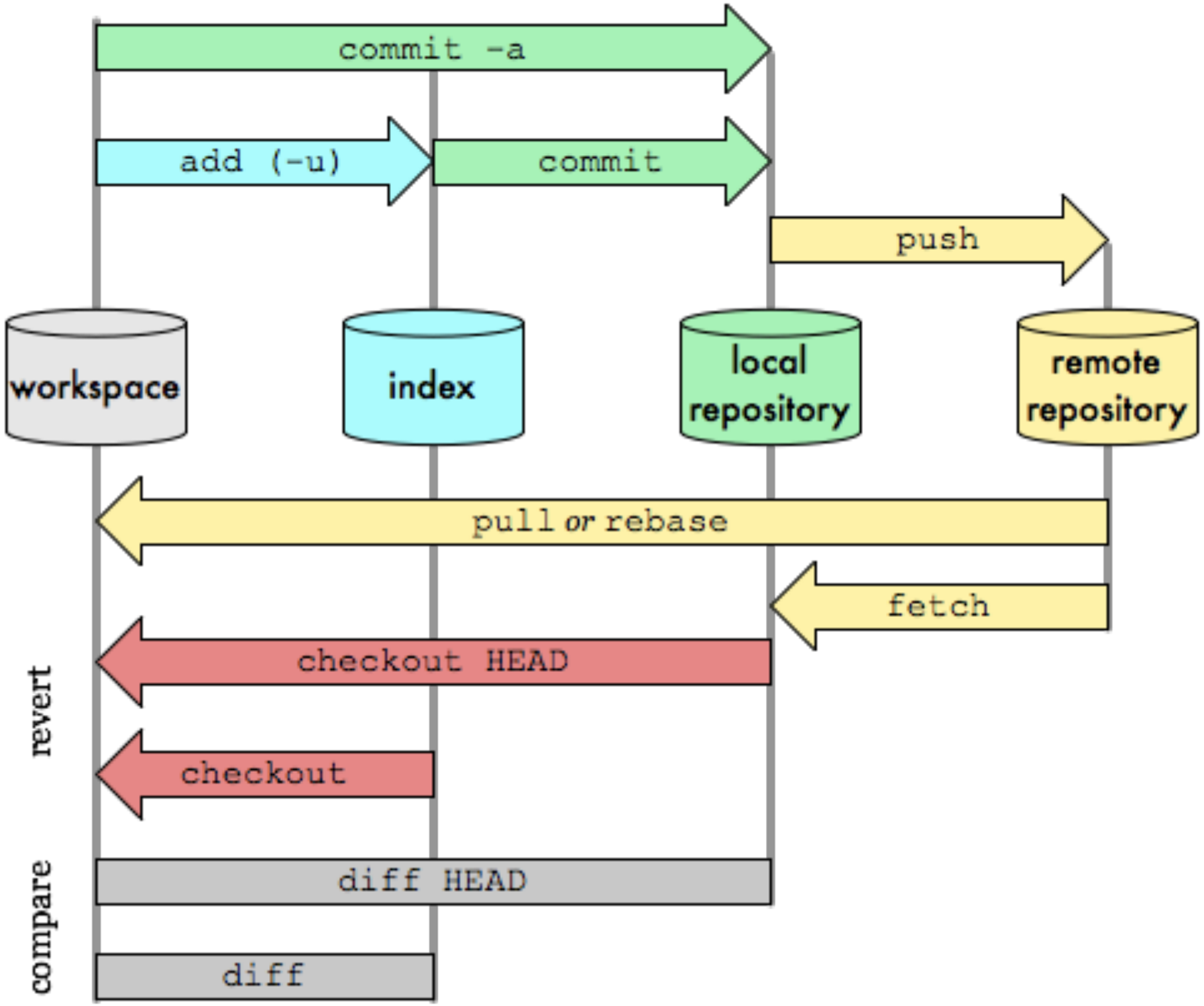
https://github.com/jtr13/codehelp/blob/main/R/pivot_longer.Rmd

Git/GitHub Workflows

1. GitHub only
2. **GitHub + local main branch**
   https://edav.info/github.html#the-no-branch-workflow
3. GitHub + local main plus additional branches on your repo
4. Contribute to someone else's repo

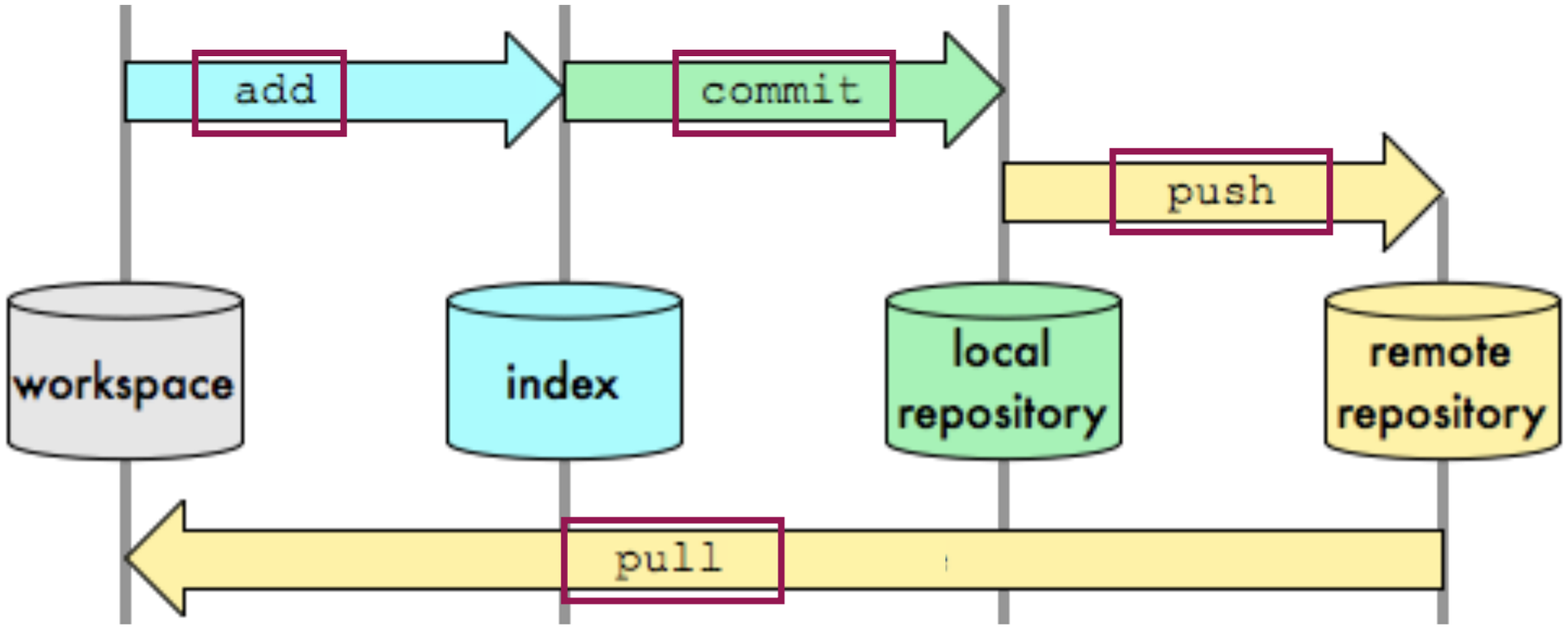
Git Data Transport Commands

http://osteele.com



Git Data Transport Commands

http://osteele.com



2. Create a local clone of our GitHub repository

Happy Git with R by Jenny Bryan et al.

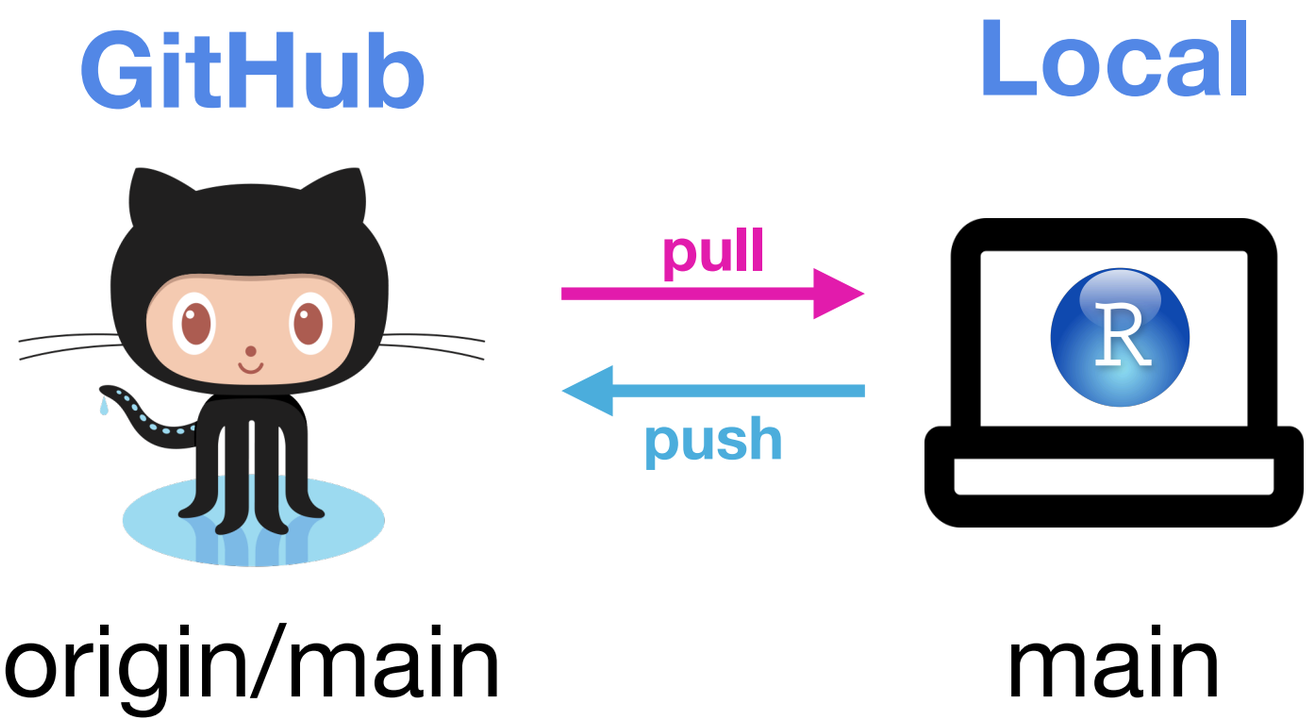
1. Create a GitHub account: https://github.com/join
2. Install git locally: https://happygitwithr.com/install-git.html
3. Introduce yourself to git:

   https://happygitwithr.com/hello-git.html
4. Connect to GitHub:

   https://happygitwithr.com/push-pull-github.html
5. Connect RStudio to git and GitHub:

   https://happygitwithr.com/rstudio-git-github.html

Our new model



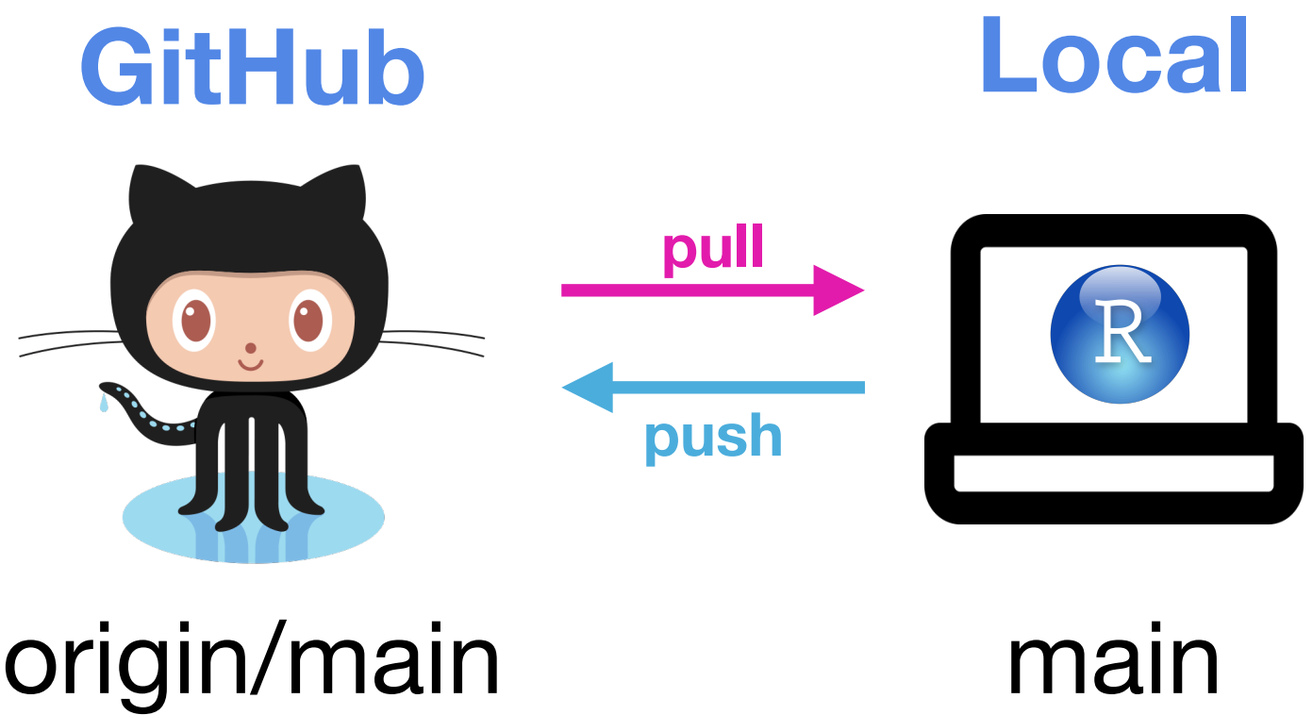
- This workflow is described in more detail in *Happy Git with R,* Chapter 16 "New project, GitHub first"

To begin: clone the repo

- This only needs to be done once.
- Click this on GitHub: 

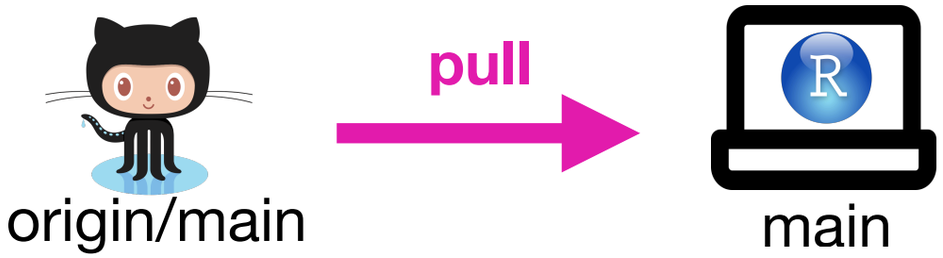
- Copy the link.
- Switch to RStudio.
- Click: "File" "New Project..."
 "Version Control"  "Git"
- Paste the URL from GitHub, click "Create Project" and we're ready to go.

Now we're ready to start.



The workflow is: **pull, work, commit/push.** Since we just cloned the repo, we don't really need to start with pull, but we will do so anyway so we start the pattern on step 1.

Step 1. Pull

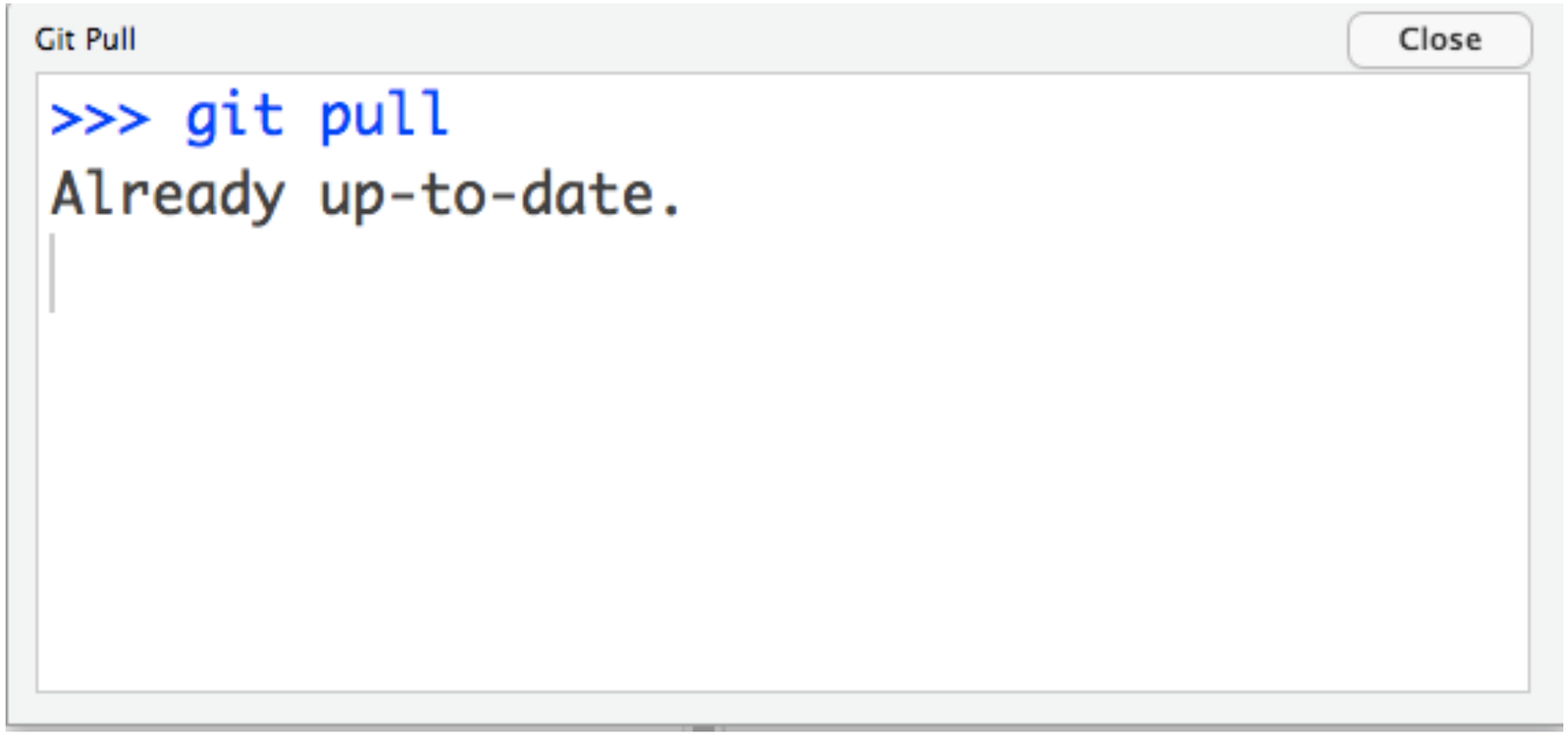


- We want to make sure that we begin working locally, we're up-to-date with the remote.

- Since nothing has changed we will get a message that we're already up to date.

Step 1. Pull

- After clicking the pull button (down arrow) in the Git pane, we see:

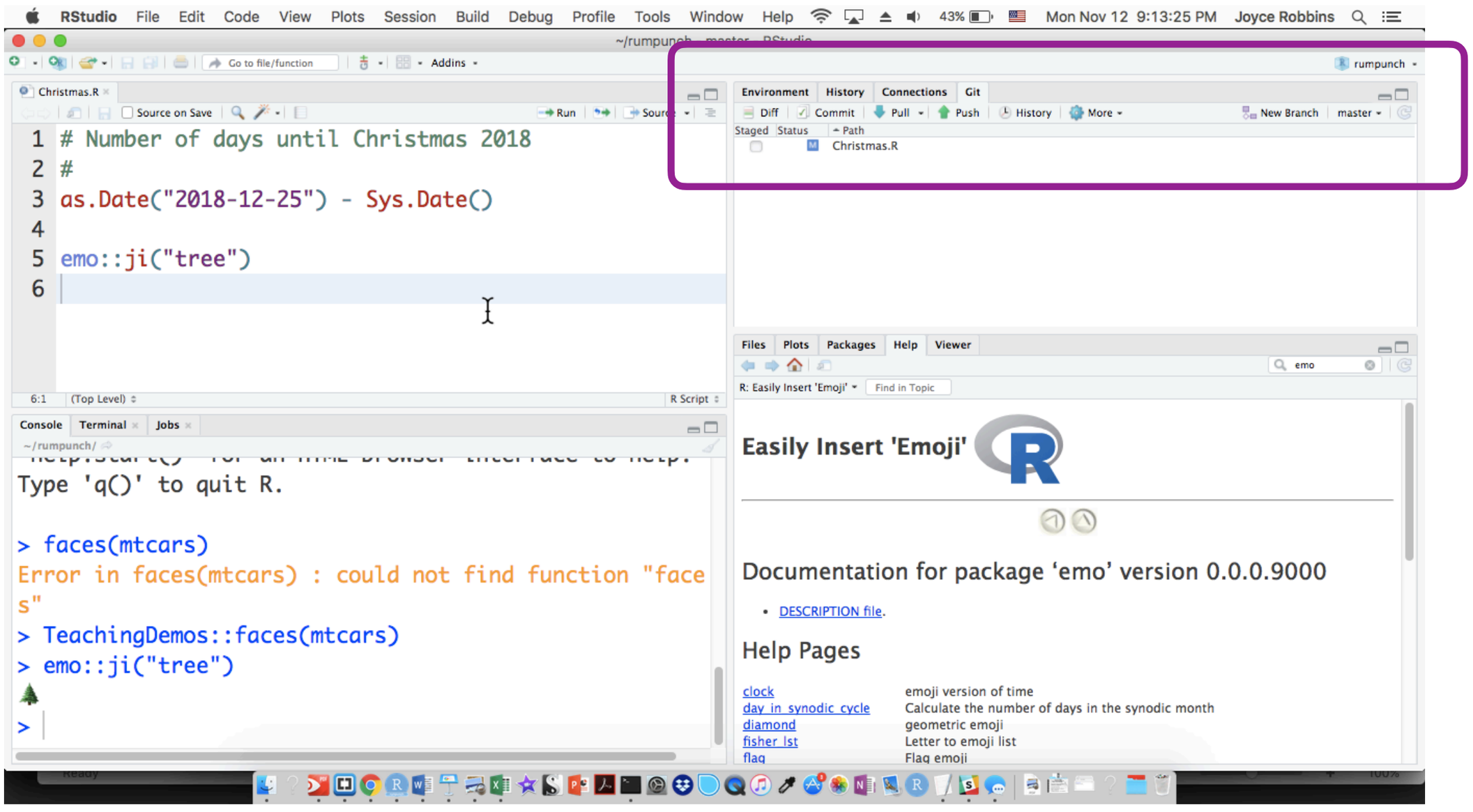


Workflow 2

Step 2. Work

main

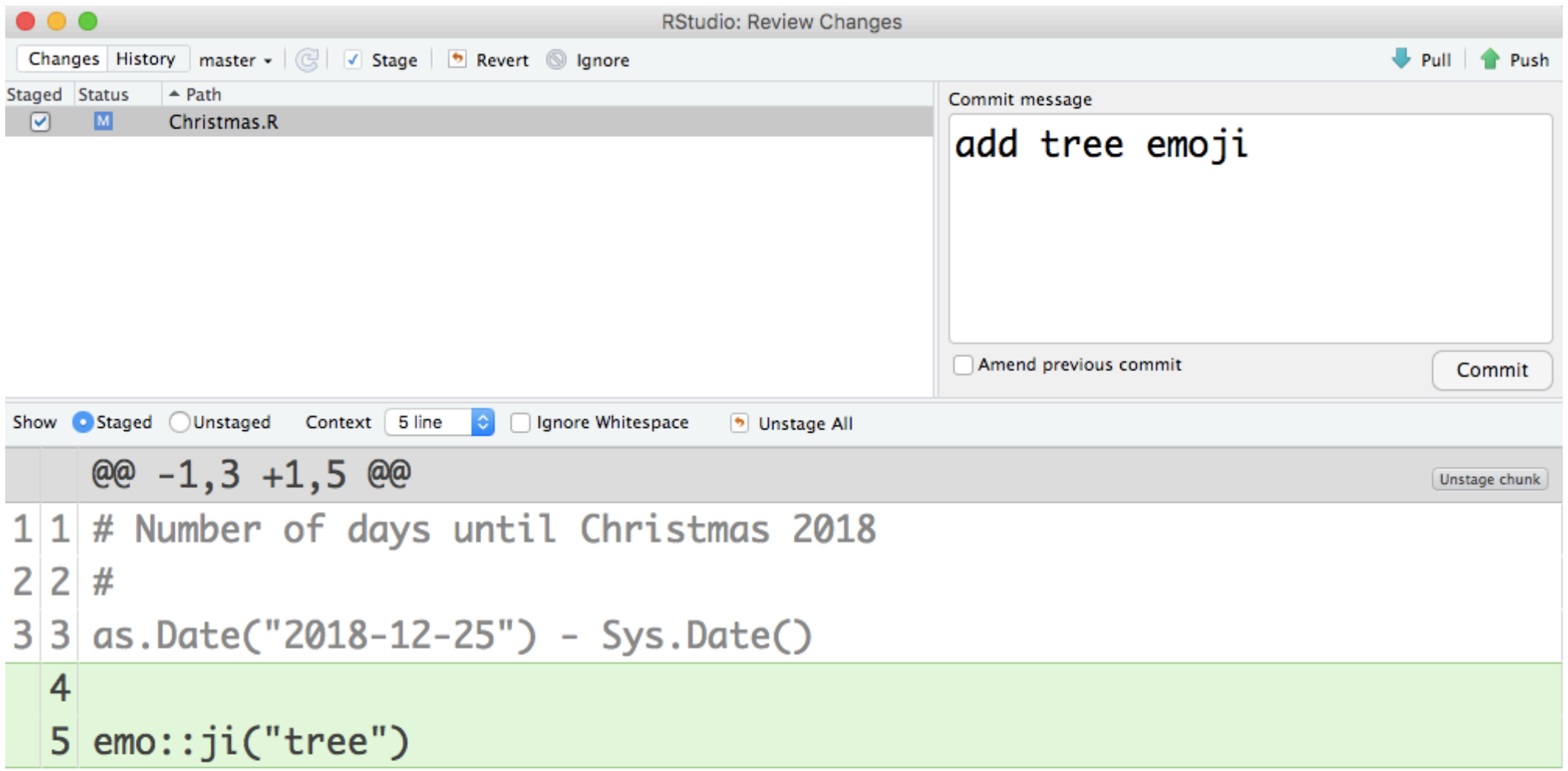
(demo in RStudio)



Workflow 2

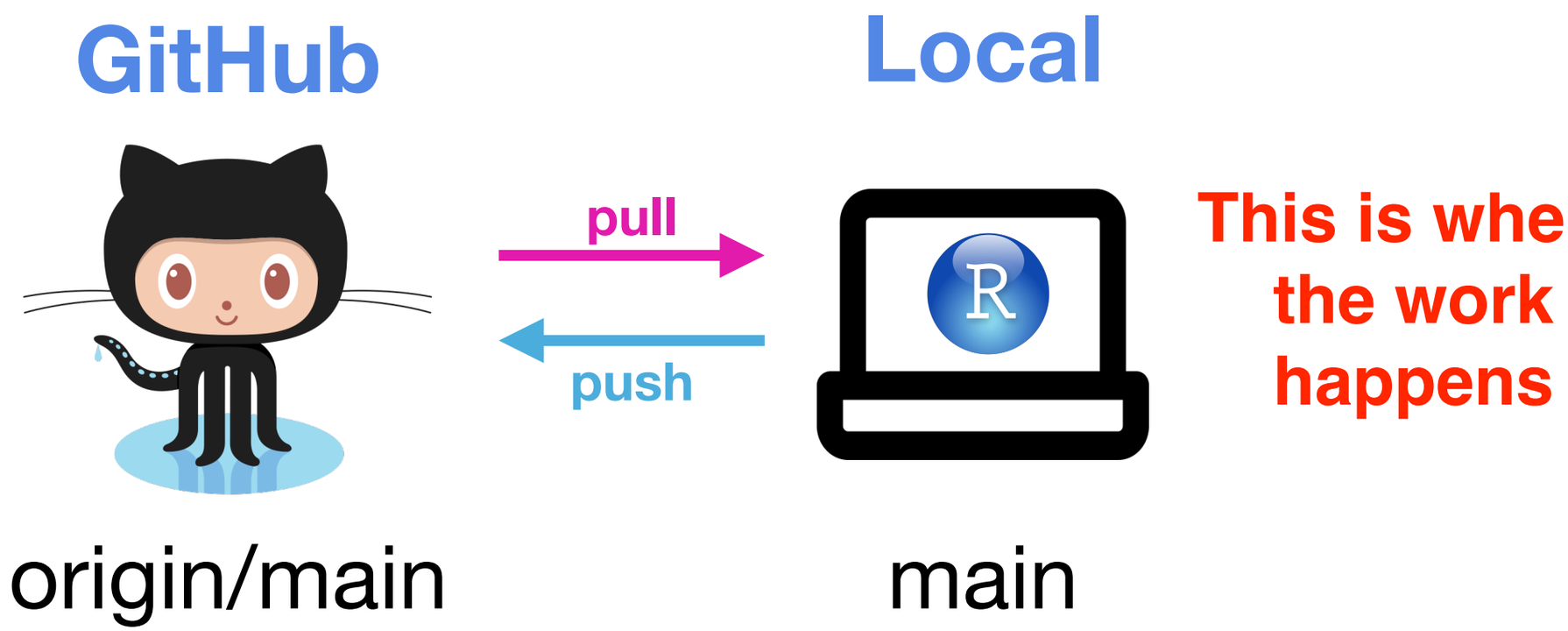
Step 3. Commit/Push

(demo in RStudio / GitHub)



Workflow 2

Our new model (summary)



Git/GitHub Workflows

1. GitHub only
2. GitHub + local main branch
3. **GitHub + local main plus additional branches on your repo**

https://jtr13.github.io/EDAV/github.html#branching-your-repo

4. Contribute to someone else's repo

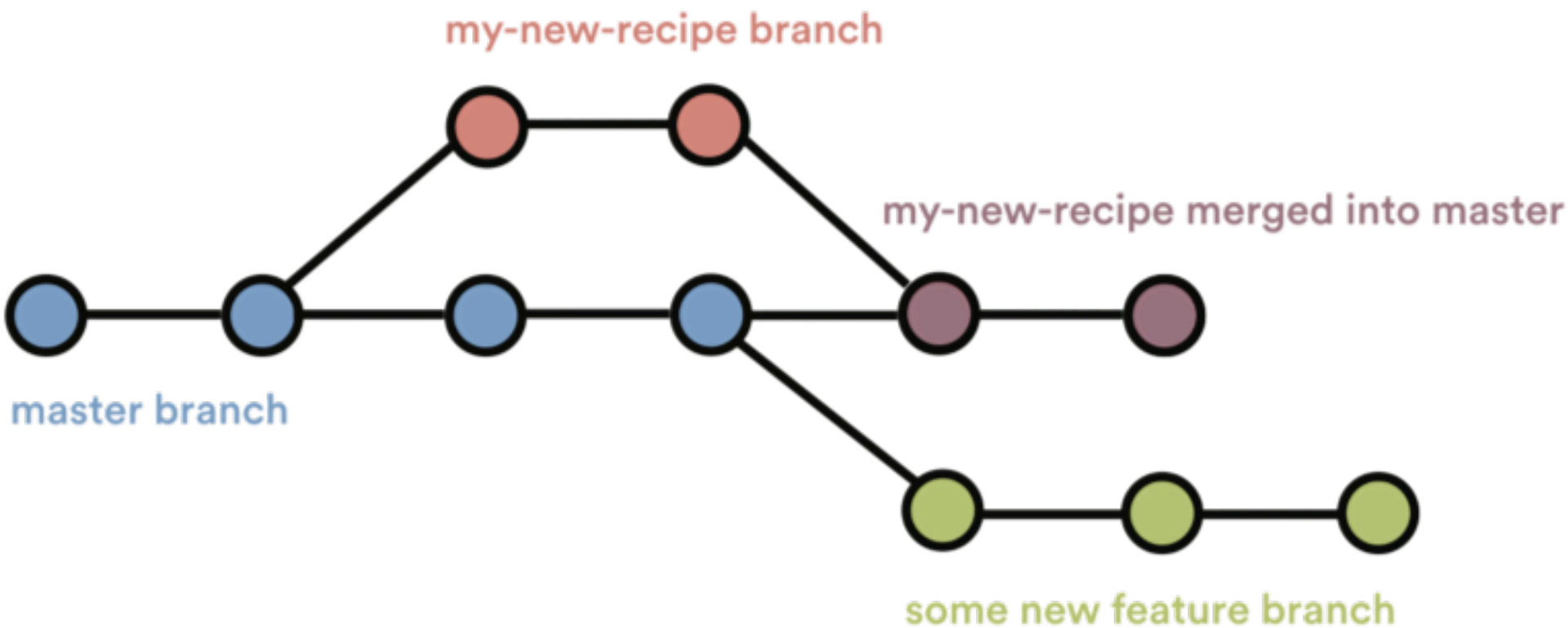
3. Create a new branch

Why?

- By working on a branch, we can allow collaborators to review our code before merging to main.
- Branching is useful regardless as it allows us to test new code without affecting the working version.

3. Remote + local
main + other branches

From the perspective of the project:



Workflow 3

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

Your perspective

Start by creating a repo on GitHub

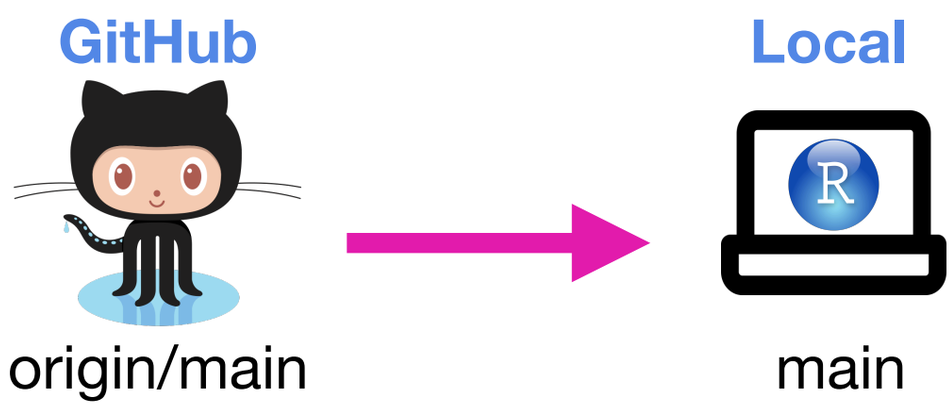
origin/main

Origin/main

Working, shared code lives here

origin/main

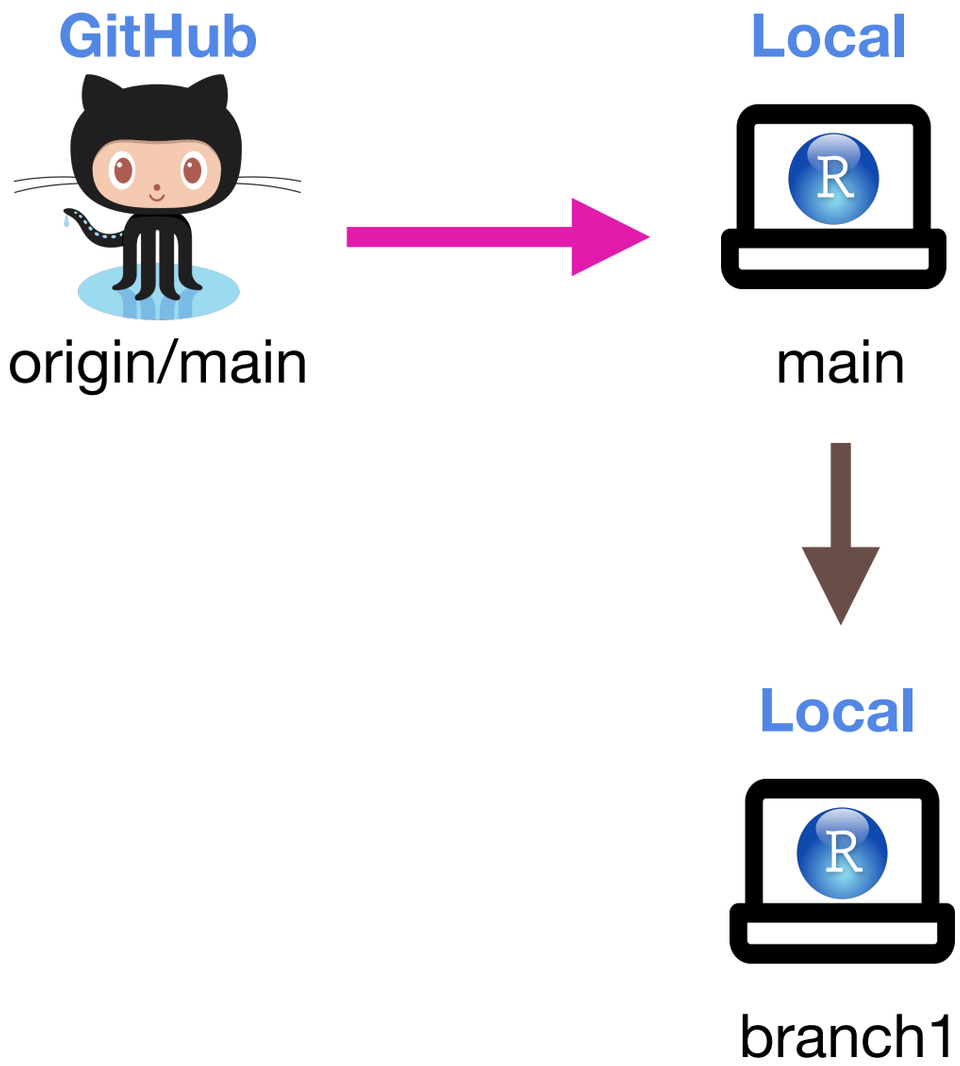
Clone it once



Make a local copy

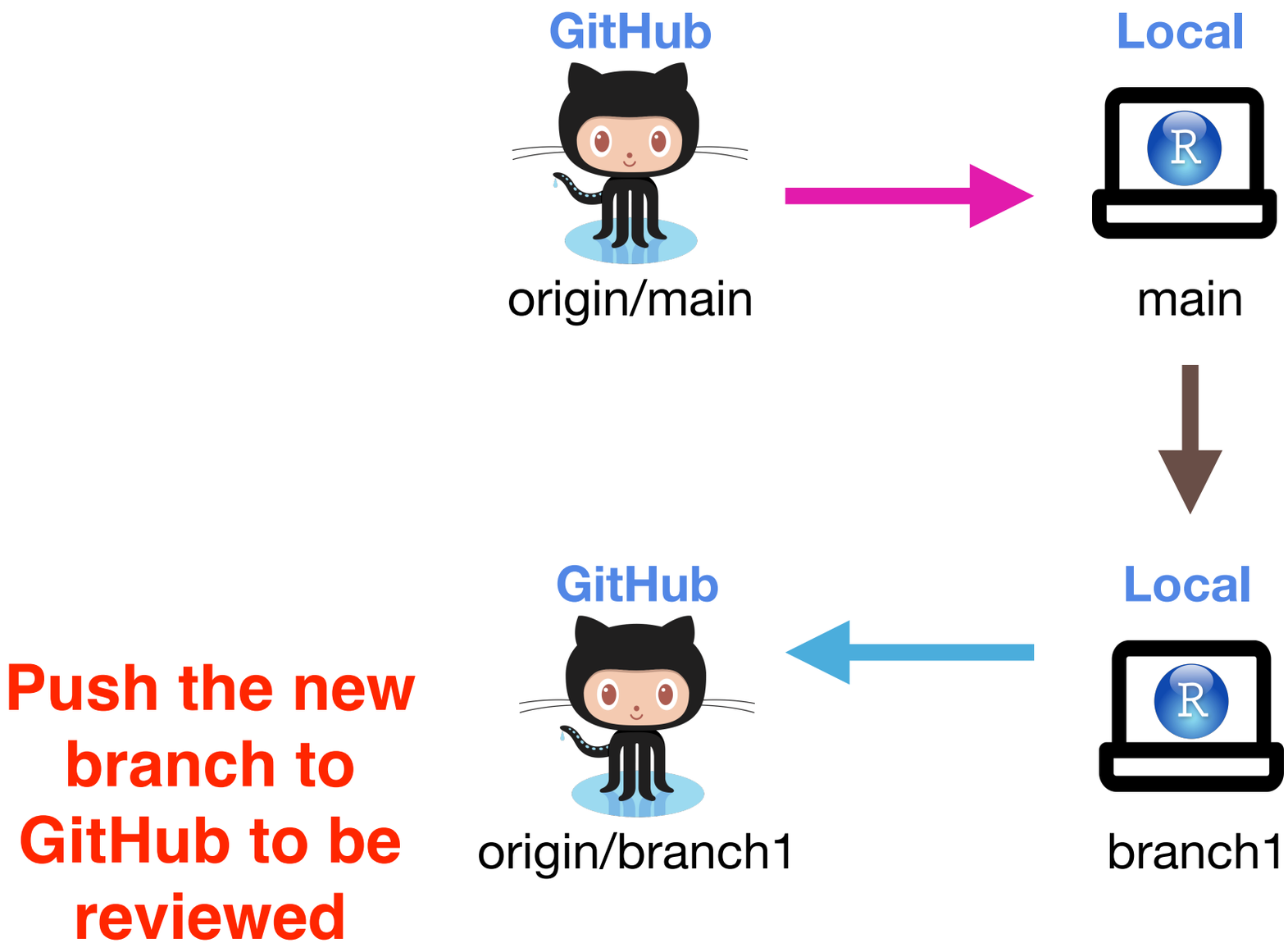
(Next time, this step will be a PULL)

Create a branch



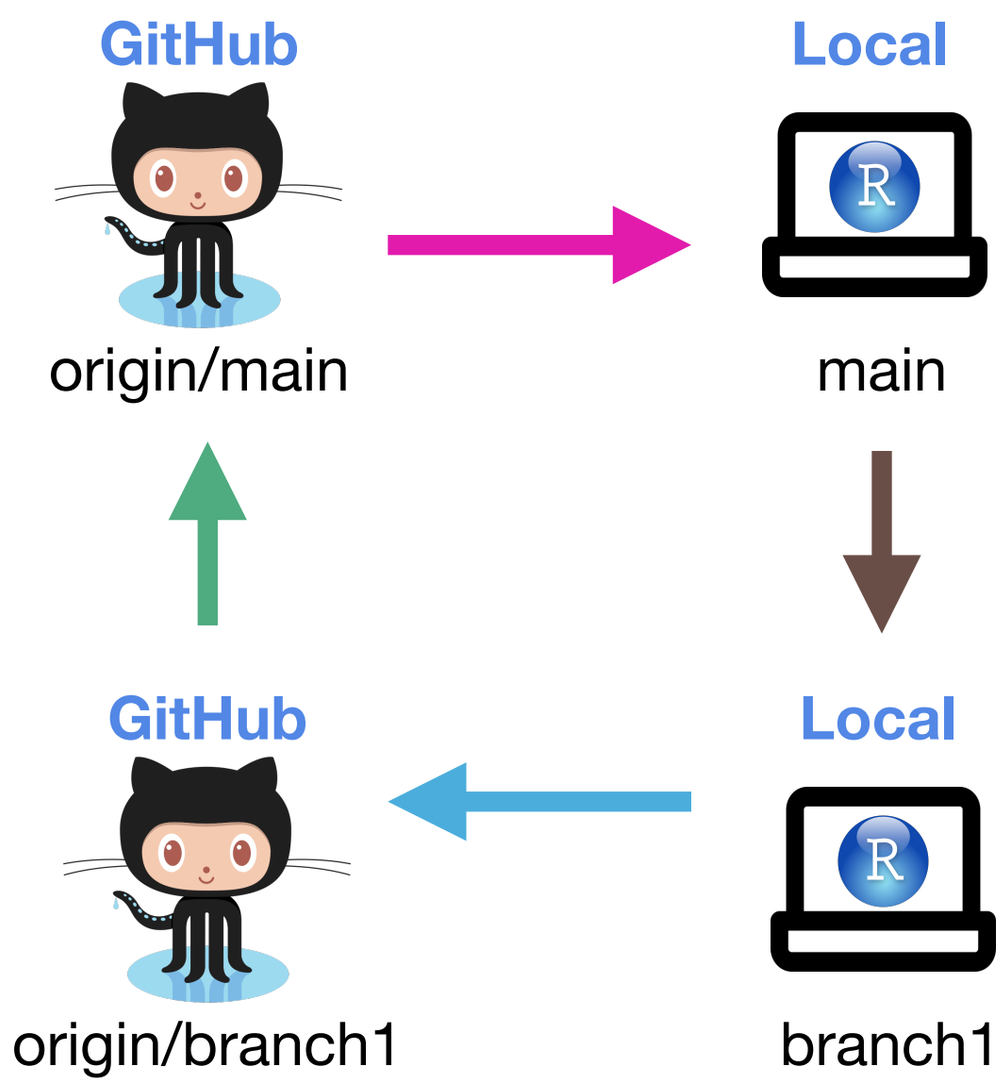
Create a branch to do your new work

Save / commit / push changes

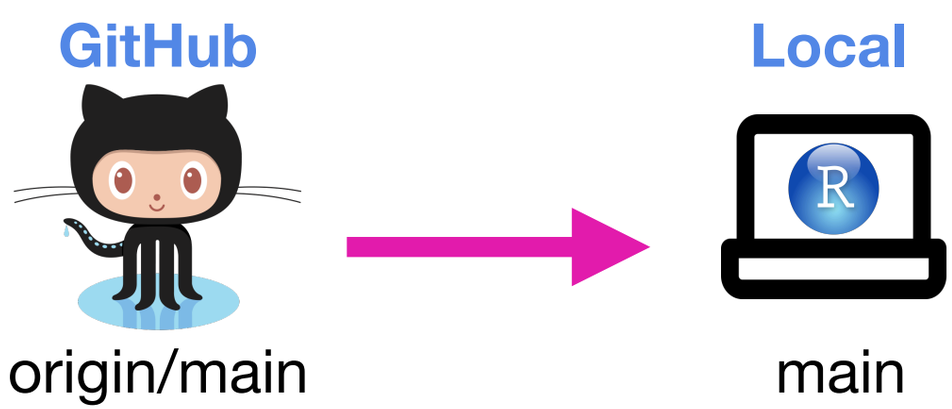


Pull request and merge

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

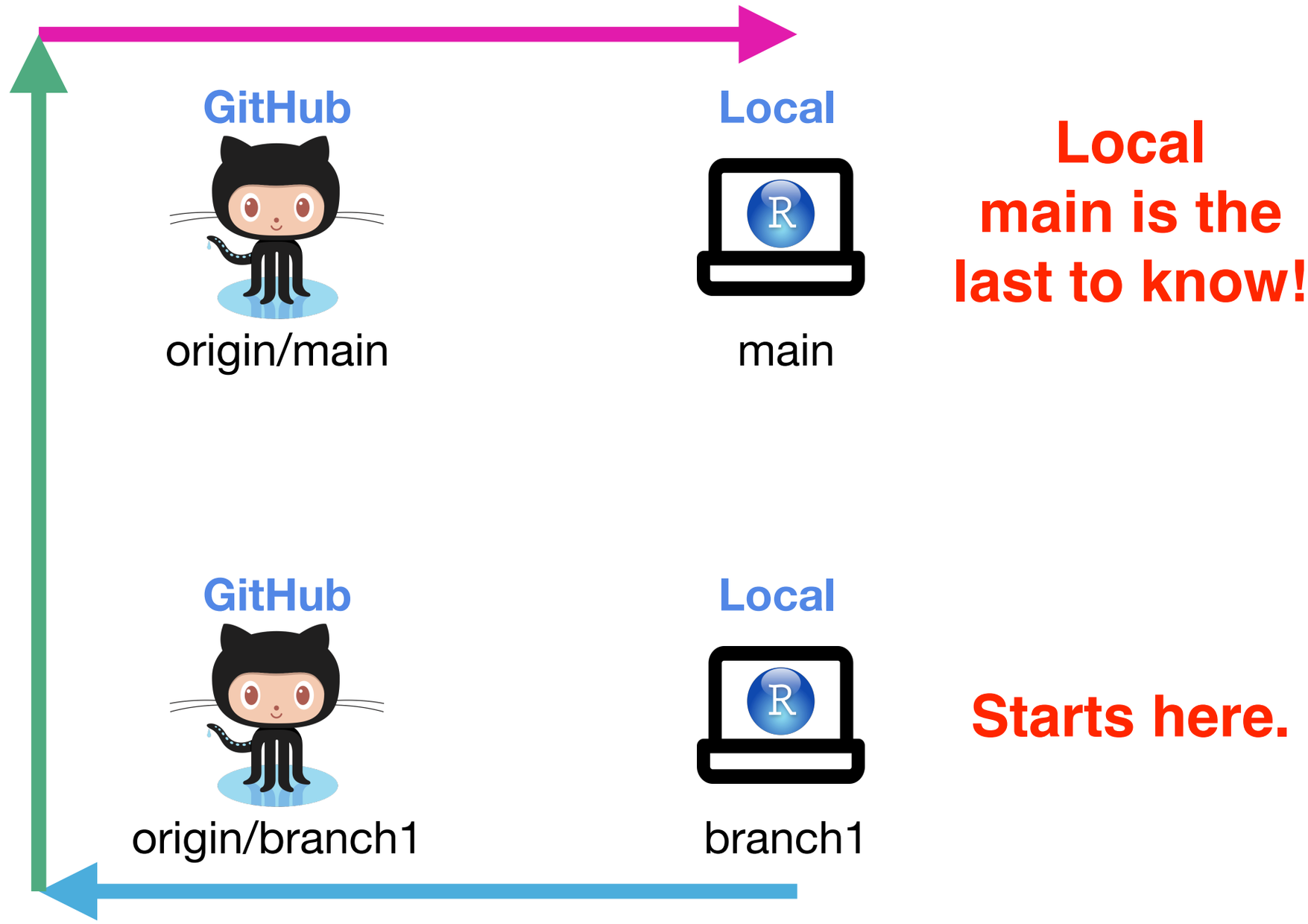


Your perspective

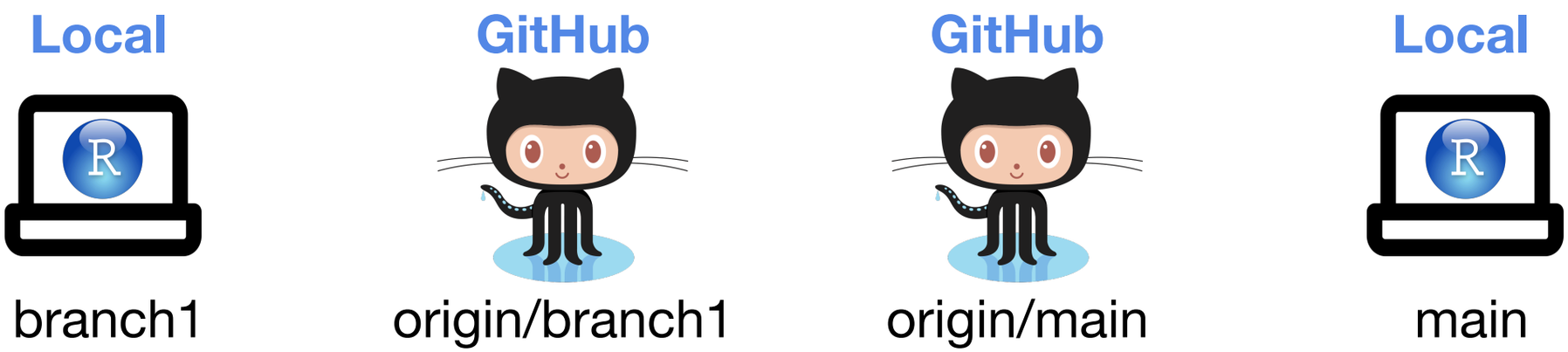


Your branch is deleted and the new stuff is pulled into your copy of the main branch.

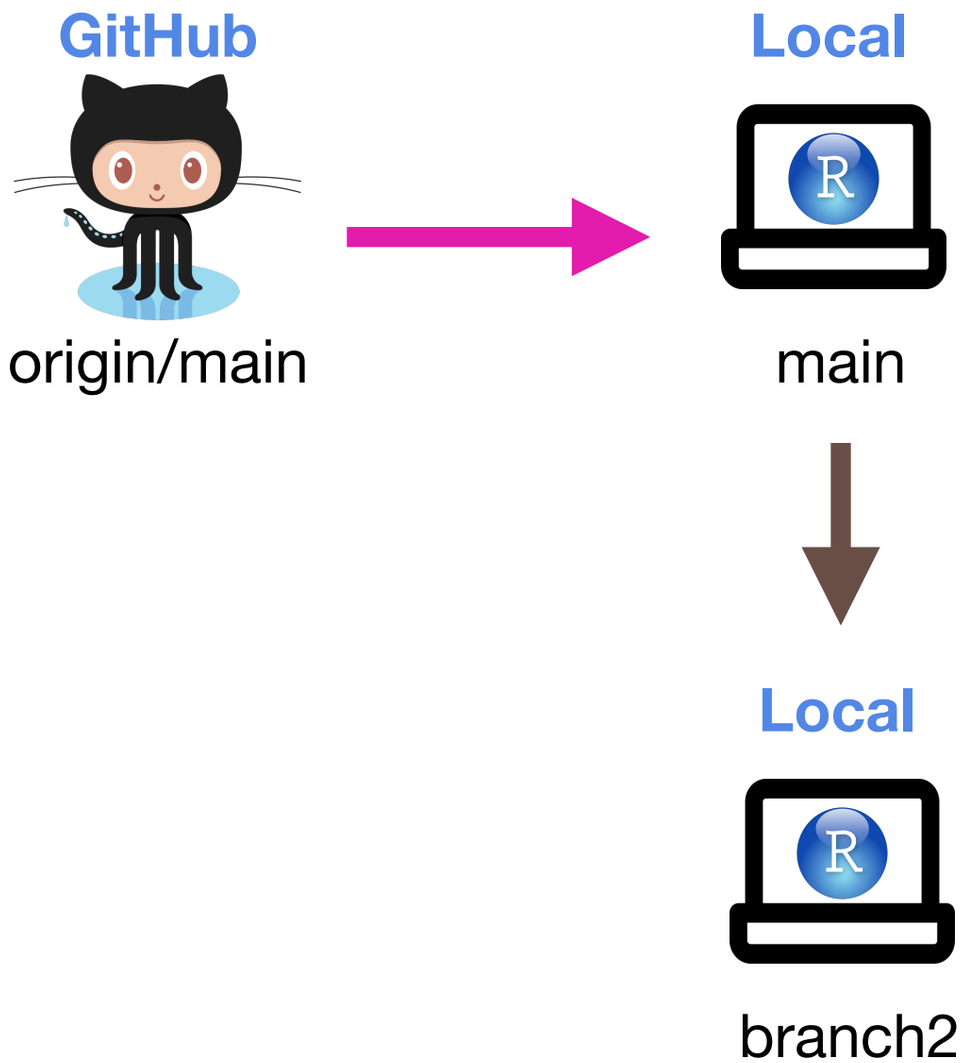
Note the flow of new code



Note the flow of new code



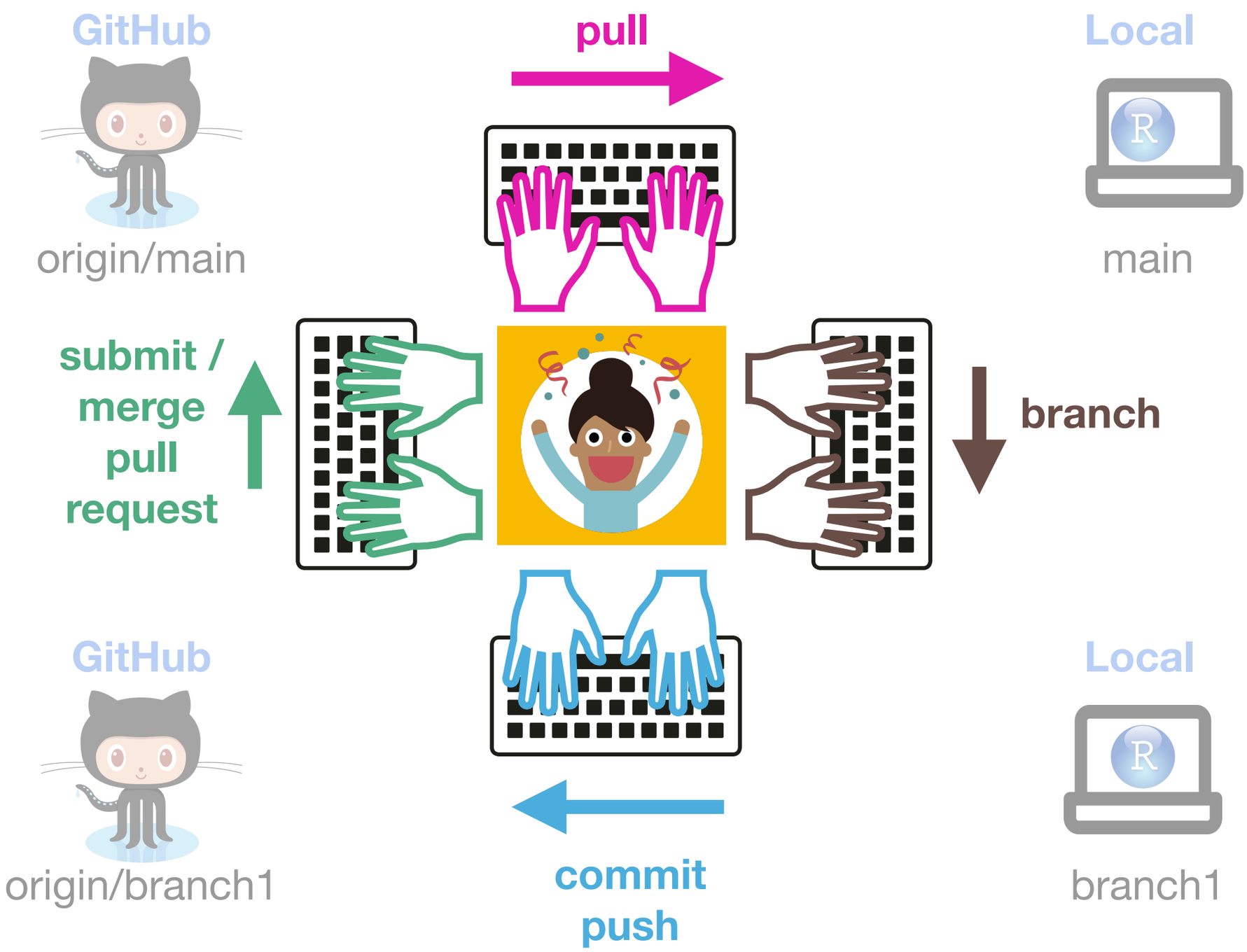
The second change...



Create a branch to do your new work

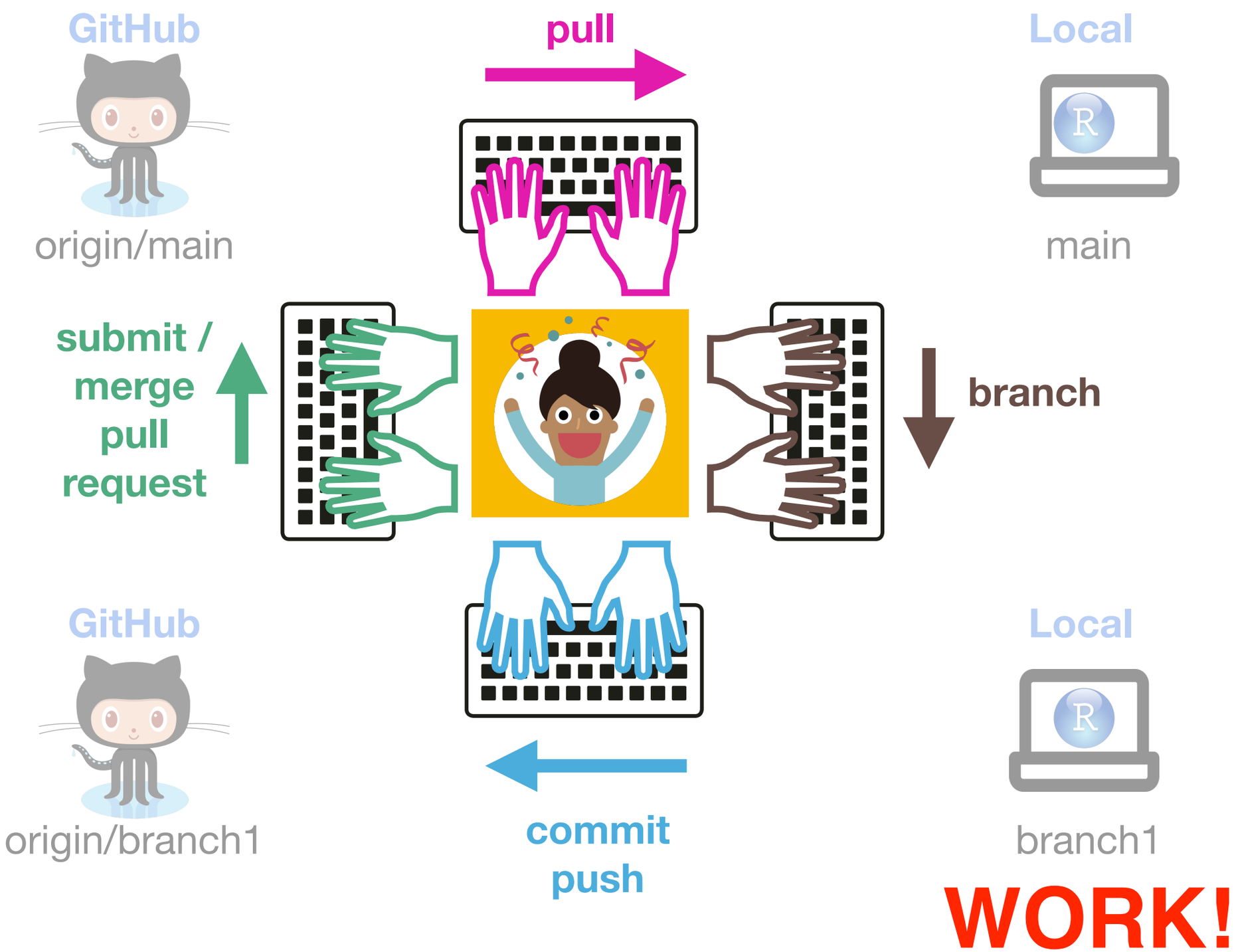
And so on and so on...

Your perspective

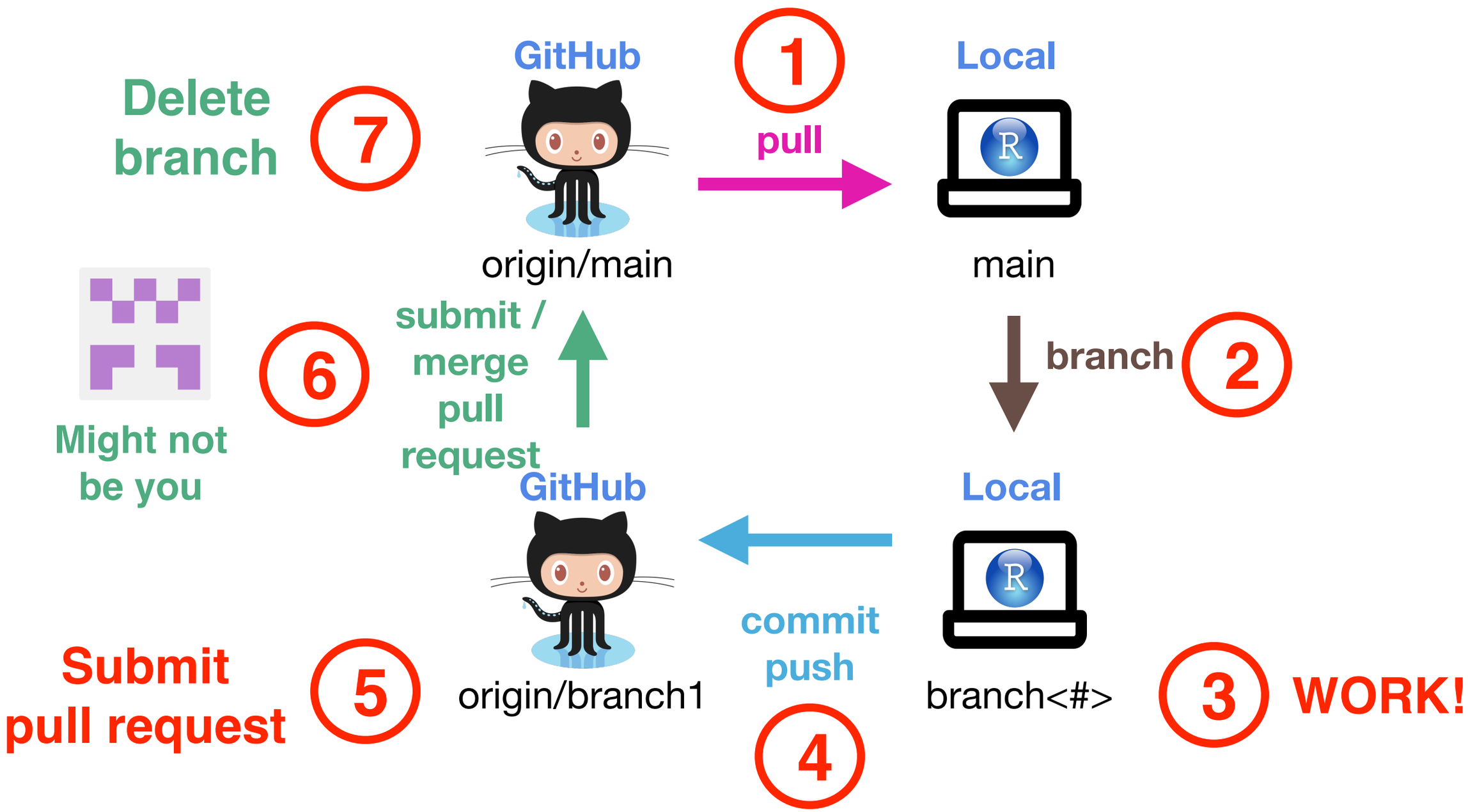


Workflow 3

Your perspective

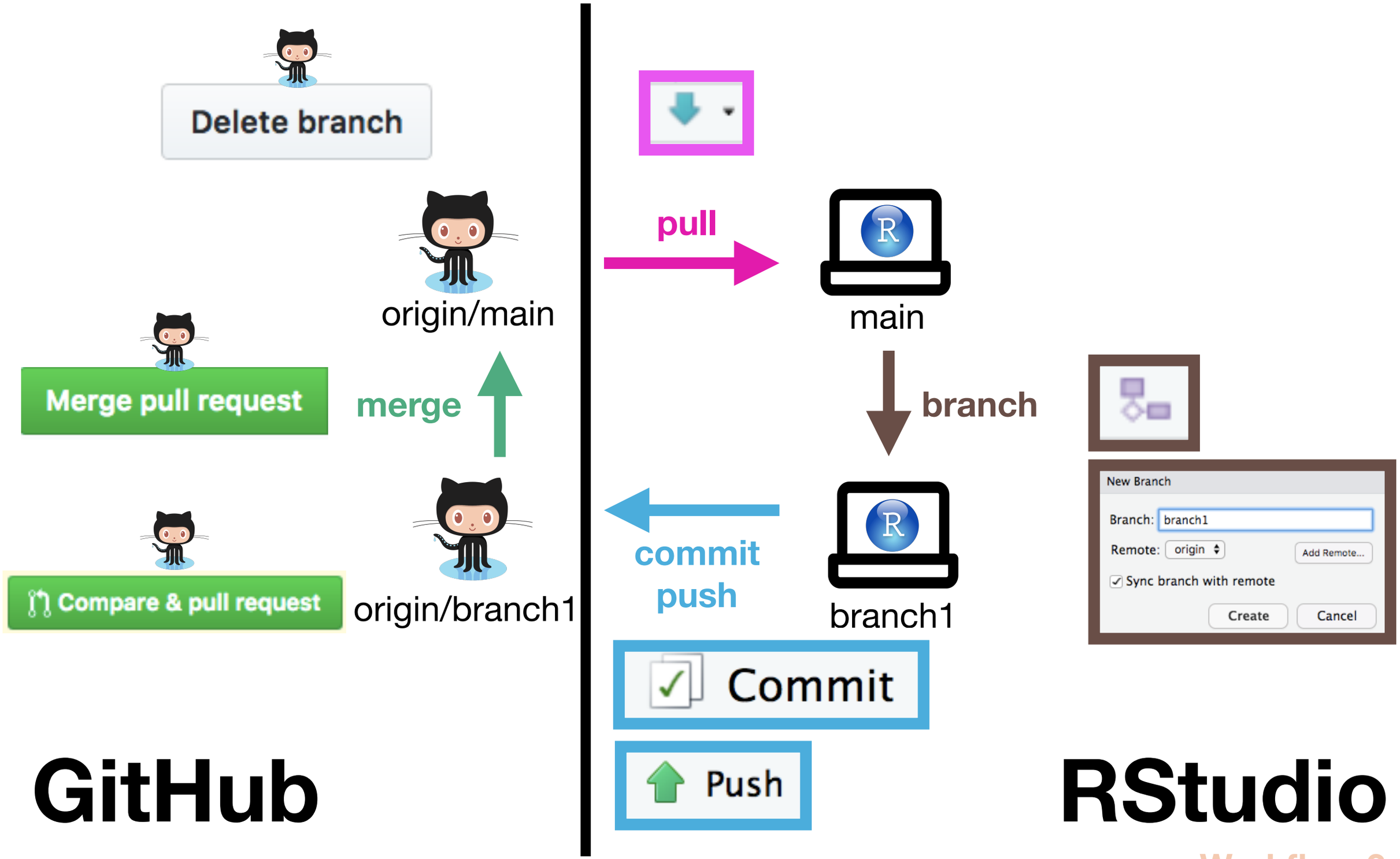


Your workflow



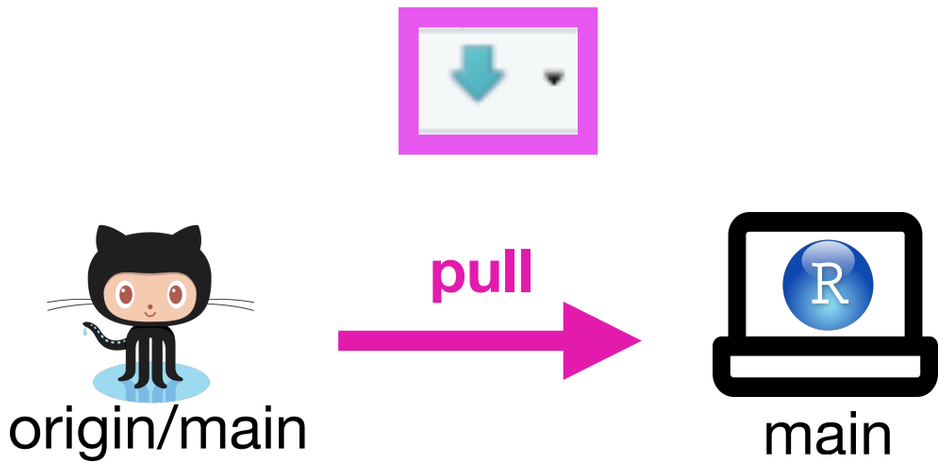
Workflow 3

What's happening where



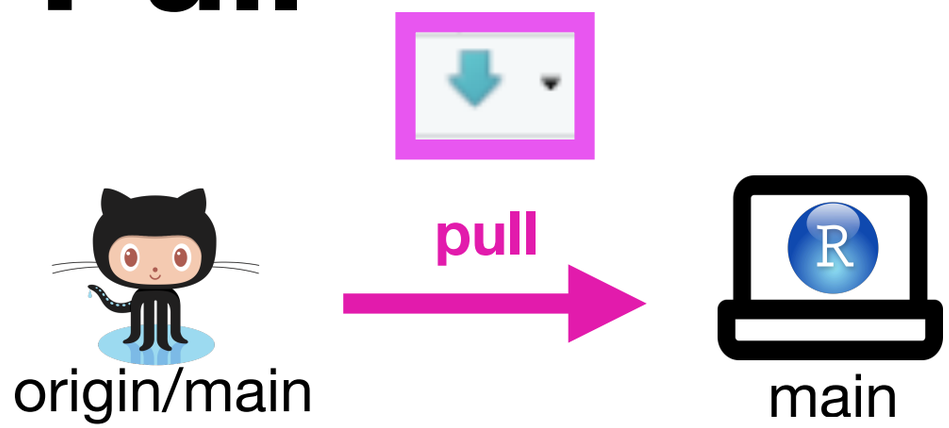
Workflow 3

Step 1. Pull



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

Step 1. Pull

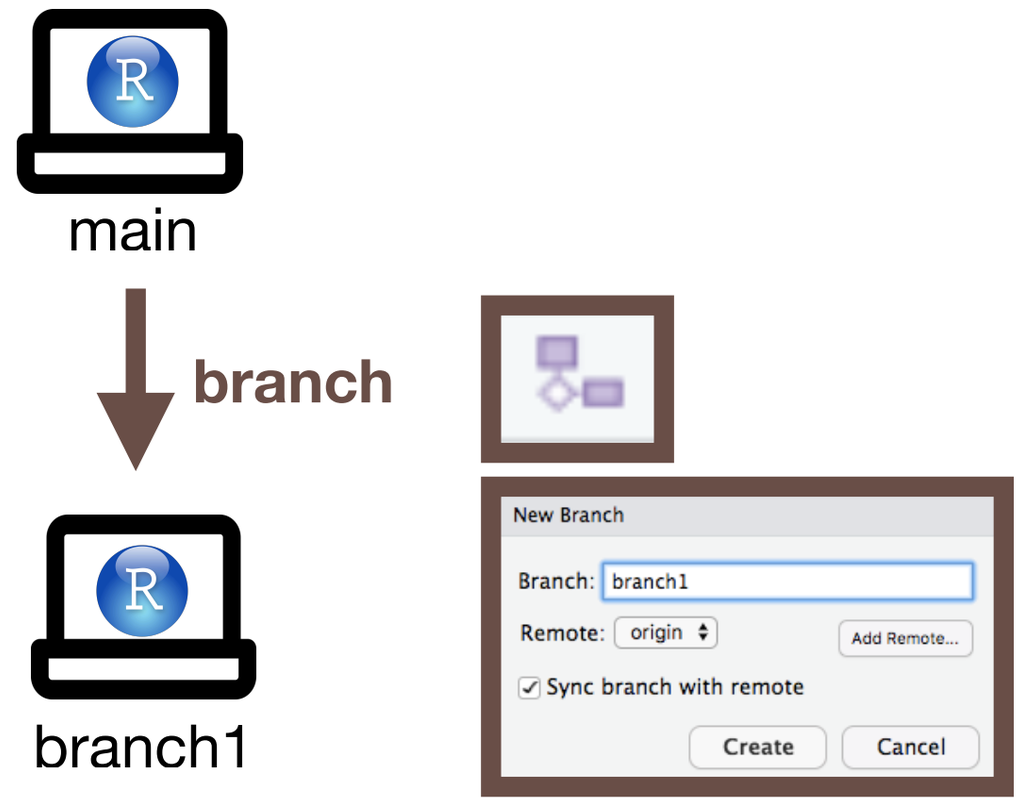


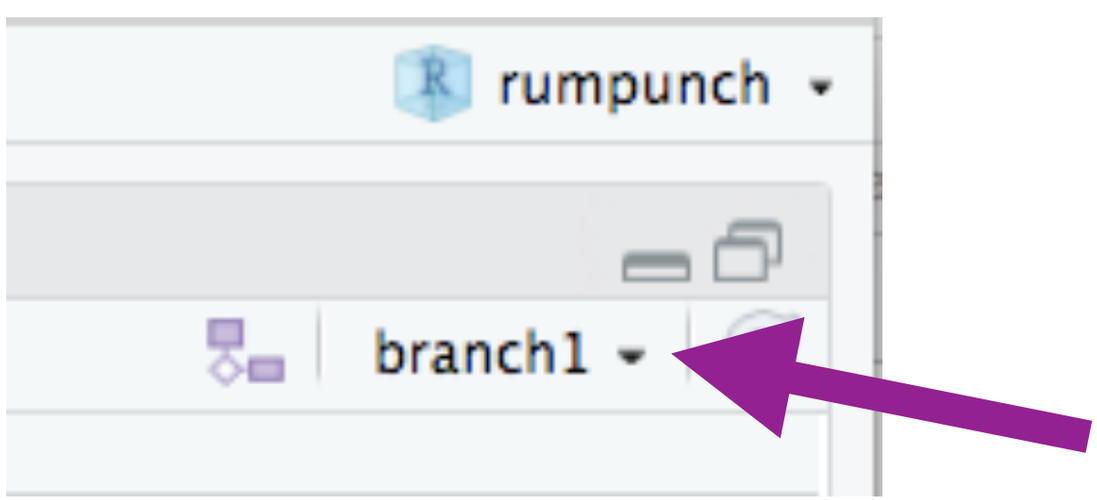
- If all goes well (no conflicts), our copy of main will be updated:

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

Step 2: Create a new branch

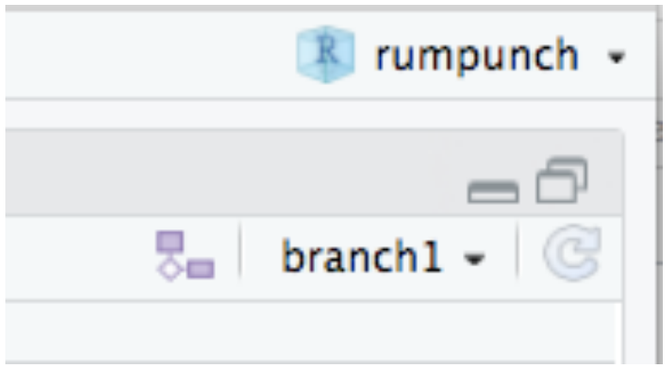
- We'll do our work on this branch.
- Check the top right corner to be sure you're in the right place:





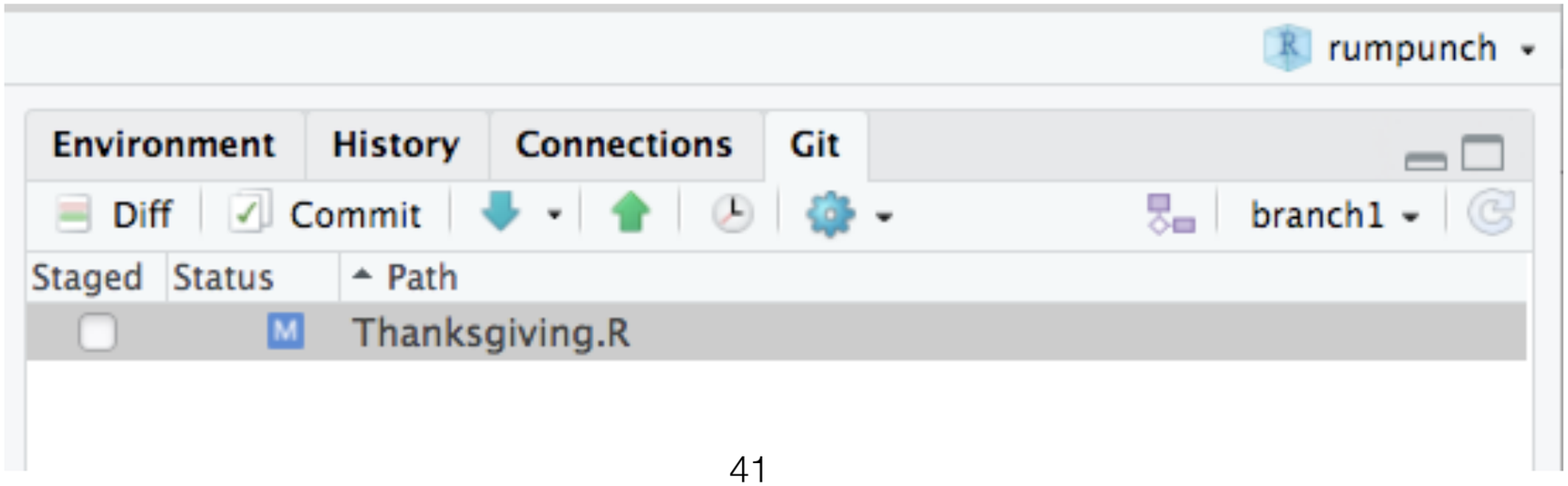
Step 3: Work

```r
## @param .op Can be a function or a quoted name of a function. If a
##   quoted name, the default environment is the [base
##   environment][rlang::base_env] unless you supply a
##   [quosure][rlang::quo].
quo_reduce <- function(..., .op) {
  stopifnot(is_symbol(.op) || is_function(.op))

  dots <- quos(...)
  if (length(dots) == 0) {
    abort("At least one expression must be given")
  } else if (length(dots) == 1) {
    return(dots[[1]])
  }

  op_quo <- as_quosure(.op, base_env())
  op <- quo_get_expr(op_quo)

  expr <- reduce(dots, function(x, y) expr((!!op)((!!x), (!!y))))
  new_quosure(expr, quo_get_env(op_quo))
}
```

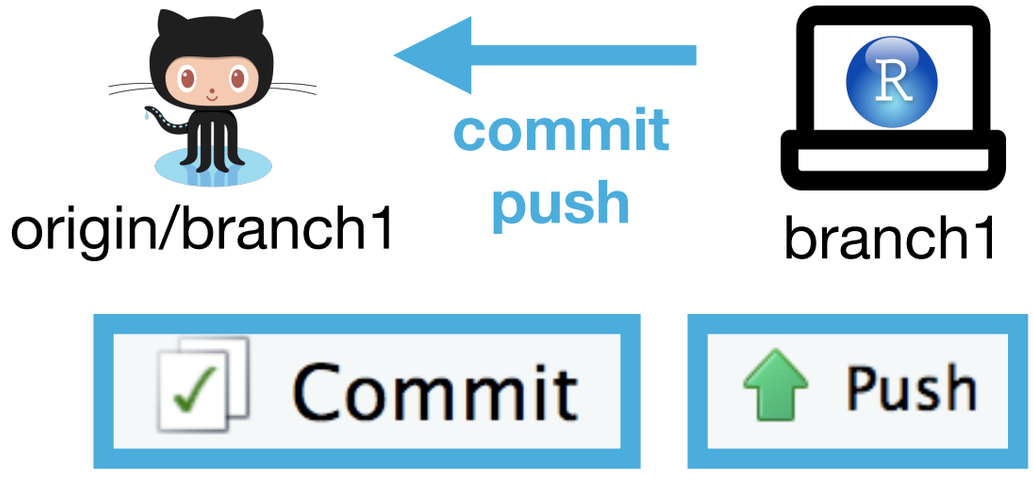


branch1

- Observe changing files in the Git pane:



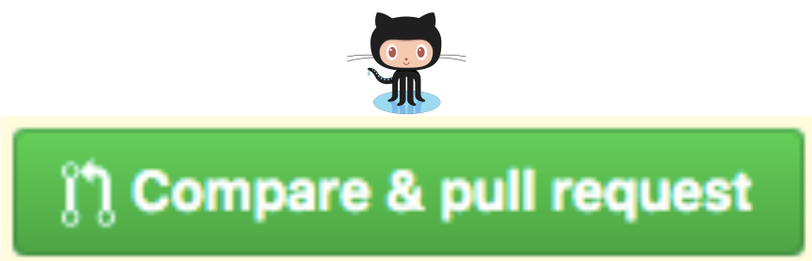
Step 4: Commit and push



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

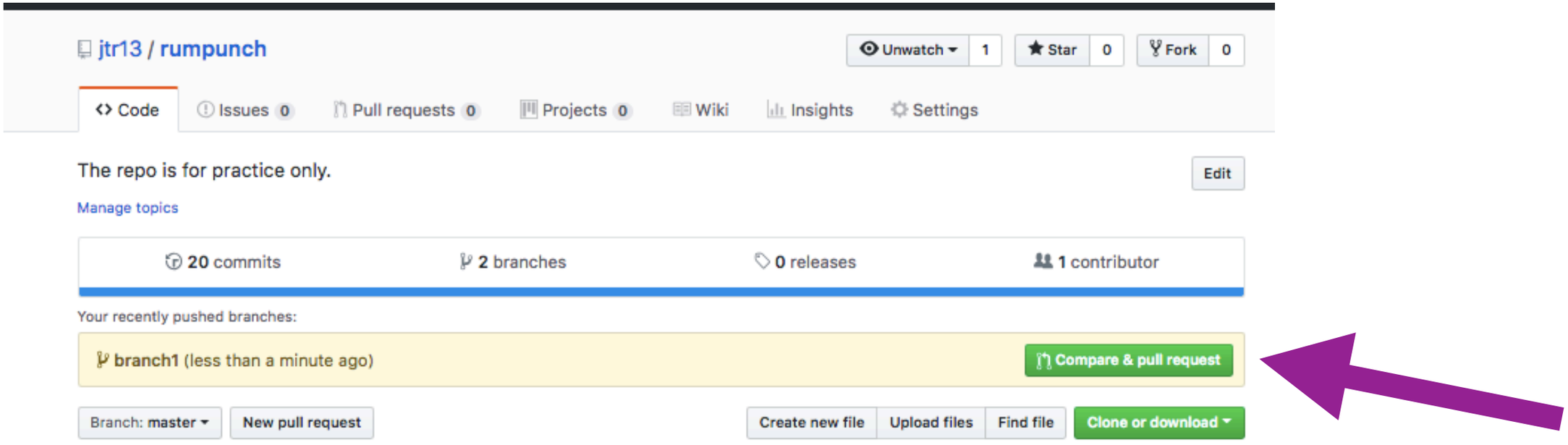
```
>>> git push origin refs/heads/branch1
To https://github.com/jtr13/rumpunch.git
   7424222..6cf5975  branch1 -> branch1
```

Step 5: Submit a pull request



origin/branch1

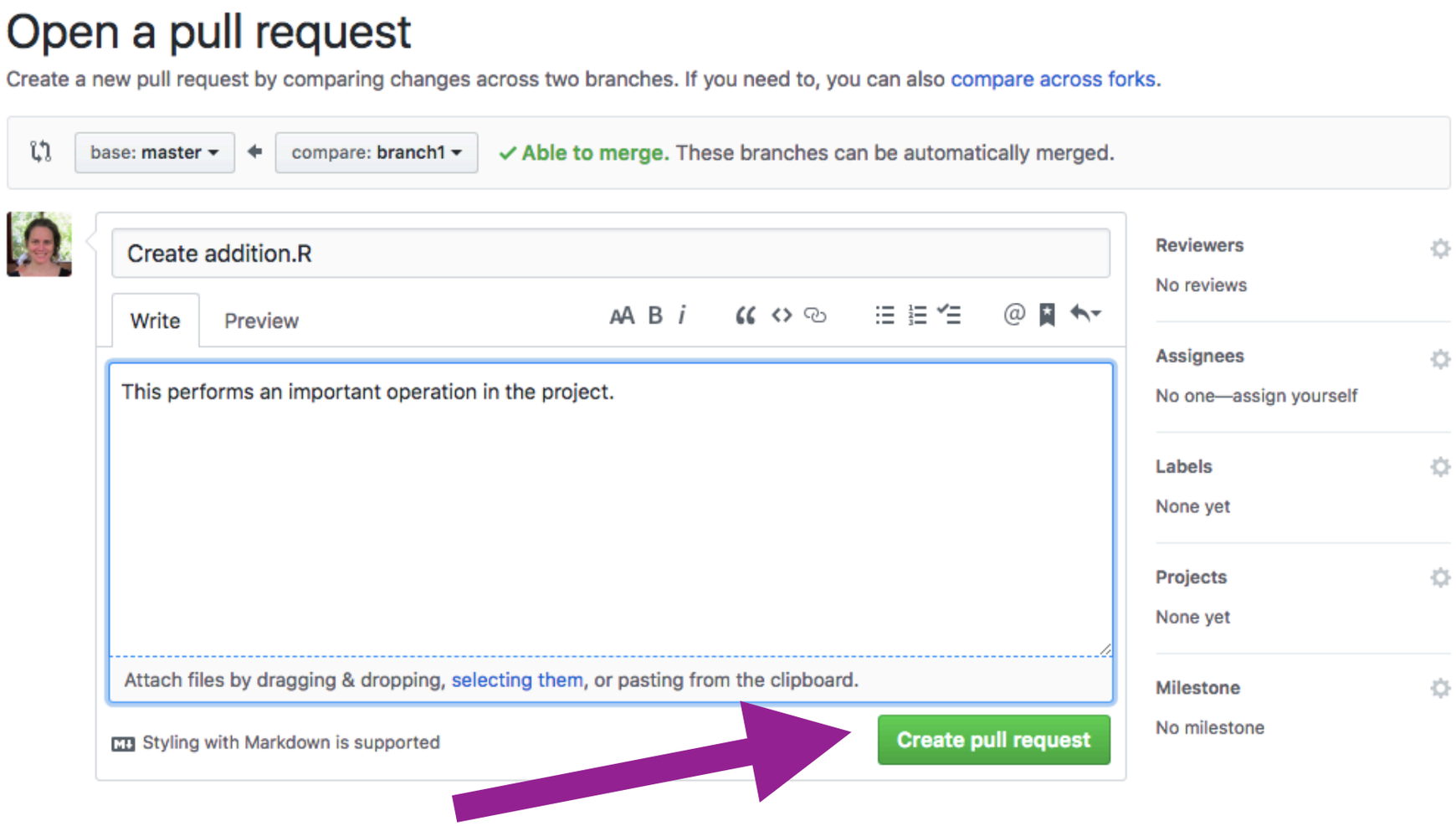
- GitHub detects a difference between the main branch and branch1:



Workflow 3

Step 5: Submit a pull request

- Click: Compare & pull request
- Add a description



- Then click "Create pull request"

Workflow 3

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.

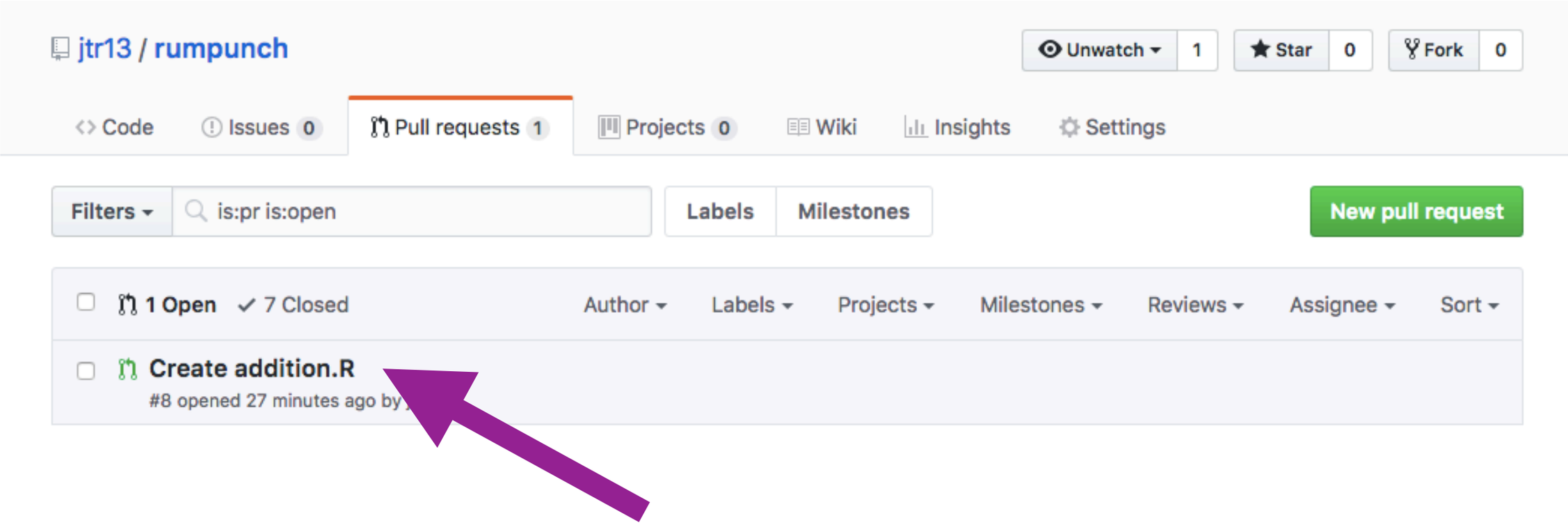
Step 6: Merging a pull request

- 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

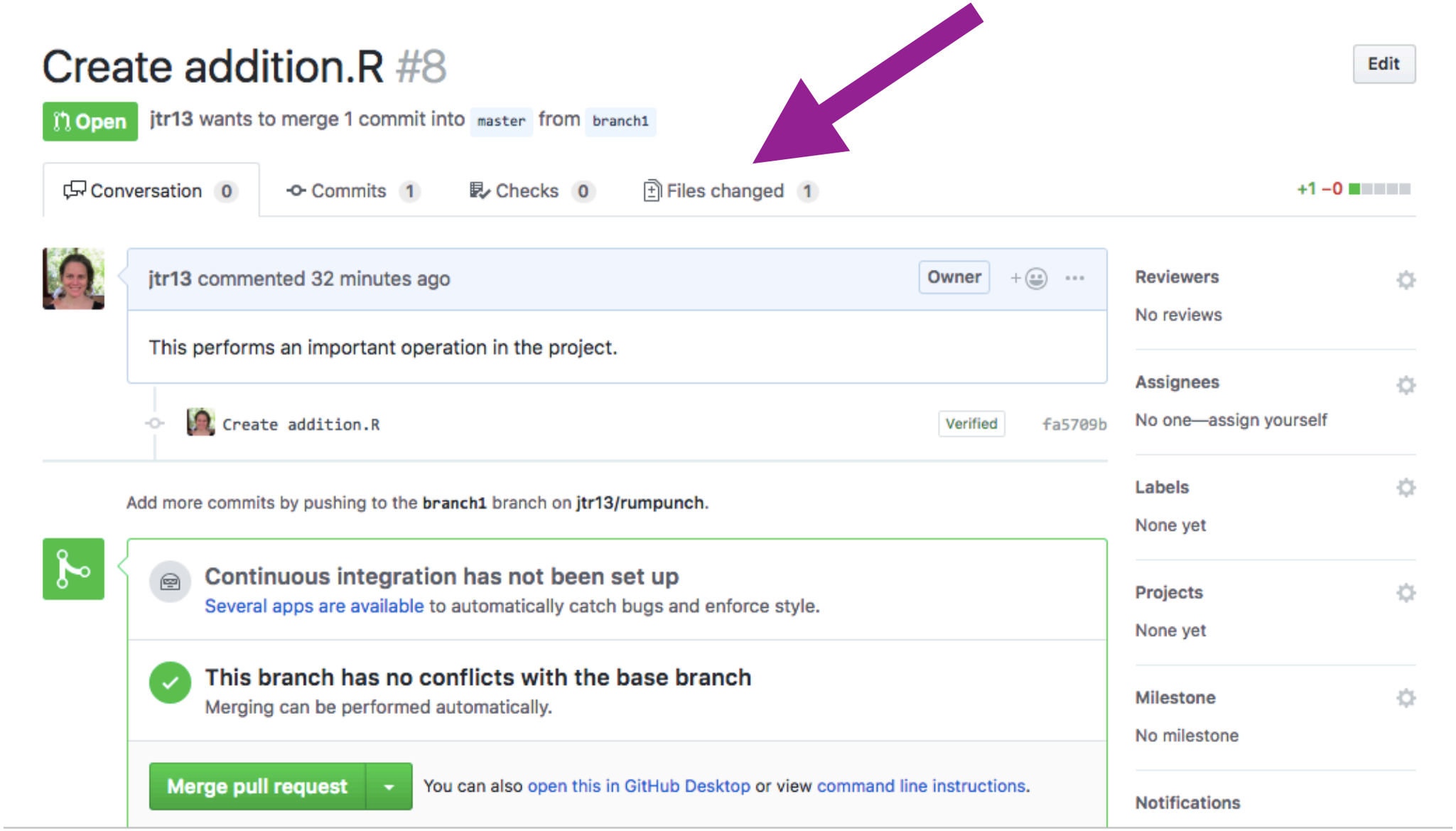
- If you're the one merging the pull request, click the "Pull Requests" tab and you'll see something like this:



- Click the title of the pull request

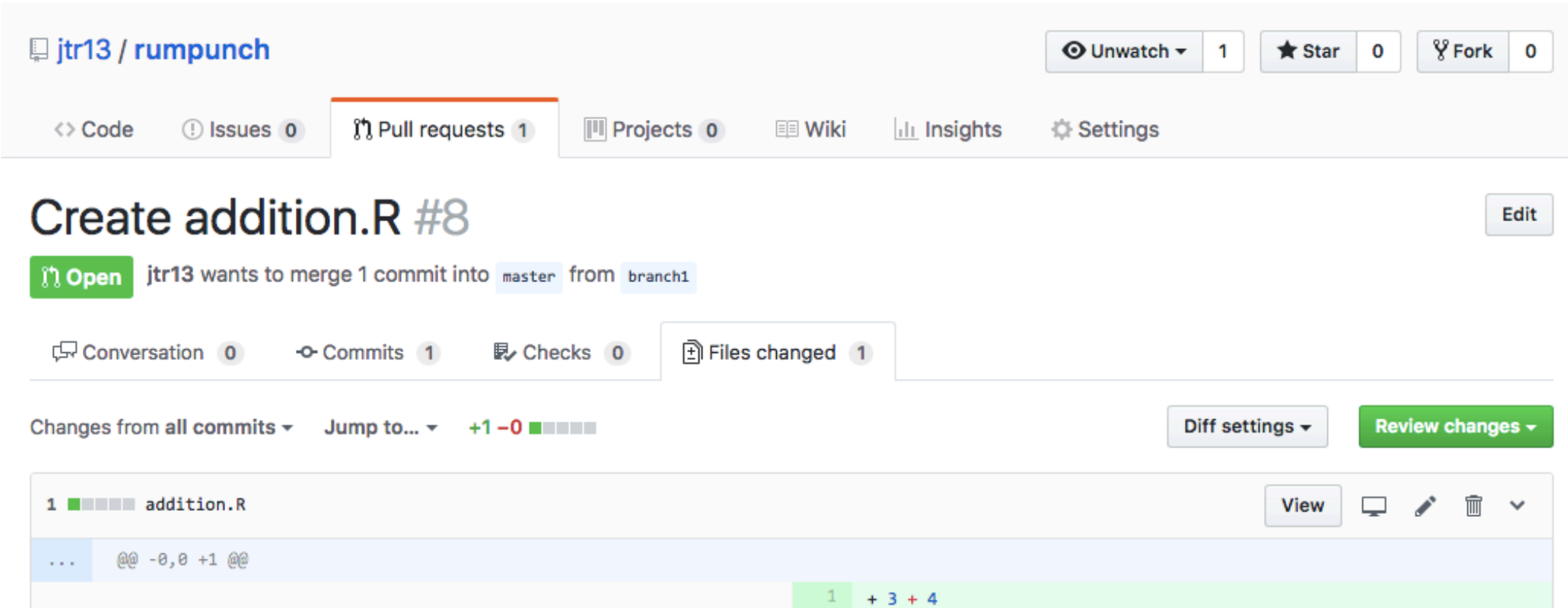
Step 6: Merging a pull request

- Click "Files changed"



Step 6: Merging a pull request

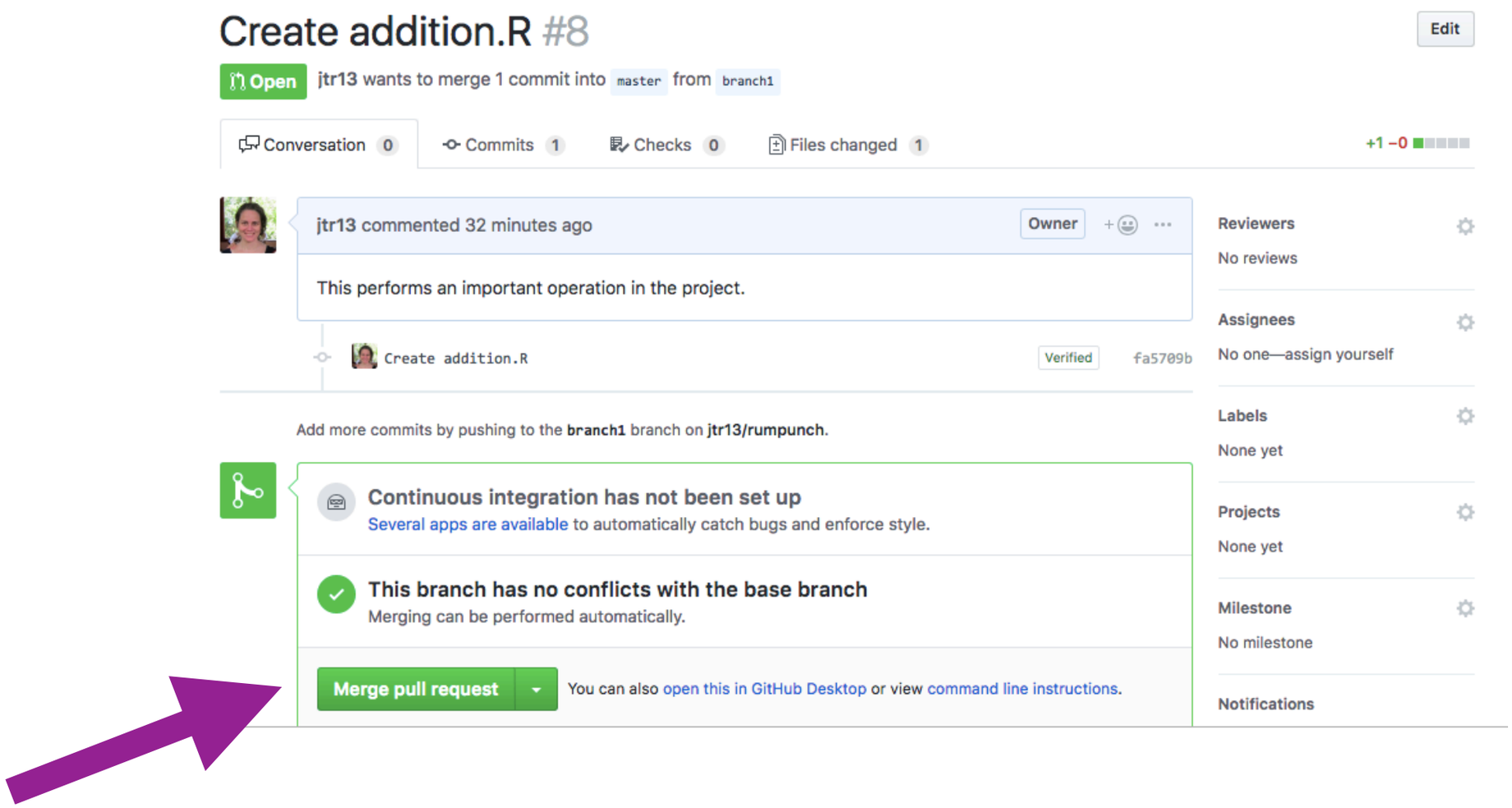
- Review the changes



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

Step 6: Merging a pull request

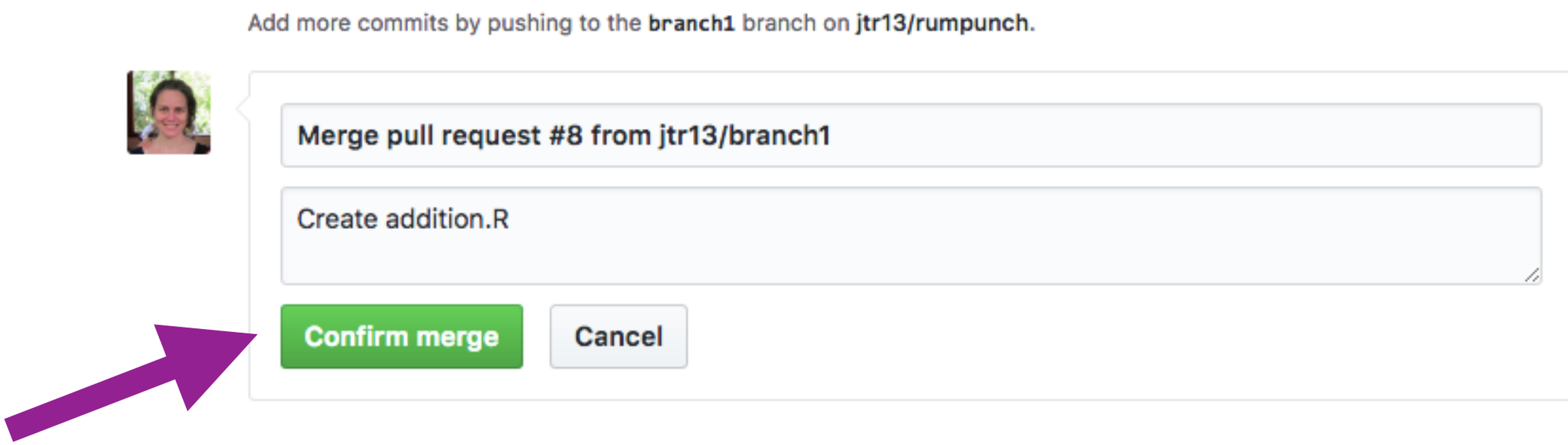
- Click back to return the pull request



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

Step 6: Merging a pull request

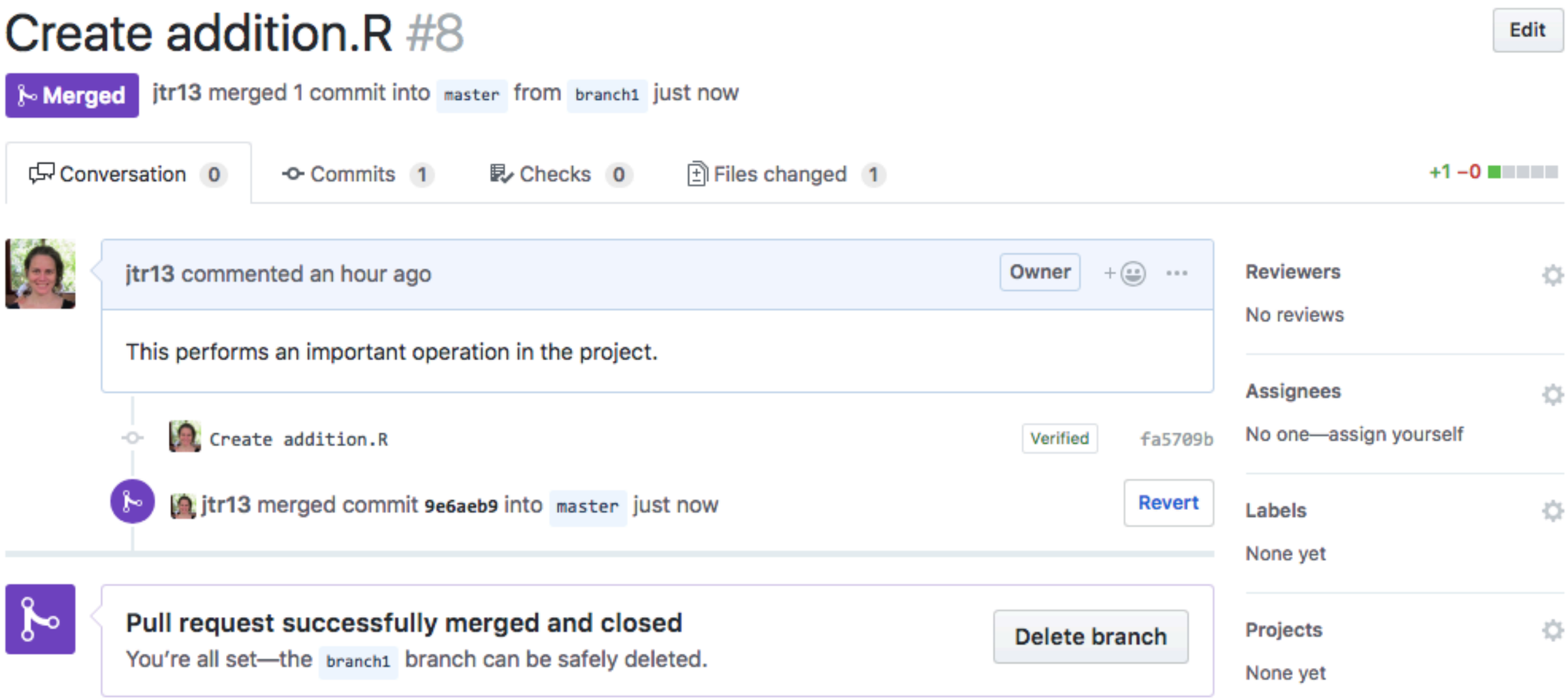
- Almost done...



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

Step 6: Merging a pull request

- Success!



Workflow 3

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

https://jtr13.github.io/EDAV/github.html#branching-someone-elses-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

Terminology

How New Yorkers define “upstate”

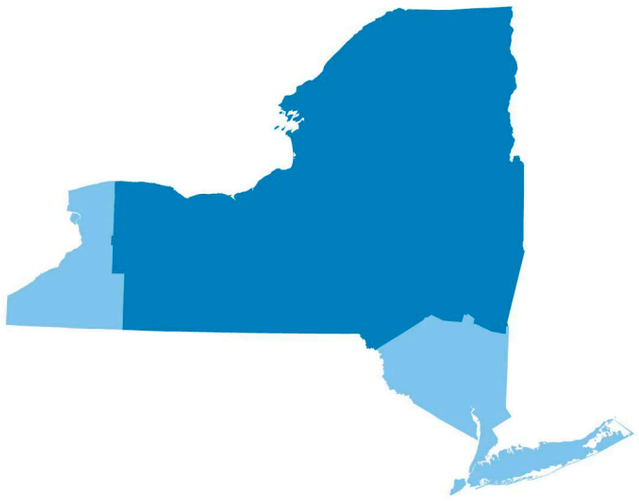
Per Public Policy Polling, April 2016.

North of New York City: **25%**

North of Westchester: **29%**

North of Poughkeepsie: **22%**

North of Poughkeepsie, but not Buffalo: **7%**



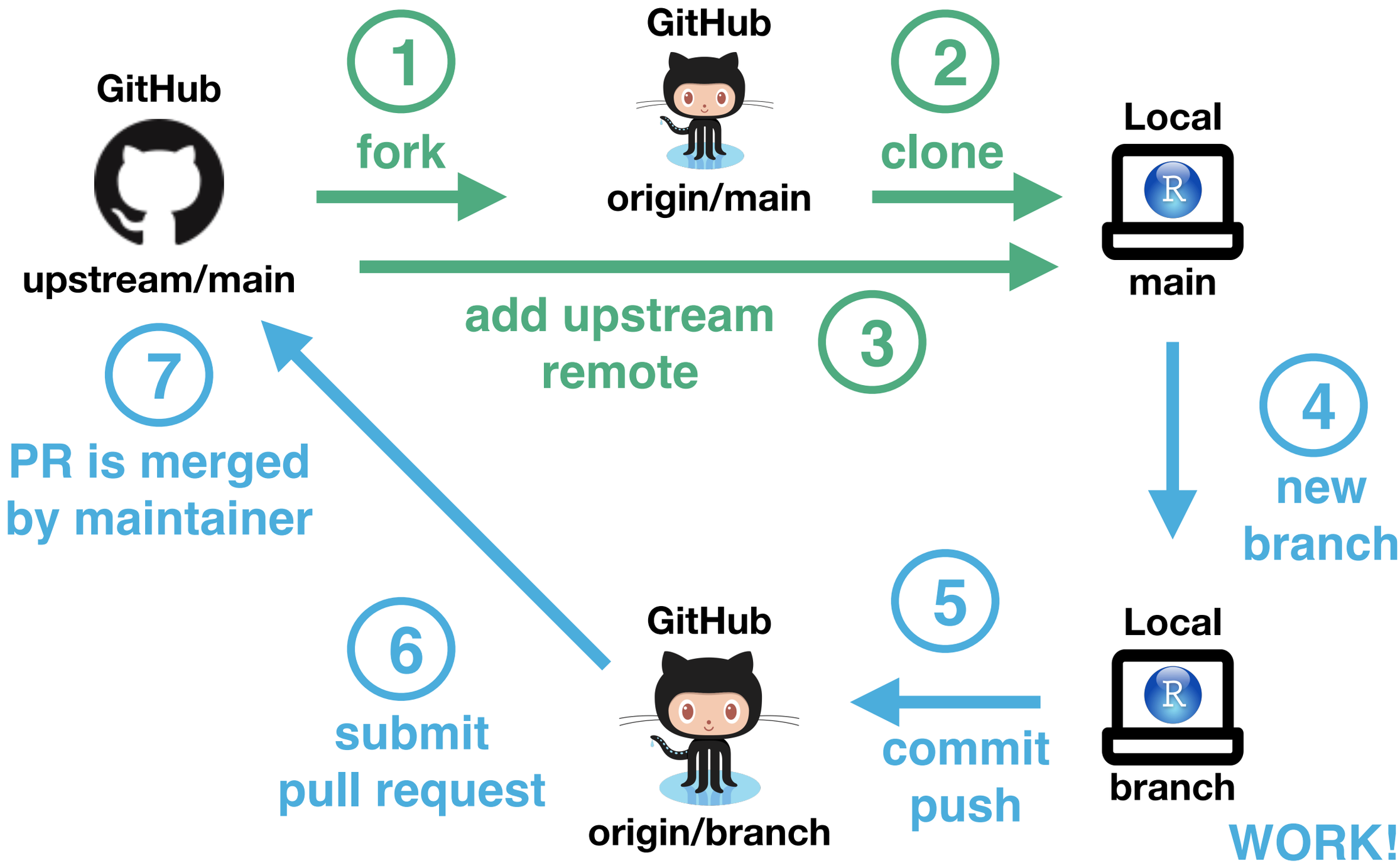
Source: “So What is ‘Upstate’ New York Exactly?” *The Washington Post*

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

1st pull request



The Workflow -- 1st PR

1. **new**: fork repo (once)
2. clone repo (once)
3. **new**: configure a remote that points to the upstream repository (once)

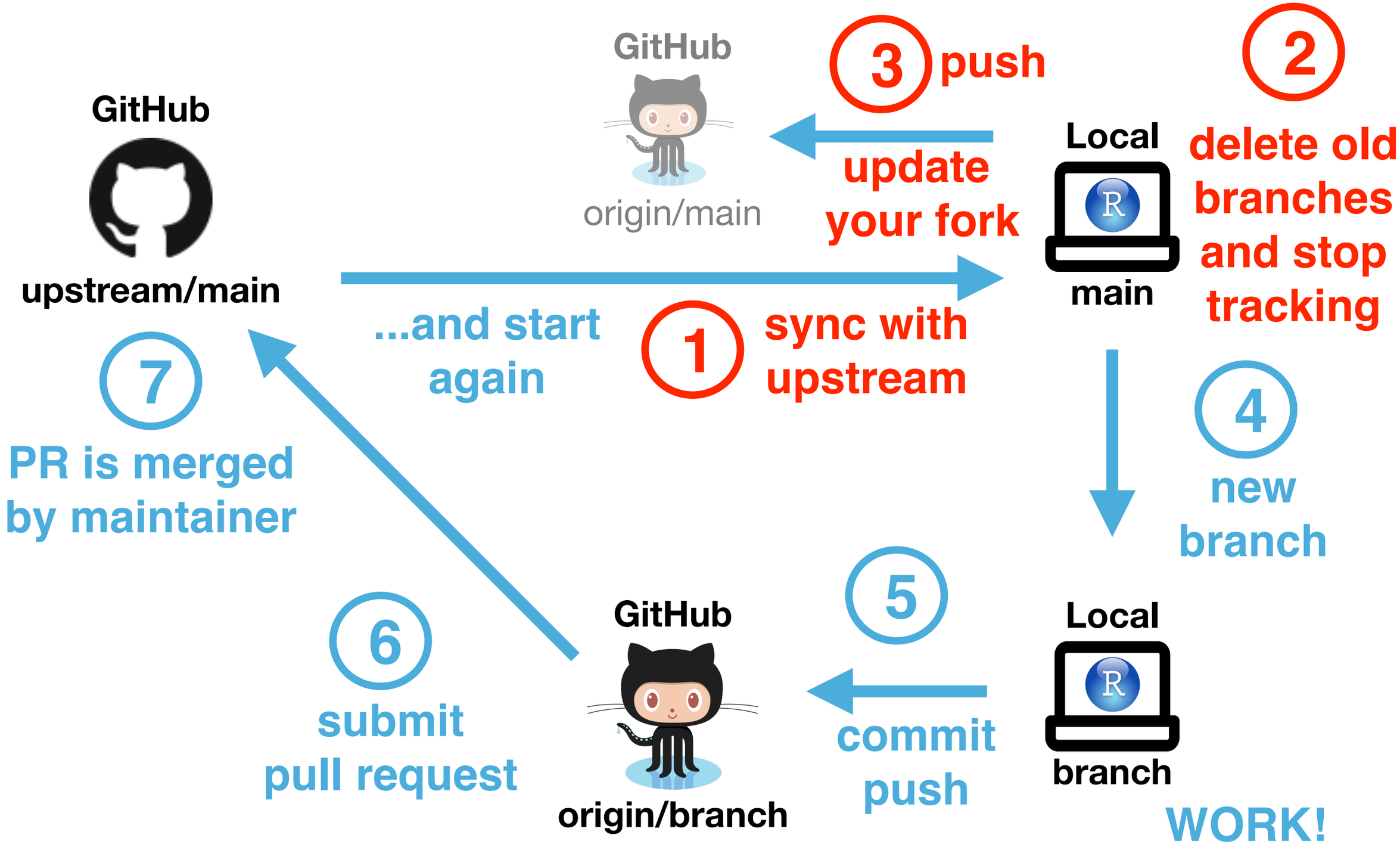
https://help.github.com/articles/configuring-a-remote-for-a-fork/

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

The Workflow -- 1st PR

4. branch
5. work, commit/push
6. submit pull request
7. wait for PR to be merged

2nd, 3rd, ... pull request



The Workflow -- 2nd, 3rd, ... PR

1. **new**: sync local main with upstream main:

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

> git fetch upstream

> git checkout main

> git merge upstream/main

The Workflow -- 2nd, 3rd, ... PR

2. Delete old branch(es) (see Part 3)

a) delete branch on GitHub

b) > git branch -d <branchname>

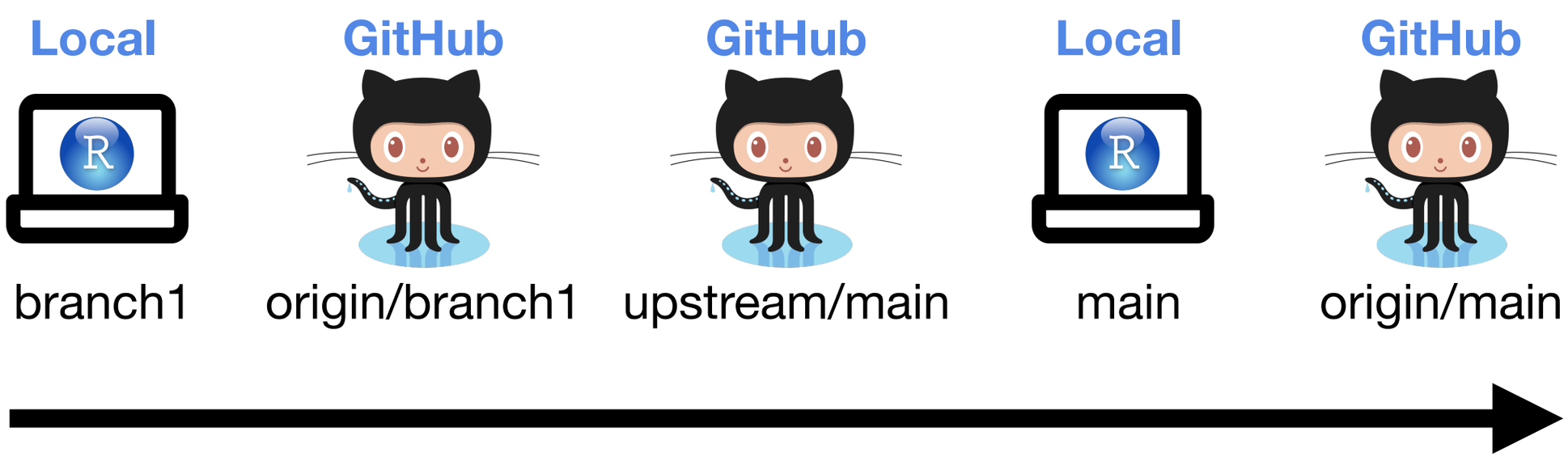
c) > git fetch -p

3. Push to update origin/main

4. (cont.)

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

Flow of new code:



Yes, it's not what you might expect!

The Workflows (in brief)

1. GitHub only: **work locally & upload; propose a file change to another repo**

2. GitHub + local main branch: **pull, work, commit/push**

3. GitHub + local main plus feature additional branches on your repo: **clone (once), pull, branch, work, commit/push, submit pull request, [merge pull request], delete branch on GitHub, delete locally**

The Workflows (in brief)

4. Contribute to someone else's repo:
   fork (once), clone (once), sync, branch, work, commit/push, submit pull request, [merge pull request], [delete branch on GitHub], delete local branch, push change to GitHub