

# **Git/GitHub Workflows**

## **(DRAFT)**

Joyce Robbins  
Dept. of Statistics  
Columbia University

This is a draft presentation for an R Ladies NYC workshop which will take place in January 2019.  
Feedback welcome by issue or email:  
[jtr13@columbia.edu](mailto:jtr13@columbia.edu)

# **Git/GitHub Workflows that will be covered here**

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

(To be added:

4. Contributing to someone else's repo)

# The Workflows

1. GitHub only: **work, upload**
2. GitHub + local master branch:  
**pull, work, commit/push**
3. GitHub + local master plus feature additional branches on your repo:  
**pull, branch, work, commit/push, submit pull request, [merge pull request], [delete branch on GitHub], delete local branch**

# Git/GitHub Workflows

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

# 1. GitHub only



# GitHub only



- It's very simple.
- You just create an account on GitHub.
- If you want to share files, create a repository and give it a name.
- You can then upload whatever you'd like to the repository.

# GitHub only

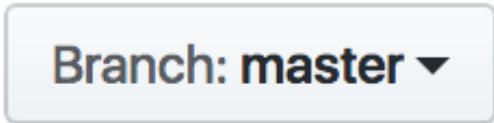


- It's an easy way to share files.
- Other people can copy (fork) the repository, submit pull requests, and/or create issues.
- If you want them to be able to read material on GitHub without downloading, write in **markdown** or share **pdfs**.

# GitHub only



## Notes:

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



# GitHub only



Examples:

[https://github.com/jtr13/codehelp/  
blob/master/R/reorder.md](https://github.com/jtr13/codehelp/blob/master/R/reorder.md)

[https://github.com/jtr13/codehelp/  
blob/master/GitHubWorkflow.pdf](https://github.com/jtr13/codehelp/blob/master/GitHubWorkflow.pdf)

# Git/GitHub Workflows

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

## 2. Create a local clone of our GitHub repository

Why?

- It's hard to write code on GitHub since you can't run it.
- The GitHub version serves as a backup--with code that works--while I experiment locally.

# The Setup

- There are a few things you need to do to get setup, including downloading Git. A great resource is:

<http://happygitwithr.com>

Part I Installation

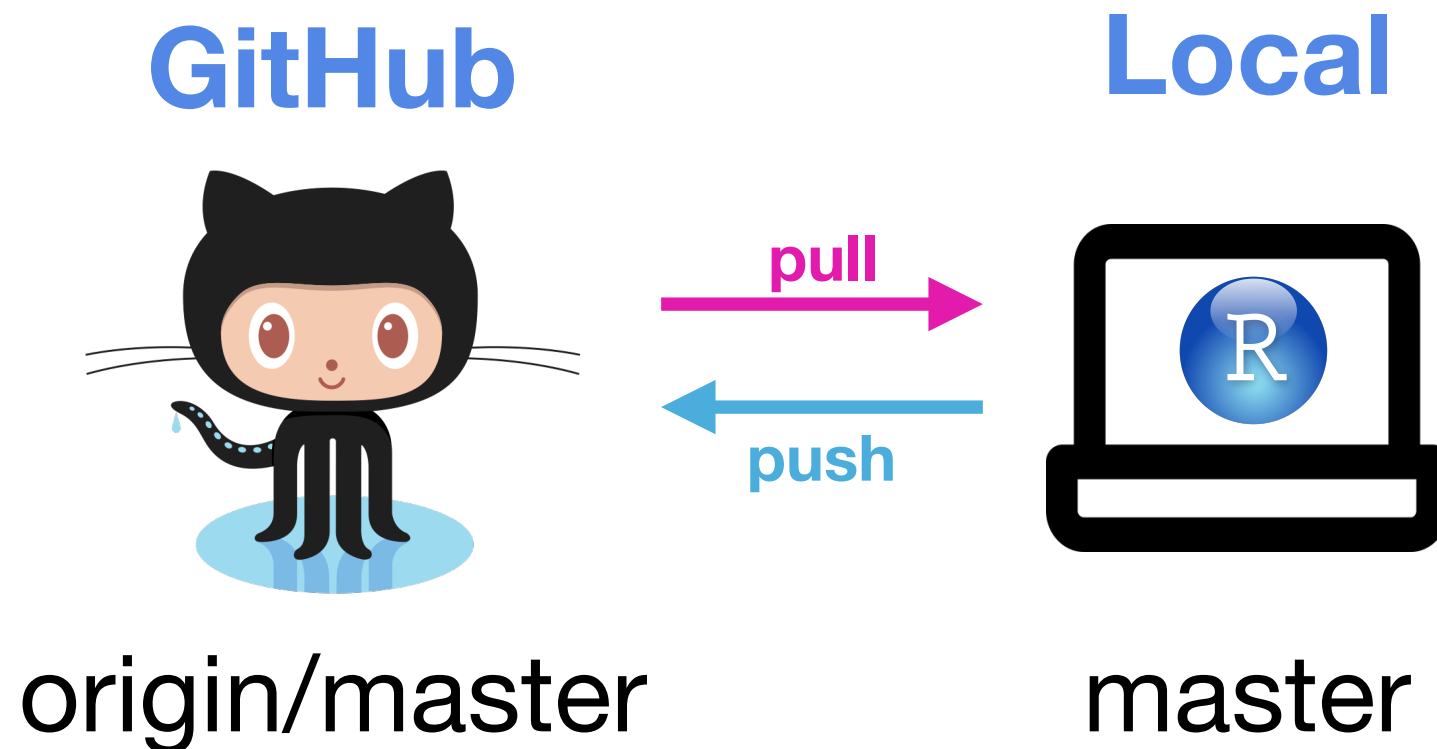
Part II Connect Git, GitHub, RStudio

by Jenny Bryan, the STAT 545 TAs,  
Jim Hester

# The Setup




- Do not be intimidated by the number of chapters in these two parts. Why?
- Some of it you've already done.
- Some of the chapters are very short.
- A lot of the material deals with Other Operating Systems.
- A lot of the material is designed to help you troubleshoot and may not apply.

# Our new model

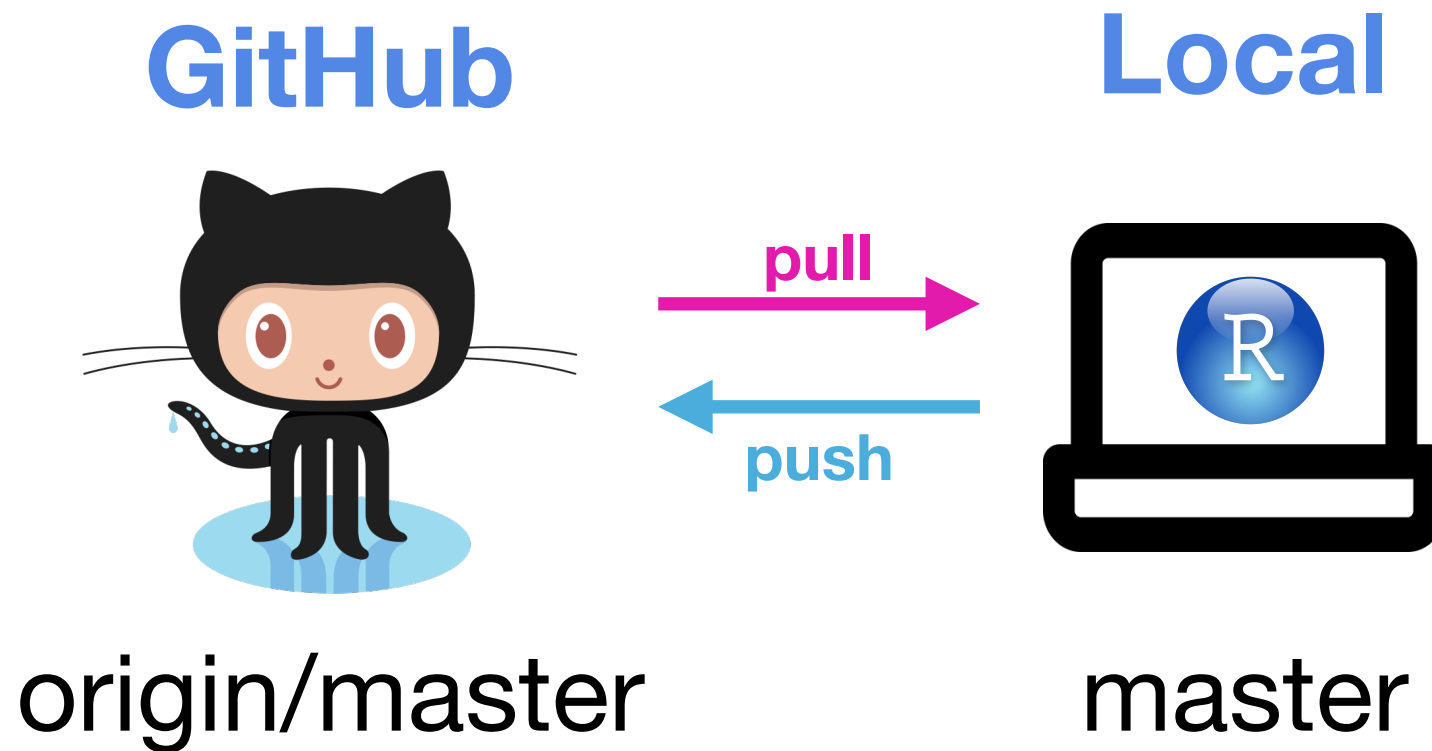


- This workflow is described in more detail in *Happy Git with R*, Chapter 16 "New project, GitHub first"

# 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.

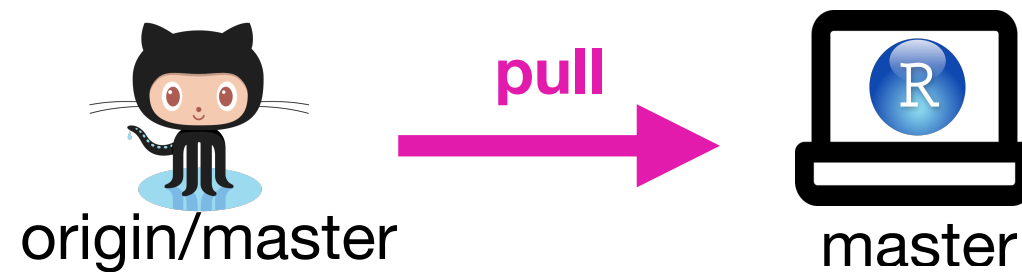
# Now we're ready to start.



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

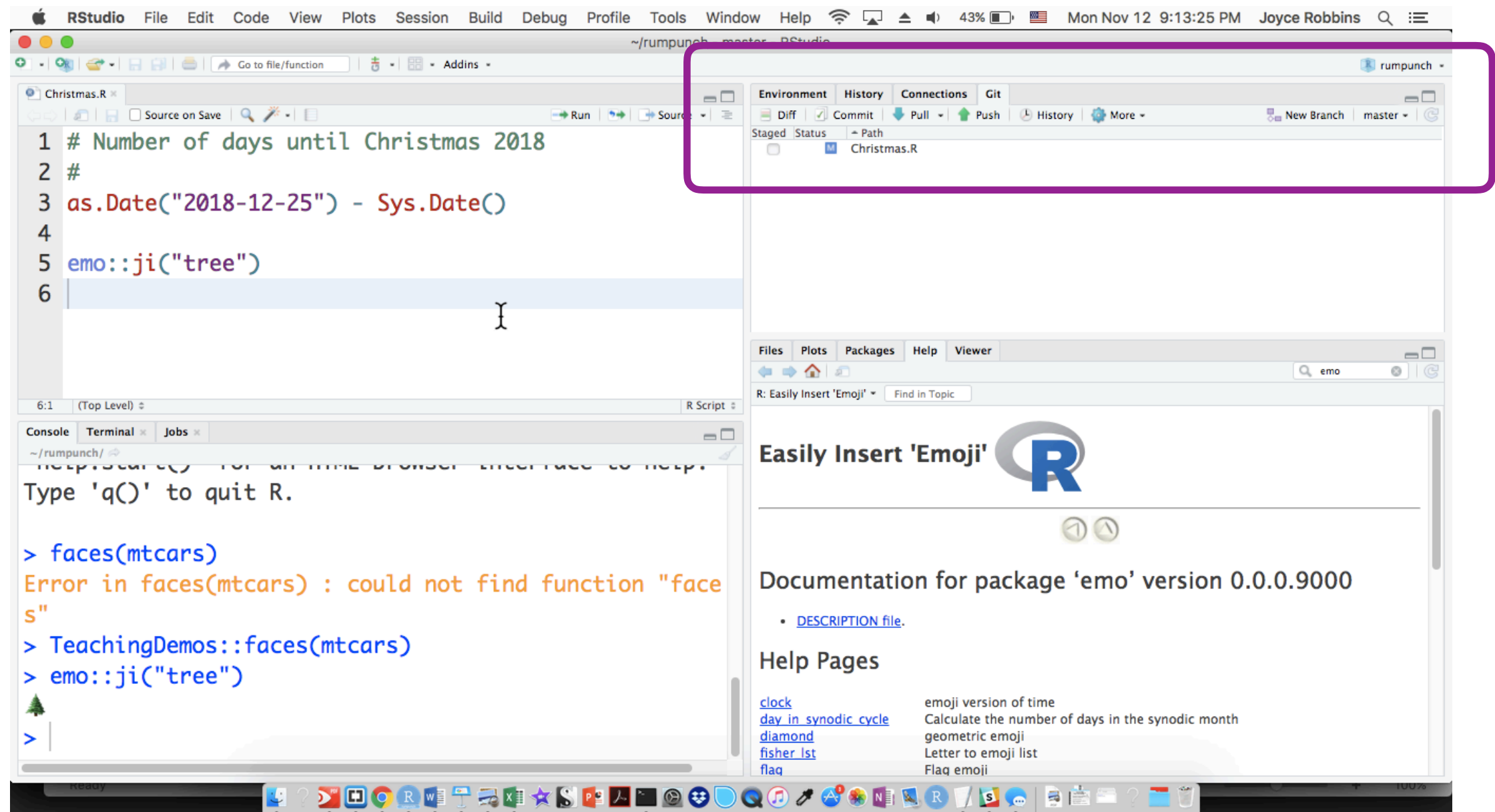


- After clicking the pull button (down arrow) in the Git pane, we see:

```
Git Pull Close  
>>> git pull  
Already up-to-date.
```

# Step 2. Work

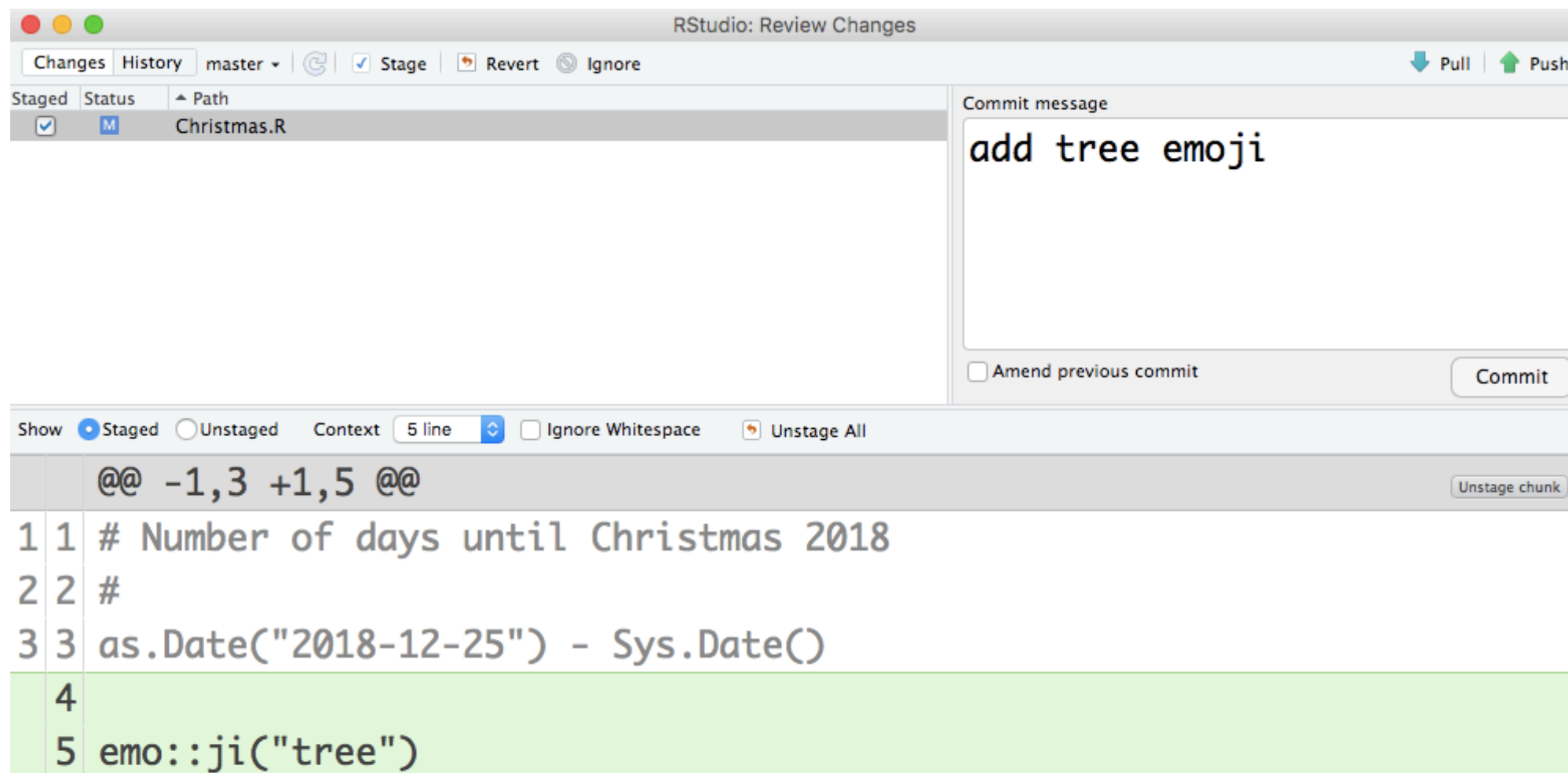
(demo in RStudio)



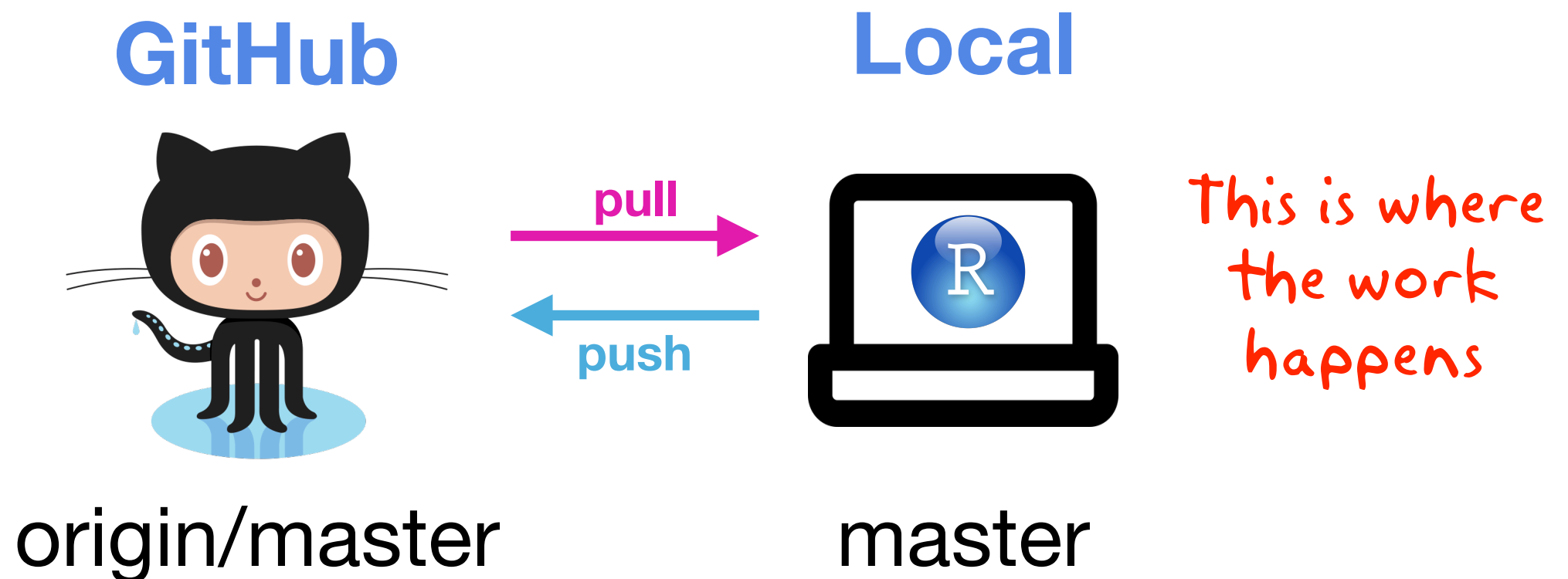
# Step 3. Commit/Push



(demo in RStudio / GitHub)



# Our new model (summary)



# Git/GitHub Workflows

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

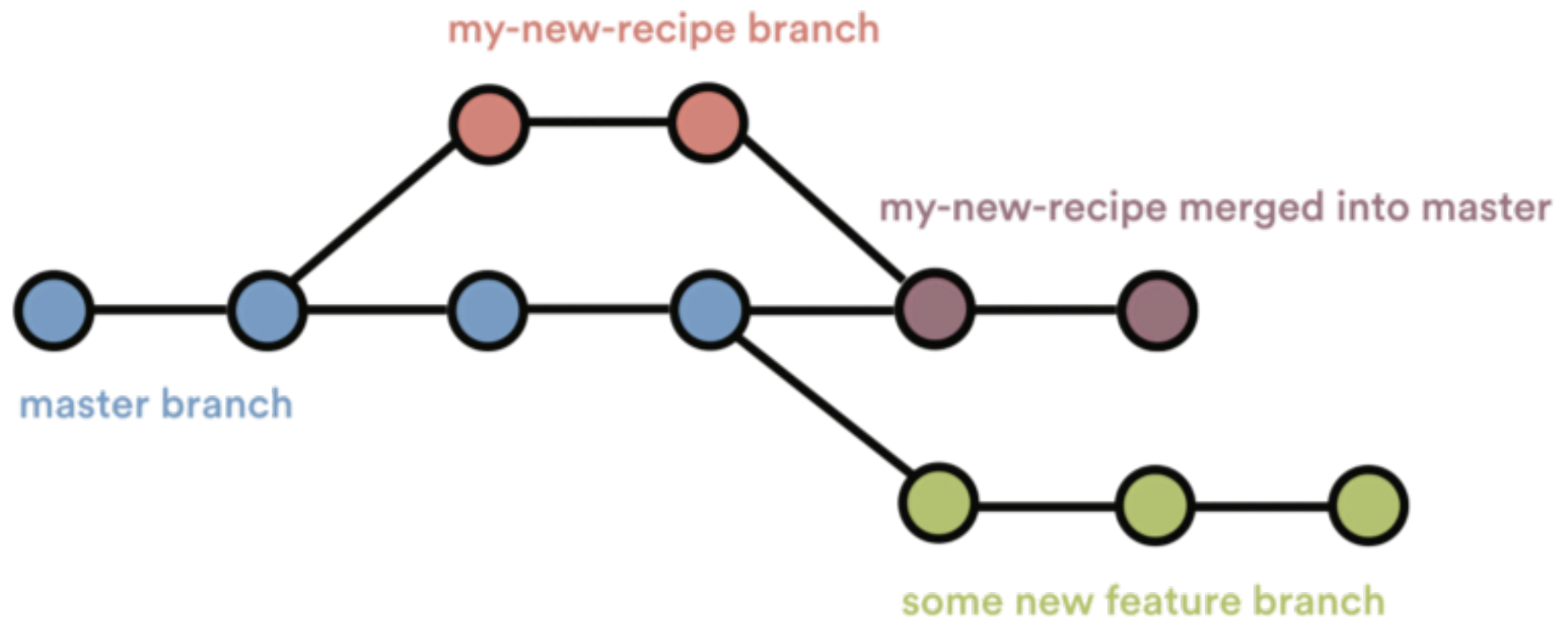
# 3. Create a new branch

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:





# Your perspective

Start by creating or forking repo on GitHub,  
then cloning it (once)



origin/master

# Your perspective

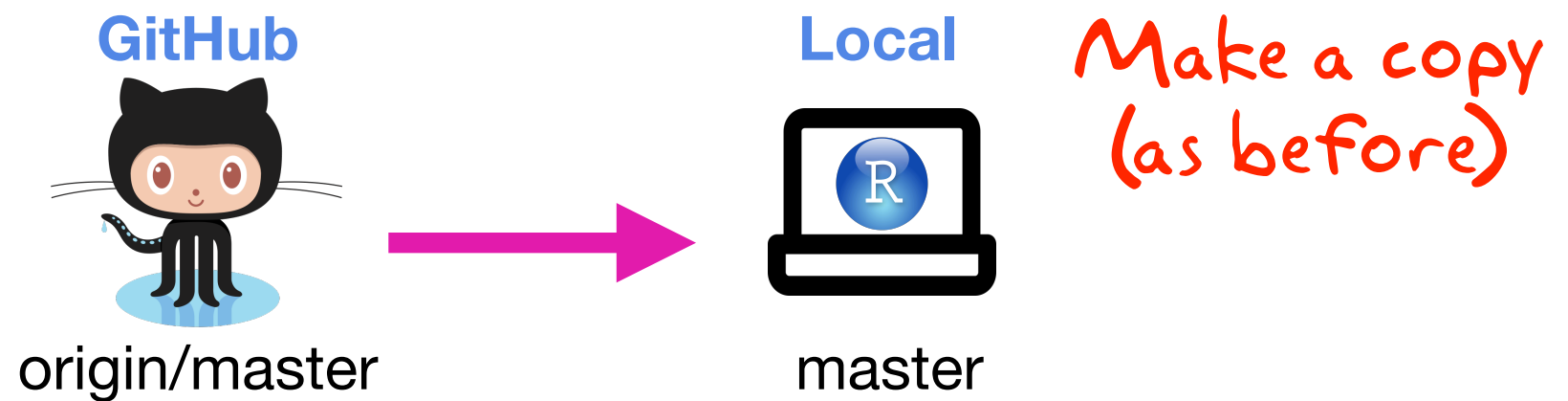
Working, shared  
code lives here

GitHub

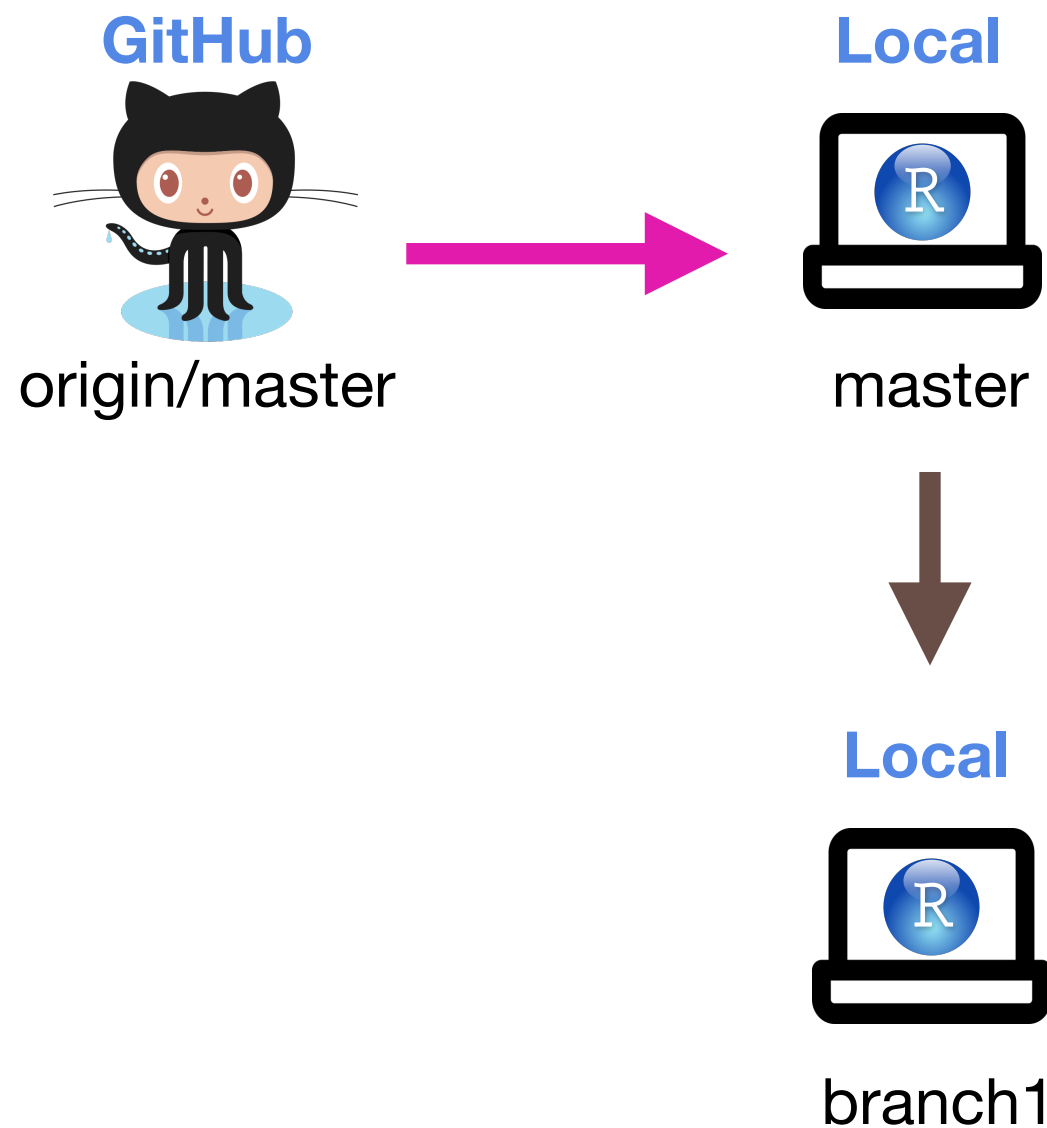


origin/master

# Your perspective

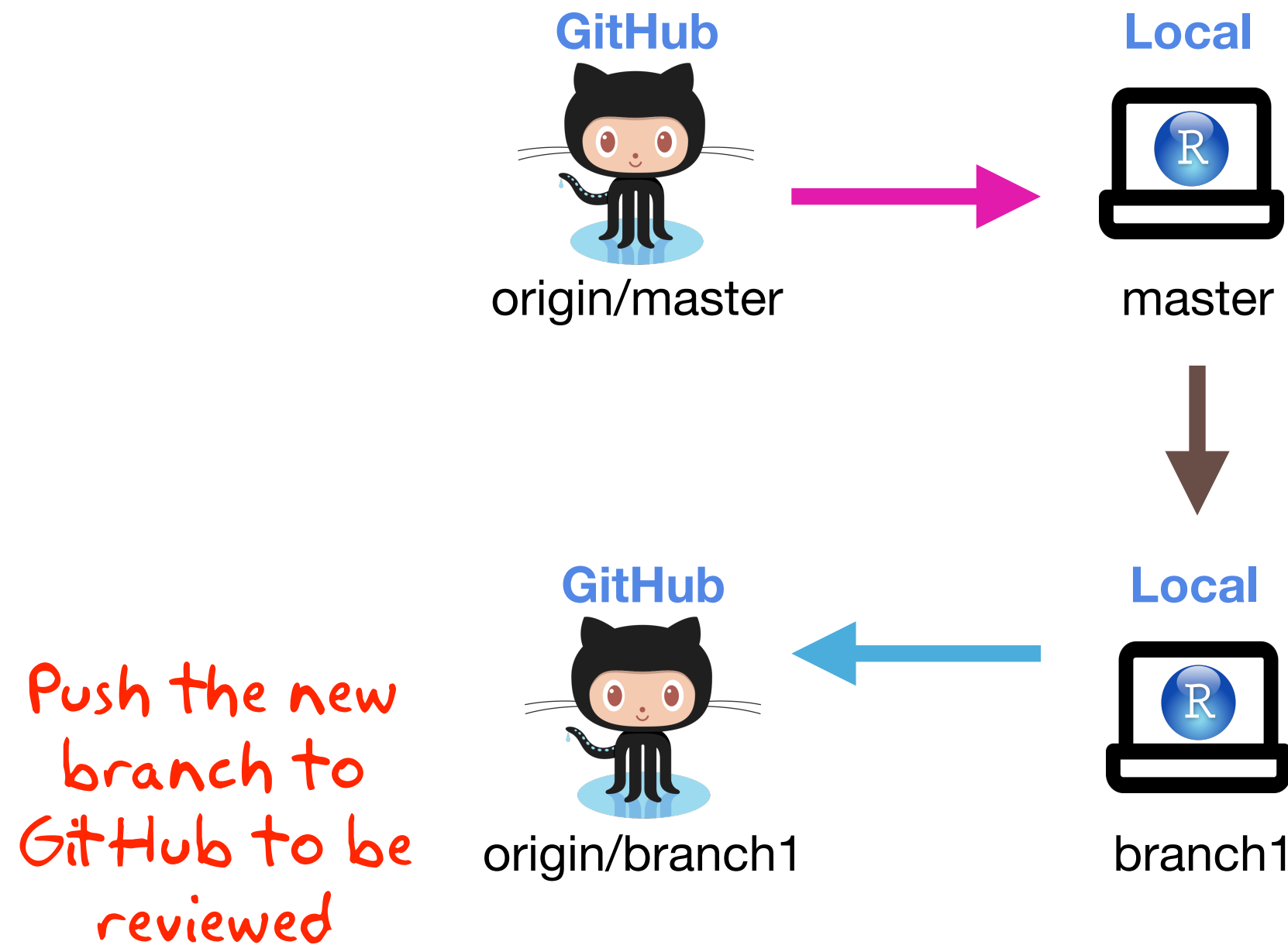


# Your perspective

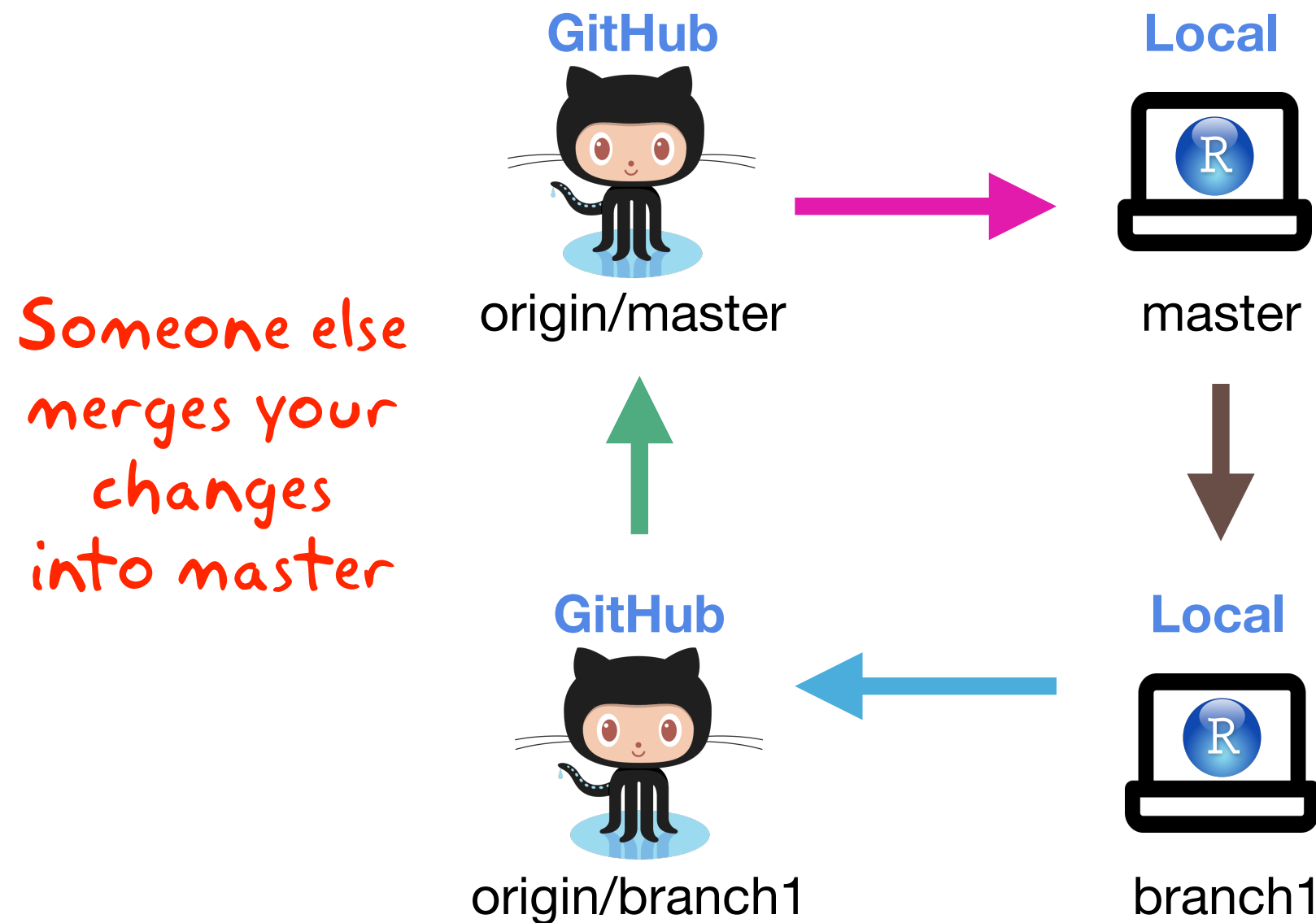


Create a branch  
to do your  
new work

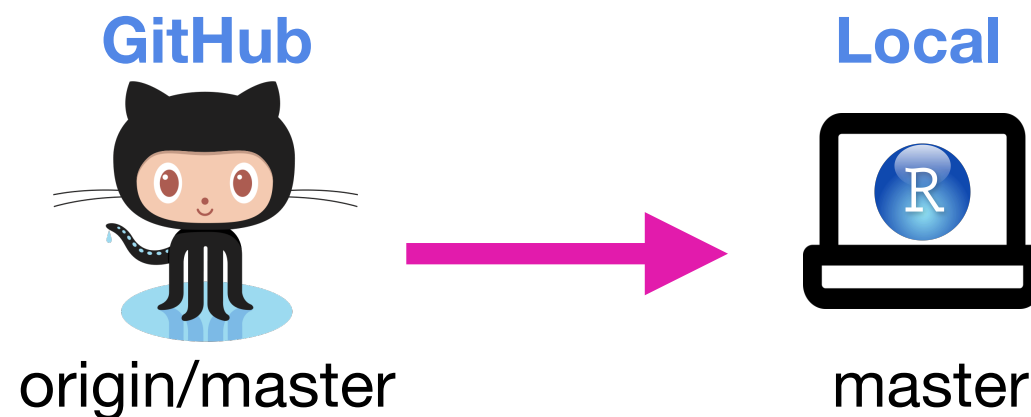
# Your perspective



# Your perspective

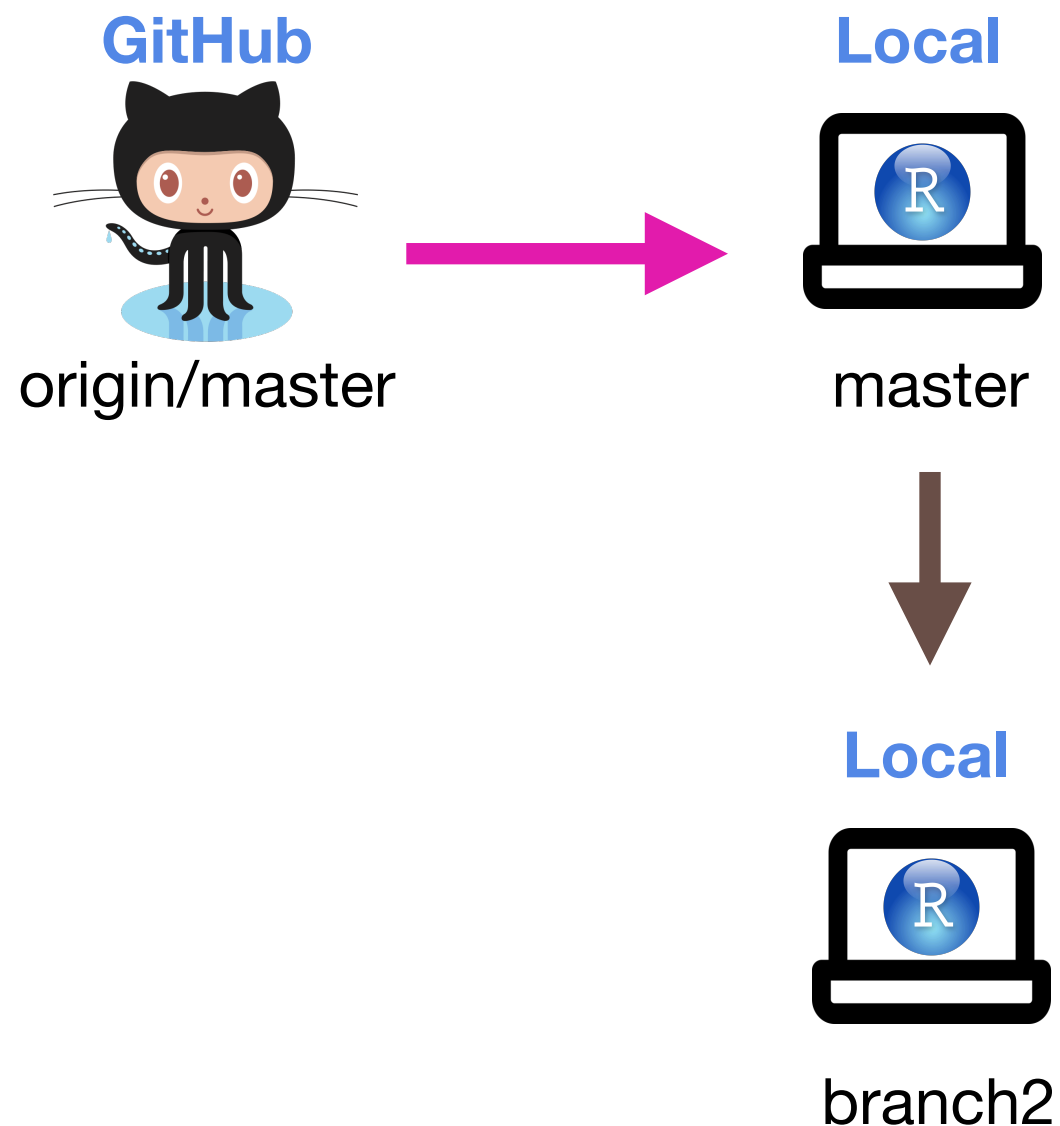


# Your perspective



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

# Your perspective

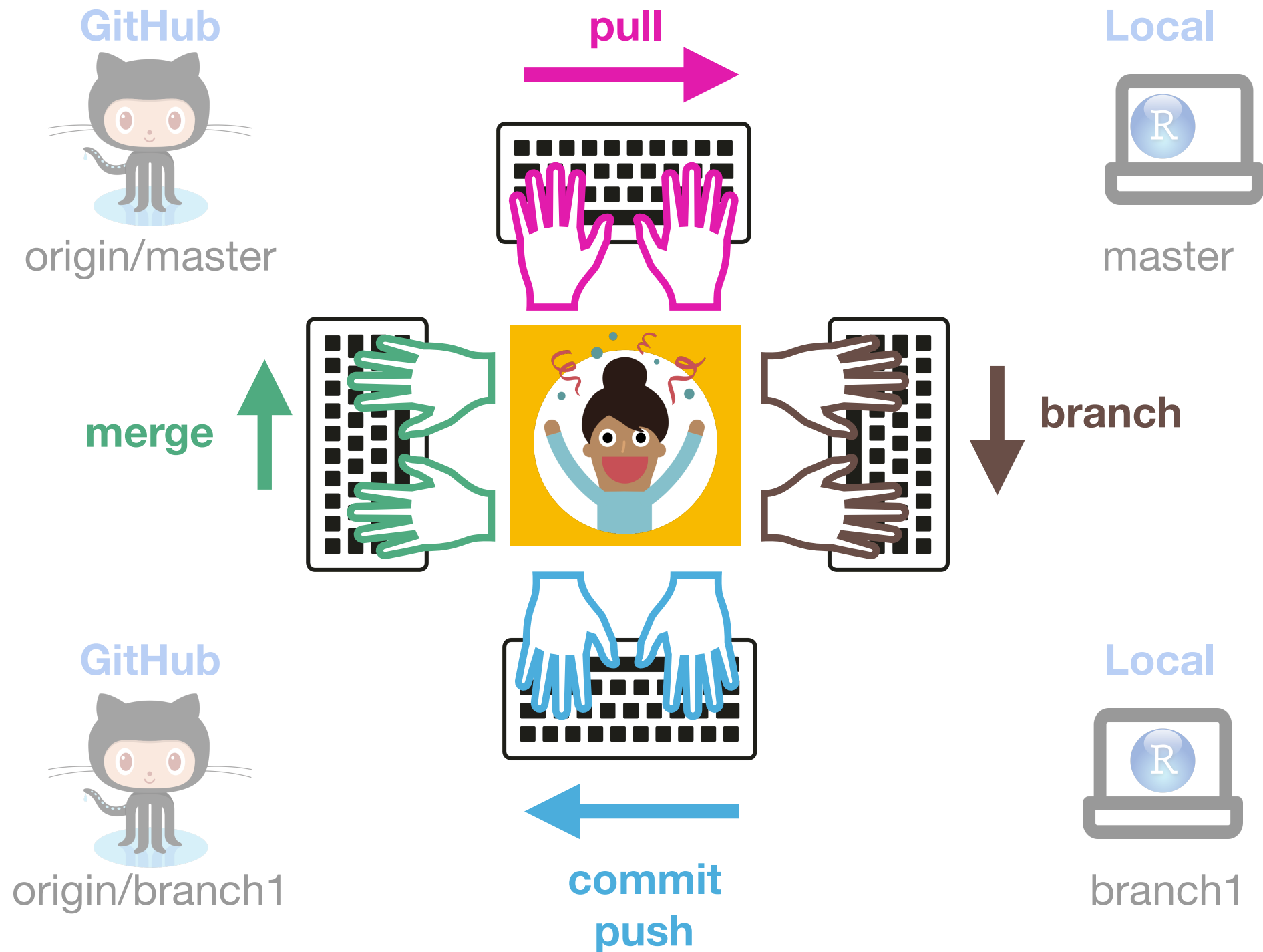


Create a branch  
to do your  
new work

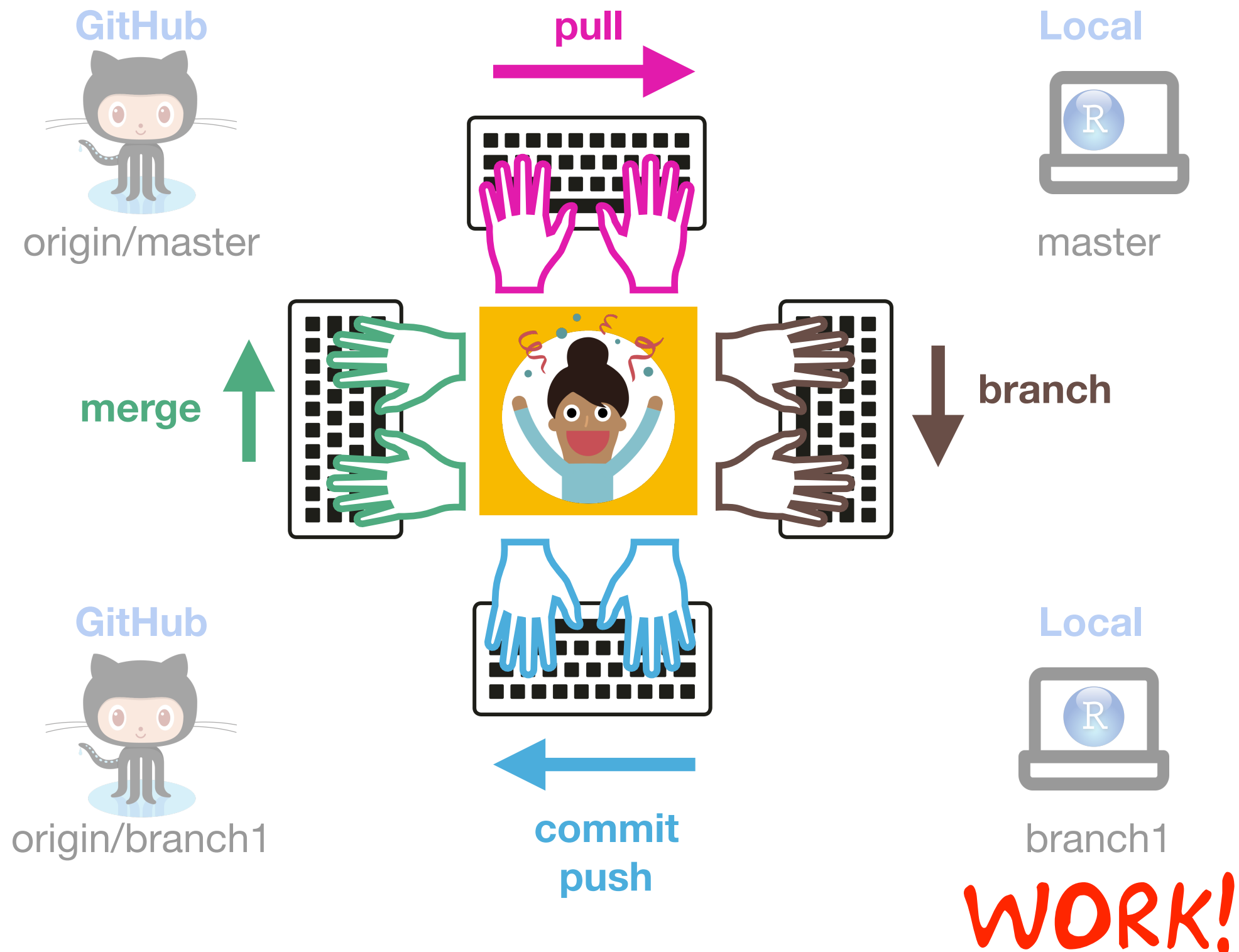
And so on and so on...



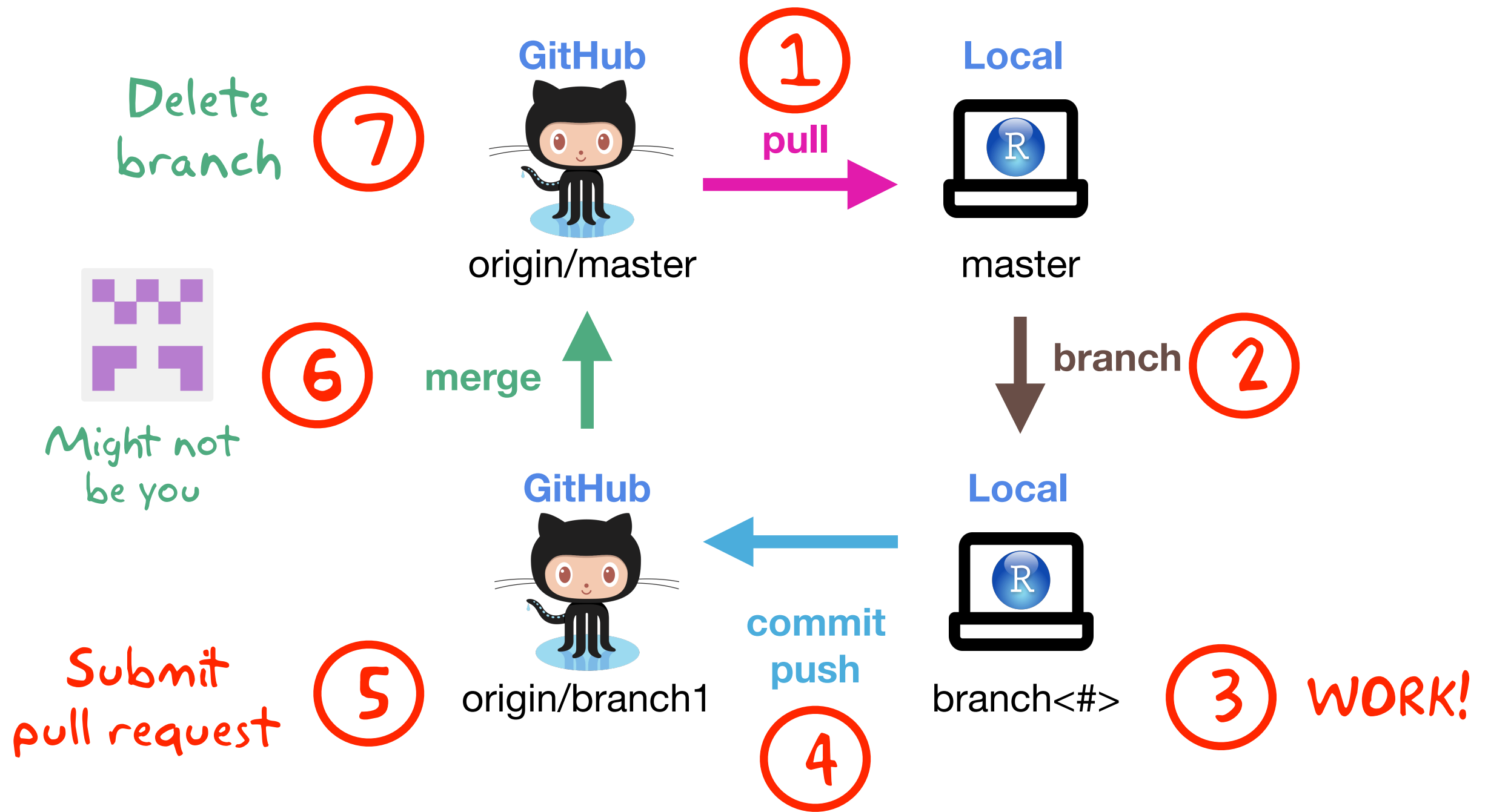
# Your perspective



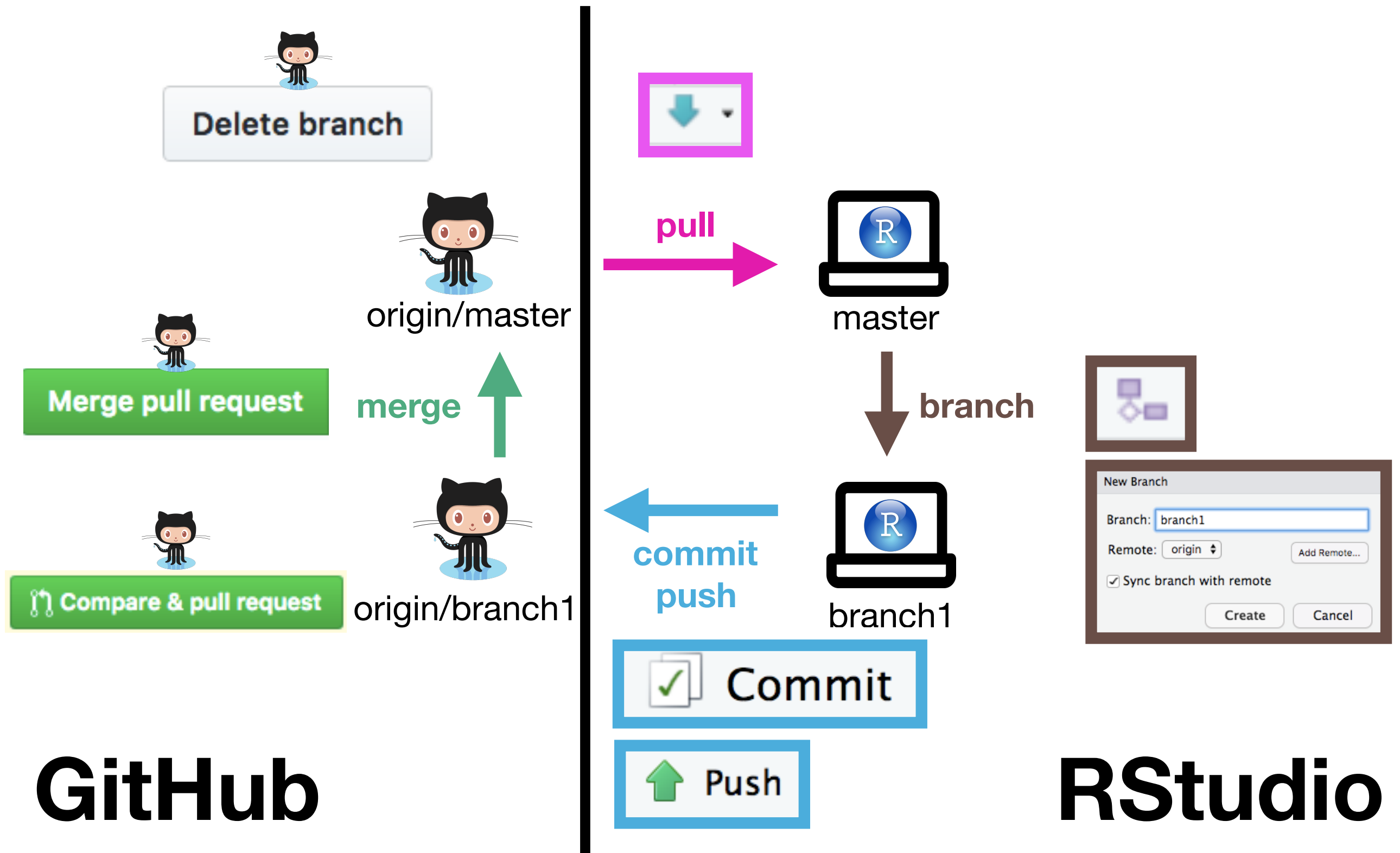
# Your perspective



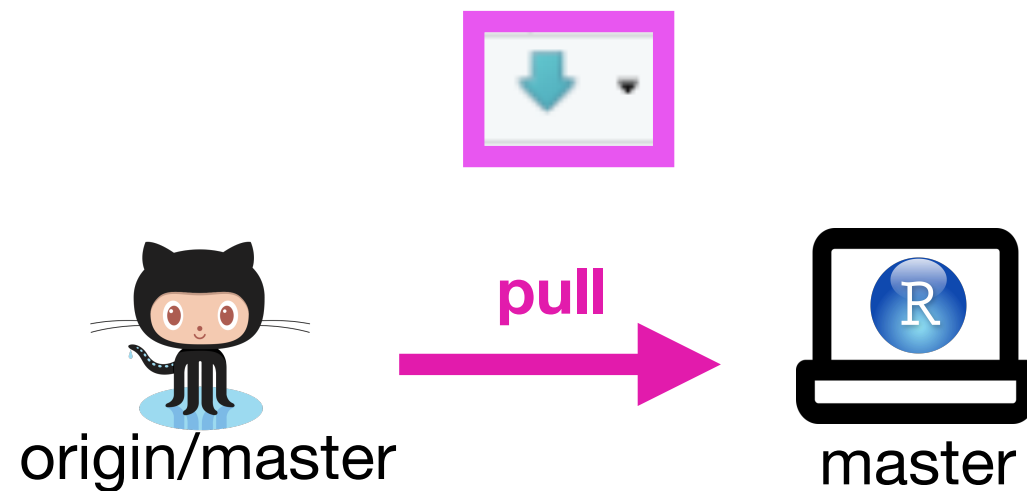
# Your workflow



# What's happening where

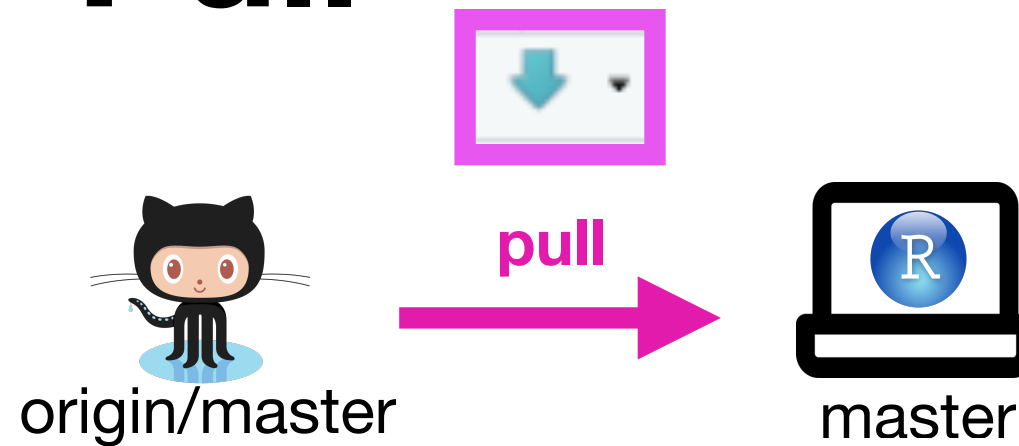


# Step 1. Pull



- 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

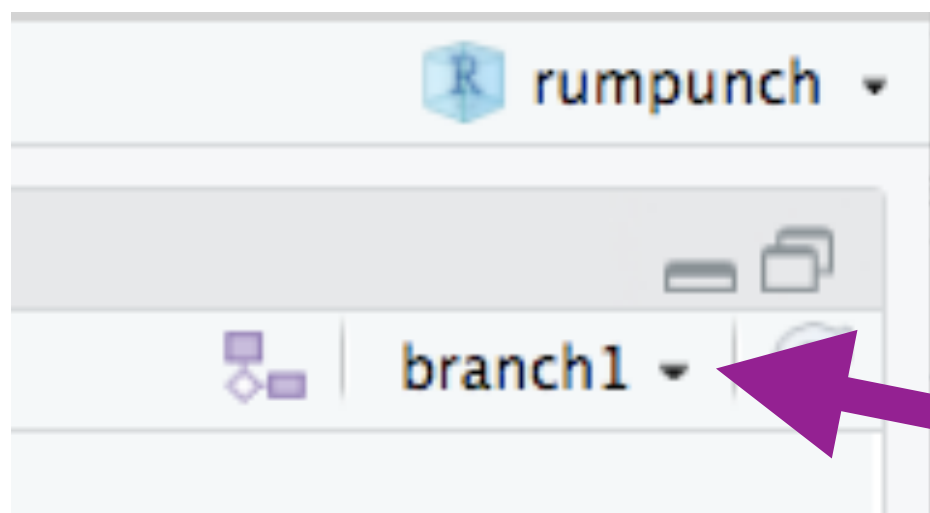
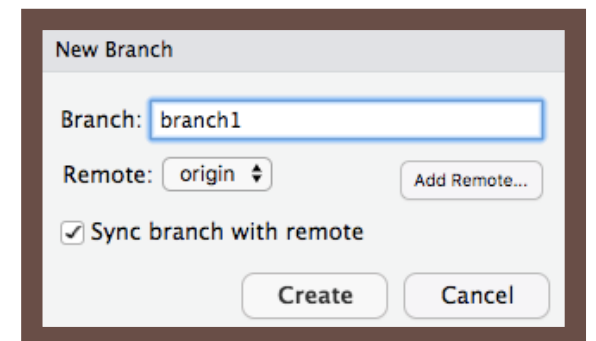


- If all goes well (no conflicts), our copy of master will be updated:

```
>>> 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(+)
```

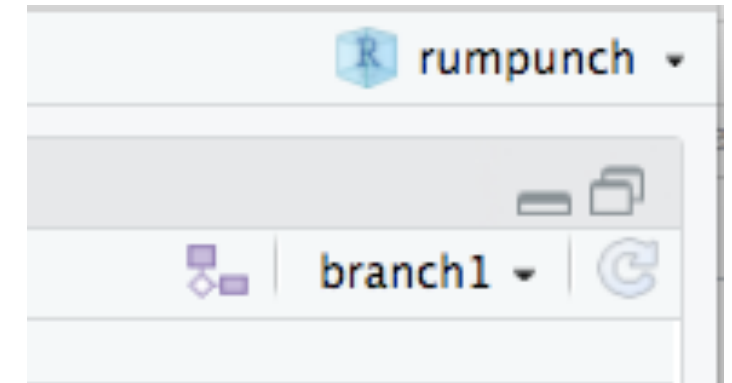
# Step 2: Create a new branch

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



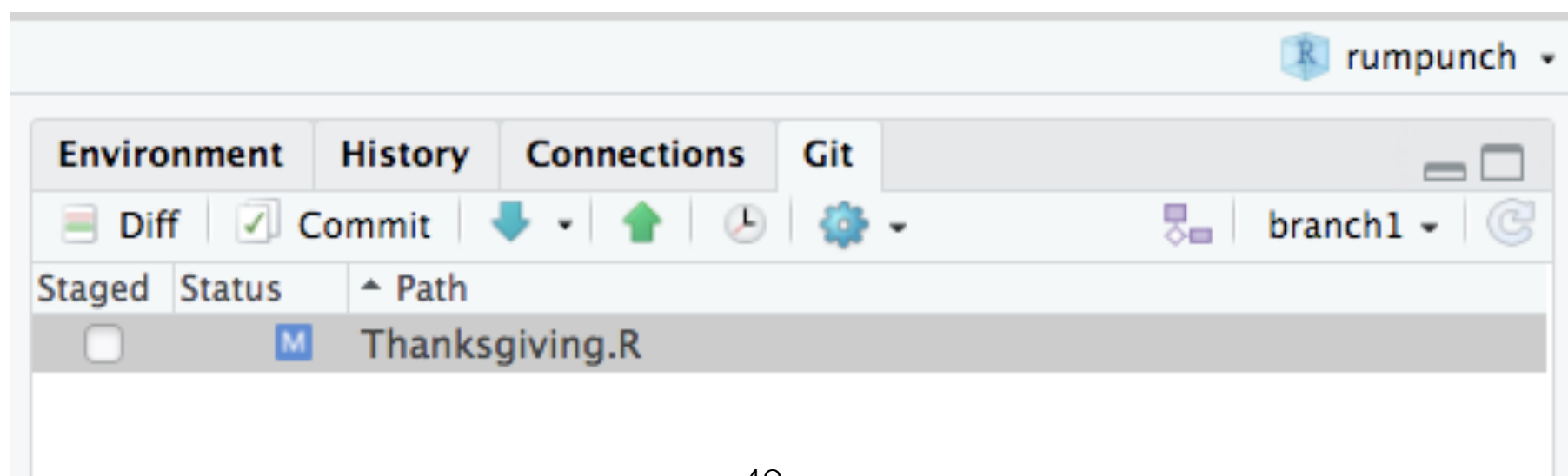
# Step 3: Work

```
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 }
```



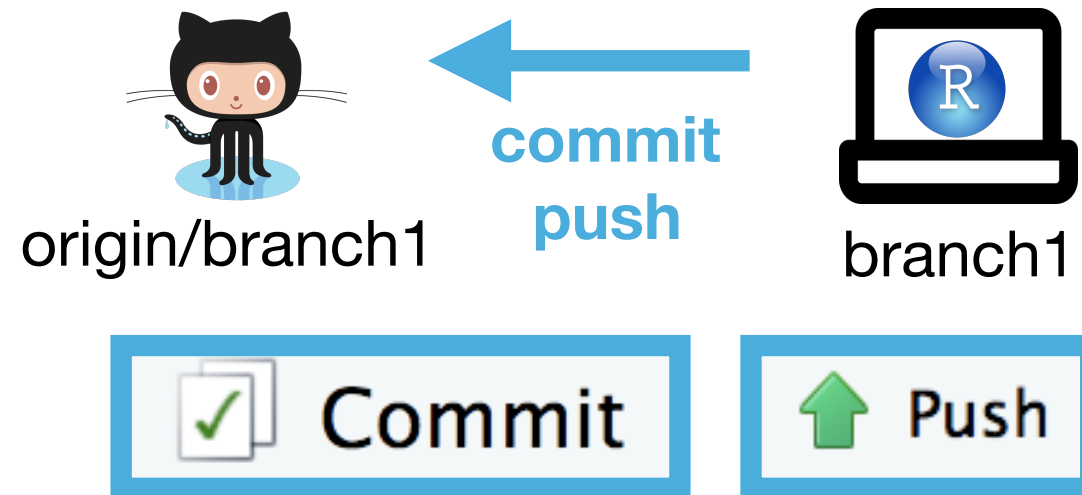
branch1

- 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

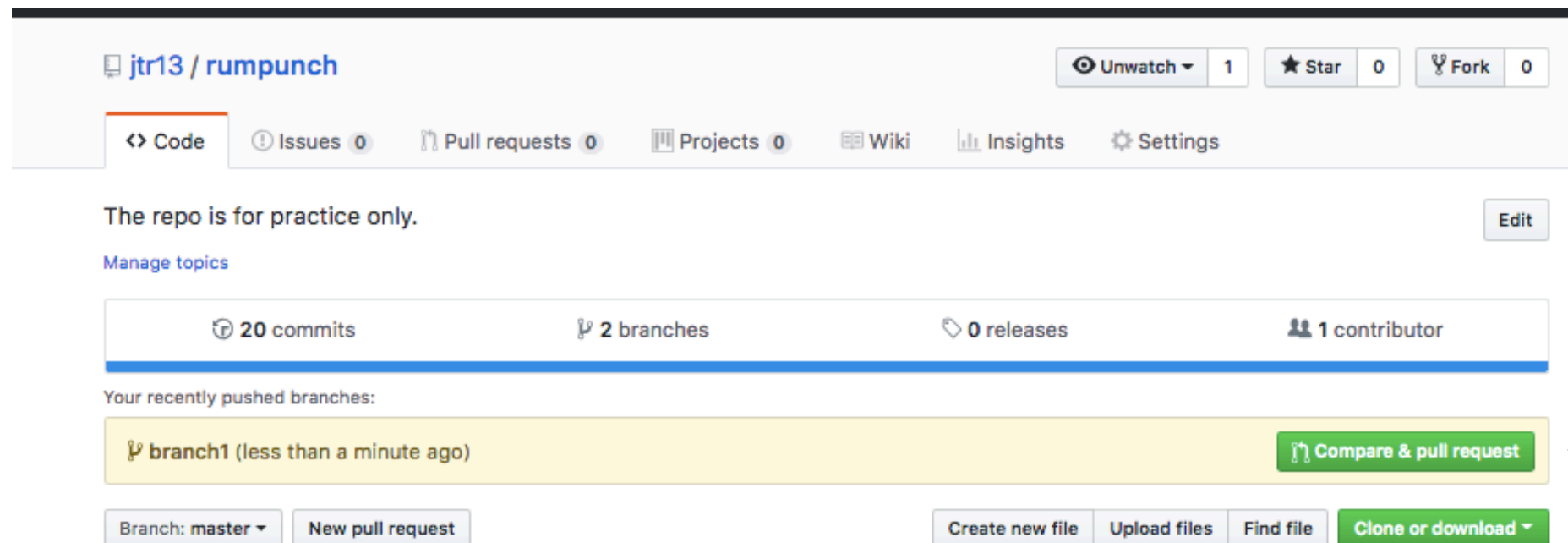


 **Compare & pull request**




origin/branch1

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

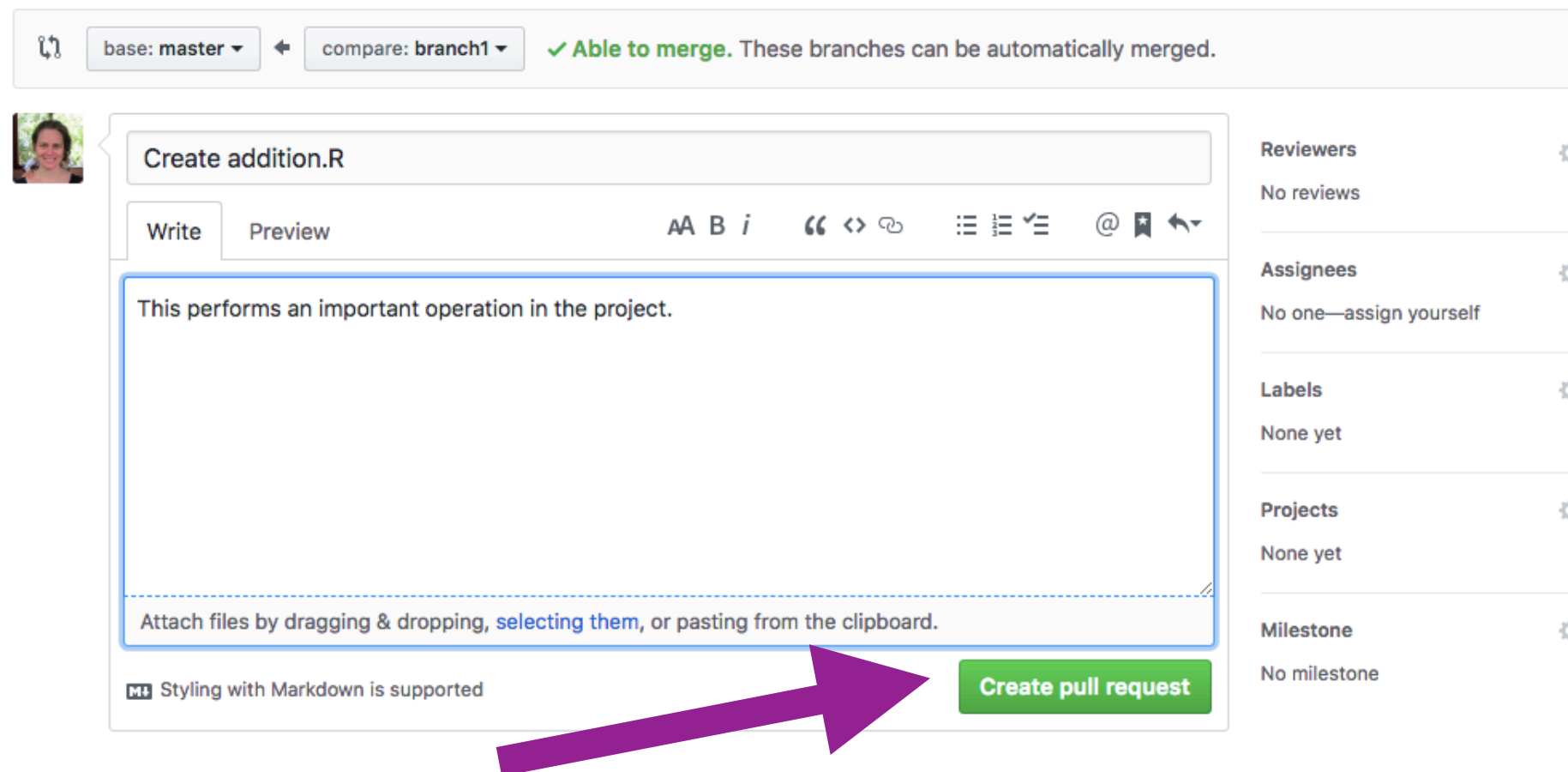


# Step 5: Submit a pull request

- Click: 
- Add a description

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



base: master ← compare: branch1 ✓ Able to merge. These branches can be automatically merged.

Create addition.R

Write Preview

AA B i “ <> @ \* ↶

This performs an important operation in the project.

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

M Styling with Markdown is supported

Reviewers  
No reviews

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Create pull request

- Then click "Create pull request"

# Step 6: Merging a pull request

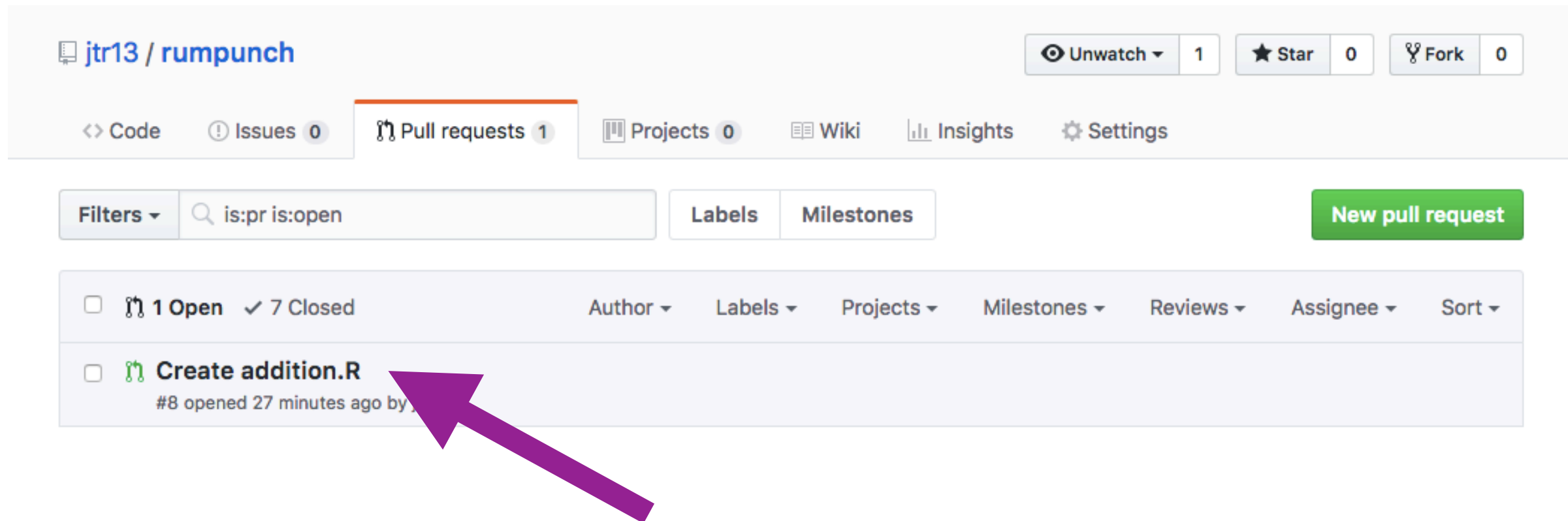
- There are a lot of opinions on who should merge the pull request: the original author (you) or someone else
- What's most important is that you communicate with your collaborators and decide how you're going to manage the pull requests.
- Practice both merging your own pull requests and letting someone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

Create addition.R #8 Edit

Open jtr13 wants to merge 1 commit into master from branch1

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -0

jtr13 commented 32 minutes ago Owner + 😊 ...

This performs an important operation in the project.

Create addition.R Verified fa5709b

Add more commits by pushing to the **branch1** branch on **jtr13/rumpunch**.

**Continuous integration has not been set up**  
Several apps are available to automatically catch bugs and enforce style.

**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

**Assignees** ⚙️  
No one—assign yourself

**Labels** ⚙️  
None yet

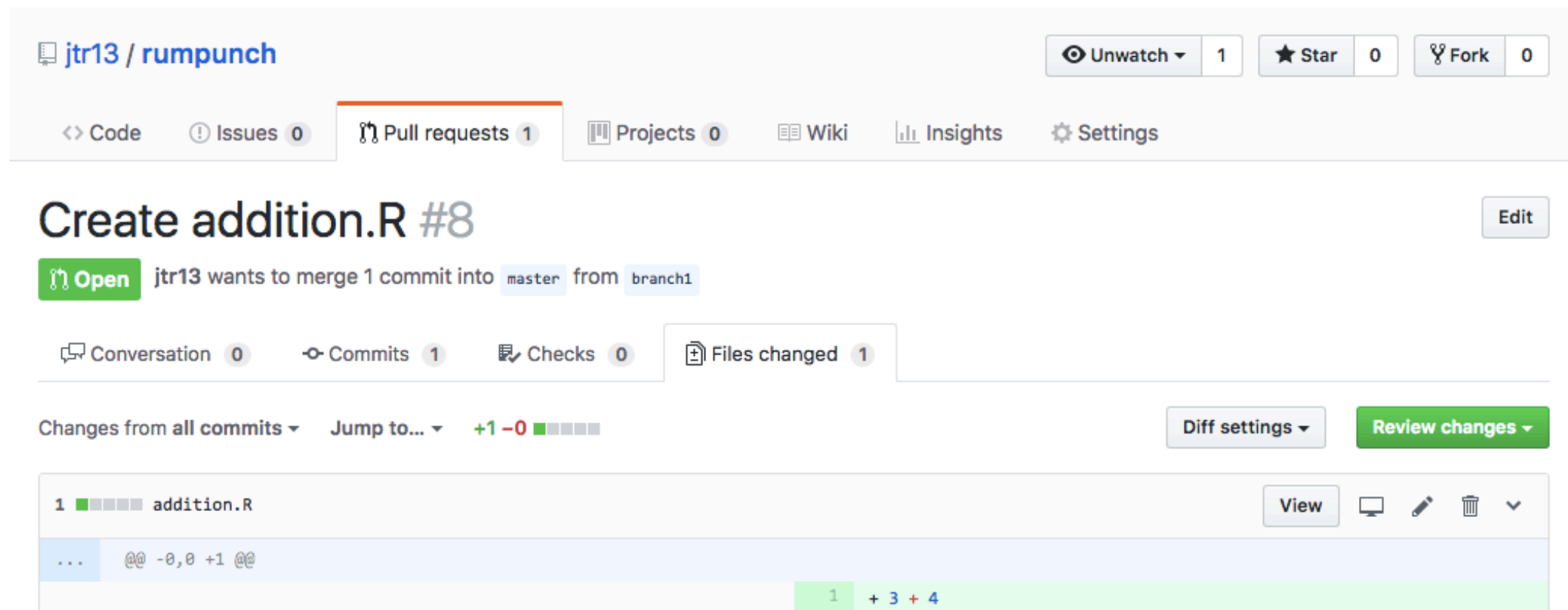
**Projects** ⚙️  
None yet

**Milestone** ⚙️  
No milestone

**Notifications**

# Step 6: Merging a pull request

- Review the changes

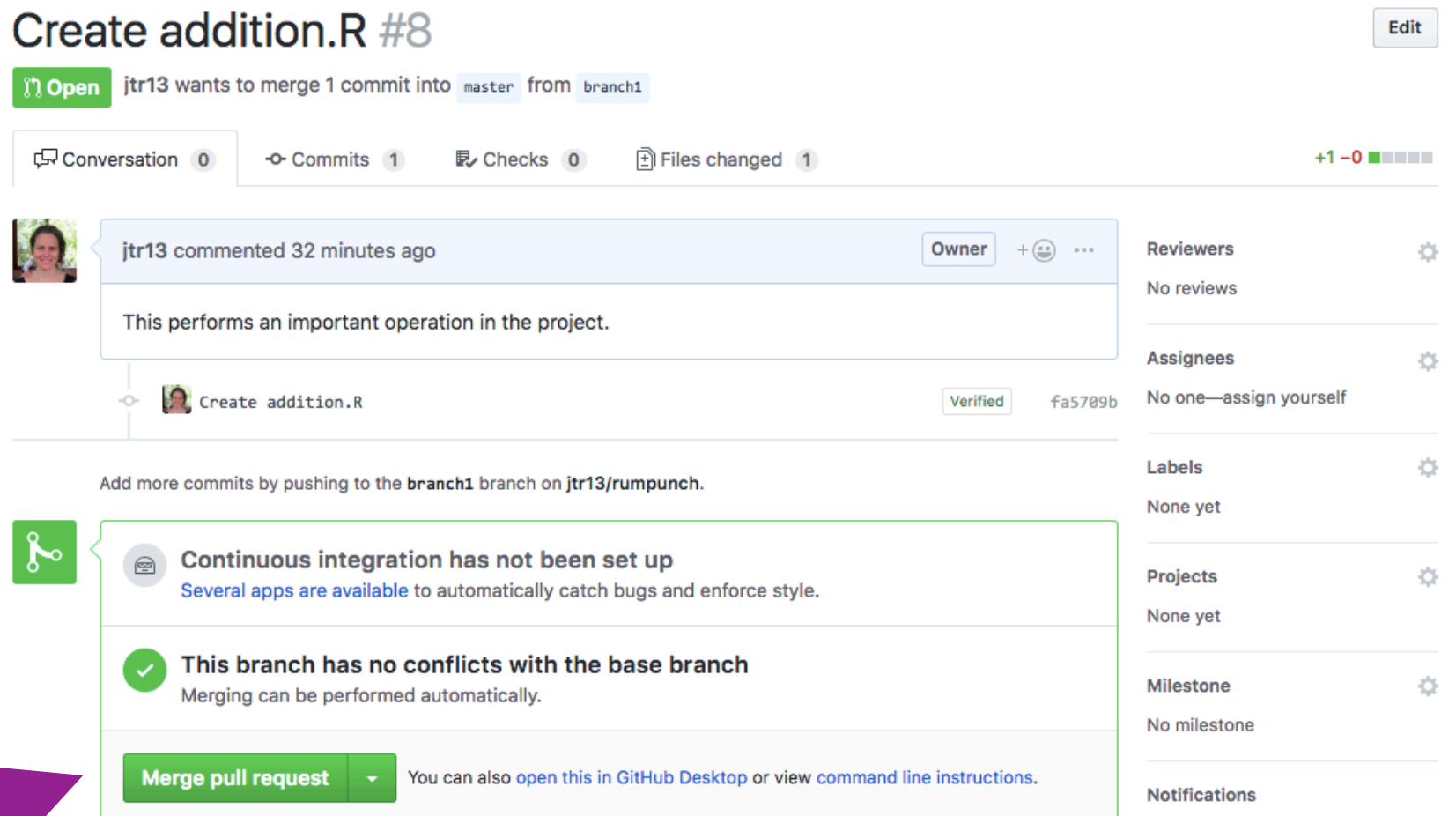


- Leave comments to the author to make edits (if applicable)



# Step 6: Merging a pull request

- Click back to return the pull request

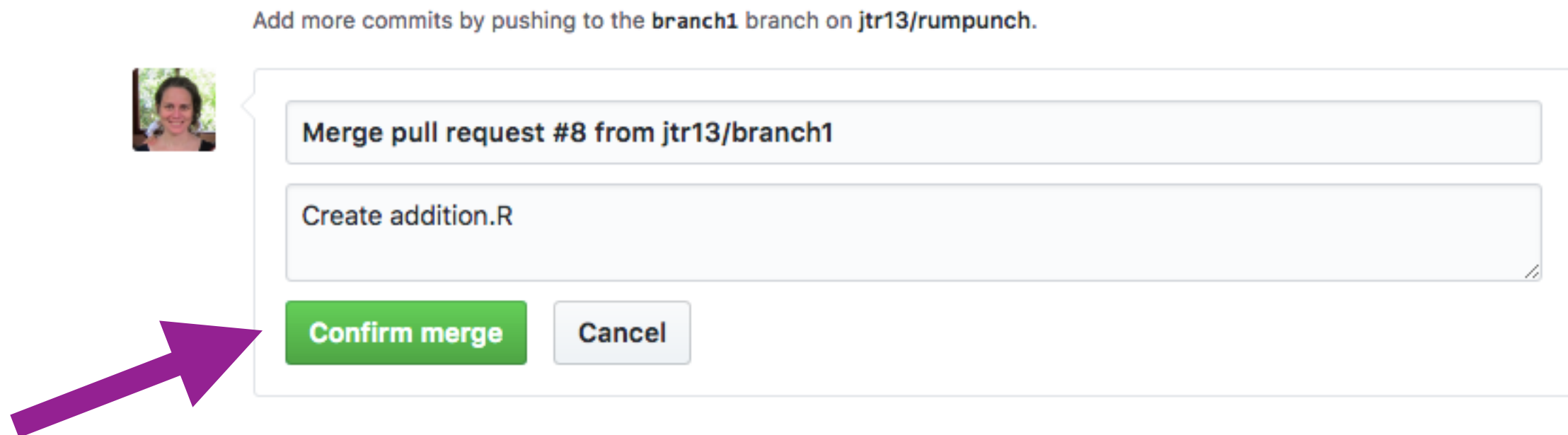


The screenshot shows a GitHub pull request titled "Create addition.R #8". At the top, it says "jtr13 wants to merge 1 commit into master from branch1". Below this, there are tabs for "Conversation" (0), "Commits" (1), "Checks" (0), and "Files changed" (1). A comment from the owner "jtr13" states: "This performs an important operation in the project." Below the comment is a commit titled "Create addition.R" with a "Verified" status and hash "fa5709b". A green box contains two status messages: "Continuous integration has not been set up" and "This branch has no conflicts with the base branch". At the bottom of this box is a green button labeled "Merge pull request". A purple arrow points to this button. To the right of the main content area, there are sections for "Reviewers", "Assignees", "Labels", "Projects", "Milestone", and "Notifications", each with a gear icon for settings.

- 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!

## Create addition.R #8

[Edit](#)

 **Merged** jtr13 merged 1 commit into `master` from `branch1` just now

 Conversation 0

 Commits 1

 Checks 0

 Files changed 1

+1 -0 



jtr13 commented an hour ago

Owner

+  ...

This performs an important operation in the project.



Create addition.R

Verified

fa5709b



jtr13 merged commit 9e6aeb9 into `master` just now

[Revert](#)



**Pull request successfully merged and closed**

You're all set—the `branch1` branch can be safely deleted.

[Delete branch](#)

Reviewers



No reviews

Assignees



No one—assign yourself

Labels



None yet

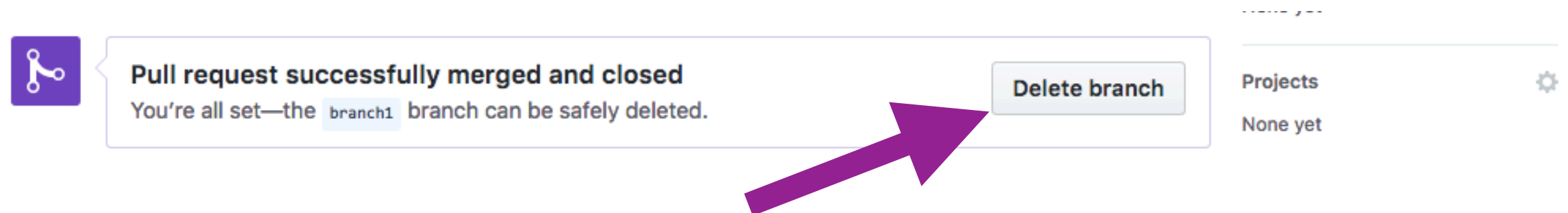
Projects



None yet

# 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
```

# Workflow for project groups

1. Create an organization, or one person creates the repo and gives others write access
2. **Discuss** a workflow, for example, take turns being the one who merges pull requests
3. Create a test repo to practice and try to create merge conflicts

# More resources

<https://edav.info/github.html>

# 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