# Current Title:
# "How I Use Git/GitHub"

Future title: "Simple Git/GitHub Workflows for Beginners"

Comments welcome!

Submit a pull request, create an issue, or email me:

jtr13@columbia.edu  Joyce Robbins

# Notes to Reviewers

The real slides will not have so much text, as I will be making the points verbally.  Please focus on the content, not the format of the slides.

# Main Points of Talk

- You don't have to be an expert to derive great benefit from Git/GitHub. I am not an expert and yet I couldn't live without these tools.
- It seems to me that there's a need for a tutorial that focuses on simple workflows when things are going right, to supplement "How do you...?" type tutorials.

# Git/GitHub Workflows that will be covered here

1. GitHub only
2. GitHub + local master branch
3. GitHub + local master branch plus feature branch

# 1. GitHub only

# 1. 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.

# 1. 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.**

# 1. GitHub only

- Example:

# 1. Remote Master Branch

GitHub

origin/master

What can you do?

- Upload files to share (any type)
- Create files (generally text / markdown files since there's no place to code.
- Edit files

Benefits:

- Share and back up files, get comments (issues) and corrections (pull requests) from others

# 1. Remote Master Branch

GitHub

origin/master

What can you do?

- Upload files to share (any type)
- Create files (generally text / markdown files since there's no place to code.
- Edit files

Benefits:

- Share and back up files, get comments (issues) and corrections (pull requests) from others

# Terminology

Think in terms of *repositories* and *branches*

**Types of Repositories (from your perspective)**

**local** repository -- resides on *your* computer

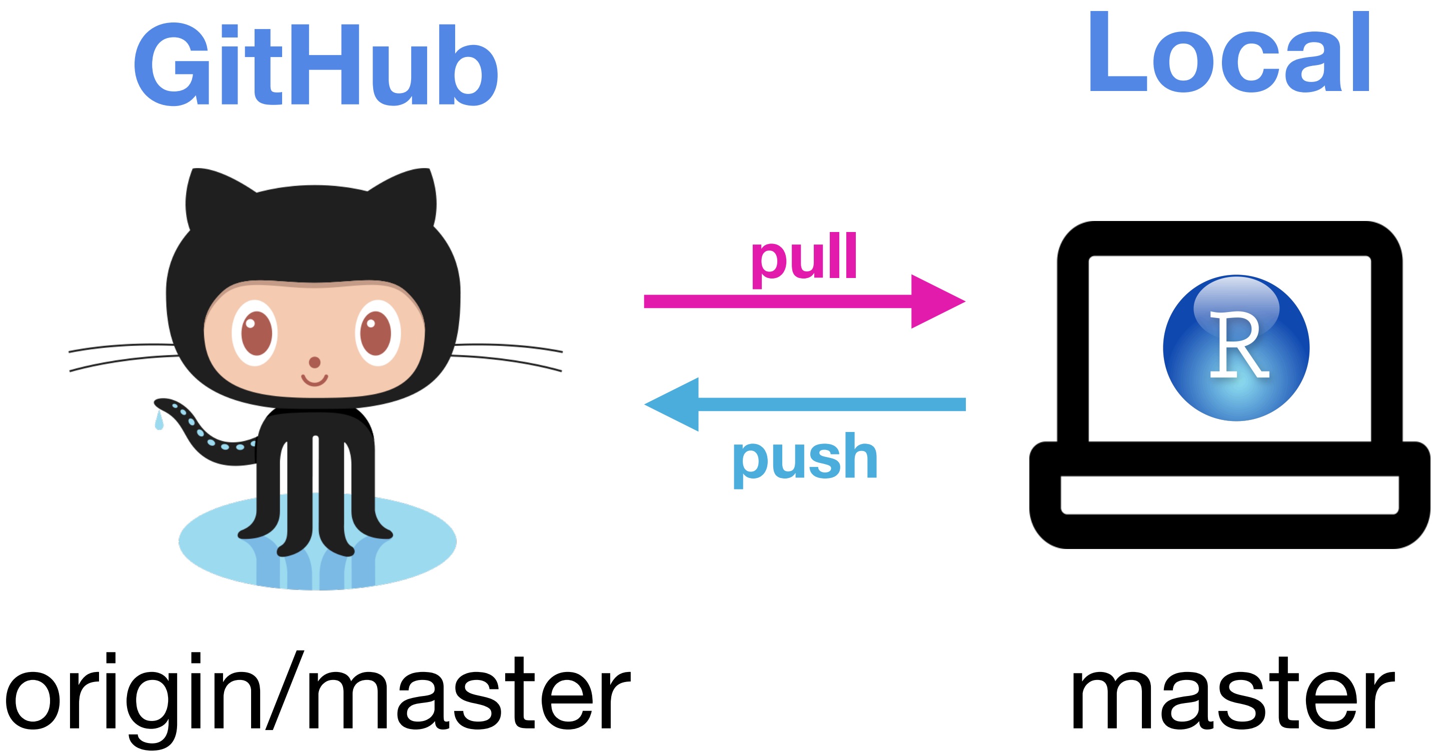
**remote** repository -- resides somewhere else

**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

# 2. Remote + local master branches



GitHub

Local

pull

push

origin/master

master

What can you do? All of 1. plus:

- Work locally with your favorite tools
- Keep a working version available to the public while you add features
- Share your code on GitHub when you're ready

# 3. Remote + local master + feature branches

## From the perspective of the project:

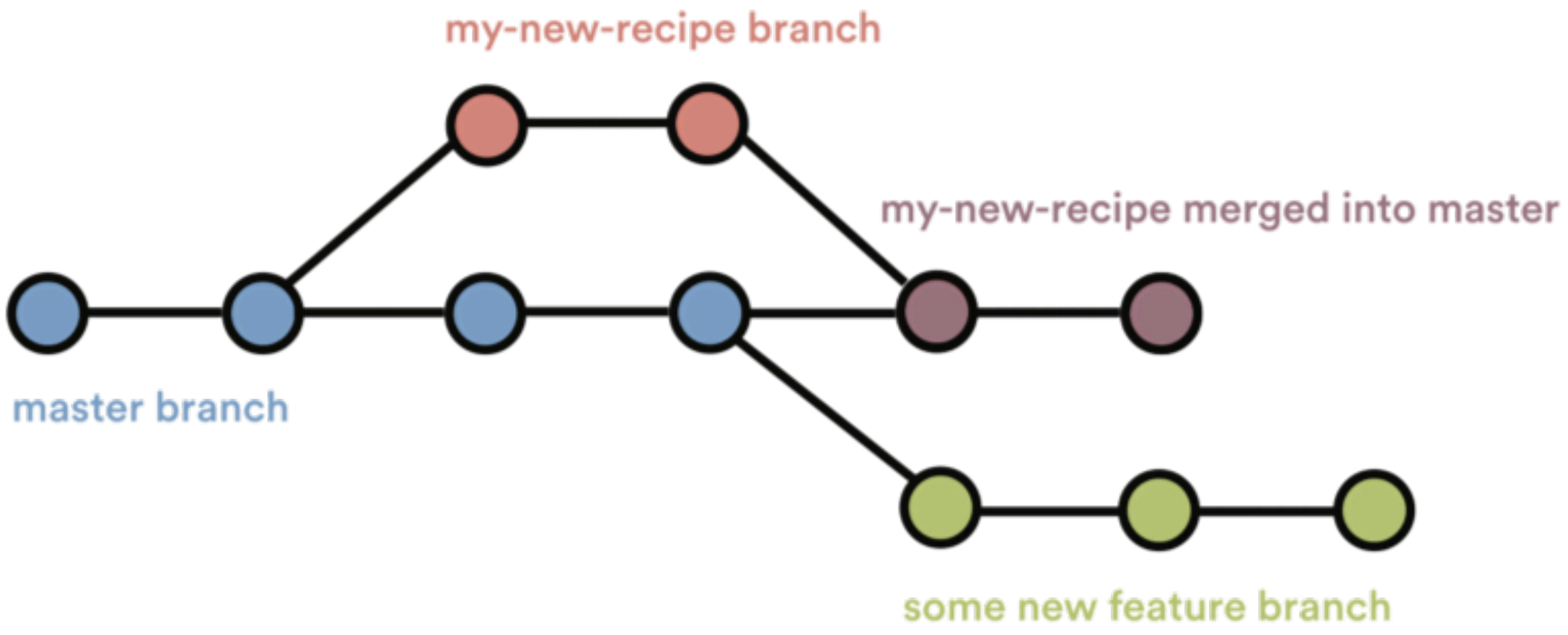
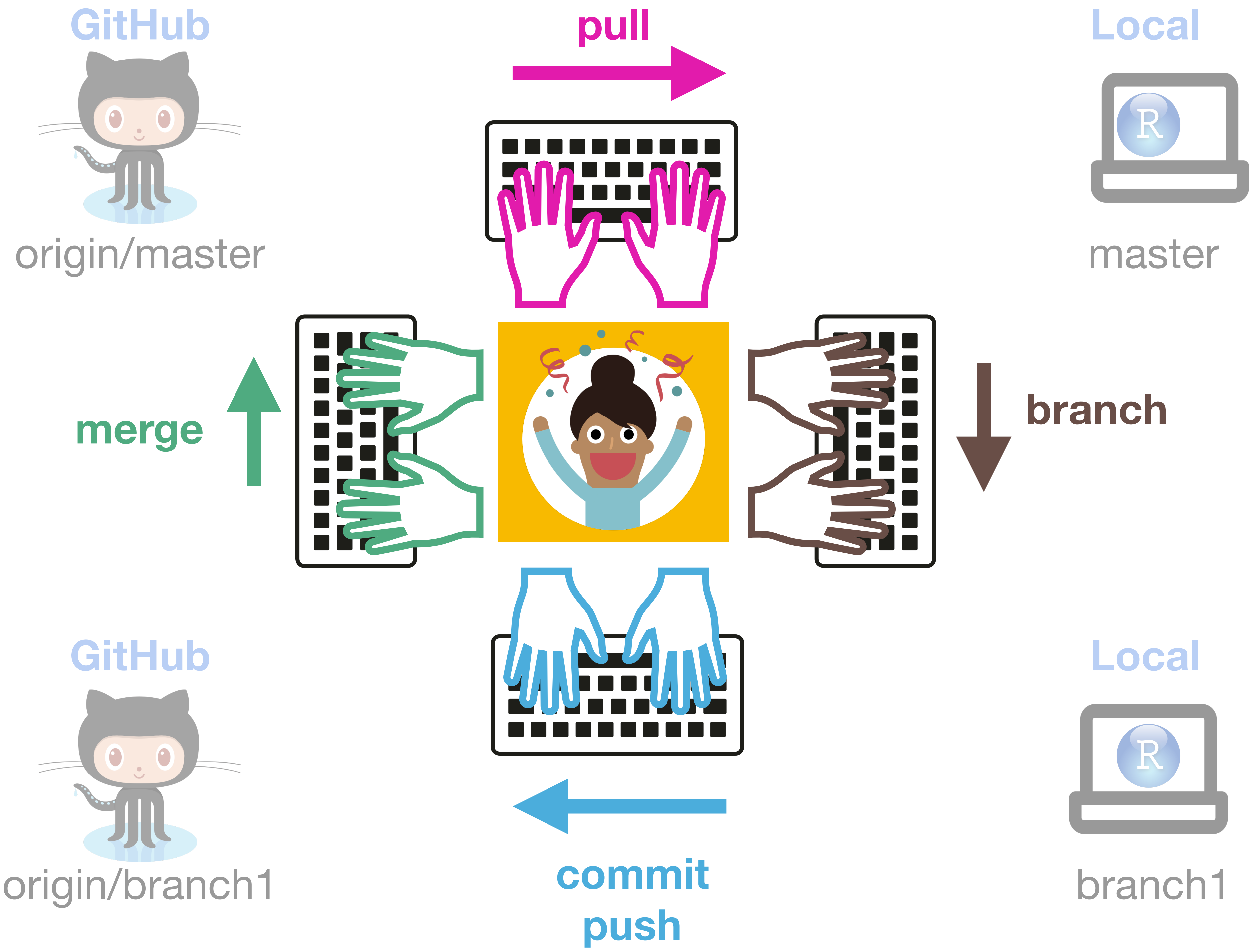
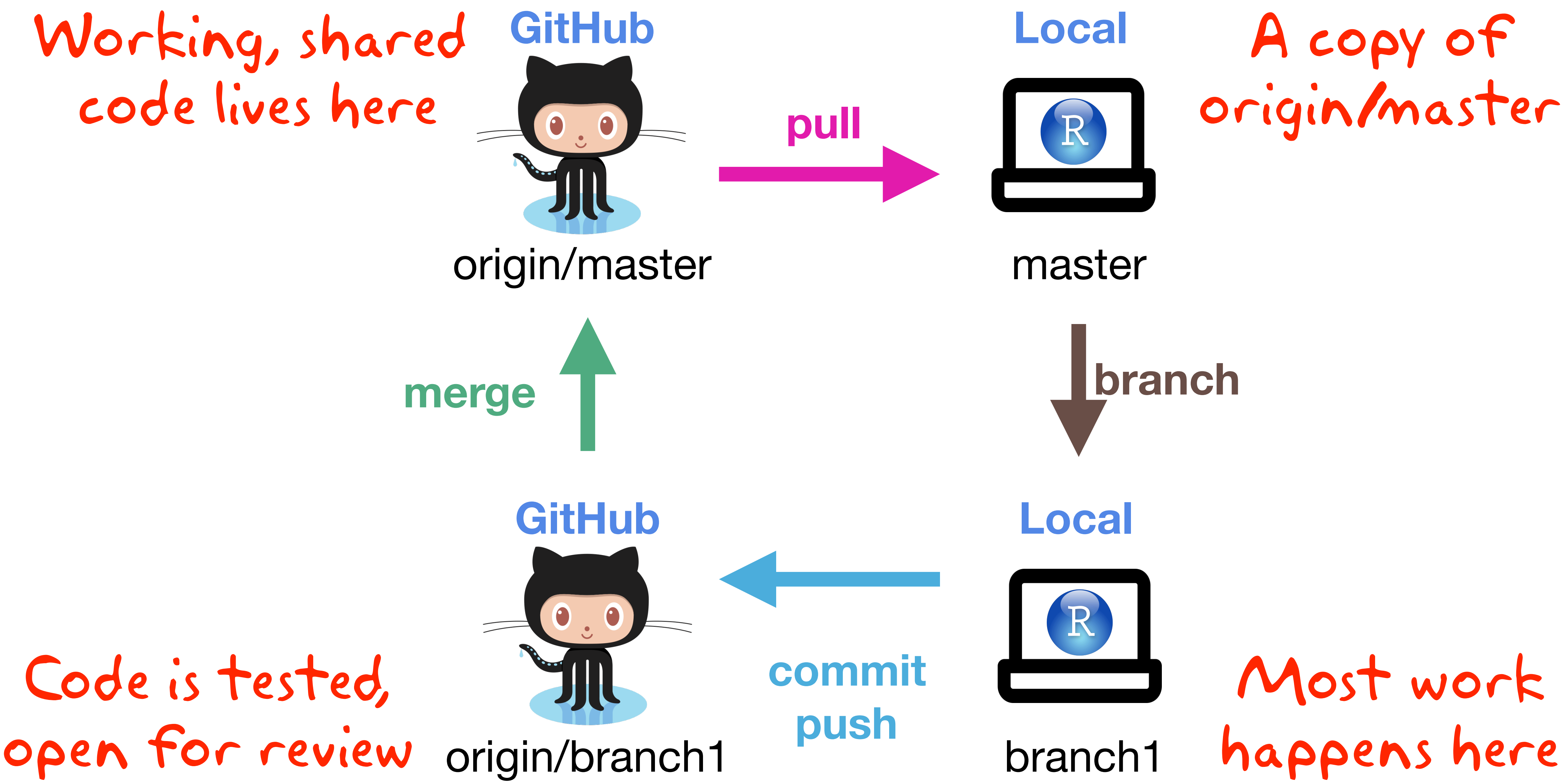


Image source: https://medium.com/designing-atlassian/learn-to-love-git-part-three-collaboration-a778157d3a1c

# From your perspective:



# From your perspective:



Working, shared code lives here

GitHub

origin/master

pull

Local

master

A copy of origin/master

branch

Local

branch1

Most work happens here

commit push

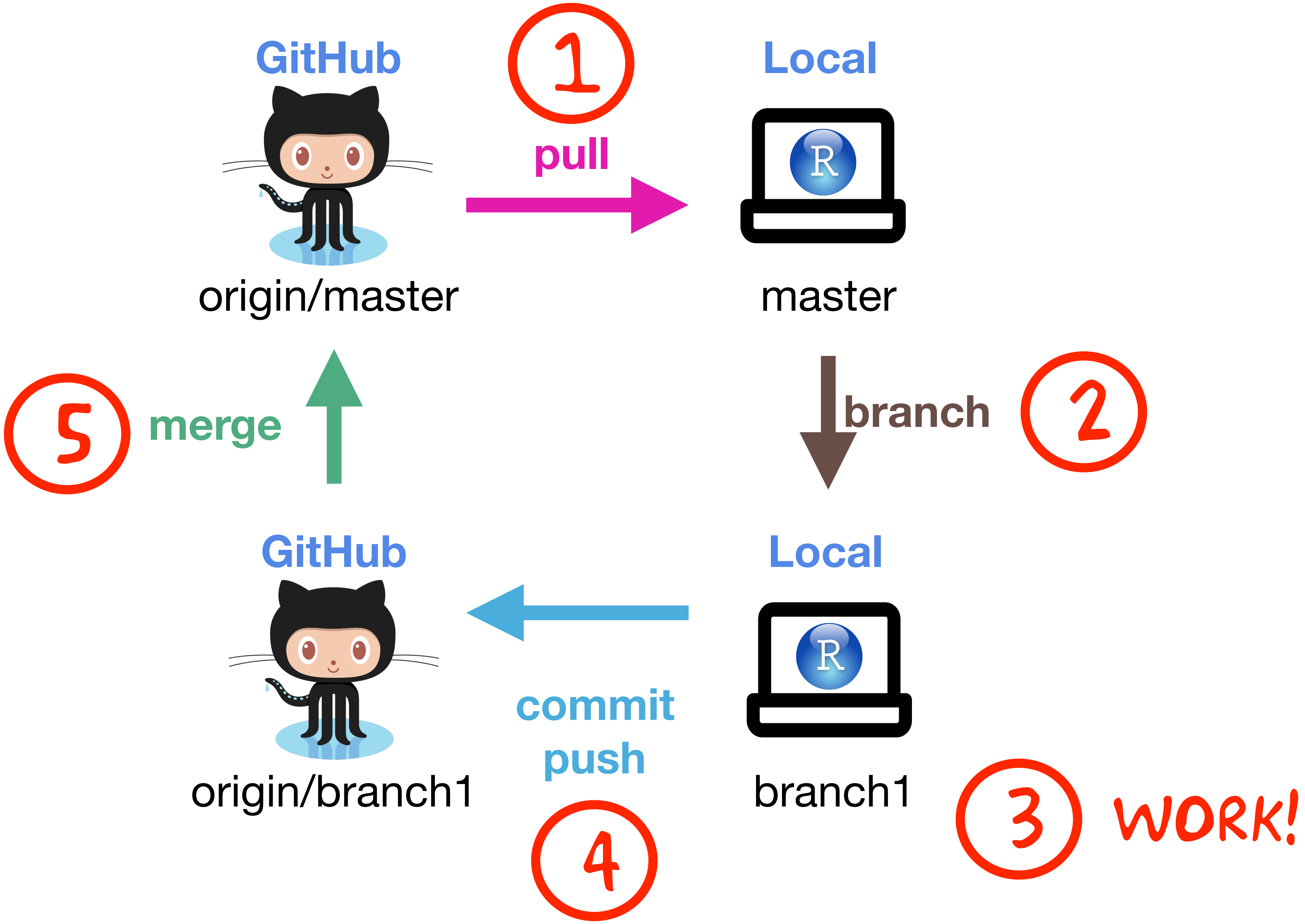
GitHub

origin/branch1

Code is tested, open for review

merge

# Your workflow



GitHub

origin/master

1 pull

Local

master

branch 2

Local

branch1

3 WORK!

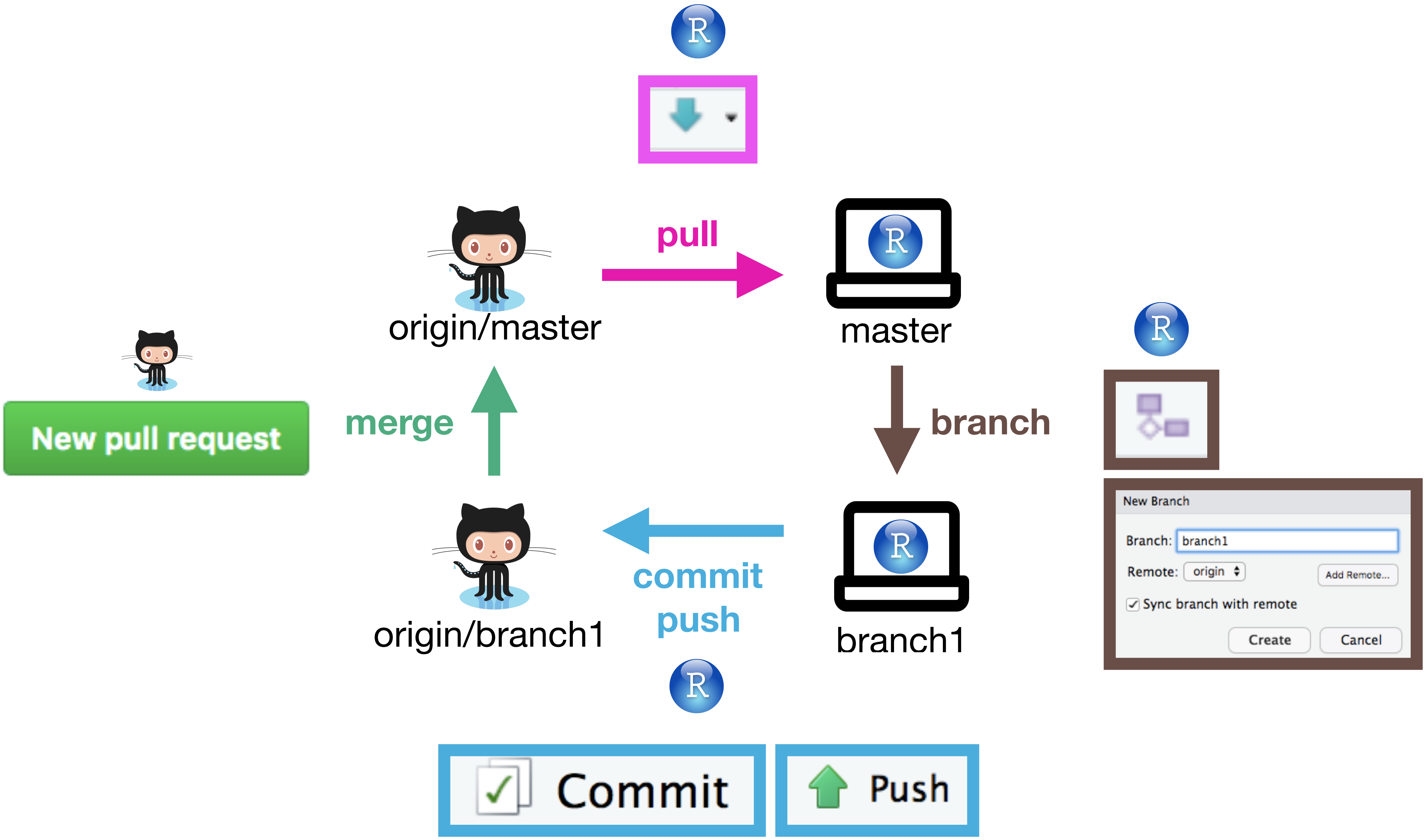
commit
push
4

GitHub

origin/branch1

5 merge

# Workflow: RStudio + GitHub



New Branch

Branch: branch1

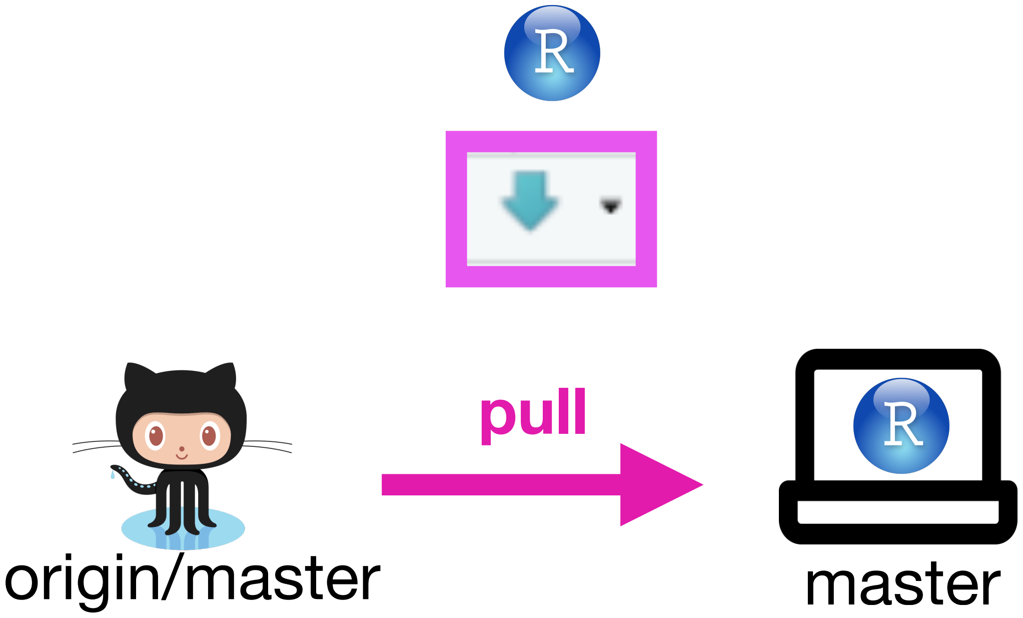
Remote: origin

Add Remote...

☑ Sync branch with remote

Create Cancel

# Step 1.  Pull



pull

origin/master

master

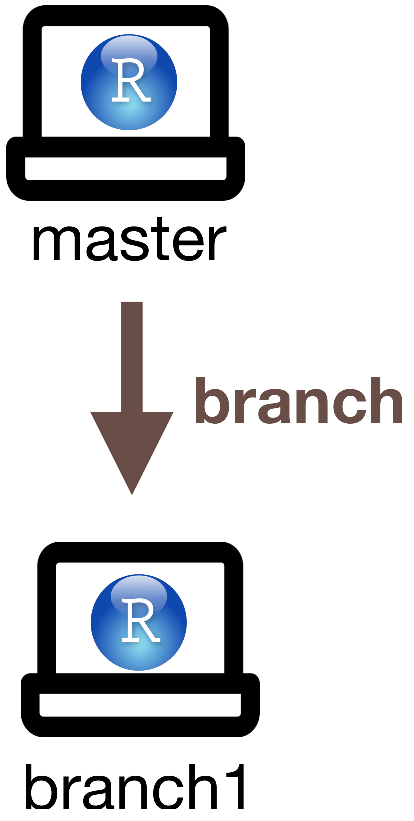
Points:

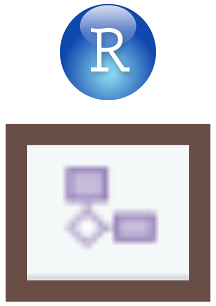
*  We want to make sure that we begin working locally, we're up-to-date with the remote (GitHub).

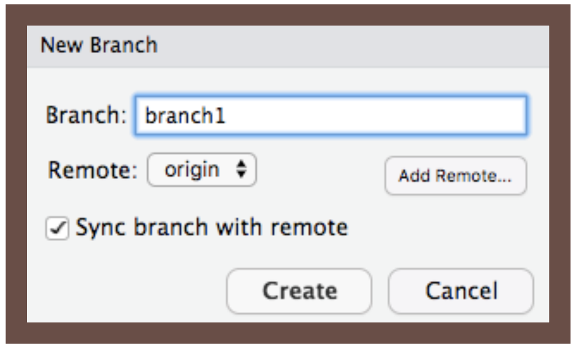
*  Assuming there are no conflicts, we'll either get a message that we're already up-to-date, or something like this:

```
>>> 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






New Branch

Branch: branch1

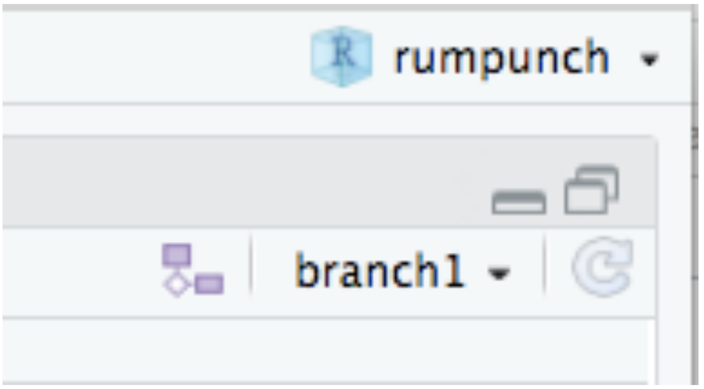
Remote: origin

☑ Sync branch with remote

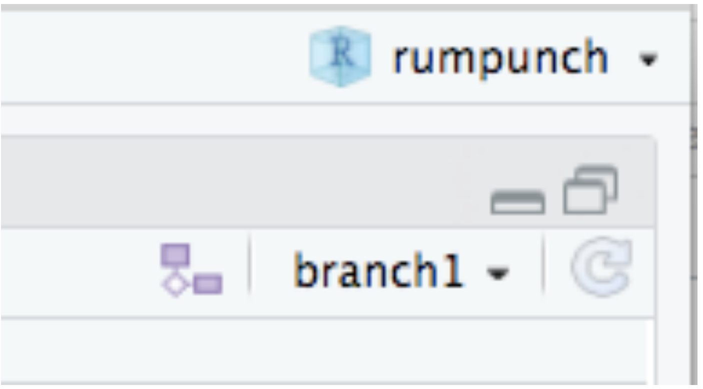
Points:

* This is where we'll do our work.

* Check the top right corner of the screen frequently to make sure you're on the right branch:

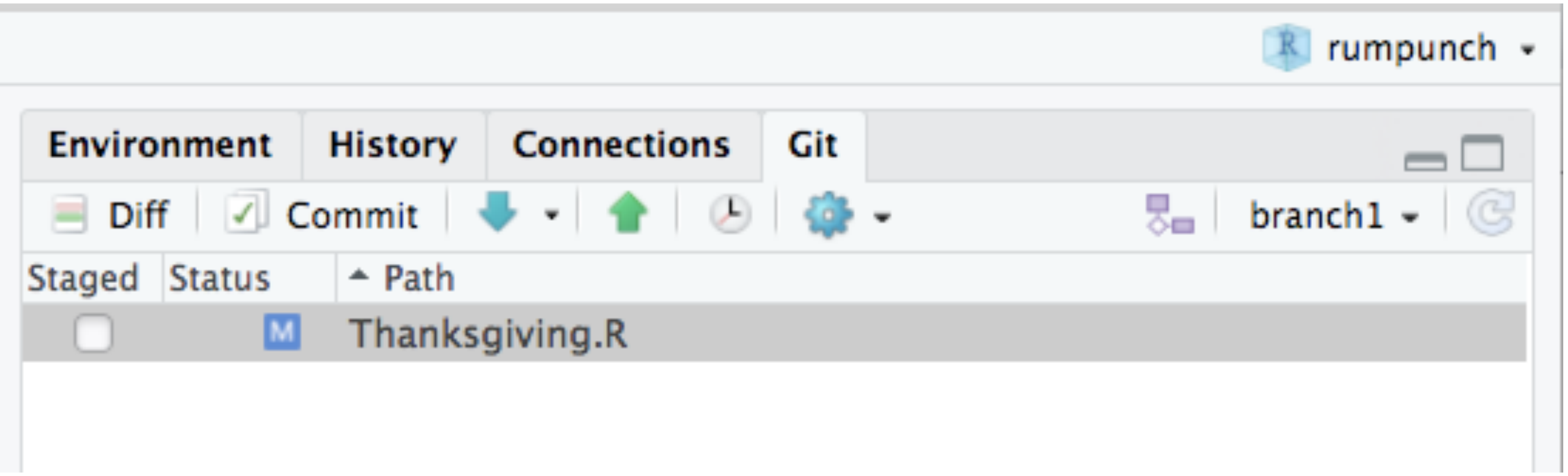


# Step 3: Work



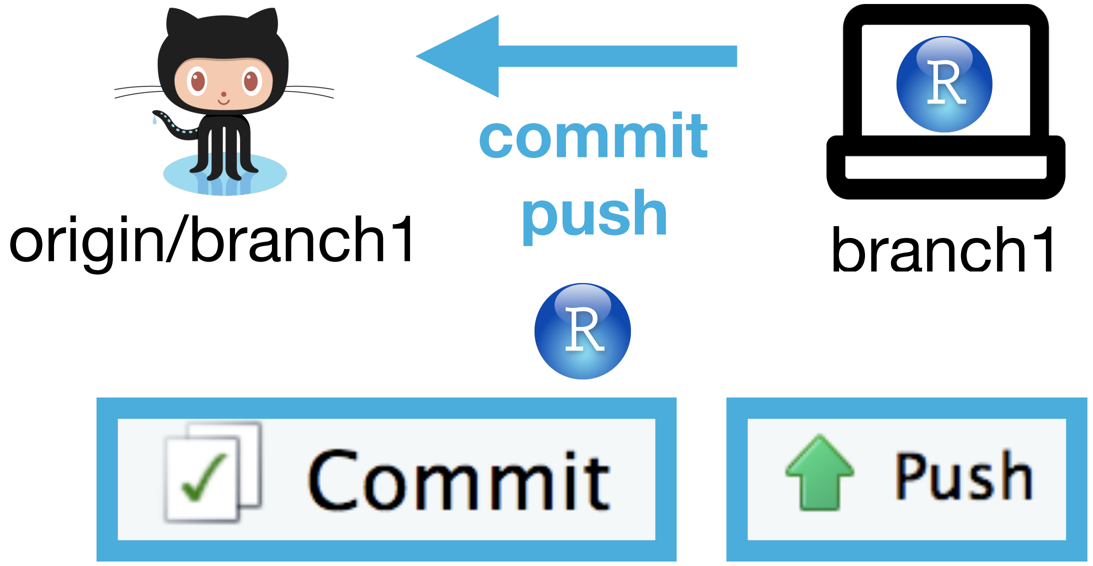
Points:

* Do all your work as you usually do.
* Save your work as usual.
* Add files to .gitignore if you don't want them to be synced with remote
* Check the Git panel to see how things are changing:



| Staged | Status | Path |
|---|---|---|
| □ | M | Thanksgiving.R |

# Step 4: Commit and push



origin/branch1

commit
push

branch1

Points:

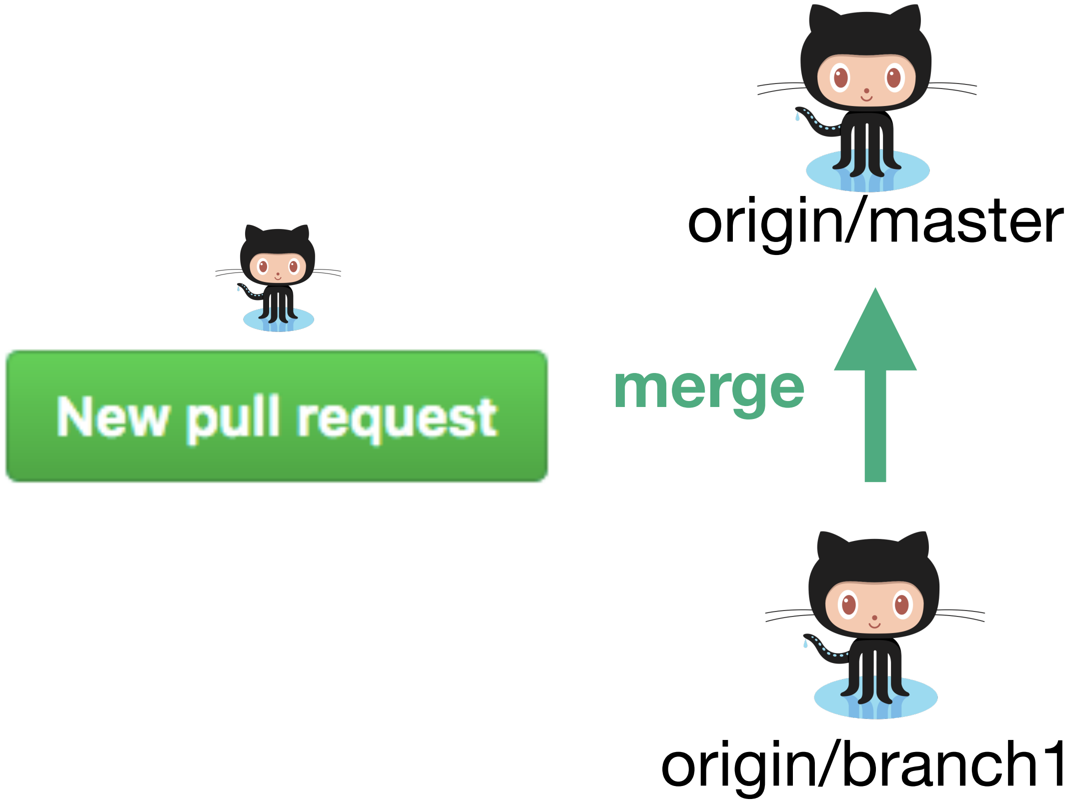
* When you're ready to merge your work into master, check the files that you want to commit in the Staged column.

* Click "Commit", add a commit message, click "Commit" (again).

* Click the push button. If all goes well, you'll get a message like this:

```
>>> git push origin refs/heads/branch3
To https://github.com/jtr13/rumpunch.git
   7424222..6cf5975  branch3 -> branch3
```

# Step 5: Merge

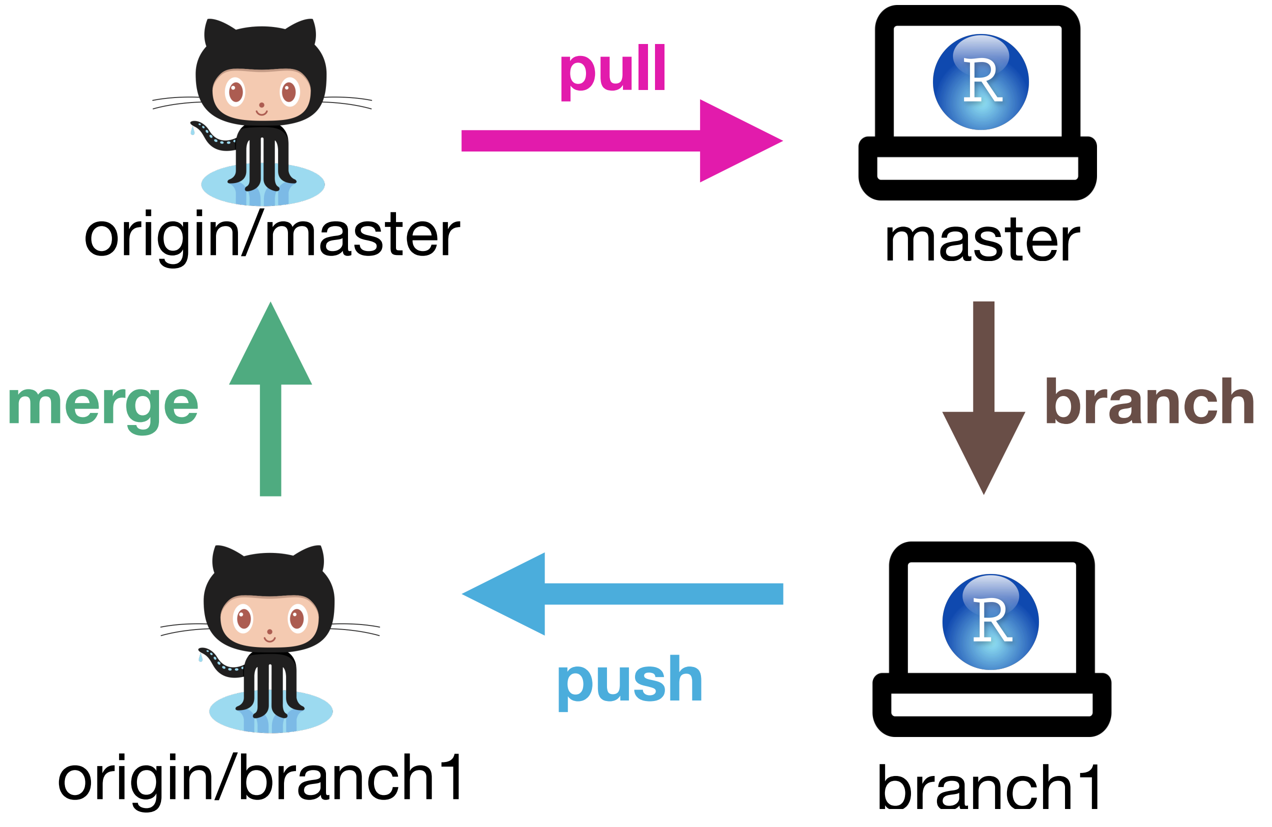


origin/master

merge

origin/branch1

# 3. RStudio workflow



origin/master

pull

master

branch

branch1

push

origin/branch1

merge