

# **Git and GitHub**

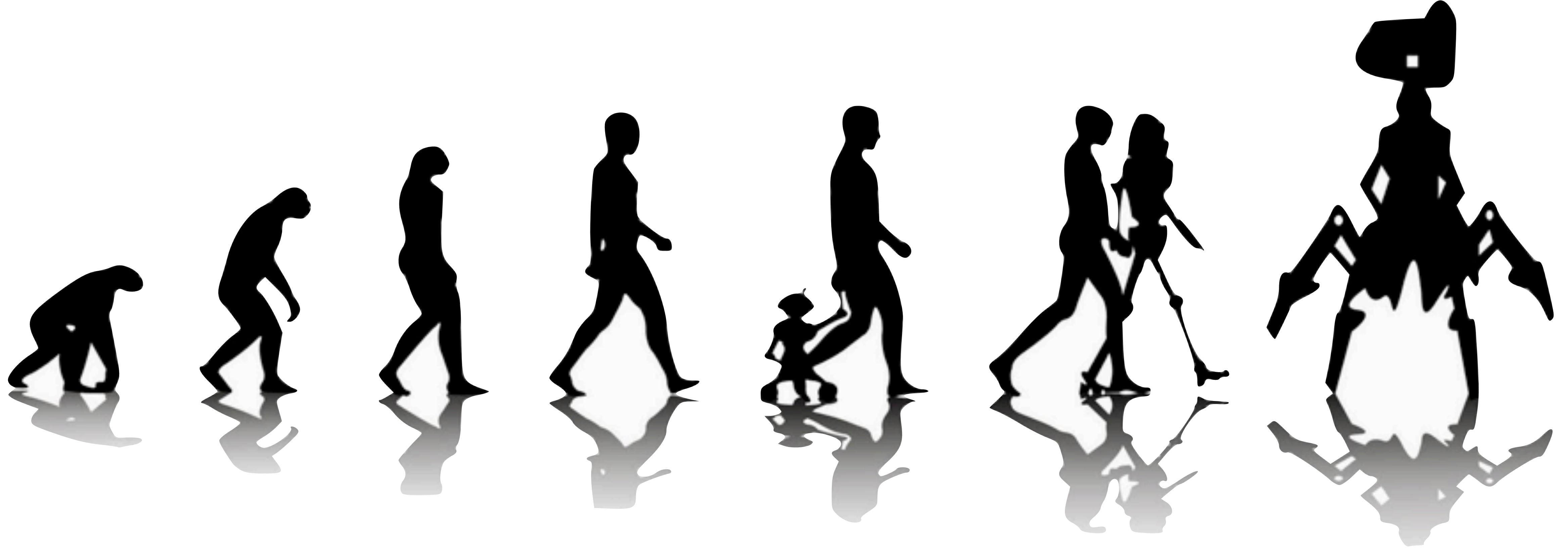
**Part 1: Early workflows**

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

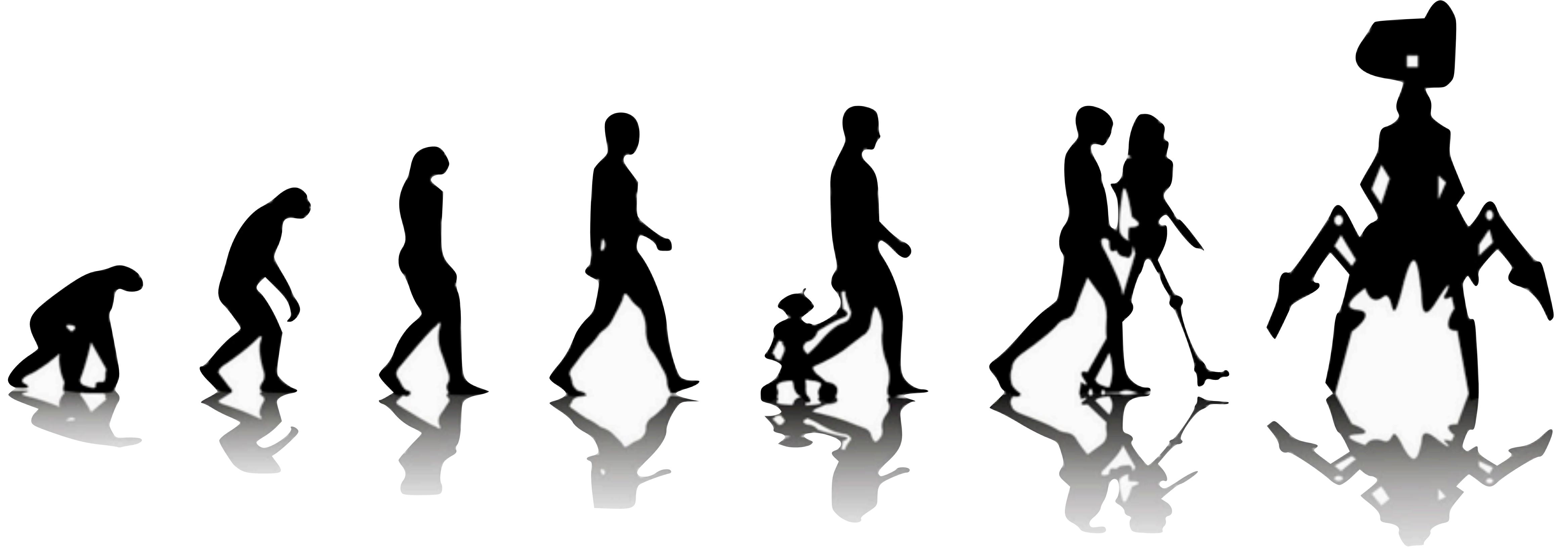
To view a copy of this license, visit  
<http://creativecommons.org/licenses/by-sa/4.0/>

*Deep  
Thoughts*

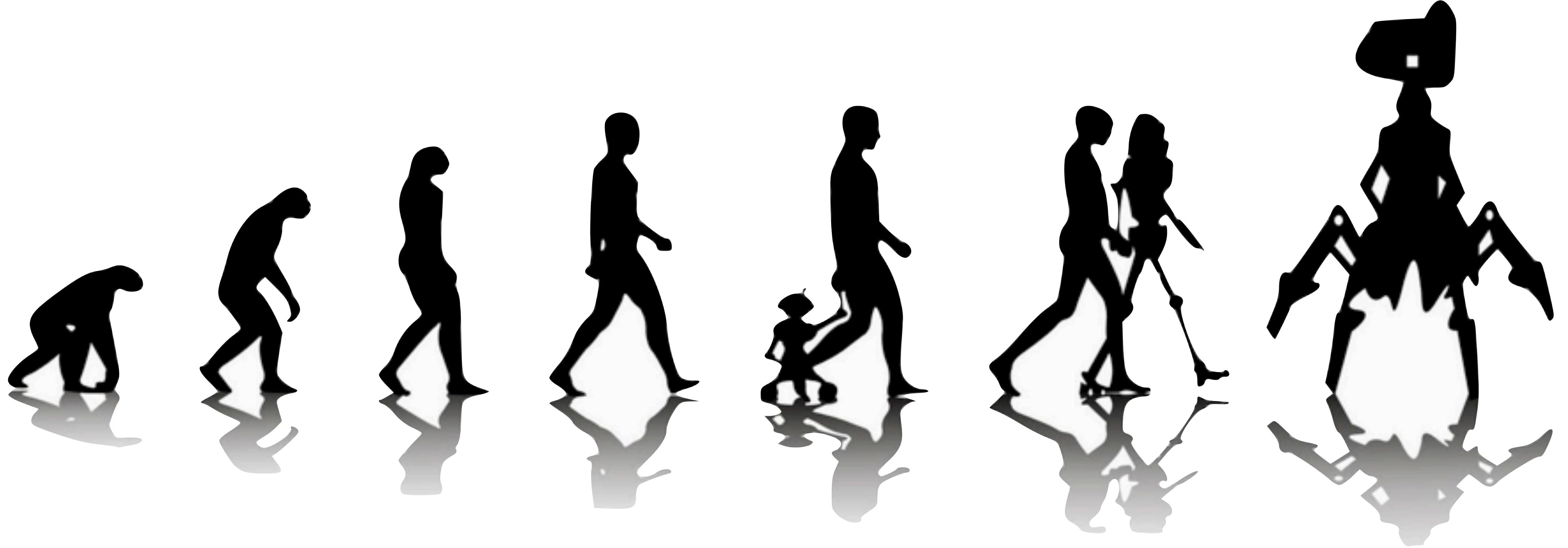




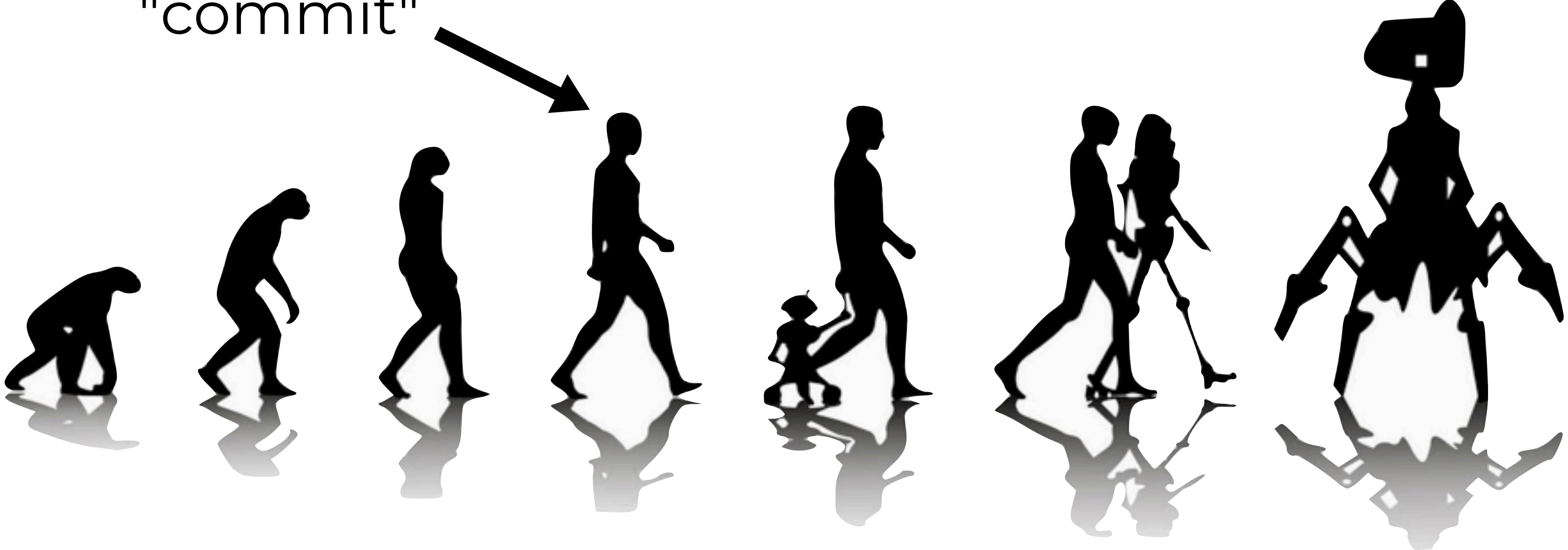
use version control



we teach Git & GitHub



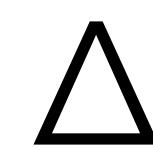
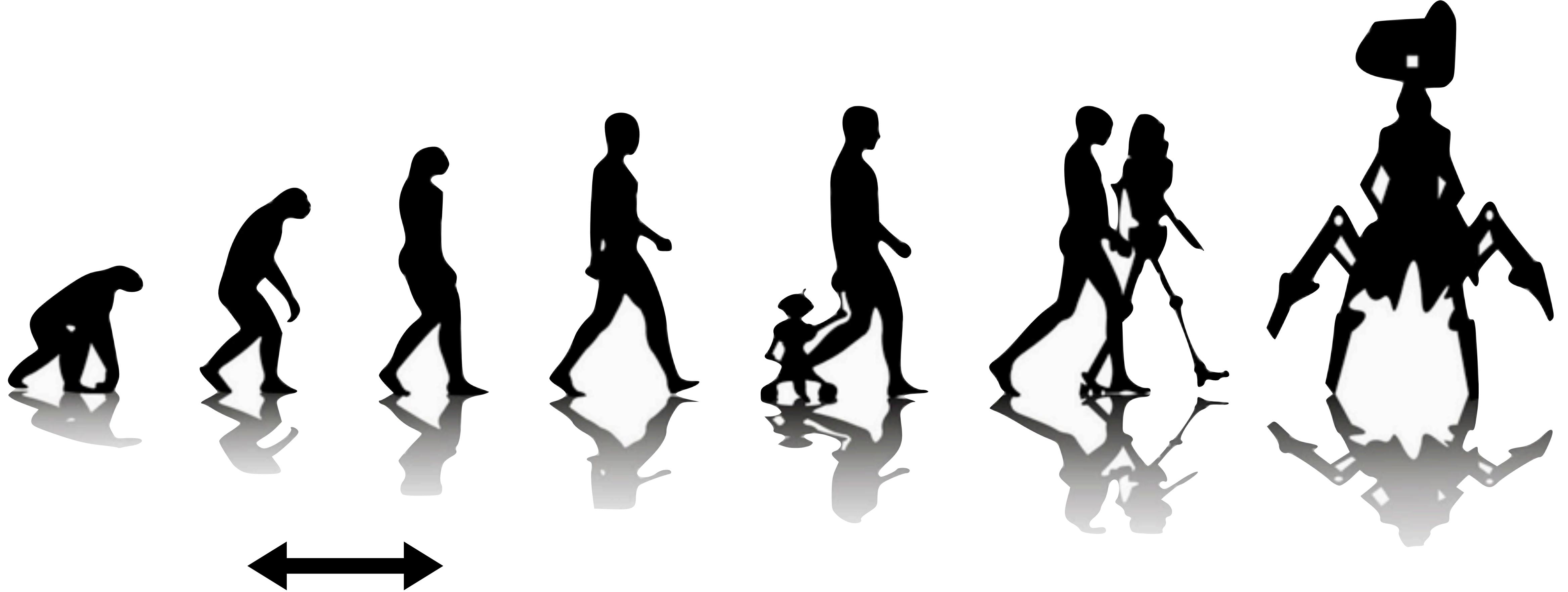
a **repository or repo** is a collection of files,  
representing a project,  
which might also be a an RStudio Project



"commit"

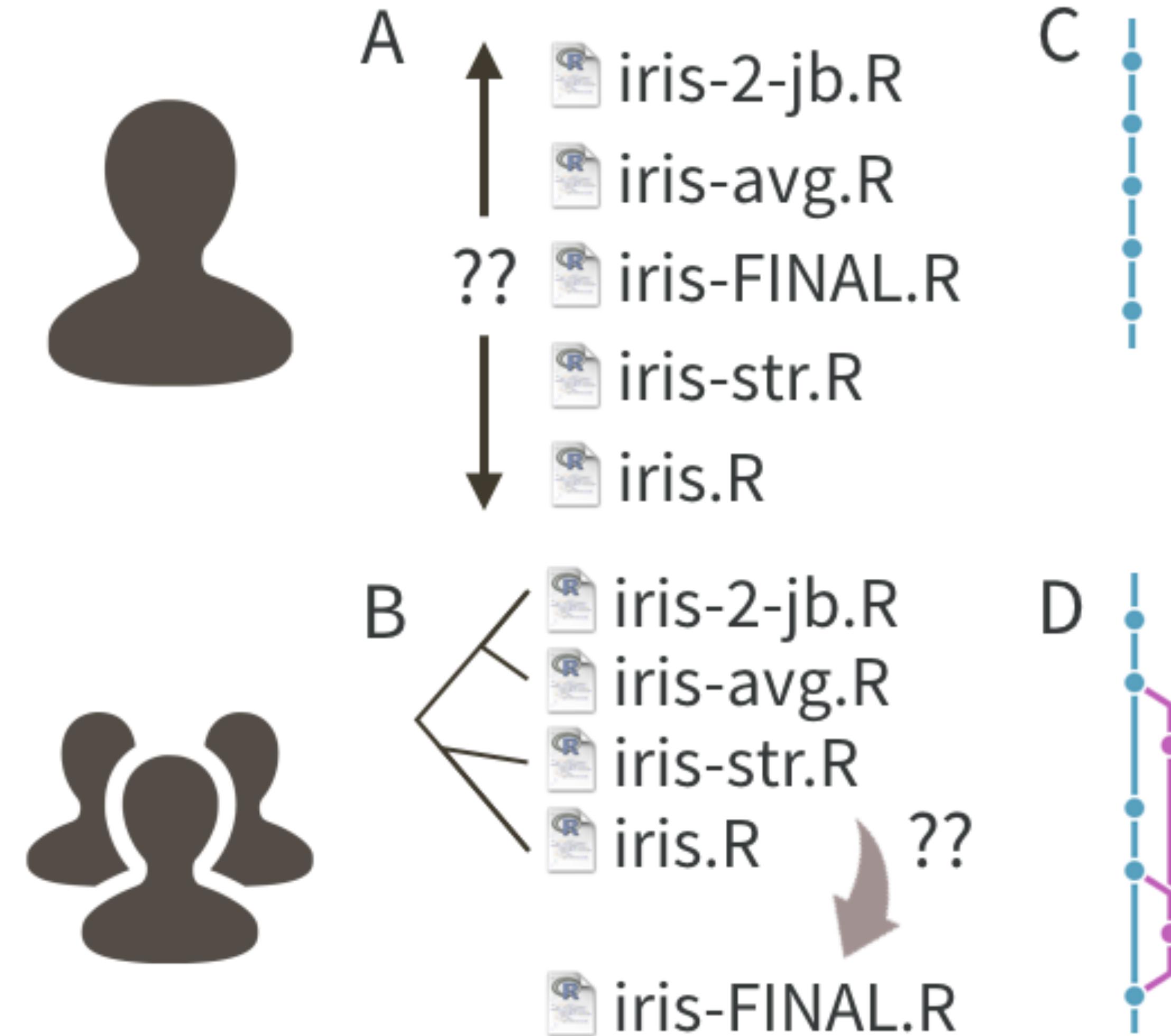
A black and white illustration showing the progression of human evolution. It starts on the left with a knuckle-walking primate and ends on the right with a person wearing a virtual reality headset. The figures are shown in various stages of development, with their shadows cast onto the ground below them. An arrow points from the word "commit" towards the final figure on the right.

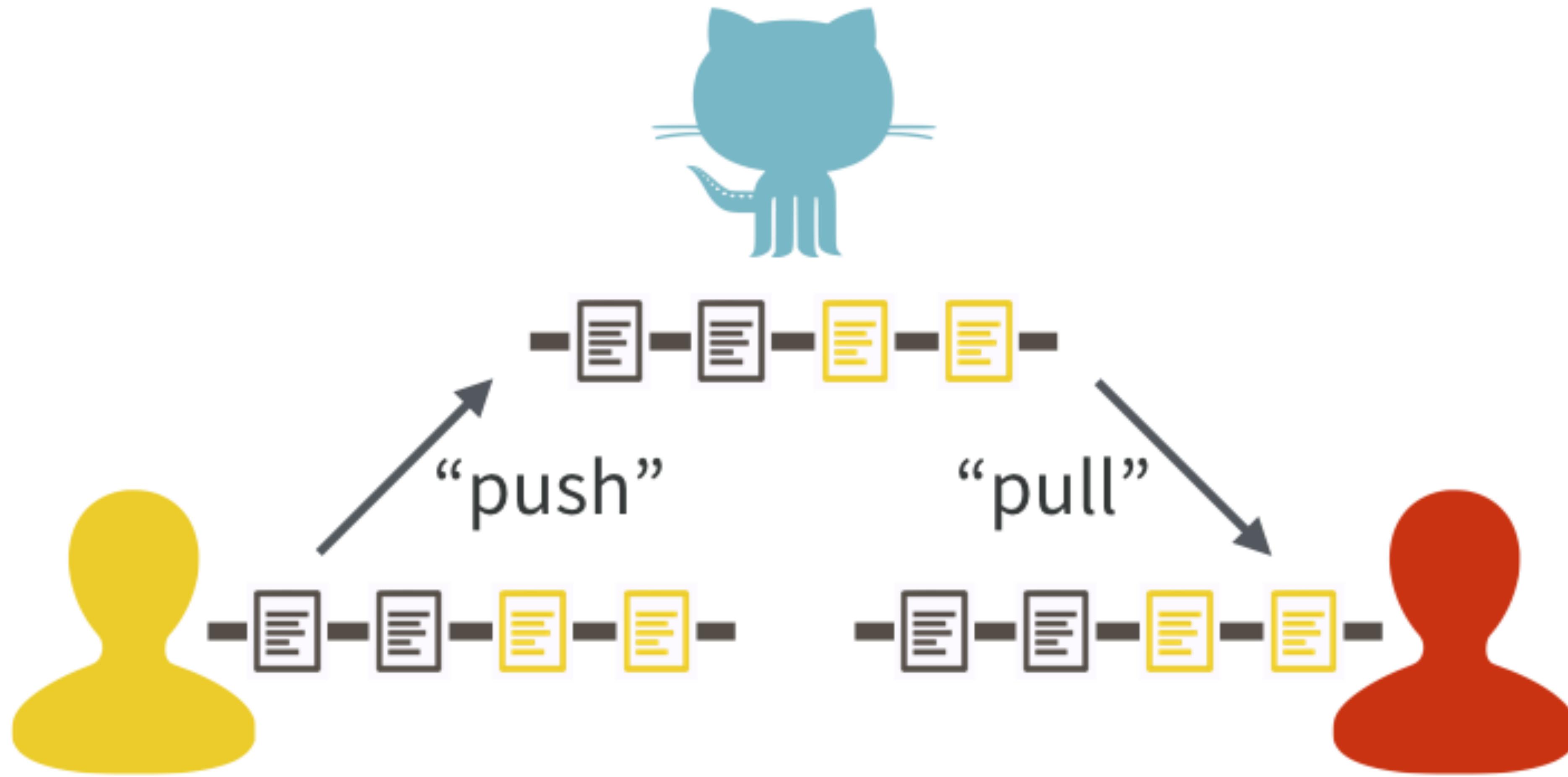
a file or project state that is **meaningful to you**  
for inspection, comparison, or restoration



"diff"

What changed here?  
Why?



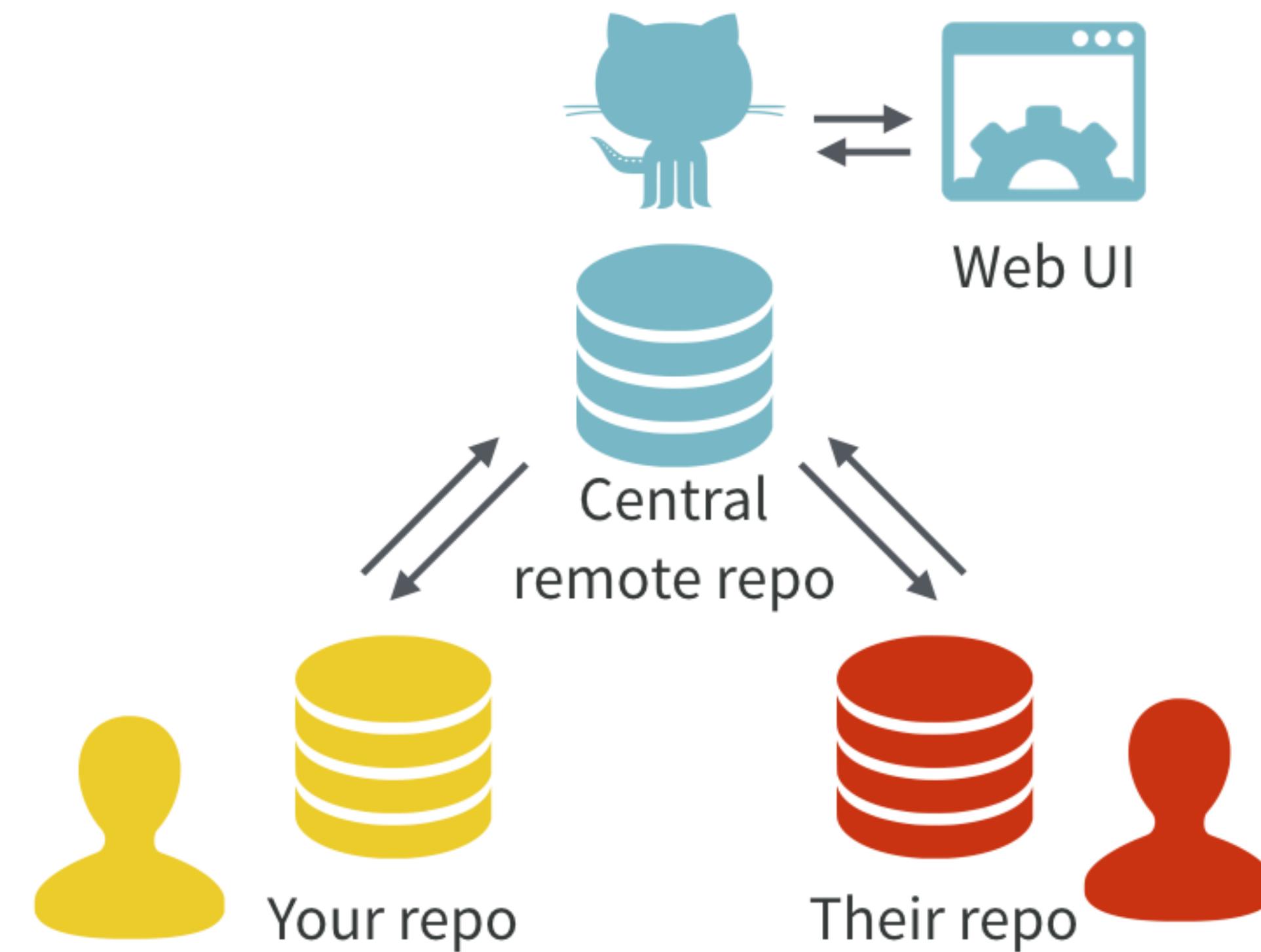


# collaboration

# Excuse me, do you have a moment to talk about version control?

<https://doi.org/10.1080/00031305.2017.1399928> in The American Statistician

<https://doi.org/10.7287/peerj.preprints.3159v2> in PeerJ Preprints



# Git for humans

Alice Bartlett

Senior Developer, Financial Times

@alicebartlett

Excellent presentation by Alice Bartlett,  
originally delivered in 2016 at UX Brighton.

<https://speakerdeck.com/alicebartlett/git-for-humans>

happygitwithr.com



# Why use version control?

- experiment without fear
- explore cause and effect
- embrace incrementalism
- expose your work
- collaborate

# how Git feels



# get off the beach!



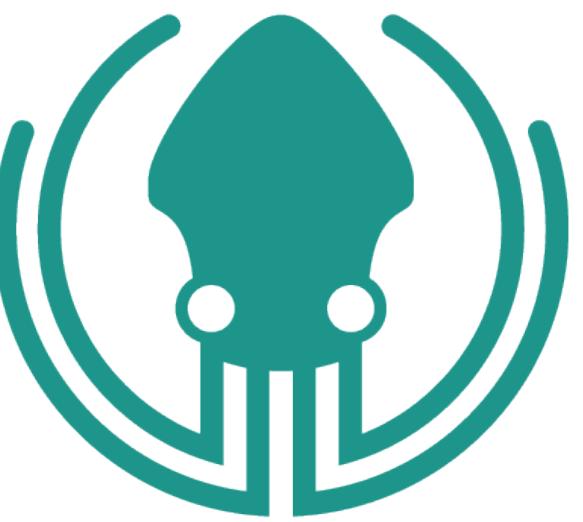


accumulate reps



agony : flow





agony:flow

# Use a graphical Git client, if you want

- No one is giving out merit badges to people who only use command line Git.
- I use RStudio and GitKraken.
- <http://happygitwithr.com/git-client.html>





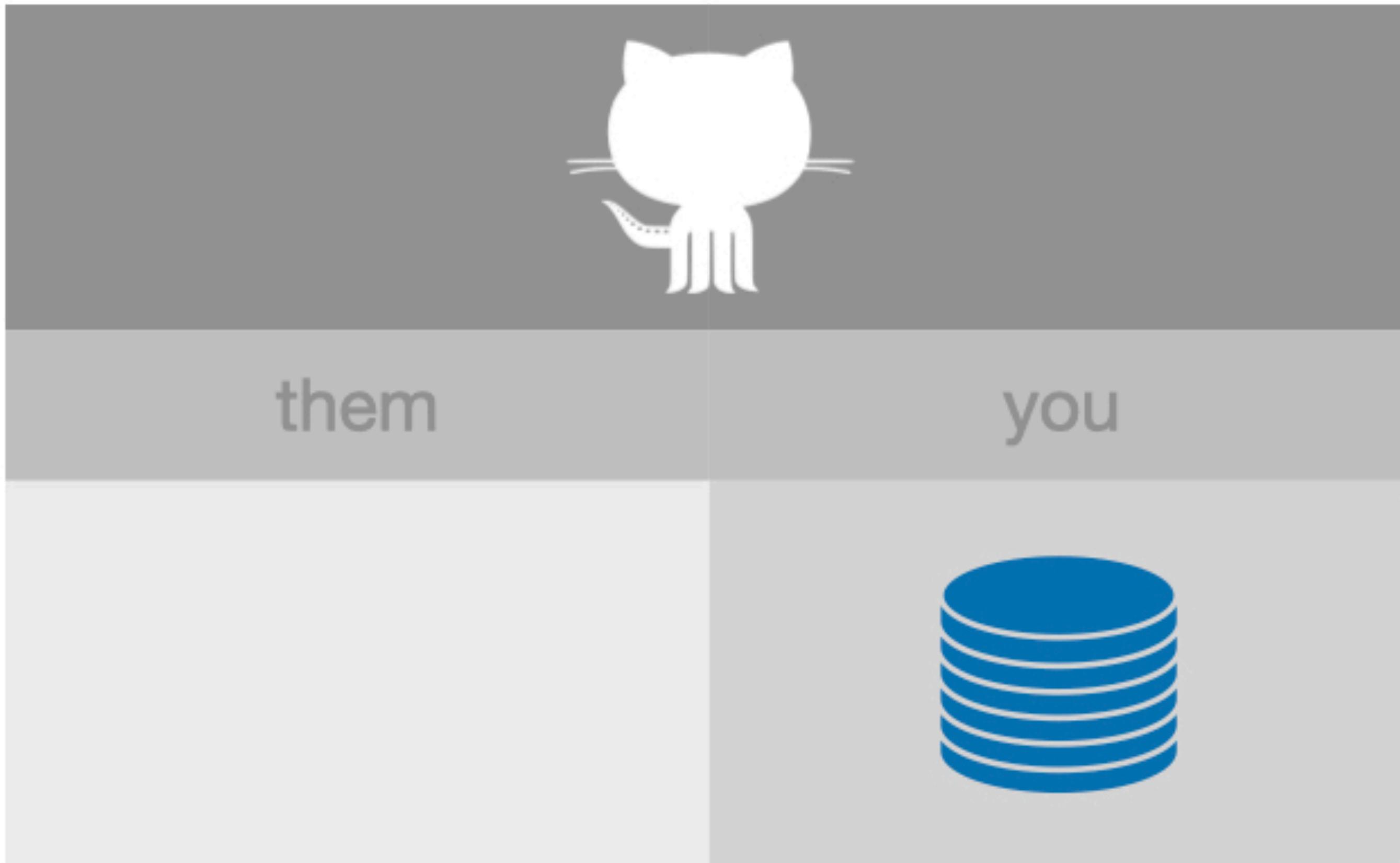
them

you

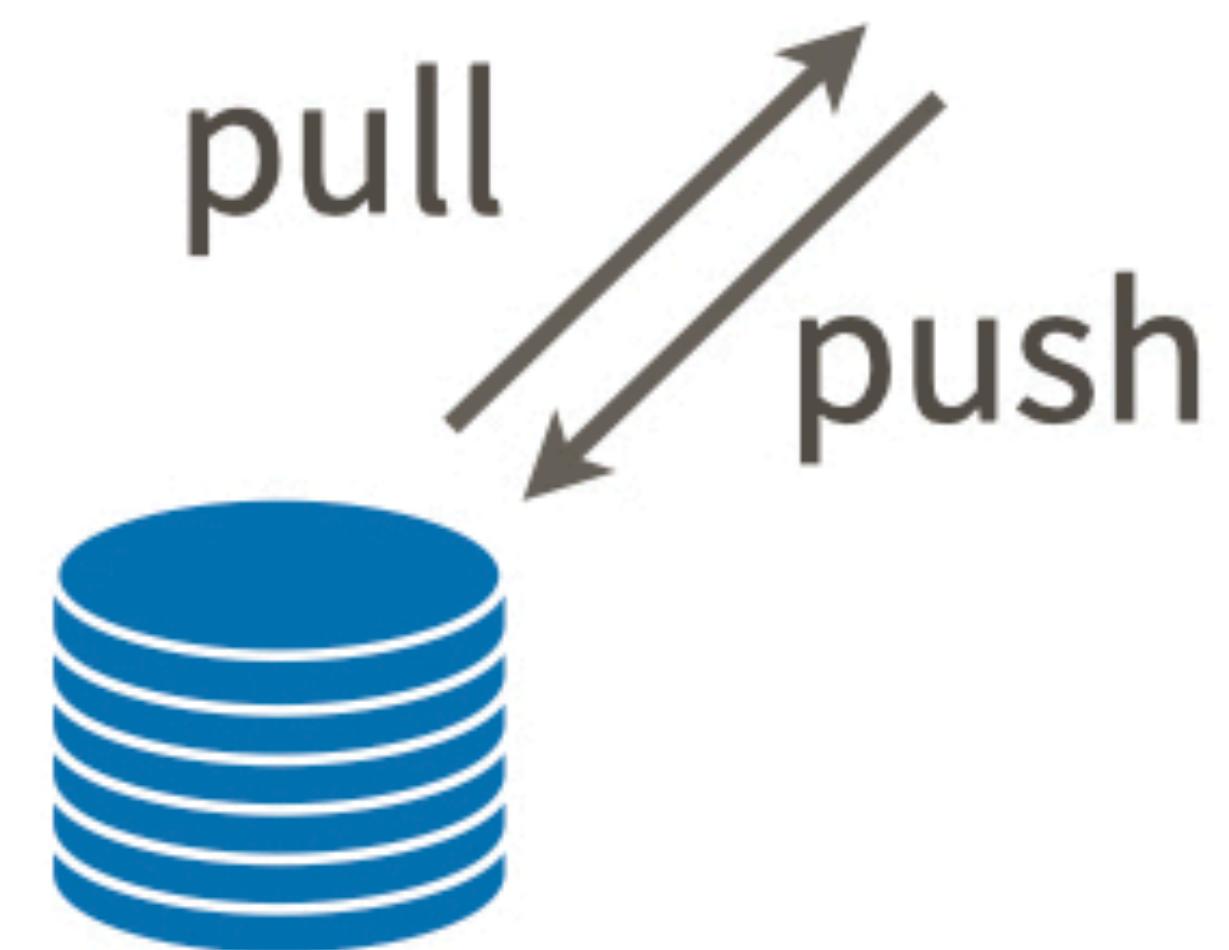
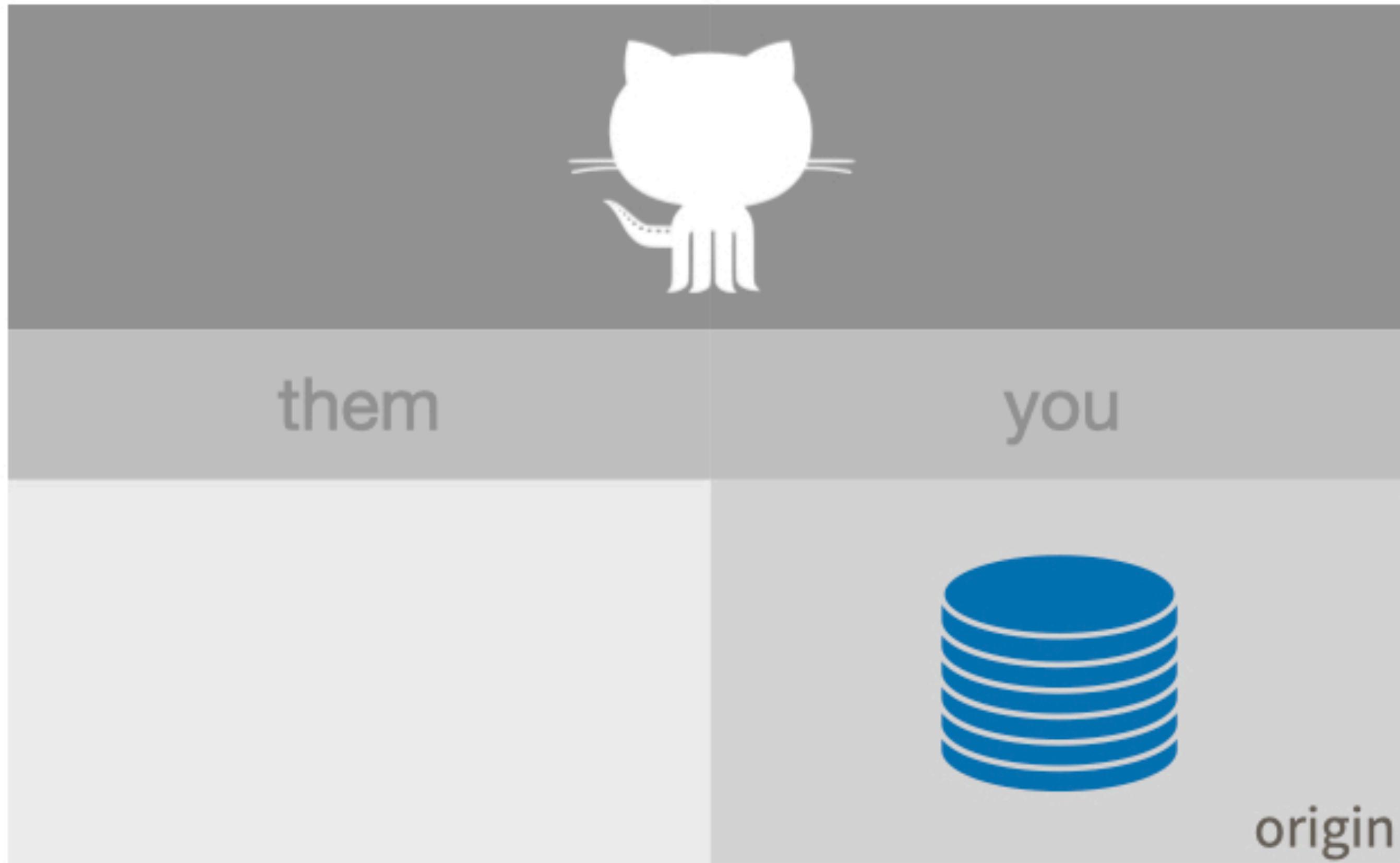


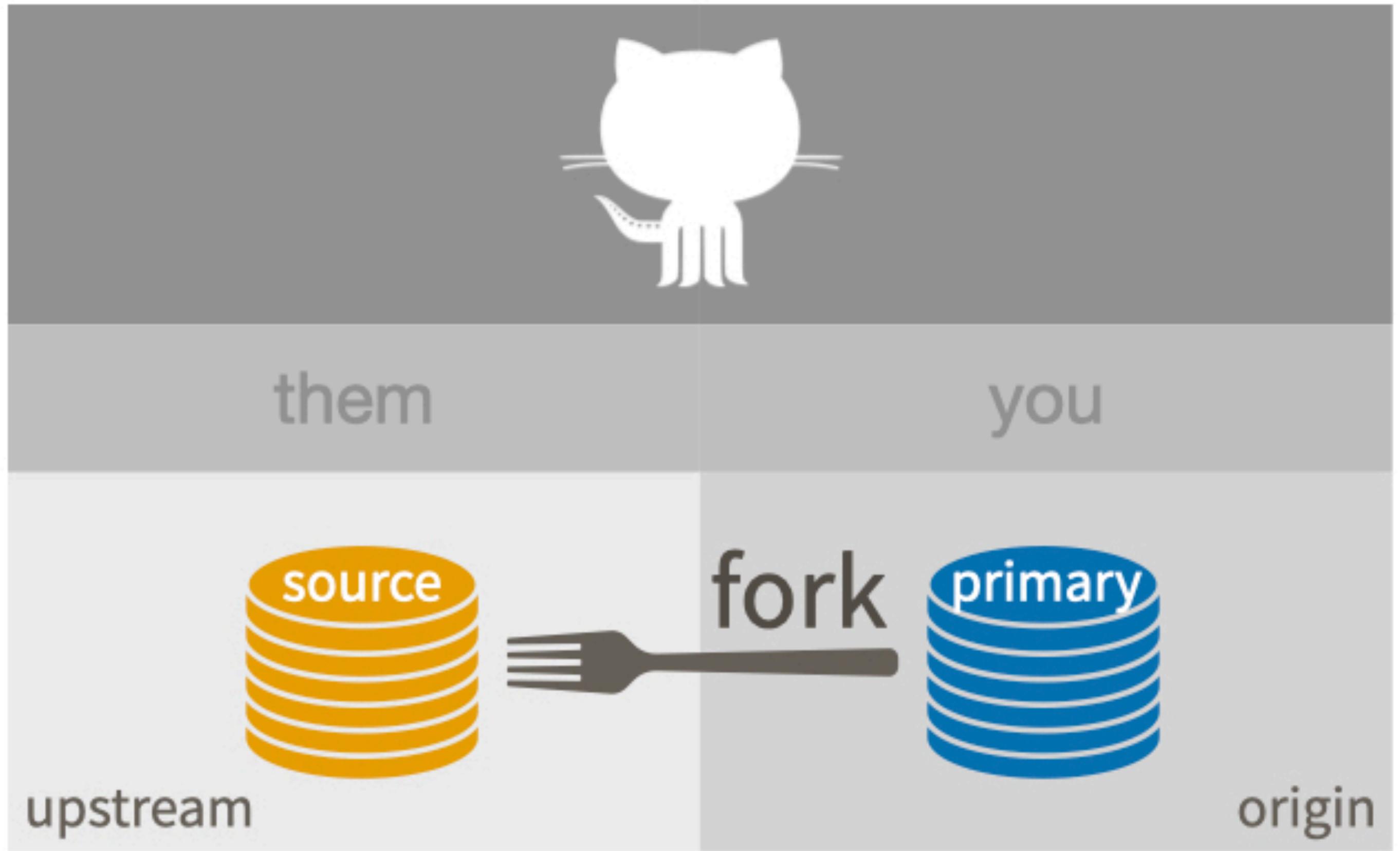
# Workflows for project initiation

- **Clone**: Make a local project from a remote one.
- Fork: Make a remote copy of another repo.
- **Fork and clone**: Make a remote copy of another repo. Then make a local project from that.
- Make a remote repo from a local project.

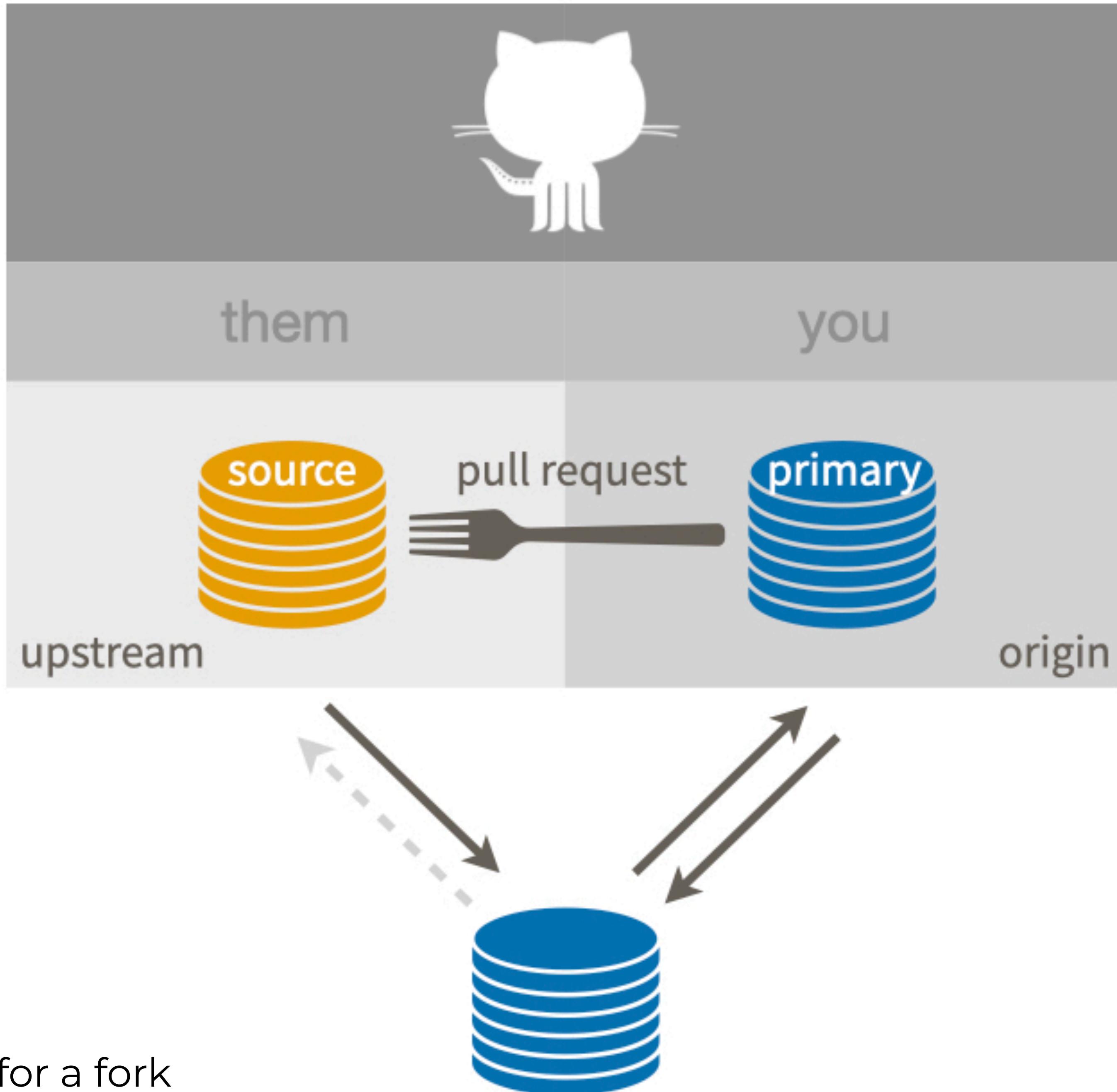


In Happy Git:  
New project, GitHub first  
Existing project, GitHub first

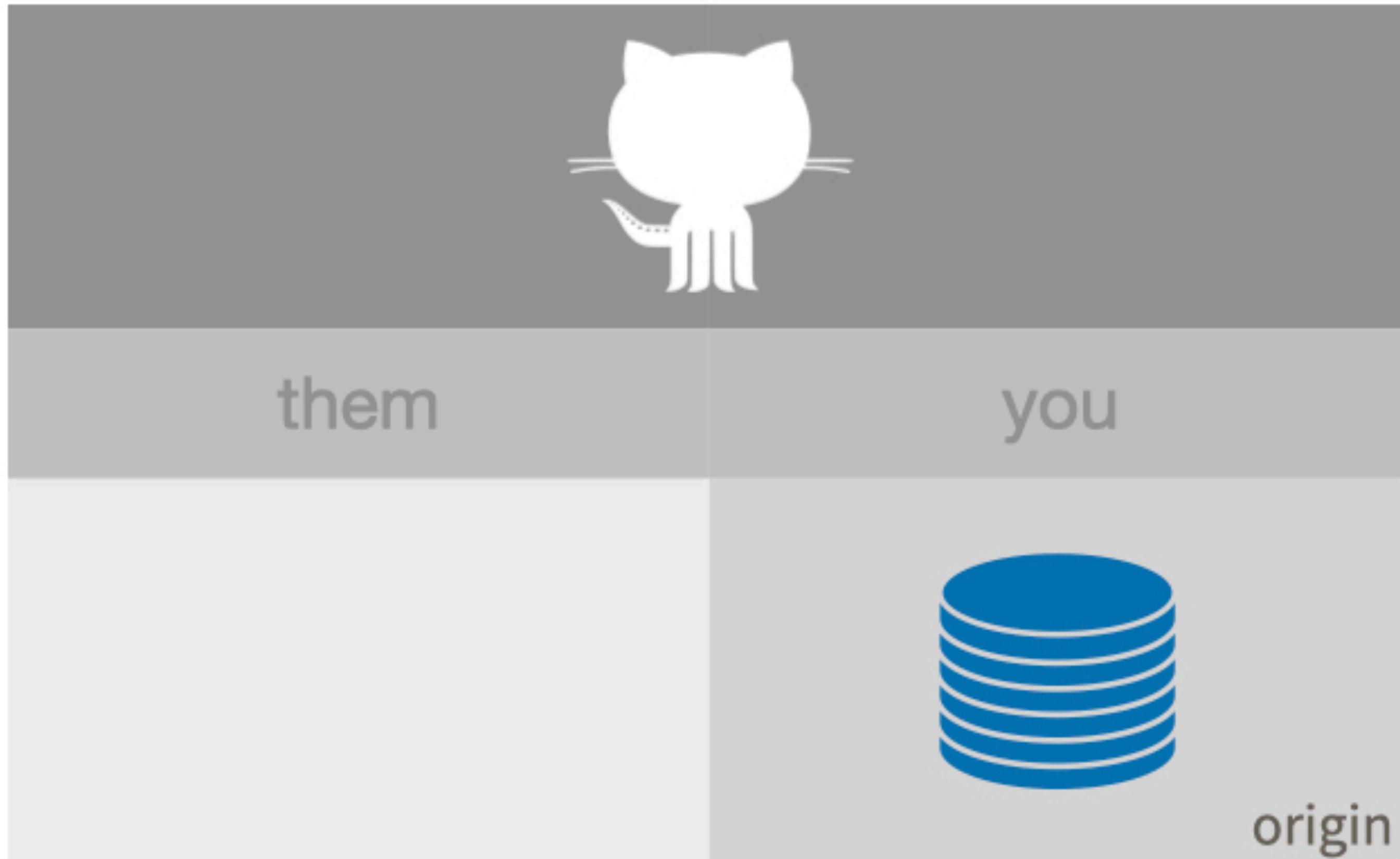




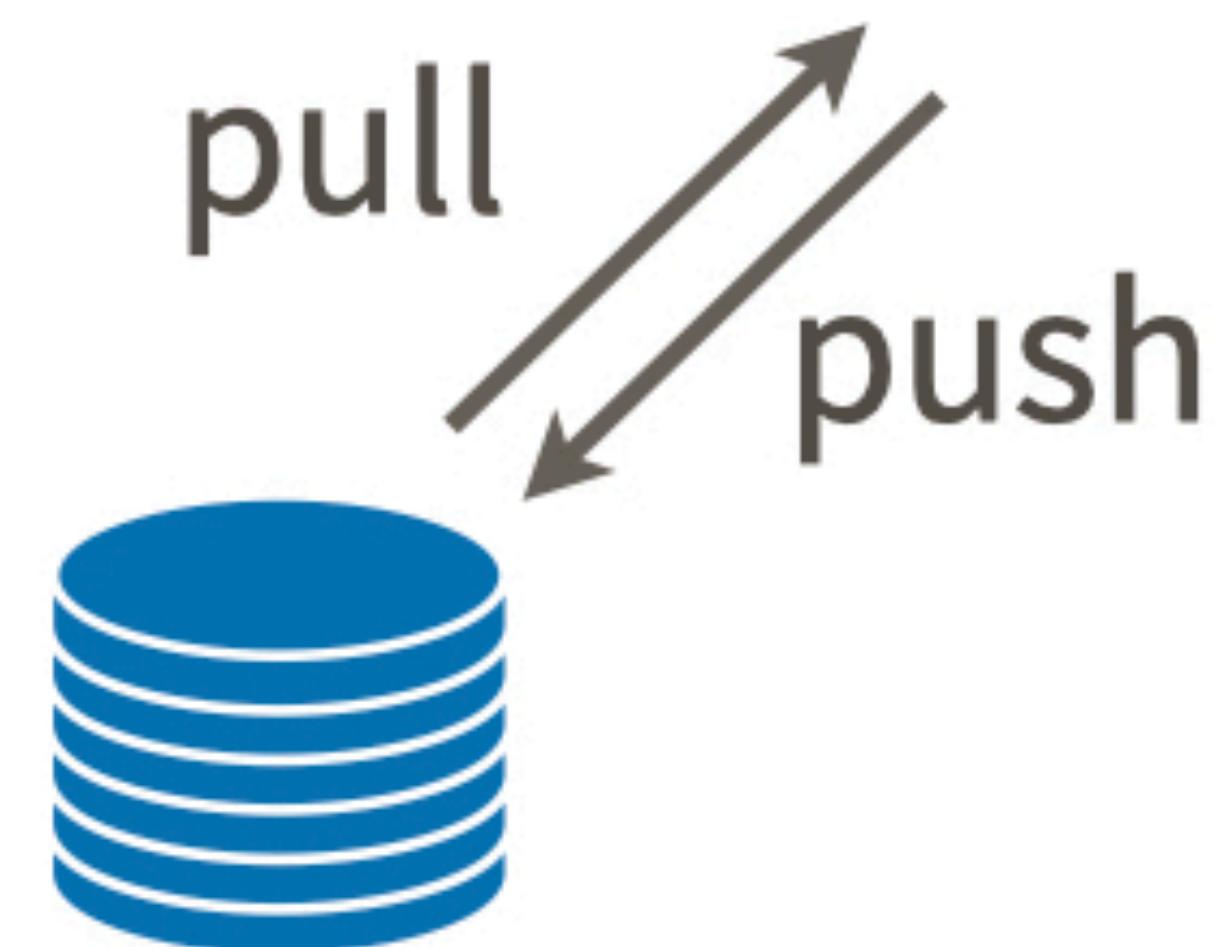
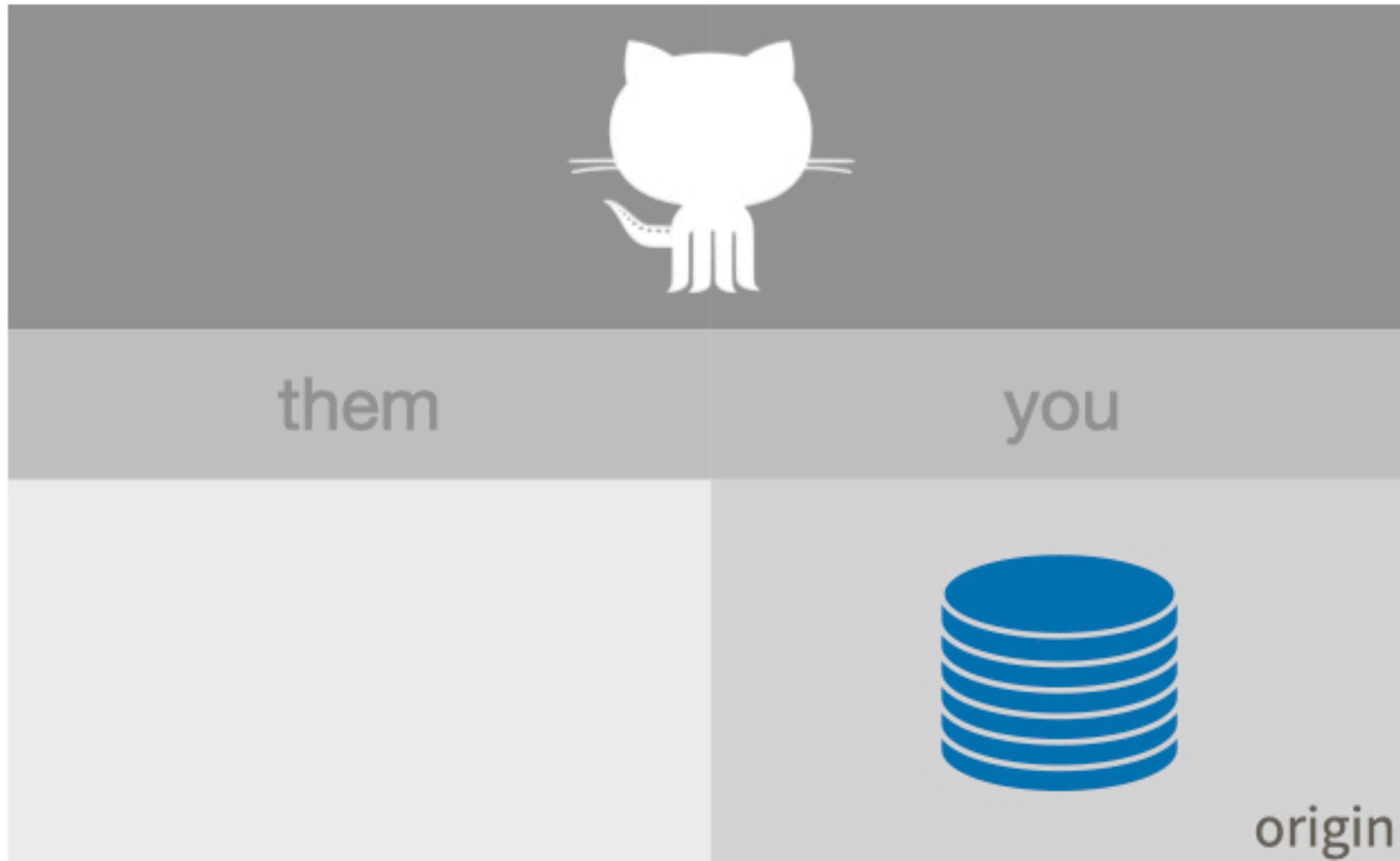
In Happy Git:  
Fork and clone



In Happy Git:  
Get upstream changes for a fork



In Happy Git:  
Existing project, GitHub last



# GitHub Personal access token

- We're going verify you've got a PAT set up.
- <https://happygitwithr.com/https-pat.html>
- Key commands
  - `usethis::gh_token_help()`
  - `usethis::git_sitrep()`
  - `usethis::create_github_token()`
  - `gitcreds::gitcreds_set()`

## git operations via ssh

example `git clone git@github.com:OWNER/REPO.git`

creds local private ssh key + public key on GitHub

## git operations via https

example `git clone https://github.com/OWNER/REPO.git`

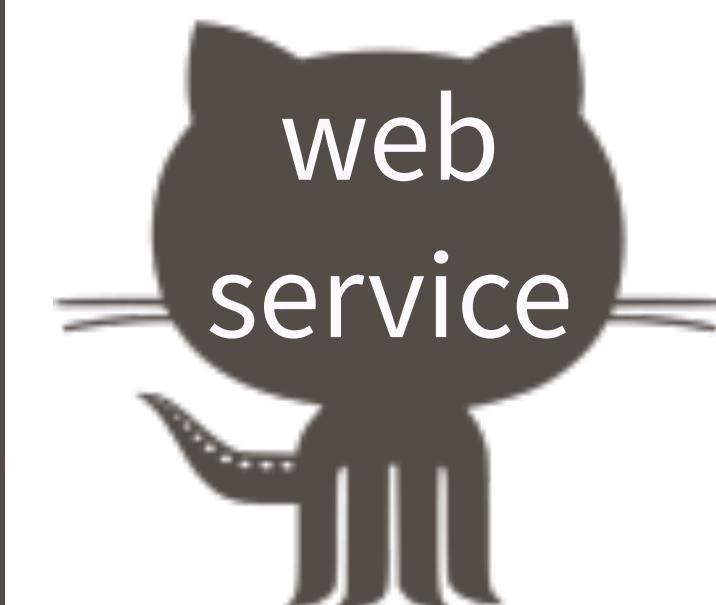
creds username + password (password can be **GITHUB\_PAT**)



## GitHub API requests via REST

example `curl -H "Authorization: token $GITHUB_PAT" https://api.github.com/user/repos`

creds **GITHUB\_PAT**



# "New project, GitHub first"

- We'll walk through this together.
- <http://happygitwithr.com/new-github-first.html>
- I suggest
  - repo / Project / folder name = "packages-report"
  - locate as sibling to any folders/Projects created earlier

# Building on "New project, GitHub first"

- Create a new .R file in the local repo.
- Use a little bit of code developed earlier today.
- Notice what's changed in the Git pane, inspect the diff, stage the file, commit, push.
- Verify the new .R file is now on GitHub.
- Wait ... is a .R file really all I want to share?

what you need to write

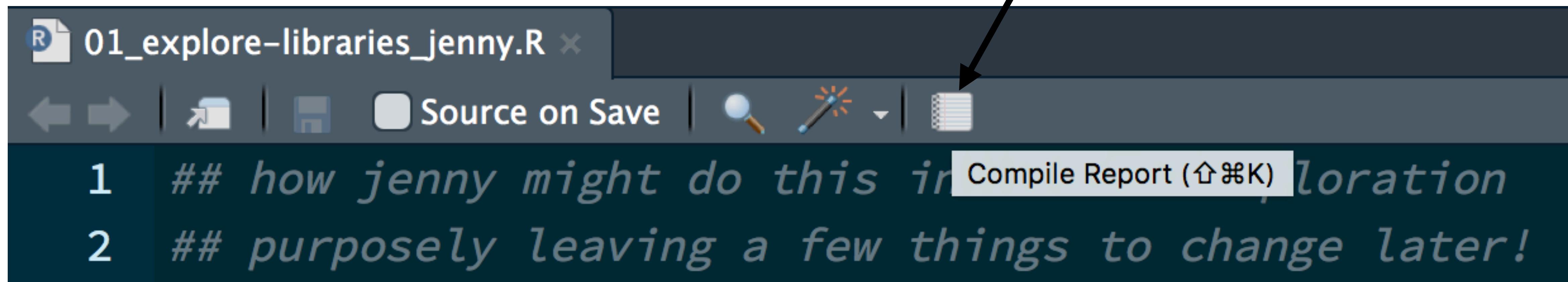
foo.R  
foo.Rmd



what people like to read

foo.md  
foo.html

## Compile Report



A screenshot of the RStudio interface. The top bar shows a file named "01\_explore-libraries\_jenny.R". Below the top bar are several icons: back, forward, file, source on save (which is checked), search, and compile report. A red arrow points from the word "Compile Report" in the title to the icon in the toolbar. The main code editor area contains two lines of R code:

```
1 ## how jenny might do this in exploration  
2 ## purposely leaving a few things to change later!
```

```
≈ rmarkdown::render("whatever.R")
```

## Compile Report from R Script

Create a standalone report that contains the code and output from your R script.

For more information on compiling reports, see the documentation at [Compiling Reports from R Scripts](#)

Report output format:

HTML

Compile

Cancel

Sure, HTML is fine ... for now.



What changed in Git pane?

Inspect the diff. Or not.

Stage.

Commit.

Push.

Verify the .html file is now on GitHub.

Wait ... is .html immediately useful on GitHub?

248 lines (201 sloc) | 723 KB

Raw

Blame

History



```
1 <!DOCTYPE html>
2
3 <html xmlns="http://www.w3.org/1999/xhtml">
4
5 <head>
6
7 <meta charset="utf-8" />
8 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9 <meta name="generator" content="pandoc" />
10
11
12 <meta name="author" content="jenny" />
13
14
15 <title>01_explore-libraries_jenny.R</title>
16
17
18 <meta name="viewport" content="width=device-width, initial-scale=1" />
19
20 <script src="data:application/x-javascript;base64,LyohCiAqIEJvb3RzdHJhcCB2My4zLjUgKGh0dHA6Ly9nZXRib290c3RyYXAuY29tKQogKiBDb3B!
21 <script src="data:application/x-javascript;base64,LyoqCiogQHByZXNlcnZlIEhUTUw1IFNoaXYgMy43LjIgfCBAYWZhcmthcyBAamRhRvbIBAam9i
22 <script src="data:application/x-javascript;base64,LyohIFJlc3BvbQuanMgdjEuNC4y0iBtaW4vbWF4LXdpxHRoIG1lZGlhIHf1ZXJ5IHBvbHlmaWx
23 <script src="data:application/x-javascript;base64,CgovKioKICogalF1ZXJ5IFBsdWdpbjogU3RpY2t5IFRhYnMKICoKICogQGF1dGhvcIBBaWRhbIB
24 <link href="data:text/css;charset=utf-8,%2Ehljs%2Dliteral%20%7B%0Acolor%3A%20%23990073%3B%0A%7D%0A%2Ehljs%2Dnumber%20%7B%0Aco
--
```



**NO, raw .html **is NOT immediately useful**\* on GitHub.**

**But Markdown = .md **is useful.****

**Let's render .R to .md instead of .html!**

\* it obviously is useful in actual web publishing workflows

`foo.R` → `foo.html`

```
#' ---  
#' output: html_document  
#' ---
```

`foo.R` → `foo.md` → `foo.html`

```
#' ---  
#' output:  
#'   html_document:  
#'     keep_md: yes  
#' ---
```

`foo.R` → `foo.md`

```
#' ---  
#' output: md_document  
#' ---
```

`foo.R` → `foo.md`

```
#' ---  
#' output: github_document  
#' ---
```

```
#' ----  
#' output: github_document  
#' ----
```

Add this YAML frontmatter to your .R file.

Re-Compile Notebook

What changed? Look at the Git diff.

This is what I mean by "explore cause and effect" and "experiment without fear".

Get comfortable with the diff.  
Stage.  
Commit.  
Push.  
Verify the .md file is now on GitHub.

Revel in how nice the .md looks!

# 01\_explore-libraries\_jenny.R

jenny Sat Jan 27 22:46:07 2018

```
## how jenny might do this in a first exploration  
## purposely leaving a few things to change later!
```

Which libraries does R search for packages?

```
.libPaths()
```

```
## [1] "/Users/jenny/resources/R/library"  
## [2] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library"
```

```
## let's confirm the second element is, in fact, the default library  
.Library
```

```
## [1] "/Library/Frameworks/R.framework/Resources/library"
```

This is what I mean by "expose your work".

# Take away #1

- It is absolutely OK to track rendered or derived products in Git and push them to GitHub. Often it's a good idea!
- Just because someone can fork, clone, install all necessary packages, then run your code, it doesn't mean they want to or will.
- Be humane. Be realistic.

# Take away #2

- For consumption on GitHub, Markdown (.md) is vastly more useful than .html, .docx, .pdf, etc.
- Binary formats like .docx and .pdf are also a reliable source of merge conflicts. Think carefully before you track them with Git.

# How to think about files + Git

- Not all file types play equally well with Git. Plain text works best.
  - "Excuse Me, ..." article has a section on "Which files to commit"
- Not all file types play equally well with GitHub. Markdown is especially awesome.
  - <https://happygitwithr.com/workflows-browsability.html>

# Keep doing "Existing project, GitHub first"

- Continue to port your earlier work on library exploration into your new Git/GitHub repo. Or bring an example solution over.
- Make lots of small additions and changes.
- Play with rendering to markdown.
- Look at diffs, stage, commit, push, verify.
- This is what I mean by "embrace incrementalism".

# Why use version control?

- experiment without fear
- explore cause and effect
- embrace incrementalism
- expose your work
- collaborate

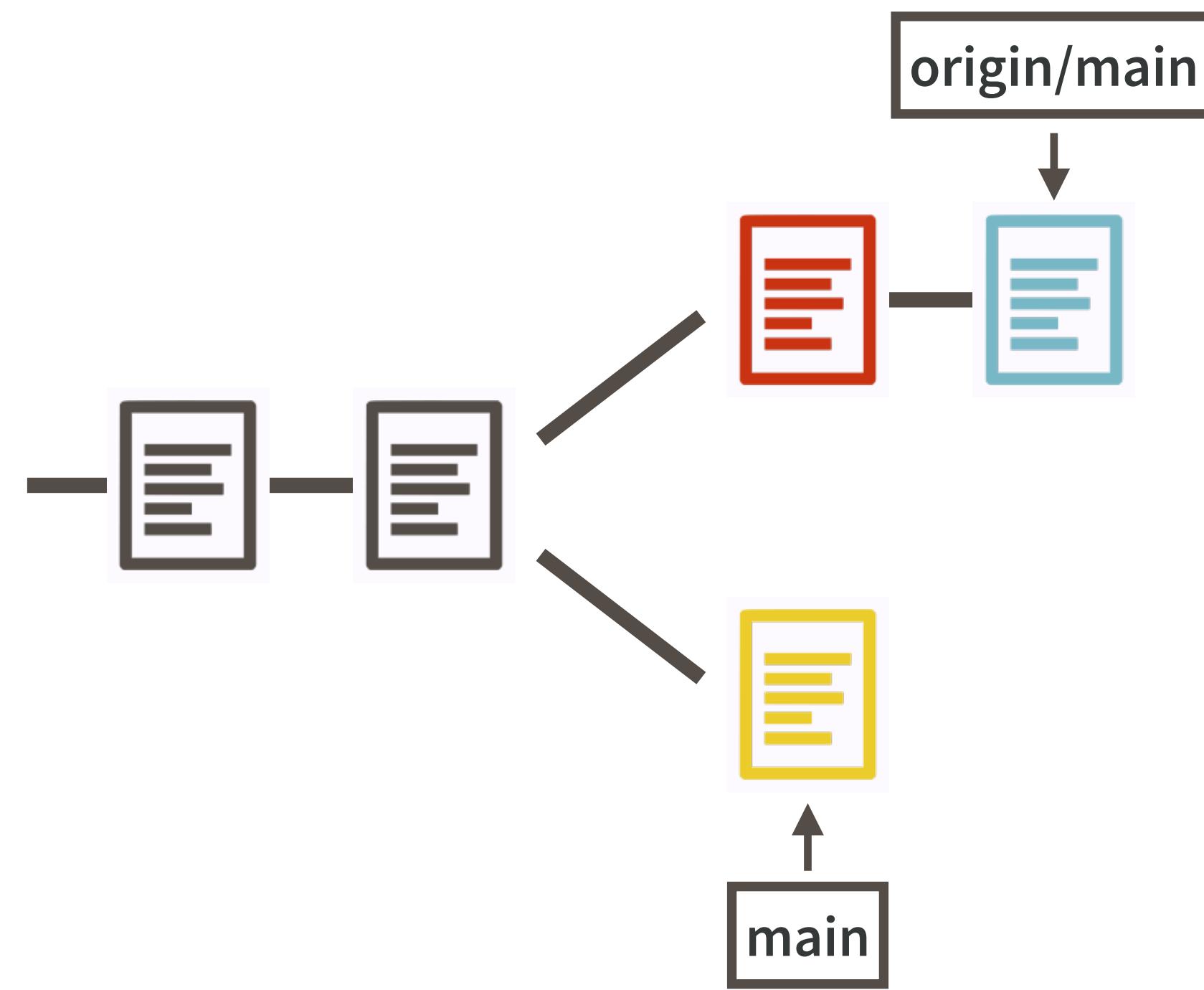


let's practice dealing with  
unpleasant situations

# Dealing with push rejection

- A push attempt will fail if your local commit history is incompatible with that on the remote.

```
~/rrr/rstats-wtf/wtf-repos/wtf-ascii-funtimes % git push origin  
To https://github.com/jennybc/wtf-ascii-funtimes.git  
! [rejected]      main -> main (non-fast-forward)  
error: failed to push some refs to 'https://github.com/jennybc/wtf-ascii-funtimes.git'  
hint: Updates were rejected because the tip of your current branch is behind  
hint: its remote counterpart. Integrate the remote changes (e.g.  
hint: 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```



git push is rejected.

You must git pull first.

```
~/rrr/rstats-wtf/wtf-repos/wtf-ascii-funtimes % git push origin  
To https://github.com/jennybc/wtf-ascii-funtimes.git  
 ! [rejected]      main -> main (non-fast-forward)  
error: failed to push some refs to 'https://github.com/jennybc/wtf-ascii-funtimes.git'  
hint: Updates were rejected because the tip of your current branch is behind  
hint: its remote counterpart. Integrate the remote changes (e.g.  
hint: 'git pull ...') before pushing again.  
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

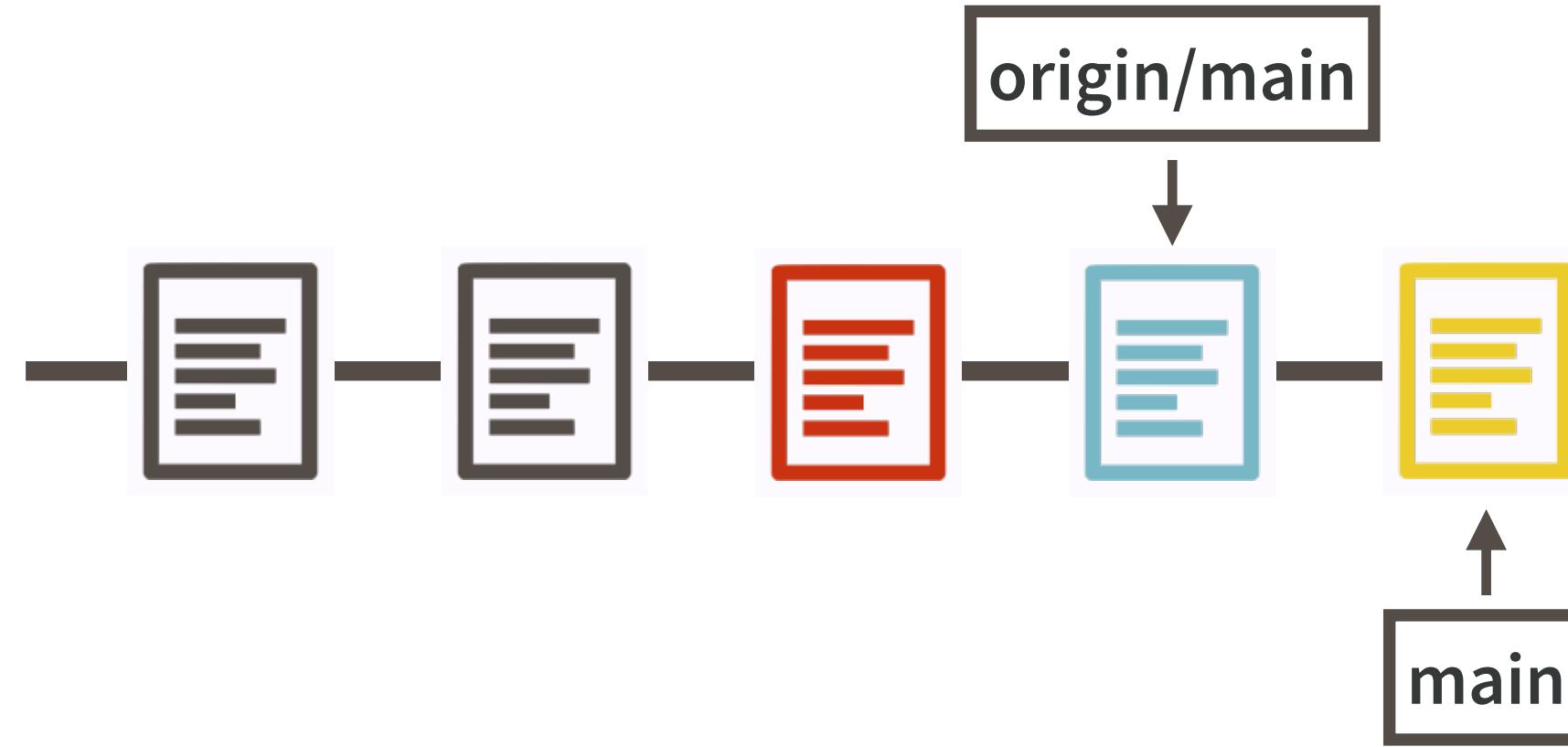
At this point, we (or at least I) will create the troublesome situation.

How?

Make a commit on GitHub, via the browser.  
(Don't pull.)

Make a local commit.

Try to push.



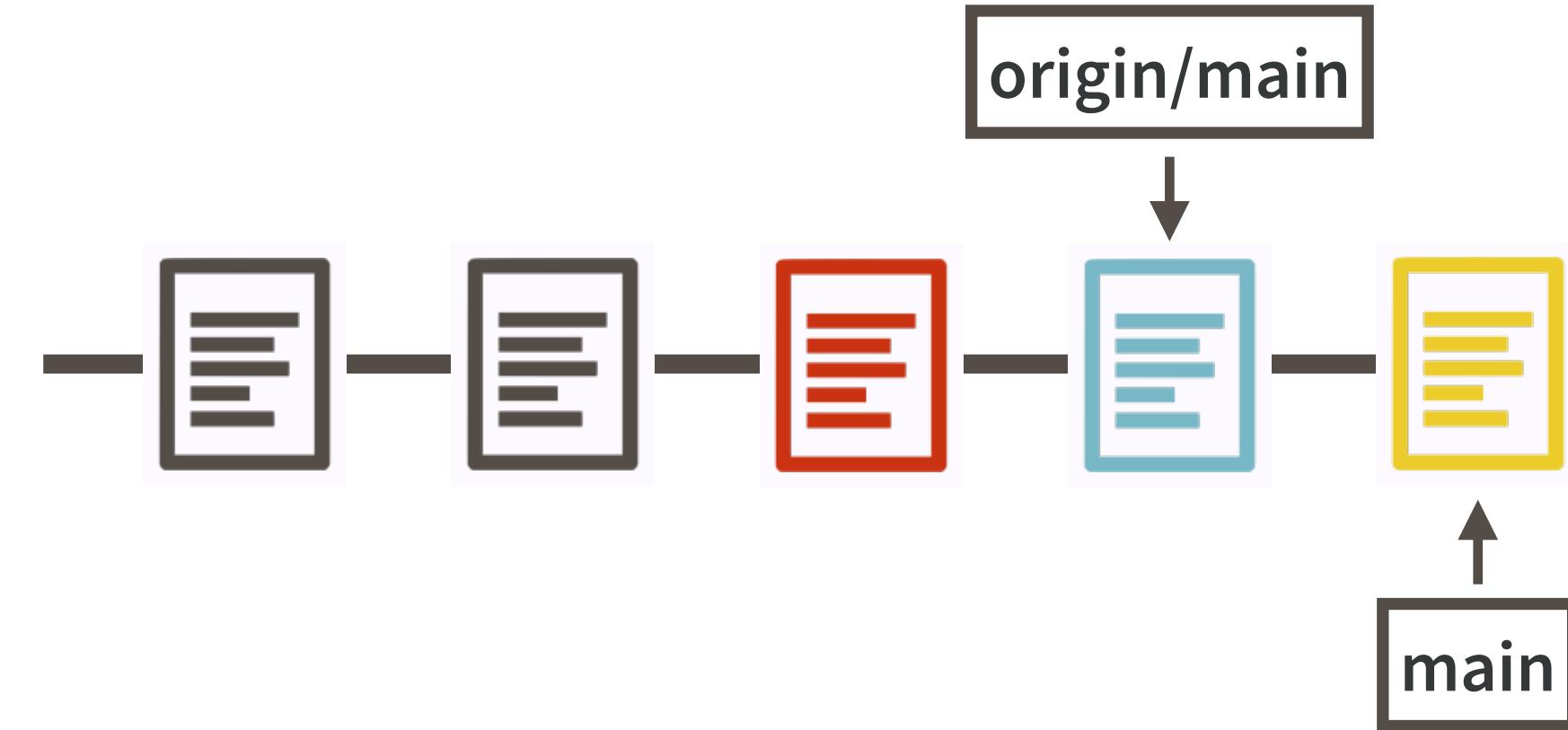
My preferred resolution.  
I love a linear history.

Not always possible, but often is.  
Depends on the situation, i.e. what's in the diffs.

Achieved by a "pull with rebase".

One-off invocation:

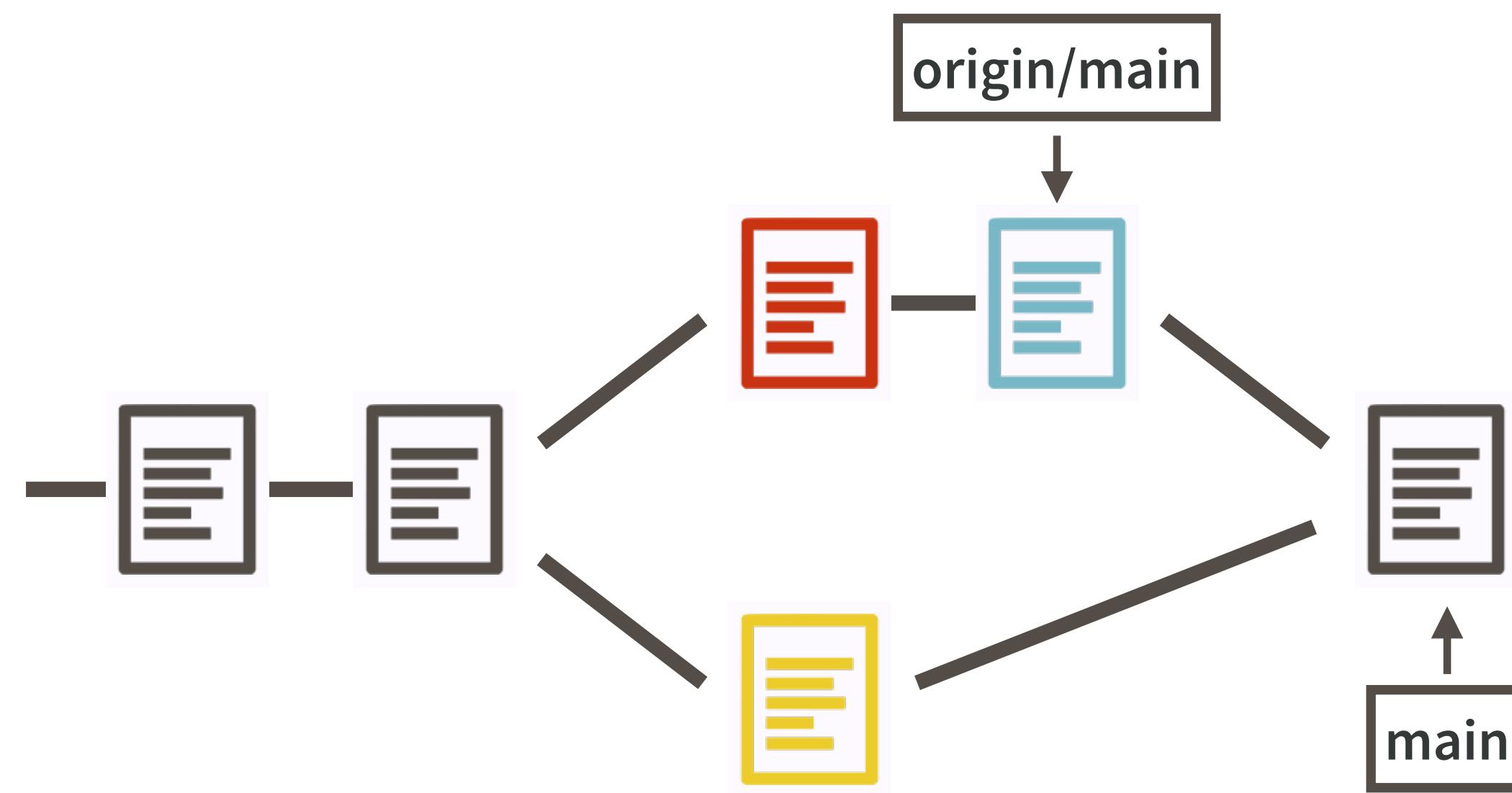
```
git pull --rebase
```



I recommend "pull with rebase" as a lifestyle.  
IMO it should be the default.

Opt-in like so:

```
git config --global pull.rebase true
```

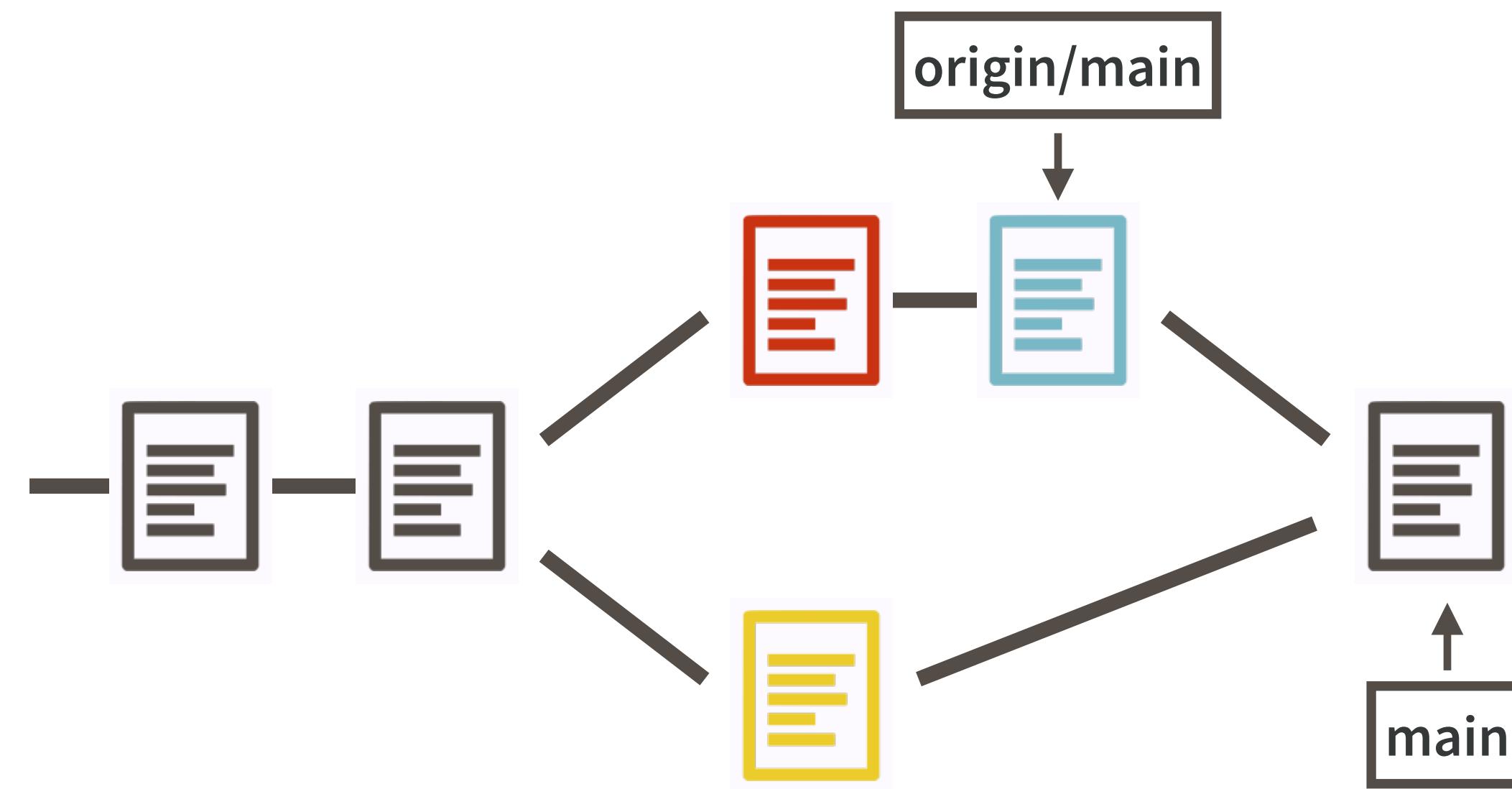


An acceptable outcome. (But I don't care for it.)

Resolution via a merge commit.

(Sort of) the default behaviour, but recent versions of Git will complain and encourage you to be more explicit.

Also, awkwardness about generating the commit message.

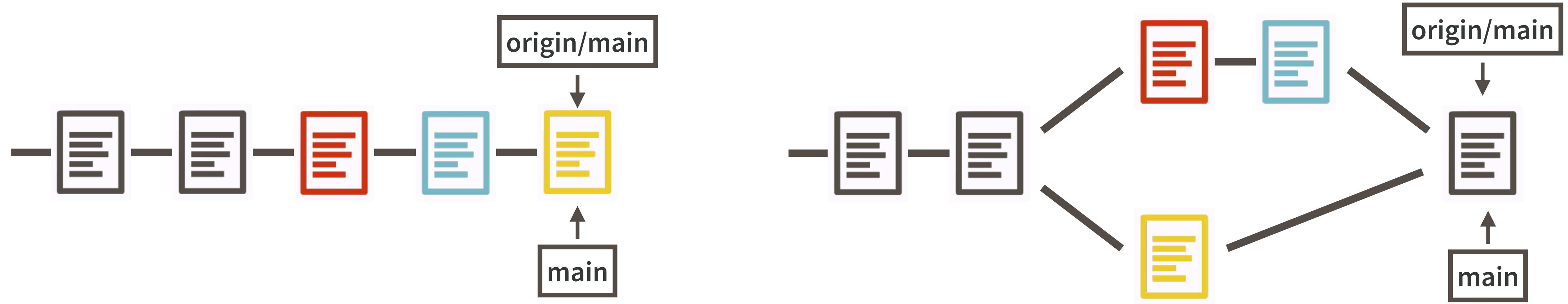


If you're in the middle of this and you want to back out:

```
git merge --abort
```

If you want to complete the merge:

```
git commit -m 'Merge commit'
```



No matter what, don't forget to `git push`!

You want to get back to a happy place where local and remote are synced up.

Read more in Happy Git:

Dealing with push rejection

Pull, but you have local work

# Dealing with a merge conflict

- A pull attempt will fail if Git can't figure out how to do the rebase or merge.

```
From github.com:jennybc/bunny-scarf
  958548f..3357952  master      -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

If you simply cannot deal with the mess right now, back out and come back later.

If it's a "pull with rebase":

```
git rebase --abort
```

If it's a "fetch and merge":

```
git merge --abort
```

We (or at least I) am going to soldier on and resolve the conflict.

# Every locus of conflict looks like this.

How things look locally (or on the currently checked out branch).

How things look remotely (or on the branch you're merging).

```
<<<<< HEAD
Wingardium Leviosaaaaaaa
=====
Wing-GAR-dium Levi-0-sa
>>>>> 3357952
```

\*conflict markers appear in orange

At each locus of conflict, you must form a consensus state and remove the conflict marker lines.

```
<<<<< HEAD  
Wingardium Leviosaaaaaaaa  
===== → Wingardium Levi-0-sa  
Wing-GAR-dium Levi-0-sa  
>>>>> 3357952
```

Stage and commit all affected files.  
Commit.  
Push.  
Carry on.

\*conflict markers appear in orange

# Powerful Git clients, like GitKraken, offer support for resolving merge conflicts.

