```
                     Reminders

Upcoming due dates
Mon Jan 12th Quiz 1
Wed Jan 14th Pre-course survey
Fri Jan 16th Discussion Lab 1
             #FinAid quiz on Canvas

Discussion section this week is a tour
of Datahub and an ungraded Python
review notebook
```

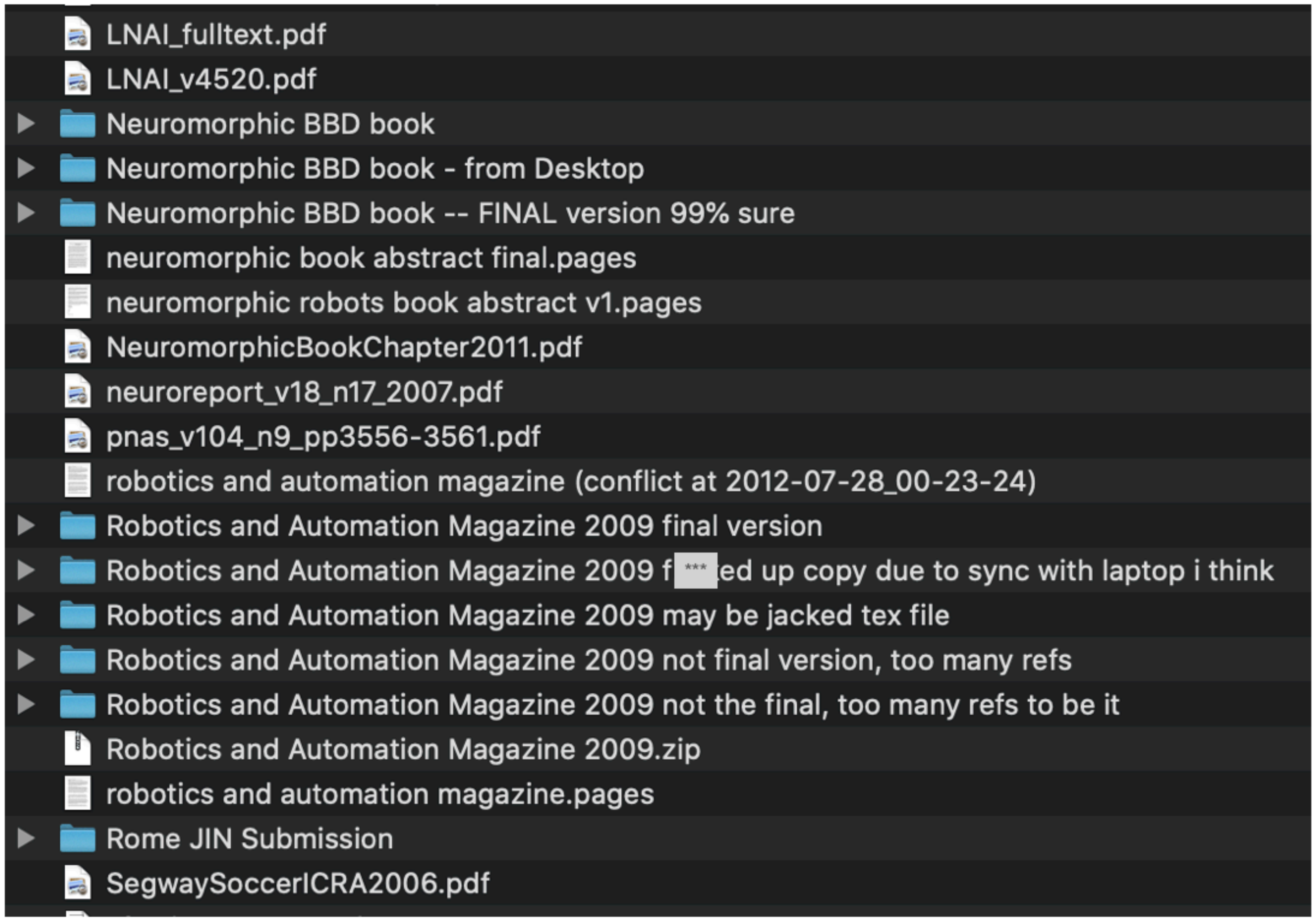
# Version Control 1.0

## Data Science in Practice

**Jason G. Fleischer, PhD** **Dept. of Cognitive Science** **UC San Diego** **https://jgfleischer.com**

# This sucks

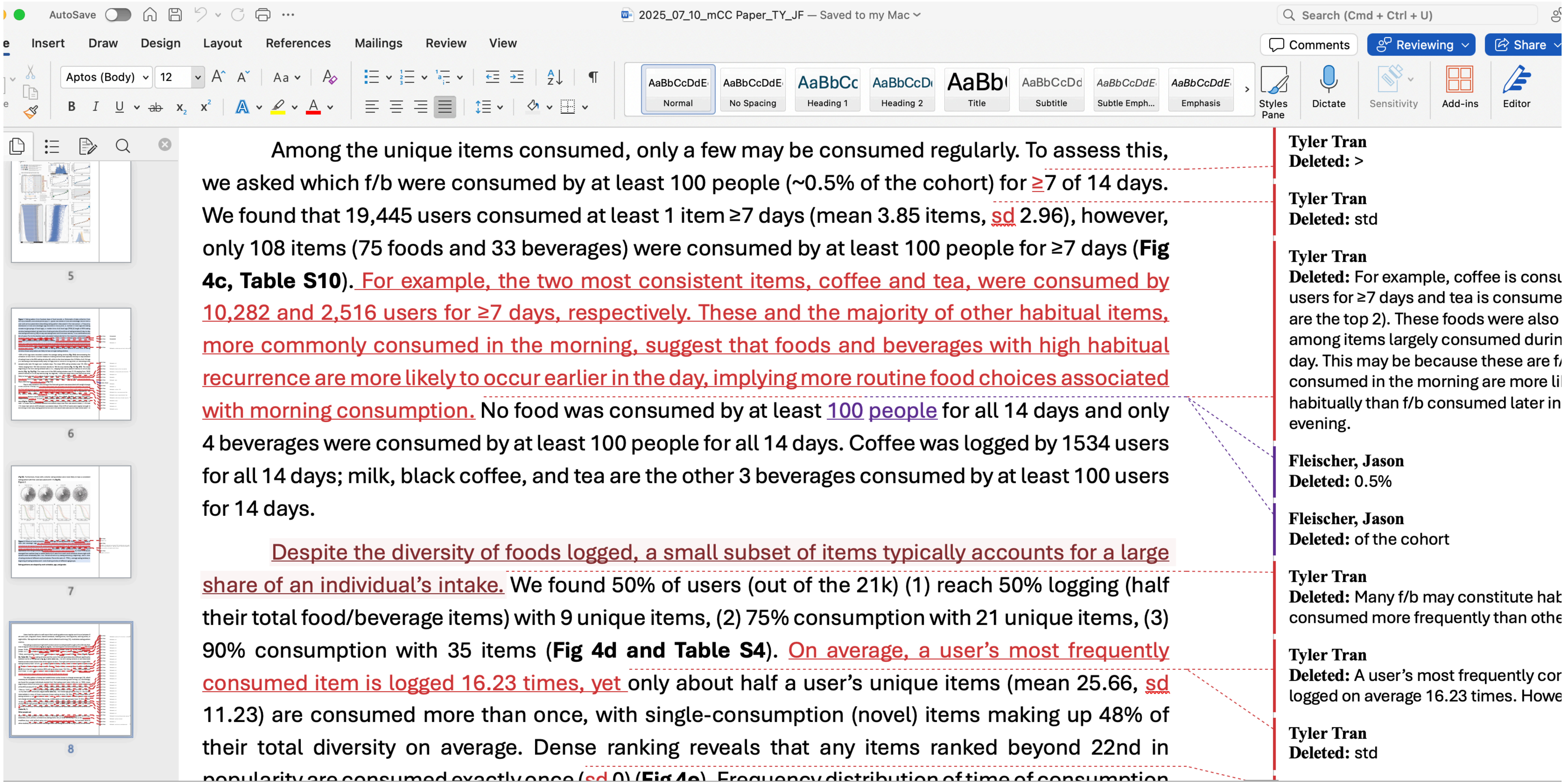
archived version of my Documents folder from ~ 2012



LNAI_fulltext.pdf
LNAI_v4520.pdf
Neuromorphic BBD book
Neuromorphic BBD book - from Desktop
Neuromorphic BBD book -- FINAL version 99% sure
neuromorphic book abstract final.pages
neuromorphic robots book abstract v1.pages
NeuromorphicBookChapter2011.pdf
neuroreport_v18_n17_2007.pdf
pnas_v104_n9_pp3556-3561.pdf
robotics and automation magazine (conflict at 2012-07-28_00-23-24)
Robotics and Automation Magazine 2009 final version
Robotics and Automation Magazine 2009 f***ed up copy due to sync with laptop i think
Robotics and Automation Magazine 2009 may be jacked tex file
Robotics and Automation Magazine 2009 not final version, too many refs
Robotics and Automation Magazine 2009 not the final, too many refs to be it
Robotics and Automation Magazine 2009.zip
robotics and automation magazine.pages
Rome JIN Submission
SegwaySoccerICRA2006.pdf

Several months after finishing a writing project, I wanted to keep only the final version of the many different revisions… figuring out which one was the version actually sent to the publisher was hard!

# A step in the right direction



Among the unique items consumed, only a few may be consumed regularly. To assess this, we asked which f/b were consumed by at least 100 people (~0.5% of the cohort) for ≥7 of 14 days. We found that 19,445 users consumed at least 1 item ≥7 days (mean 3.85 items, sd 2.96), however, only 108 items (75 foods and 33 beverages) were consumed by at least 100 people for ≥7 days (**Fig 4c, Table S10**). For example, the two most consistent items, coffee and tea, were consumed by 10,282 and 2,516 users for ≥7 days, respectively. These and the majority of other habitual items, more commonly consumed in the morning, suggest that foods and beverages with high habitual recurrence are more likely to occur earlier in the day, implying more routine food choices associated with morning consumption. No food was consumed by at least 100 people for all 14 days and only 4 beverages were consumed by at least 100 people for all 14 days. Coffee was logged by 1534 users for all 14 days; milk, black coffee, and tea are the other 3 beverages consumed by at least 100 users for 14 days.

Despite the diversity of foods logged, a small subset of items typically accounts for a large share of an individual's intake. We found 50% of users (out of the 21k) (1) reach 50% logging (half their total food/beverage items) with 9 unique items, (2) 75% consumption with 21 unique items, (3) 90% consumption with 35 items (**Fig 4d and Table S4**). On average, a user's most frequently consumed item is logged 16.23 times, yet only about half a user's unique items (mean 25.66, sd 11.23) are consumed more than once, with single-consumption (novel) items making up 48% of their total diversity on average. Dense ranking reveals that any items ranked beyond 22nd in popularity are consumed exactly once (sd 0) (Fig 4e). Frequency distribution of time of consumption

**Tyler Tran**
**Deleted:** >

**Tyler Tran**
**Deleted:** std

**Tyler Tran**
**Deleted:** For example, coffee is consu
users for ≥7 days and tea is consume
are the top 2). These foods were also
among items largely consumed durin
day. This may be because these are f
consumed in the morning are more li
habitually than f/b consumed later in
evening.

**Fleischer, Jason**
**Deleted:** 0.5%

**Fleischer, Jason**
**Deleted:** of the cohort

**Tyler Tran**
**Deleted:** Many f/b may constitute hab
consumed more frequently than othe

**Tyler Tran**
**Deleted:** A user's most frequently co
logged on average 16.23 times. Howe

**Tyler Tran**
**Deleted:** std

## Version Control

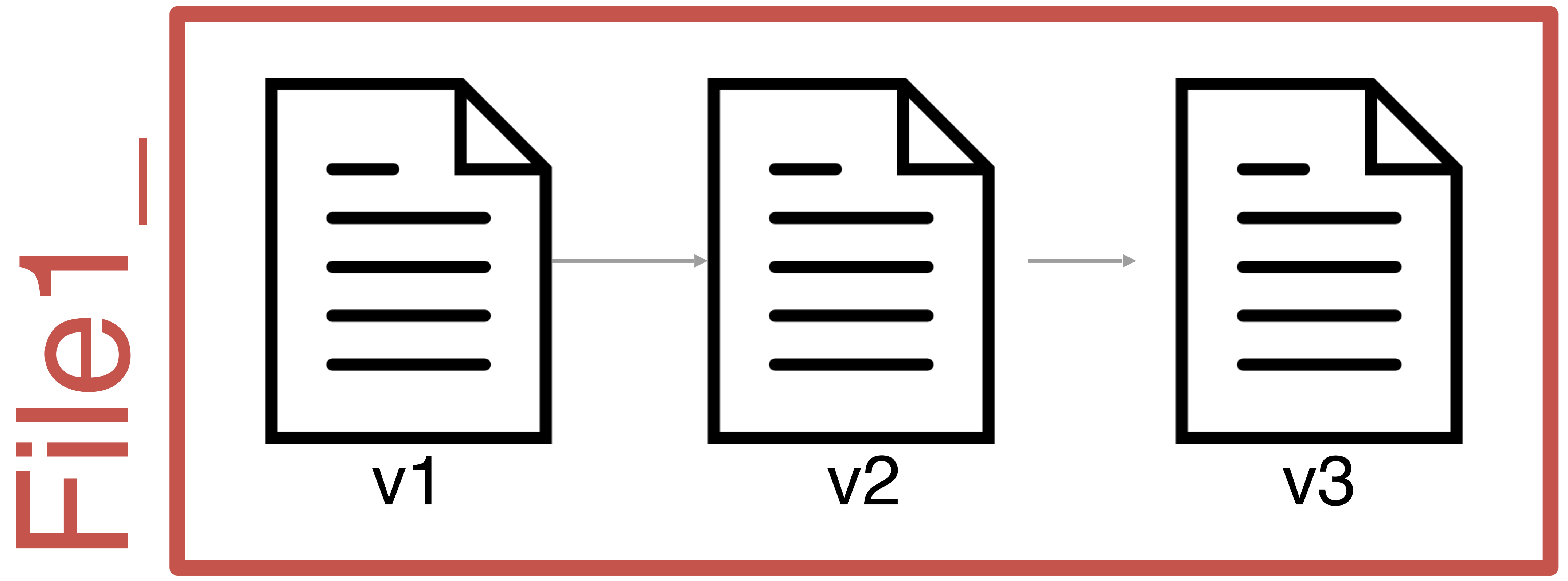
- Enables multiple people to simultaneously work on a single project.
- Each person edits their own copy of the files and chooses when to share those changes with the rest of the team.
- Thus, temporary or partial edits by one person do not interfere with another person's work

## What is version control?

A way to manage the evolution of a set of files

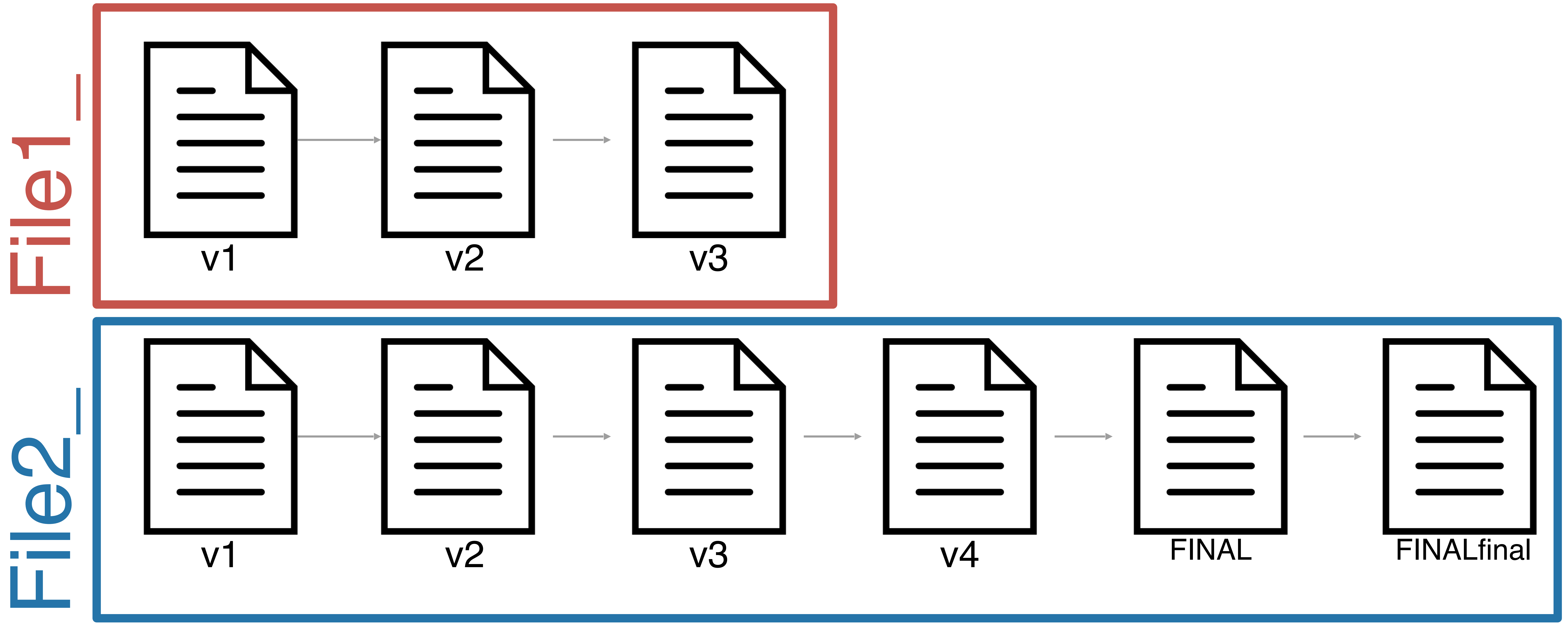
## What is version control?

A way to manage the evolution of a set of files



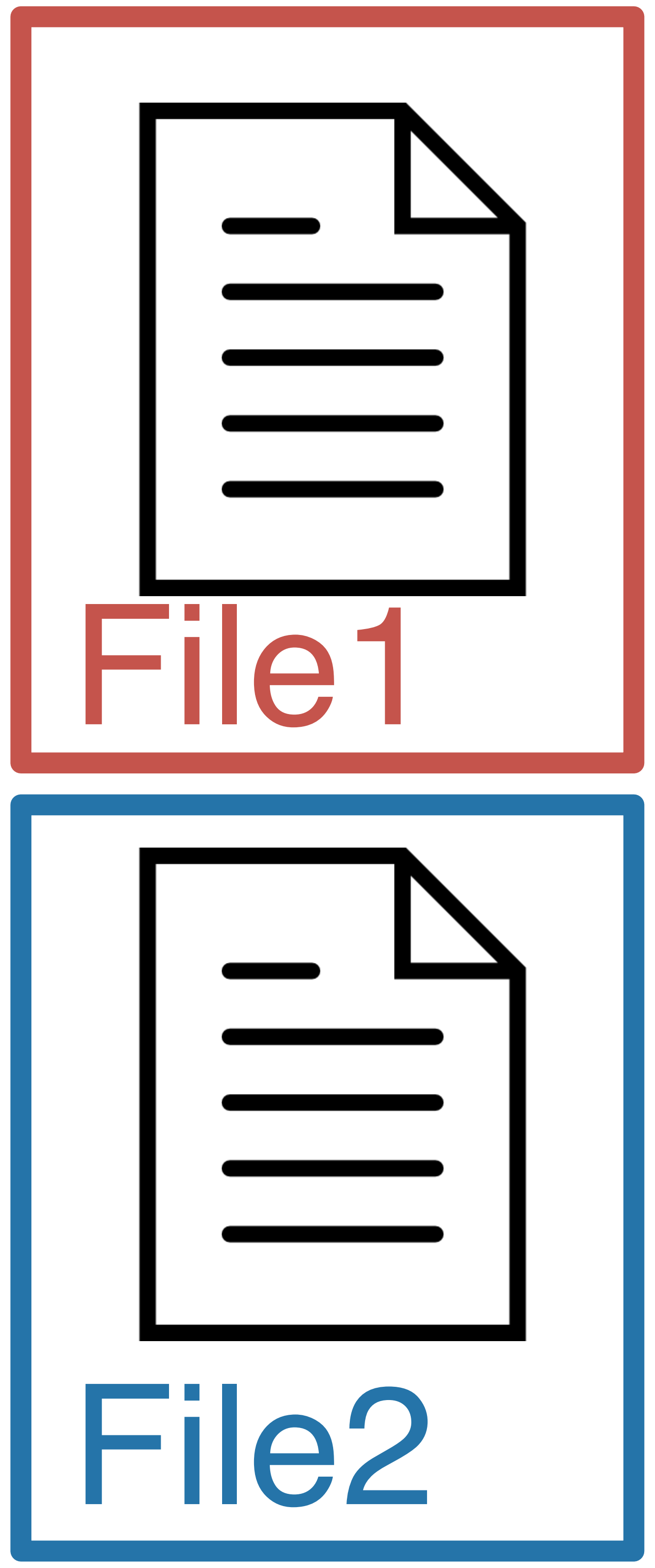
## What is version control?

A way to manage the evolution of a set of files



## What is version control?

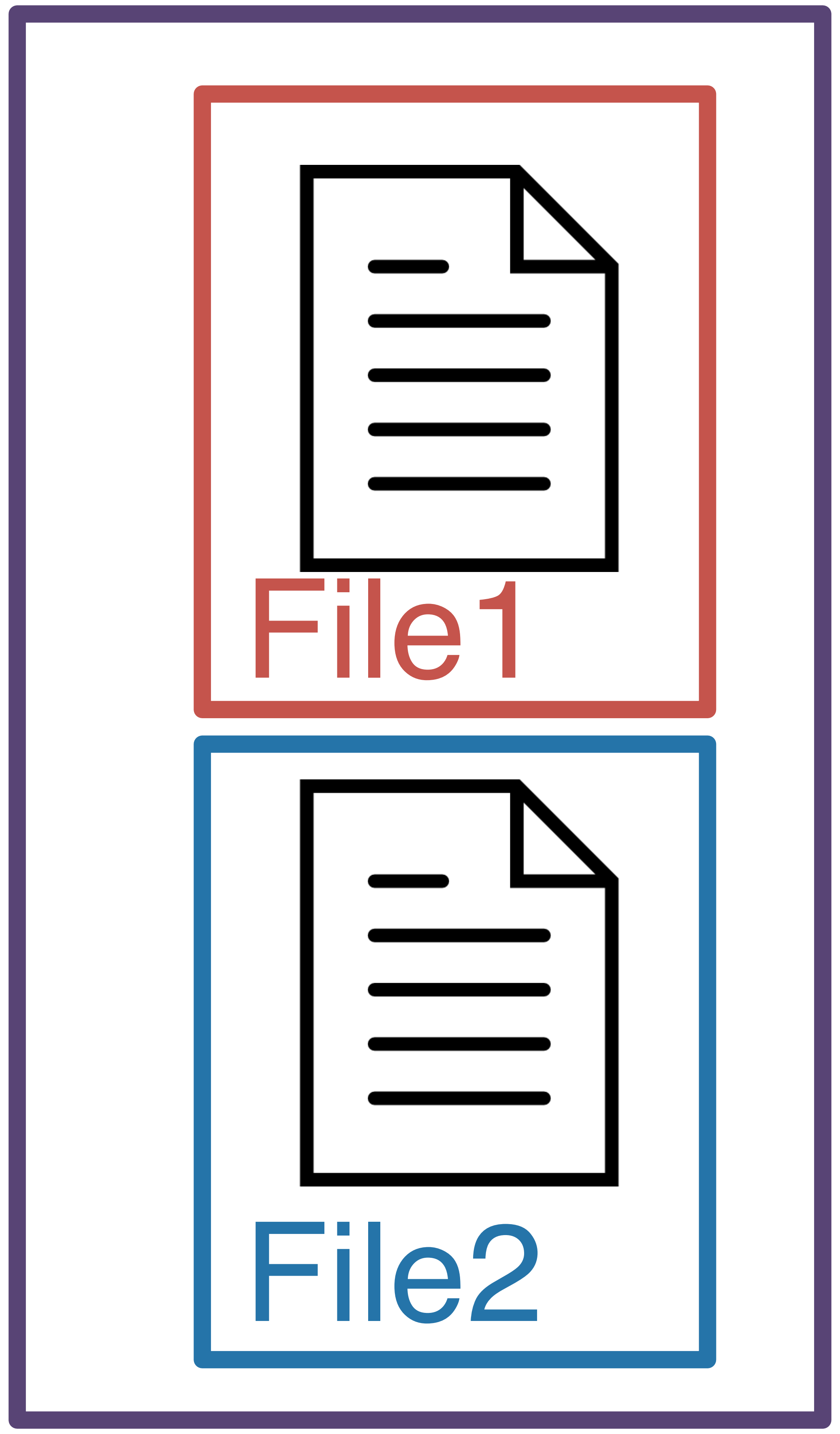
A way to manage the evolution of a set of files



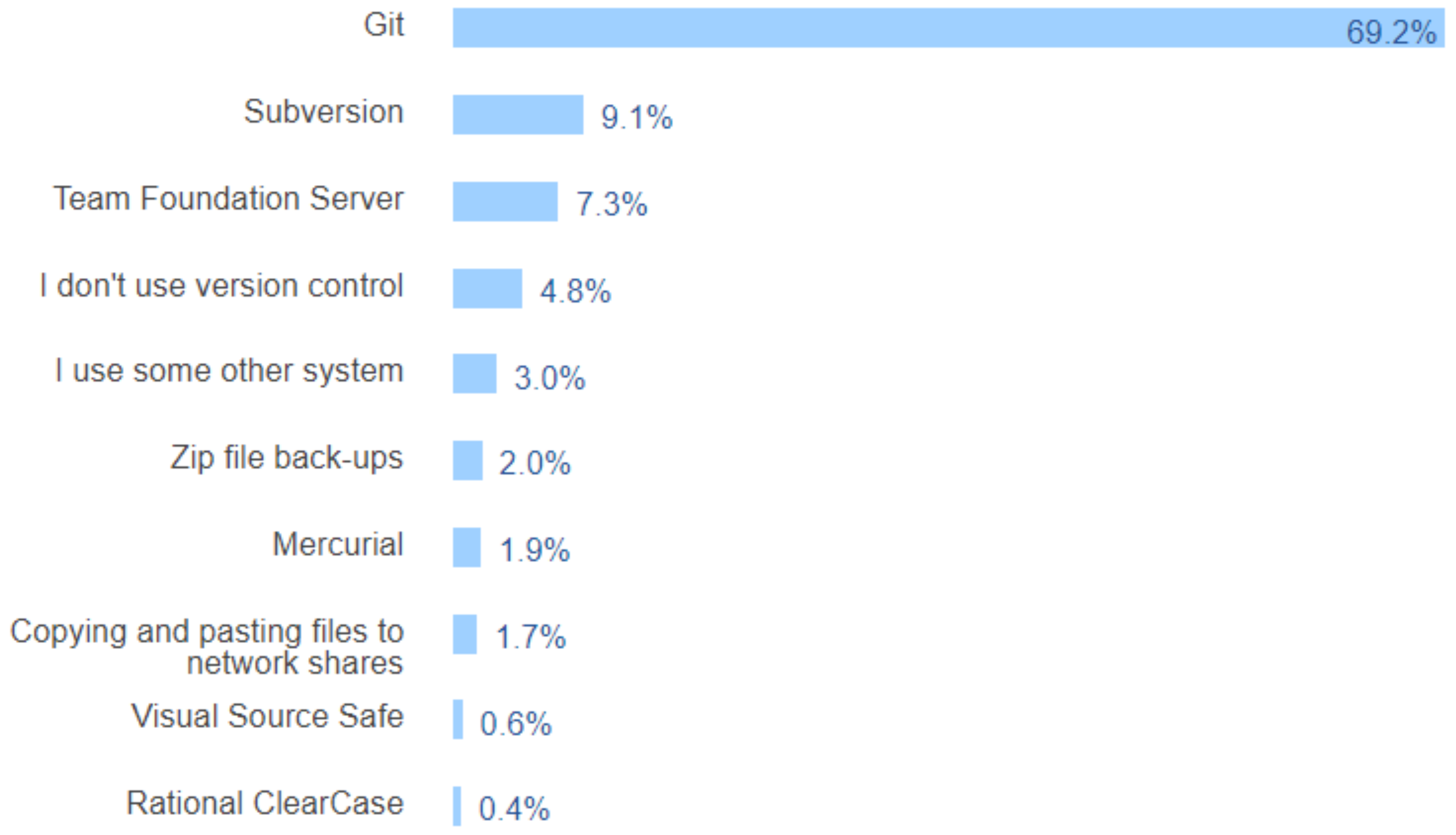
When using a version control system, you have **one copy of each file** and the *version control system tracks the changes* that have occurred over time

## What is version control?

A way to manage the evolution of a set of files



The set of files is referred to as a **repository (repo)**


Git
69.2%
Subversion
9.1%
Team Foundation Server
7.3%
I don't use version control
4.8%
I use some other system
3.0%
Zip file back-ups
2.0%
Mercurial
1.9%
Copying and pasting files to network shares
1.7%
Visual Source Safe
0.6%
Rational ClearCase
0.4%

## git & GitHub

git

the version control system

~ Track Changes from Microsoft Word….on steroids

**GitHub** (or Bitbucket or GitLab) is the home **where your git-based projects live** on the Internet.

~ Dropbox + social media for programmers

## Do you even version control?

[https://forms.gle/8UeUL2Ux4YtG2CVr8](https://forms.gle/8UeUL2Ux4YtG2CVr8)

## Why version control with git and GitHub?

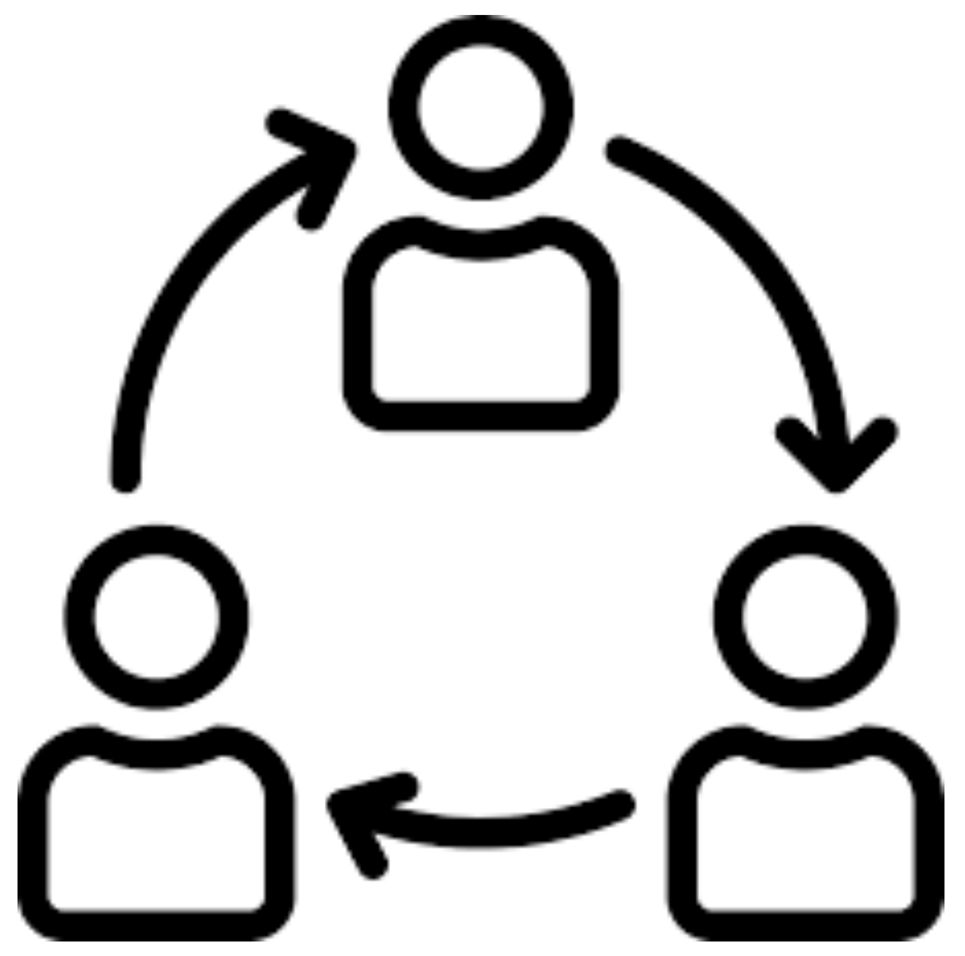
Collaboration

Returning to a safe state

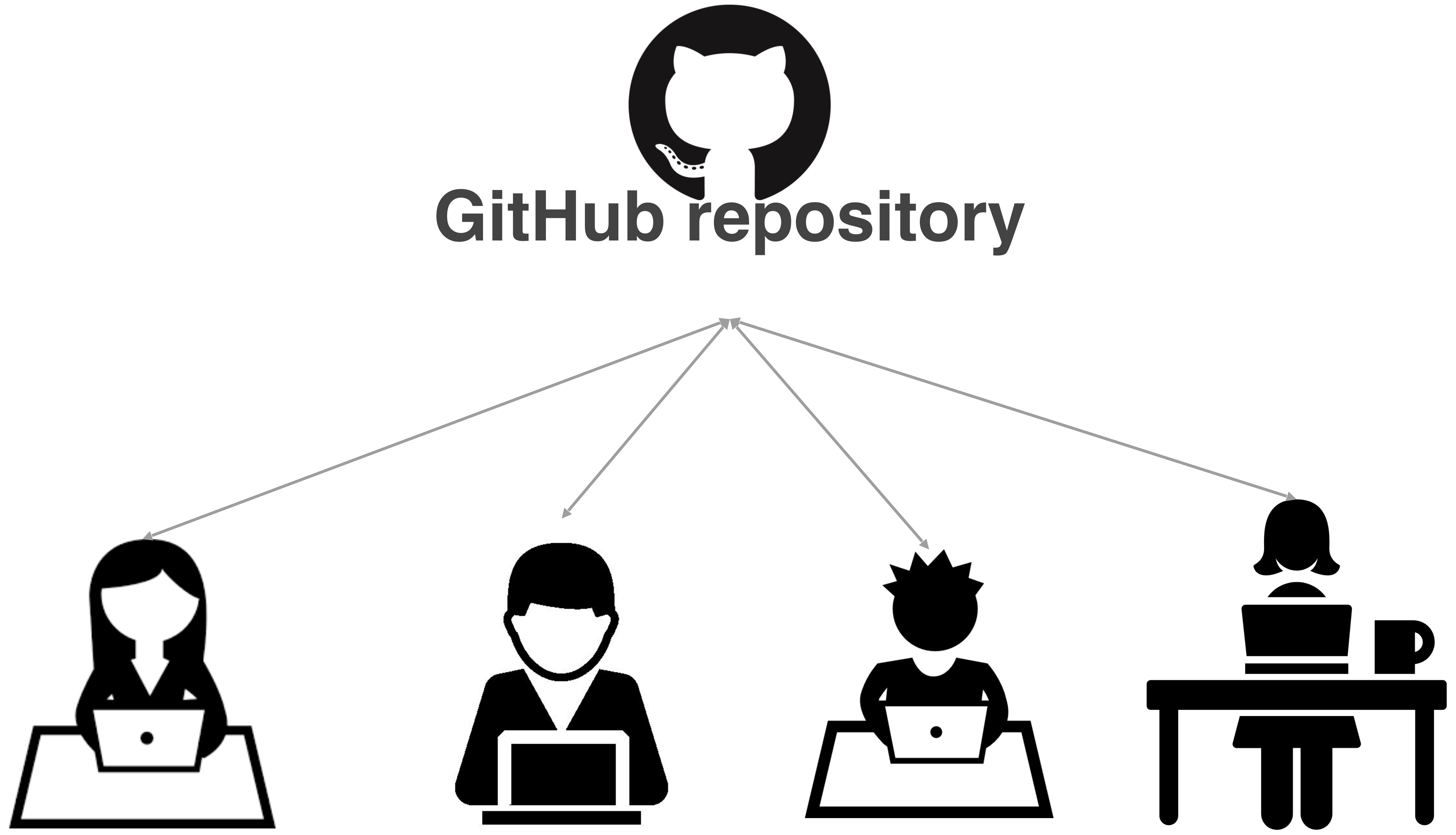
Exposure for your work

Tracking others' work

## Collaborate like you do with Google Docs



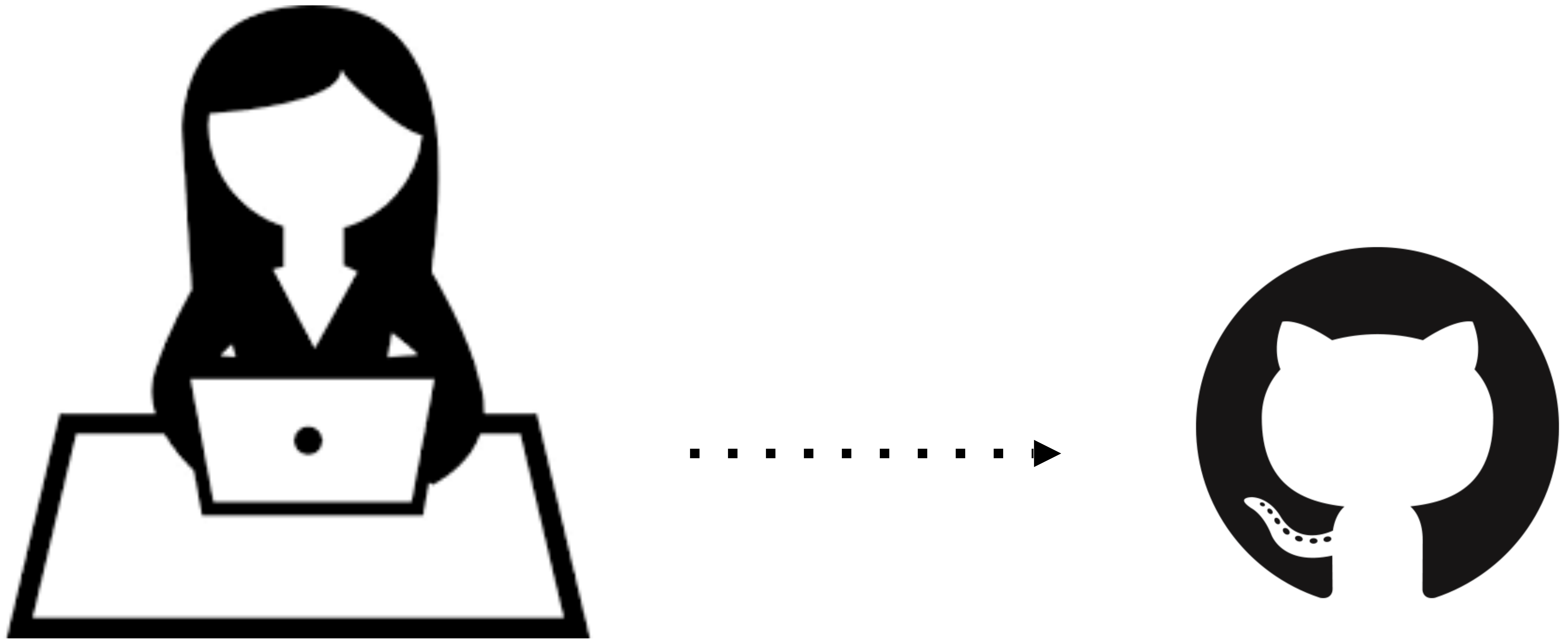
Collaboration



Each person is making changes **locally** (on their computer)

## Make changes locally, while knowing a stable copy exists

Returning to a safe state



You’re free and safe to **try things out locally**. You’ll only send changes to the repo when you’re at a stable point

## Your repositories will be visible to others!

Exposure for your work

Your public GitHub repos are your coding social media

## And vice versa, you can search for the code you need

For instance, this might come in handy when thinking about class projects

[https://github.com/topics/datascience-projects](https://github.com/topics/datascience-projects)

Search or jump to...
/
Pull requests
Issues
Marketplace
Explore
Explore
Topics
Trending
Collections
Events
GitHub Sponsors

# april-fools

DuckMasterAI / rickroll-bot

Star 5
Code
Issues
Pull requests

A simple bot to rickroll your friends on Discord!

discord-bot discord-py april-fools rickroll never-gonna-give-you-up never-gonna-let-you-down rick-astley discord-py-bot

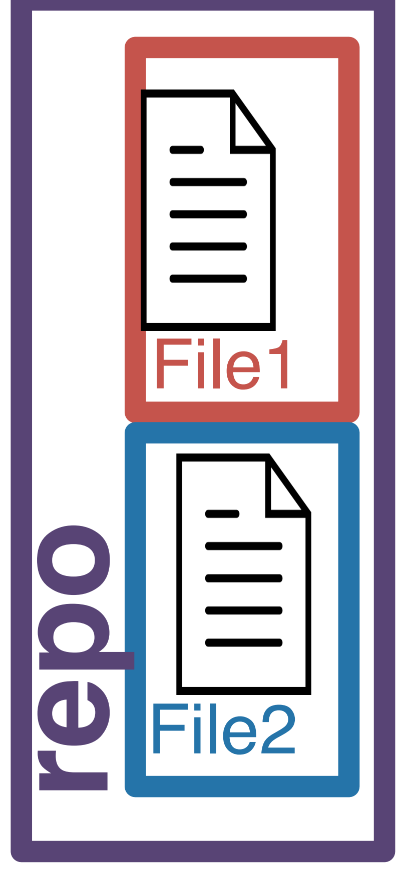
Updated 2 hours ago Python

## Keep up with others' work easily

Tracking others' work

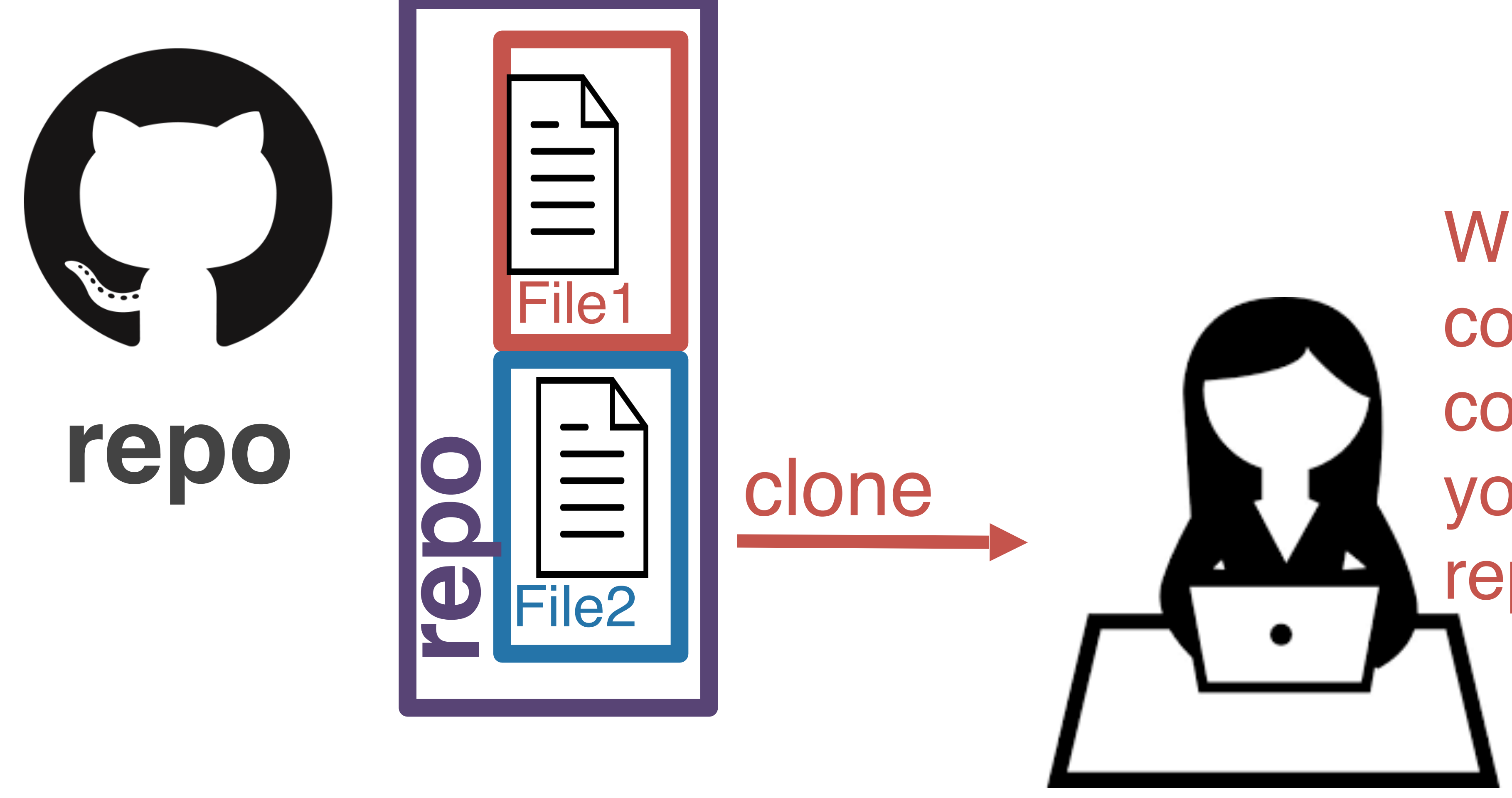
As a social platform, you can see others' work too!

repo

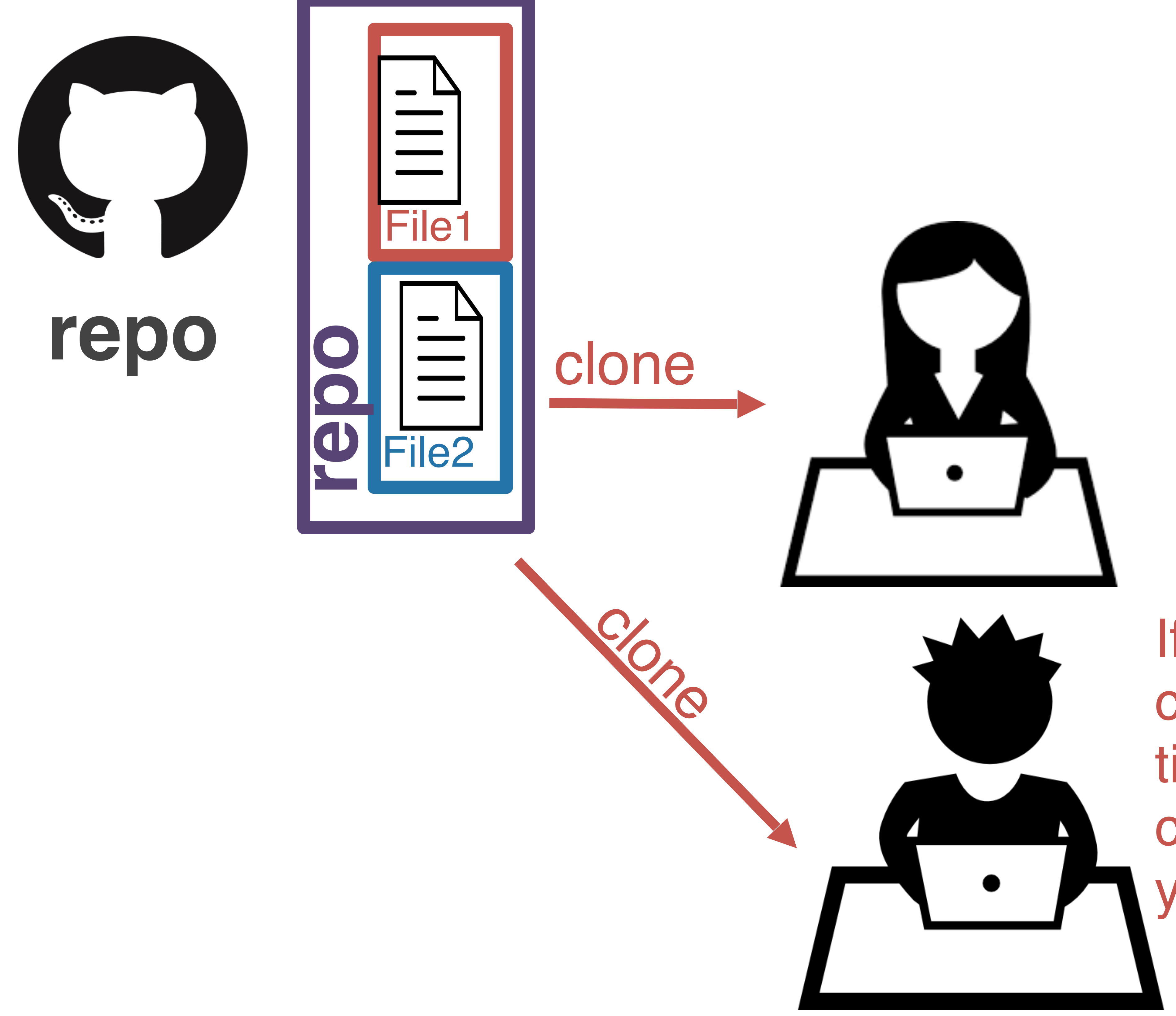


A **GitHub repo** contains all the files and folders for your project.

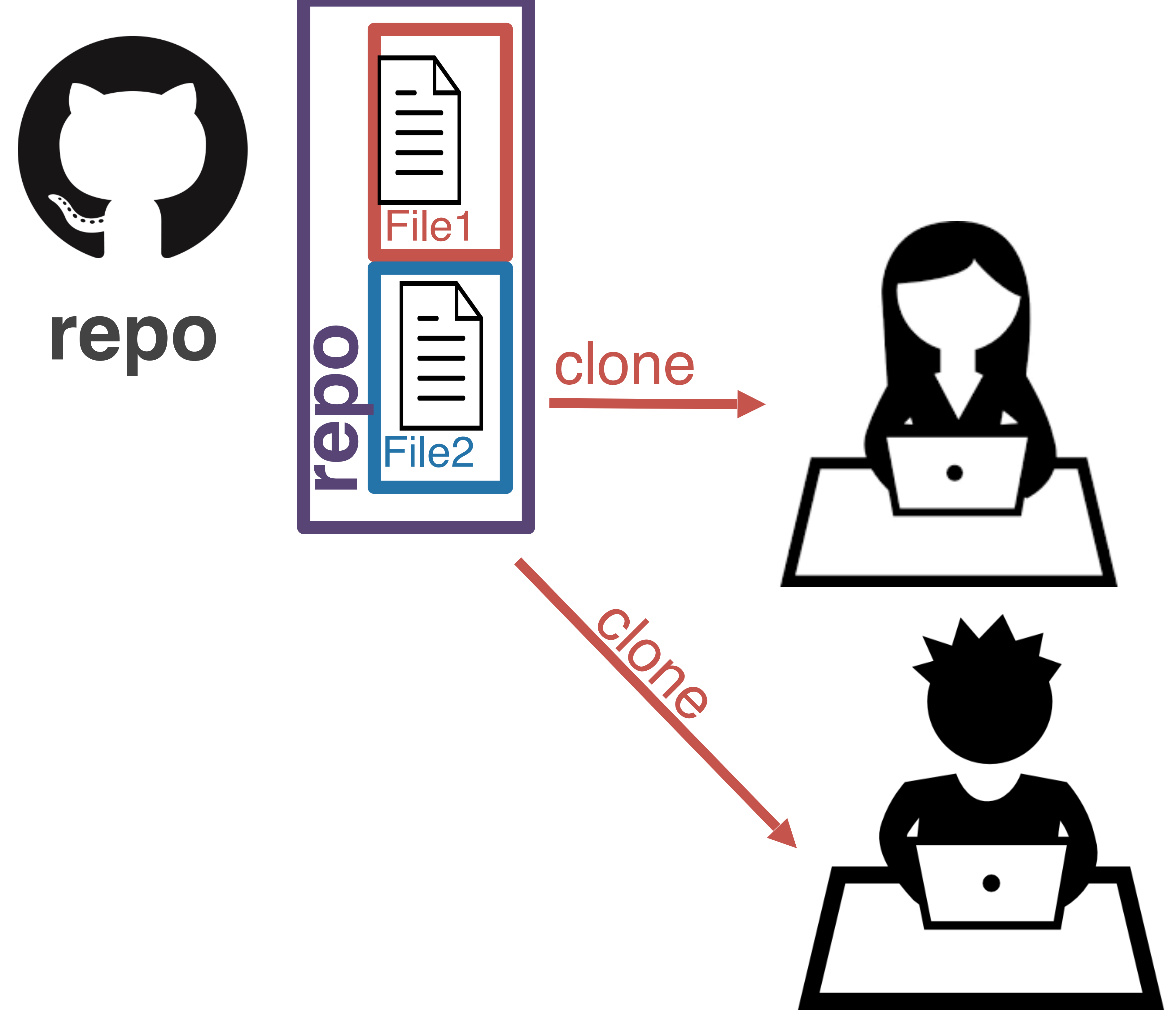
GitHub is a **remote host**. The files are geographically distant from any files on your computer.



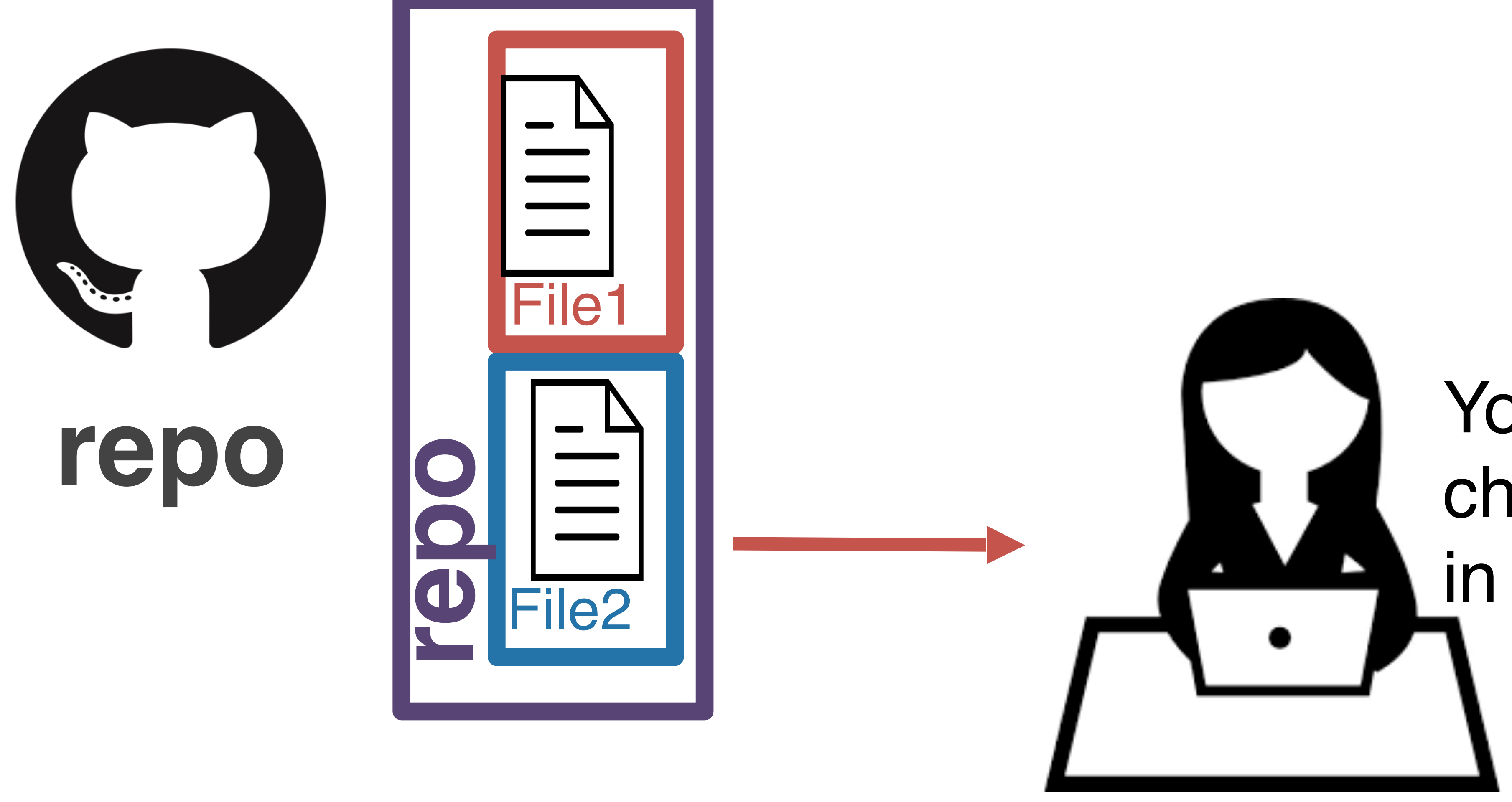
When you first make a copy onto your local computer (read: laptop), you **clone** the repository.



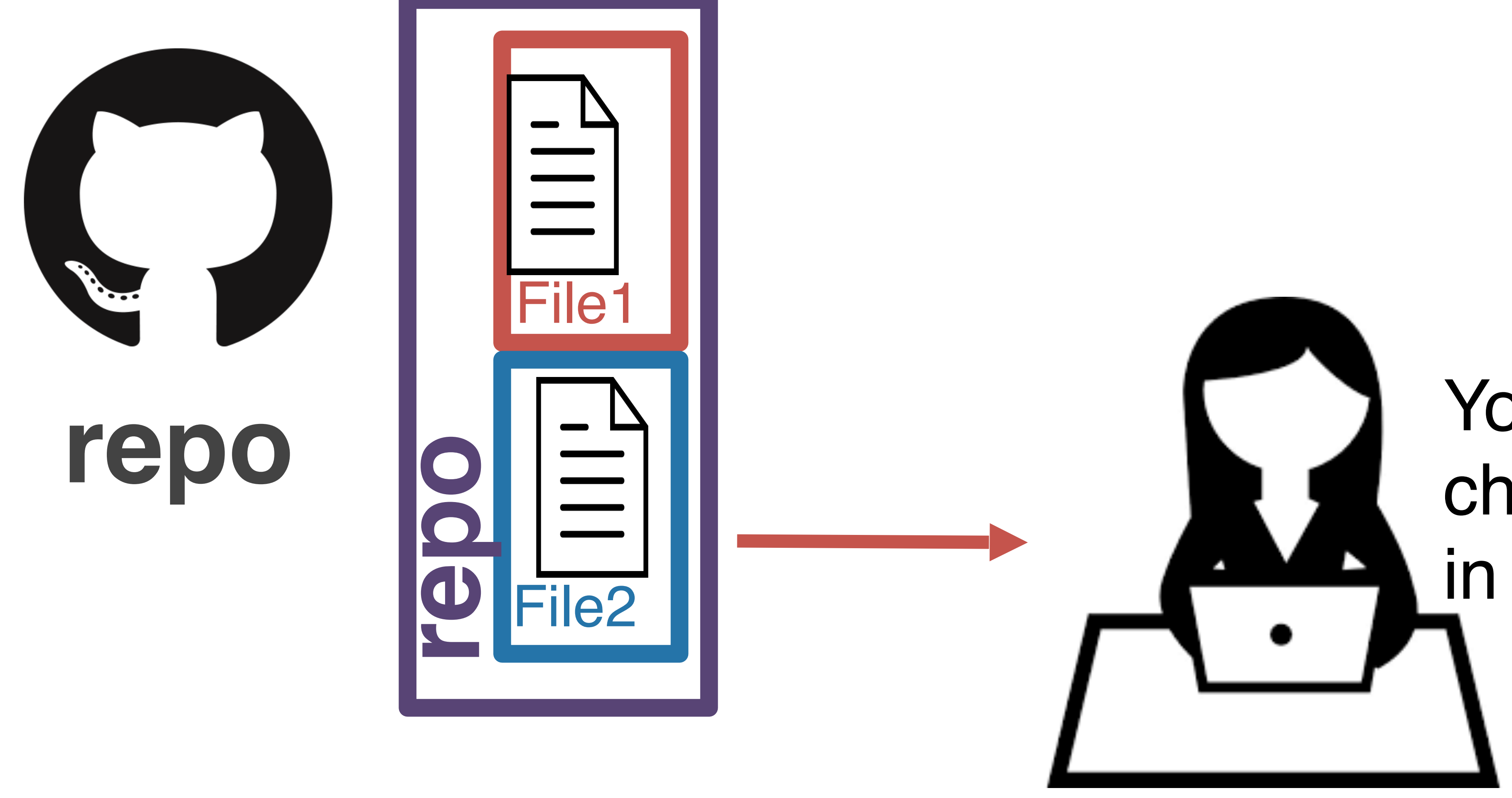
If someone else on your project cloned the repo at the same time, you would have identical copies of the project on each of your computers.



Yay! Everyone can work on the project!

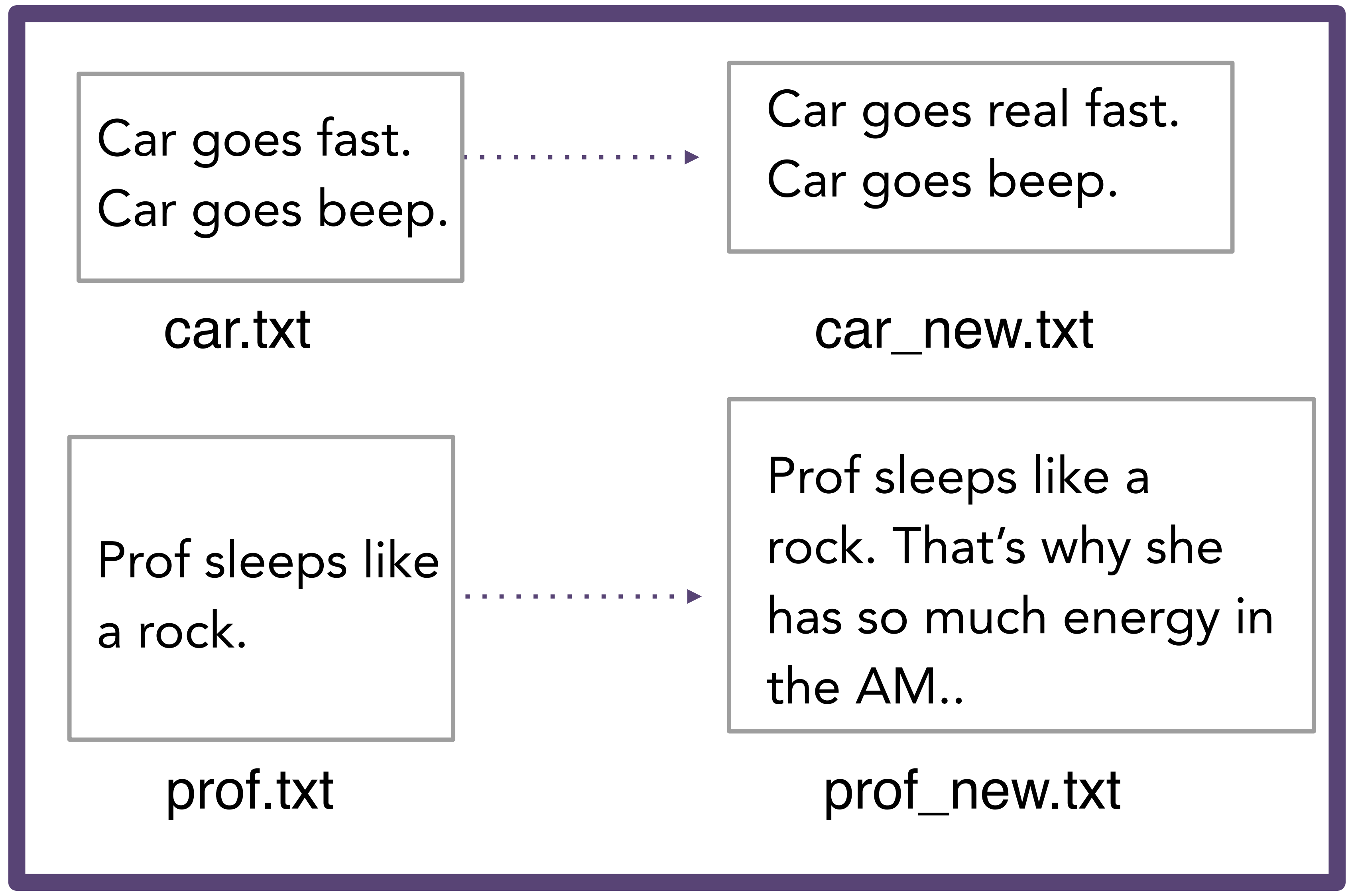


You decide you want to change some of the text in the project.



You decide you want to change some of the text in the project.

repo
repo
File1
File2


Car goes fast.
Car goes beep.

car.txt

Car goes real fast.
Car goes beep.

car_new.txt

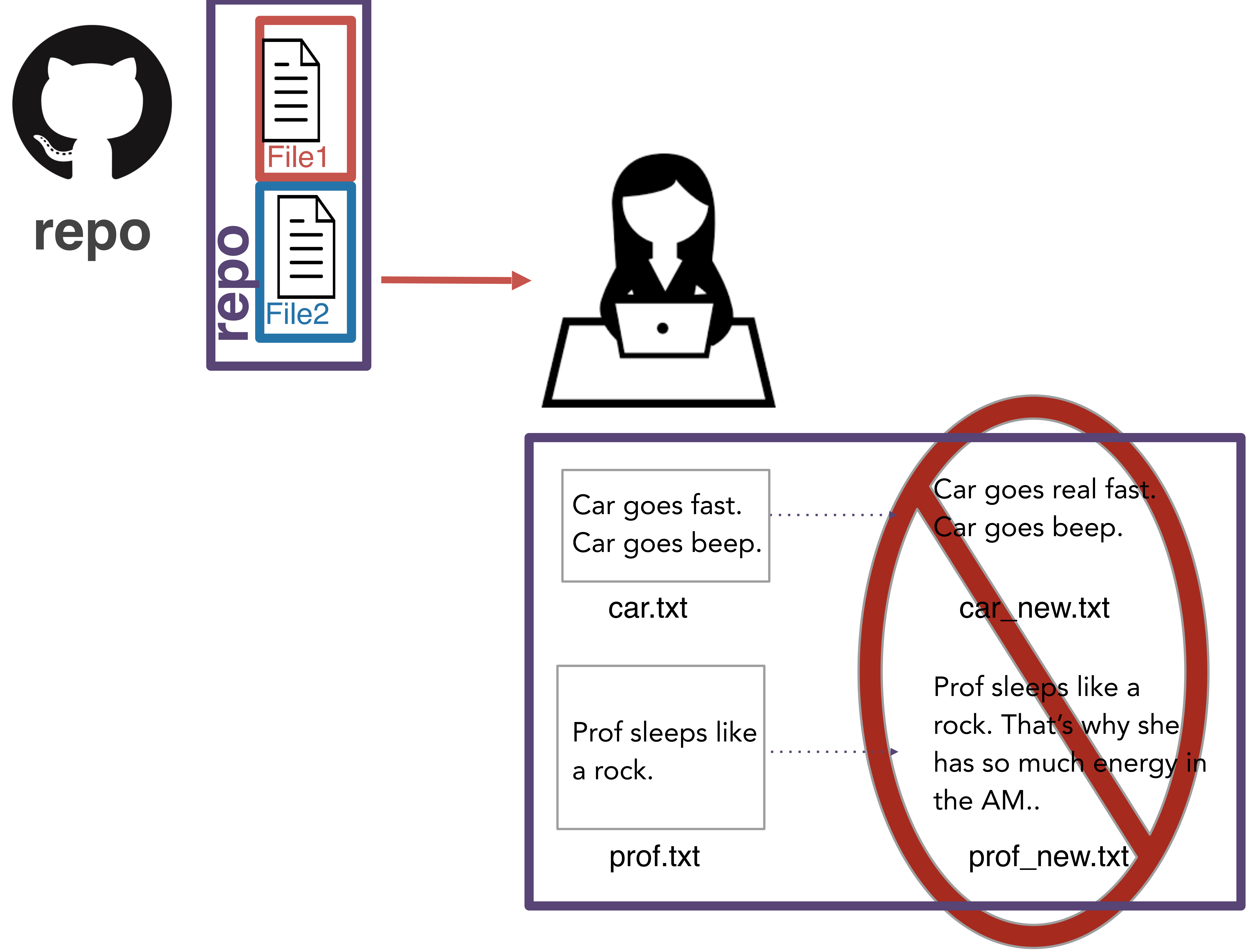
Prof sleeps like a rock.

prof.txt

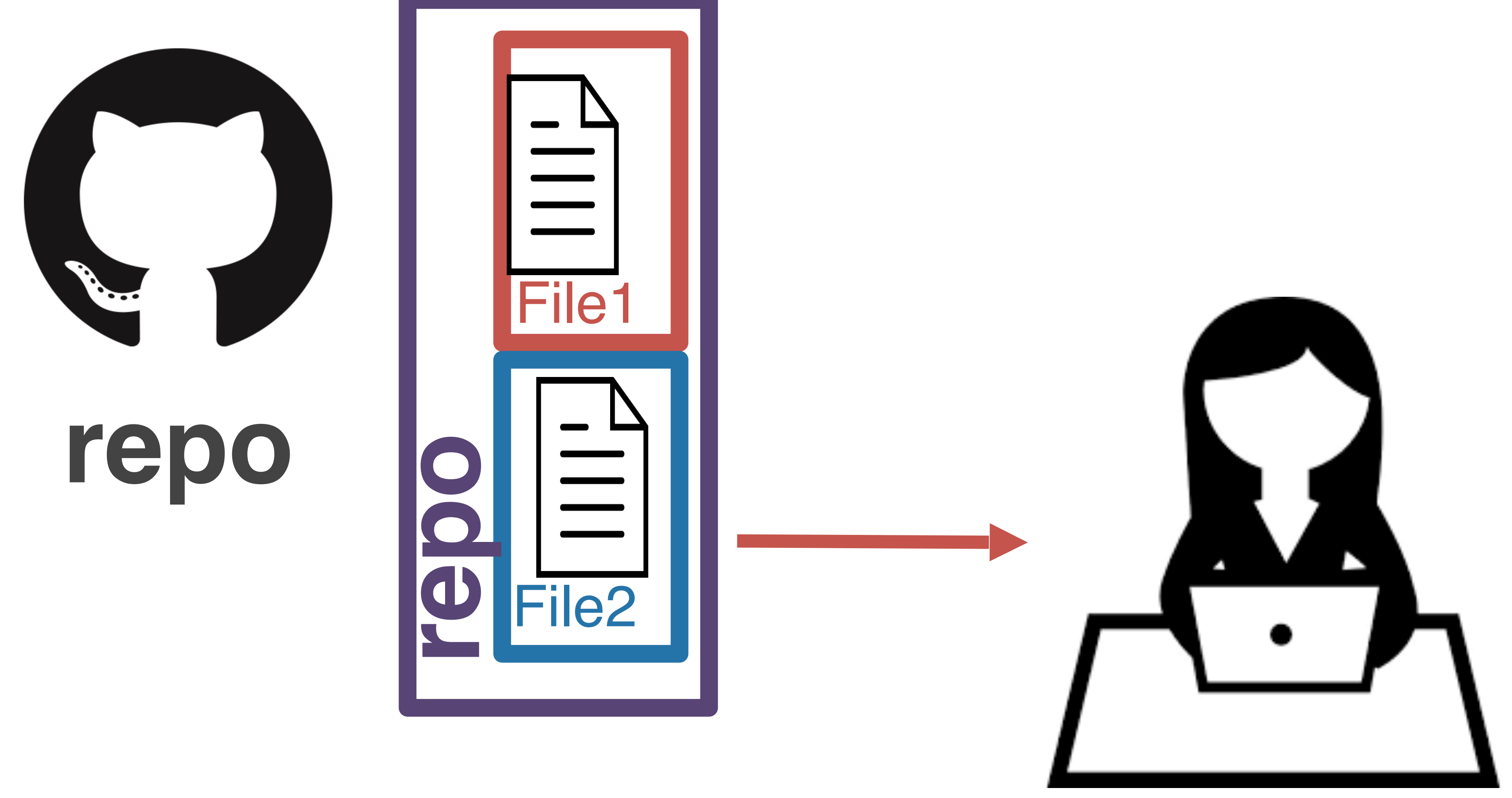
Prof sleeps like a rock. That's why she has so much energy in the AM..

prof_new.txt

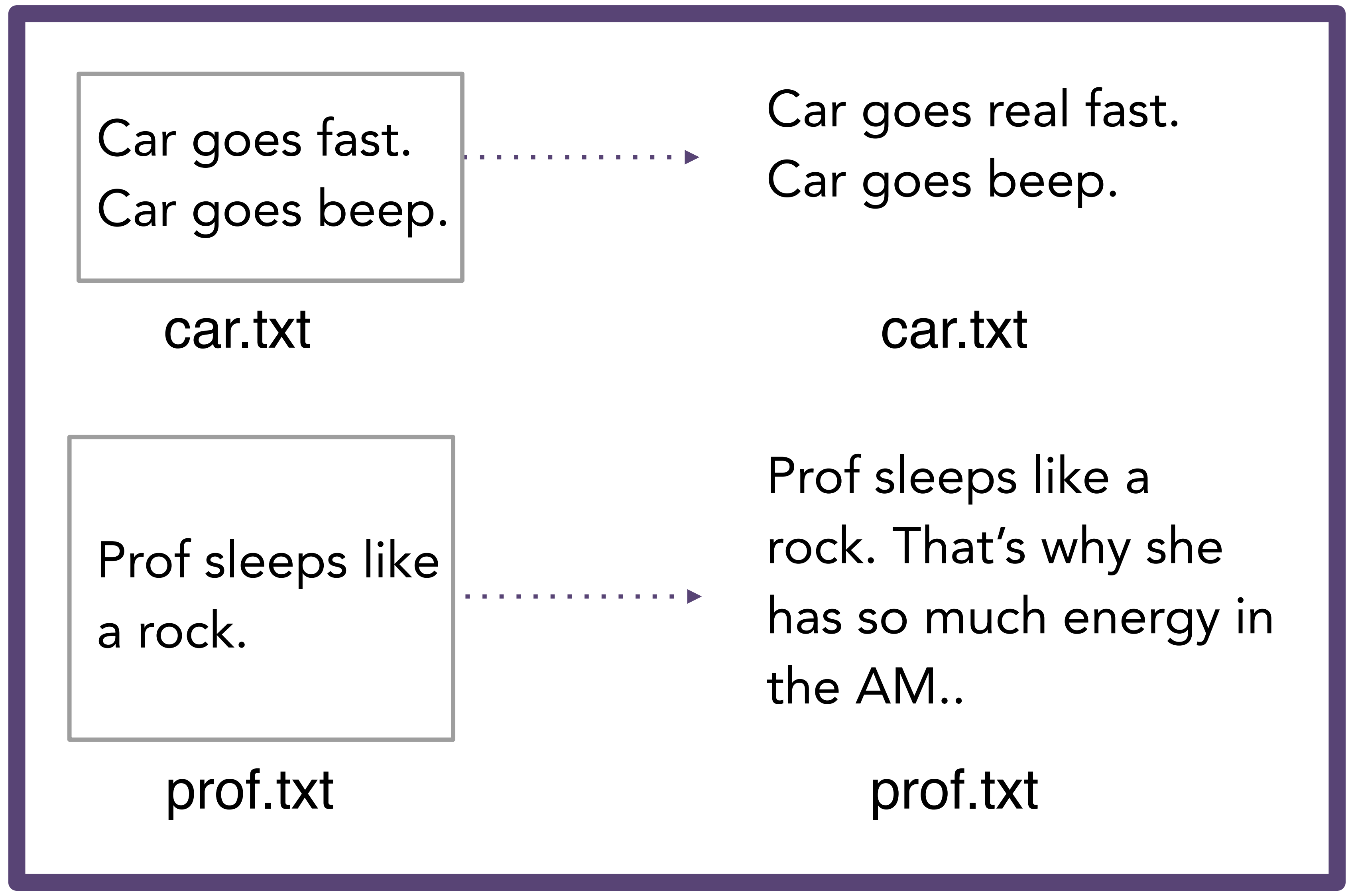
without git...you'd likely rename these files….



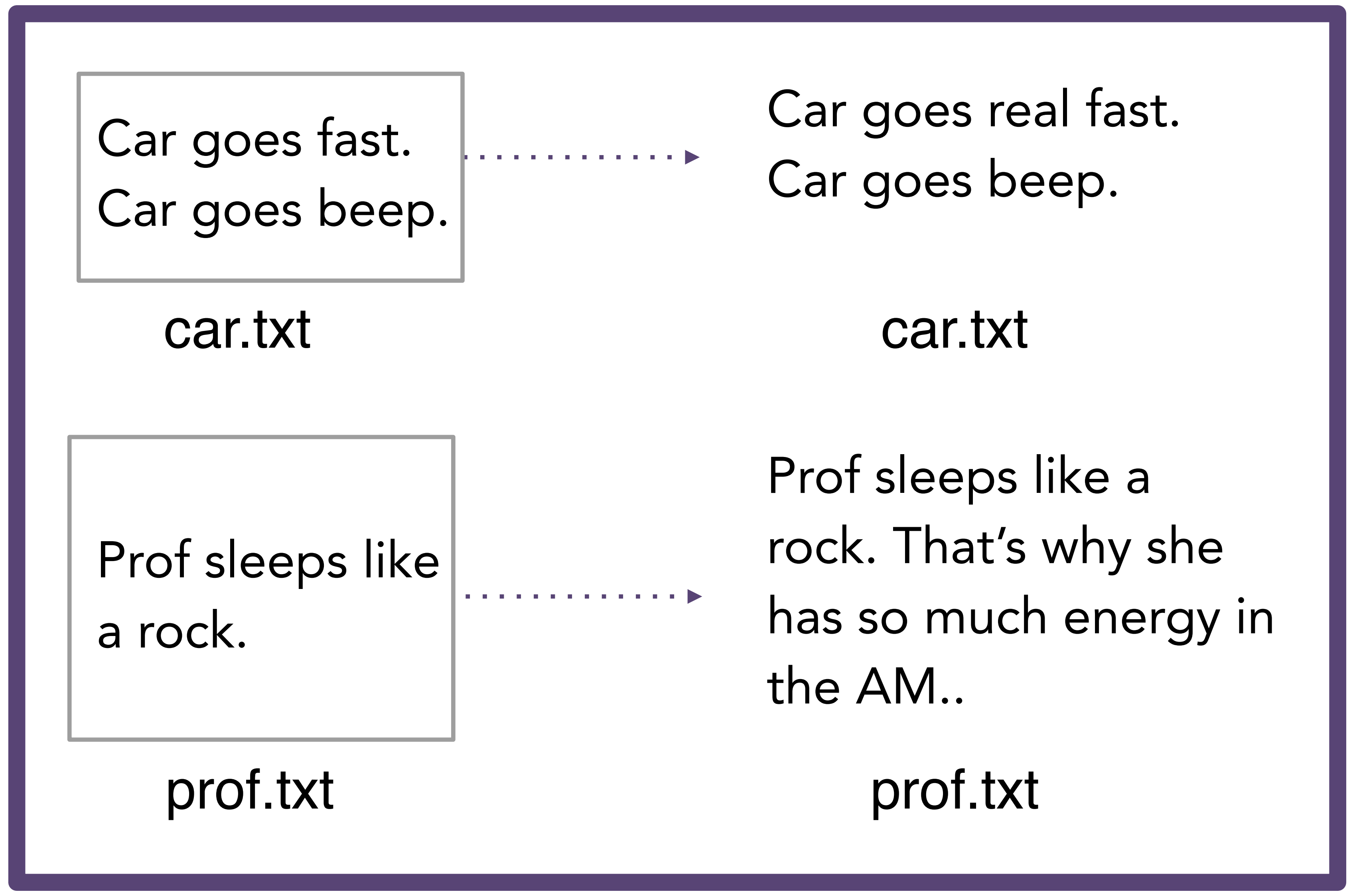
Thank goodness those days are over!



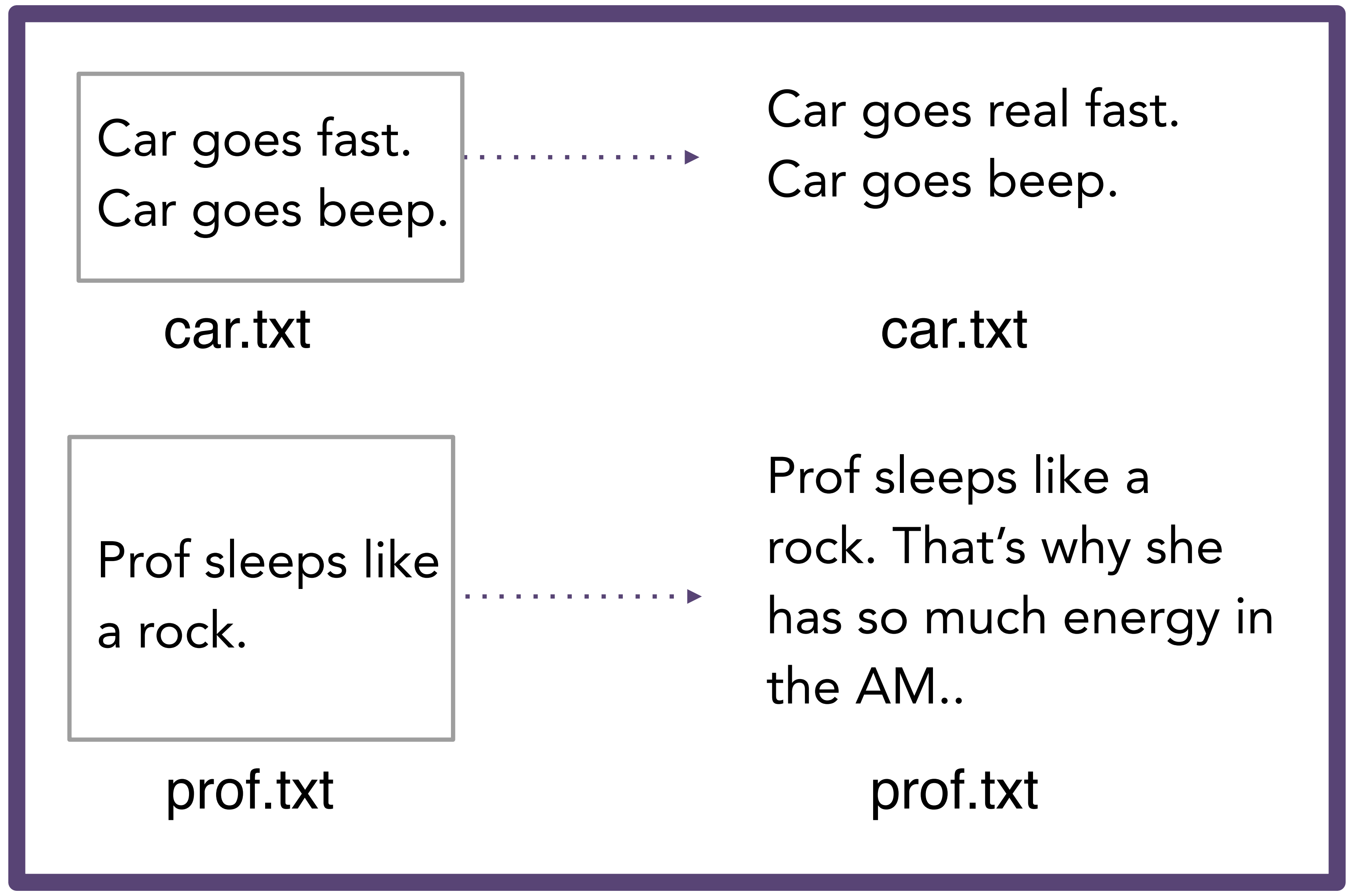
| | |
|---|---|
| `git add file` | stages specified file (or folder) |
| `git add .` | stages new and modified files |
| `git add -u` | stages modified and deleted files |
| `git add -A` | stages new, modified, and deleted files |
| `git add *.csv` | stages any files with .csv extension |
| `git add *` | use with caution: stages everything |



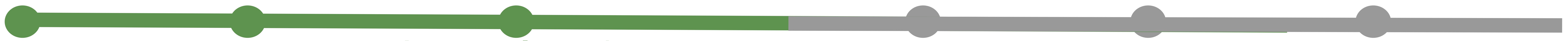
Instead, you tell git which files you'd like to keep track of using **add**. This process is called *staging*.



Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.



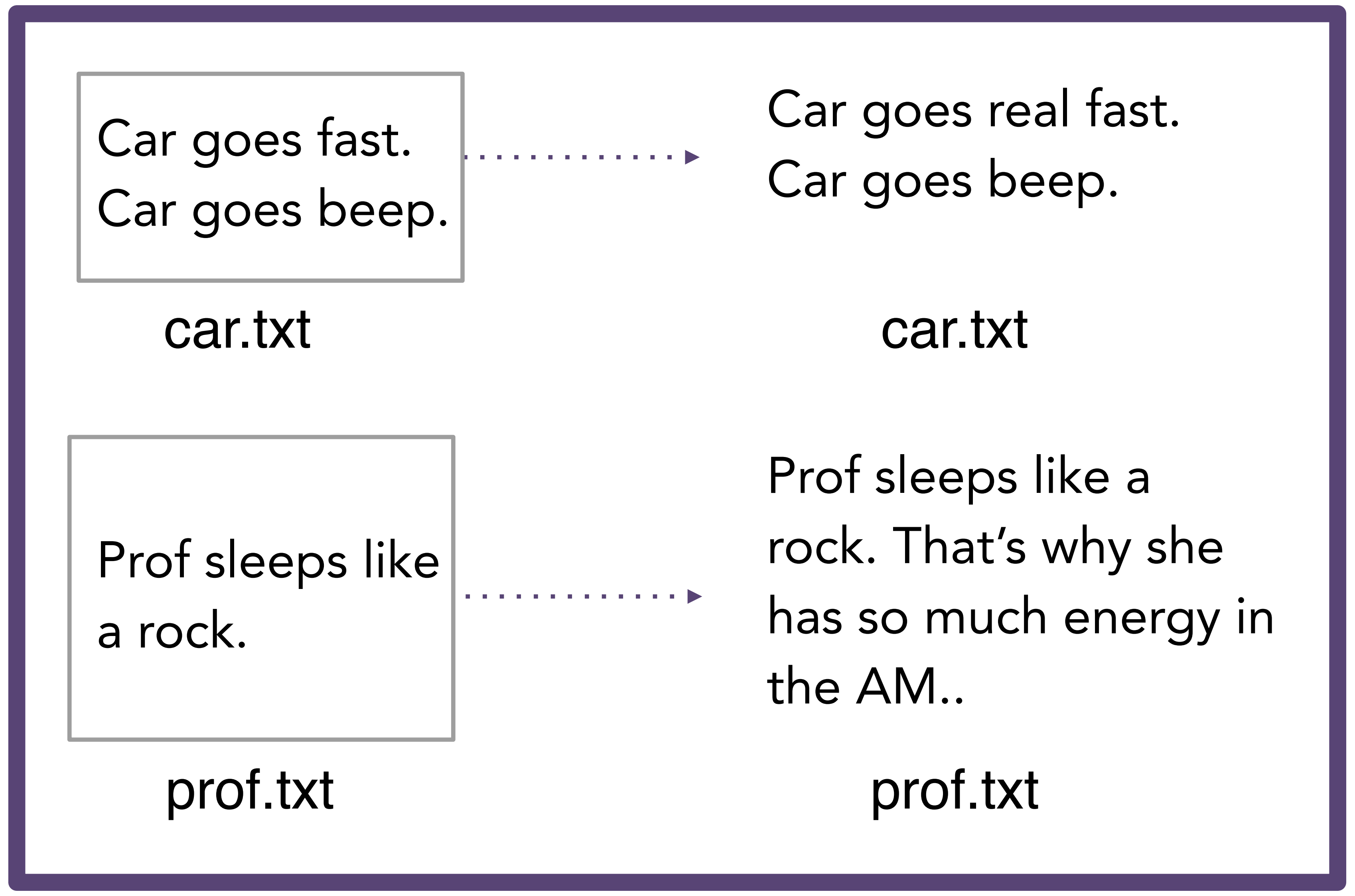
Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.



A **commit** tracks who, what, and when

You can make commits more informative by adding a **commit message**.

Example: `git commit -m 'fix typos in car and prof'`



Car goes fast.
Car goes beep.

car.txt

Car goes real fast.
Car goes beep.

car.txt

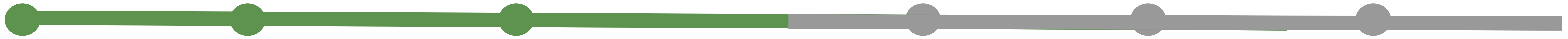
Prof sleeps like a rock.

prof.txt

Prof sleeps like a rock. That's why she has so much energy in the AM..

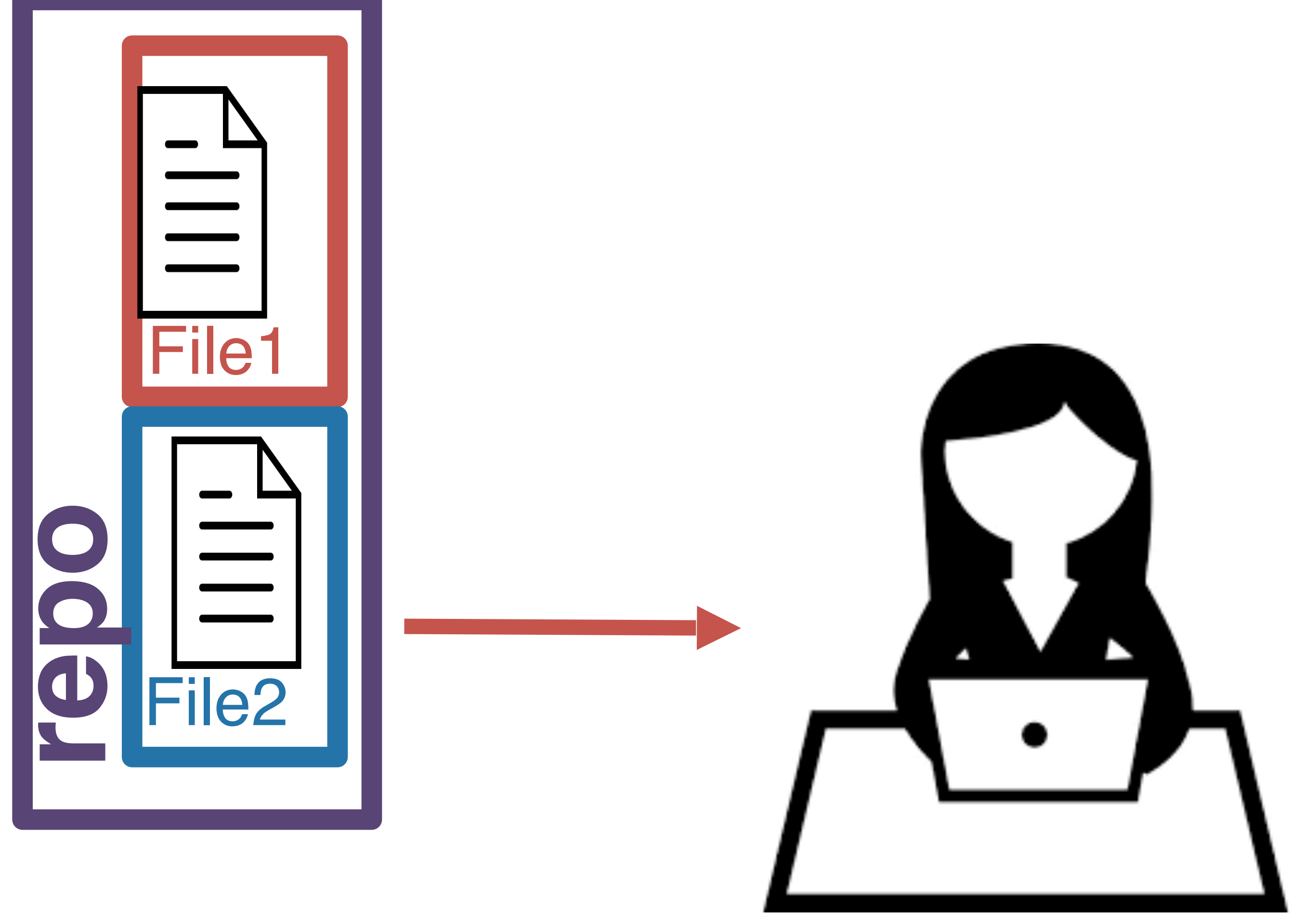
prof.txt

Then, you create a snapshot of your files at this point. This snapshot is called a **commit**.

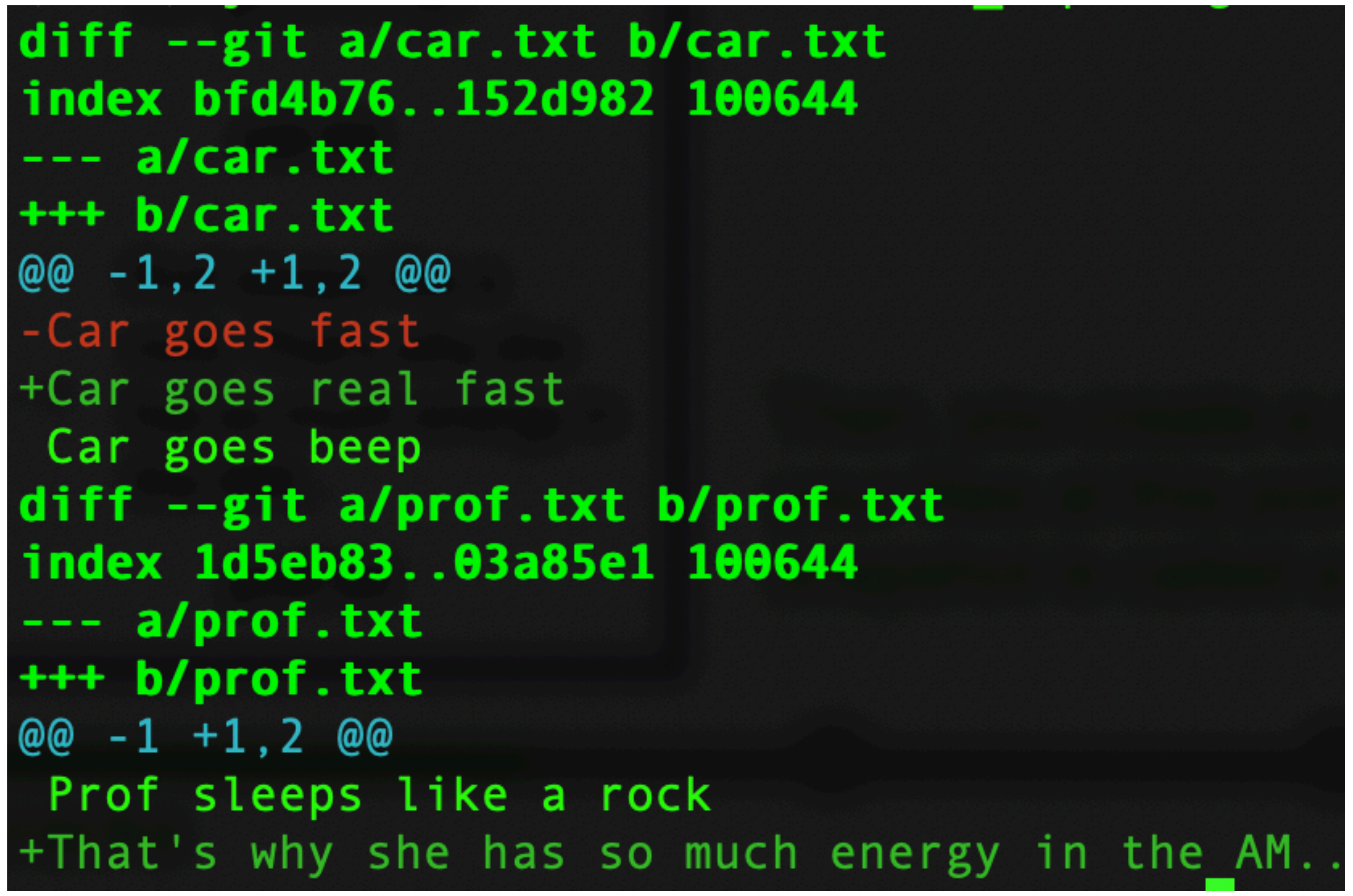


A **commit** tracks who, what, and when

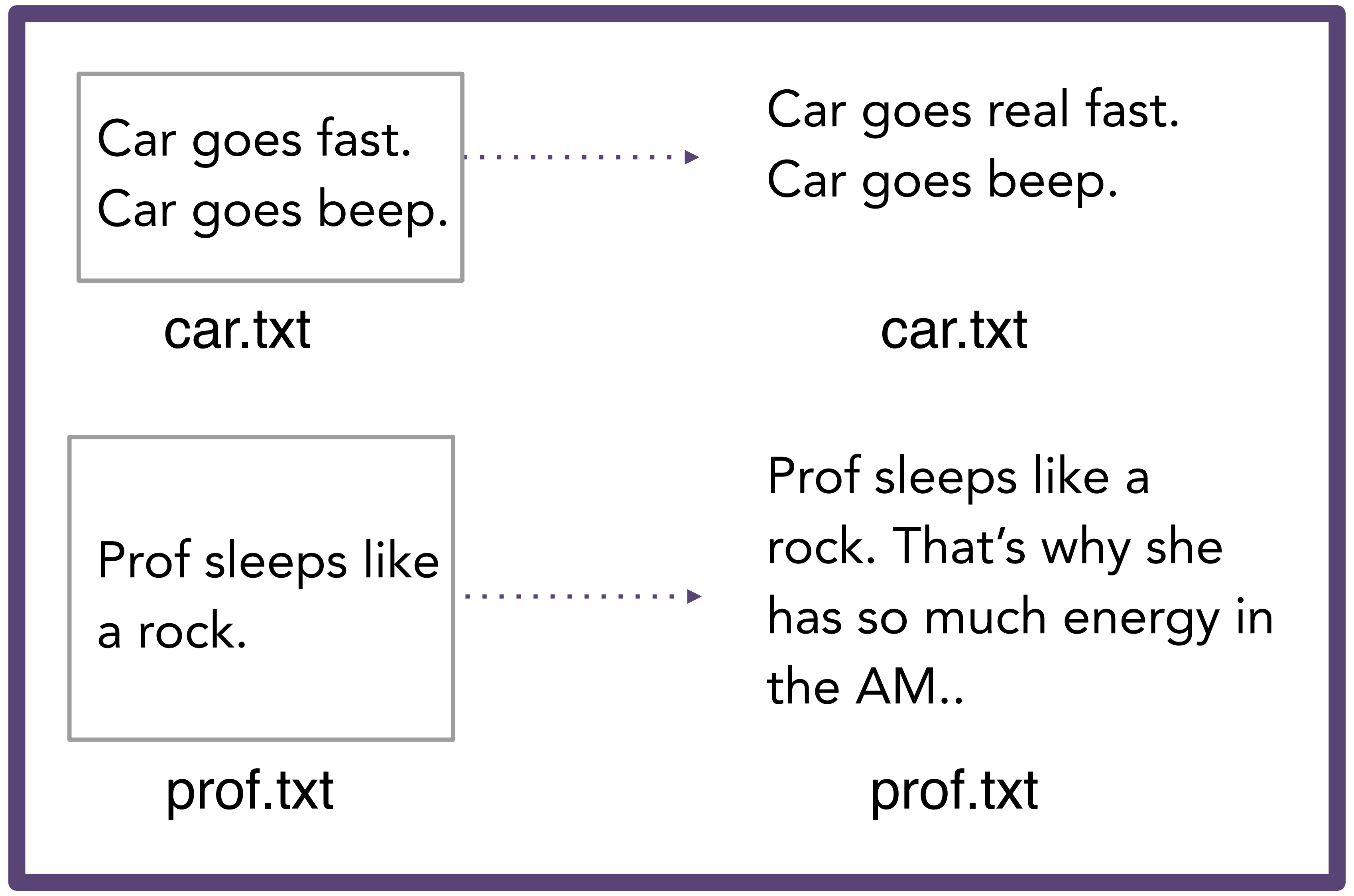
repo



## A commit is stored as a **diff**



```
diff --git a/car.txt b/car.txt
index bfd4b76..152d982 100644
--- a/car.txt
+++ b/car.txt
@@ -1,2 +1,2 @@
-Car goes fast
+Car goes real fast
 Car goes beep
diff --git a/prof.txt b/prof.txt
index 1d5eb83..03a85e1 100644
--- a/prof.txt
+++ b/prof.txt
@@ -1 +1,2 @@
 Prof sleeps like a rock
+That's why she has so much energy in the AM..
```



Car goes fast.
Car goes beep.

car.txt

Car goes real fast.
Car goes beep.

car.txt

Prof sleeps like a rock.

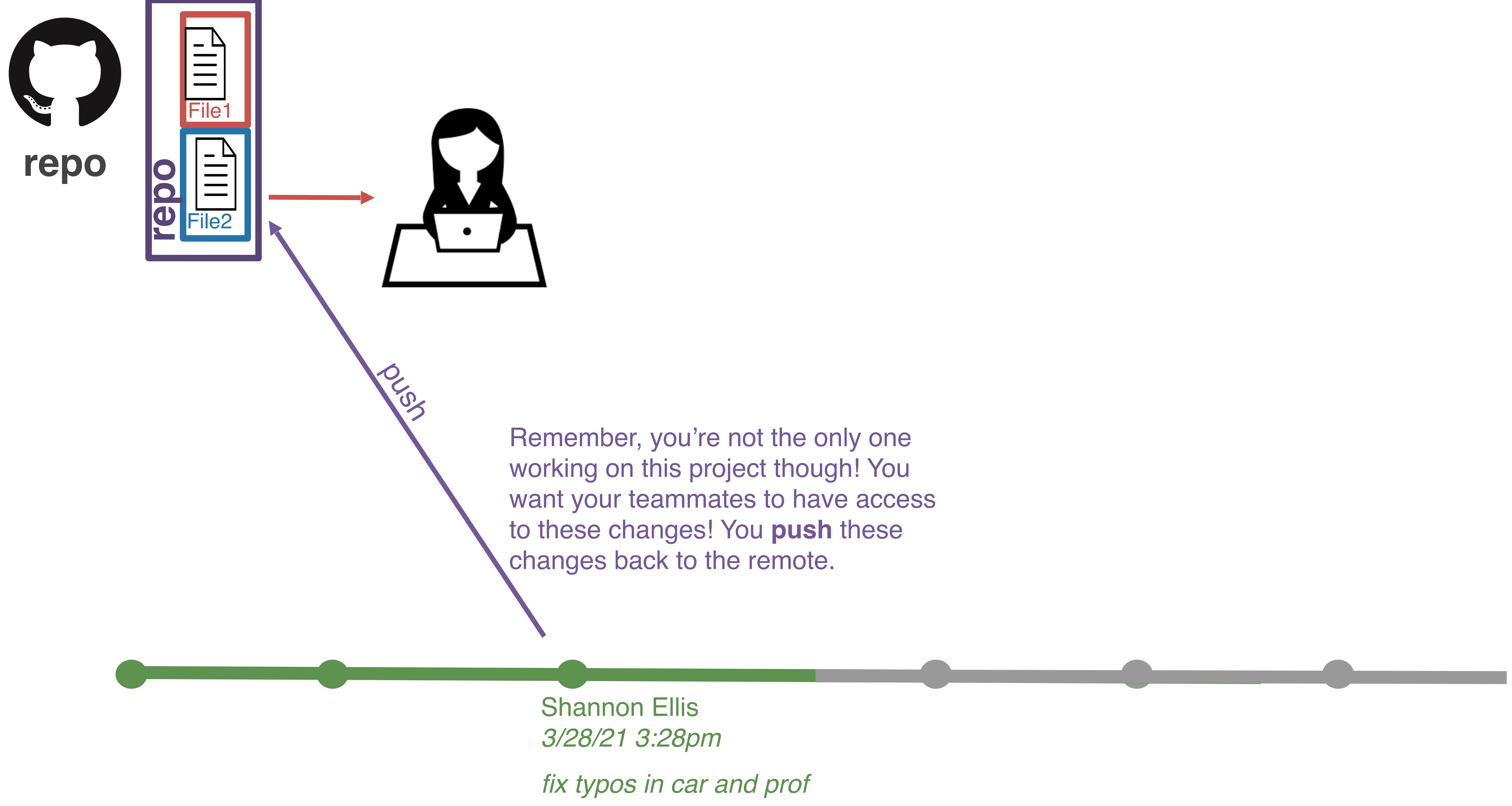
prof.txt

Prof sleeps like a rock. That's why she has so much energy in the AM..

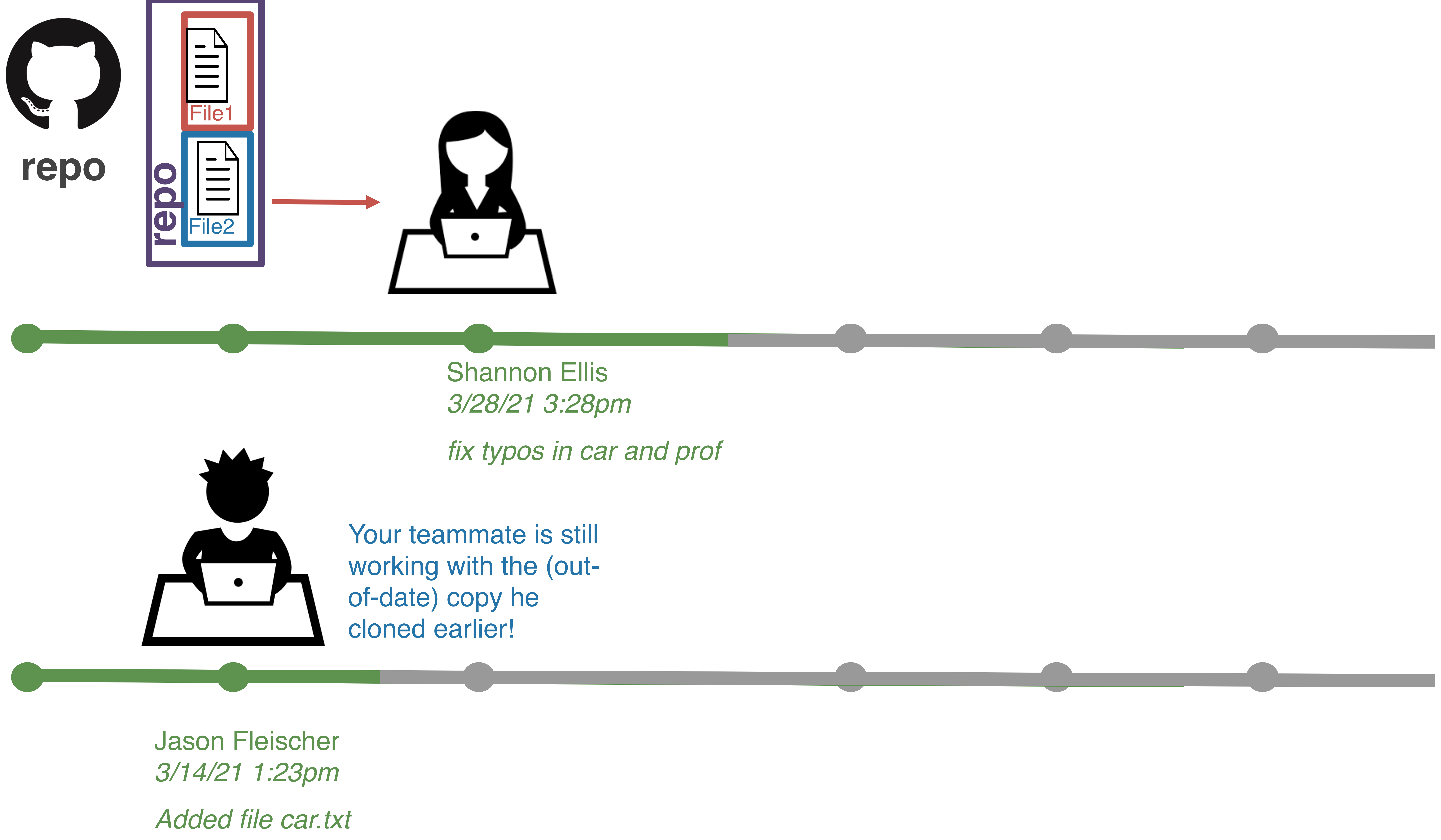
prof.txt

Shannon Ellis
*3/28/21 3:28pm*

*fix typos in car and prof*

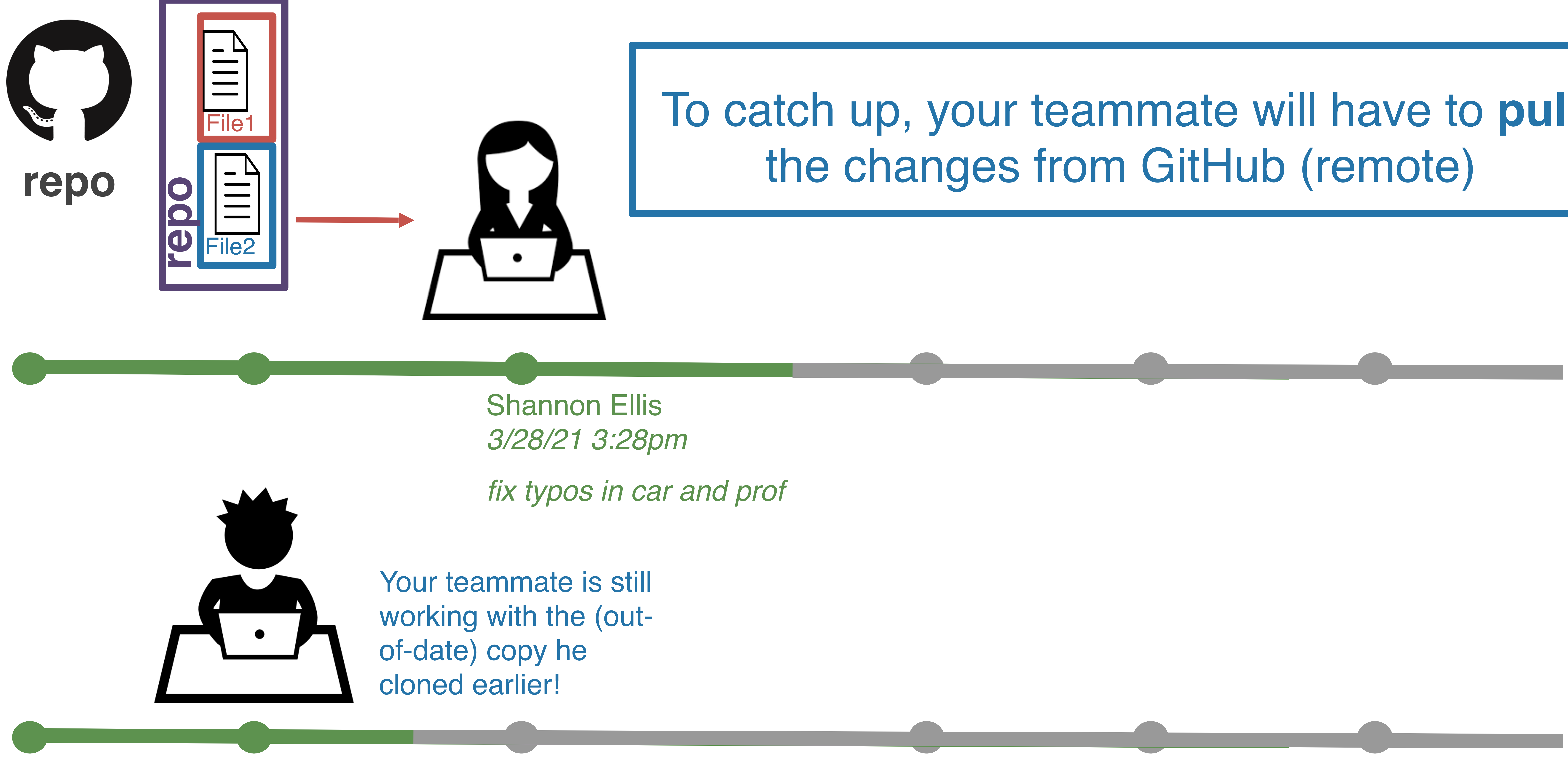


Remember, you're not the only one working on this project though! You want your teammates to have access to these changes! You **push** these changes back to the remote.

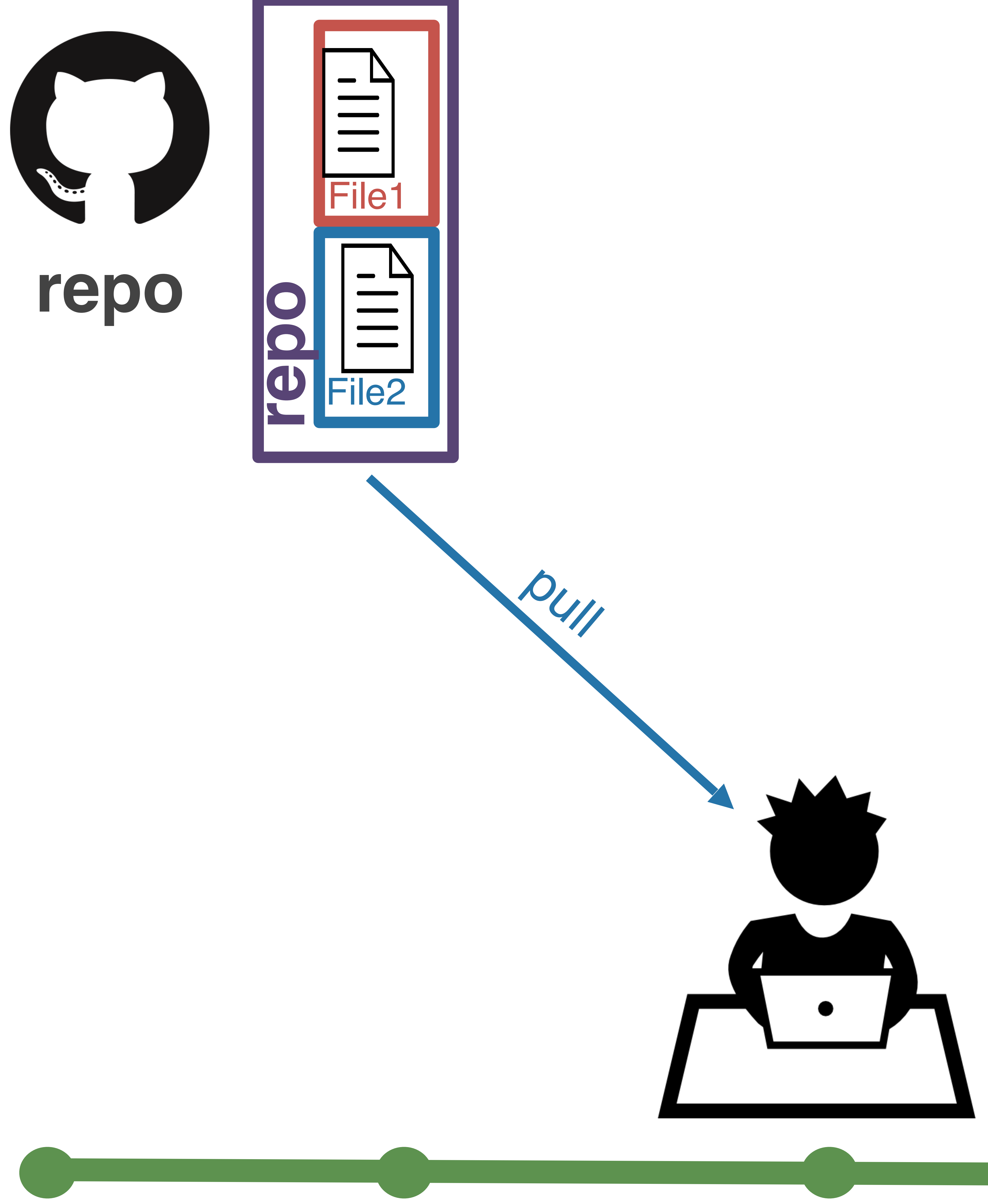


Your teammate is still working with the (out-of-date) copy he cloned earlier!

To catch up, your teammate will have to **pull** the changes from GitHub (remote)

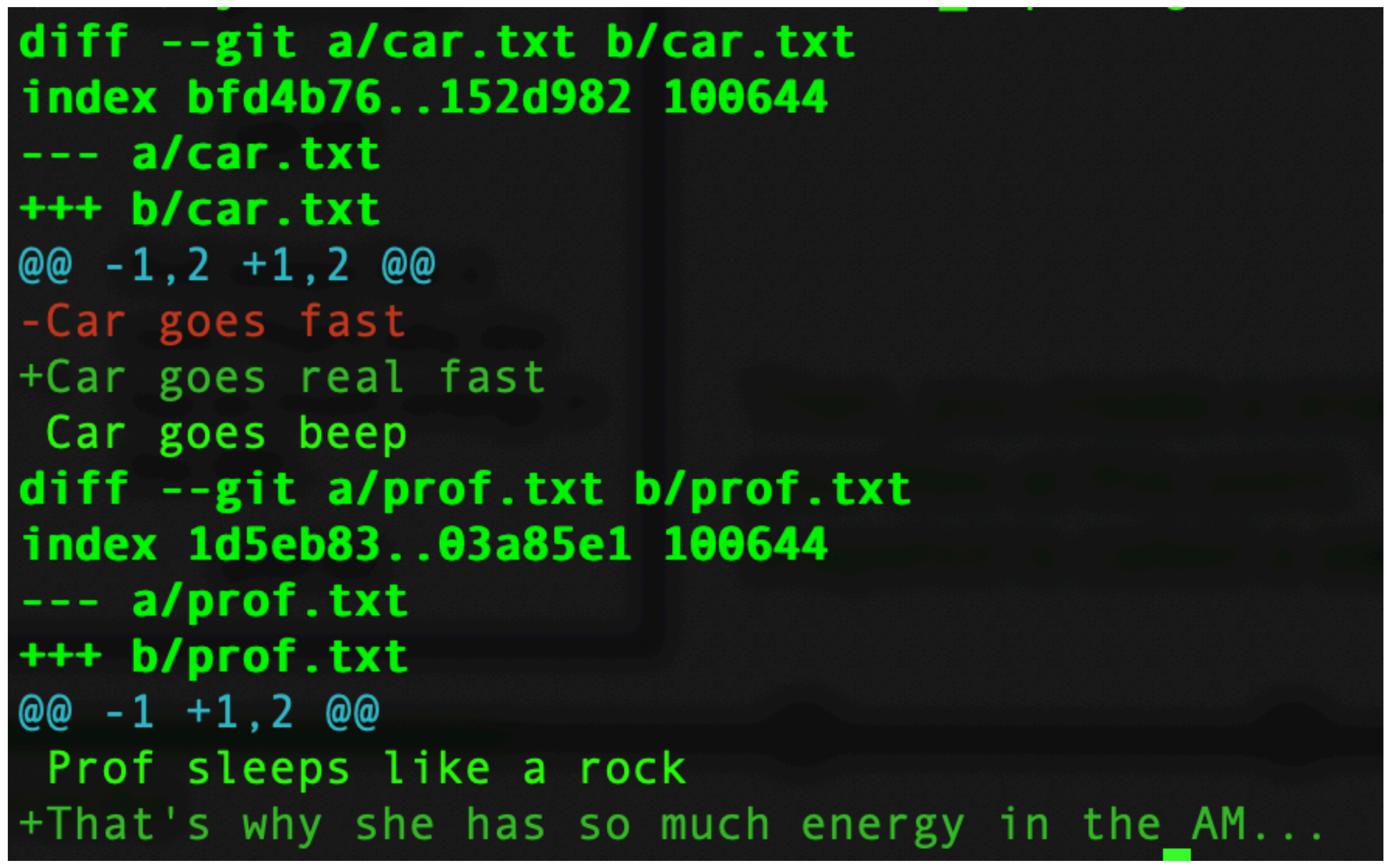


Your teammate is still working with the (out-of-date) copy he cloned earlier!

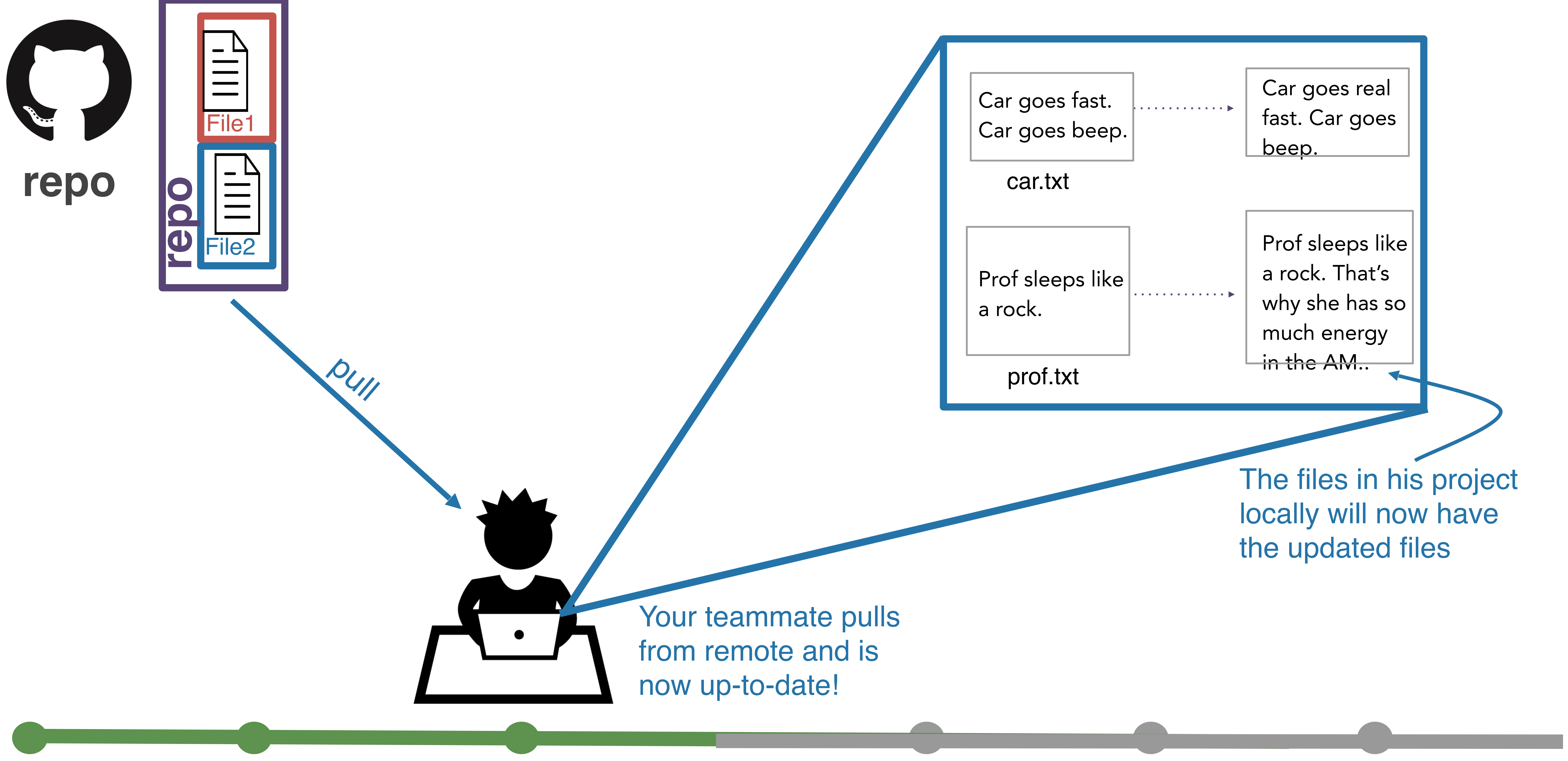


Your teammate pulls from remote and is now up-to-date!

The pull transports a **diff** and applies it to your repo



```
diff --git a/car.txt b/car.txt
index bfd4b76..152d982 100644
--- a/car.txt
+++ b/car.txt
@@ -1,2 +1,2 @@
-Car goes fast
+Car goes real fast
 Car goes beep
diff --git a/prof.txt b/prof.txt
index 1d5eb83..03a85e1 100644
--- a/prof.txt
+++ b/prof.txt
@@ -1 +1,2 @@
 Prof sleeps like a rock
+That's why she has so much energy in the AM...
```



Car goes fast. Car goes beep.

Car goes real fast. Car goes beep.

Prof sleeps like a rock.

Prof sleeps like a rock. That's why she has so much energy in the AM..

The files in his project locally will now have the updated files

Your teammate pulls from remote and is now up-to-date!

Shannon Ellis

*3/28/21 3:28pm*

*fix typos in car and prof*

repo
remote host
File1
File2
repo
clone
local host
push
pull
commit

# Let’s recap real quick!

**repo** - set of files and folders for a project
**remote** - where the repo lives
**clone** - get the repo from the remote for the first time
**add** - specify which files you want to stage (add to repo)
**commit** - snapshot of your files at a point in time
**pull** - get new commits to the repo from the remote
**push** - send your new commits to the remote

## Basic version control

https://forms.gle/wHA2GSyuycFre5qr6

## What version control looks like

```
$ git clone https://www.github.com/username/repo.git
$ git pull

[… edit some files, make changes …]

$ git add -A
$ git commit -m “informative commit message”
$ git push
```

Git on command line

## Git exercise #1

1. Login to datahub.ucsd.edu
2. Open a terminal
3. In the terminal type:

```
git config --global user.email “you@ucsd.edu”
git config --global user.name “Your Name”

git clone https://github.com/COGS108/Lectures-Wi26
```

4. Ask yourself, how will the command look different when I want to update my lecture repo with the new slides next week. Try that command out!

## Git exercise #2 - updating a repo

1. Change directory to the new repo Lectures-Wi26
2. Figure out if what (if anything) is different from the last commit, type:
```
git status
```
3. Create a brand new file
```
touch 007-new-movie.mov
```
4. Figure out if what (if anything) is different from the last commit, type:
```
git status
```
5. Stage your changes
```
git add 007-new-movie.mov
```
6. Commit your changes
```
git commit -m “unreleased James Bond movie”
```
7. Question: What happens if you push to Github? Why?
8. Undo these unwanted changes to prevent problems in the future:
```
git reset —hard HEAD~1
```