

# Collaborative Coding

## Version Control with Rstudio/GitHub

Compscibio2023

# Welcome!

## Module expectations & conduct

- Welcoming and supportive space for everyone regardless of background or identity: please use inclusive language
- Show respect and appreciation for participants and facilitators
- Accept constructive criticism with grace
- Everyone has different levels of experience and everyone, including facilitators, has space to learn
- Encourage questions
- “No drop” module: speak up if you fall behind so we can help



# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account
- IV. Walk through and help installing/upgrading software
- V. Rstudio/GitHub integration
- VI. Workflows for collaborative coding with version control
- VII. Practice working collaboratively on a (non-coding) project
- VIII. Bonus: project management boards, issues, milestones

# Systems Supported:

## Systems that are supported:

- Windows (honestly don't know - a modern one?)
- Mac OS 11 or higher

## Systems that are not supported:

- Older Windows systems (How old? Dunno)
- Mac OS <11
- Linux :(

# Rob Baker

## Data Scientist/National Park Service

- Formal education entirely biology (plant evolution, development, genetics, etc)
- First introduced to R in 1999. But really... came kicking and screaming to R during my Ph.D.
- GitHub user since 2013 . . . but similar trajectory: took a long time to really use it
- Ex-Academic and currently work for the National Park Service in Colorado
- Mainly develop in R, build tools for data management, metadata construction, etc (“[NPSdataverse](#)”)
- Even as a “power user” of R and GitHub I only scratch the surface the capability/potential



# Co-facilitators

Emma Girolami



- Recent Miami grad - Spring of 2023!
- Biology major
- Currently works at Hueston Woods State Park

# Co-Facilitators

## Thad Deiss

- Recently completed post-doctoral work at Uppsala University
- Interests in the evolution of vertebrate adaptive immune systems (especially cartilaginous fish and bovids)



# Introduce yourself

~30 seconds each

- Name
- Affiliation/what you do
- Why you chose this module
- What is your most frequently used emoji? (👀)

# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account
- IV. Walk through and help installing/upgrading software
- V. Rstudio/GitHub integration
- VI. Workflows for collaborative coding with version control
- VII. Practice working collaboratively on a (non-coding) project
- VIII. Bonus: project management boards, issues, milestones

# What is Version Control?

Version Control is the practice of tracking and managing changes (to software, code, documentation, etc)

You're probably already using version control!

- If you've ever used track changes in a word document or google doc, that's a type of version control
- If you've ever made changes and then saved your file so that you have files such as “final\_draft1.docx”, “final\_draft2.docx” that is also version control

# So why use GitHub for version control?

## Cons:

- It isn't easy to set up
- It's a very different way of thinking about how you work
- Will require re-engineering how you save your code
- Substantial learning curve: can be painful!



# So why use GitHub for version control?



## Pros:

- Very easy to share code and data with collaborators
- Essentially provides a cloud backup and support for working from different computers
- Easy to work collaboratively with colleagues on coding (or other!) projects
- Code and data can be publicly available for publication purposes
- It's easy to get wrong - but it's hard to do something unrecoverable
- Project management tools can be built in

# What is a Repository?

“Repo”

Repository - a collection of files that are managed in a structured way

- These can be static - think of something like data archives or a collection of previously published journal articles
  - JSTOR
  - Dryad
- Or they can be dynamic - these files are subject to change, updates, edits, and versions. We will primarily be dealing with dynamic repositories. These are kind of like DropBox or GoogleDrive or OneDrive but much better!
  - GitHub
  - BitBucket
  - GitLab

# Why GitHub?

- Free!
- Public or private repos
- Advanced functionality (Issues, pull requests, automation, handles merging)
- Great for R development (particularly if you want to develop packages!)
- Fairly ubiquitous across academic, government, and industry institutions

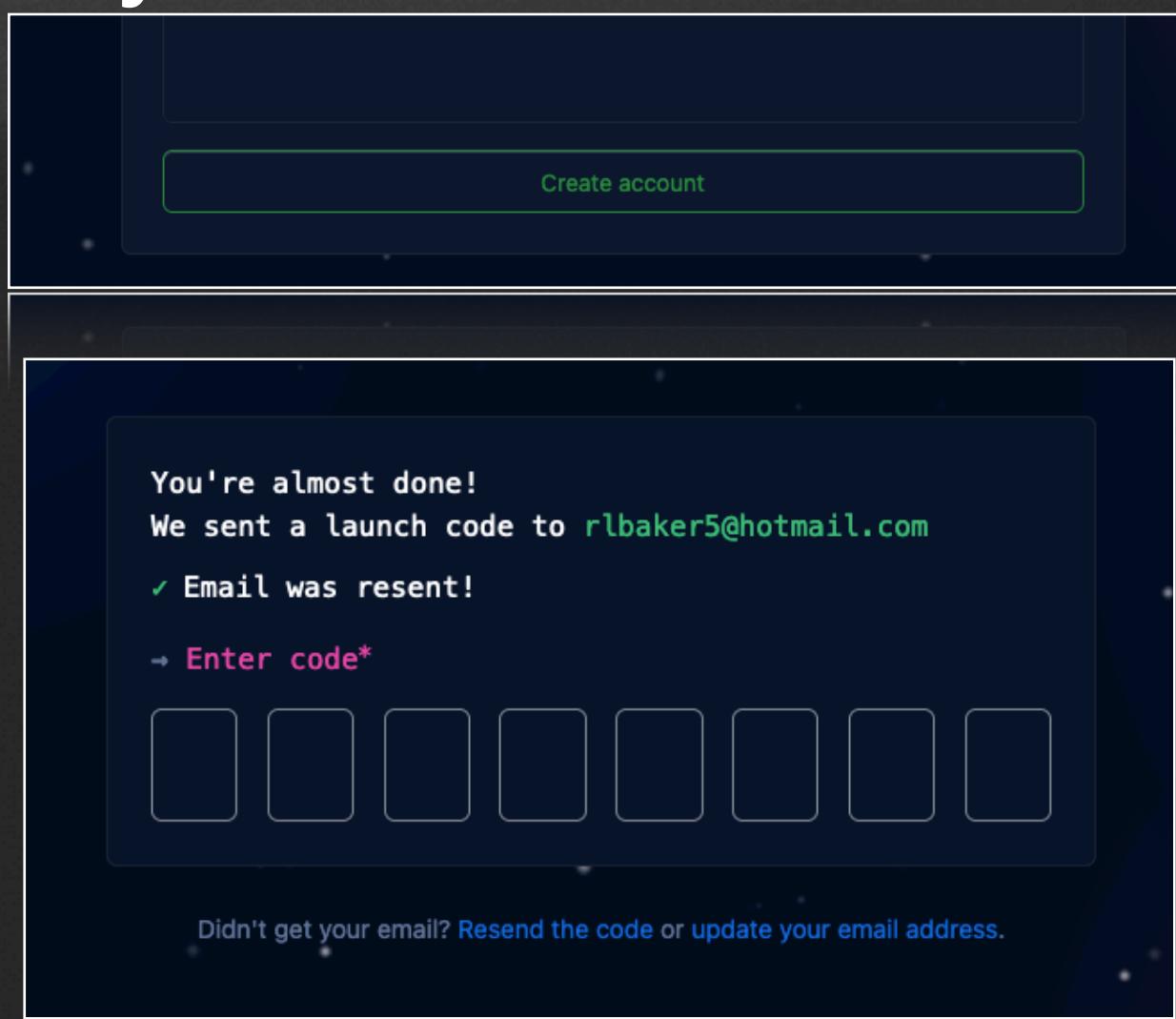
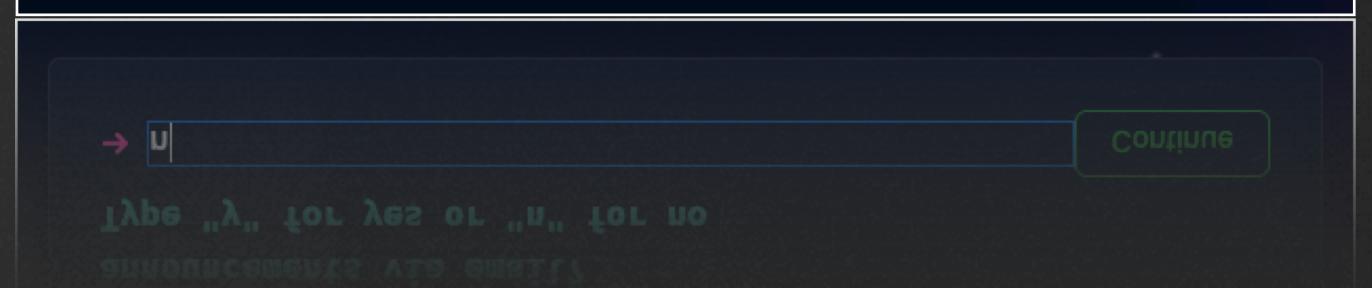
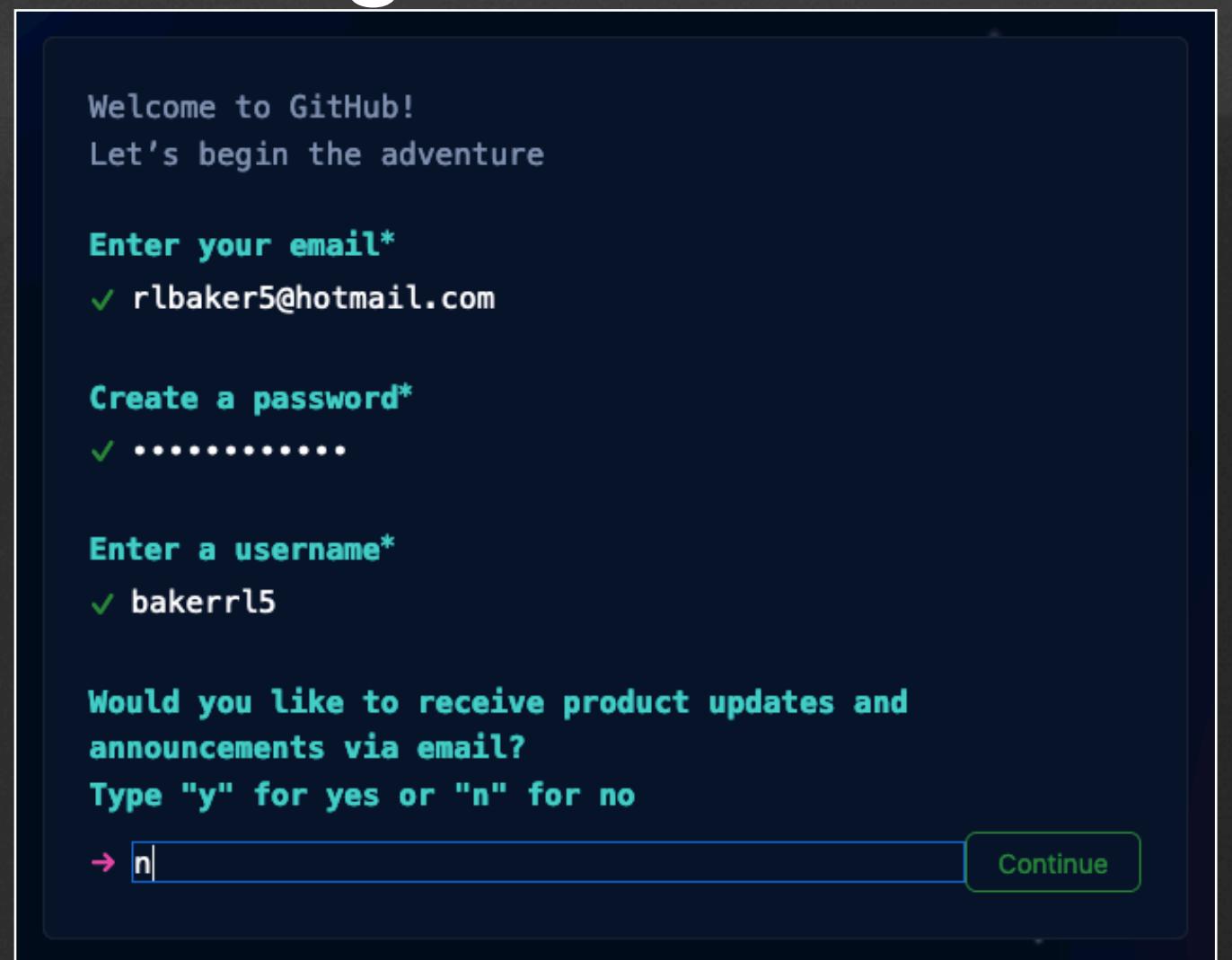
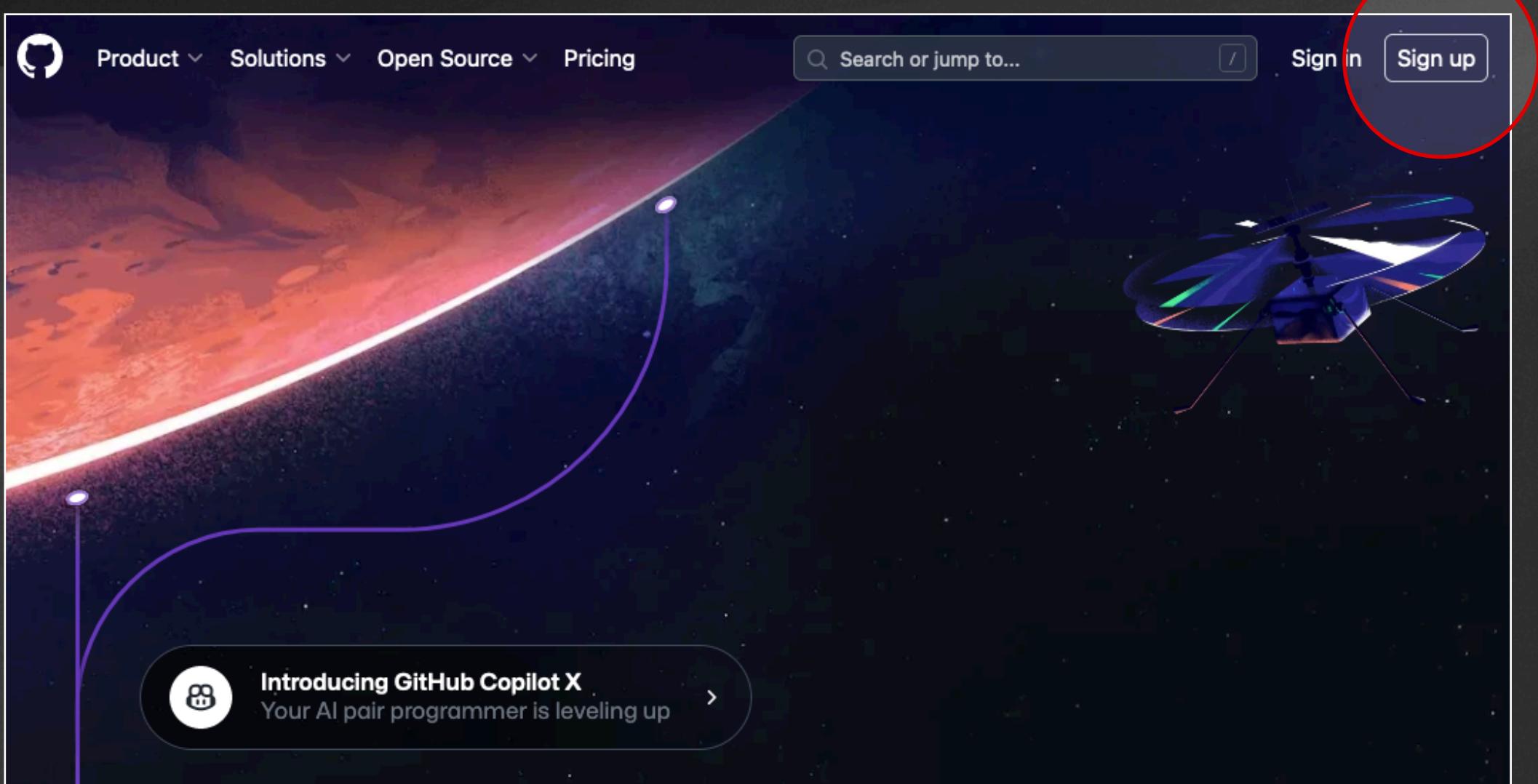
(Yes you can and should put your GitHub account on your CV!)

# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account**
- IV. Walk through and help installing/upgrading software
- V. Rstudio/GitHub integration
- VI. Workflows for collaborative coding with version control
- VII. Practice working collaboratively on a (non-coding) project
- VIII. Bonus: project management boards, issues, milestones

# Set up: Get a GitHub account (Step 1 of 4)

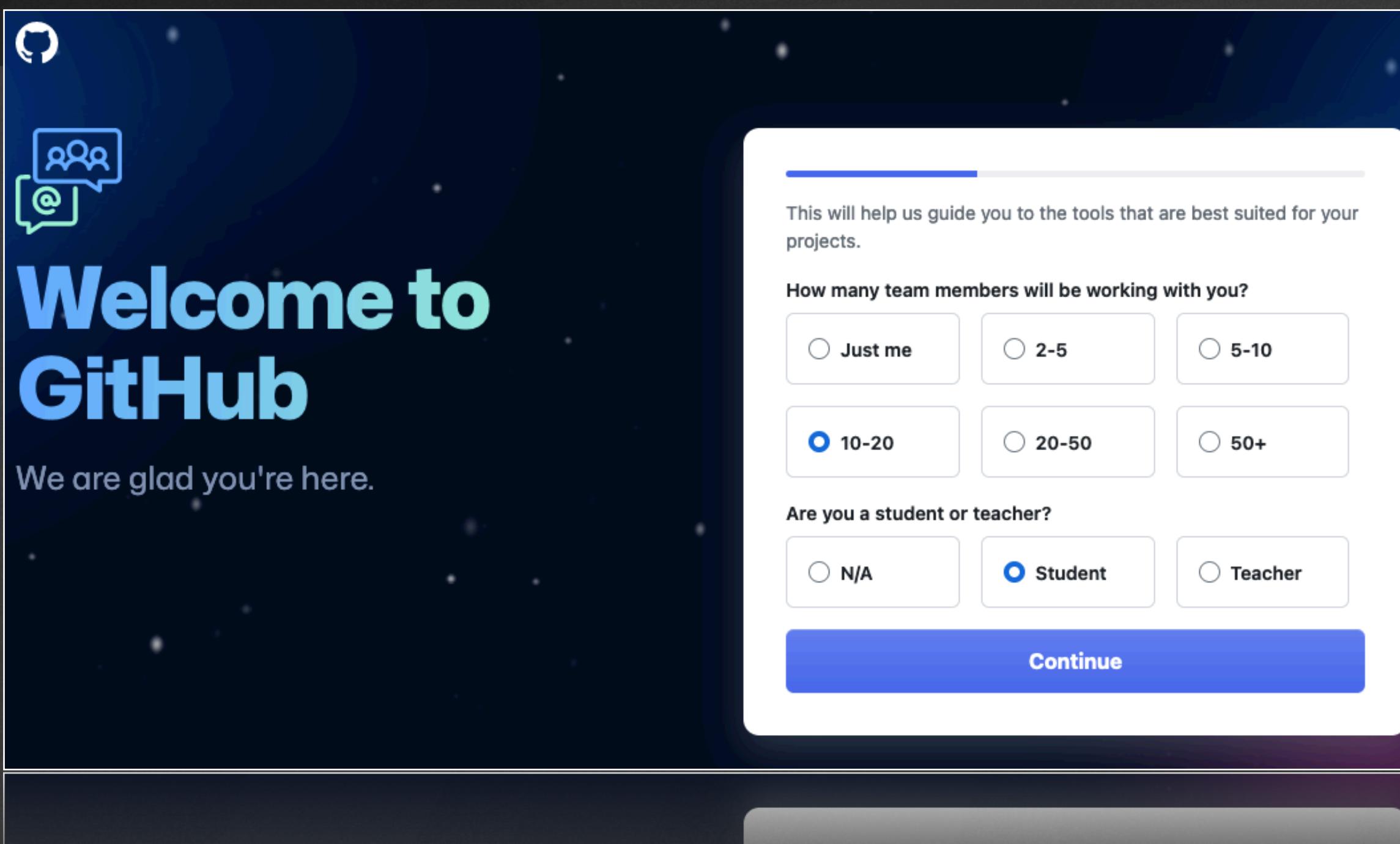
- Go to <https://github.com>
- Click “sign up” in the top right corner
- Supply Email & password
- Username can be changed later
- Click Create Account
- Enter code sent to your email



# Set up

## Get a GitHub account (Step 1 of 4)

- Fill out the Welcome questions. Not sure this really matters that much.
- Scroll down and select the **Free** version! (unless you don't like your money)



The image shows the GitHub pricing and benefits page. The top section features a large blue banner with the text "Learn to ship software like a pro." and a subtext "GitHub gives students free access to the best developer tools so they can learn by doing." Below this, there are two main sections: "Free" and "Get additional student benefits". The "Free" section lists three benefits:

- > Unlimited public/private repositories
- > 2,000 CI/CD minutes/month  
Free for public repositories
- > 500MB of Packages storage  
Free for public repositories

The "Get additional student benefits" section lists two benefits:

- GitHub Pro**
  - > Protect your branches  
Ensure that collaborators on your repository cannot make irrevocable changes to branches.
  - > Draft pull requests

# Set up

## Get a GitHub account (Step 1 of 4)

We will pause here until everyone is caught up

bakerrl5

Joined 2 days ago

Popular repositories

You don't have any public repositories yet.

1 contribution in the last year

Less More

This is your **contribution graph**. Your first square is for joining GitHub and you'll earn more as you make [additional contributions](#). More contributions means a higher contrast square color for that day. Over time, your chart might start looking something like this.

Contribution settings ▾

# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account
- IV. Walk through and help installing/upgrading software**
- V. Rstudio/GitHub integration
- VI. Workflows for collaborative coding with version control
- VII. Practice working collaboratively on a (non-coding) project
- VIII. Bonus: project management boards, issues, milestones

# Set up

## Get (or upgrade) R (step 2 of 4)

- If you already have R:
- Open R and see what version it is
- Or type:  
>R.version.string  
to retrieve the version
- If you have v4.3.1 (“Beagle Scouts”) you are ready to go
- If not, follow along to upgrade

```
R version 4.2.2 (2022-10-31) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Oh no! My R version is out of date!

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

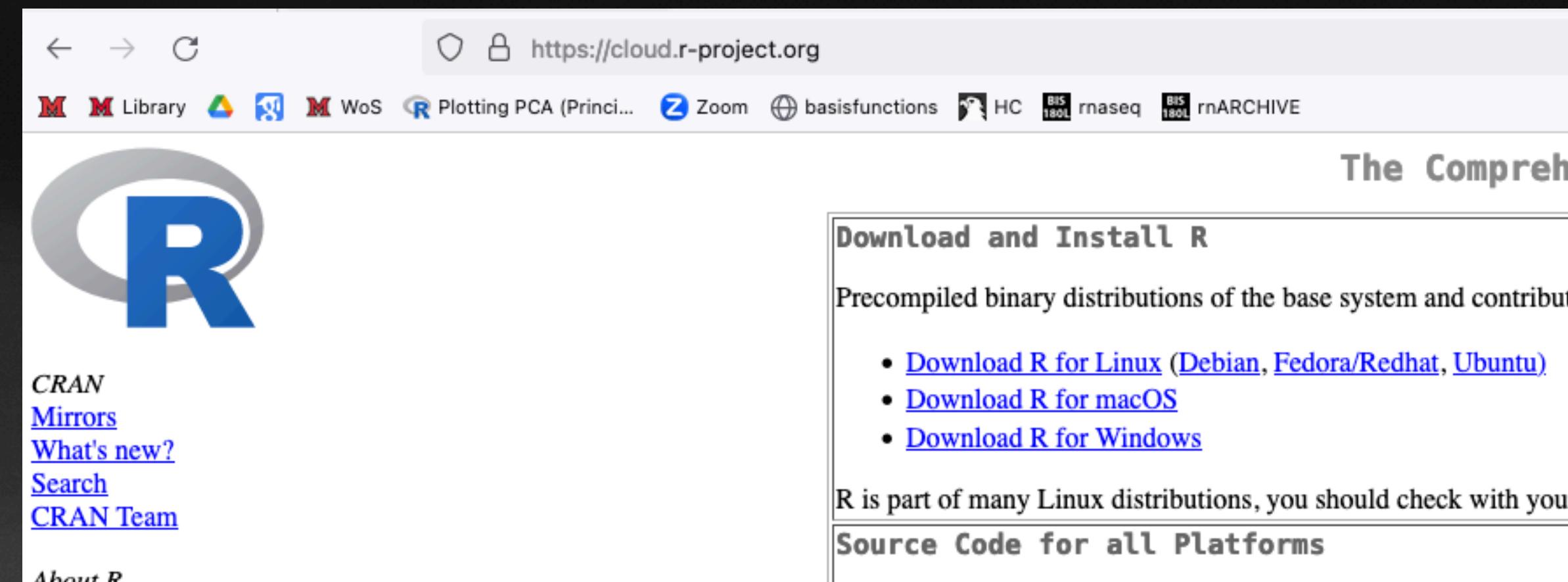
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> R.version.string
[1] "R version 4.2.2 (2022-10-31)"
> |
> |
[1] "R version 4.2.2 (2022-10-31)"
> R.version.string
```

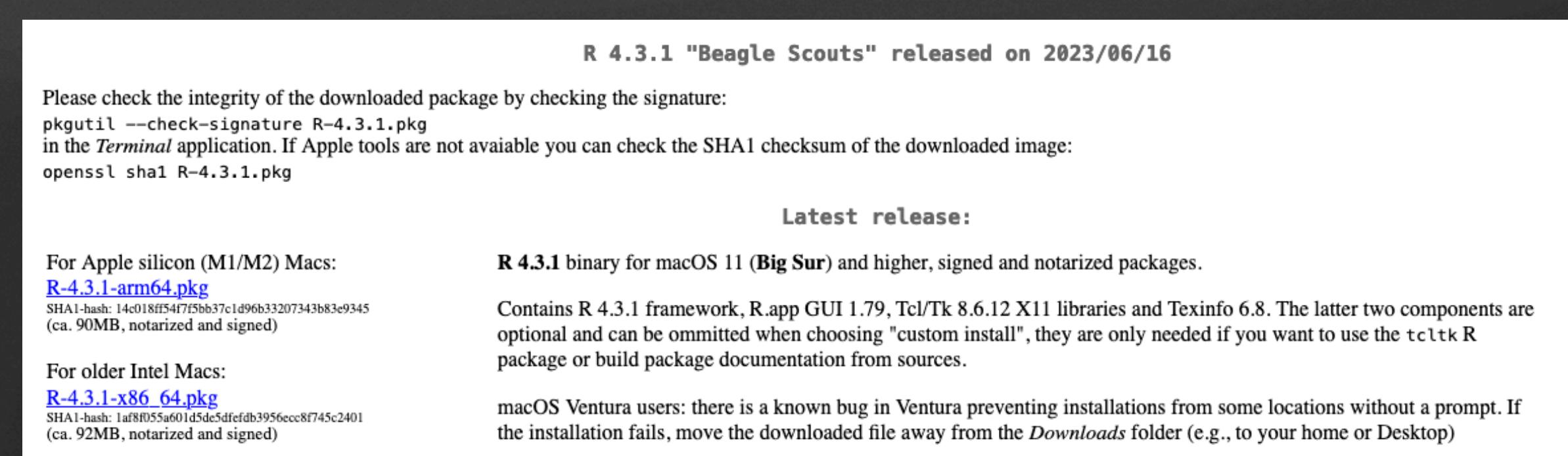
# Set up

## Get (or upgrade) R (step 2 of 4)

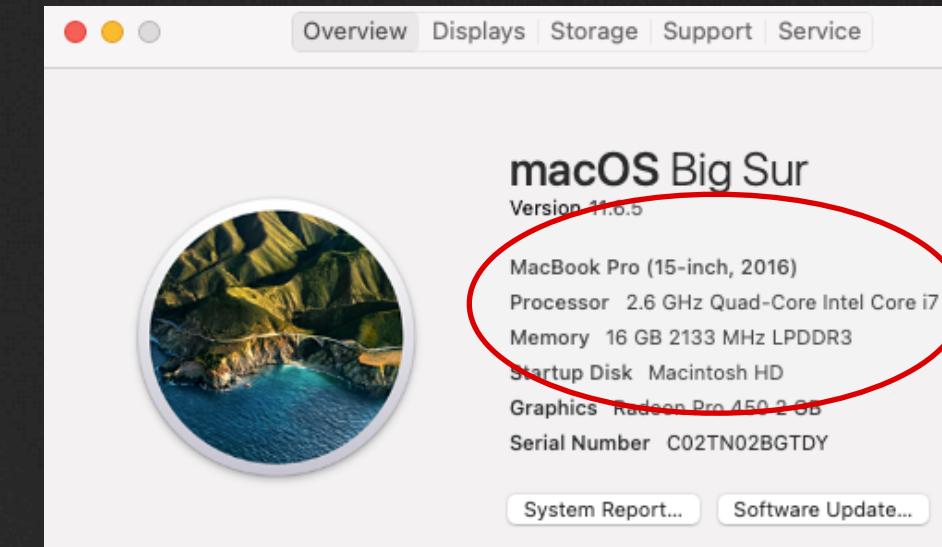
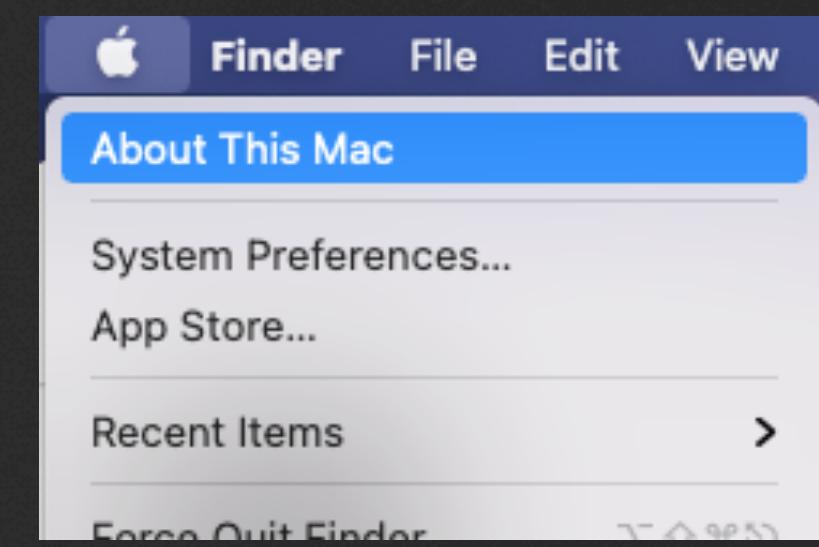
- To install or upgrade go to:  
<https://cloud.r-project.org>
- Click on the correct Download button for your operating system
- **Mac only:** you will need to choose the installation for your processor
- **Mac only:** To determine your processor, click on the Apple icon in the top left of your screen and choose “About this Mac”
- Download R and install using all the defaults



The screenshot shows the R Project website at <https://cloud.r-project.org>. The main navigation bar includes links for Library, WoS, Plotting PCA, basisfunctions, HC, rnaseq, rnARCHIVE, Zoom, and basisfunctions. A large R logo is on the left. On the right, there's a sidebar with "The Compreh" (partially visible), "Download and Install R" (with links for Linux, macOS, and Windows), and "Source Code for all Platforms". Below the sidebar, a message says "R is part of many Linux distributions, you should check with your distribution's package manager".



The screenshot shows the R 4.3.1 "Beagle Scouts" release page from June 16, 2023. It includes instructions for checking package integrity using `pkgutil --check-signature R-4.3.1.pkg` or `openssl sha1 R-4.3.1.pkg`. It highlights the "Latest release" as "R 4.3.1 binary for macOS 11 (Big Sur) and higher, signed and notarized packages". A note for macOS Ventura users mentions a bug preventing installations from some locations without a prompt.



# Set up

## Get (or upgrade) R (step 2 of 4)

- Check to make sure the install went well
- Open R
- Make sure it says:

**“R version 4.3.1  
(2023-06-16) —  
“Beagle Scouts”**

```
R version 4.3.1 (2023-06-16) -- "Beagle Scouts"  
Copyright (C) 2023 The R Foundation for Statistical Computing  
Platform: x86_64-apple-darwin20 (64-bit)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.
```

```
Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.
```

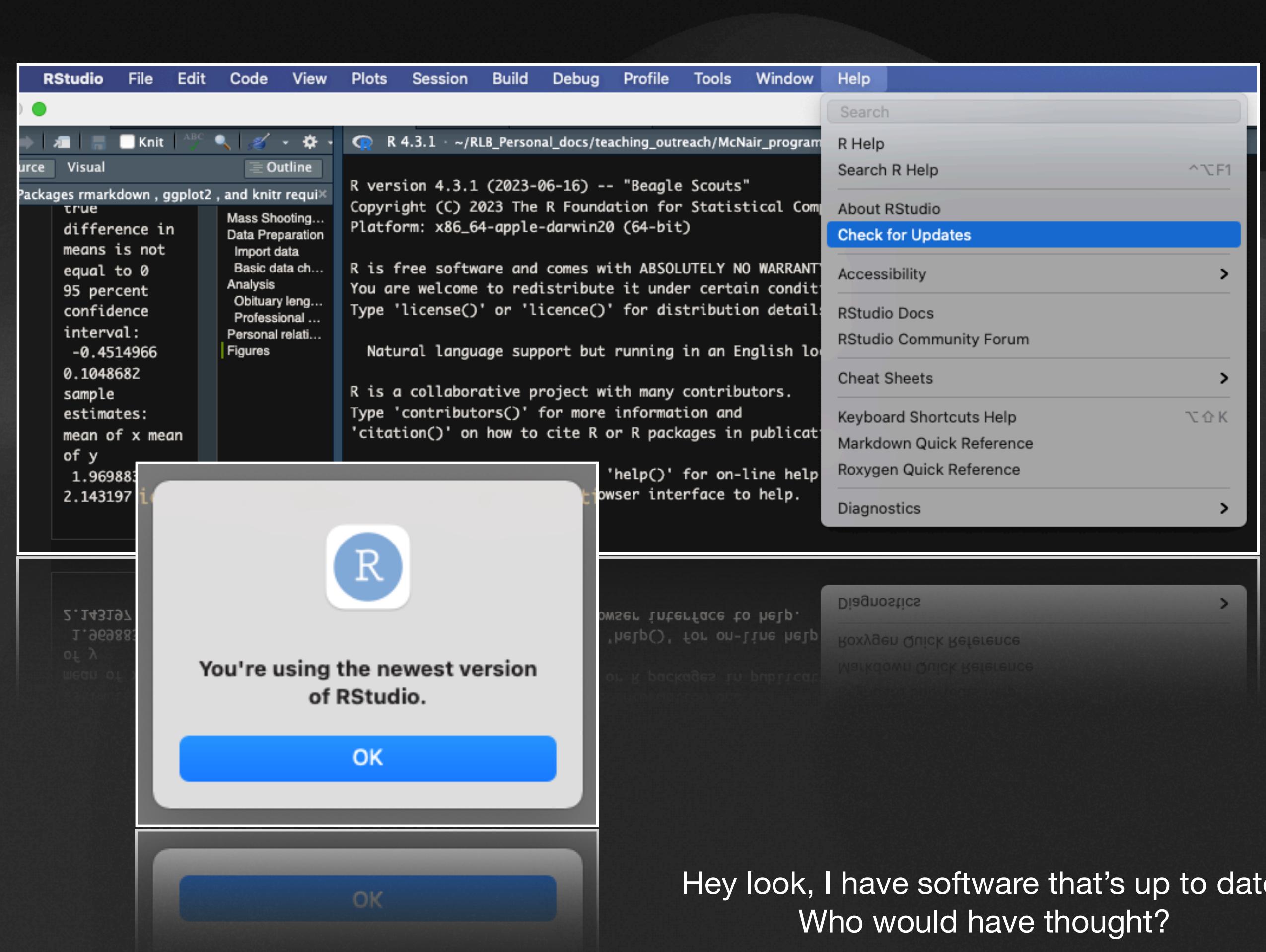
```
> |
```

**We will pause here until  
everyone is caught up**

# Set up

## Get (or upgrade) Rstudio (step 3 of 4)

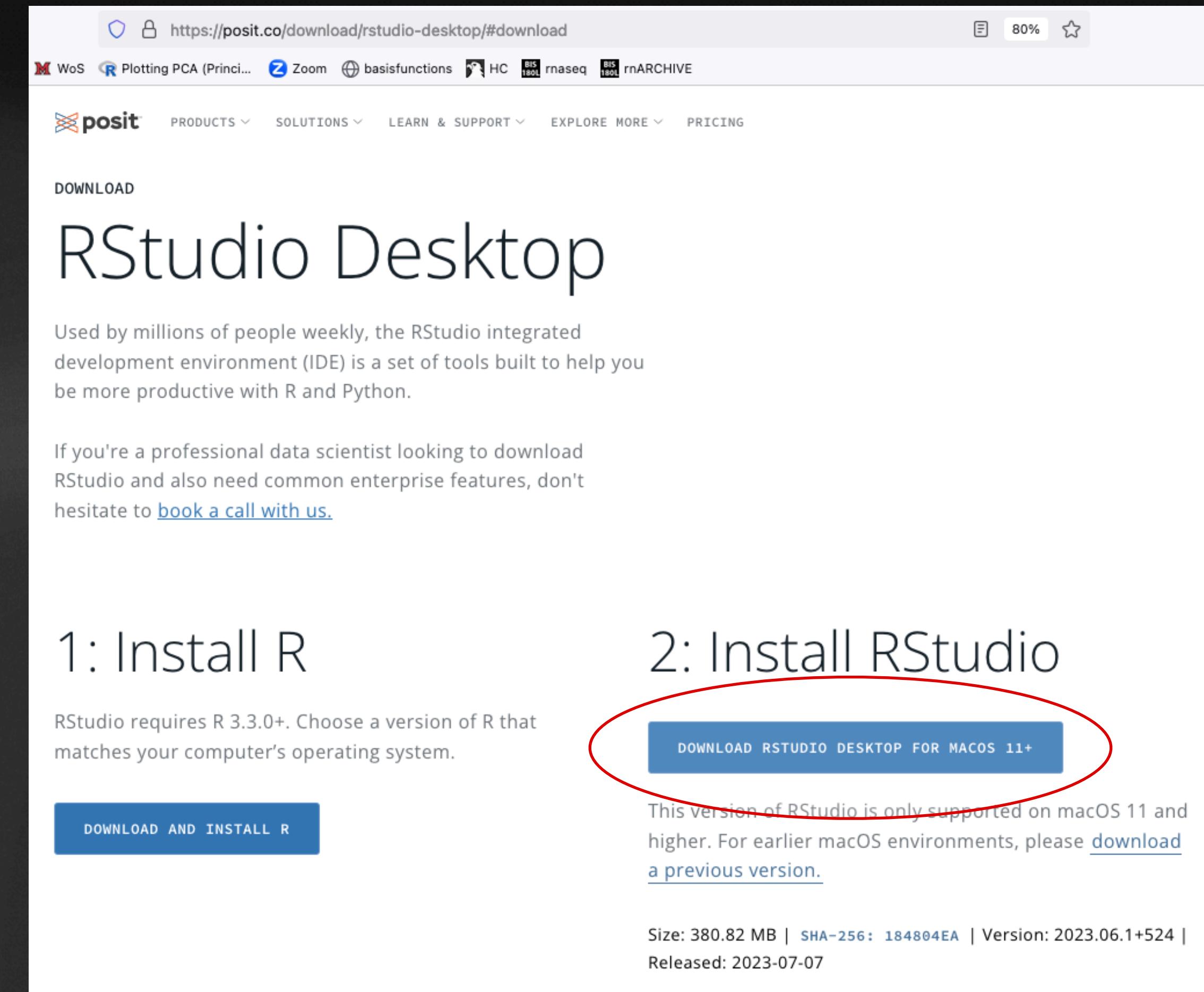
- If you already have R studio, make sure you have the latest version.
- Open Rstudio, go to the “Help” drop down menu at the top of the window
- Select “Check for Updates”
- If you have the latest version, you’re good to go!
- If you see “Updated Available” follow along to update Rstudio.



# Set up

## Get (or upgrade) Rstudio (step 3 of 4)

- Go to:  
<https://posit.co/download/rstudio-desktop/#download>
- Click “Download R Studio” (it will automatically detect your OS)
- Install Rstudio using all the defaults.

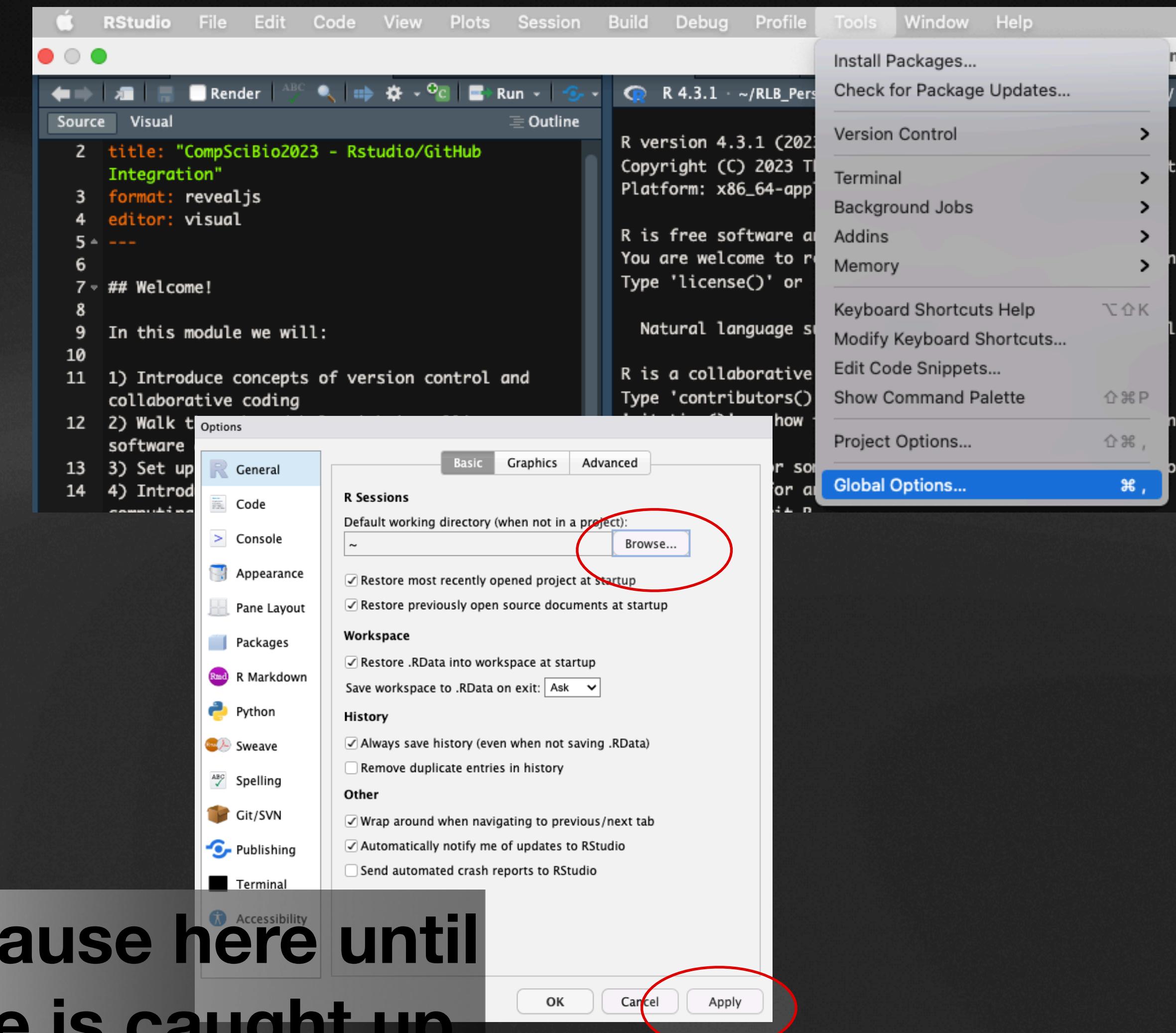


The screenshot shows the RStudio Desktop download page on the posit.co website. The URL in the browser is https://posit.co/download/rstudio-desktop/#download. The page features a large "RStudio Desktop" heading and a brief description: "Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python." Below this, there's a note for professional data scientists: "If you're a professional data scientist looking to download RStudio and also need common enterprise features, don't hesitate to [book a call with us](#)." The page is divided into two main sections: "1: Install R" on the left and "2: Install RStudio" on the right. A red oval highlights the "DOWNLOAD RSTUDIO DESKTOP FOR MACOS 11+" button in the "2: Install RStudio" section. Below it, a note states: "This version of RStudio is only supported on macOS 11 and higher. For earlier macOS environments, please [download a previous version](#)." At the bottom, technical details are provided: "Size: 380.82 MB | SHA-256: 184804EA | Version: 2023.06.1+524 | Released: 2023-07-07".

# Set up

## Get (or upgrade) Rstudio (step 3 of 4)

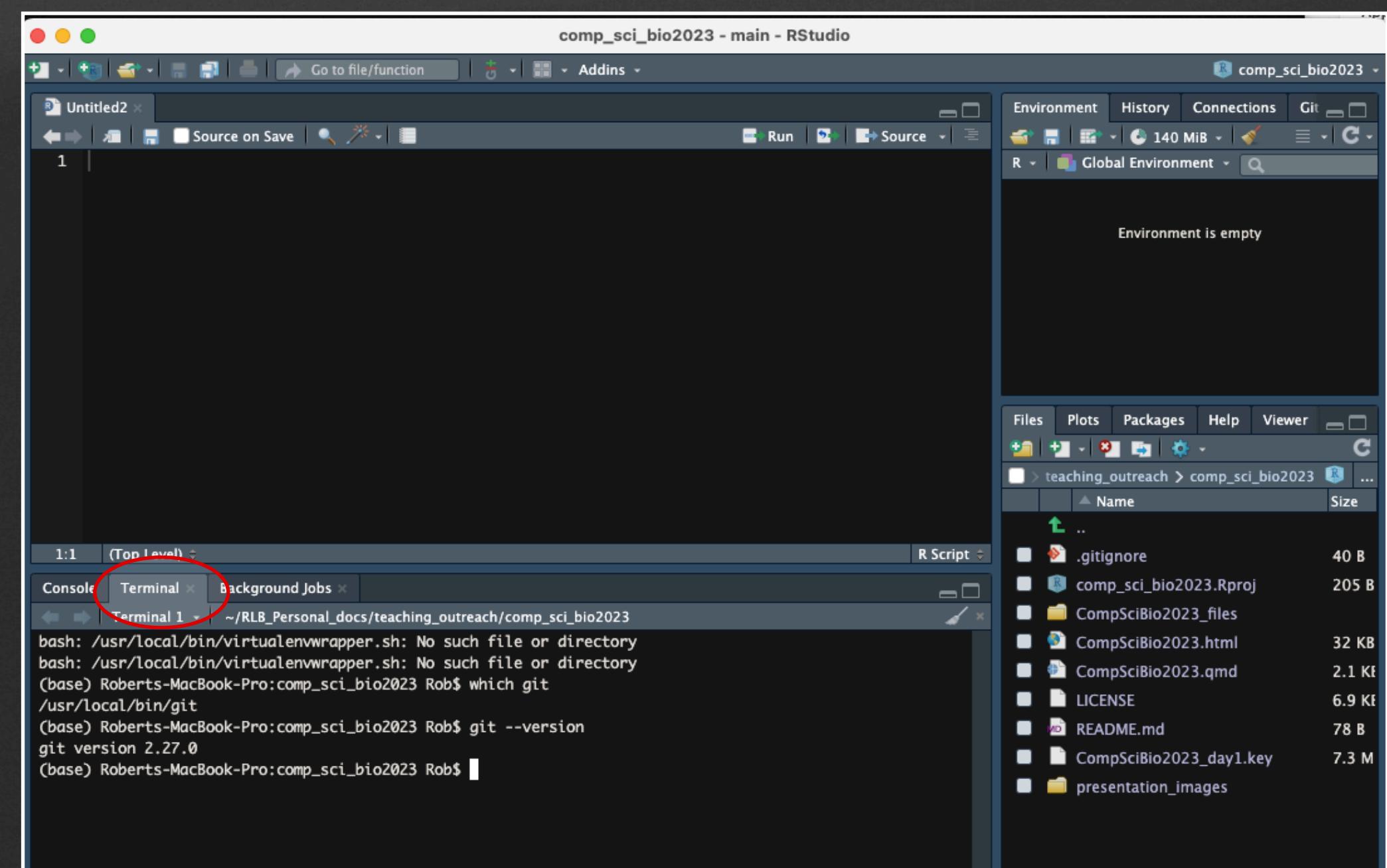
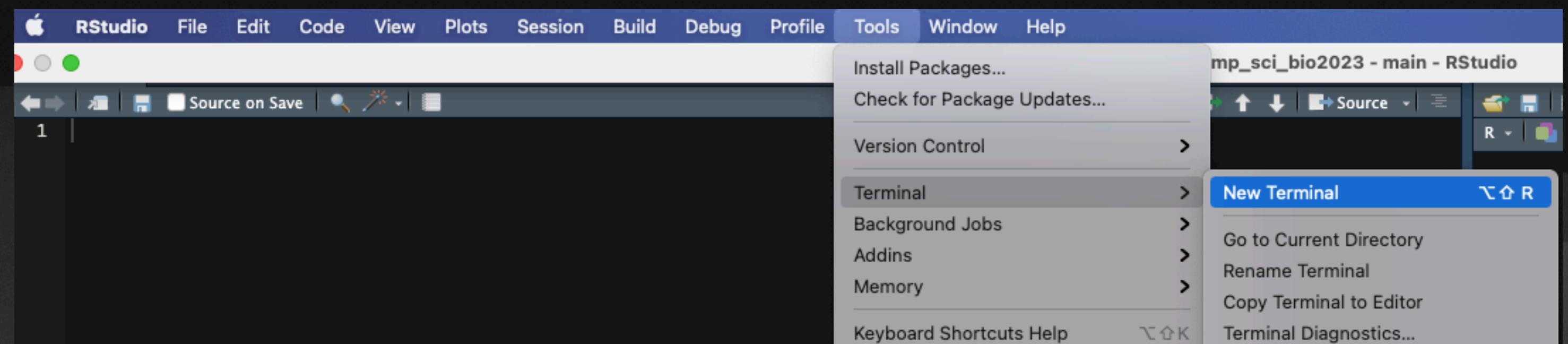
- Open Rstudio
- Make sure it is accessing the correct version of R (4.3.1)
- If it is not, select “Global Options”
- Click “Browse” and navigate to your newly installed version of R (4.3.1)
- Click “Apply” (you may need to restart Rstudio too)



# Set up

## Get (or upgrade) Git (step 4 of 4)

- Open Rstudio
- Find the “Terminal” Window
- Type:  
\$ which git  
response: “/usr/bin/git”
- Type:  
\$ git --version  
response: “git version 2.41.0”
- If you don’t have Git (response: “git: command not found”) or it is not version 2.41.0, follow along to install/upgrade



# Set up

## Get (or upgrade) Git (step 4 of 4)

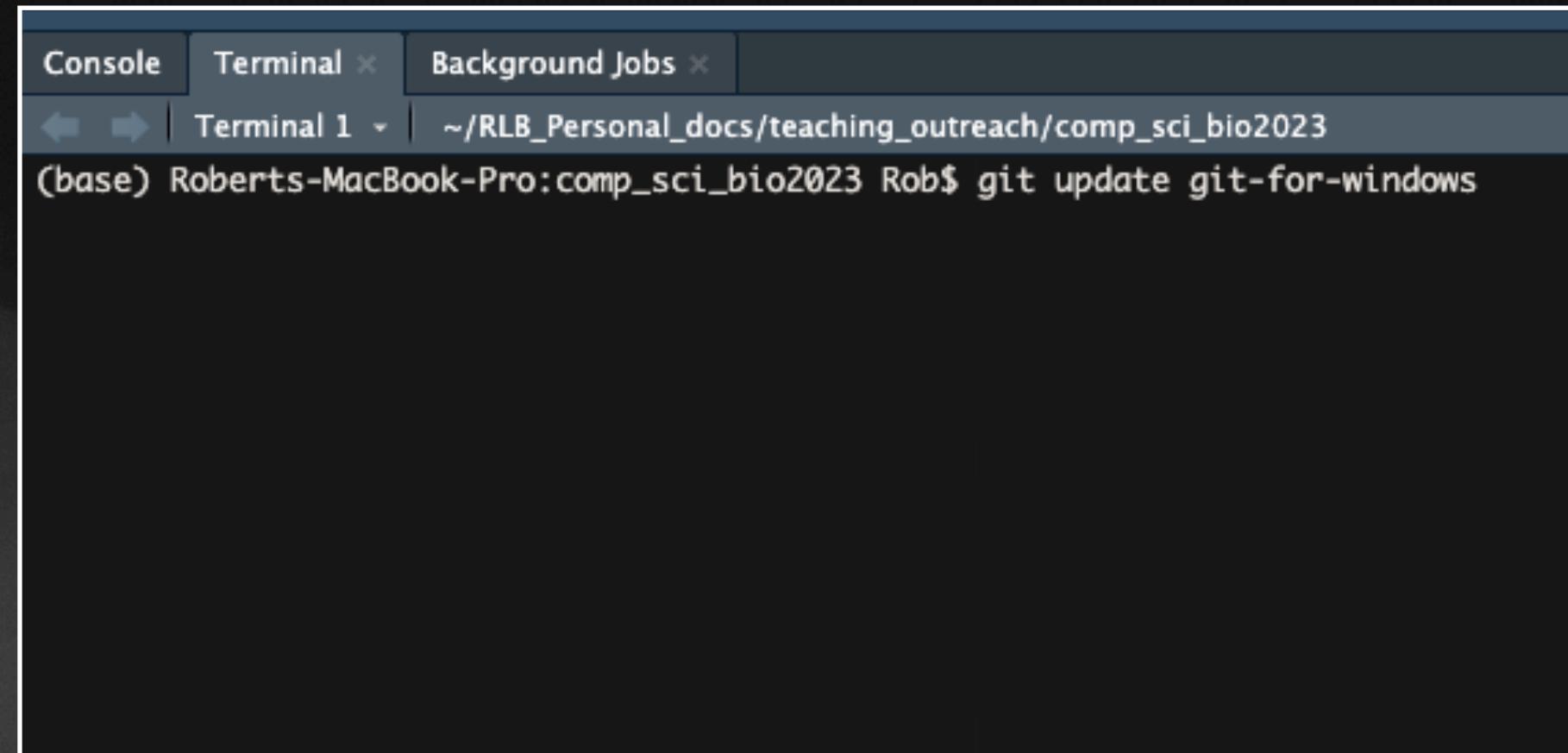
### WINDOWS: Upgrading Git

- In the terminal, type:

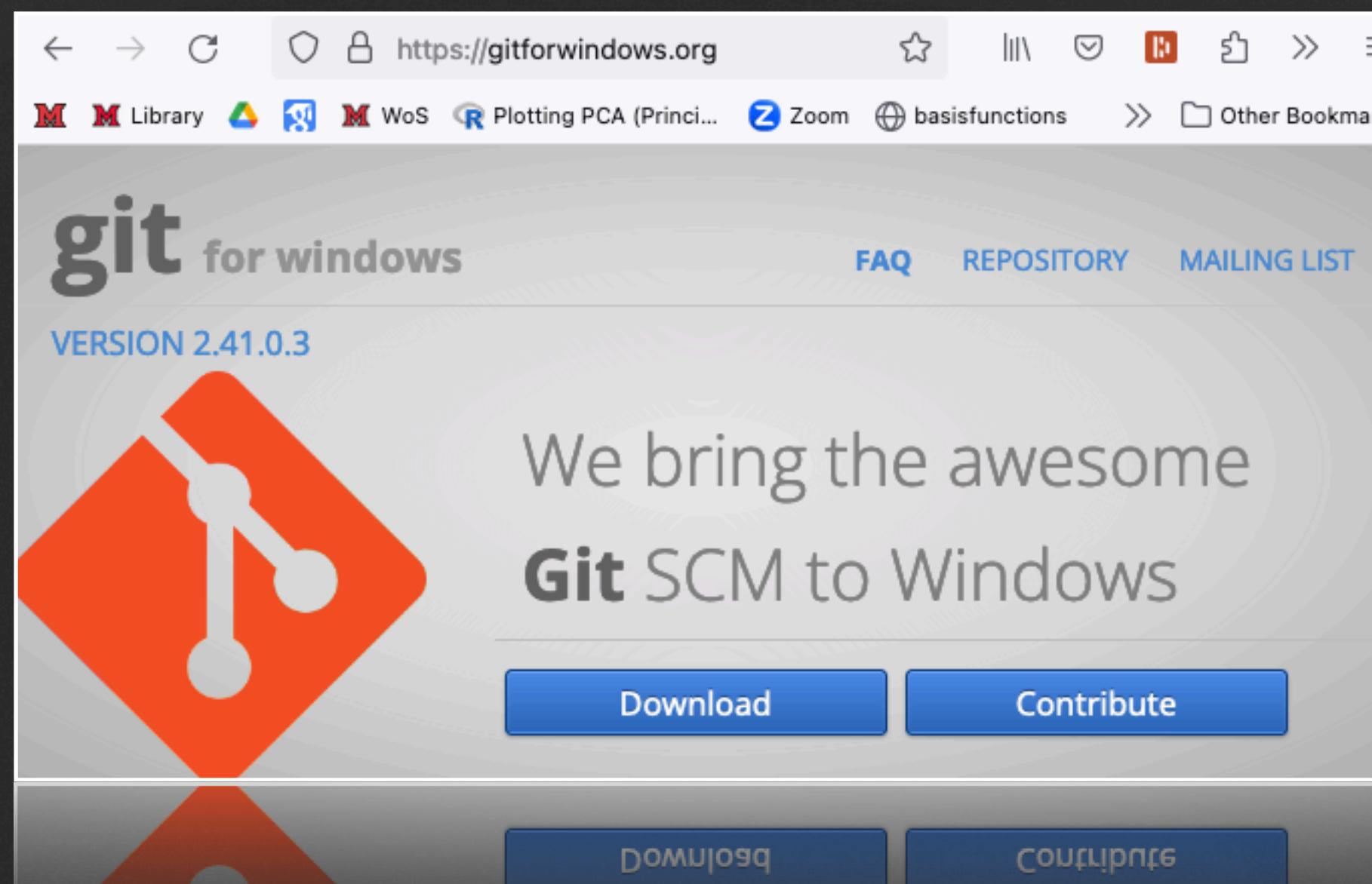
```
$ git update git-for-windows
```

### WINDOWS: Git Initial Install

- Go to <https://gitforwindows.org>
- Click Download & Install:
  - When asked to “adjust your PATH environment”, select “Git from the command line and also from 3rd-party software”
  - Install in the default location



A screenshot of a macOS terminal window titled "Terminal 1". The window shows the command `git update git-for-windows` being typed at the prompt. The terminal interface includes tabs for "Console", "Terminal", and "Background Jobs", and a status bar indicating the current directory is `~/RLB_Personal_docs/teaching_outreach/comp_sci_bio2023`.



# Set up

## Get (or upgrade) Git (step 4 of 4)

### MAC: Upgrade or Install Git

#### 1. Install home-brew

- Got to <https://brew.sh>
- In the terminal, type:

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)" (you can copy this from the brew.sh site)
```

- Enter your password when prompted

#### 2. Install git

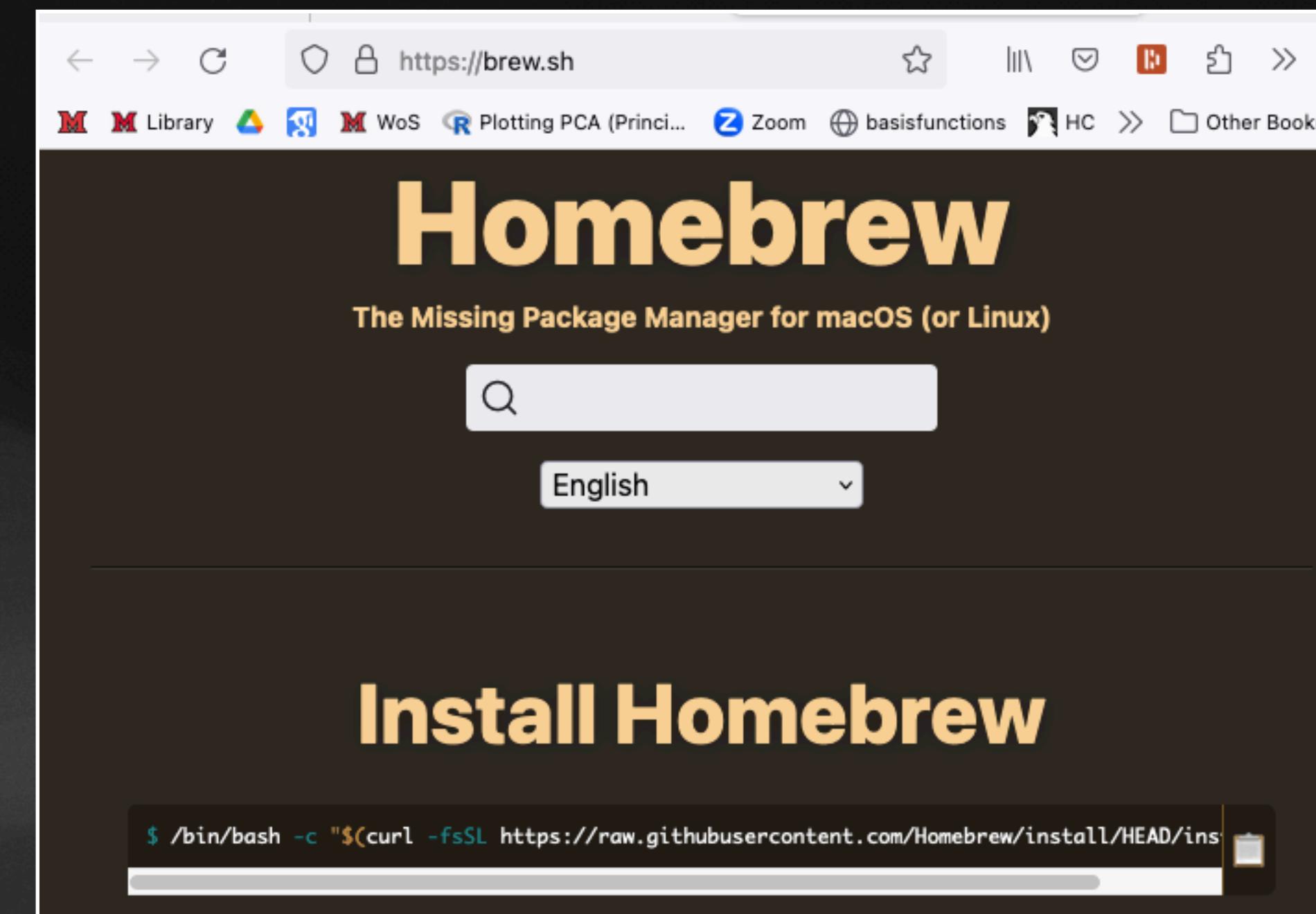
- In the Terminal type:

```
$ brew install git
```

#### 3. Check that you have the updated version

```
$ git --version
```

We will pause here until  
everyone is caught up



The screenshot shows a terminal window with three tabs: "Console", "Terminal", and "Background Jobs". The "Terminal" tab is active and shows the command `$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"` being run. The output of the command is displayed below, showing the installation of Homebrew and the cleanup of old Git versions. It also mentions that Tk GUIs and Subversion interoperability are now handled by other formulas.

```
==> Summary
🍺 /usr/local/Cellar/git/2.41.0_2: 1,633 files, 48.8MB
==> Running `brew cleanup git`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
Removing: /usr/local/Cellar/git/2.27.0... (1,478 files, 48.5MB)
==> Caveats
==> git
The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the `git-gui` formula.
Subversion interoperability (git-svn) is now in the `git-svn` formula.

zsh completions and functions have been installed to:
/usr/local/share/zsh/site-functions
(base) Roberts-MacBook-Pro:comp_sci_bio2023 Rob$ git --version
git version 2.41.0
(base) Roberts-MacBook-Pro:comp_sci_bio2023 Rob$
```

# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account
- IV. Walk through and help installing/upgrading software
- V. Rstudio/GitHub integration
- VI. Version Control and Collaborative Coding Workflows
- VII. Practice working collaboratively on a (non-coding) project
- VIII. Bonus: project management boards, issues, milestones

# Rstudio/GitHub integration

## Configure Git

From the Terminal pane in Rstudio  
run these four lines:

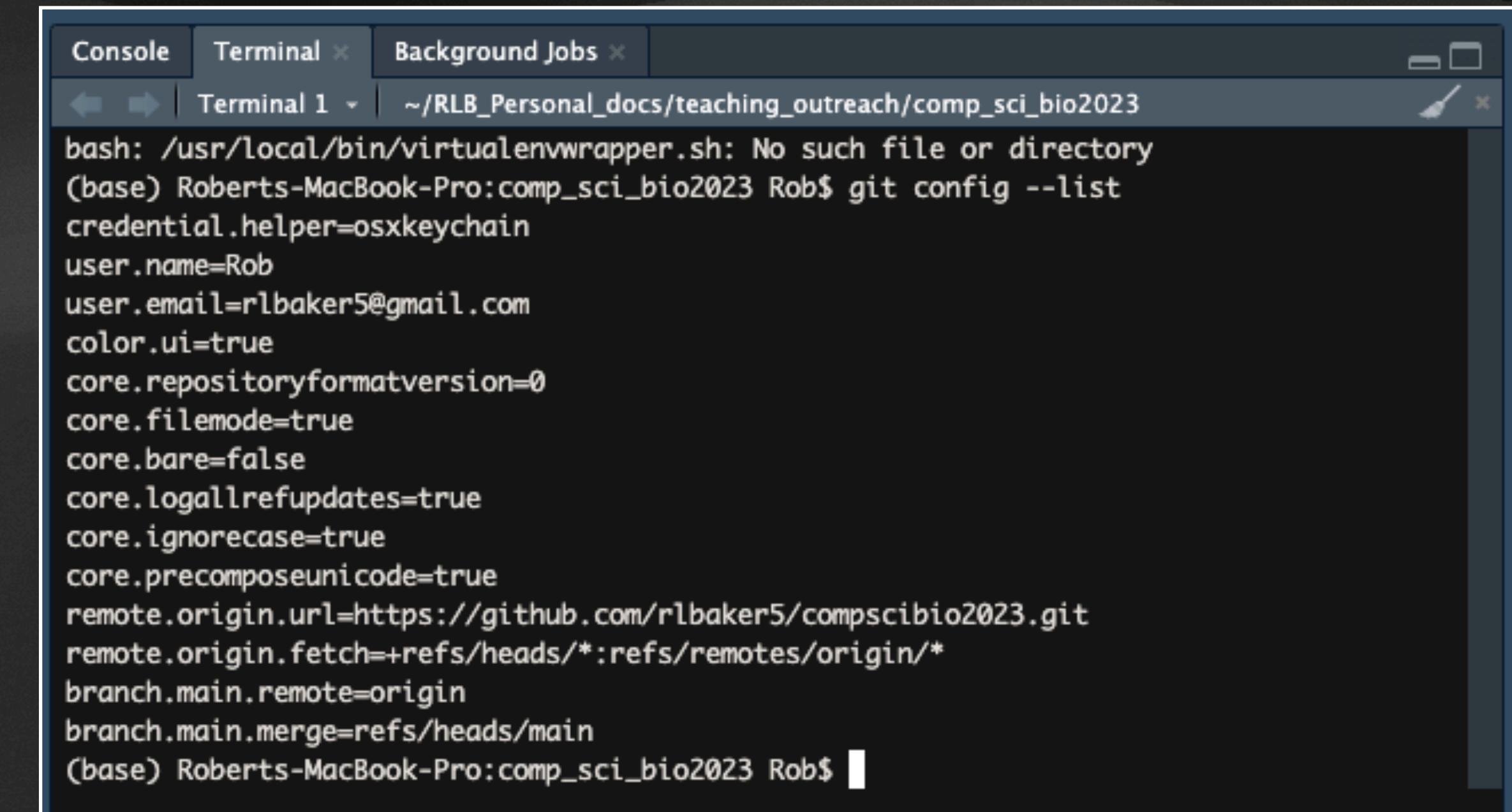
(For user.name and user.email use your GitHub username  
and email associated with your GitHub account)

```
git config --global user.name 'rlbaker5'
```

```
git config --global user.email 'rlbaker5@gmail.com'
```

```
git config --global --list
```

```
git config --global init.defaultBranch main
```



```
Console Terminal x Background Jobs x
Terminal 1 ~~/RLB_Personal_docs/teaching_outreach/comp_sci_bio2023
bash: /usr/local/bin/virtualenvwrapper.sh: No such file or directory
(base) Roberts-MacBook-Pro:comp_sci_bio2023 Rob$ git config --list
credential.helper=osxkeychain
user.name=Rob
user.email=rlbaker5@gmail.com
color.ui=true
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
remote.origin.url=https://github.com/rlbaker5/compscibio2023.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main
(base) Roberts-MacBook-Pro:comp_sci_bio2023 Rob$
```

# Rstudio/GitHub Integration

## Get access to Git

- You can use SSL or HTTPS
- We will use HTTPS because it's just easier
- You can switch back and forth at any time if you choose

# Rstudio/GitHub Integration

## Get access to Git

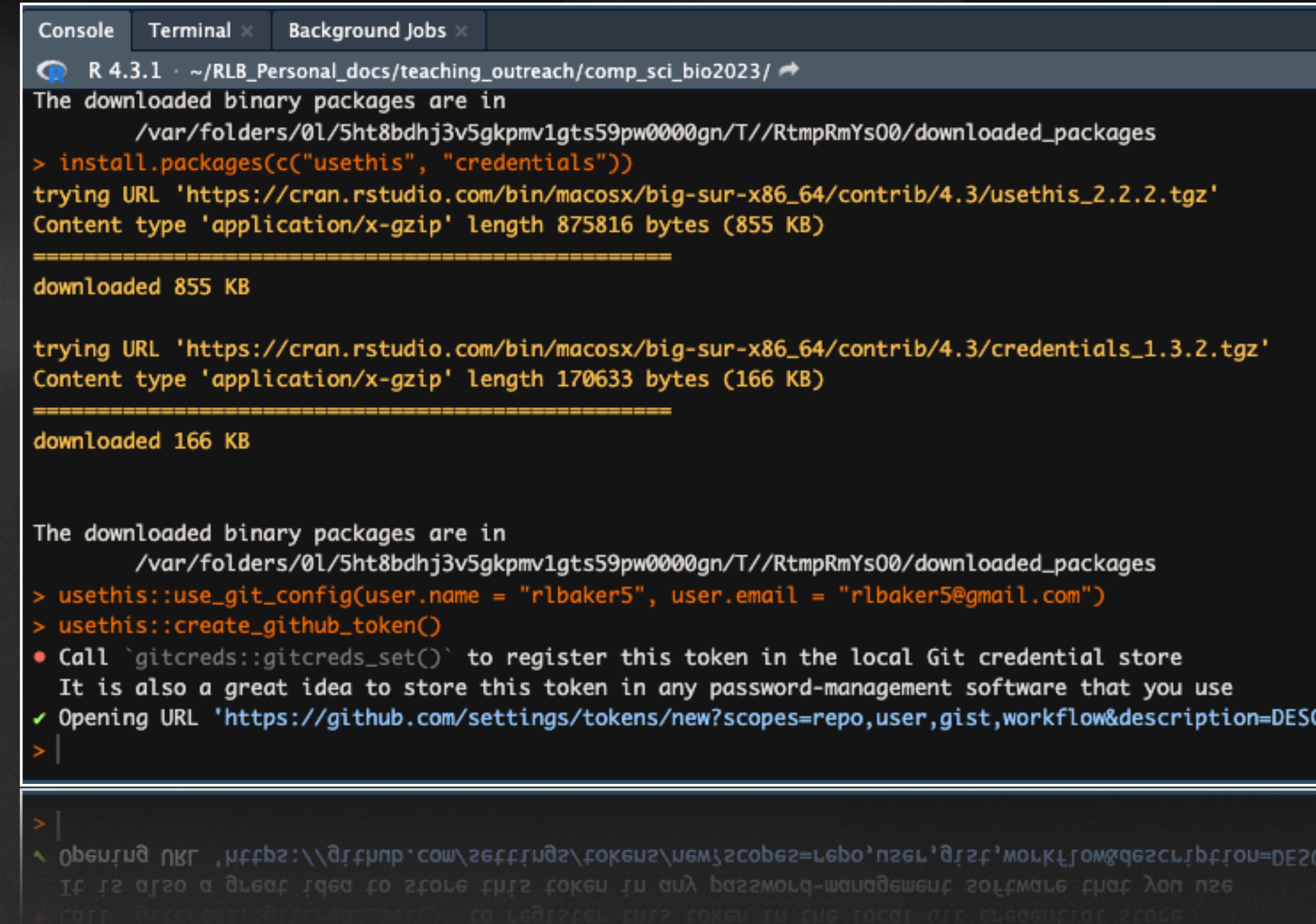
From the Console in Rstudio run the following 3 lines of code:

```
>install.packages(c("usethis",  
"credentials"))
```

```
>usethis::use_git_config(user.name =  
"YourName", user.email = "your@mail.com"  
(Use your GitHub username and email address)
```

```
>usethis::create_github_token()
```

- This should open a browser window for GitHub.com



The screenshot shows the RStudio interface with the 'Console' tab selected. The console window displays the following R session:

```
Console Terminal × Background Jobs ×  
R 4.3.1 · ~/RLB_Personal_docs/teaching_outreach/comp_sci_bio2023/ ↵  
The downloaded binary packages are in  
/var/folders/01/5ht8bdhj3v5gkpmv1gts59pw0000gn/T//RtmpRmYs00/downloaded_packages  
> install.packages(c("usethis", "credentials"))  
trying URL 'https://cran.rstudio.com/bin/macosx/big-sur-x86_64/contrib/4.3/usethis_2.2.2.tgz'  
Content type 'application/x-gzip' length 875816 bytes (855 KB)  
=====  
downloaded 855 KB  
  
trying URL 'https://cran.rstudio.com/bin/macosx/big-sur-x86_64/contrib/4.3/credentials_1.3.2.tgz'  
Content type 'application/x-gzip' length 170633 bytes (166 KB)  
=====  
downloaded 166 KB  
  
The downloaded binary packages are in  
/var/folders/01/5ht8bdhj3v5gkpmv1gts59pw0000gn/T//RtmpRmYs00/downloaded_packages  
> usethis::use_git_config(user.name = "rlbaker5", user.email = "rlbaker5@gmail.com")  
> usethis::create_github_token()  
• Call `gitcreds::gitcreds_set()` to register this token in the local Git credential store  
It is also a great idea to store this token in any password-management software that you use  
✓ Opening URL 'https://github.com/settings/tokens/new?scopes=repo,user,gist,workflow&description=DESC'  
>  
  
>  
↳ Open in browser https://github.com/settings/tokens/new?scopes=repo,user,gist,workflow&description=DESC  
If it is also a great idea to store this token in any password-management software that you use  
• Call `gitcreds::gitcreds_set()` to register this token in the local Git credential store
```

# Rstudio/GitHub Integration

## Get access to Git

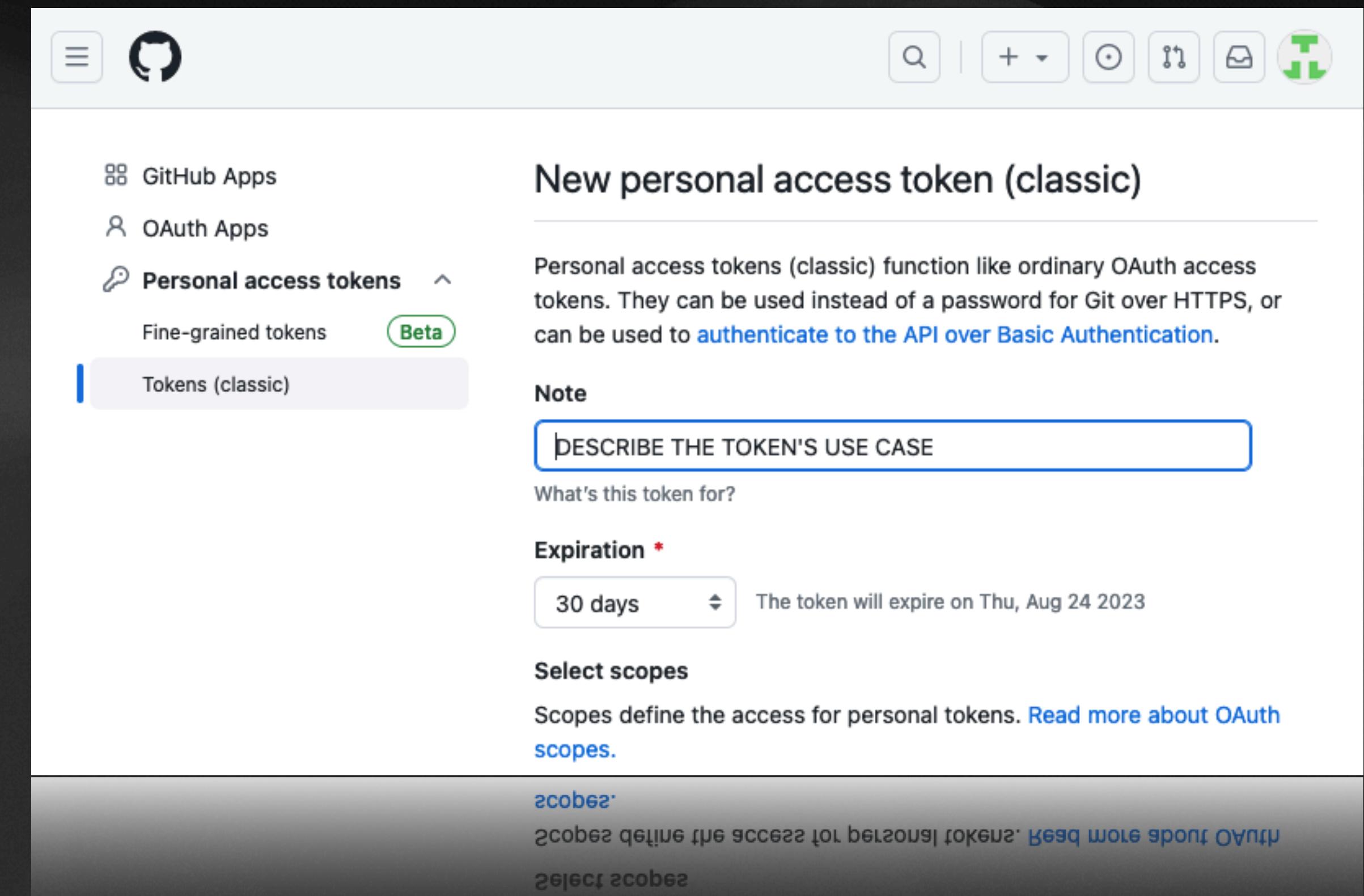
After signing in, you will be brought to a “New personal access token (classic)” page

Describe the token (“compscibio2023”)

Set an expiration date

Use the default “Scopes”

Scroll down and click “Generate Token”

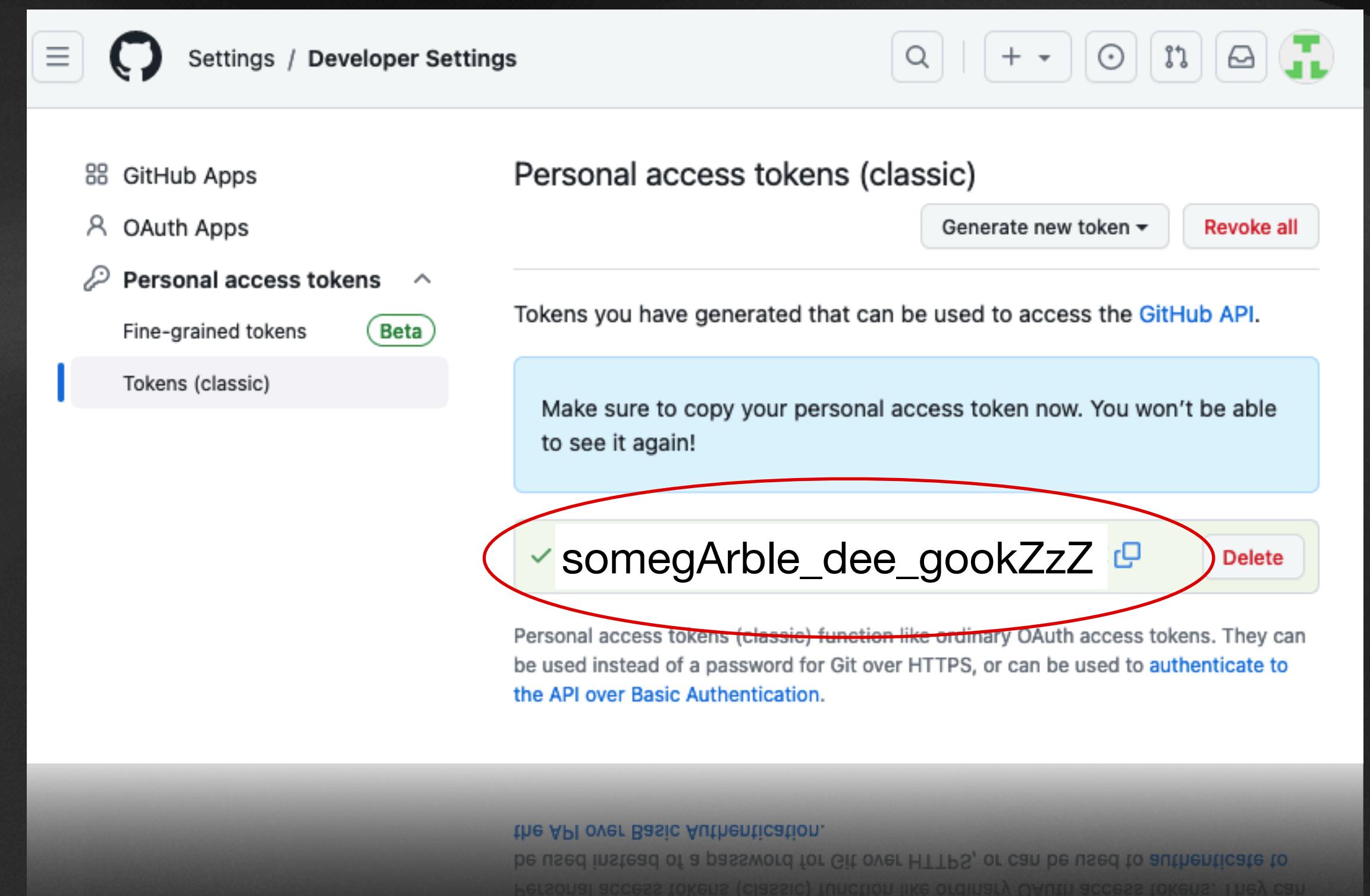


# Rstudio/GitHub Integration

## Get access to Git

Copy your personal access token (PAT) to your clipboard!

(Click the two linked squares next to the token)



# Rstudio/GitHub Integration

## Get access to Git

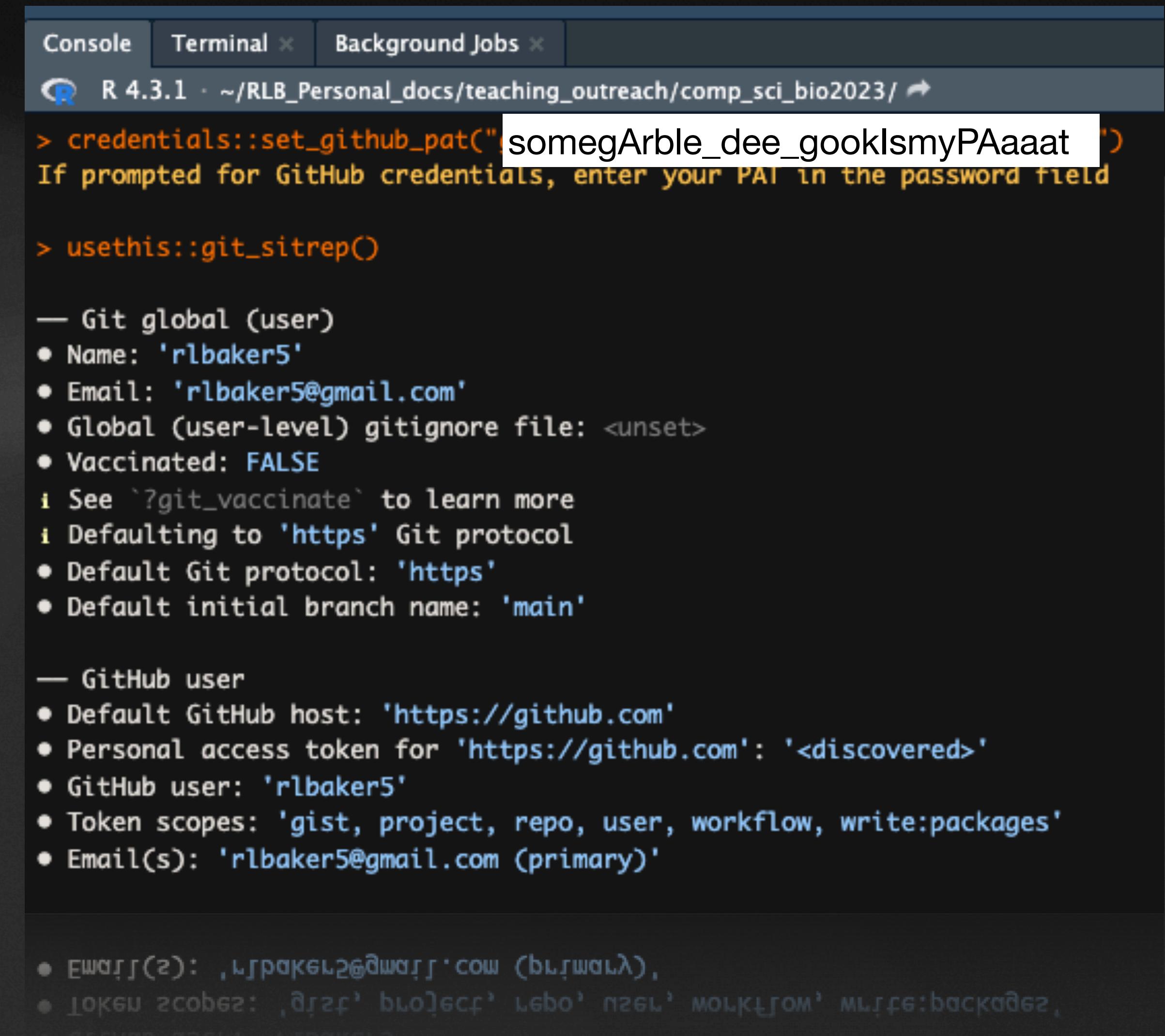
Back in the Rstudio console....

```
>credentials::set_github_pat("<paste_your_pat>")
```

#don't forget the “” around your PAT!

(If prompted, enter your password)

```
>usethis::git_sitrep()
```



The screenshot shows the RStudio interface with the R console tab active. The console window displays the following R code and its execution results:

```
Console Terminal × Background Jobs ×
R 4.3.1 · ~/RLB_Personal_docs/teaching_outreach/comp_sci_bio2023/ ↵
> credentials::set_github_pat("somegArble_dee_gooklsmyPAaaat")
If prompted for GitHub credentials, enter your PAT in the password field

> usethis::git_sitrep()

— Git global (user)
• Name: 'rlbaker5'
• Email: 'rlbaker5@gmail.com'
• Global (user-level) gitignore file: <unset>
• Vaccinated: FALSE
  i See `?git_vaccinate` to learn more
  i Defaulting to 'https' Git protocol
• Default Git protocol: 'https'
• Default initial branch name: 'main'

— GitHub user
• Default GitHub host: 'https://github.com'
• Personal access token for 'https://github.com': '<discovered>'
• GitHub user: 'rlbaker5'
• Token scopes: 'gist, project, repo, user, workflow, write:packages'
• Email(s): 'rlbaker5@gmail.com (primary)'

• Email(s): , rlbaker5@gmail.com (primary),
• Token scopes: , gist, project, repo, user, workflow, write:packages,
```

# Rstudio/GitHub Integration

## Get access to Git

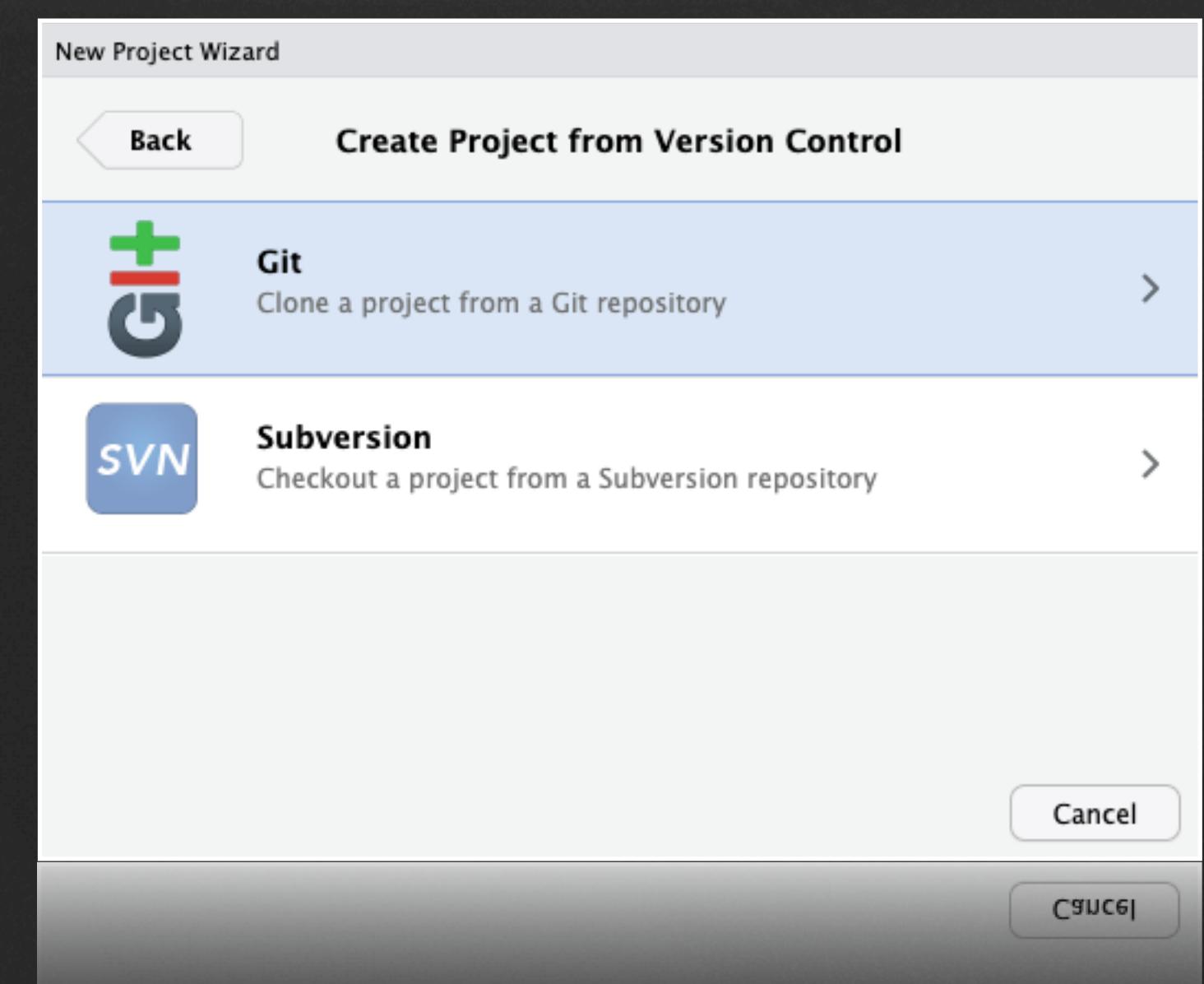
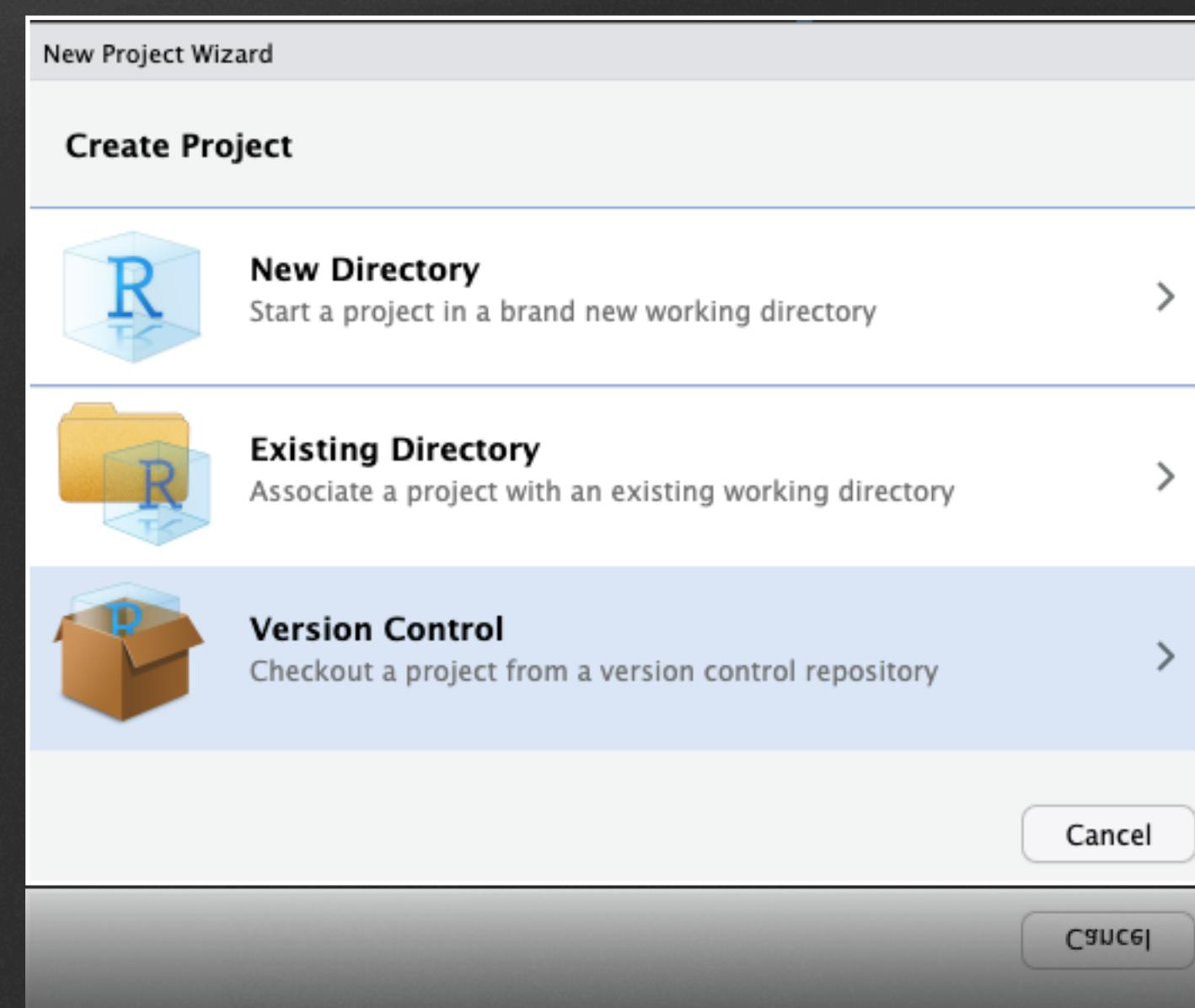
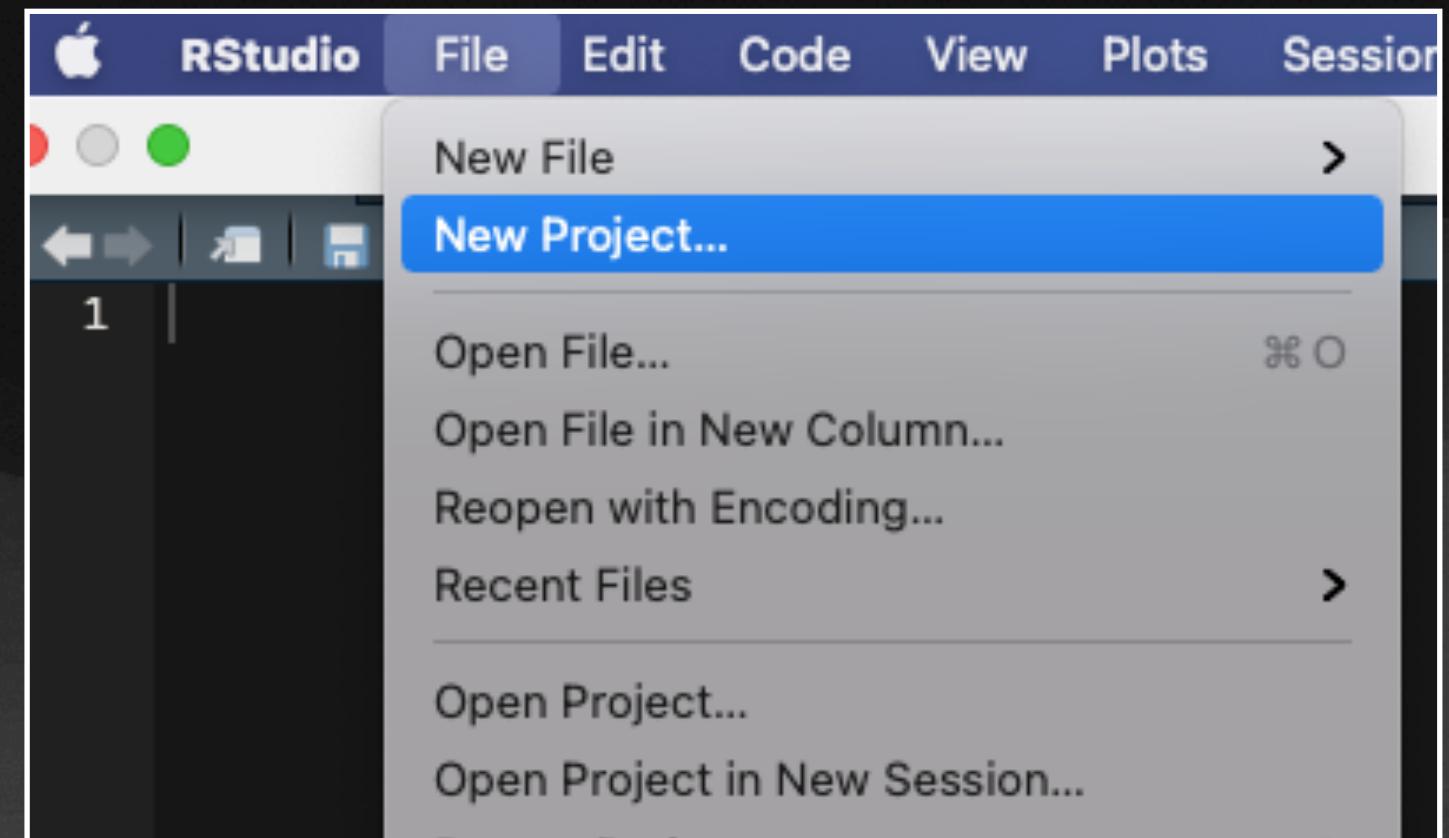
Did it work?

- In Rstudio go to File > New Project

If you see an option for Version Control, that is good!

- Select “Version Control”

If you see an option to “Clone a project from a Git repository” that is good!



# Rstudio/GitHub Integration

## Get access to Git

Ruh-roh. It didn't work . . .

1. Quit and **restart Rstudio**. . . seriously, just do it. If that didn't fix it. . .

2. Rstudio needs to be able to find git. Usually this is automatic, but you can do it manually. Find your git executable:

- From the Terminal in Rstudio:

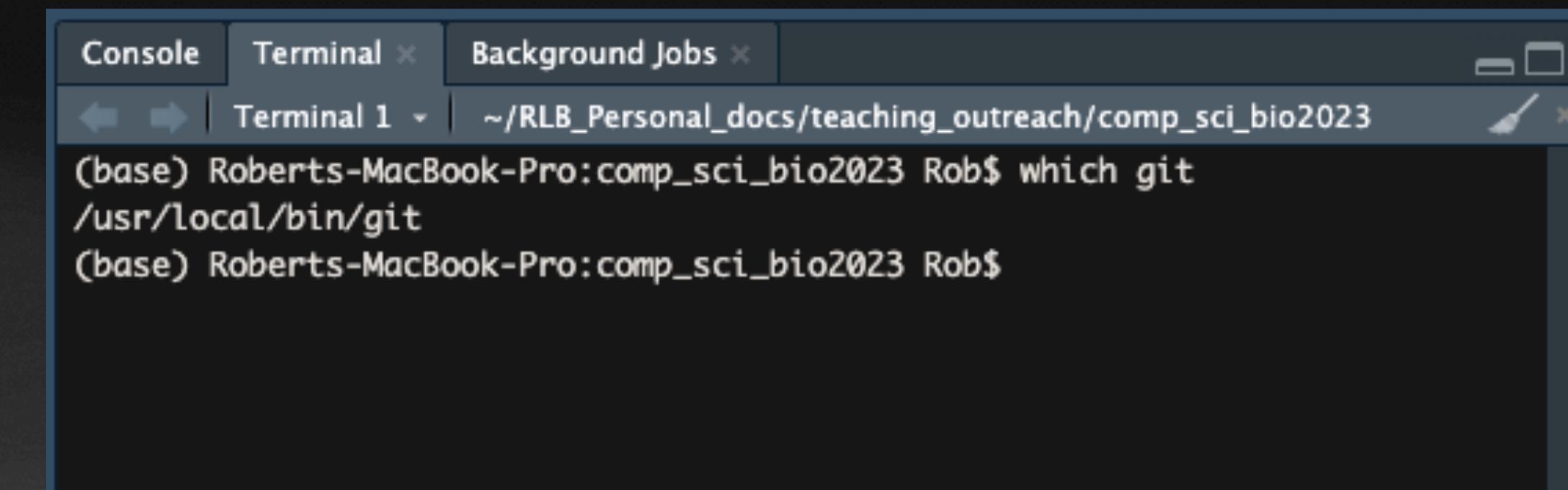
Windows: \$ where git

Mac: \$ which git

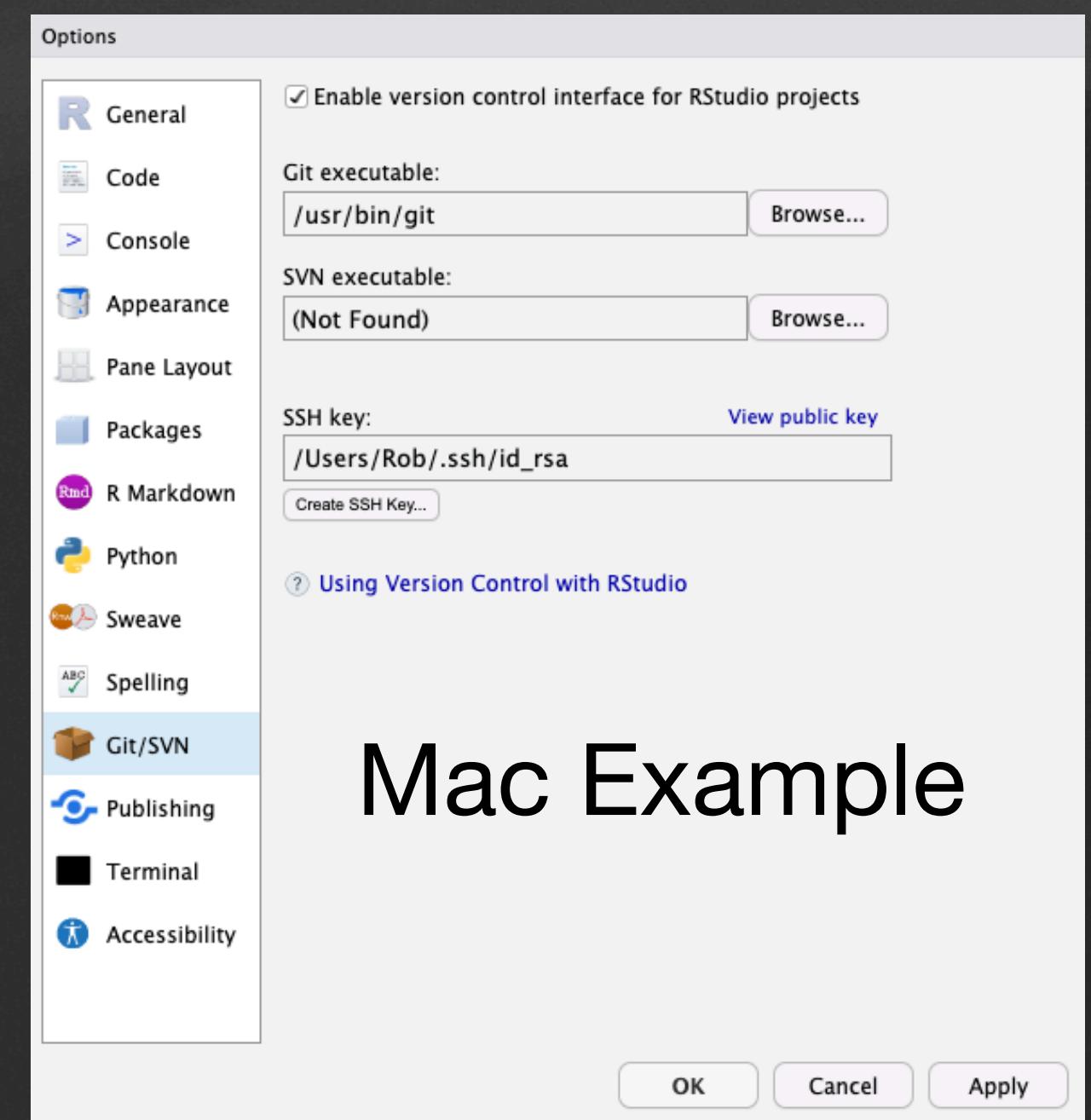
3. In Rstudio: Tools > Global Options > Git/SVN

- Make sure the Git executable path points to your git executable file

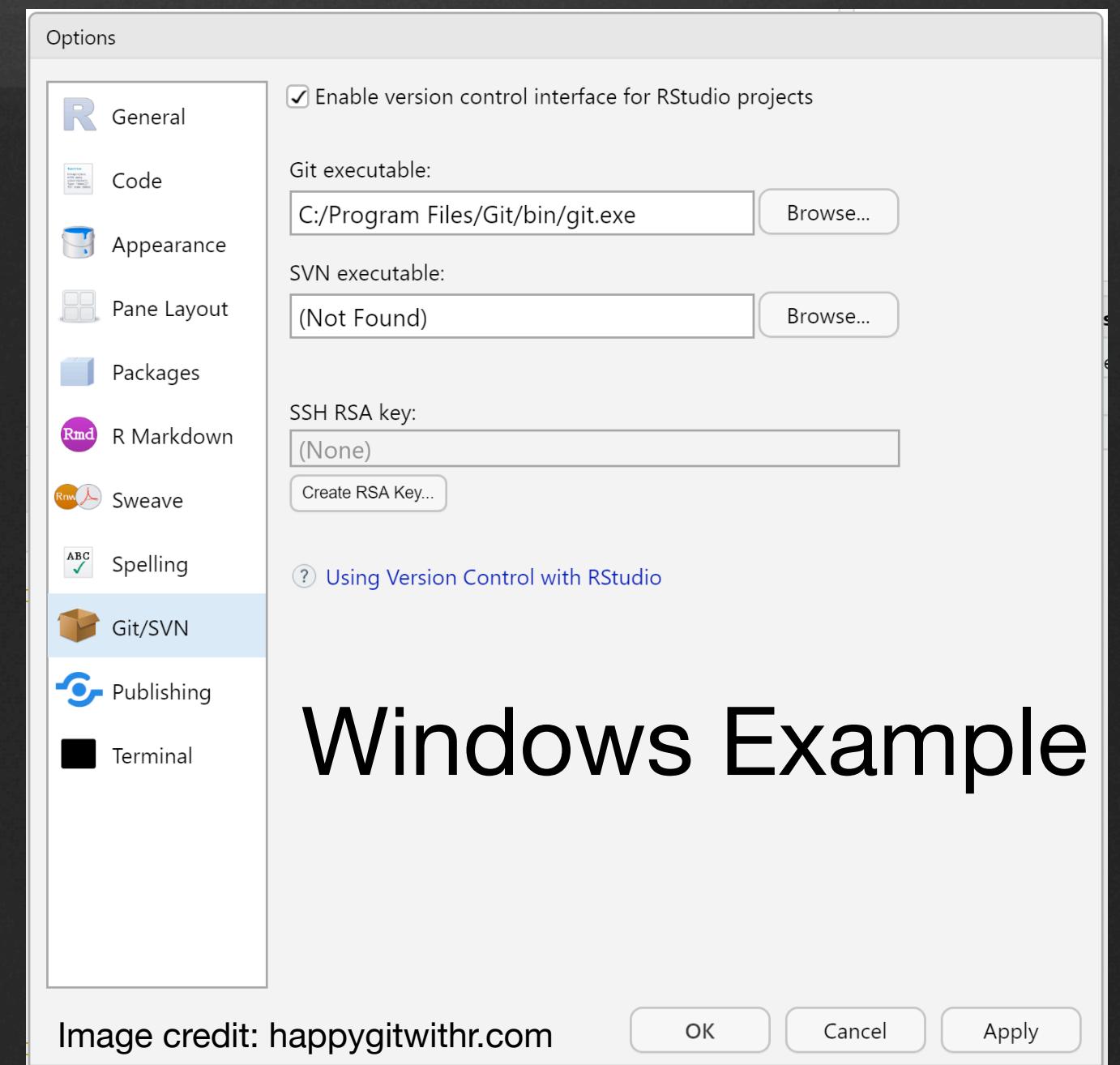
4. **Re-restart Rstudio** if you made changes



```
Console Terminal x Background Jobs x
Terminal 1 ~ /RLB_Personal_docs/teaching_outreach/comp_sci_bio2023
(base) Roberts-MacBook-Pro:comp_sci_bio2023 Rob$ which git
/usr/local/bin/git
(base) Roberts-MacBook-Pro:comp_sci_bio2023 Rob$
```



Mac Example



Windows Example

Image credit: happygitwithr.com

# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account
- IV. Walk through and help installing/upgrading software
- V. Rstudio/GitHub integration
- VI. **Version Control and Collaborative Coding Workflows**
- VII. Practice working collaboratively on a (non-coding) project
- VIII. Bonus: project management boards, issues, milestones

# Version Control

## Working on your own project:

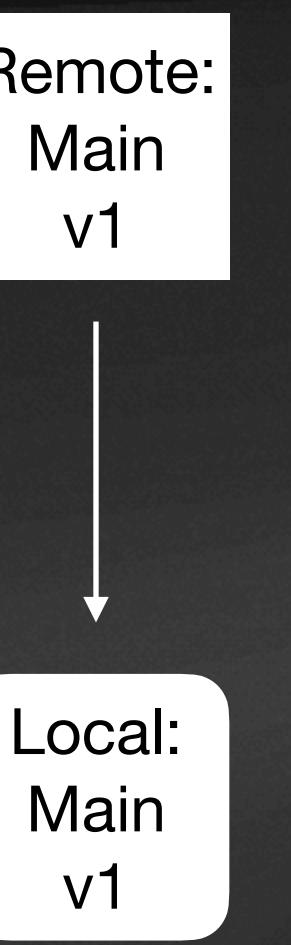
- Start with a repository (typically remote)

Remote:  
Main  
v1

# Version Control

## Working on your own project:

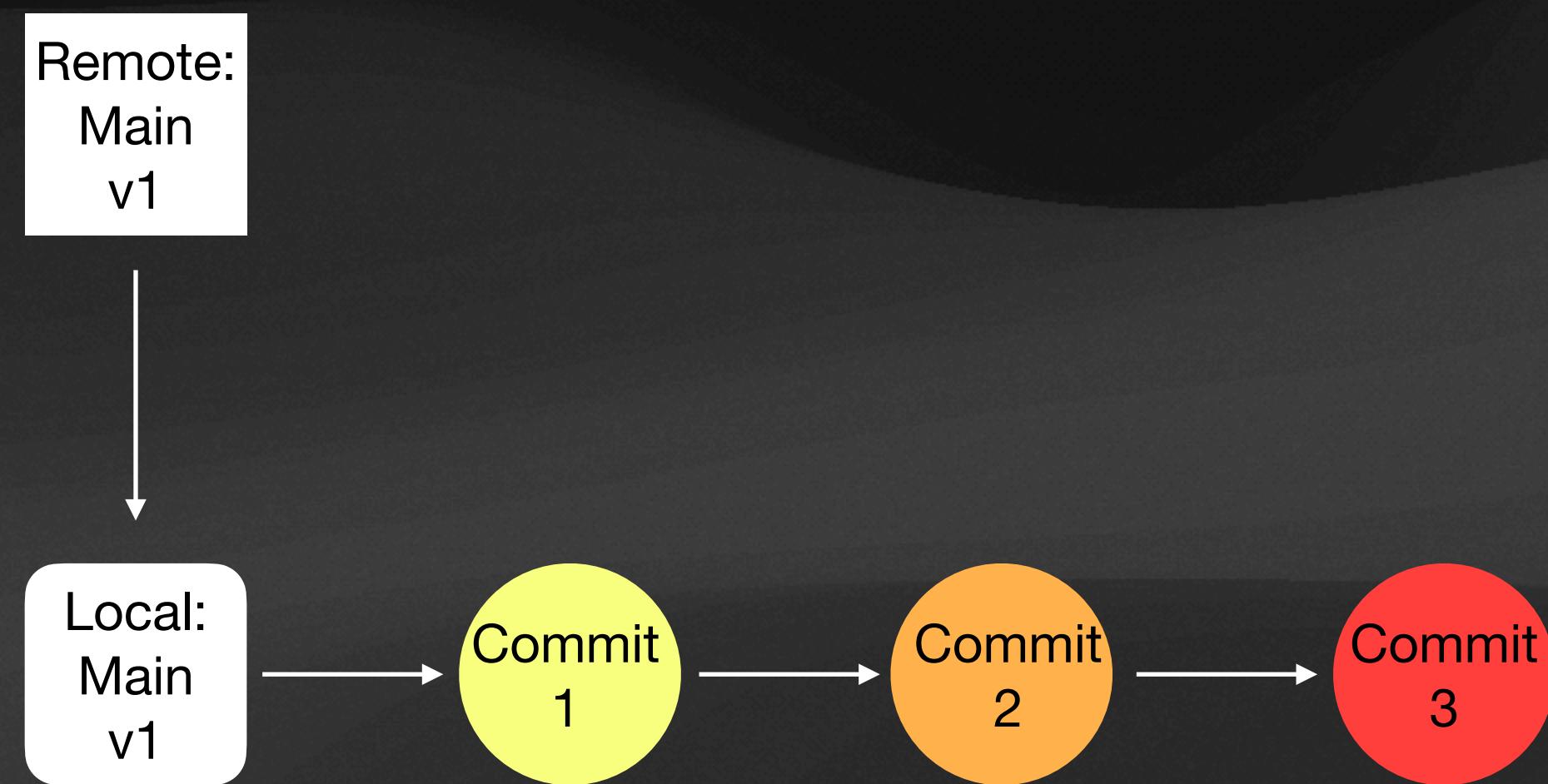
- Start with a repository (typically remote)
- Clone (or “pull”) the repository to local machine



# Version Control

## Working on your own project:

- Start with a repository (typically remote)
- Clone (or “pull”) the repository to local machine

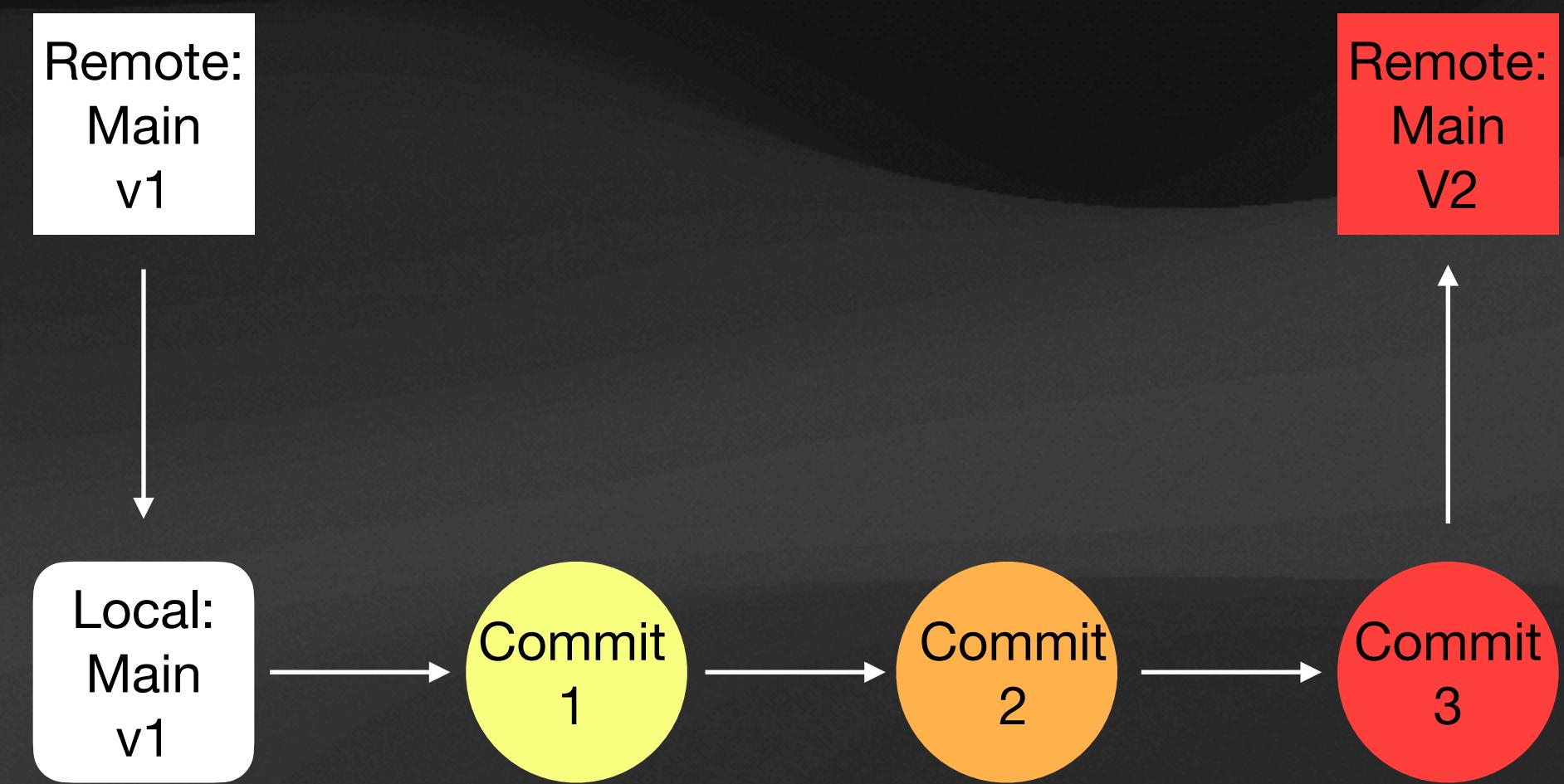


- Work locally, making snapshots as you go (“commits”)

# Version Control

## Working on your own project:

- Start with a repository (typically remote)
- Clone (or “pull”) the repository to local machine



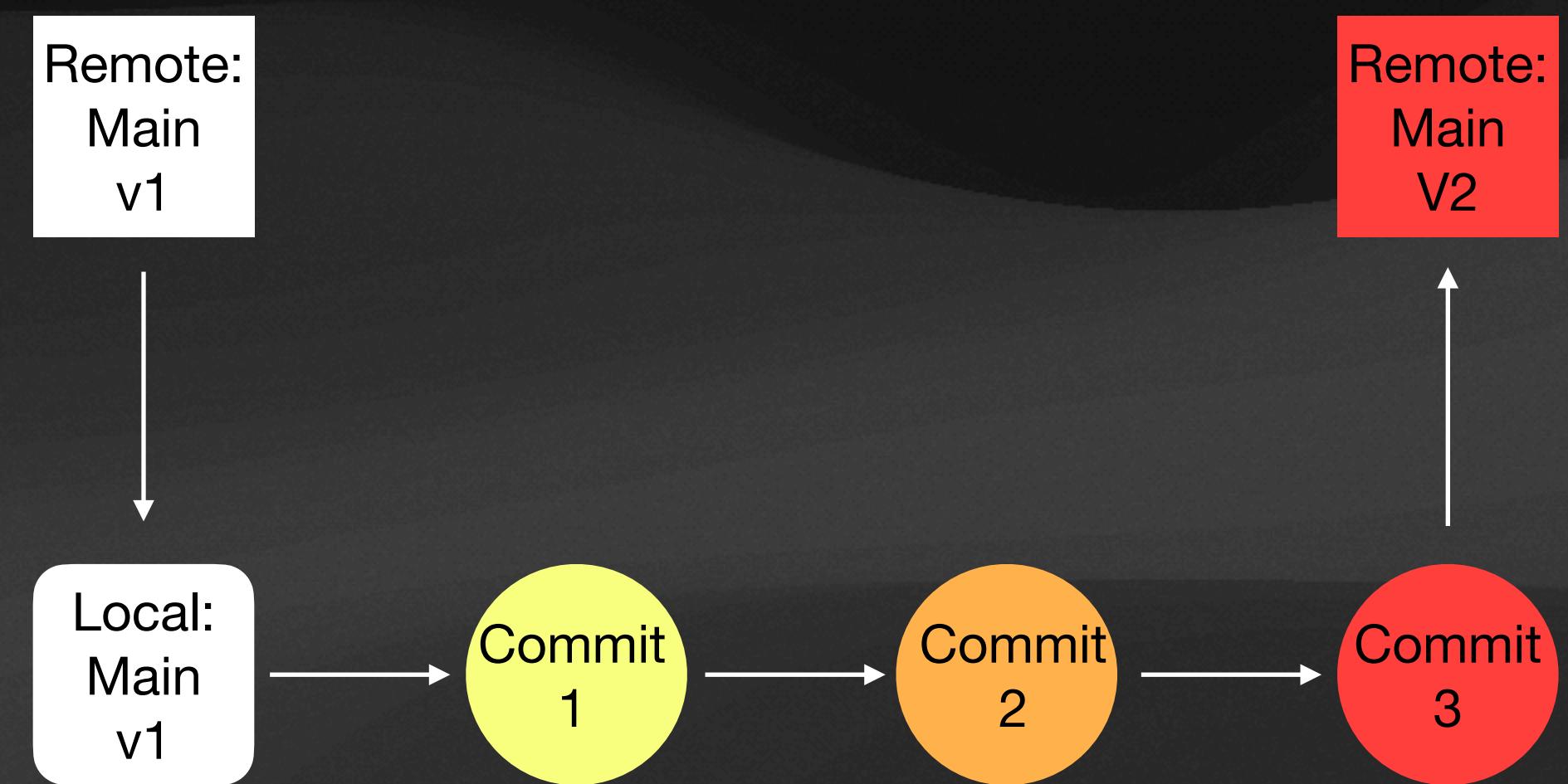
- Work locally, making snapshots as you go (“commits”)

- “Push” changes to your remote repository

# Version Control

## Working on your own project:

- Start with a repository (typically remote)
- Clone (or “pull”) the repository to local machine



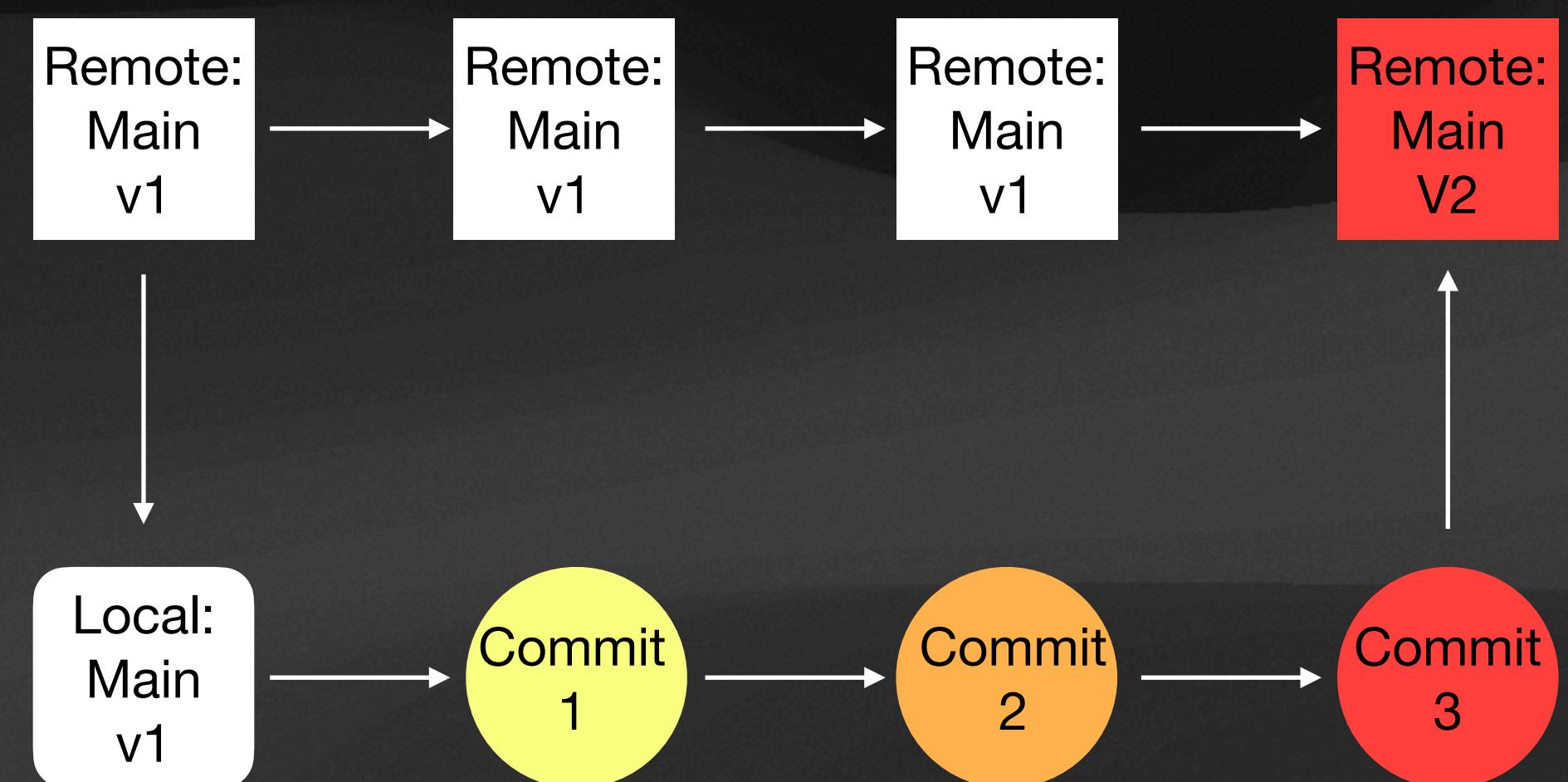
- Work locally, making snapshots as you go (“commits”)

- “Push” changes to your remote repository
- GitHub now reflects most recent snapshot & all commits

# Version Control

## Working on your own project:

- Start with a repository (typically remote)
- Clone (or “pull”) the repository to local machine



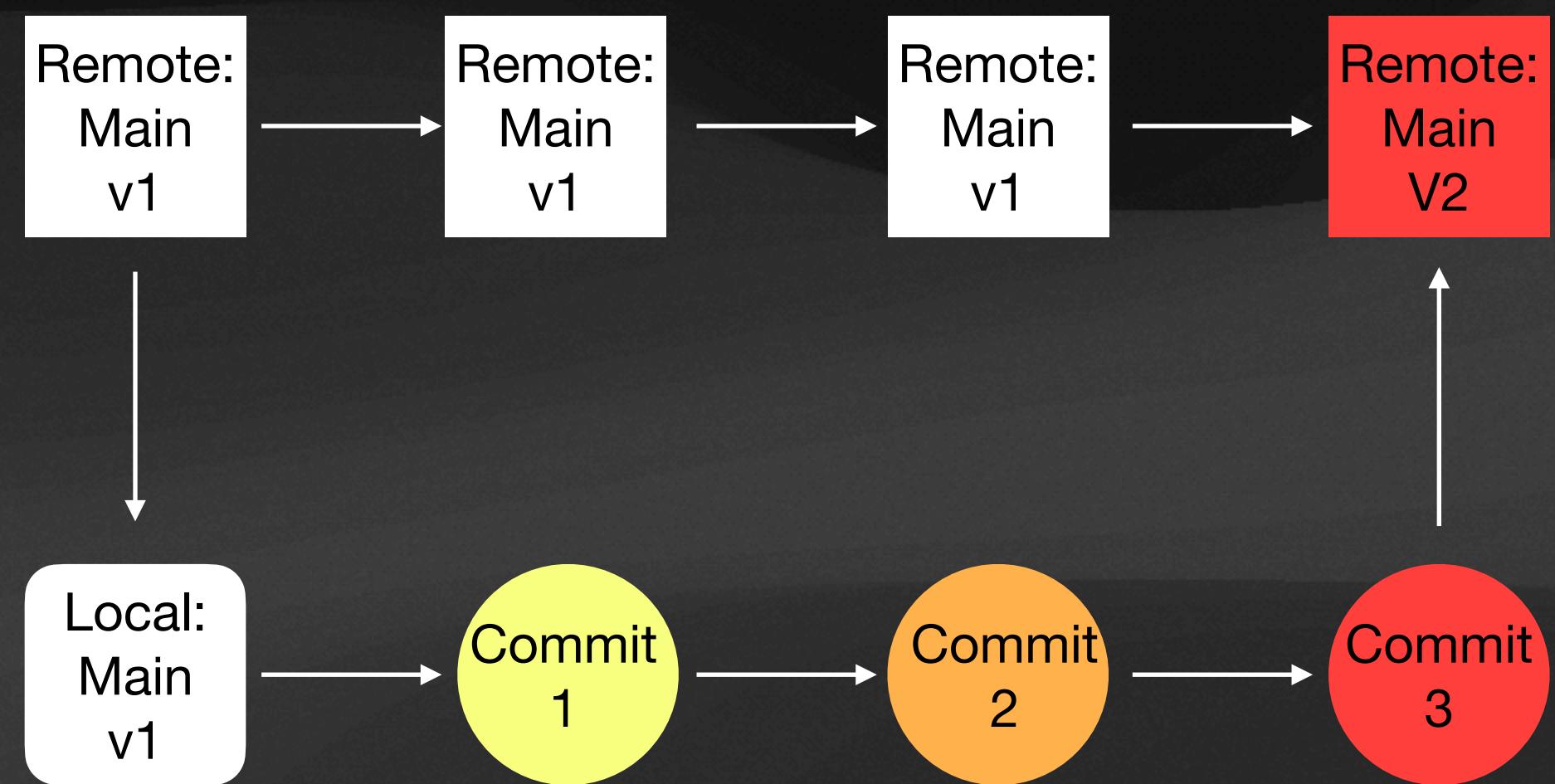
- Work locally, making snapshots as you go (“commits”)

- “Push” changes to your remote repository
  - GitHub now reflects most recent snapshot & all commits
  - Remote repository available throughout development

# Version Control

## Working on your own project:

- Start with a repository (typically remote)
- Clone (or “pull”) the repository to local machine



- Work locally, making snapshots as you go (“commits”)

- “Push” changes to your remote repository
  - GitHub now reflects most recent snapshot & all commits
  - Remote repository available throughout development
  - Version Control: commit history maintained; changes to be rolled back

# Collaborative Coding

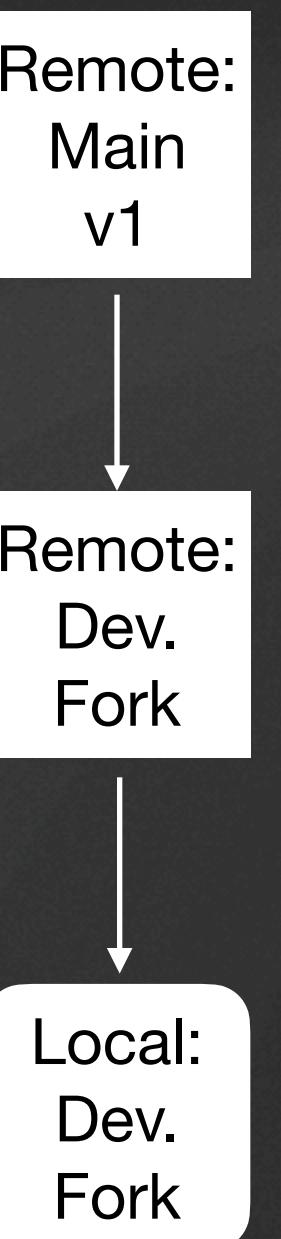
## Forking Workflow

- Lightweight, relatively easy
- Highlight your own work on your own GitHub account
- Safe! You aren't changing anything on the main repo until you're really ready; you can test it out after upload
- Can request code reviews, etc before your changes get incorporated into the main repo

# Collaborative Coding

## Forking Workflow

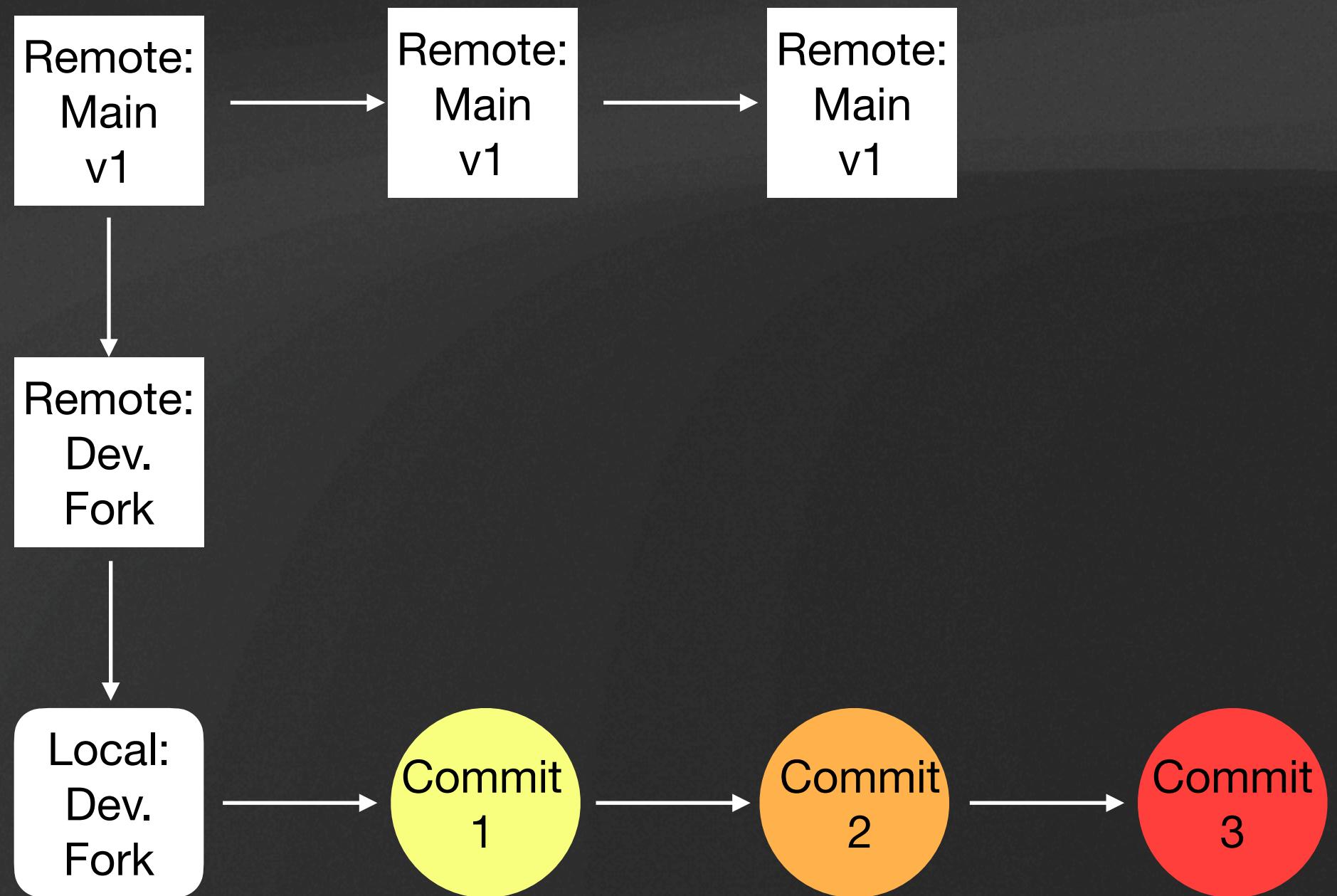
- Remote Main v1 - might not be your repo (maybe a lab repo?)
- “Fork” the lab repo to a repo to create a copy on your own GitHub account
- Clone (or pull) your development fork to your local machine



# Collaborative Coding

## Forking Workflow

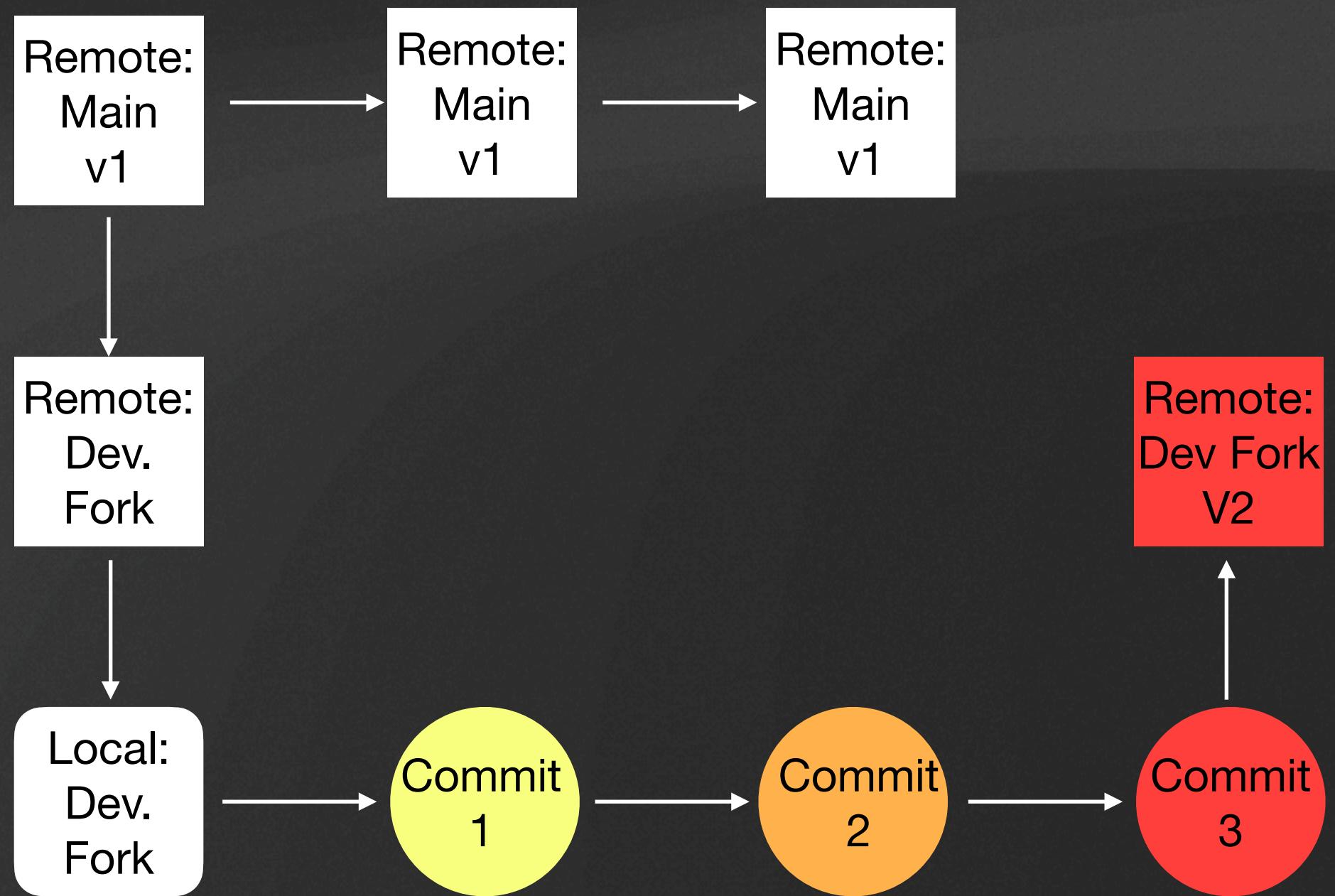
- Remote Main v1 - might not be your repo (maybe a lab repo?)
- “Fork” the lab repo to a repo to create a copy on your own GitHub account
- Clone (or pull) your development fork to your local machine



# Collaborative Coding

## Forking Workflow

- Remote Main v1 - might not be your repo (maybe a lab repo?)
- “**Fork**” the lab repo to a repo to create a copy on your own GitHub account
- “**Clone**” (or “**Pull**”) your development fork to your local machine

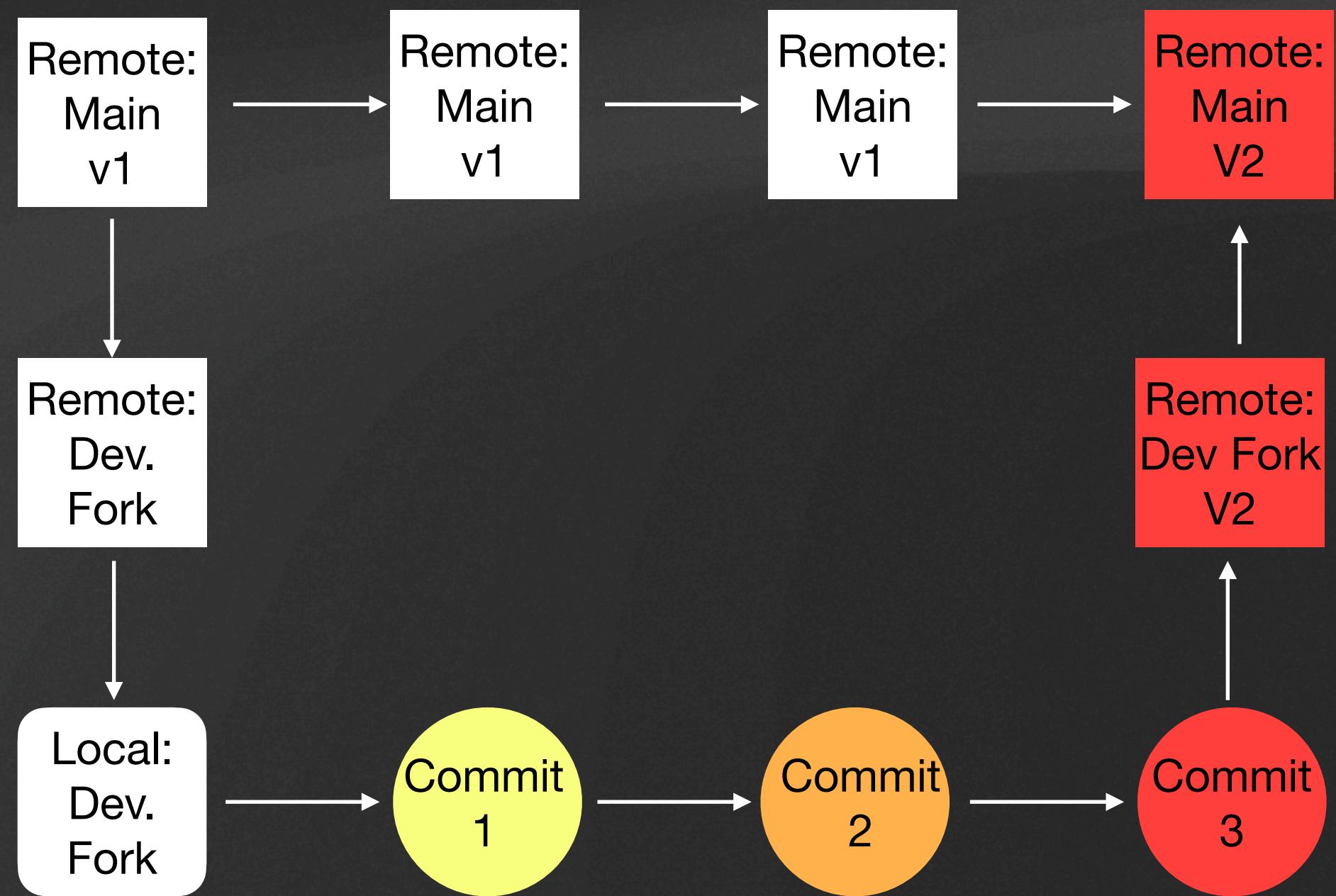


- Initiate a “**Pull request**”: ask that someone incorporate your changes to the lab account (NOT a PULL)
- “**Push**” your changes to the Forked copy of the repo on your own GitHub account
- Make your edits (“**commits**”) locally as before

# Collaborative Coding

## Forking Workflow

- Remote Main v1 - might not be your repo (maybe a lab repo?)
- “**Fork**” the lab repo to a repo to create a copy on your own GitHub account
- “**Clone**” (or “**Pull**”) your development fork to your local machine

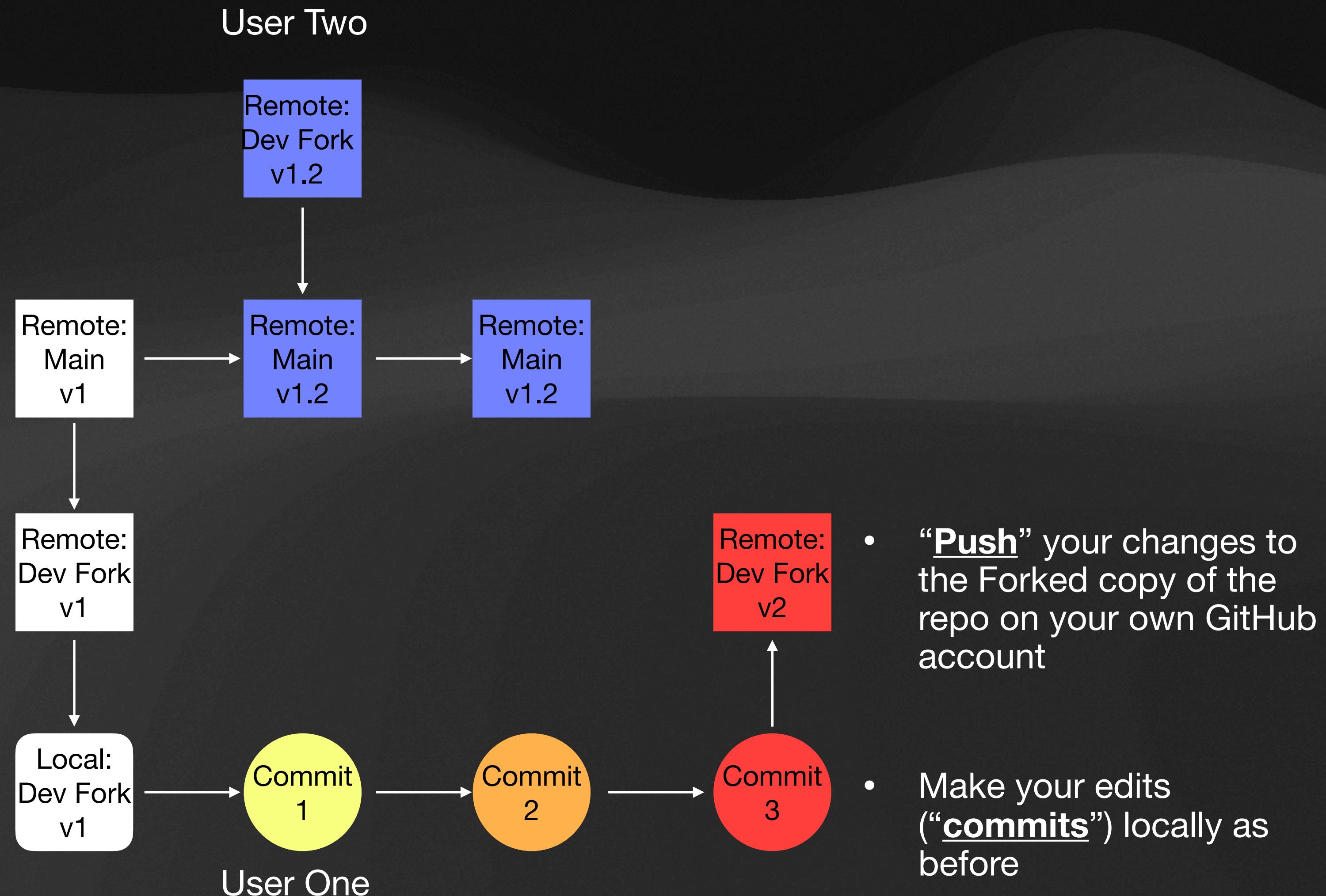


- Initiate a “**Pull request**”: ask that someone incorporate your changes to the lab account (NOT a PULL)
- “**Push**” your changes to the Forked copy of the repo on your own GitHub account
- Make your edits (“**commits**”) locally as before

# Collaborative Coding

## Forking Workflow

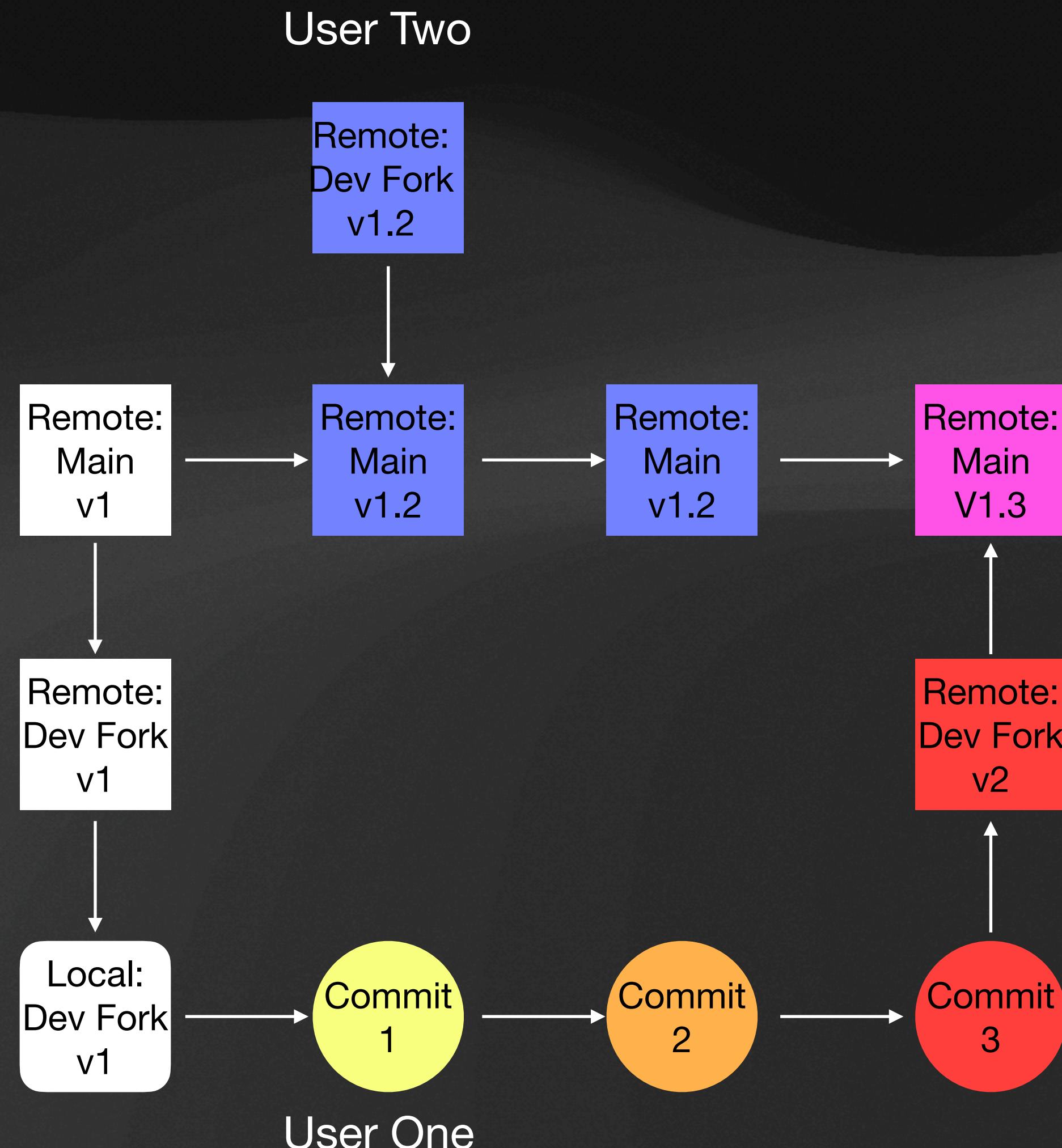
- Remote Main v1 - might not be your repo (maybe a lab repo?)
- “**Fork**” the lab repo to a repo to create a copy on your own GitHub account
- “**Clone**” (or “**Pull**”) your development fork to your local machine



# Collaborative Coding

## Forking Workflow

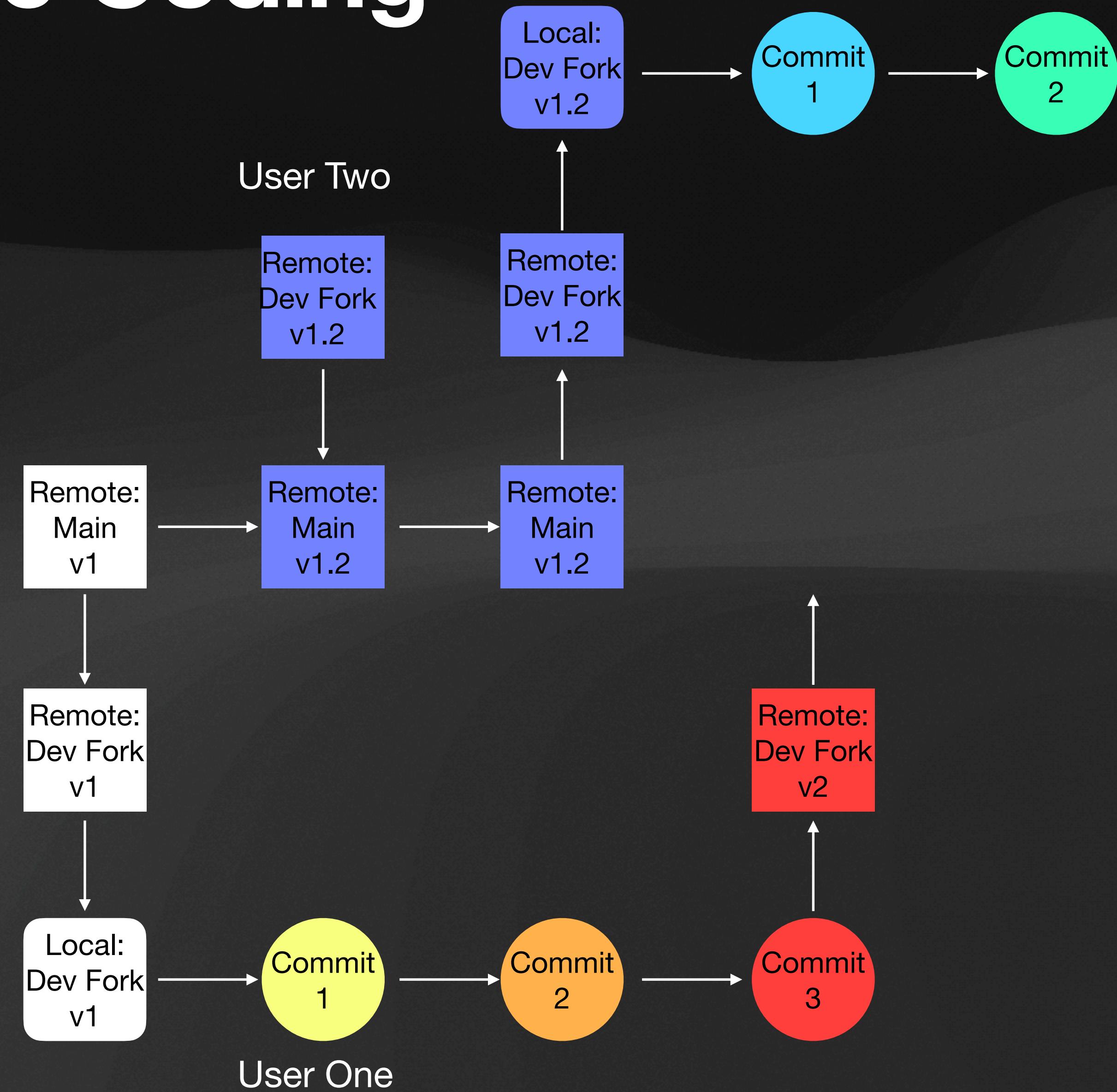
- Remote Main v1 - might not be your repo (maybe a lab repo?)
- “**Fork**” the lab repo to a repo to create a copy on your own GitHub account
- “**Clone**” (or “**Pull**”) your development fork to your local machine



- Initiate a “**Pull request**”: ask that someone incorporate your changes to the lab account (NOT a PULL)
- “**Push**” your changes to the Forked copy of the repo on your own GitHub account
- Make your edits (“**commits**”) locally as before

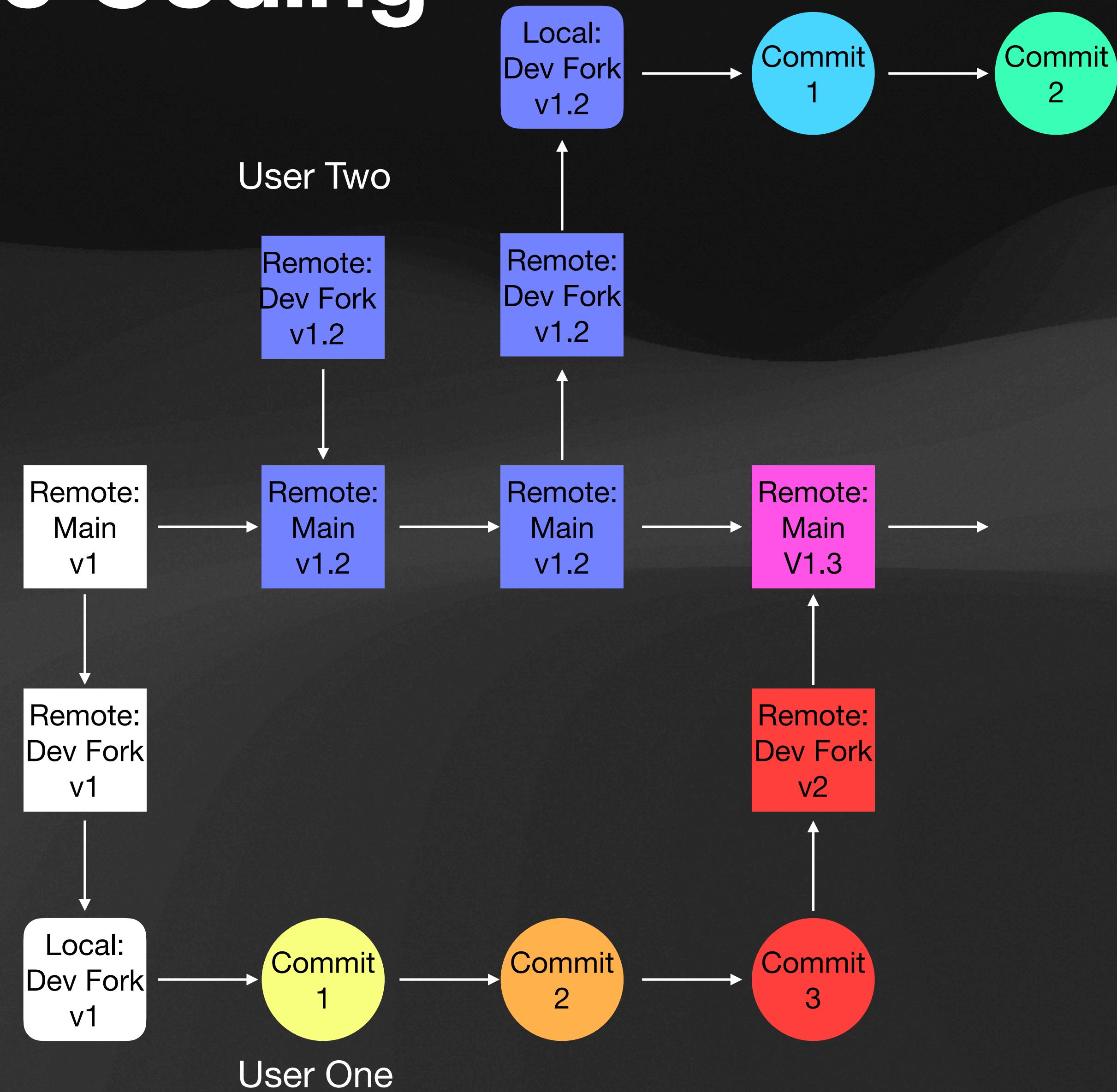
# Collaborative Coding

## Forking Workflow



# Collaborative Coding

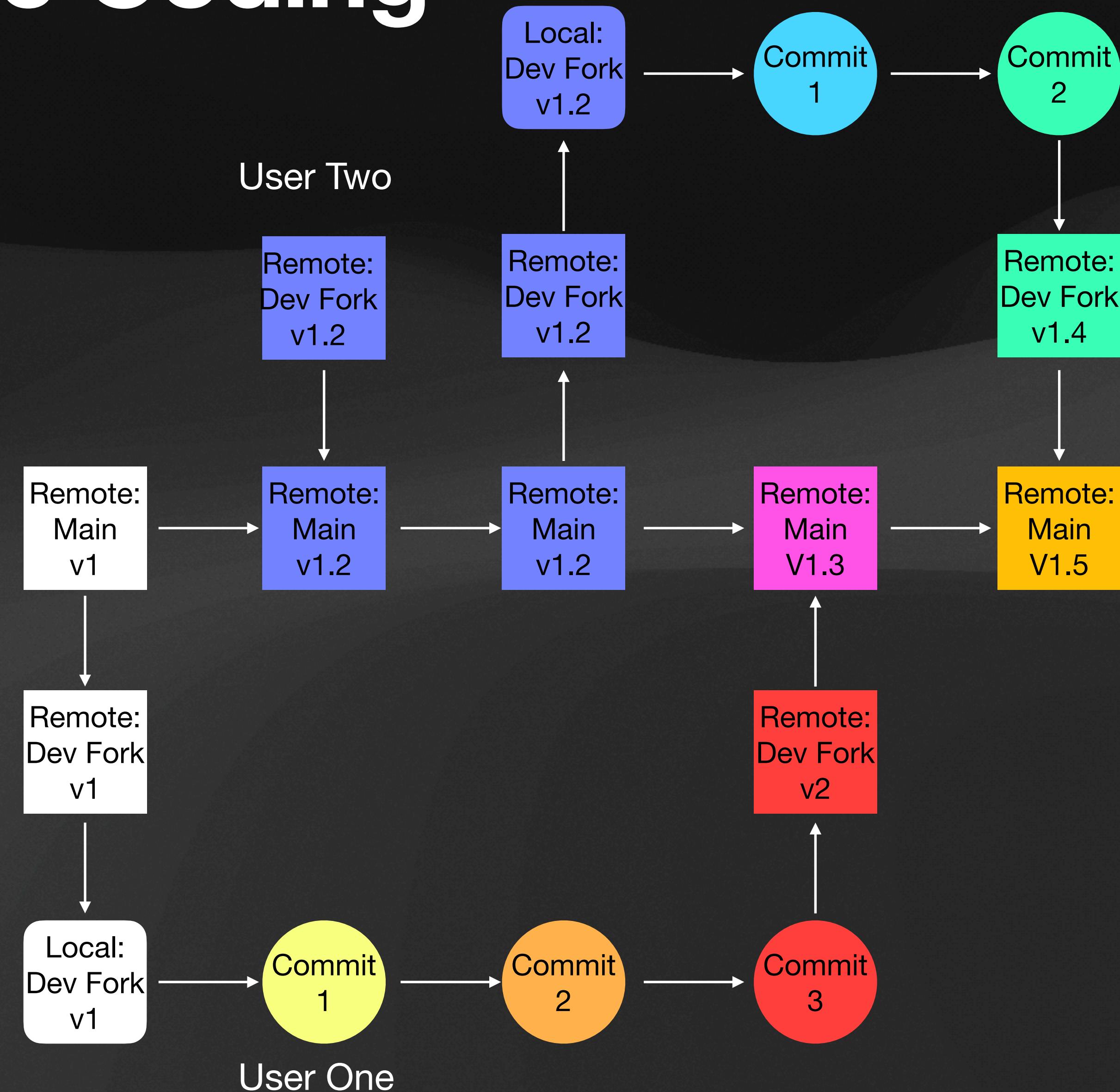
## Forking Workflow



# Collaborative Coding

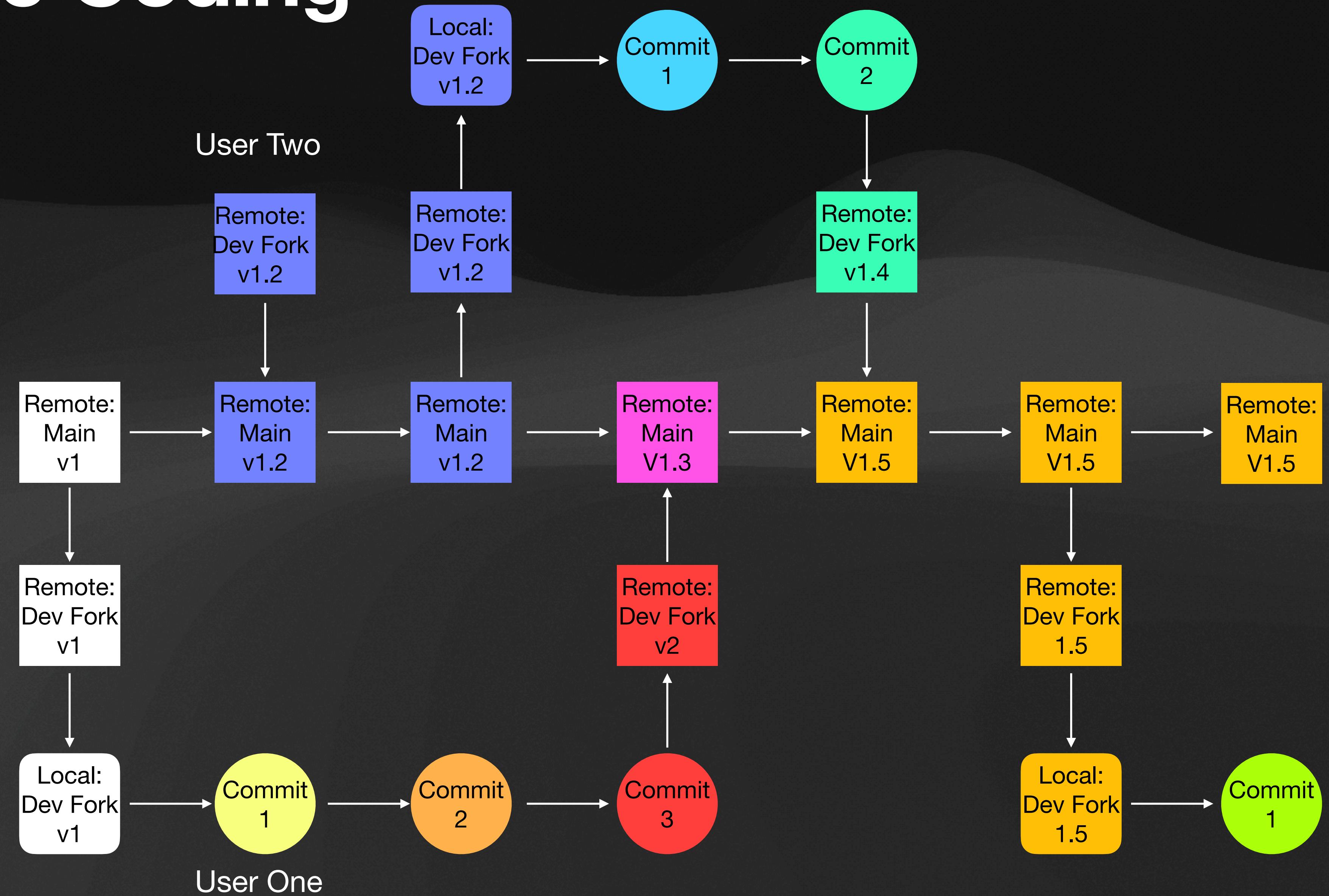
## Forking Workflow

- Remote Main v1 - might not be your repo (maybe a lab repo?)
- “**Fork**” the lab repo to a repo to create a copy on your own GitHub account
- “**Clone**” (or “**Pull**”) your development fork to your local machine



# Collaborative Coding

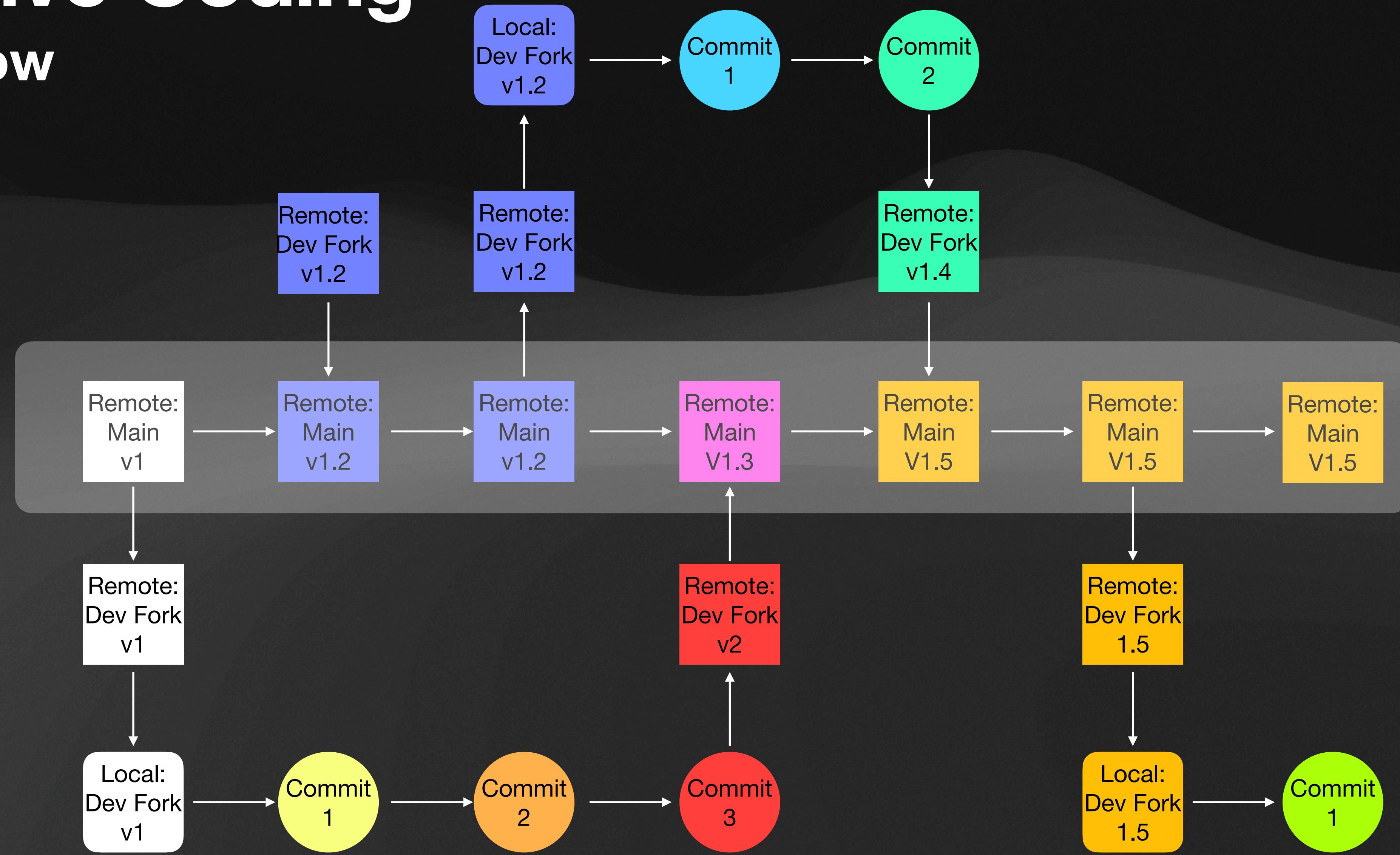
## Forking Workflow



# Collaborative Coding

## Forking Workflow

- Two (or more) users can simultaneously collaborate on the same project
- The main branch is always available to the public
- Multiple different development forks can be worked on simultaneously



# Version Control And Collaborative Coding Terminology

- Repository (repo) - managed collection of files
- Version Control - keeping track of changes in files
- Fork - create a copy of a repo on a different account
- Branch - A new line of development; may be on a different repo or account (result of a Fork) or may be in the same repo.
- Clone - create a copy of a repo on a local machine
- Pull - pull all the changes from a repo to your local machine
- Commit - a versioned change, with a comment
- Push - move all the commits to your branch
- Pull Request - Request that your (modified) Forked branch be incorporated into a the main branch
- Merge - the process of reconciling two branches upon a Push or a Pull Request
- Merge conflict - when the two branches cannot automatically be reconciled and must be manually adjusted

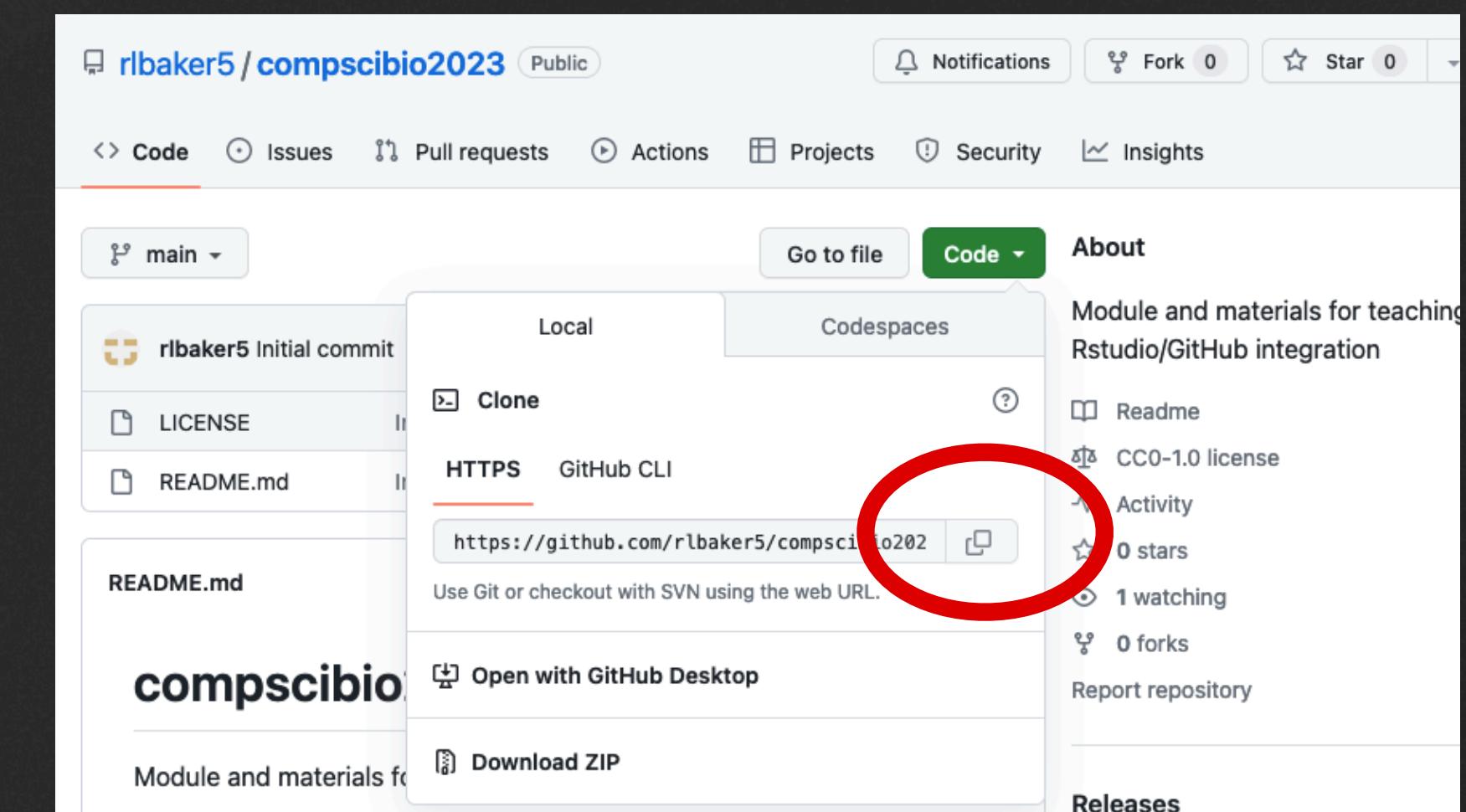
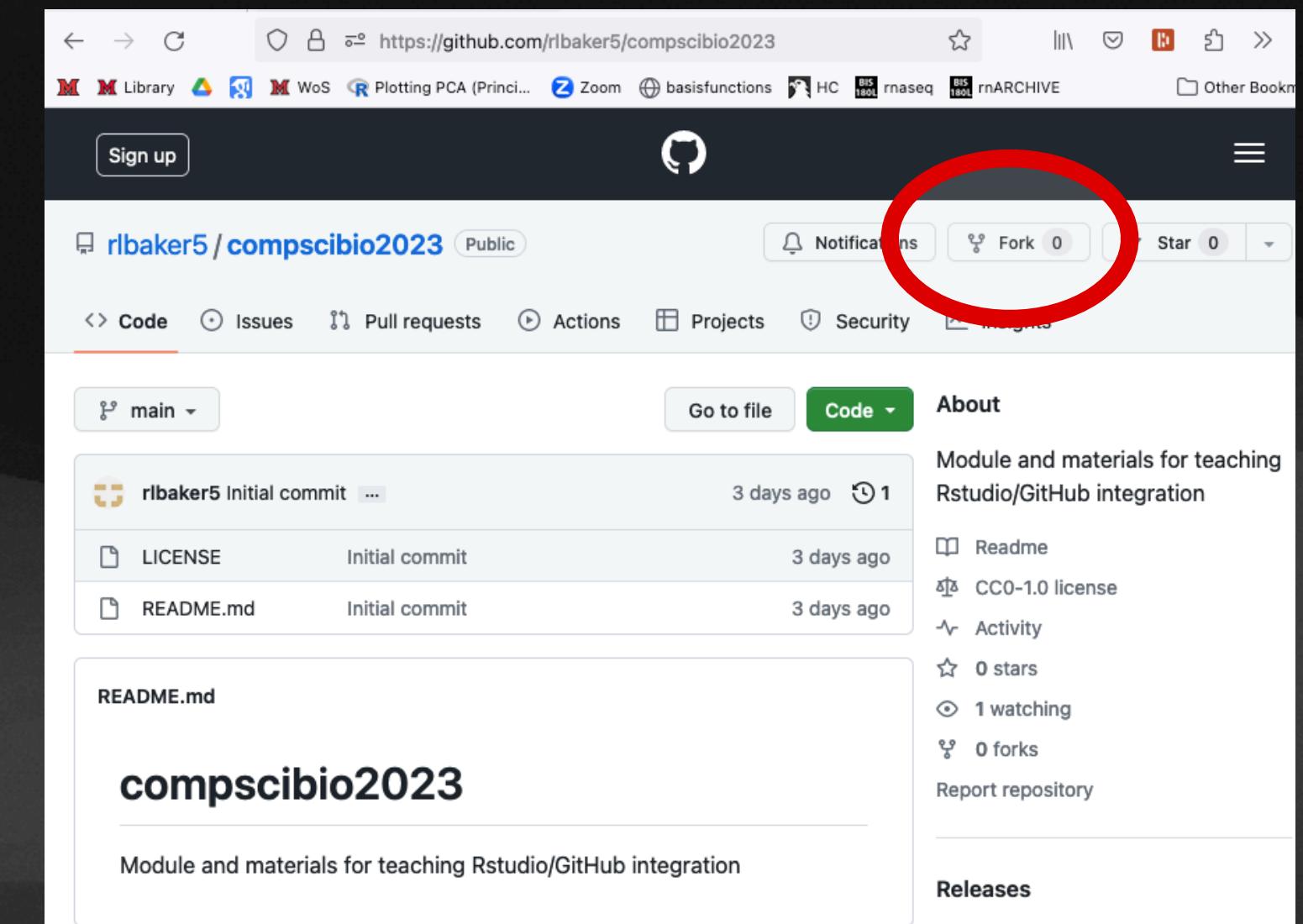
# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account
- IV. Walk through and help installing/upgrading software
- V. Rstudio/GitHub integration
- VI. Version Control and Collaborative Coding Workflows
- VII. **Practice: collaborative version control**
- VIII. Bonus: project management boards, issues, milestones

# Practice: collaborative version control

## Fork a Repo

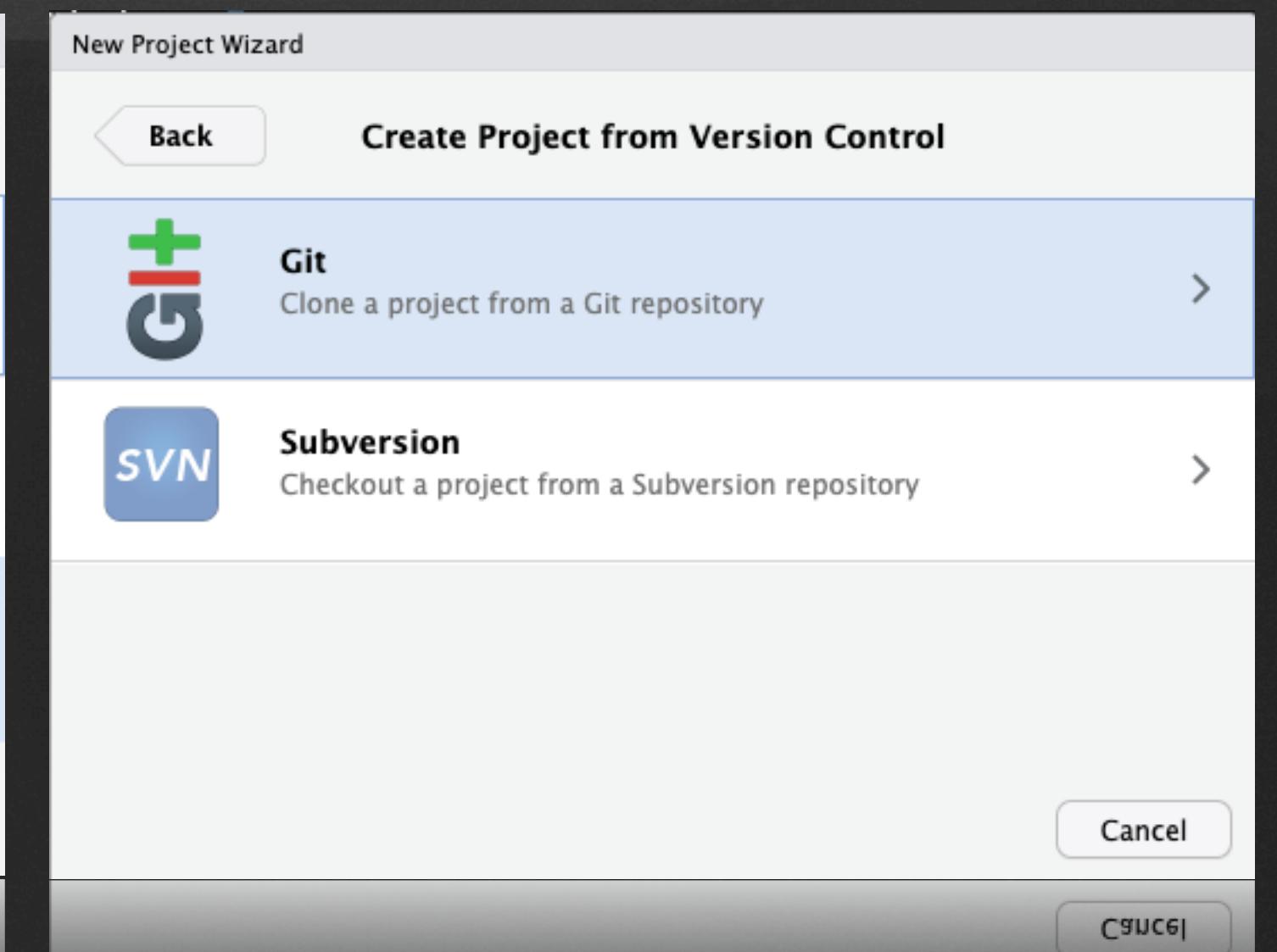
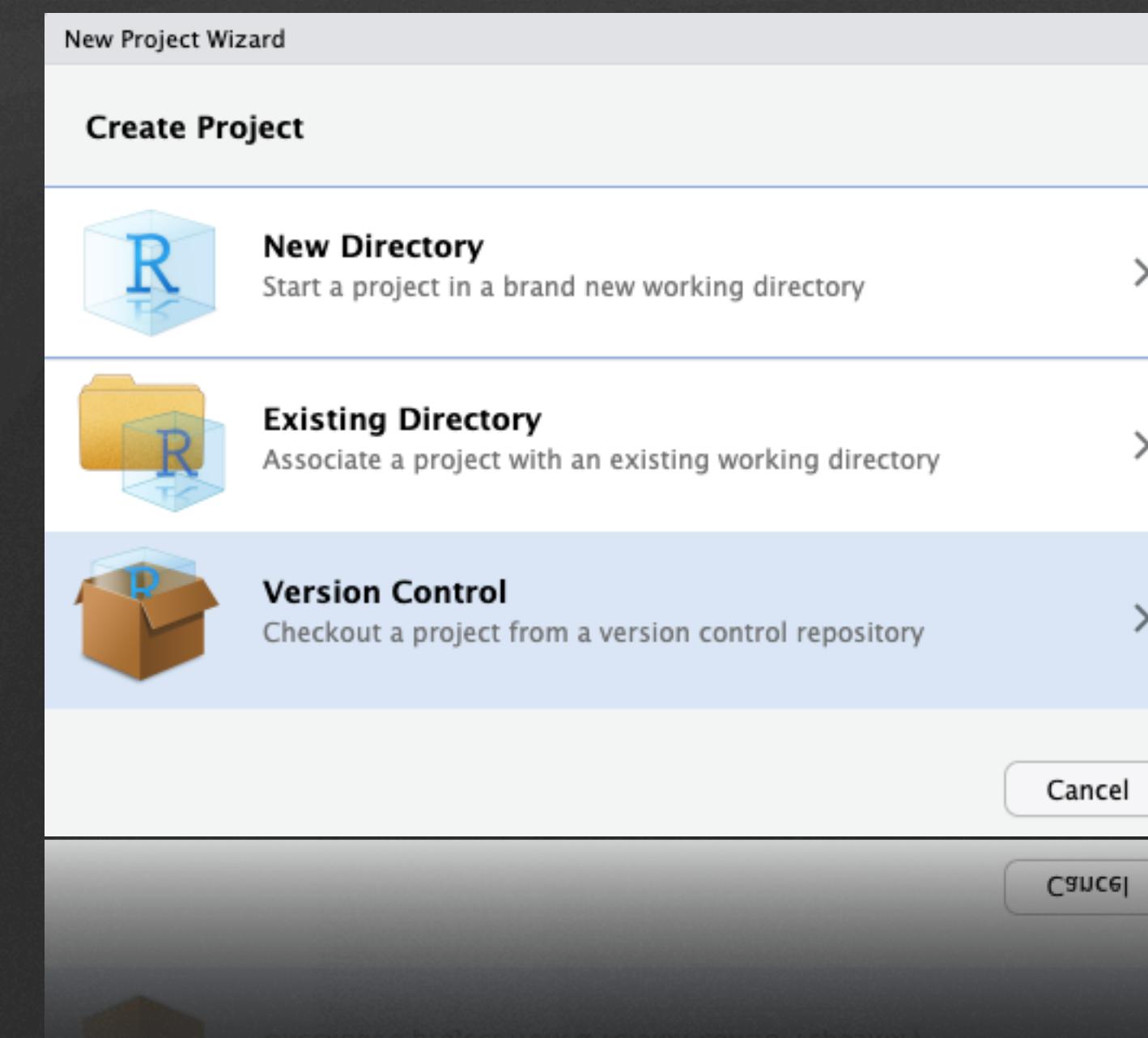
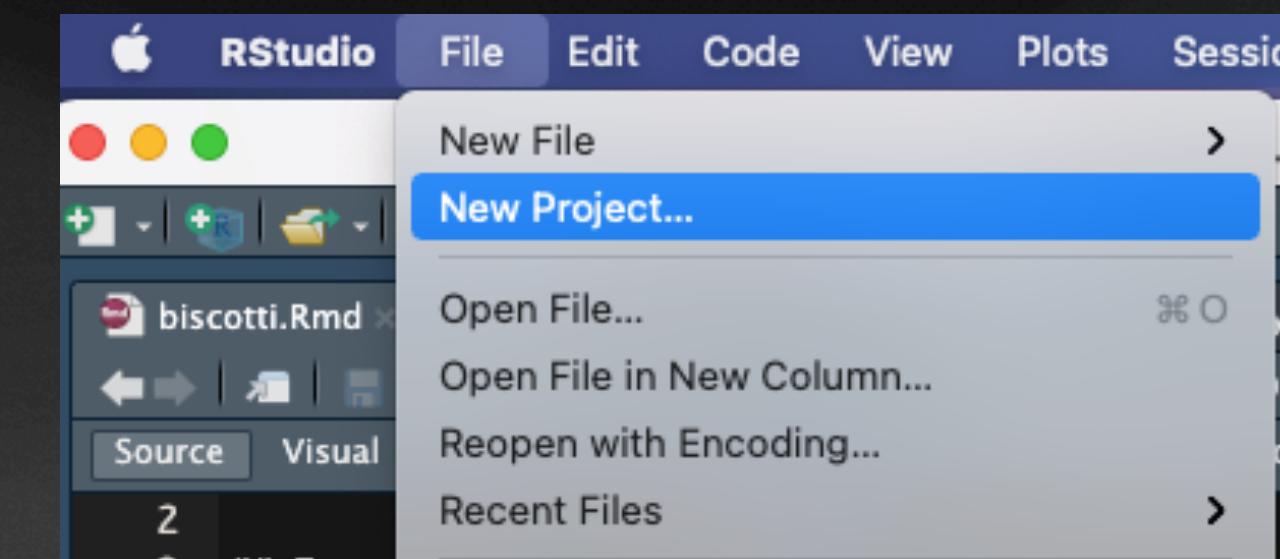
- Log in to your [GitHub.com](https://GitHub.com) account
- Navigate to:  
[GitHub.com/rbaker5/compscibio2023](https://GitHub.com/rbaker5/compscibio2023)
- Fork the repo to your own account
- Navigate to your own profile and enter the newly forked repo ([github.com/<username>/compscibio2023](https://github.com/<username>/compscibio2023))
- Use the green “Code” drop-down to copy the URL by clicking on the interlocking squares



# Practice: collaborative version control

## Clone a Repo

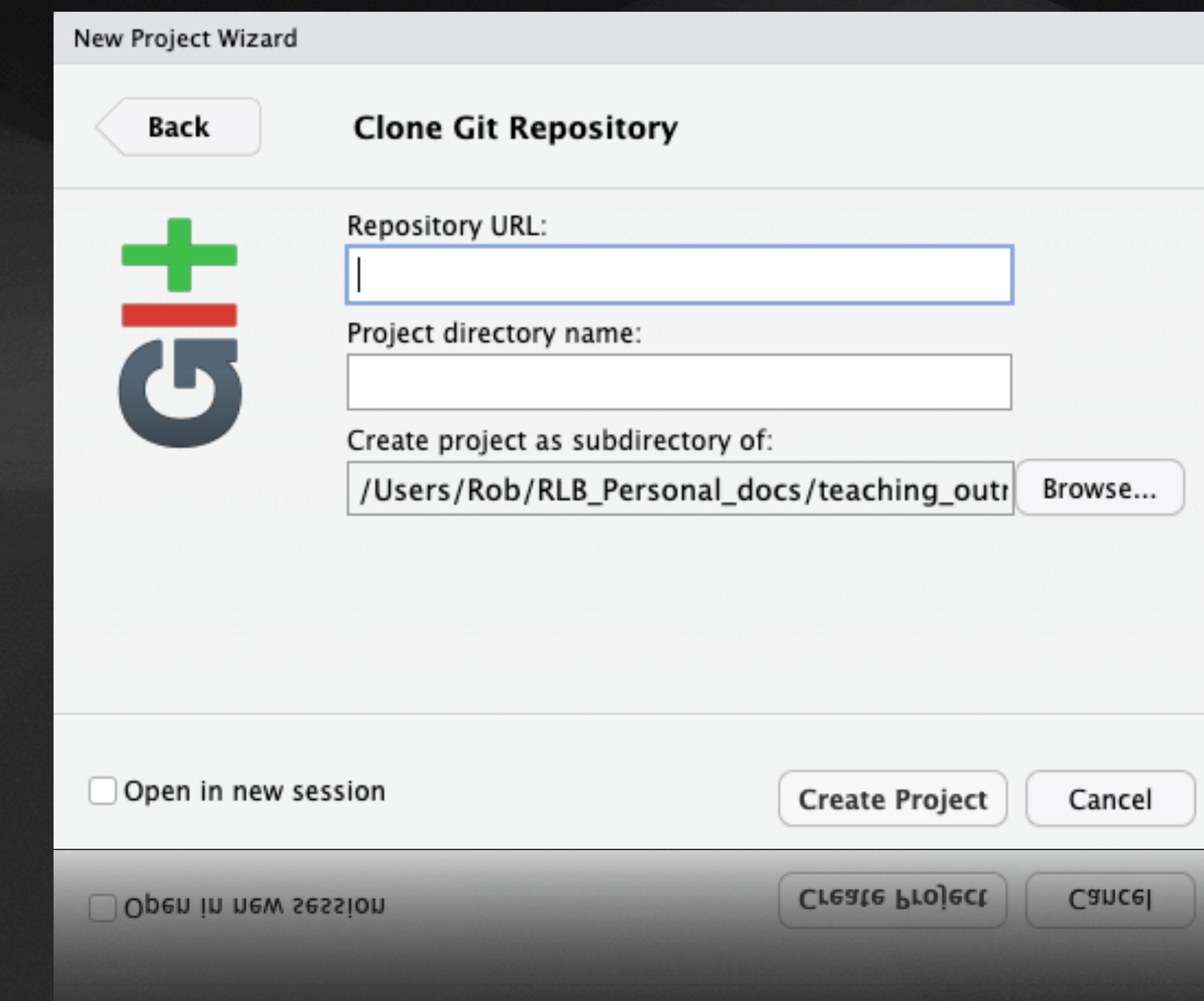
- Open Rstudio
- Using From the “File” drop-down menu, select “New Project”
- Select “Version Control”
- Then choose “Git (“clone a project from a Git repository”)



# Practice: collaborative version control

## Clone a Repo

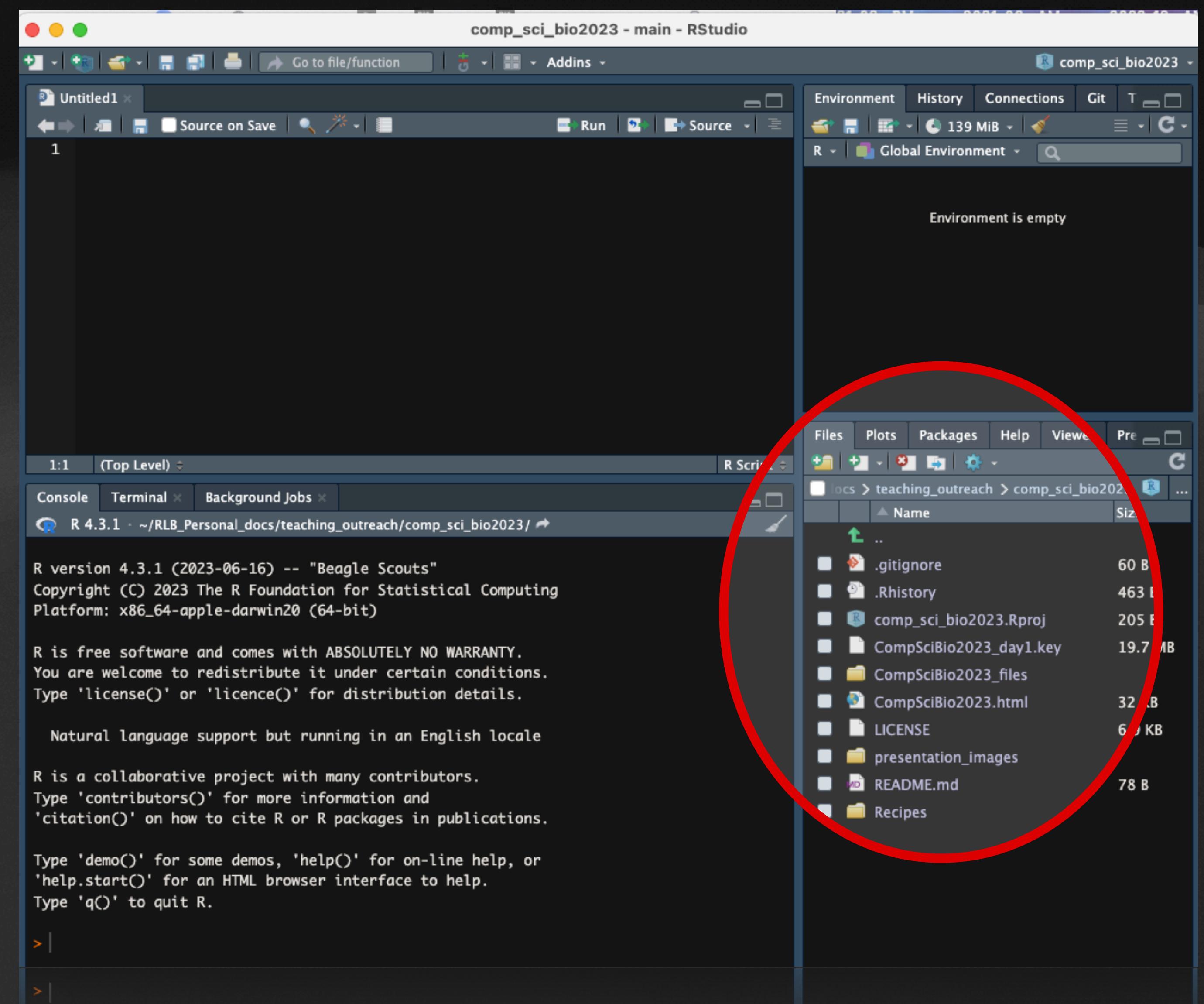
- Paste in the repo URL
- Give your project a name (e.g. “compscibio2023\_dev”)
- Select a location where you want the project to live on your hard drive
- Click “Create Project”



# Practice: collaborative version control

## Clone a Repo

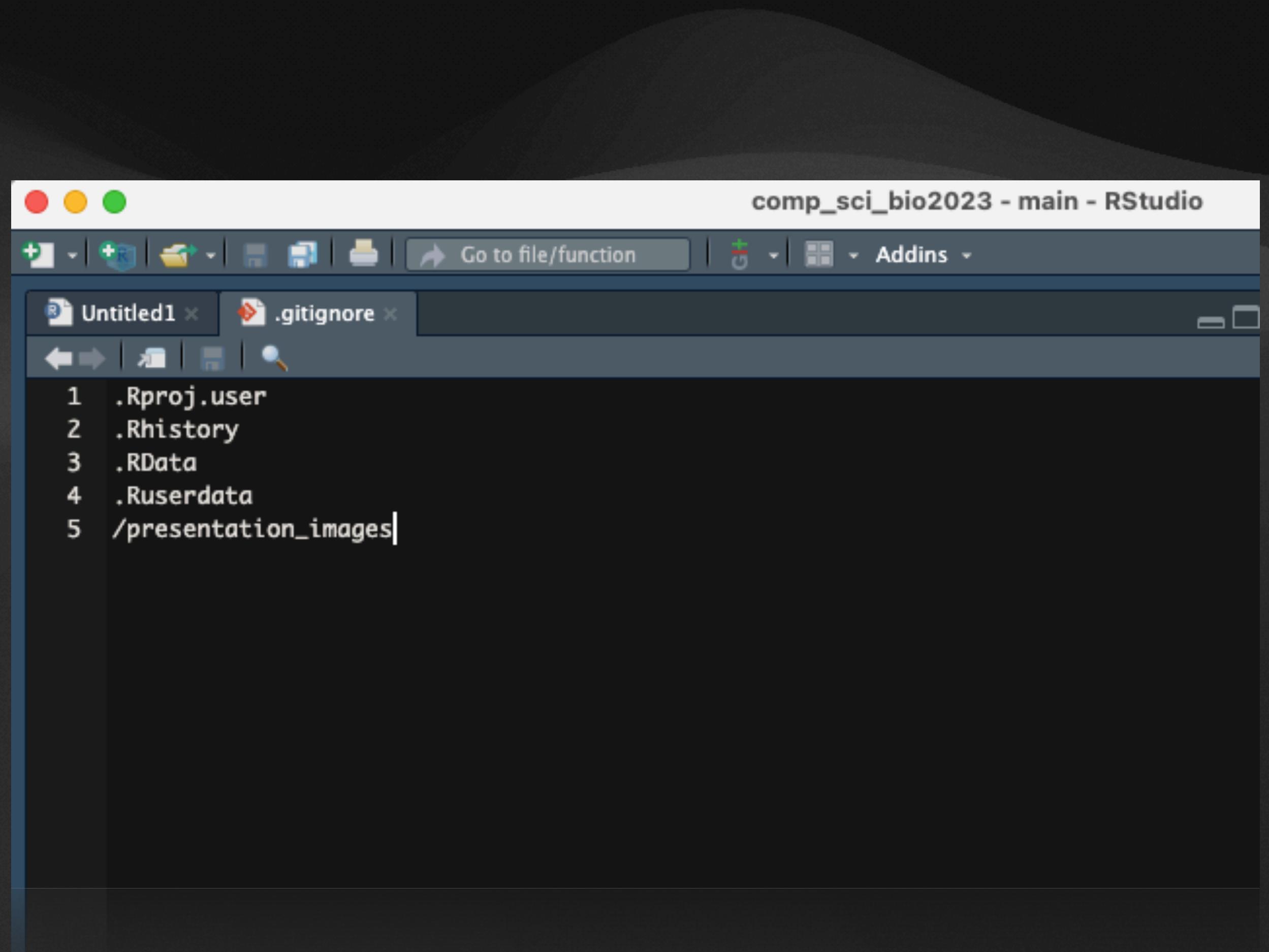
- Your Rstudio should look something like this (probably with a different color scheme)
- But notice all the files from the repo are now listed under Files and are now on your local machine!
- (Your files may differ slightly from the ones shown here)



# Practice: collaborative version control

## .gitignore

- All the changes you make will be pushed to your (public) repo.
- There are likely files you don't want to make public (don't want to track via version control)
- You can tell Git to ignore these files or folders by adding them to “.gitignore”
- Click .gitignore to open it
- Add some files to ignore. Good ones include .Rhistory and .xlsx



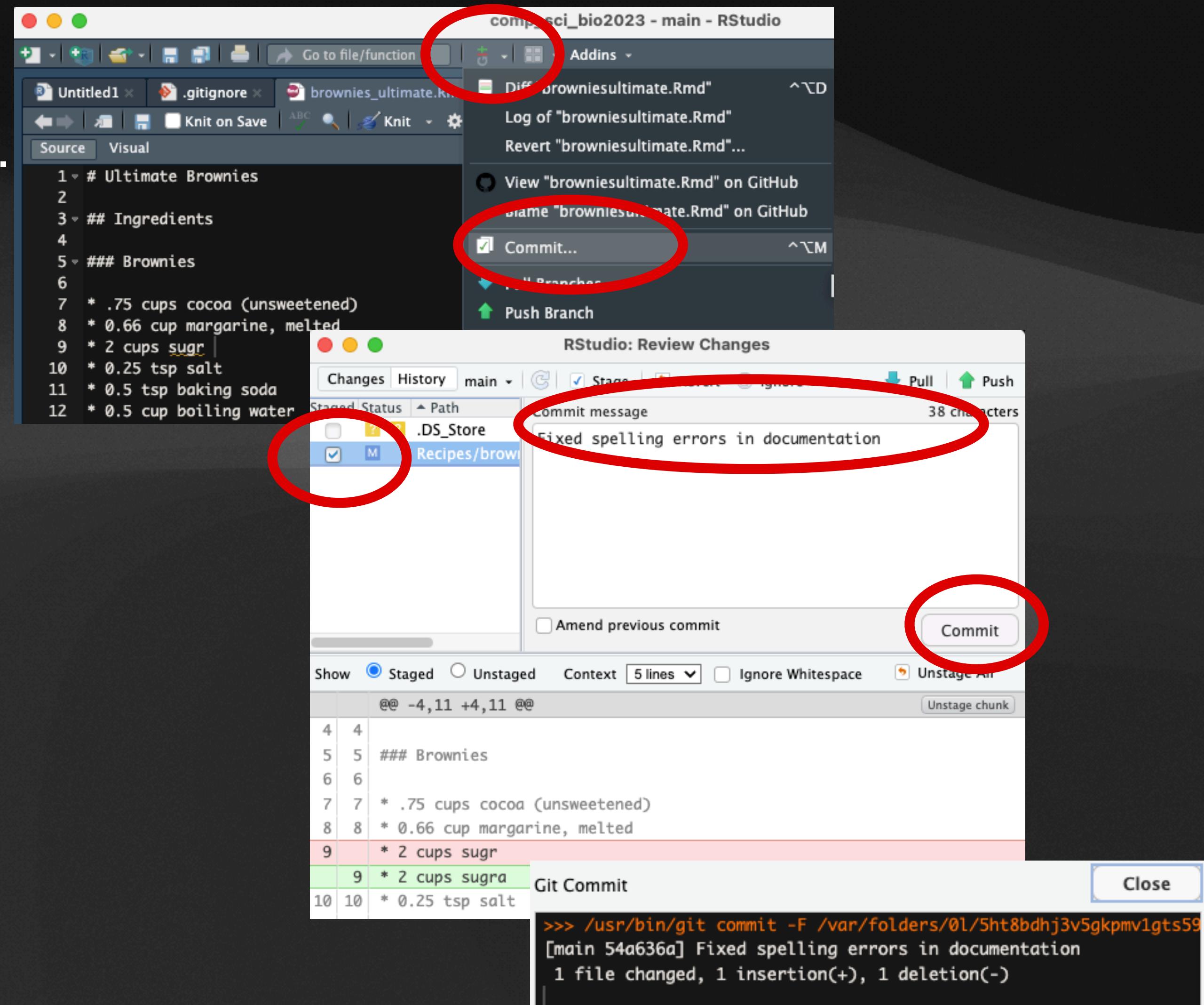
The screenshot shows the RStudio interface with a dark theme. The title bar reads "comp\_sci\_bio2023 - main - RStudio". In the center, there is a code editor window with two tabs: "Untitled1" and ".gitignore". The ".gitignore" tab is active, displaying the following content:

```
1 .Rproj.user
2 .Rhistory
3 .RData
4 .Ruserdata
5 /presentation_images|
```

# Practice: collaborative version control

## Commits

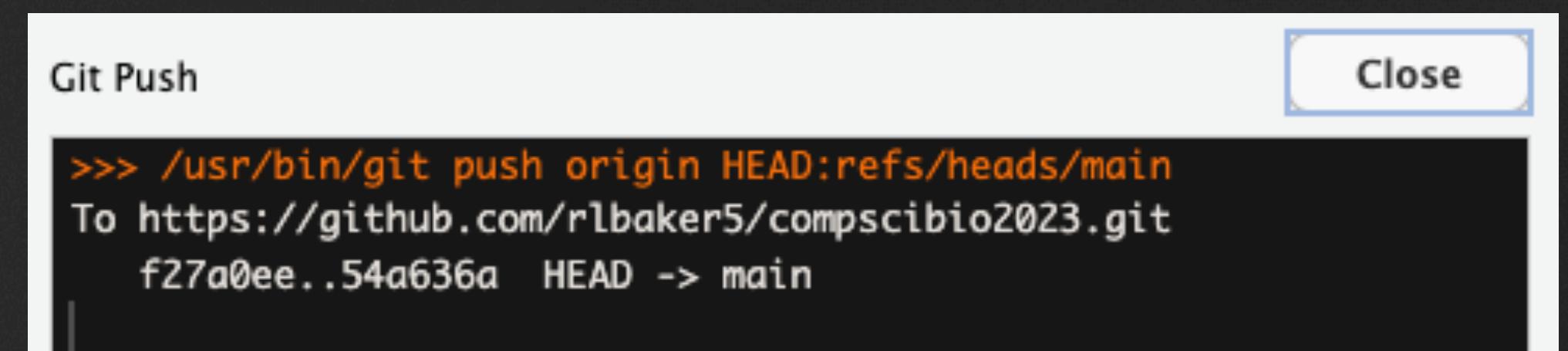
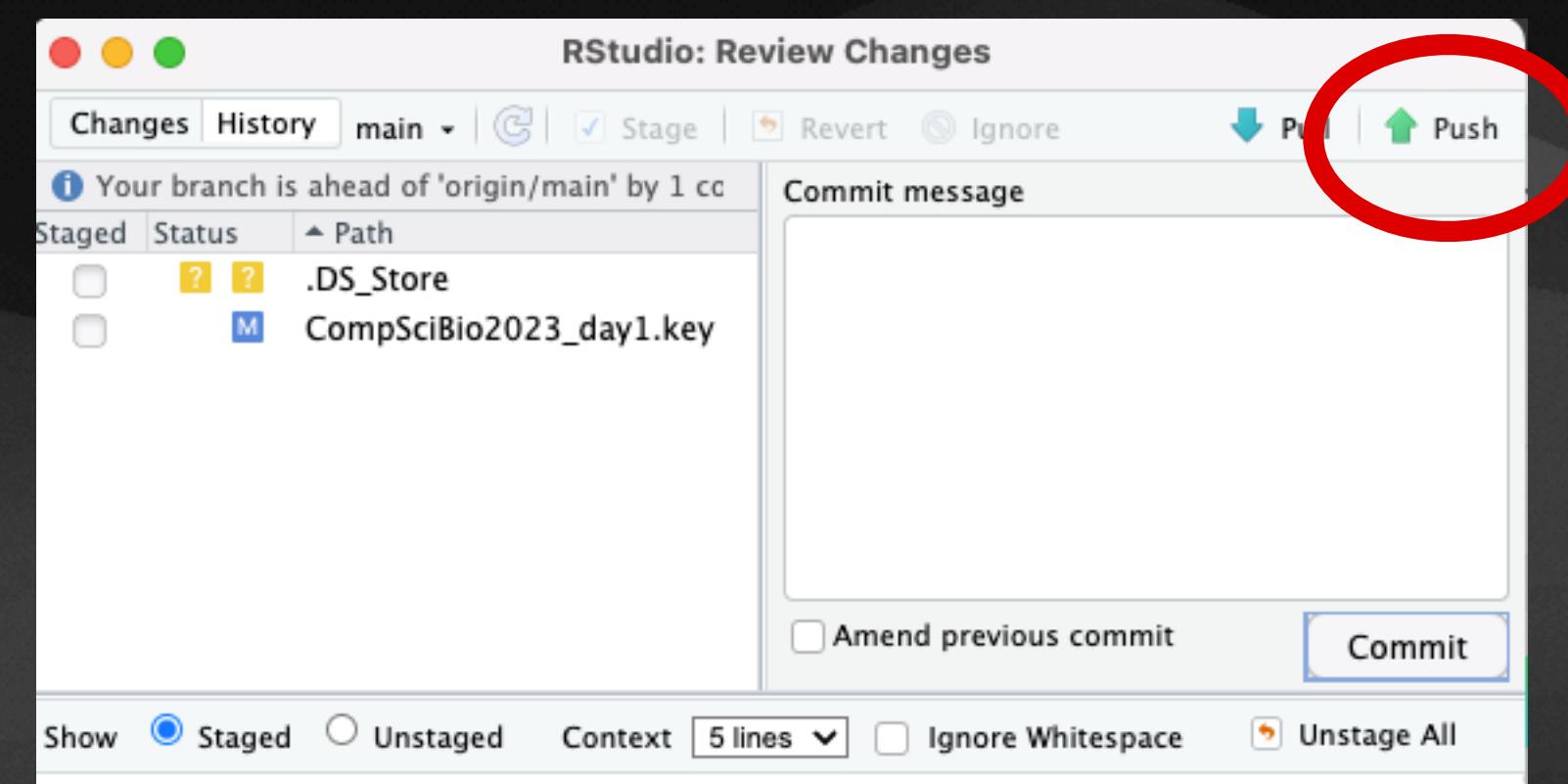
- In Rstudio, click on the “Git” drop down menu.
- Select “Commit” and a new window will open up.
- Select the file you changed and make sure to “stage” it by clicking the check box.
- Enter a comment into the Commit Message pane and then click “Commit”
- A new window will open with the results of your commit.
- Best practices: commit often, document well.



# Practice: collaborative version control

## Pushing to your GitHub repo

- Close the “Git Commit” Window.
- Notice that in the “Review Changes” Window your edited file is no longer listed (in my case I made changes to this keynote presentation and so it is listed)
- Click the “Push” button to push your commits to your GitHub repo.
- A new window pops up with the results of your Push.
- Best practices: push occasionally (when you are done with your current work, or at the end of the day)



# Practice: collaborative version control

## Inspect your GitHub Repo

- Navigate to your GitHub account and the repo you are working on
- URL should look something like:  
GitHub.com/username/reponame
- You should be able to see your time-stamped commit with your comment
- You can also see that your “branch” is ahead of the main branch that it was forked from.
- This is a great time to identify any mistakes, test out code, see if web pages and documentation render correctly, etc.

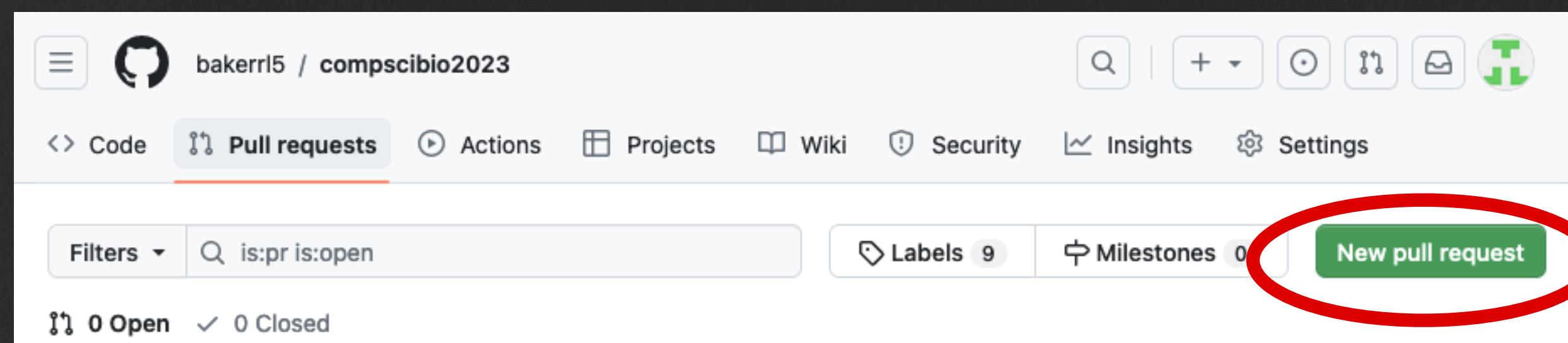
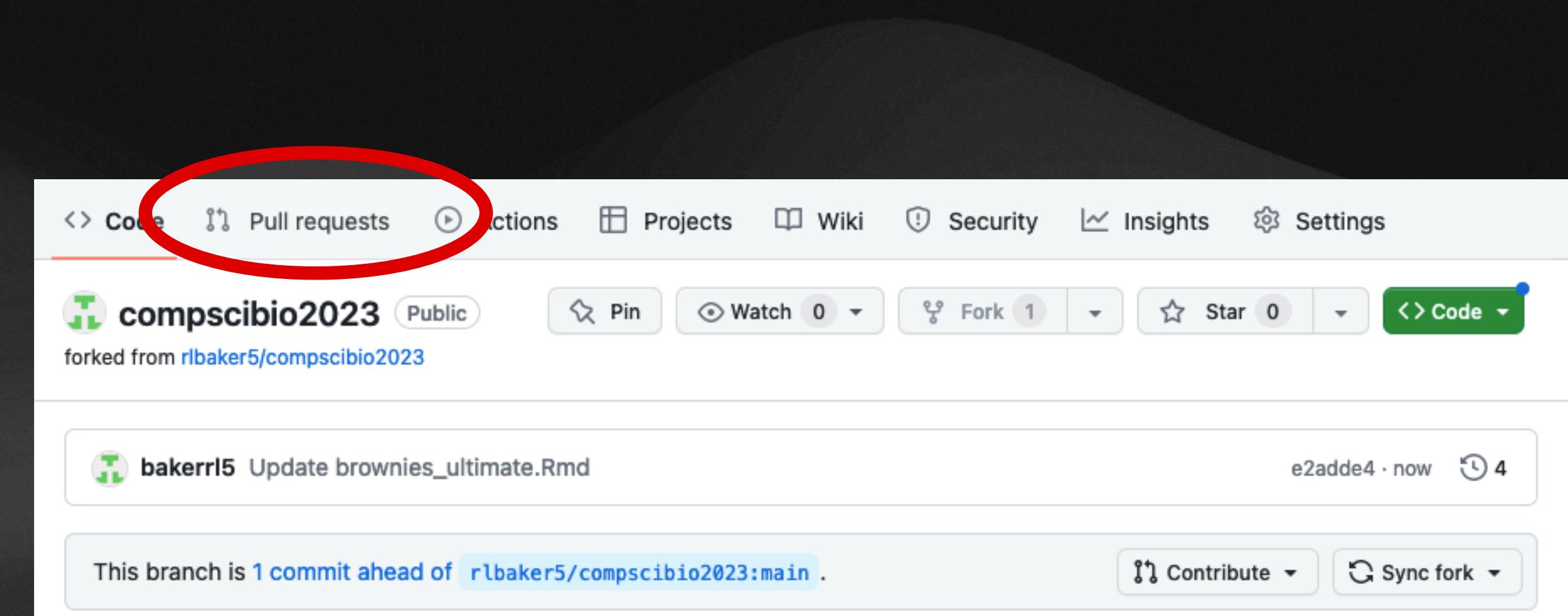
The screenshot shows a GitHub repository page for 'compscibio2023'. The status bar at the top indicates 'This branch is 1 commit ahead of rlbaker5/compscibio2023:main'. The commit history table lists several commits:

Commit	Message	Time
initial commit	CompSciBio2023_files/libs	28 minutes ago
initial commit	Recipes	now
initial commit	.gitignore	28 minutes ago
initial commit	CompSciBio2023_day1.key	28 minutes ago
initial commit	ComSciBio2023_Session1.Rmd	28 minutes ago
initial commit	Session1.Rmd	28 minutes ago
initial commit	Recipes	28 minutes ago

# Practice: collaborative version control

## Initiate a Pull Request

- Now it's time to (ask to) add your changes to the main repository (perhaps run by a lab manager?)
- Click on “Pull Requests”, which will take you to a new page.
- Click on “New Pull Request” to open yet another page.
- It’s worth inspecting this page closely to make sure you’re doing what you meant to do....



# Practice: collaborative version control

## Initiate a Pull Request

- Check to make sure you are moving files FROM the right place and TO the right place
- Hopefully GitHub tells you that the files can be merged automatically... (but if not, don't worry)
- Check that the files you wanted to change are the ones being changed
- You can even inspect the specific changes (again)
- When you're happy, click “Create pull request”

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: rbaker5/compscibio2023 ▾ base: main ↺ head repository: bakerrl5/compscibio2023 ▾ compare: main ▾

Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

-o 1 commit

1 file changed

1 contributor

Commits on Jul 27, 2023

Update brownies\_ultimate.Rmd

bakerrl5 committed 1 minute ago

Verified e2adde4

Showing 1 changed file with 2 additions and 2 deletions.

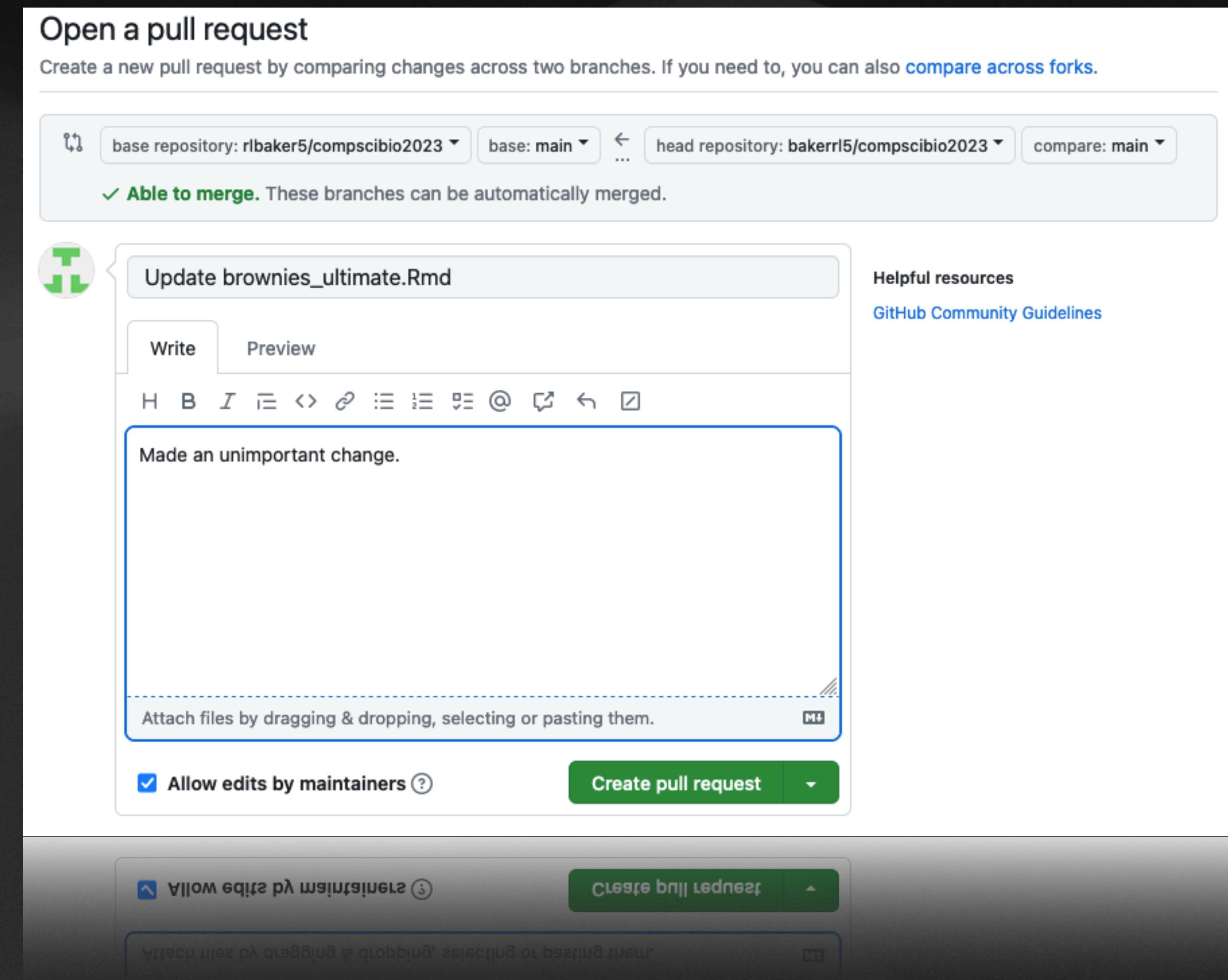
Recipes/brownies\_ultimate.Rmd

...	@@ -5,7 +5,7 @@	
5	5	## Brownies
6	6	
7	7	* .75 cups cocoa (unsweetened)
8		- * 0.66 cup margarine, melted
	8	+ * 0.66 cup margarine, melted
	8	+ * 0.66 cup margarine, melted
	8	- * 0.66 cup margarine, melted
1	1	* 12 cups cocoa (unsweetened)
2	2	150 MINUTES

# Practice: collaborative version control

## Initiate a Pull Request

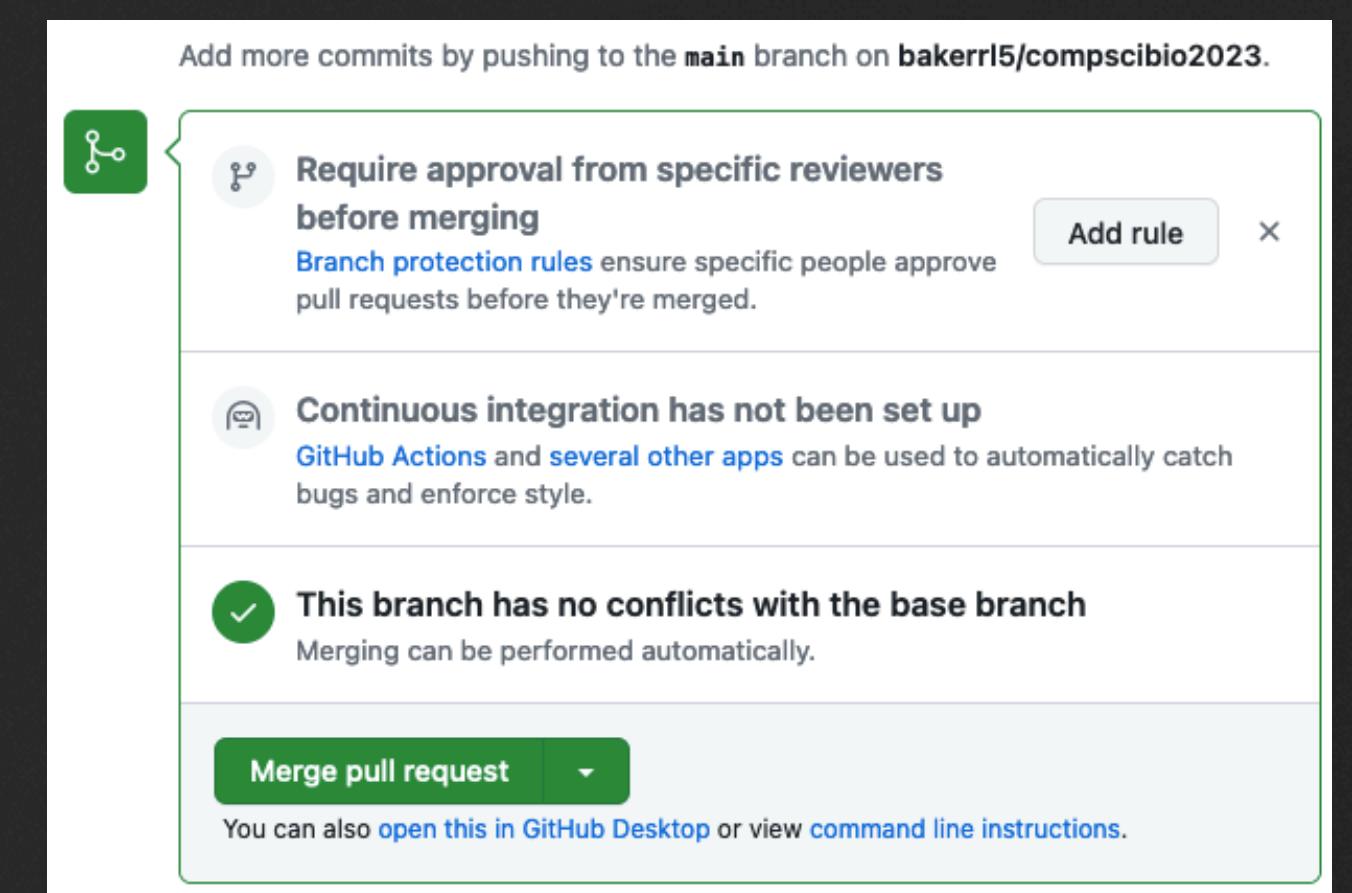
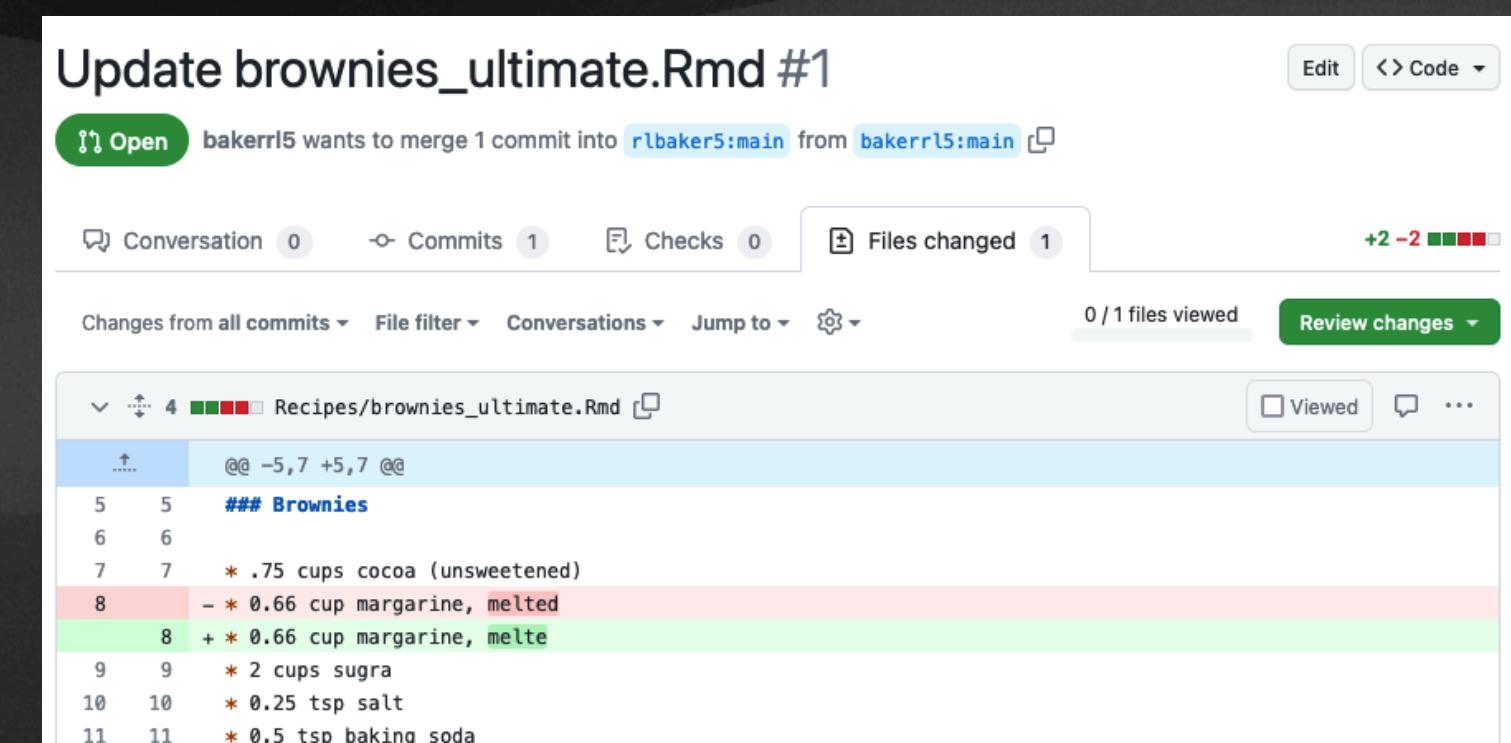
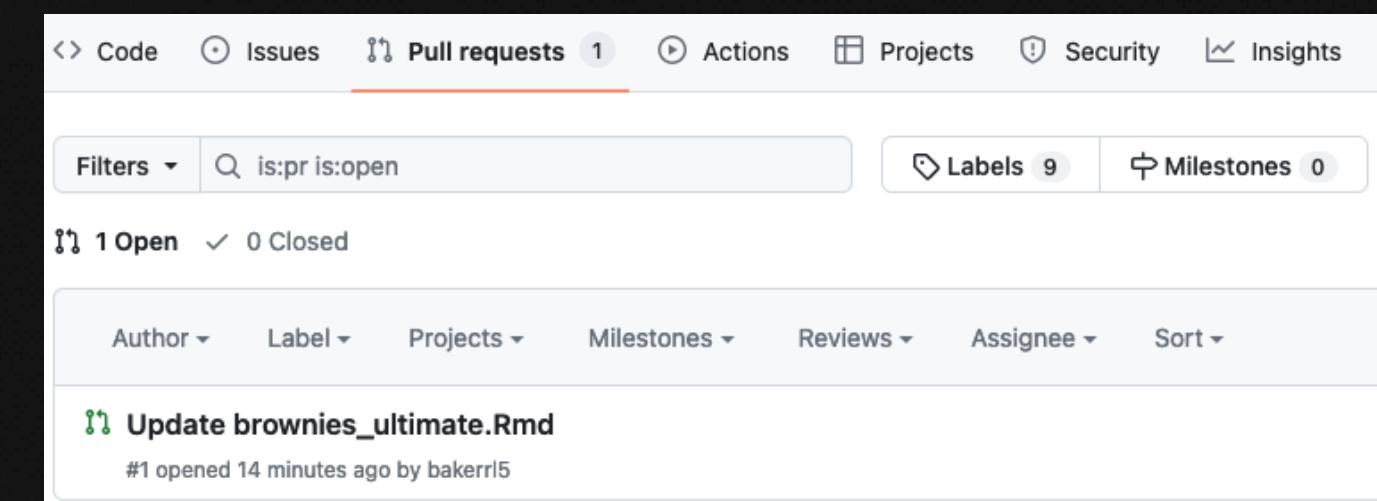
- GitHub may have tried to give your Pull Request a title, which you can edit.
- You can also add some helpful comments so that whoever is reviewing your Pull Request knows what you did (or tried to do)
- Once you are happy, click “Create Pull Request”.
- Whoever controls the main repository will be notified that you want to merge in changes.
- You can't delete the Pull Request! However, you can still add more comments or you can close it.
- Best practices: open Pull Requests infrequently and only when you are done with whatever task you were working on (or if you need to hot-fix a bug!).
- Best practices: Pull Requests can also be a great way of asking for a code review, even if you're not quite ready to merge into main.



# Practice: collaborative version control

## What happens to your Pull Request?

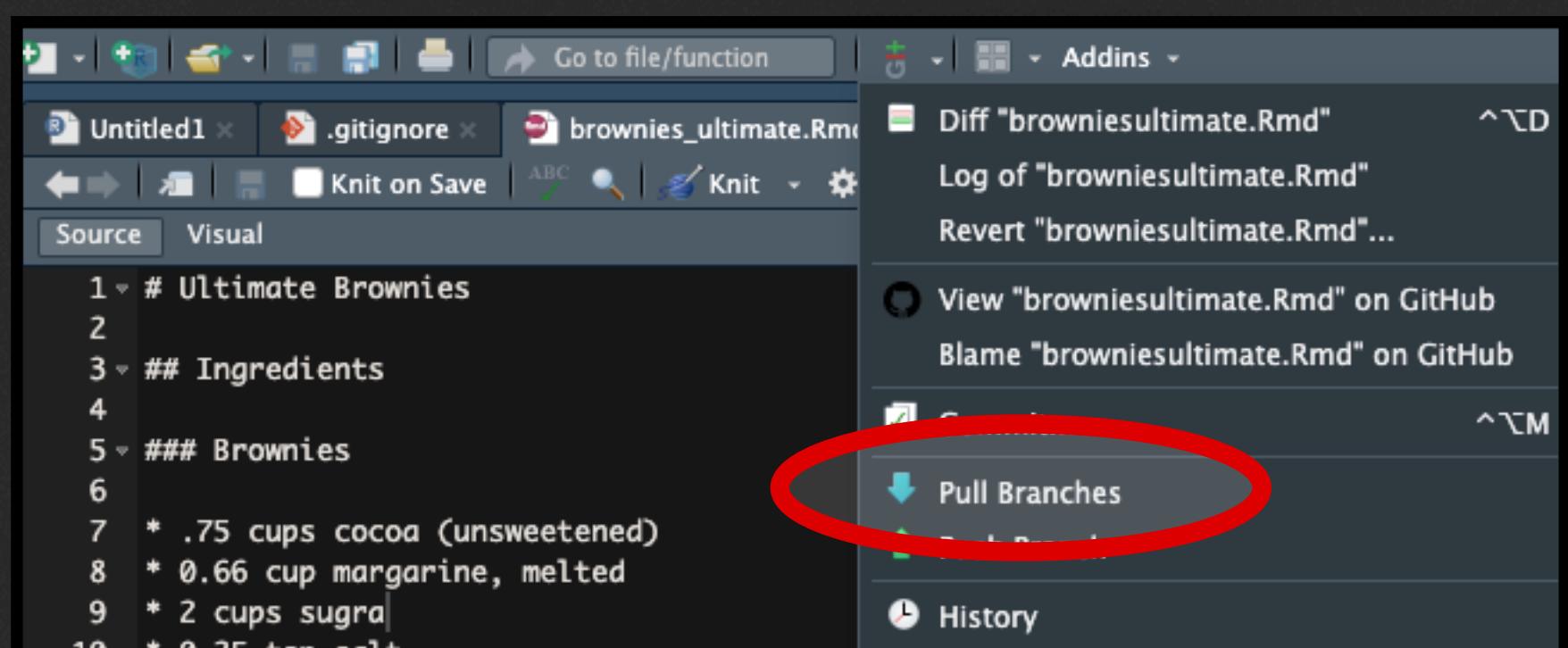
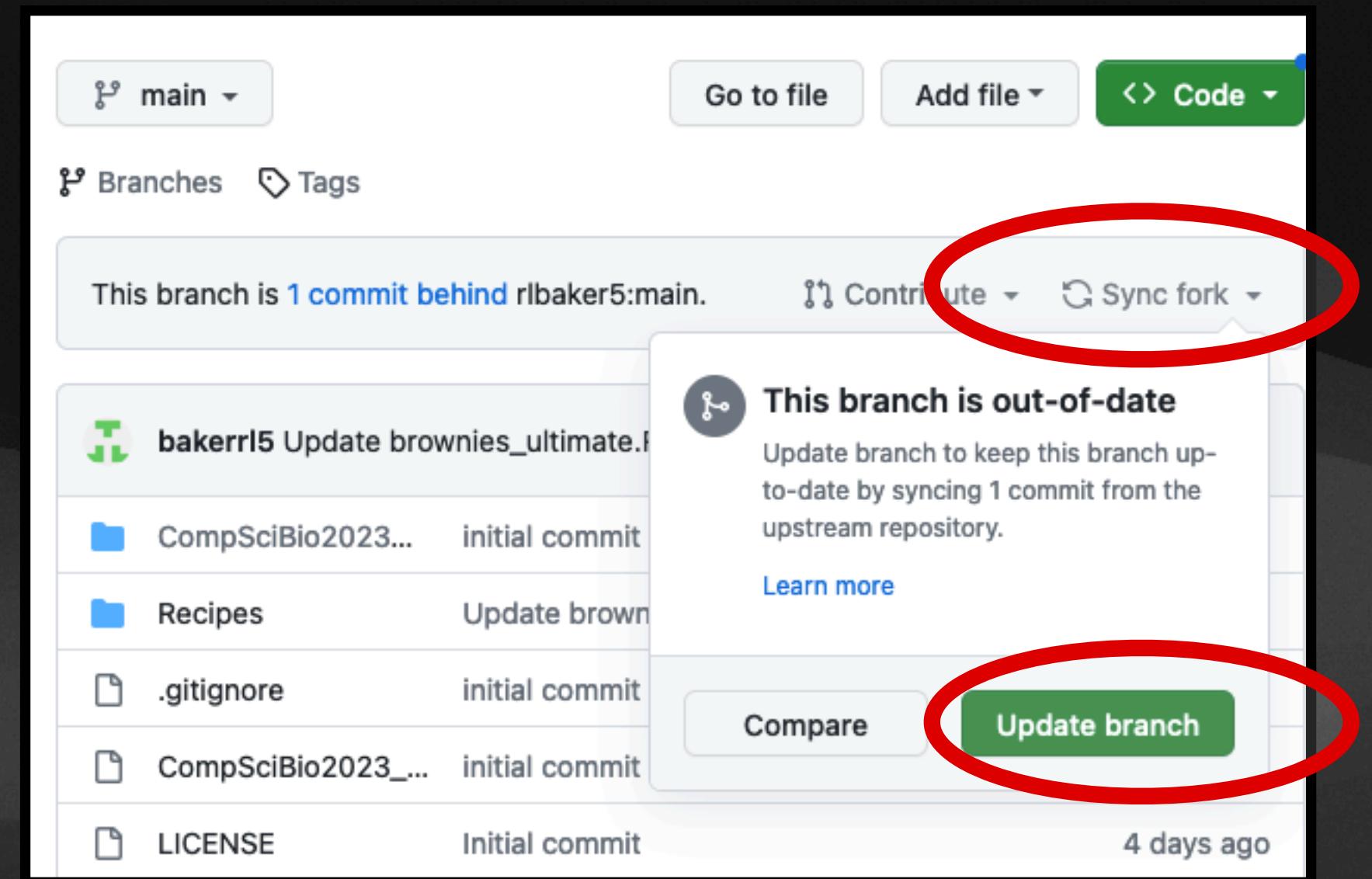
- Whoever has access to the repo you requested your changes be pulled into gets notified that there is an open Pull Request.
- They can examine your PR, look at specific changes you made, and can comment on it to request clarification or approve the changes.
- They can ignore it, close it, or merge it in (at which point it automatically closes)



# Practice: collaborative version control

## What Next?

- If your PR gets merged, there is a little maintenance you need to do to on your end:
- Go to your fork of the repo on GitHub and sync it to the main repo (notice the “merge” counts as a commit so your forked repo is already behind the main repo by 1 commit!)
- In Rstudio, go to the Git tab and select “Pull” to pull down any changes and update your local copy.
- Any time you start to work on a project you should sync it and then pull it. Don’t just open Rstudio and start working!



# Practice: collaborative version control

## Merge Conflicts: here be dragons!

- If multiple people are each working on different files, Git/GitHub can automatically merge the changes
- If multiple people are working on different lines within the same file, Git/GitHub can automatically handle the merge
- Merge Conflict: multiple people work on the same line in the same file generates a Merge Conflict that must be manually resolved.

The screenshot shows two GitHub interface snippets illustrating a merge conflict.

**Top Snippet: Comparing changes**

This snippet shows a comparison between two repositories:

- base repository: rlbaker5/compscibio2023
- base: main
- head repository: bakerrl5/compscibio2023
- compare: main

A red error message states: "Can't automatically merge. Don't worry, you can still create the pull request." A blue button at the bottom right says "Create pull request".

**Bottom Snippet: Pull Request Detail**

This snippet shows a pull request titled "Update recipe\_credits.Rmd #2" from the user "bakerrl5".

Key details of the pull request:

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

The commit message from "bakerrl5" reads: "Sorry for the extra work manually correcting merge requests. Better luck next time."

At the bottom, it says: "Add more commits by pushing to the **main** branch on **bakerrl5/compscibio2023**".

**Conflict Resolution Alert**

An alert message indicates: "This branch has conflicts that must be resolved". It notes: "Only those with [write access](#) to this repository can merge pull requests." It lists "Conflicting files" and "Recipes/recipe\_credits.Rmd".

# Practice: collaborative version control

## Resolving Merge Conflicts

- Best Case Scenario: merge conflicts are simple and can be resolved via GitHub web page.
- The merge conflicts are identified; delete the rows you want to remove and delete the indicators.
- Click “Mark Resolved”; Commit the merge.
- Merge the Pull Request. Done!

The screenshot shows a GitHub interface for resolving merge conflicts in an R Markdown file named `Recipes/recipe_credits.Rmd`. The top status bar indicates "2 conflicts". The main area displays the code with conflict markers. A red circle highlights the conflict markers around line 14, which contains the text "## Nonna (1901-1981)". A modal dialog titled "1 conflicting file" shows the conflict details, including the conflicting file name and the specific line number. The bottom right corner of the modal has a green checkmark and the text "✓ Resolved". The code itself describes family recipes and contributors, including Nonna, Grandma, and Mom.

```
1 ---  
2 output:  
3   word_document: default  
4   html_document: default  
5   pdf_document: default  
6 ---  
7 # Credits for these recipes  
8  
9 These recipes are all family recipes that have been traded around, handed down, modified and loved by several generations. I've st  
10  
11 Four women contributed the most to these recipes.  
12  
13 <<<<< main  
14 ## Nonna (1901-1981)  
15 "Nonna " was born Leontina Lucci Cianfuglia and after marrying  
16 ## Grandma (1922-2019)  
17 "Grandma" is Elizabeth Sestini, was born in the U.S. on April 18, 1922.  
18  
19 <<<<< main  
20 ## Nonna (1901-1981)  
21 "Nonna" was born Leontina Lucci Cianfuglia and after marrying  
22 ## Grandma (1922-2019)  
23 "Grandma" is Elizabeth Sestini, was born in the U.S. on April 18, 1922.  
24  
25 >>>> main  
26  
27 >>>>> main  
28 ## Nonna (1901-1981)  
29 "Nonna" was born Leontina Lucci Cianfuglia and after marrying  
30 ## Grandma (1922-2019)  
31 "Grandma" is Elizabeth Sestini, was born in the U.S. on April 18, 1922.  
32 ## Grandma Baker (19??-1982)  
33 "Grandma Baker" was born Viola Robie in the north woods of Wisconsin.  
34  
35 ## Mom (1947 - )  
36 "Mom" is Susan Elizabeth Sestini Baker. Its hard to put into words.  
37  
38 ## Wow (1941 - )  
39 "Wow" is Susan Elizabeth Sestini Baker. Its hard to put into words.  
40  
41 >>>>> main
```

# Practice: collaborative version control

## Complicated merge conflicts:

- The best way to fix them is avoid them: communicate with your collaborators!
- Require editing off-line, but the concept is the same: delete what you don't want and keep what you do
- Additional issues can arise if you aren't careful
- Its hard to screw up so badly it's not recoverable
- Most mistakes can be fixed after some googling
- Complex problems are beyond the scope of this module

# Course Outline

- I. Introductions
- II. Concepts of version control and collaborative coding
- III. Sign up for a GitHub account
- IV. Walk through and help installing/upgrading software
- V. Rstudio/GitHub integration
- VI. Version Control and Collaborative Coding Workflows
- VII. **Practice: collaborative version control**
- VIII. **Bonus: project management boards, issues, milestones**

# GitHub project management

## Projects

- Projects are a nice way to keep track of your repo (or even many repos at once): what needs to be done, what is in progress, what has been done.
- I like “Boards” view, but super customizable
- You can customize the columns and add “issues” to each.

The screenshot shows the GitHub repository interface for 'rlbaker5 / compscibio2023'. The 'Projects' tab is selected. A search bar at the top contains 'is:open'. On the right, there's a sidebar with a 'Link a project' button and a 'New project' button, which is highlighted in blue. Below the sidebar, a message says 'Provide quick access to relevant projects' with a link to 'Add projects to view them here.'

The screenshot shows the GitHub Boards interface for 'rlbaker5 / PI / @rlbaker5's untitled proj...'. It displays a project board with three columns: 'Todo' (0 items), 'In Progress' (0 items), and 'Done' (0 items). Each column has a descriptive text below it: 'This item hasn't been started' for Todo, 'This is actively being worked on' for In Progress, and 'This has been completed' for Done. There are buttons for 'View 1', '+ New View', and 'Discard / Save' at the bottom.

# GitHub project management

## Milestones

- Milestones are a way of grouping issues
- You can break problems down into multiple issues, then assign them to a milestone
- Milestones help you keep track of progress towards a larger objective

The screenshot shows the GitHub Issues interface. At the top, there's a navigation bar with links for Code, Issues (which is highlighted), Pull requests, Actions, Projects (1), Wiki, Security, Insights, and Settings. A modal window titled "Label issues and pull requests for new contributors" is open, with a "Dismiss" button in the top right corner. Below the modal, there's a search bar with the query "is:issue is:open", and buttons for Labels (9) and Milestones (0). A green "New issue" button is also visible. The main content area has filters for Open (0) and Closed (0) issues, and dropdowns for Author, Label, Projects, Milestones, Assignee, and Sort. A section titled "New milestone" is displayed, with instructions to "Create a new milestone to help organize your issues and pull requests. Learn more about [milestones and issues](#)". The "Title" field contains "Week 2 sprint: Add some savory dishes". The "Due date (optional)" field is a date input set to "mm / dd / yyyy". The "Description" field contains the text "Add 1-5 recipes that fall more on the savory end of the spectrum rather than the sweet end." At the bottom right of the milestone form is a green "Create milestone" button.

# GitHub project management

## Advanced examples: Project Boards

The screenshot shows a GitHub Project Board titled "Dataverse planning & tracking". The board is divided into four columns: Ideas, Ready to do, In Progress, and Done.

- Ideas (17):**
  - Draft: What do we want dataverse to ultimately look like?
  - EMEditor #35: Set\_nps\_publisher function (Low, Medium)
  - NPSutils #22: Simple metadata analyses functions (Low, Medium)
  - NPSutils #21: Enable DataStore search from R (Low, High)
- Ready to do (1):**
  - EMEditor #99: get\_author\_list doesn't handle organizations (Medium, Medium)
- In Progress (3):**
  - NPSutils #29: getDataPackage: alert if data have been versioned
  - NPSutils #12: removeDataPackage function (or equiv. parameter for existing function)
  - NPSutils #27: update getDataPackages(): bad DS ref id (High, Low)
- Done (128):**
  - EMEditor #39: set\_content\_units: multiple unit (Medium, Low)
  - EMEditor #26: restful api connections to Data (Medium, High)
  - EMEditor #41: set\_drr\_doi -> set\_drr in docum (High, Low)
  - EMEditor #42: Correct documentation for set\_ (High, Low)

At the top of the board, there is a navigation bar with tabs for "Table" and "Overview", and several project-related buttons: QCKit, EMEditor, DPChecker, NPSutils, and NPSdataverse. There is also a "New View" button and a search bar. On the right side, there are buttons for "Discard" and "Save".

# GitHub project management

Advanced examples: Github Pages

# Credits and additional info

- Happy Git with R: <https://happygitwithr.com/> (you may notice we followed this very closely!)
- Posit Version control: <https://support.posit.co/hc/en-us/articles/200532077>
- Configuring R/GitHub: <https://gist.github.com/z3tt/3dab3535007acf108391649766409421>
- Data Carpentries Workshops: <https://datacarpentry.org/>
- Various amounts of googling!