Week 6: git & GitHub

Version Control

Joe Nese

University of Oregon

Fall 2022

git & GitHub

Week 6

Agenda

- Introduce reasons for version control
- Introduce
  - git
  - GitHub
  - GitKraken

Learning Objectives

- ~~Understand~~ Introduce the basics of git and git terminology
- Be able to create *repos*, *commit* changes, *push*, *clone*, and *pull*
- Be prepared to collaborate on a project

git & GitHub

What are they?

git

- a version control system
- lives on your computer and keeps track of files and all changes made to them that you register with *commits*

GitHub

- an online host for *repositories* of code and data
- lives on the internet

Credit Nathan Brouwer

git & GitHub

Why are they important?

Backup: Easy way to upload files and add notes about changes. Excellent way to keep all your R work backed-up on the cloud

Version Control: Complete history of all changes and why they were made, and multiple branches of history

Open Access: Open, transparent, and public science. Some journals allow GitHub sites to be used to host data and code to meet their open-access requirements

Visibility: GitHub makes it easy to make your open-access work visible to people who might be interested (collaborators, employers)

Collaborative tool: Git and GitHub are designed for collaboration (even if you're the only one coding). Easy to share files between collaborators, allows for merging work and reconciling conflicts.

Job Skill: This is an important data science skill

Credit Nathan Brouwer

git & GitHub

How?

1. Install git
2. Create a GitHub account (hopefully you all have done this)
3. Get a GUI and/or learn the command-line tools
    - I'm okay with either or both, as long as you (sort of) know what you're doing
    - We're going to use GitKraken in this course
4. Develop a solid understanding of the underlying structure, so you know what you want to do even if you don't know how
    - *commit*, *push*, *pull*, *fork*

I HATED GIT
NOW I HATE IT LESS
memegen.com

git can be frustrating, but...

It is transparent – mistakes and all

Can be a powerful tool for collaboration

- Communication between team members is important
- Try to avoid merge conflicts
    - One person working on one thing at a time
    - *Pull* first each and every time!
    - *Push* often/at logical points

Excellent for reproducibility (a tenet of the data science pipeline)

- Open data and code
- Transparent
- Forking paths

Notes on Open & Reproducible Research

- Use scripting
- Use dynamic documents
  - R Markdown is fantastic
  - Better than Jupyter notebooks; see here and here
- Use Git/GitHub
- Use open data whenever possible
  - Consider ways you can make portions of the data open, if not all of it
- Advanced – use tools that store software versions
  - `{renv}` (Rstudio)
  - `{target}` (formely drake)
  - `{holepunch}`

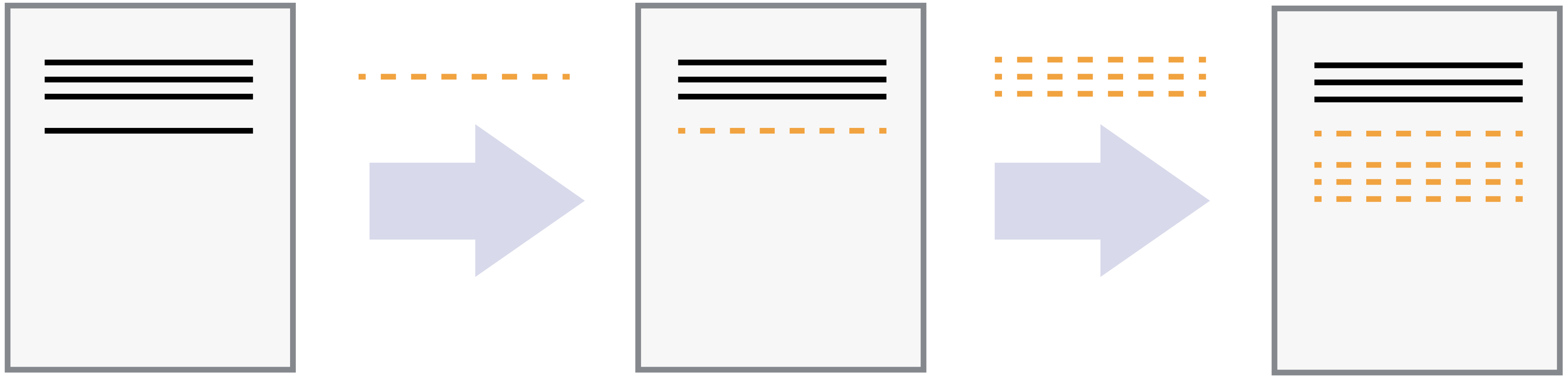
All the things

- **git** – a version control system that lives on your machine
- **GitHub** – repo host that lives online, works with git
- **GitKraken** – a GUI that lives on your machine and communicates with GitHub and git

Version Control

We can think of the changes as separate from the document

You can then think about “playing back” different sets of changes on the base document, ultimately resulting in different versions of that document



swcarpentry

Version Control

This means there are many possible versions of the same document

Version Control

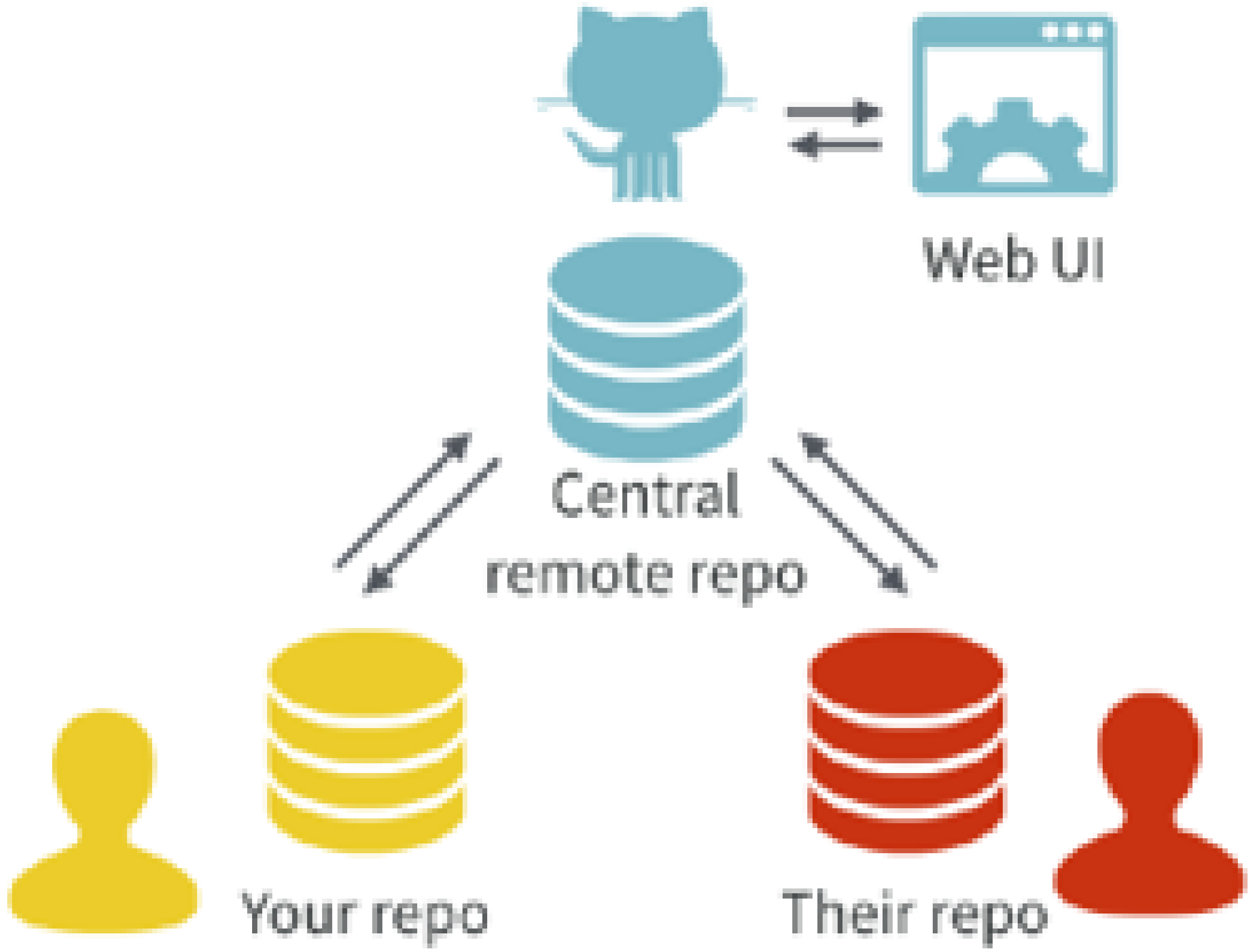
Unless there are merge conflicts, two changes from the same document can be merged together

Let's talk about "how"

- Understand the structure of git
- Actual method of completing the tasks is up to you
  - GUI? Command line? RStudio?
- I've asked you to install GitKraken
  - Worth it to manage merge conflicts alone
- If you're already using something else, that's ok!
  - Let me know

Terminology

- **version control system**
  - A tool to help us track changes. git is one such system (but there are others)
- **repository (repo)**
  - The files, full commit history, and associated metadata for a project
- **remote**
  - The cloud-based server hosting the repo (that's GitHub)


Web UI
Central remote repo
Your repo
Their repo

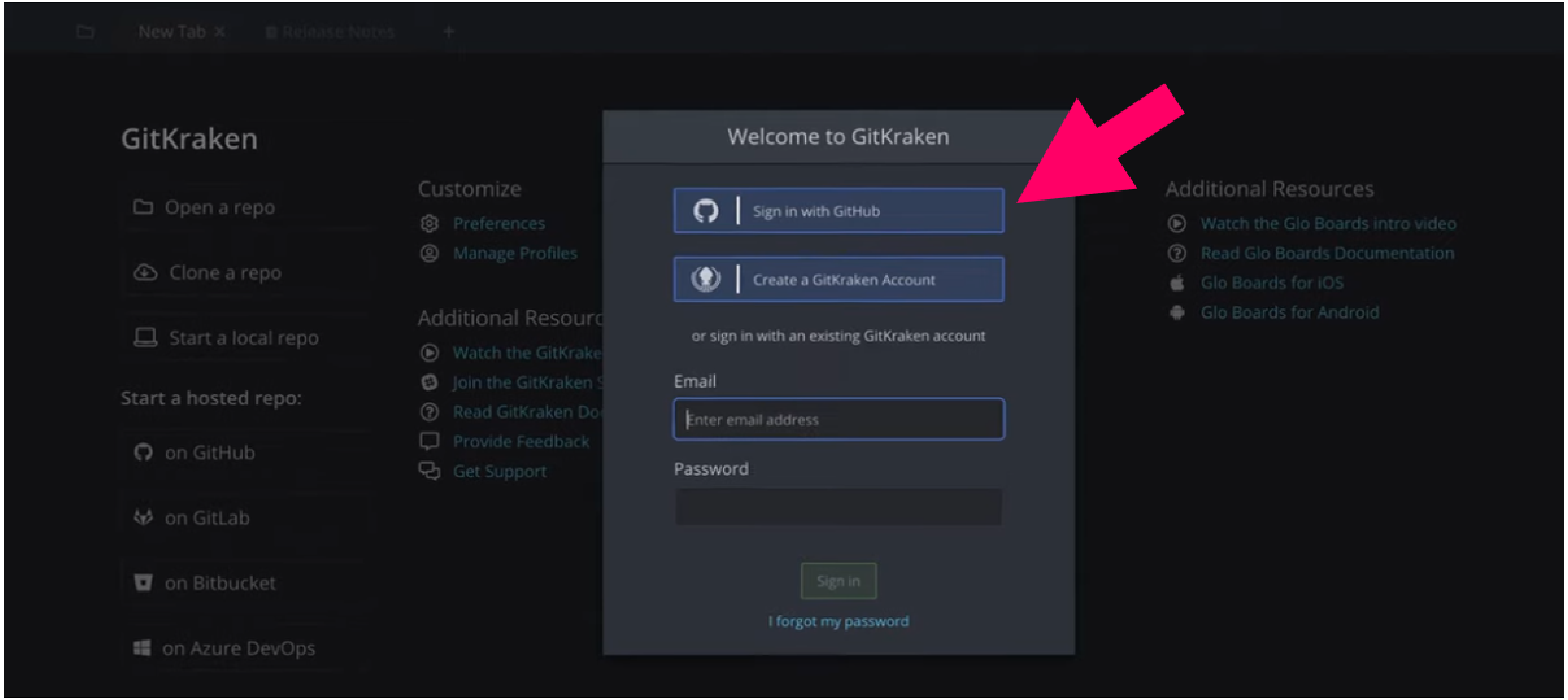
Let's connect GitHub and GitKraken

Connect GitHub & GitKraken

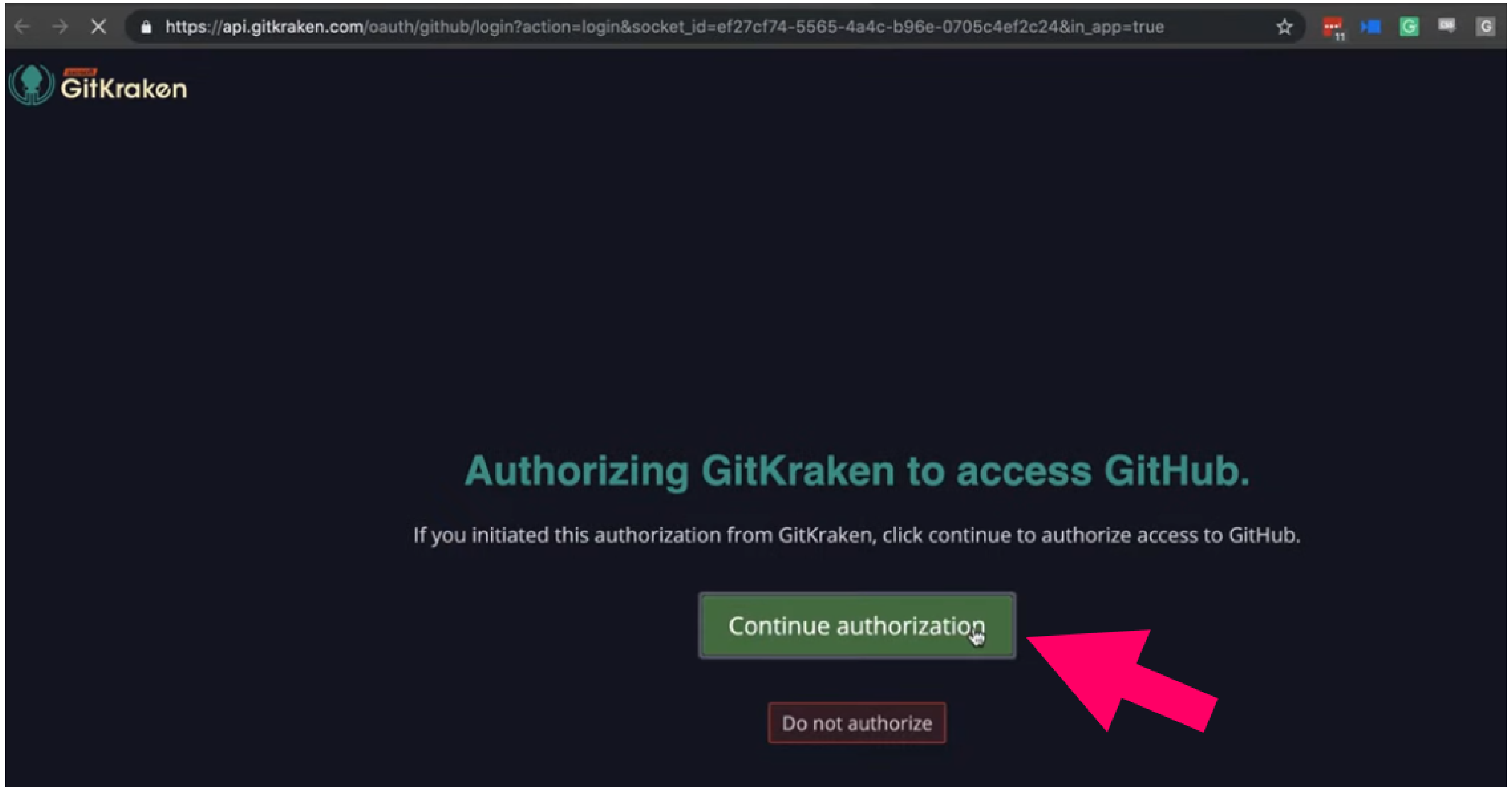
Go to GitKraken

1. "Sign in with GitHub"
2. "Continue with Authorization"
3. You should see *“Success: GitHub and GitKraken are good to go!”* or something like that
4. Check your email for an email from GitHub to verify (?)

Open GitKraken



Connect GitHub and GitKraken



https://api.gitkraken.com/oauth/github/login?action=login&socket_id=ef27cf74-5565-4a4c-b96e-0705c4ef2c24&in_app=true

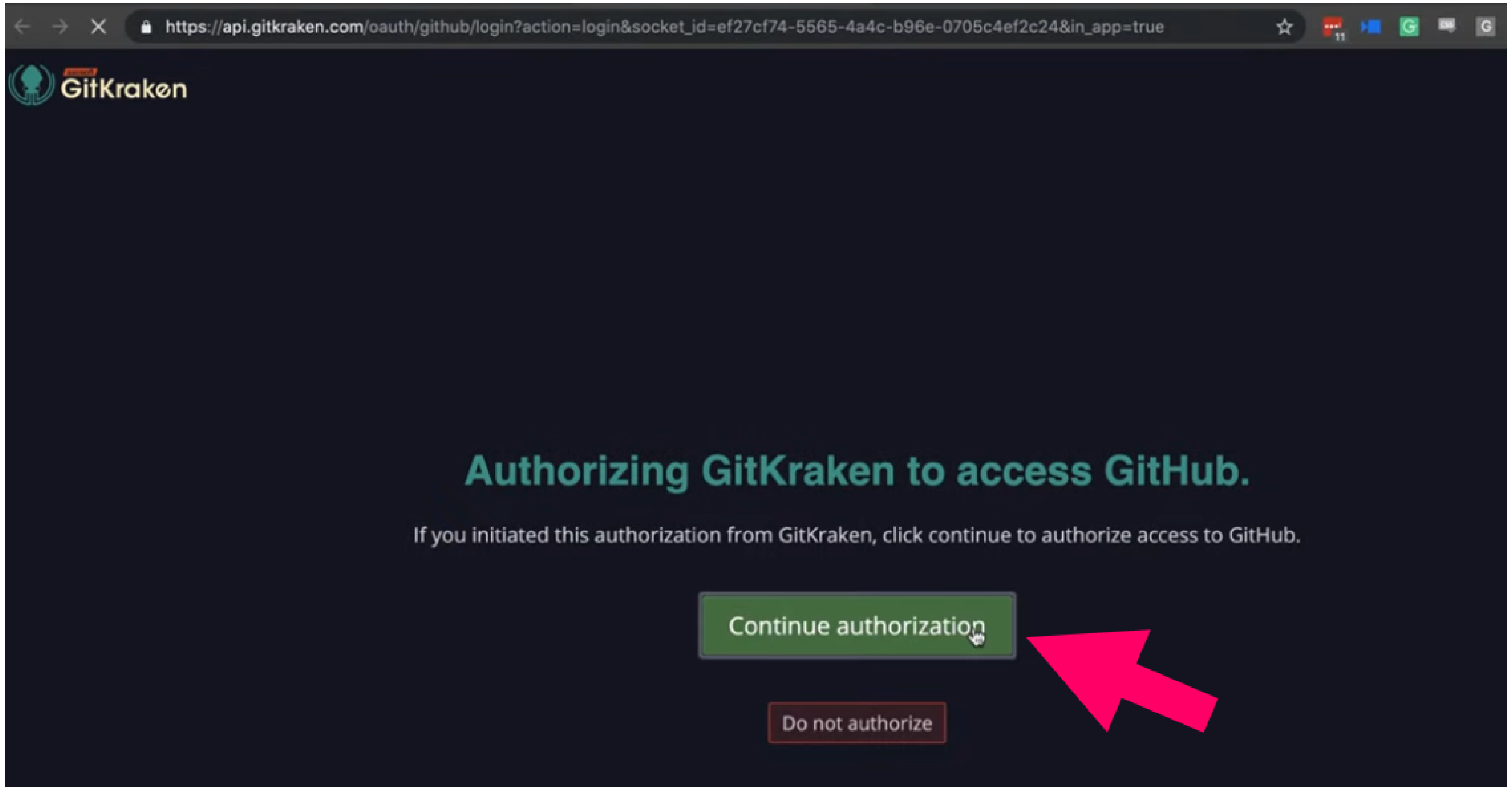
GitKraken

Authorizing GitKraken to access GitHub.

If you initiated this authorization from GitKraken, click continue to authorize access to GitHub.

Connect GitHub and GitKraken

Check your email (associated with your GitHub account)



https://api.gitkraken.com/oauth/github/login?action=login&socket_id=ef27cf74-5565-4a4c-b96e-0705c4ef2c24&in_app=true

GitKraken

Authorizing GitKraken to access GitHub.

If you initiated this authorization from GitKraken, click continue to authorize access to GitHub.

Incorporating git/GitHub/GitKraken

into your data science workflow

Proposed Workflow

1. Create your RStudio Project (as usual)
2. **GitKraken**: Open a *repo* on GitKraken
3. **GitHub**: Create new *repo* on GitHub
4. **GitKraken**: Add *remote* on GitKraken

1. Create your RStudio Project (as usual)

- Let’s make a new RStudio Project (cal it "*week6*")
  - Open a new R Markdown document and name it "*foo*"
- Note the name you gave the project (folder)
- ✅ "*Create a git repository*"
  - If ✅: Don't add `.gitignore` in GiKraken or GitHub
  - If 🚫: Add `.gitignore` in GitKraken **or** GitHub

1. Create your RStudio Project (as usual)

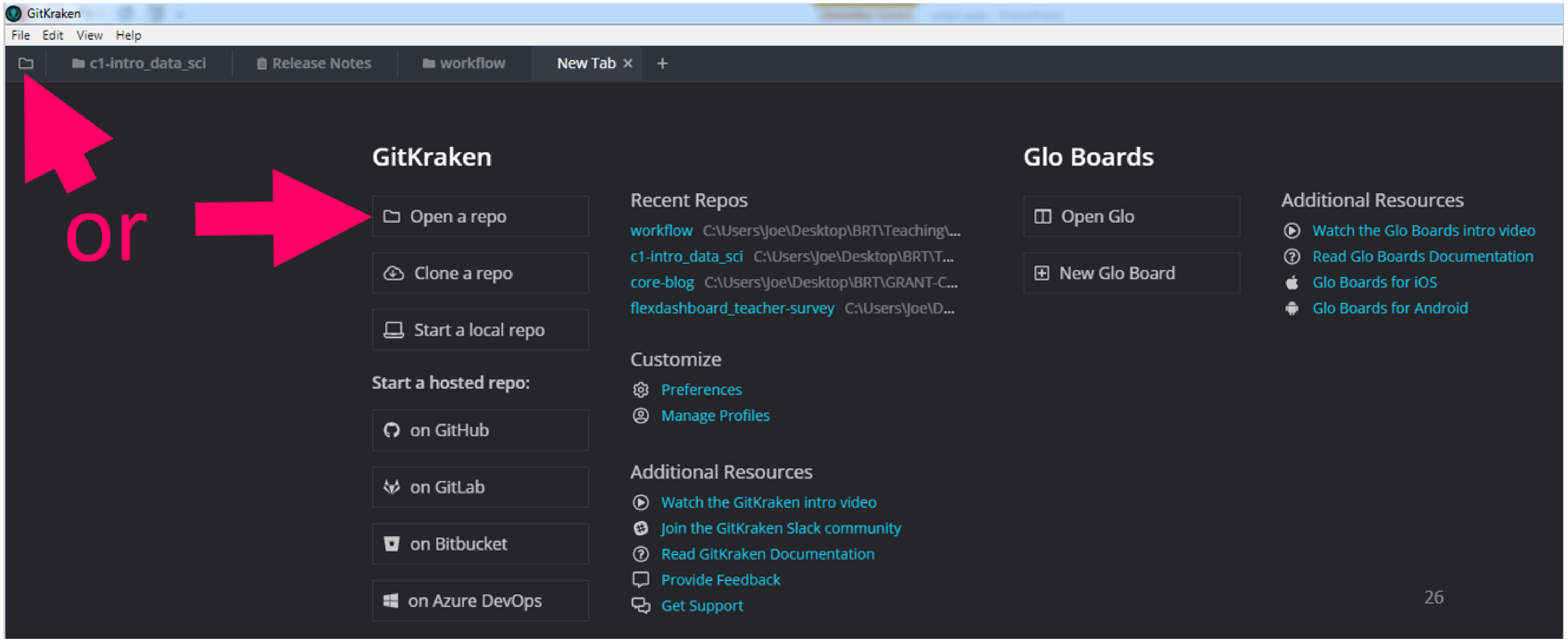
- Let’s make a new RStudio Project
  - Open a new R Markdown document and name it "*foo*"
- Note the name you gave the project (folder)
- "*Create a git repository*"
  - If ✅: Don't add `.gitignore` in GiKraken or GitHub
  - If 🚫: Add `.gitignore` in GitKraken **or** GitHub

2. Open a repo on GitKraken

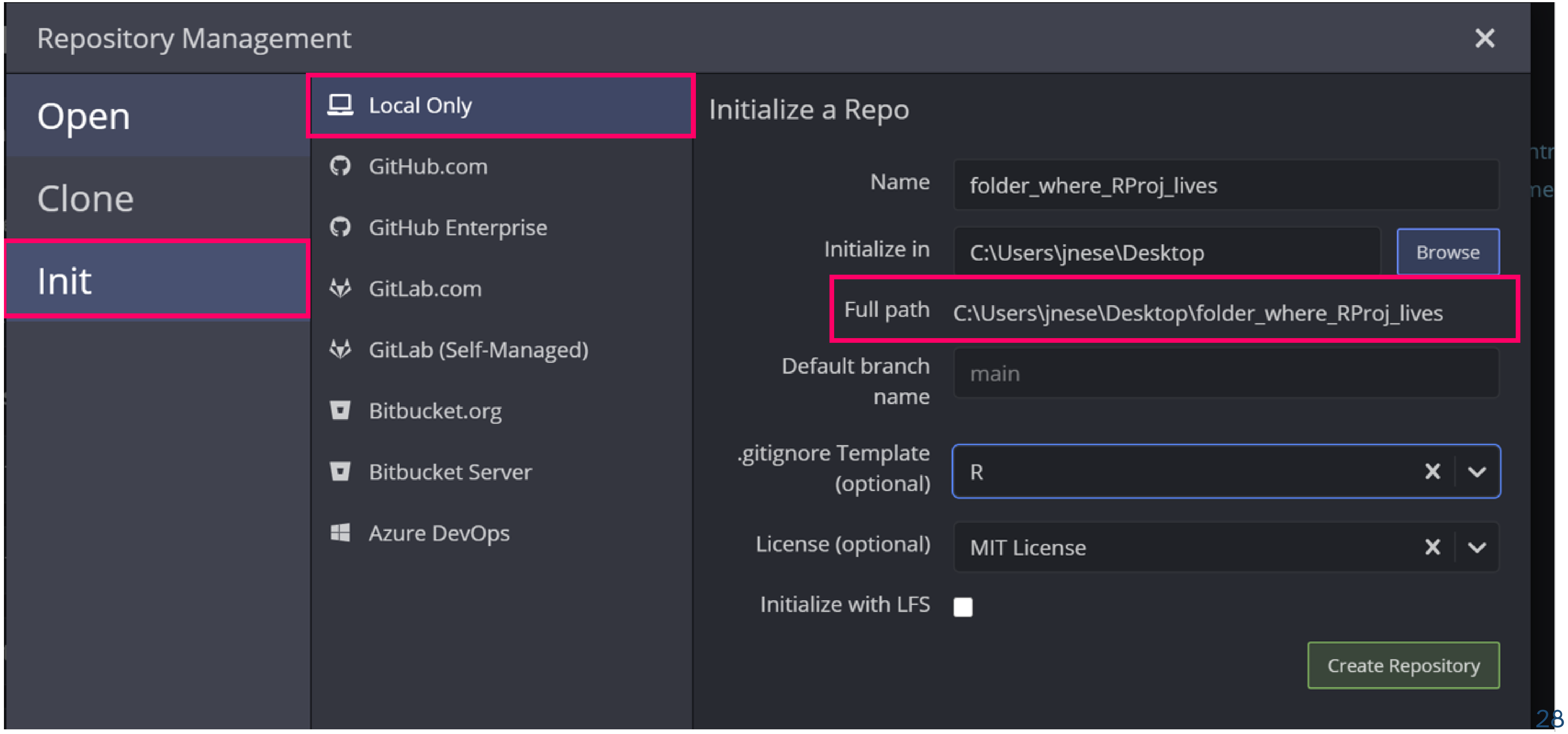
GitKraken

- Open a *repo*
- Init
- Local only

Open a repo (GitKraken)



Open a repo (GitKraken)


Repository Management

Open
Clone
Init

Local Only
GitHub.com
GitHub Enterprise
GitLab.com
GitLab (Self-Managed)
Bitbucket.org
Bitbucket Server
Azure DevOps

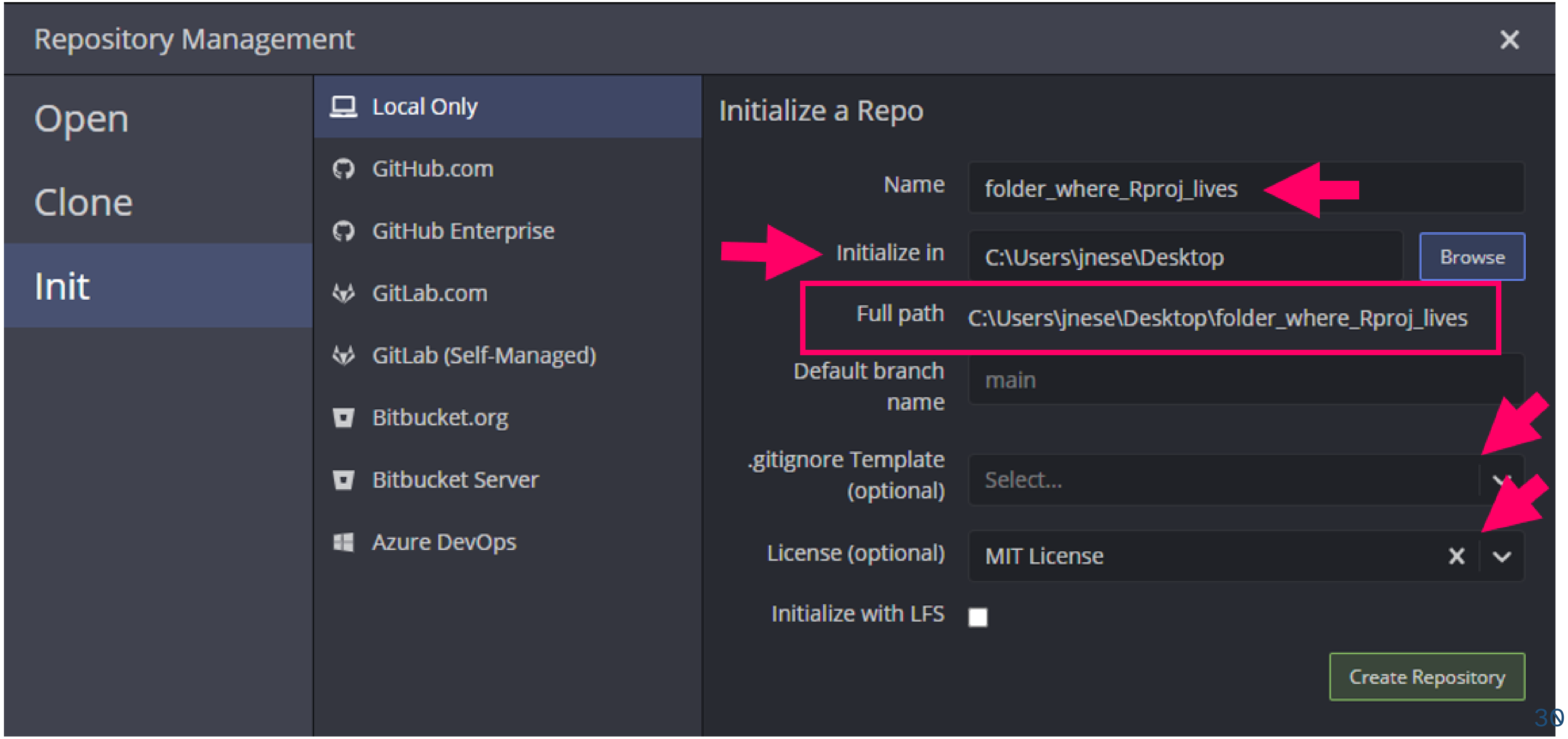
Initialize a Repo

Name: folder_where_RProj_lives
Initialize in: C:\Users\jnese\Desktop
Full path: C:\Users\jnese\Desktop\folder_where_RProj_lives
Default branch name: main
.gitignore Template (optional): R
License (optional): MIT License
Initialize with LFS: ☐

Open a repo (GitKraken)

- "*Name*"
  - folder name where the Rproject lives
- "*Initialize in*"
  - Browse to the parent folder where your Rproject lives
- "*Full path*"
  - "*Full path*" = "*Initialize in*" + "*Name*"
  - The full path should read to the path where your Rproject lives
- "*Default branch name*"
  - "main"
- "*.gitignore Template*"
  - If already created one with your project, leave empty
  - If none yet created, choose "R"
- "*License (optional)*"
  - MIT License

Open a repo (GitKraken)



Repository Management

Open
Clone
Init

Local Only
GitHub.com
GitHub Enterprise
GitLab.com
GitLab (Self-Managed)
Bitbucket.org
Bitbucket Server
Azure DevOps

Initialize a Repo

Name: folder_where_Rproj_lives
Initialize in: C:\Users\jnese\Desktop
Browse
Full path: C:\Users\jnese\Desktop\folder_where_Rproj_lives
Default branch name: main
.gitignore Template (optional): Select...
License (optional): MIT License
Initialize with LFS

Create Repository

3. Create new repo on GitHub

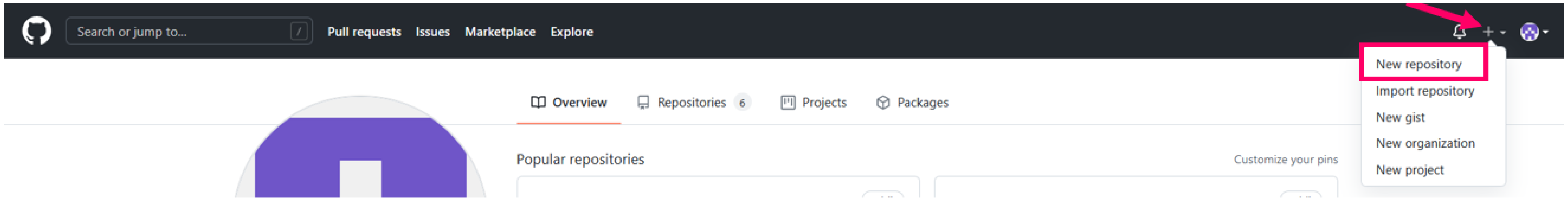
GitHub

(1) Create new *repo*

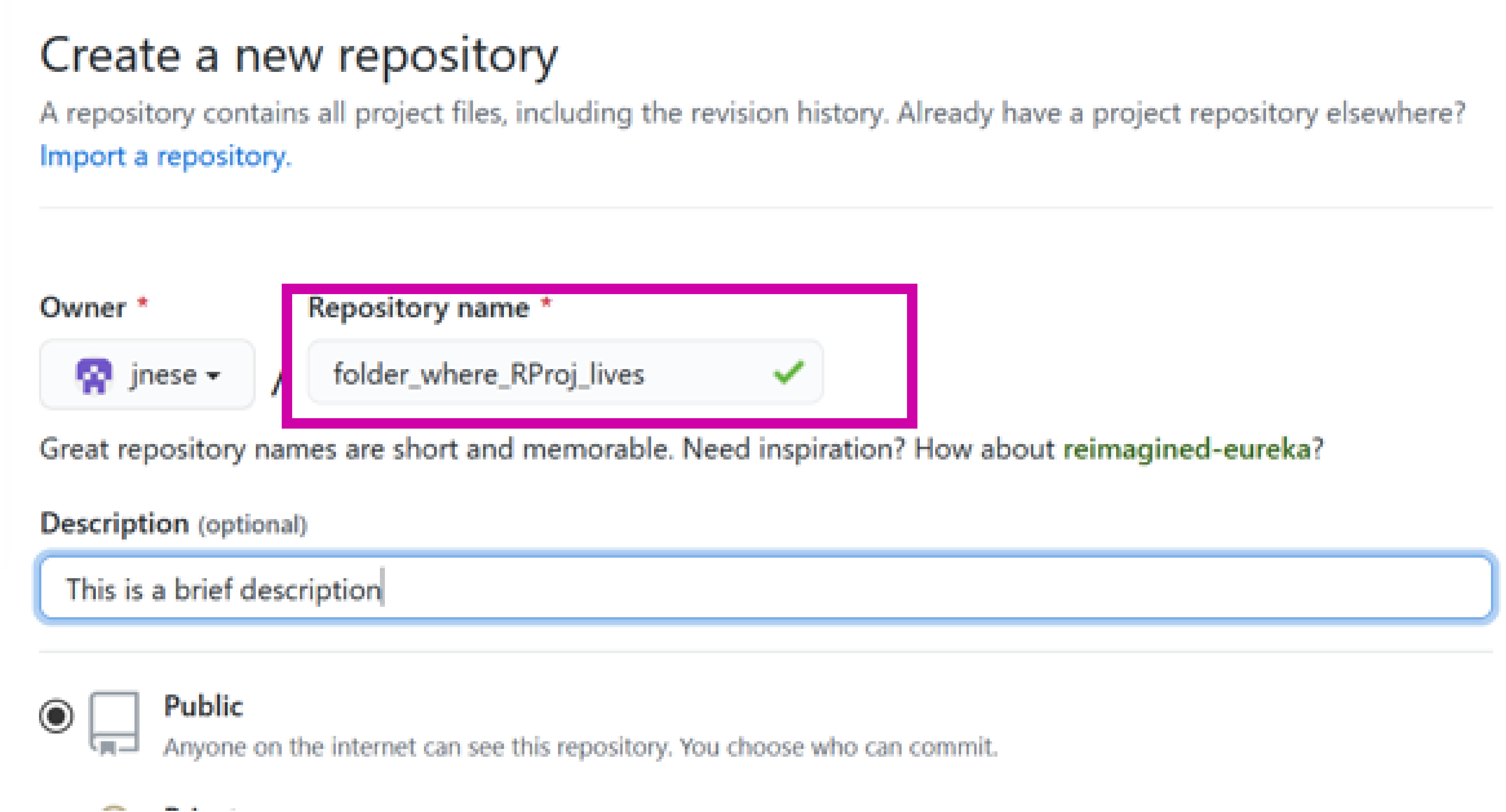
- **Repository name** – same as your Project name
- **Description** – keep it brief, informative
- **Public**
- **Do not initialize**
- **"None"**
- do **NOT** add `.gitignore`
- do **NOT** add a license

(2) Copy URL from GitHub

3. Create new repo on GitHub



3. Create new repo on GitHub



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

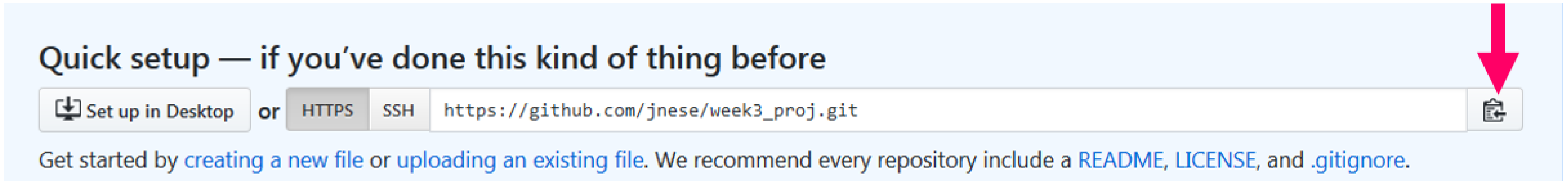
Owner *
jnese
/
Repository name *
folder_where_RProj_lives ✓

Great repository names are short and memorable. Need inspiration? How about reimagined-eureka?

Description (optional)
This is a brief description

[● Public]
Anyone on the internet can see this repository. You choose who can commit.

Copy URL from GitHub



Quick setup — if you've done this kind of thing before

```
https://github.com/jnese/week3_proj.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

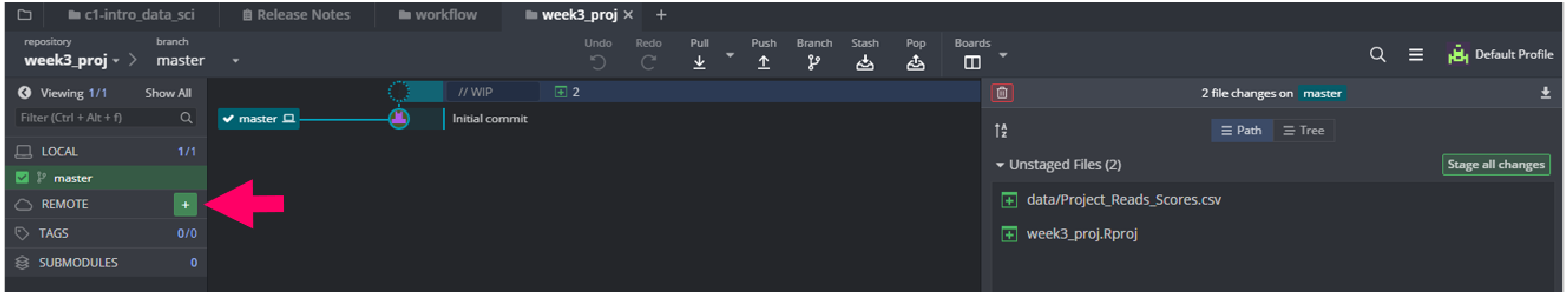
4. Back to GitKraken to Add Remote

GitKraken

Add REMOTE

- Same Name used thus far
- Click 🌐 "*URL*"
- Paste URL

REMOTE


c1-intro_data_sci
Release Notes
workflow
week3_proj
repository
week3_proj
branch
master
Undo
Redo
Pull
Push
Branch
Stash
Pop
Boards
Default Profile
Viewing 1/1
Show All
Filter (Ctrl + Alt + f)
LOCAL 1/1
master
REMOTE
TAGS 0/0
SUBMODULES 0
// WIP
2
master
Initial commit
2 file changes on master
Path
Tree
Unstaged Files (2)
Stage all changes
data/Project_Reads_Scores.csv
week3_proj.Rproj

Paste URL into "Pull URL"

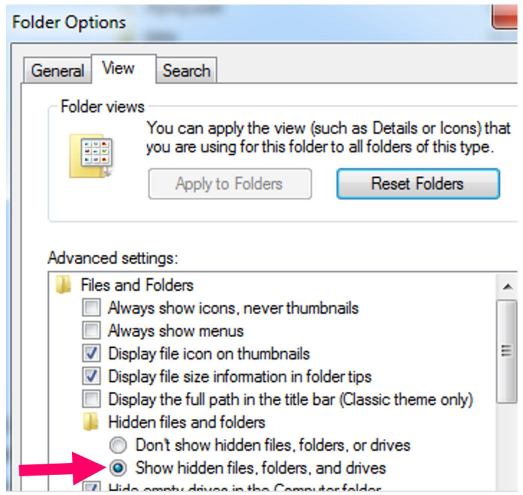
- Same Name used thus far
- Click 🌐 "*URL*"
- Paste URL

Let's see what we've done

Navigate to the folder

- Where did you save your *local*?

If you don't see the `.git` folder



You can apply the view (such as Details or Icons) that you are using for this folder to all folders of this type.

Files and Folders

- [ ] Always show icons, never thumbnails
- [ ] Always show menus
- [x] Display file icon on thumbnails
- [x] Display file size information in folder tips
- [ ] Display the full path in the title bar (Classic theme only)

Hidden files and folders

- ○ Don't show hidden files, folders, or drives
- [✓ Show hidden files, folders, and drives]

Tracking

- As long as the `.git` folder exists, the entire folder will be tracked
- **Do not** create repos inside of repos
- If you want a folder to show up, there needs to be something in it (e.g., a README.md)

Ignoring files

- When we created the .Rproj the repo, we started it with a `.gitignore` file
- The `.gitignore` file tells the repo not to track certain files
    - e.g., proprietary data
- Not that important for now, but if you need to ignore a file in your repo for whatever reason, just put the name of the file(s) at the top of the `.gitignore` file
- Or, create a folder in your .Rproj (e..g, `nopublish`) and put all your documents to be ignored in that file
    - then on the first line of your `.gitignore` file, type:

      *nopublish/**

Homework 6

Submit a link to the repo you created

- Upload the link to Canvas
- See me if you were unable to complete in class (for technical reasons etc.)

It is ok if you leave today a ~~little~~ lot confused!

Terminology

- **Stage**
  - A file that is being prepared for a commit
- **Commit**
  - Changes that have been made to the file(s)
- **Push**
  - After you commit the changes you want, push them from the local to the remote

Half the battle is learning these terms

[demo]

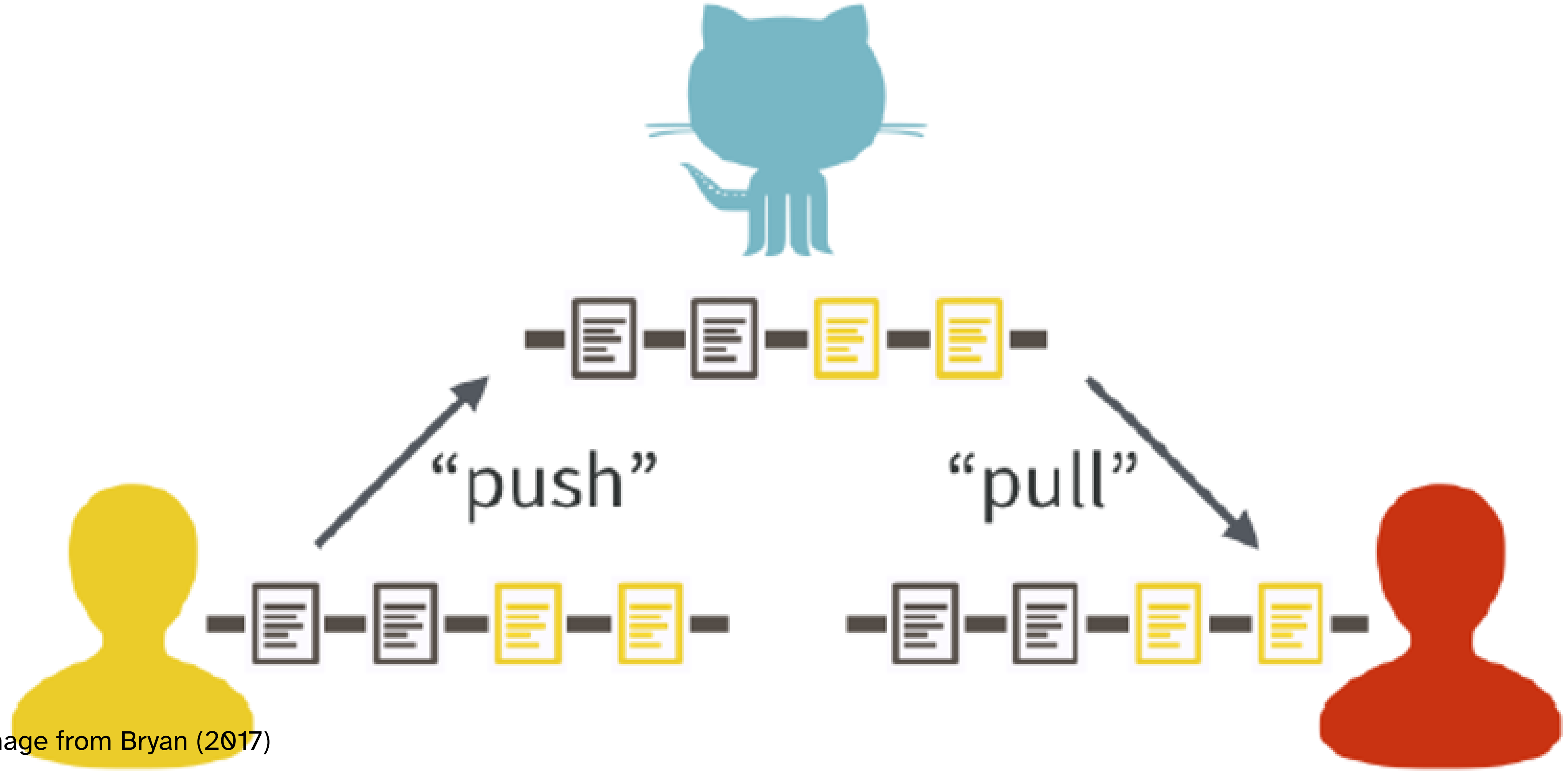
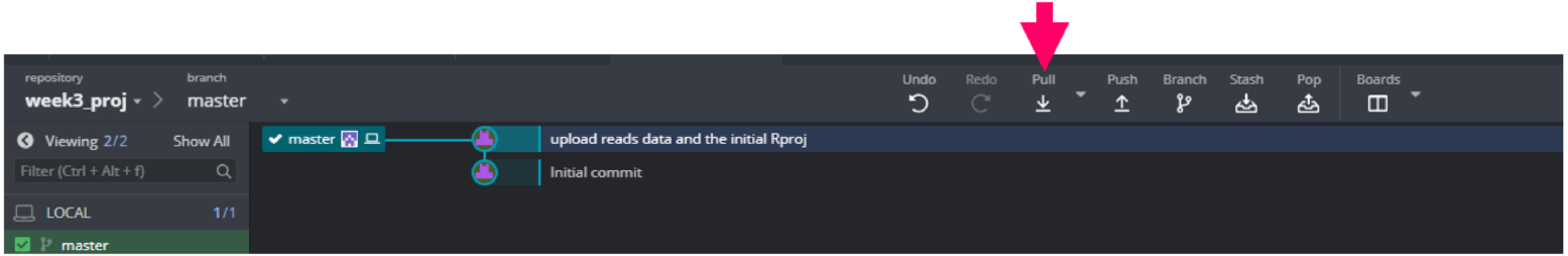


Image from Bryan (2017)

Pull



upload reads data and the initial Rproj

Initial commit

- **Pull** before you start any collaborative work!!
- **Pull** any accepted changes from the remote to your local

Pull before you start any collaborative work!!

Collaborating

Two methods

1. Add users to a repo
    - *Settings* > *Collaborators* > *Manage Access* > *Add people* (add user with username)
2. Create an organization

Collaborating

Two methods

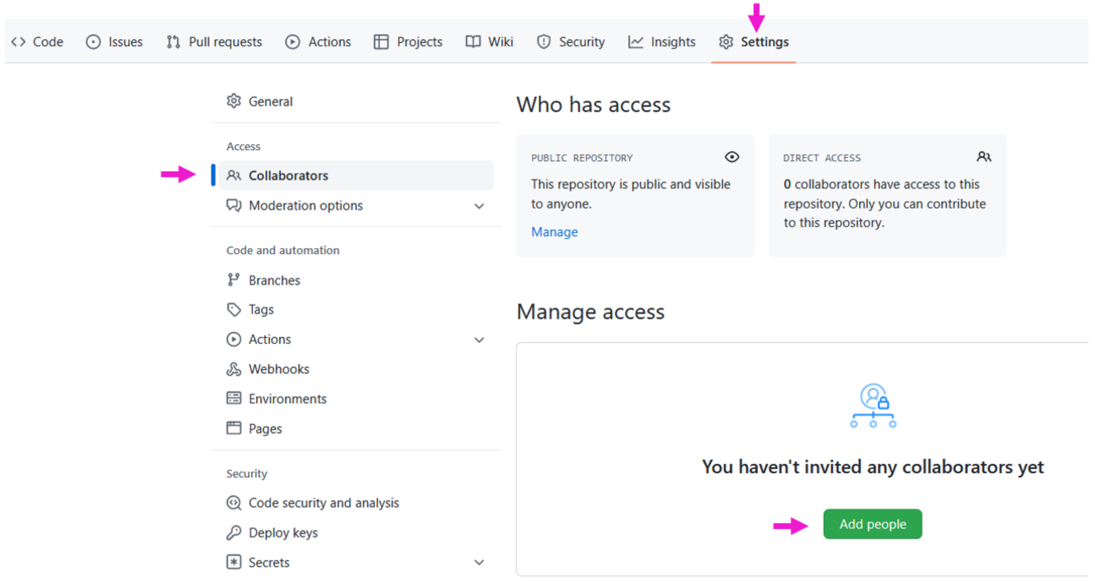
1. Add users to a repo
   - *Settings* > *Collaborators* > *Manage Access* > *Add people* (add user with username)
2. Create an organization

Collaborating

Final Project

Add users to a repo

1. One group member will Open a repo (like we just went over)
   - Potentially the person who has the project data, but not necessarily
2. Within the repo on GitHub
   Settings > *Collaborators* > *Manage Access* > *Add people* (add user with username)



Who has access

PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage

DIRECT ACCESS
0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

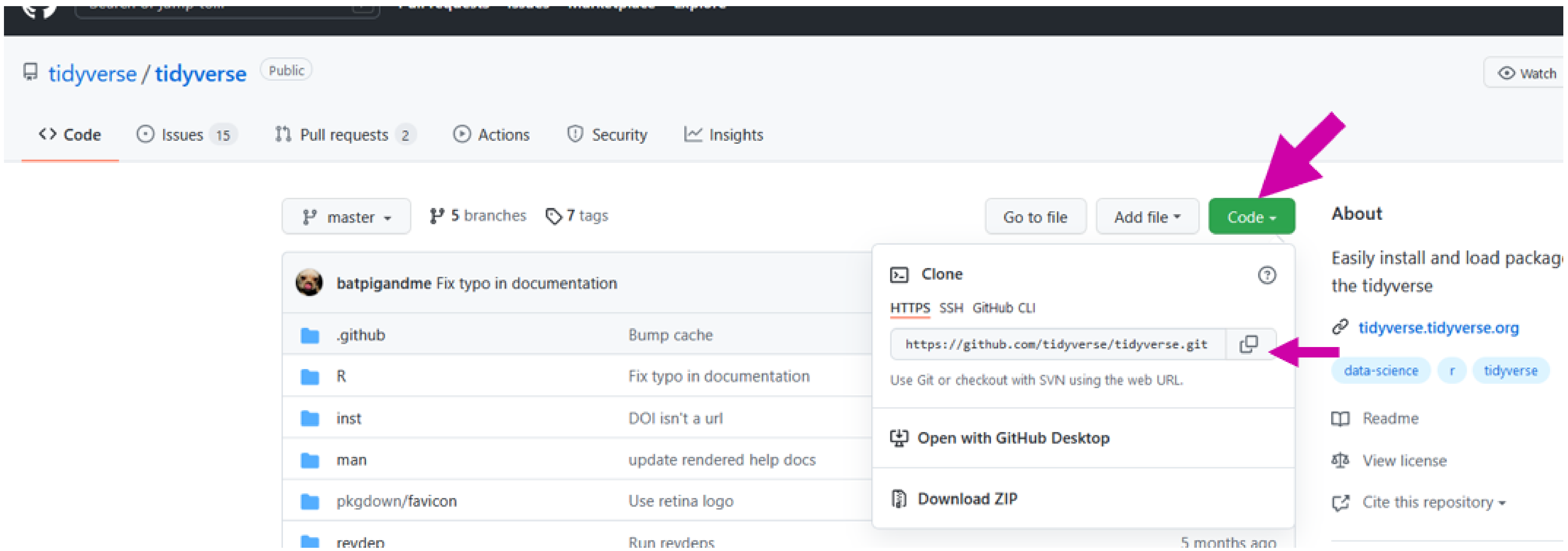
You haven't invited any collaborators yet

Collaborating

- The collaborators will receive an email from GitHub
- Accept the invitation
- *Clone* the repo

Cloning

- *Cloning* means that you are downloading the repo
- You can *clone* any repo you like (like the `{tidyverse}`)
- For the final project, you will *clone* the repo you were invited to collaborate on



batpigandme Fix typo in documentation

.github	Bump cache
R	Fix typo in documentation
inst	DOI isn't a url
man	update rendered help docs
pkgdown/favicon	Use retina logo
revdep	Run revdeps

5 months ago

```
https://github.com/tidyverse/tidyverse.git
```

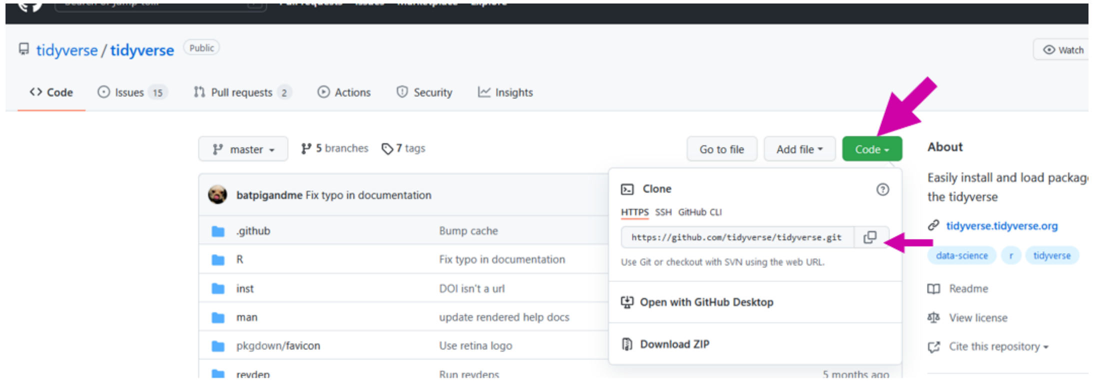
Use Git or checkout with SVN using the web URL.

Clone a repo

(1) Go to the repo on GitHub

(2) Click the green "*Code*" button near the top

(3) Copy the URL



```
https://github.com/tidyverse/tidyverse.git
```

Use Git or checkout with SVN using the web URL.

Easily install and load packag the tidyverse

tidyverse.tidyverse.org

batpigandme Fix typo in documentation

.github	Bump cache	
R	Fix typo in documentation	
inst	DOI isn't a url	
man	update rendered help docs	
pkgdown/favicon	Use retina logo	
revdep	Run revdeps	5 months ago

Clone a repo

(4) Go to GitKraken

(5) Clone > 🌐 Clone with URL

(6) Specify where it will live on your machine ("*Where to clone to*")

(7) Paste in the copied URL

(8) Click "*Clone the repo!*"

Merge Conflict

git commit
git push
meme-arsenal.ru

Merge Conflicts

- Always **pull** before you start working!!
- Try to stay in communication with your collaborators and don't work on the exact same thing at the same time
- Use GitKraken to help manage merge conflicts that do come up

Merge Conflicts

- Remember to always **pull** first!!
- GitKraken has a really nice GUI to resolve merge conflicts

link

Merge Conflicts

Let's give this a shot

Merge Conflict Activity (15 mins)

1. Get a partner
2. Partner1 open in GitHub the repo you made in class
3. Partner1 add Partner2 as a collaborator in GitHub
4. Partner2 clone the repo
5. Partner1 make changes to the *foo.Rmd* file in the repo
    - commit and push those changes
6. Partner2 **do NOT pull**
7. Partner2 make different changes to the same parts of the *foo.Rmd* file
    - Partner2 commit the changes. Try to push. What happens? Why?
8. Use GitKraken to deal with the merge conflicts

15:00

Merge Conflict Activity (15 mins)

(2) Partner1 open in GitHub the repo you made in class

(3) Partner1 add Partner2 as a collaborator in GitHub

(3) Partner2 open email associated with GitHub to accept invitation

(4) Partner2 clone the repo

- **GitHub**
- Copy URL (click clipboard icon)
- **GitKraken**
- Clone with URL
- Paste in URL
- Choose place where repo will live on your machine (local)

15:00

Merge Conflict Activity (15 mins)

(5) Partner1 open RRtudio

- Make changes to the *foo.Rmd* file in the repo
- commit and push those changes

(6) Partner2 **do NOT pull**

(7) Partner2

- make different changes to the same parts of the *foo.Rmd* file
- Partner2 commit the changes to local
- Try to push
- What happens? Why?

(8) Use GitKraken to deal with the merge conflict

15:00

Quick Recap

Basic workflow

- create an RStudio project
- create a repo
- add collaborators
- *pull*
- *stage* and *commit* changes
- *push* changes to *remote*

Revisiting git terminology

Talk with a neighbor about what each of these terms mean

- clone
- pull
- stage
- commit
- push
- repo
- remote

Next time

Before next class

- Homework
  - **Homework 6** (hopefully completed in class today)
  - **Homework 7**
- Supplemental Learning
  - RStudio Primer: Join Data Sets
  - Codecademy: Joining Tables in R

Homework 7