

**Workshop - Exploring and Analyzing Satellite and Gridded Data on Climate  
and Population with R and Git Code Tracking**

African Population Conference, May 2024, Lilongwe

# **Introduction to Github with RStudio**

Ariane Sessego & Arlette Simo Fotso

# What you will learn in this session

- **1. Git and Github for beginners**
  - What are Git and Github ?
  - Why use them ?
  - How do they work ?
- **2. Using Github with RStudio – the basics**
  - How to link you RStudio with your Github account
  - Use the main Git and Github commands inside your RStudio

# **1. Git and Github for beginners**

# Git and Github

- **Git is a version control software** created in 2005 by Linus Torvalds
  - It is used to easily deal with different versions of text file (source code, latex files, txt documents, etc.)
  - That way you can test out different things and come back to a previous version, or collaborate with other people who are doing multiple changes at once
- **Github is a website that uses Git**, and acts as a central *hub* for your Git project
  - There are other similar services, such as Gitlab
- **Together, Git and Github are like a super sophisticated google docs or dropbox**



# Why use Git and Github?

- Git and Github has become the standard version control software in the world

It allows you to:

- **Keep track of the different versions of your work**
- **To collaborate with others** (especially on code)
- **For open science: to share your code and access code shared by others.**  
If you want to publish your code and have it be reproduced, or contribute to code that someone else has published
- This is why we are going to introduce it to you today

# How does Git work?

- A **Git repository** (or repo) is equivalent to your project folder



# How does Git work?

- A **Git repository** (or repo) is equivalent to your project folder
- You will then be able to modify your **files** in this repository by creating successive ***commits***



# How does Git work?

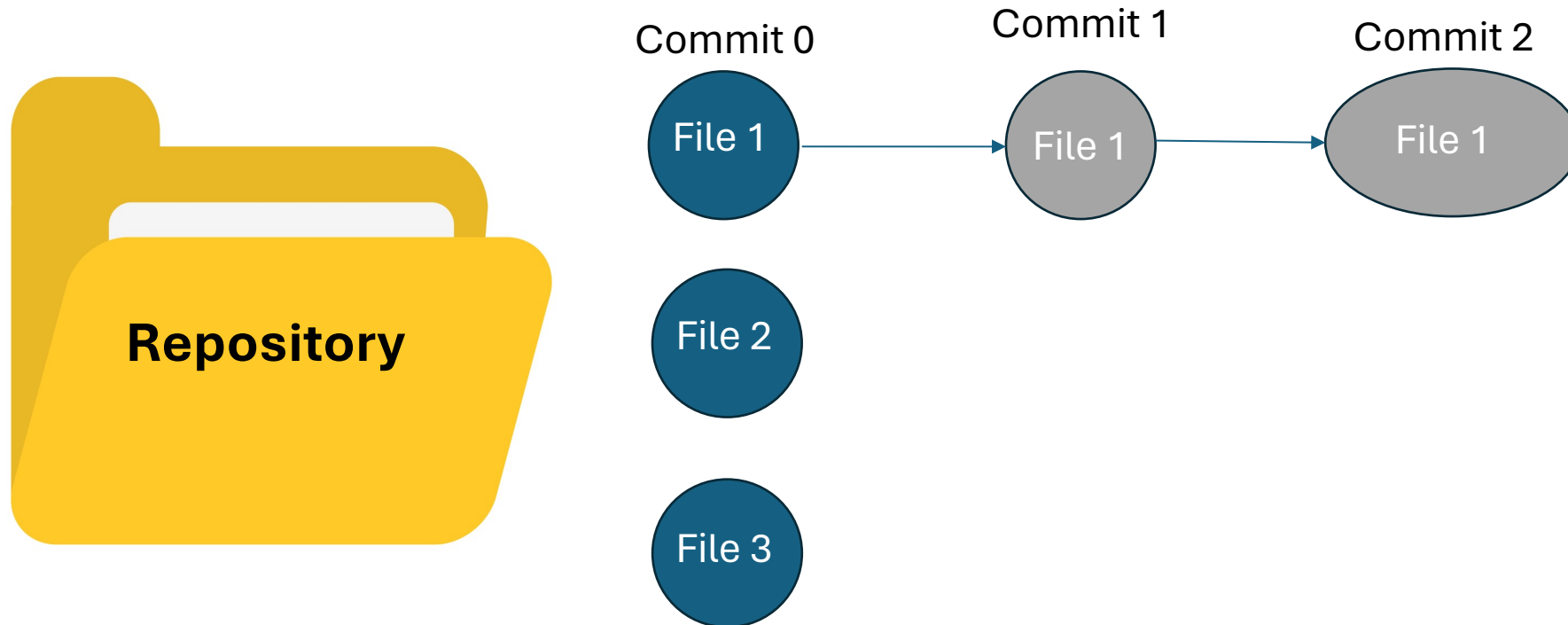
- A **Git repository** (or repo) is equivalent to your project folder
- You will then be able to modify your **files** in this repository by creating successive ***commits***





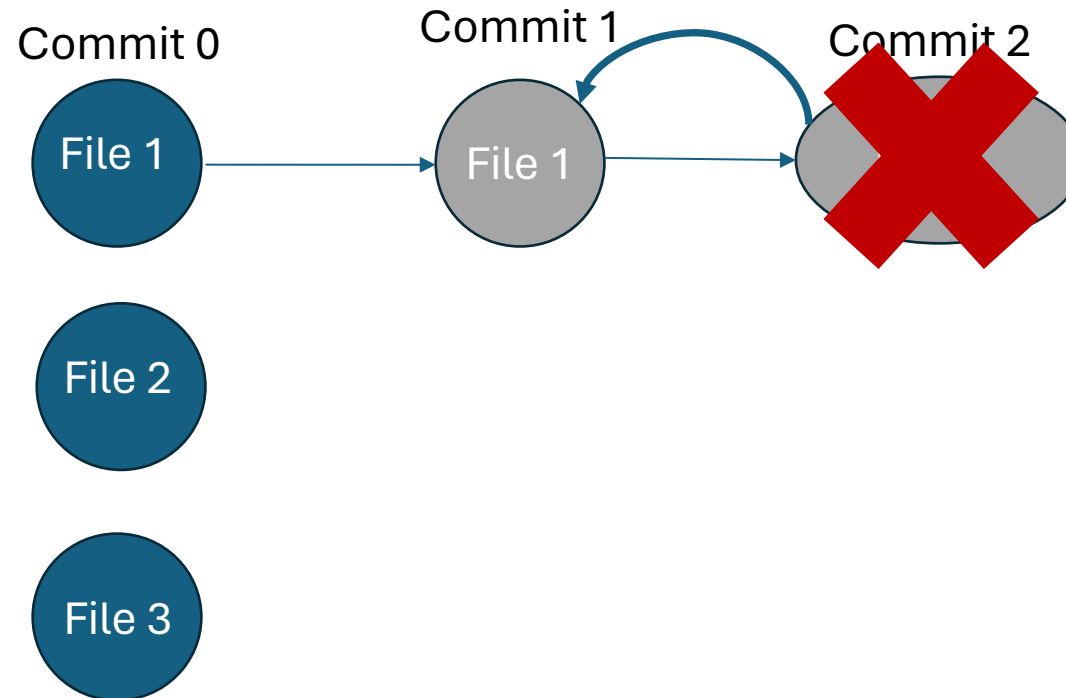
# How does Git work?

- A **Git repository** (or repo) is equivalent to your project folder
- You will then be able to modify your **files** in this repository by creating successive ***commits***



# How does Git work?

- A **Git repository** (or repo) is equivalent to your project folder
- You will then be able to modify your **files** in this repository by creating successive **commits**
- **Git keeps tracks of the history of your commits so you can revert to a previous version**



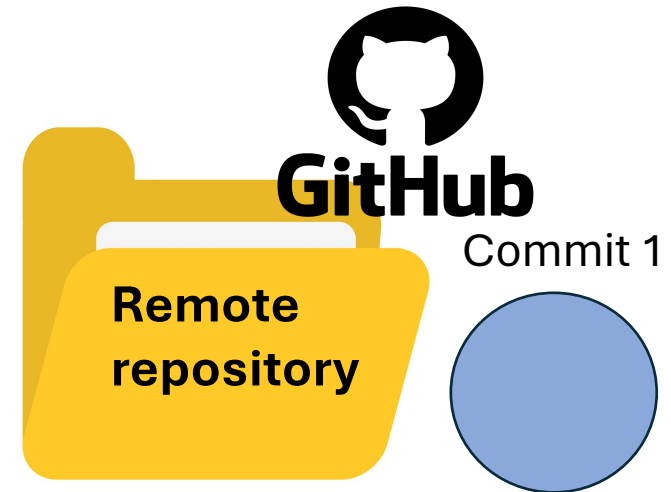
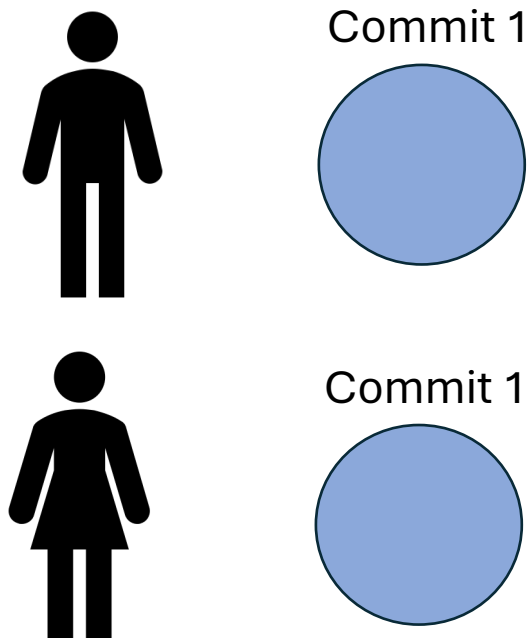
# How does Github work?

- Github acts as the central repository storing all the files and their commit history (***remote repository*** or ***origin***)



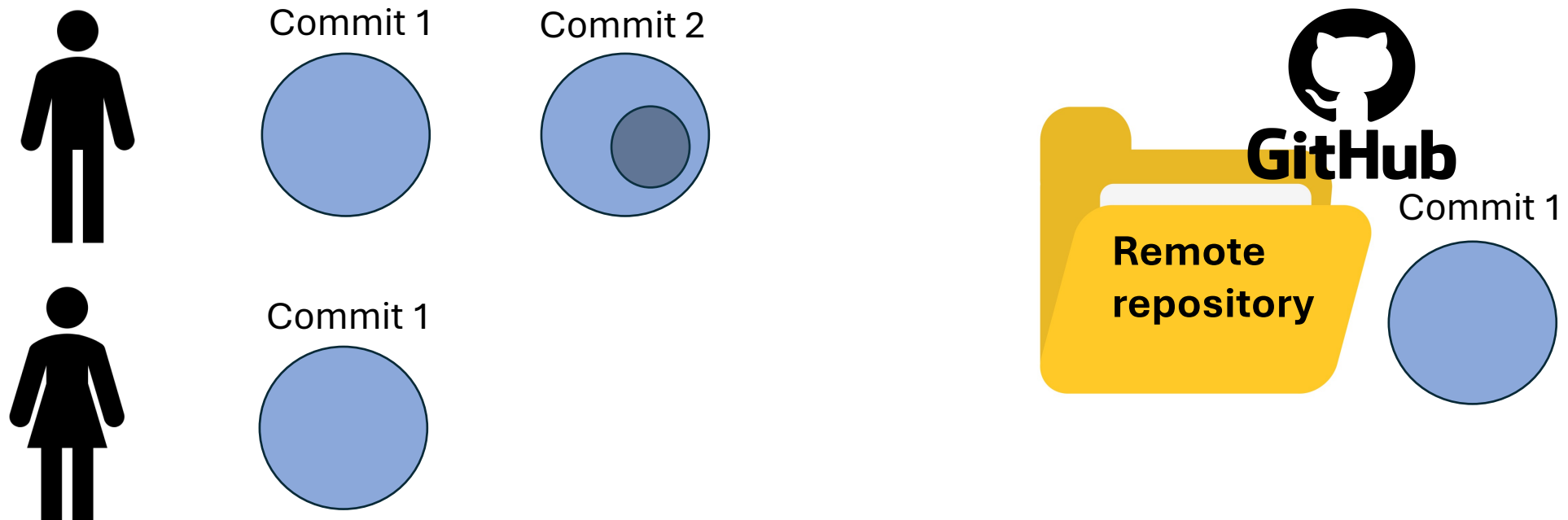
# How does Github work?

- Github acts as the central repository storing all the files and their commit history (***remote repository*** or ***origin***)
- You each have a copy of this remote repository locally, and must synchronise with it, using *pushes* and *pulls*



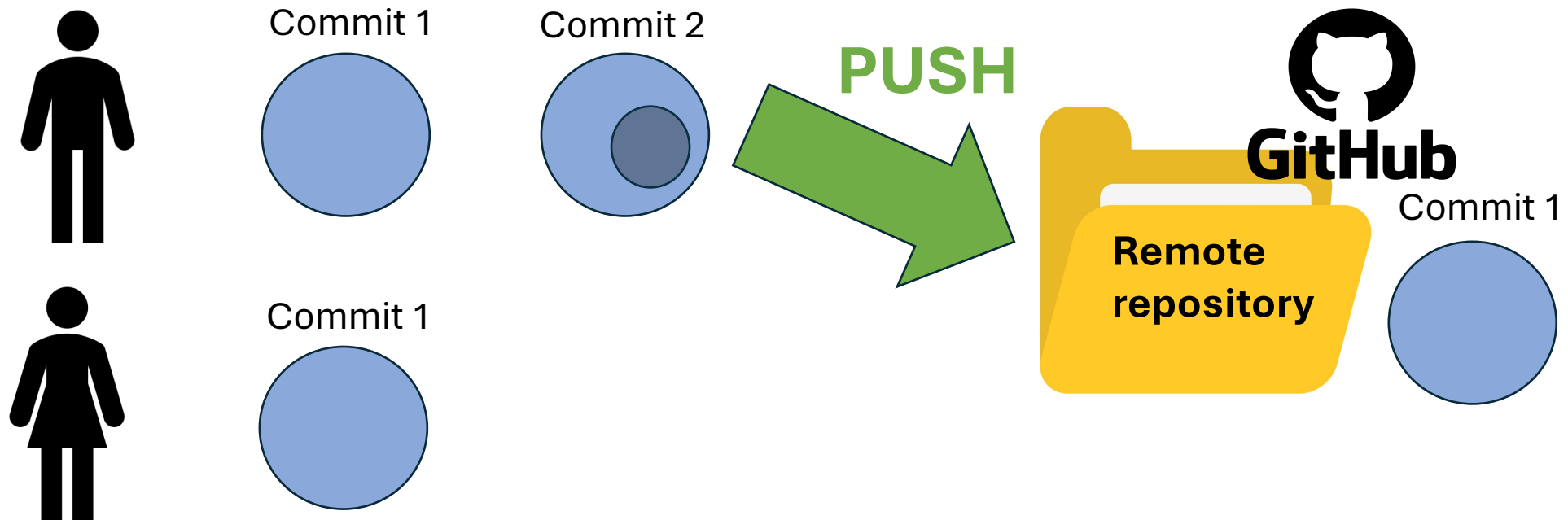
# How does Github work?

- Github acts as the central repository storing all the files and their commit history (***remote repository*** or ***origin***)
- You each have a copy of this remote repository locally, and must synchronise with it, using *pushes* and *pulls*



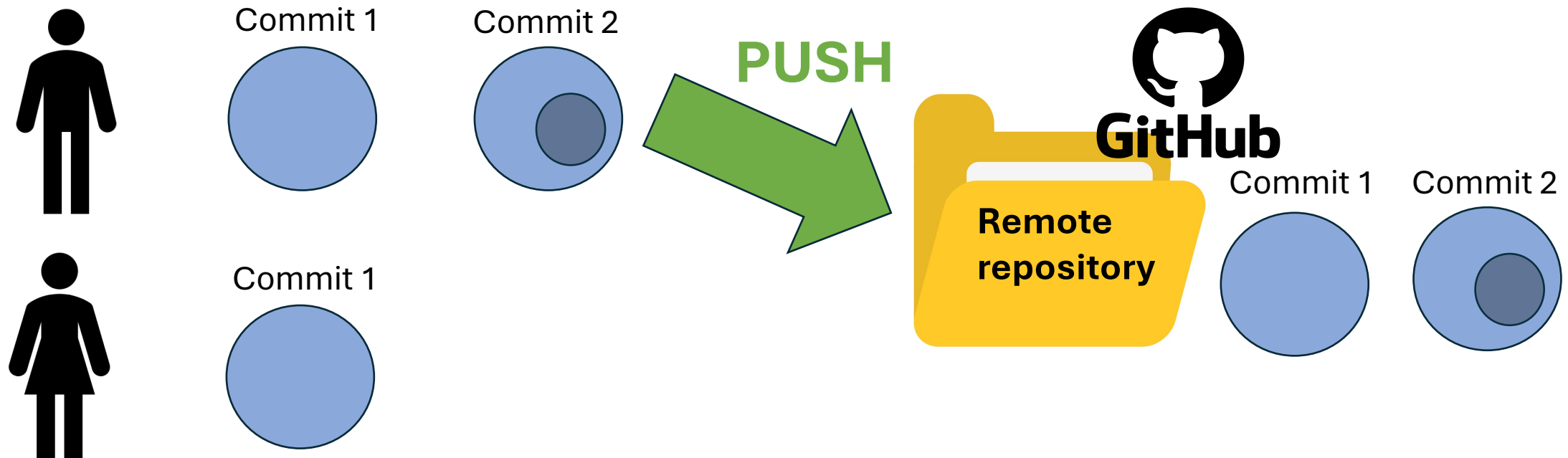
# How does Github work?

- Github acts as the central repository storing all the files and their commit history (***remote repository*** or ***origin***)
- You each have a copy of this remote repository locally, and must synchronise with it, using ***pushes*** and ***pulls***



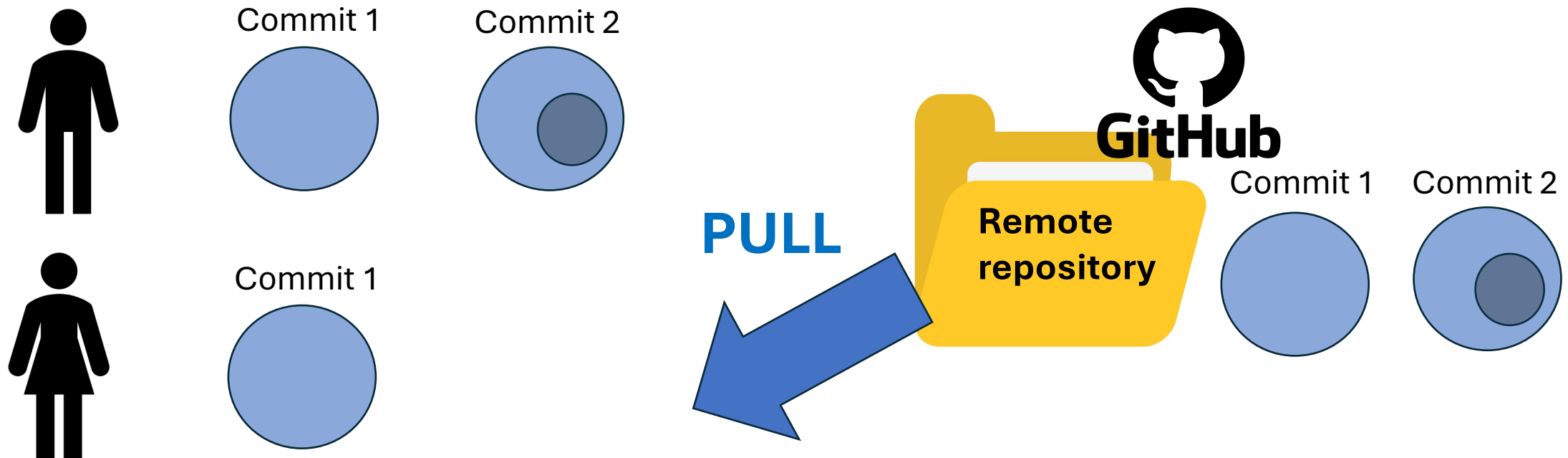
# How does Github work?

- Github acts as the central repository storing all the files and their commit history (***remote repository*** or ***origin***)
- You each have a copy of this remote repository locally, and must synchronise with it, using ***pushes*** and ***pulls***



# How does Github work?

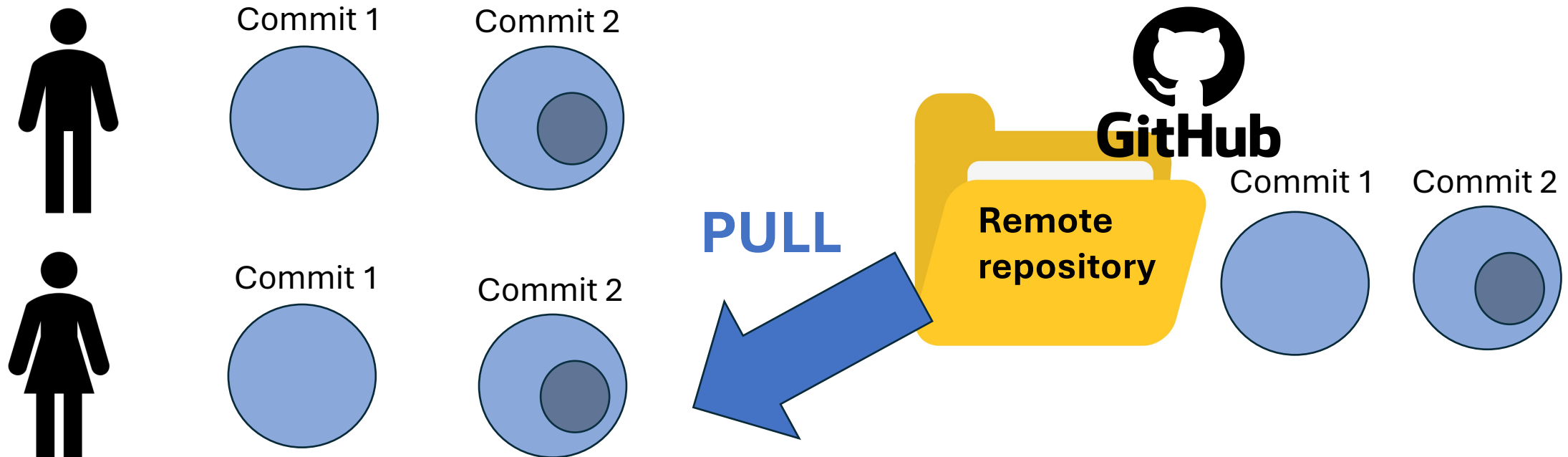
- Github acts as the central repository storing all the files and their commit history (***remote repository*** or ***origin***)
- You each have a copy of this remote repository locally, and must synchronise with it, using ***pushes*** and ***pulls***





# How does Github work?

- Github acts as the central repository storing all the files and their commit history (***remote repository*** or ***origin***)
- You each have a copy of this remote repository locally, and must synchronise with it, using ***pushes*** and ***pulls***

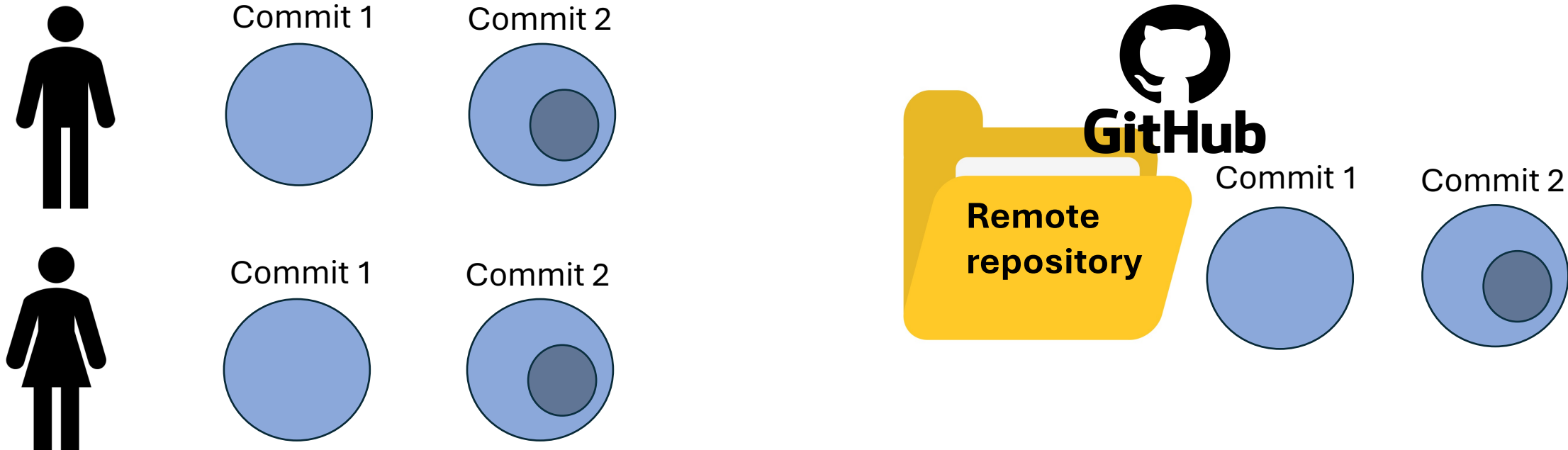


# How does Github work?

- **Push** – updates the remote repository with new local commits
- **Pull** – updates the local repository with new remote commits

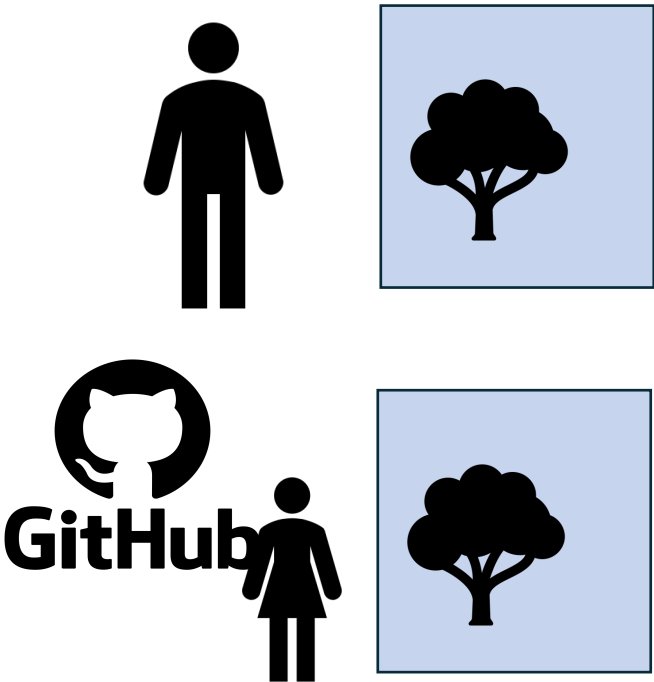
Good practice:

- **Make sure you always pull before you push!**



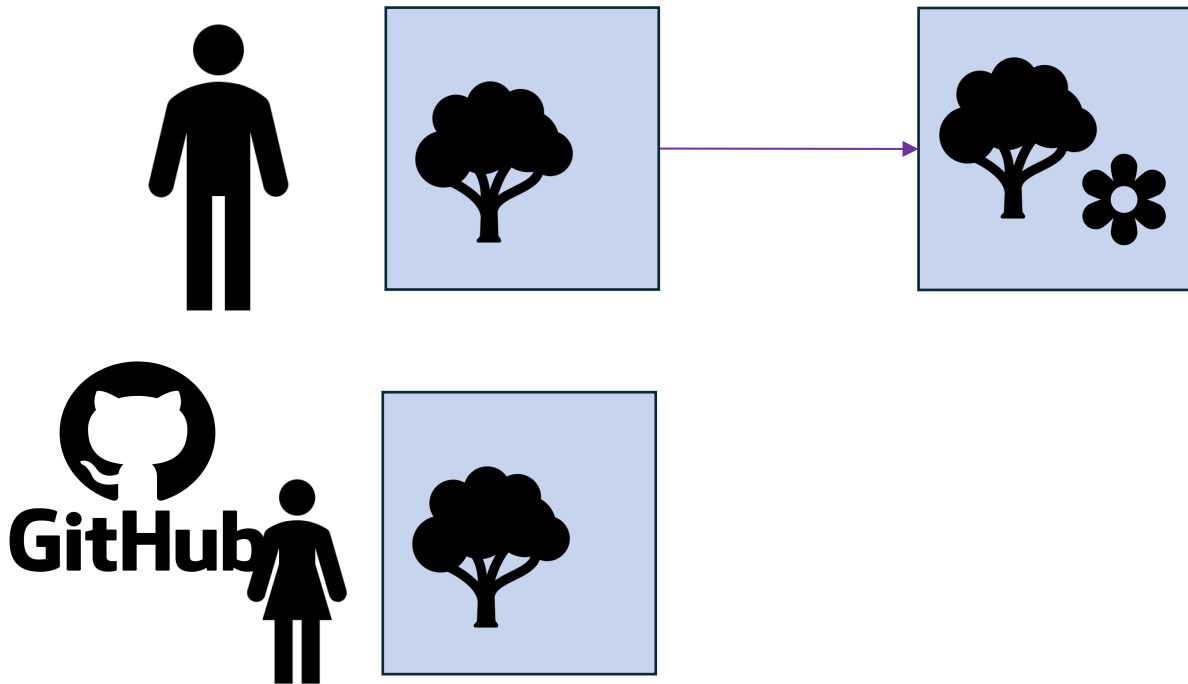
# Merge conflicts and how to resolve them

- If you modify the same lines in the same file, you can have a **merge conflict**



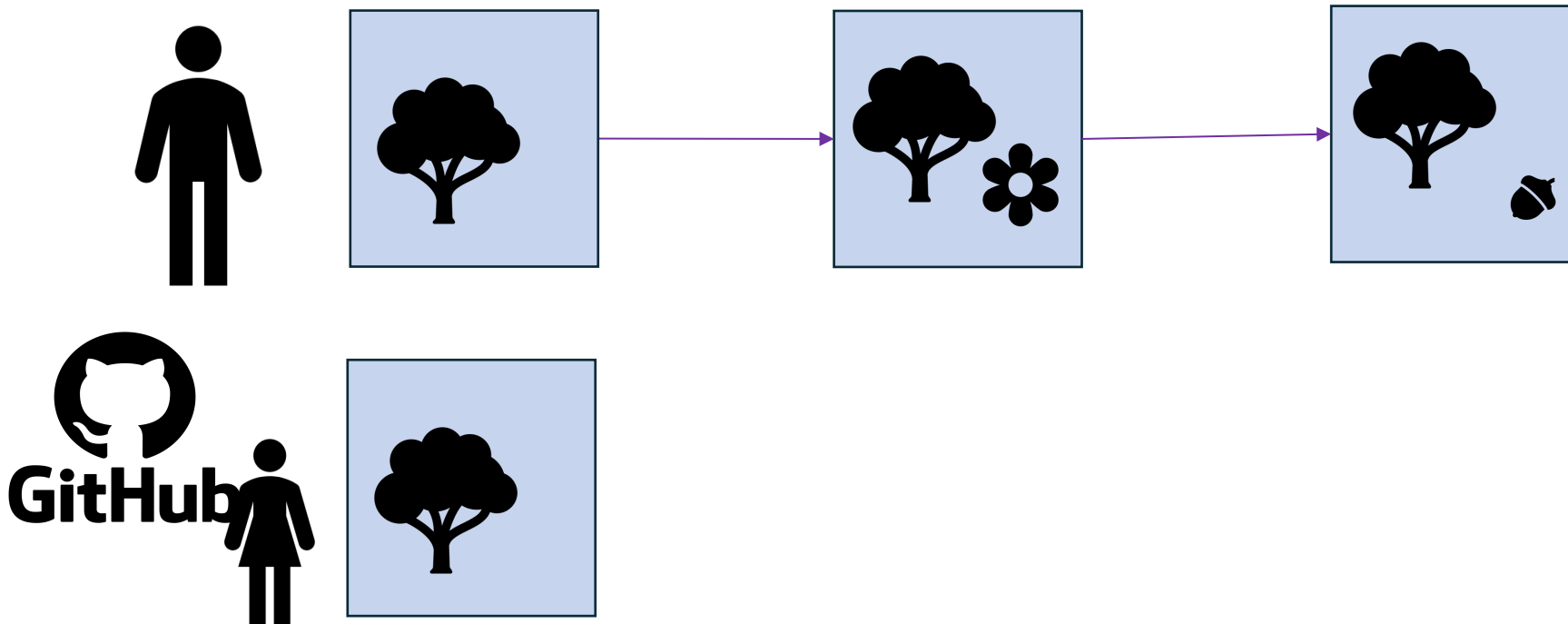
# Merge conflicts and how to resolve them

- If you modify the same lines in the same file, you can have a **merge conflict**



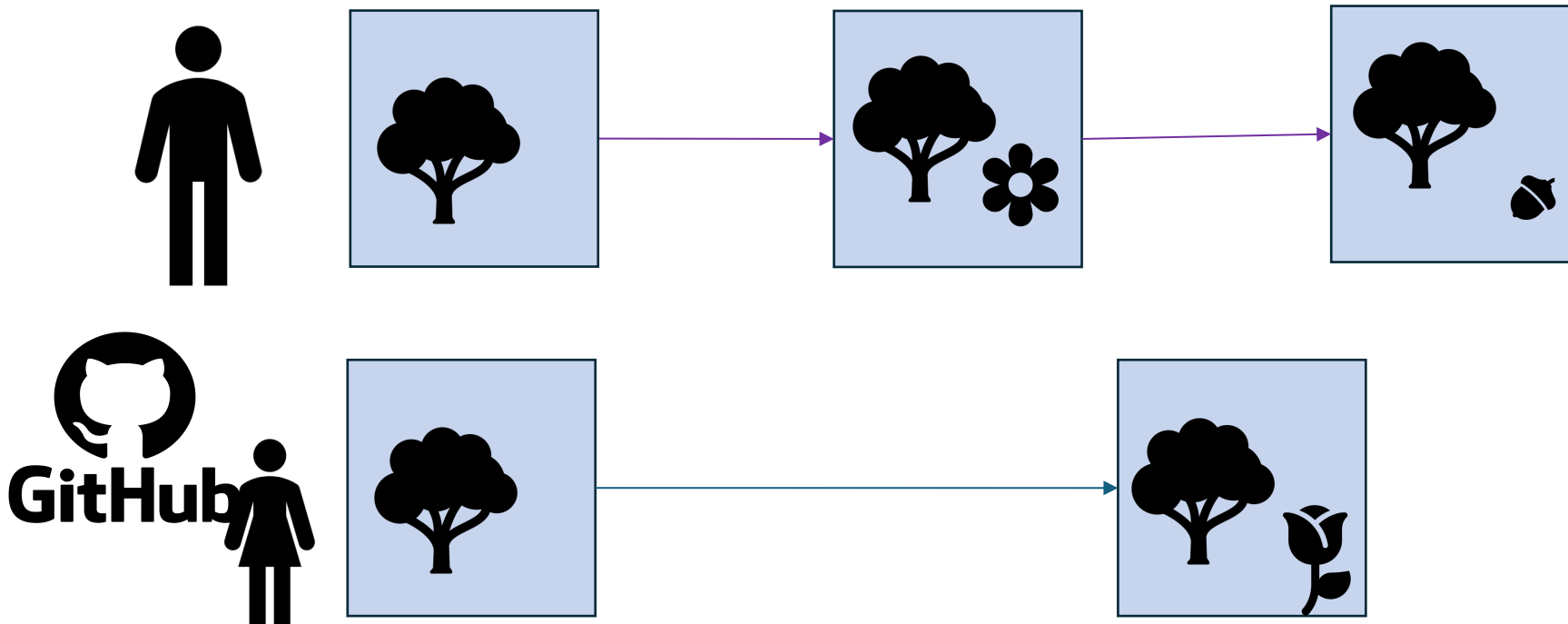
# Merge conflicts and how to resolve them

- If you modify the same lines in the same file, you can have a **merge conflict**



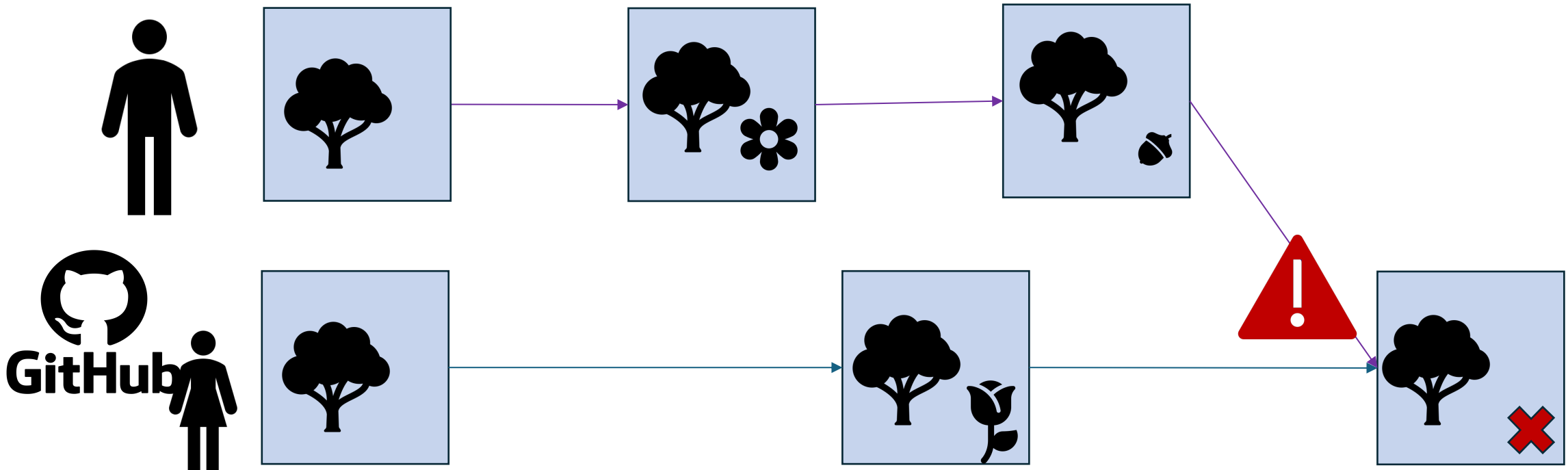
# Merge conflicts and how to resolve them

- If you modify the same lines in the same file, you can have a **merge conflict**



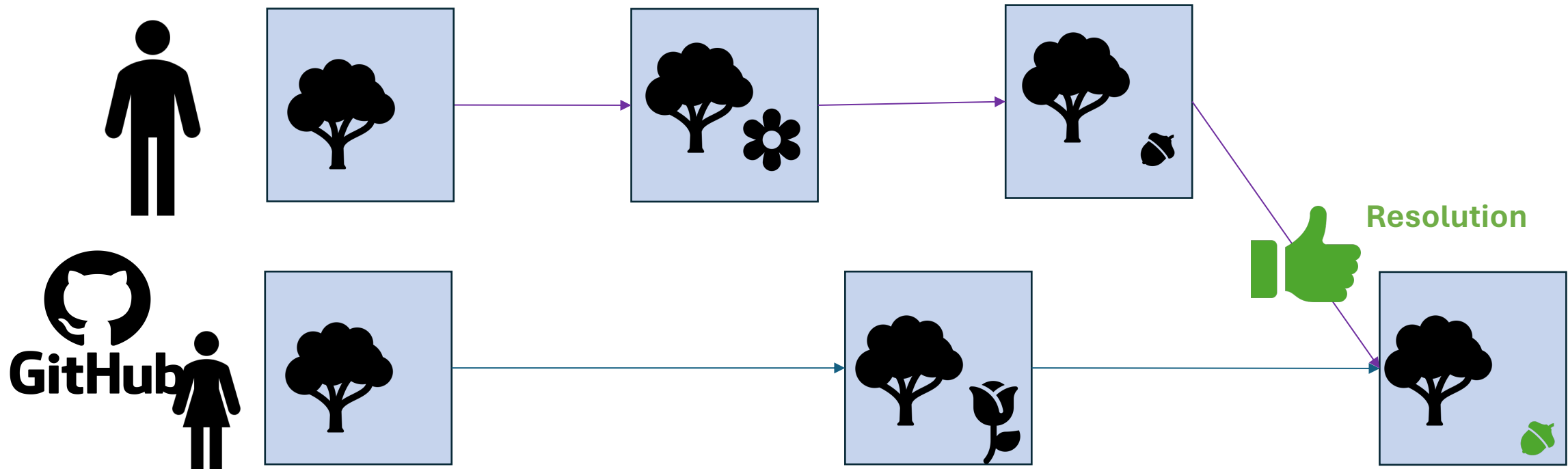
# Merge conflicts and how to resolve them

- If you modify the same lines in the same file, you can have a **merge conflict**



# Merge conflicts and how to resolve them

- If you modify the same lines in the same file, you can have a **merge conflict**
- You will have to **resolve the merge conflict** by choosing one by one the version you want to keep





# Merge conflicts and how to resolve them

- To avoid merge conflicts, it is important to communicate with your collaborators on the parts you are working on
- If you often work on the same document at the same time, using **branches** can help limiting merge conflicts.
  - We will not cover this here, but if you are interested here are additional material: <https://happygitwithr.com/git-branches>

## **2. Using Github with RStudio**

# How to use Git and Github with Rstudio

The general steps to follow:

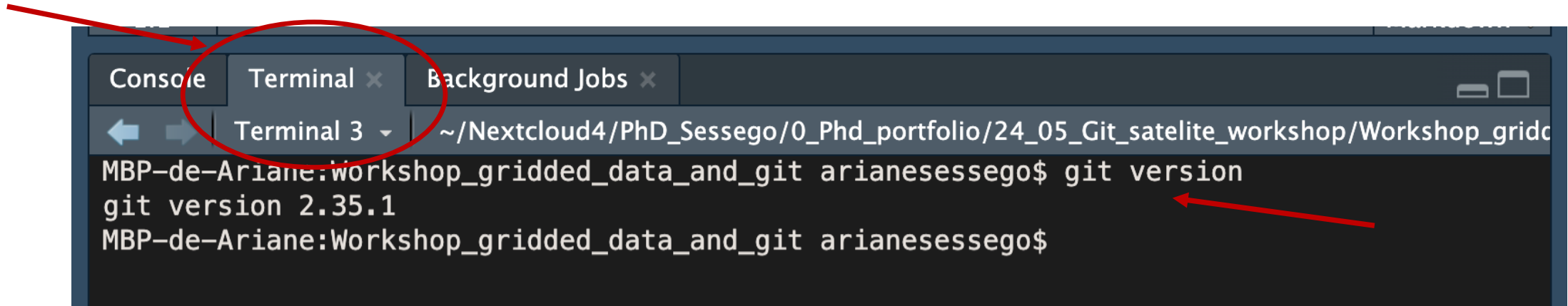
- 0. Link Rstudio with your Github account** (you will only have to do it once)
- 1. Create a repository on your Github**
- 2. Clone it on your computer through Rstudio**
- 3. Add documents, code, etc... and start working on them!**

I will take you through these steps here!

NB: You could also change the order of these steps, but this is the easiest way to do it.

# 0. Link Rstudio with your Github account

- First, let's make sure you installed Git correctly on your computer
  - Write “git version” in your RStudio Terminal



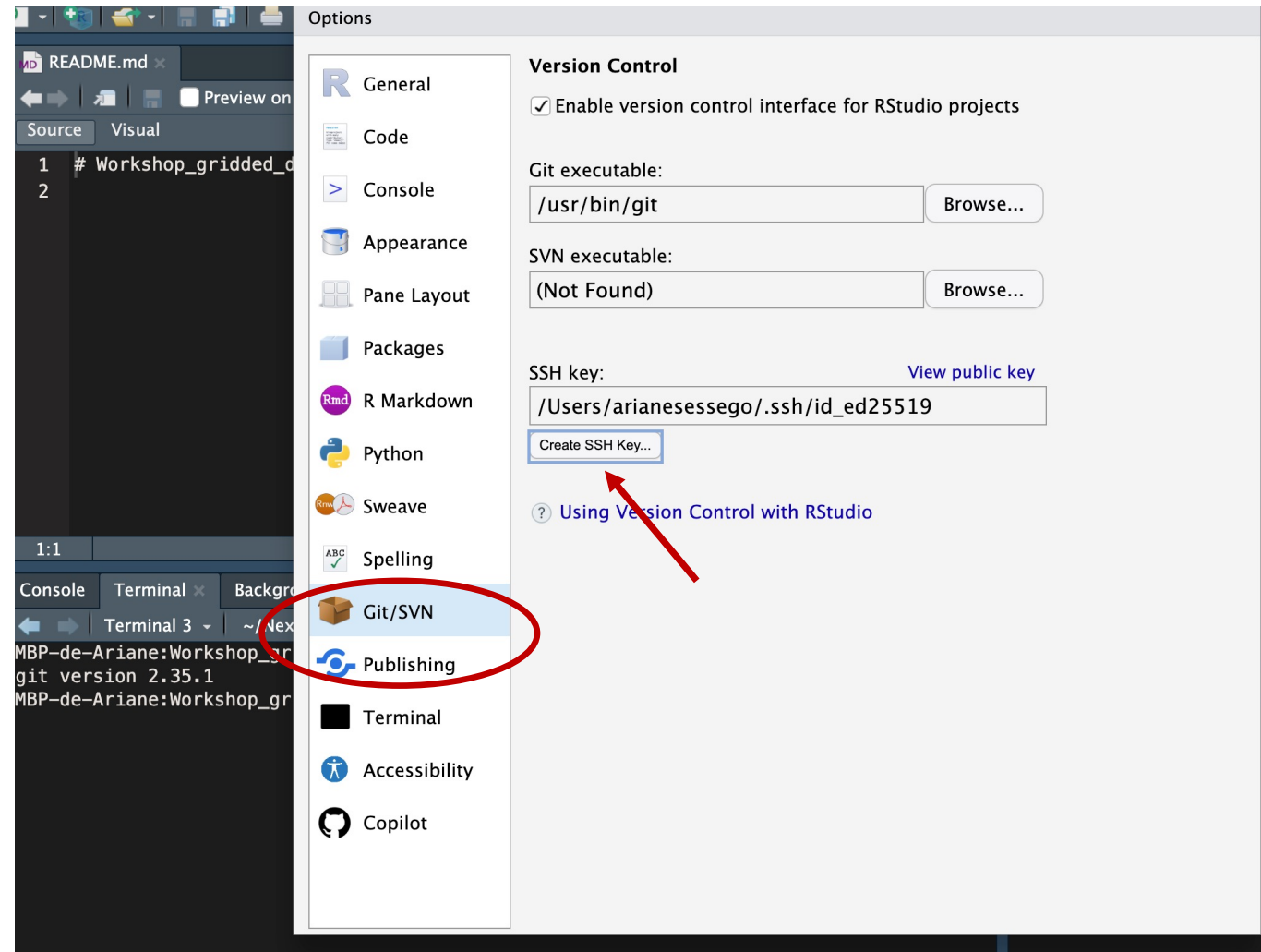
The screenshot shows the RStudio interface with the 'Terminal' pane active. The terminal title bar is 'Terminal 3' and the path is '~/Nextcloud4/PhD\_Sessego/0\_PhD\_portfolio/24\_05\_Git\_satelite\_workshop/Workshop\_gridded\_data\_and\_git'. The command prompt shows 'MBP-de-Ariane:Workshop\_gridded\_data\_and\_git arianesessego\$ git version' and the output is 'git version 2.35.1'. A red circle highlights the 'Terminal' tab, and a red arrow points to the 'git version' command.

```
MBP-de-Ariane:Workshop_gridded_data_and_git arianesessego$ git version
git version 2.35.1
MBP-de-Ariane:Workshop_gridded_data_and_git arianesessego$
```

- If your result looks like this you're all set!

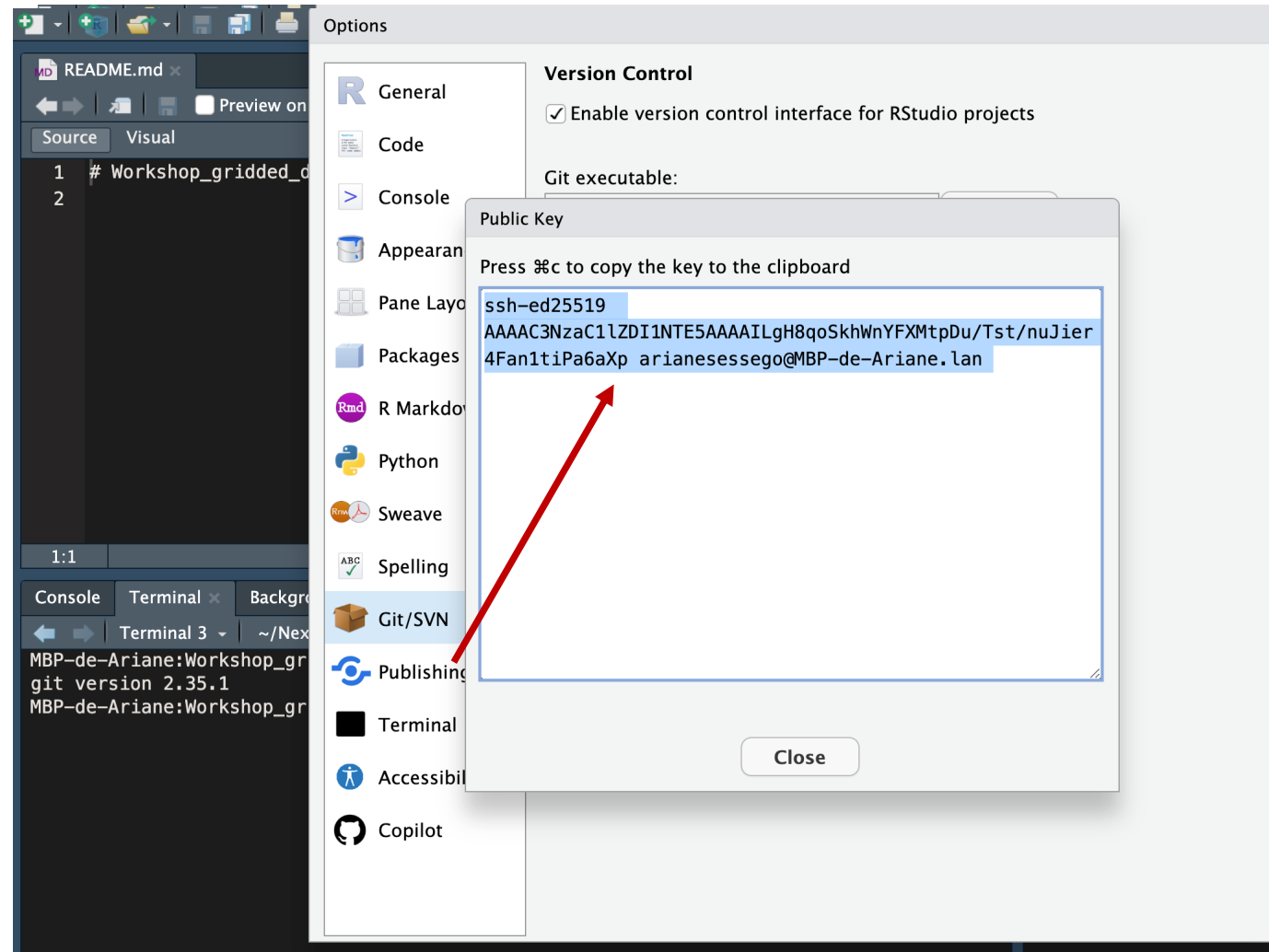
# 0. Link Rstudio with your Github account

- You first need to connect your Github account with Rstudio
  - Rstudio needs a secure way to log in your Github account
- To do so, we are going to create an SSH (Secure Shell) key
  - In RStudio go in Tools > Global Options > Git/SVN
  - Click on "Create an SSH key" > "Create" (the password is not required)



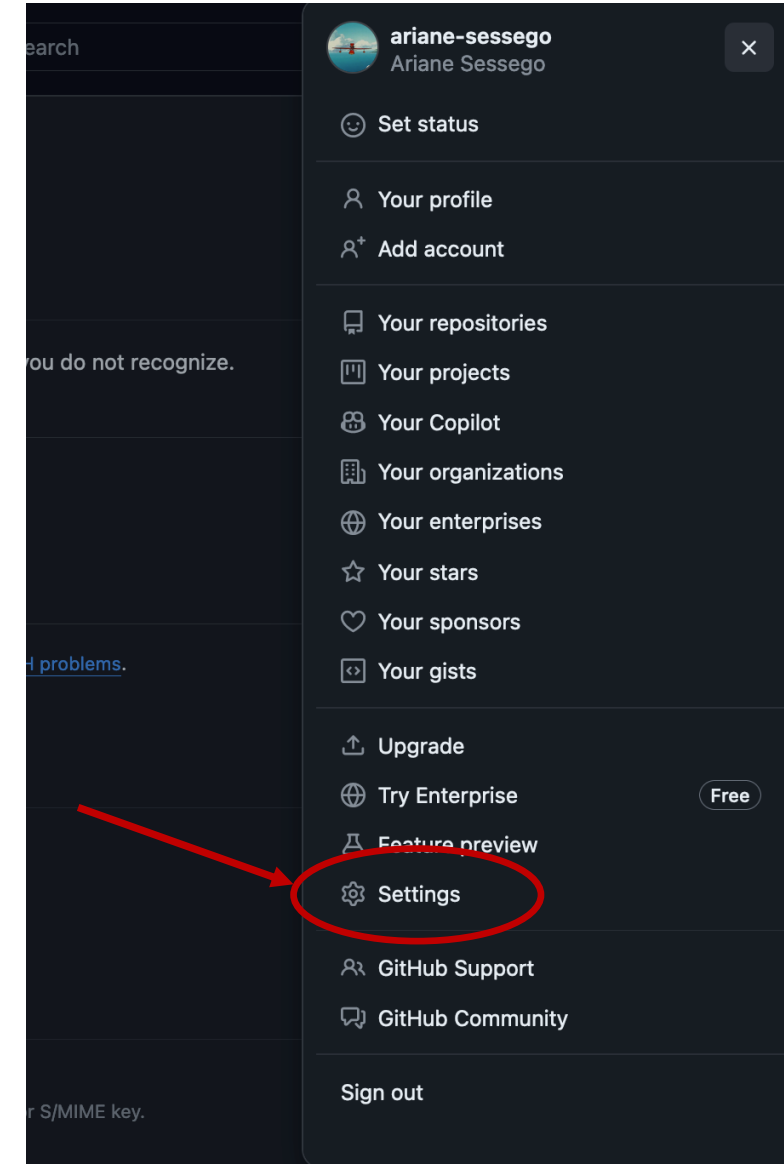
# 0. Link Rstudio with your Github account

- You first need to connect your Github account with Rstudio
  - Rstudio needs a secure way to log in your Github account
- To do so, we are going to create an SSH (Secure Shell) key
  - In RStudio go in Tools > Global Options > Git/SVN
  - Click on "Create an SSH key" > "Create" (the password is not required)
  - It shows you your private password – close the window
  - Click on view public key and copy (CTRL+V)



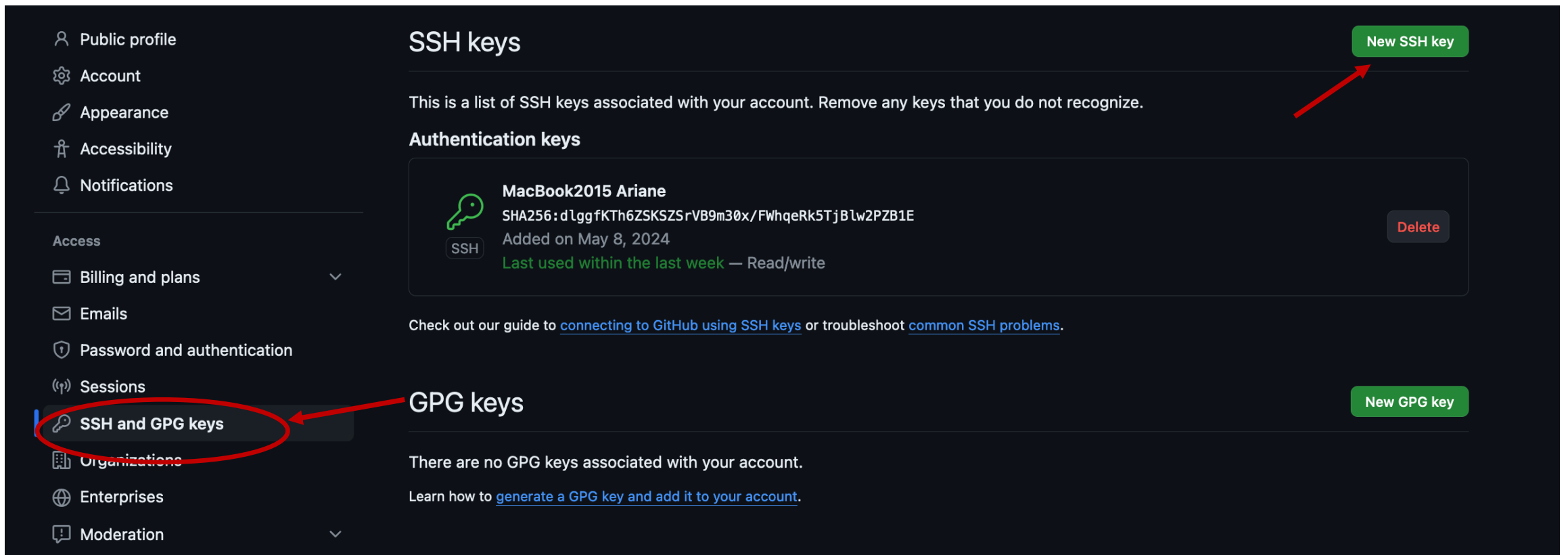
# 0. Link Rstudio with your Github account

- On your Github (<https://github.com/>), go in Settings



# 0. Link Rstudio with your Github account

- On your Github (<https://github.com/>), go in Settings
  - Go in SSH and GPG keys and click on "NEW SSH key"



The screenshot displays the GitHub Settings interface. On the left sidebar, the 'SSH and GPG keys' option is highlighted with a red circle and a red arrow pointing to it. The main content area is divided into two sections: 'SSH keys' and 'GPG keys'. The 'SSH keys' section has a 'New SSH key' button in the top right corner, indicated by a red arrow. Below this, it lists an authentication key for 'MacBook2015 Ariane' with a 'Delete' button. The 'GPG keys' section has a 'New GPG key' button in the top right corner. Both sections include links to guides for connecting to GitHub using SSH or GPG keys.

**SSH keys**

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

**Authentication keys**

Key Name	SHA256	Added	Last used	Permissions	Action
MacBook2015 Ariane	d1ggfKTh6ZSKSZsrVB9m30x/FWhqeRk5TjB1w2PZB1E	May 8, 2024	within the last week	Read/write	Delete

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

**GPG keys**

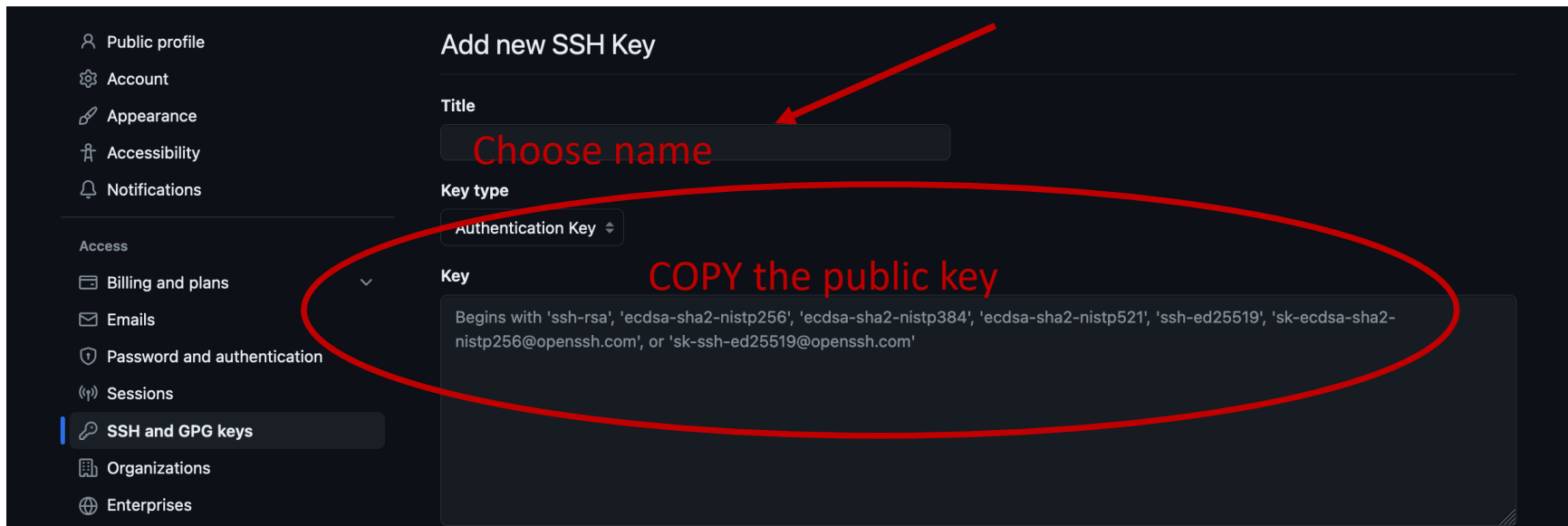
There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).




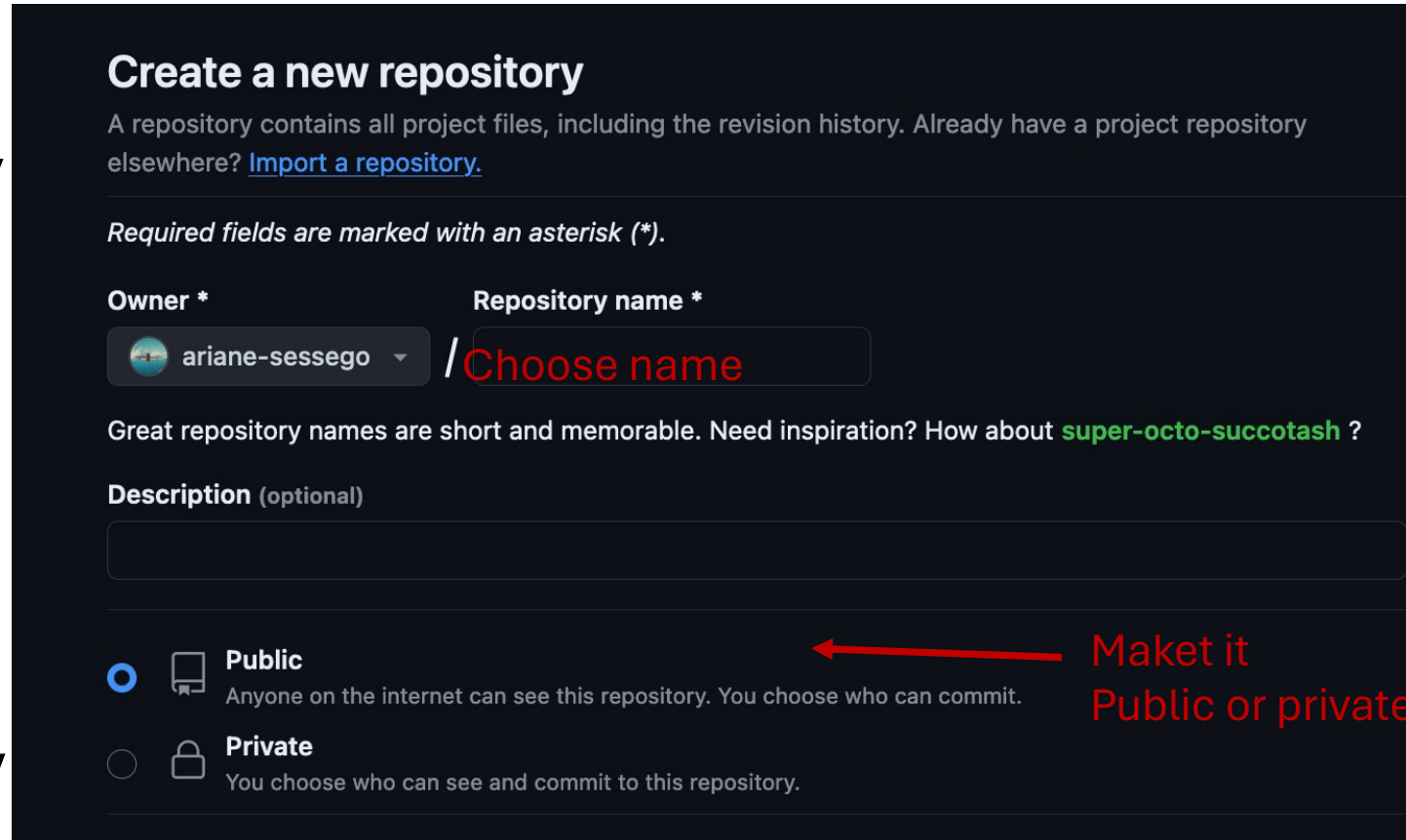
# 0. Link Rstudio with your Github account

- On your Github, go in Settings
  - Go in SSH and GPG keys and click on "NEW SSH key"
  - Give it a name and paste the public key you copied on your Rstudio, then "Add SSH key"
  - That's it! (Normally) you will not have to repeat this step for all your future projects on this computer.



# 1. Create a repository on your Github

- Sign into your Github (<https://github.com/>)
- You can create a new repository by clicking on the icon  on the top left corner
  - You will have to name your repository and be able to choose to make it public or private (parameter that you can also modify later on)
  - You can also add collaborators by going on your repo Settings



**Create a new repository**


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


*Required fields are marked with an asterisk (\*).*

**Owner \*** ariane-sessego / **Repository name \*** Choose name

Great repository names are short and memorable. Need inspiration? How about [super-octo-succotash](#) ?


**Description (optional)**

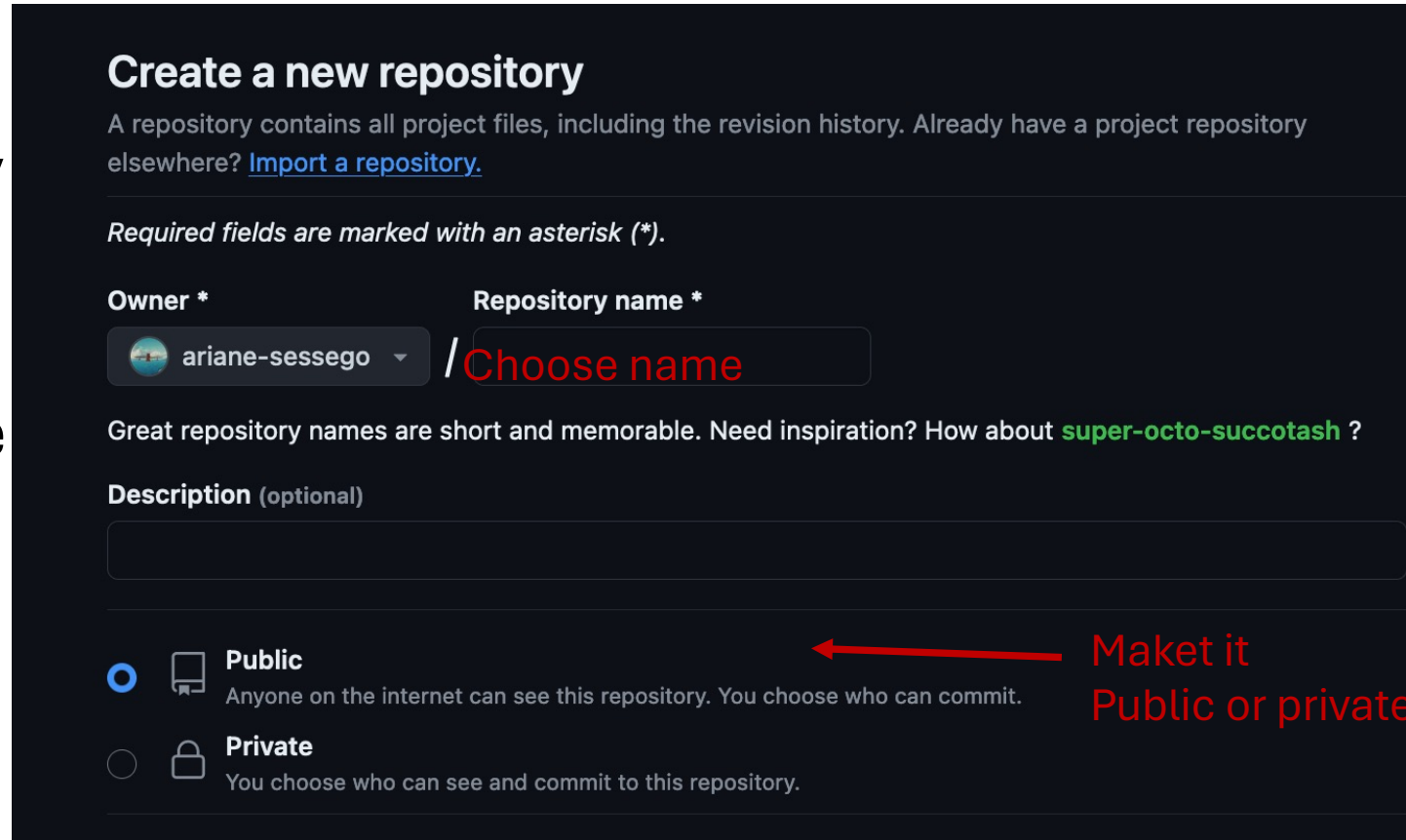
☒ **Public**  Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  You choose who can see and commit to this repository.

**Create repository**

# 1. Create a repository on your Github

- Sign into your Github (<https://github.com/>)
- You can create a new repository by clicking on the icon  on the top left corner
- Today we are not going to create a completely new repository, but **you are going to copy (or fork)** on your account the repository we created for this workshop
  - You will then follow the exact same step as if it was a repo you created




**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Required fields are marked with an asterisk (\*).


Owner \* Repository name \*

 ariane-sessego / **Choose name**

Great repository names are short and memorable. Need inspiration? How about **super-octo-succotash** ?

Description (optional)

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

**Make it Public or private**

Create repository

## Side note:

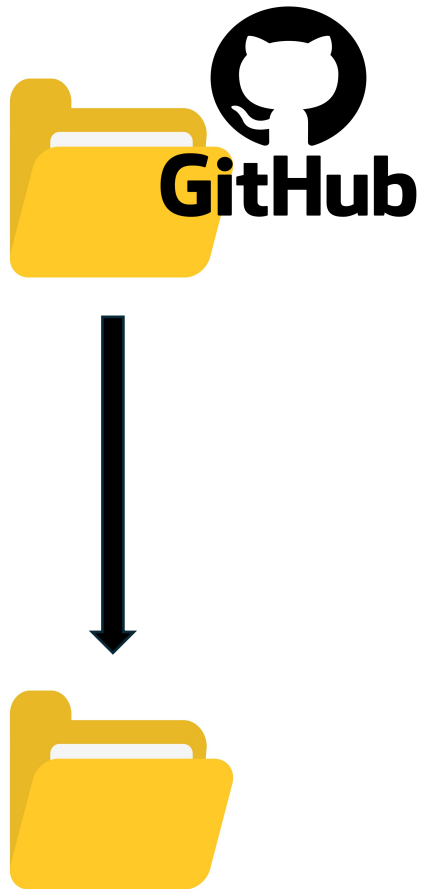
# The difference between cloning and forking

- **Cloning** a repository from Github **creates a local copy of it on your computer**, that you will interact with by pushing and pulling
  - This is what you use if you want to collaborate with others: you push your modifications, and pull theirs
  - To push your modifications on someone else's repository, you need to have their permission. If you don't, you cannot push your change
- **Forking** a repository **copies someone's repository on your Github**, you can then clone it on your computer and modify it independently from the original repo
  - This is useful when you want to reuse someone else's code or project, and adapt it to your own needs.
  - You can also then propose your changes to the owner of the original repo (by doing a pull request)

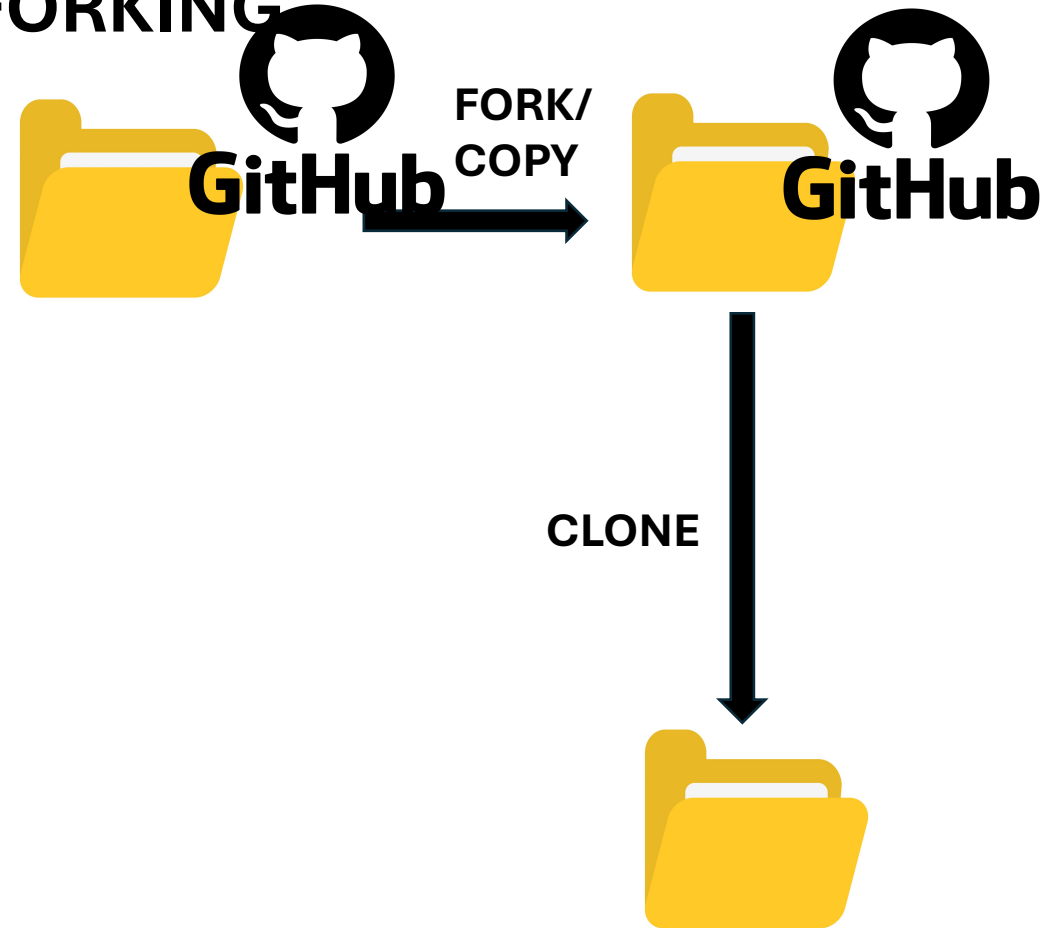
## Side note:

# The difference between cloning and forking

### CLONING



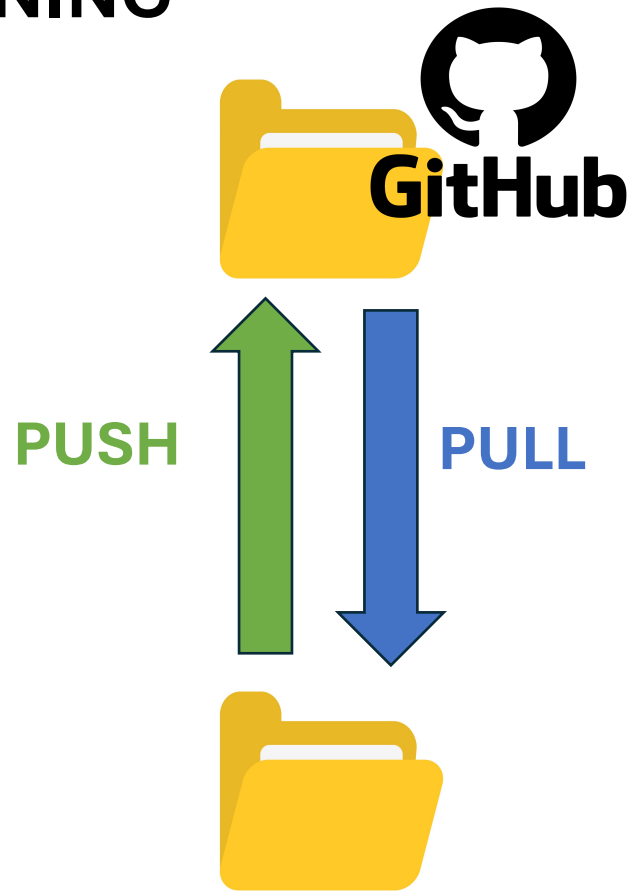
### FORKING



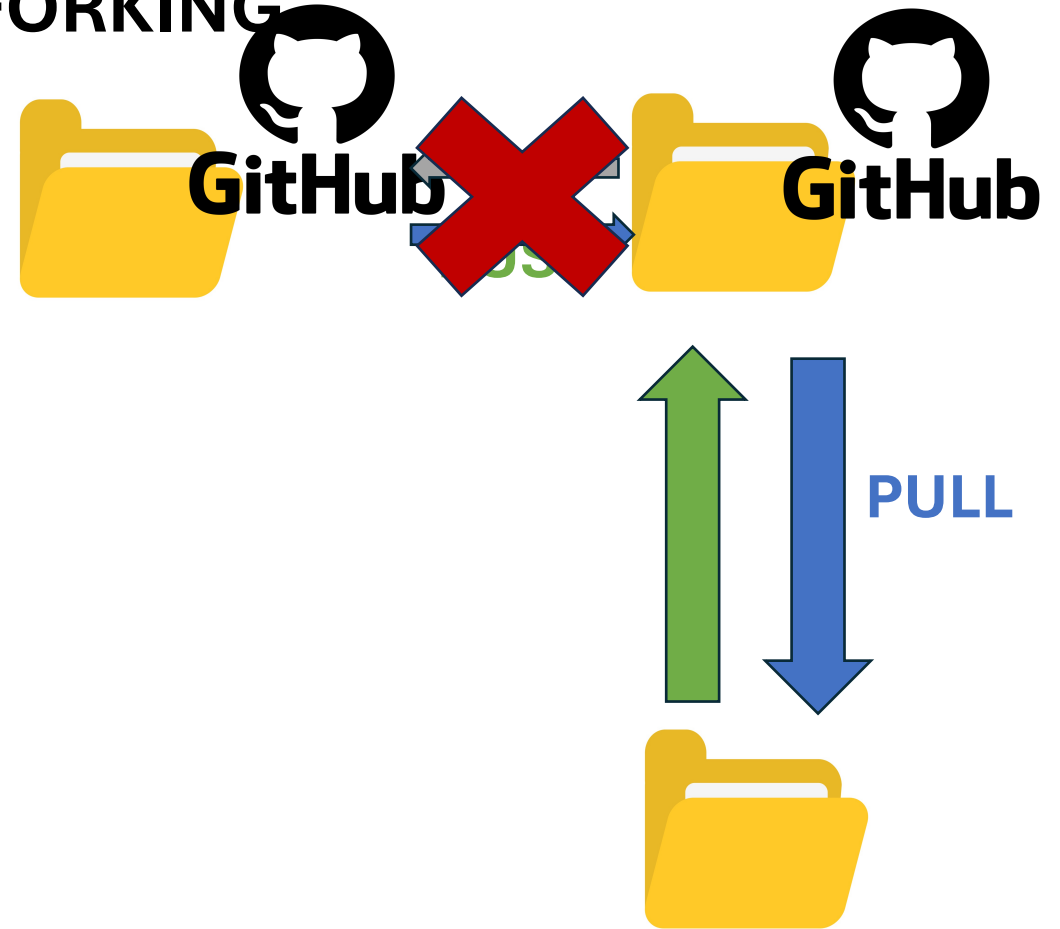
## Side note:

# The difference between cloning and forking

### CLONING



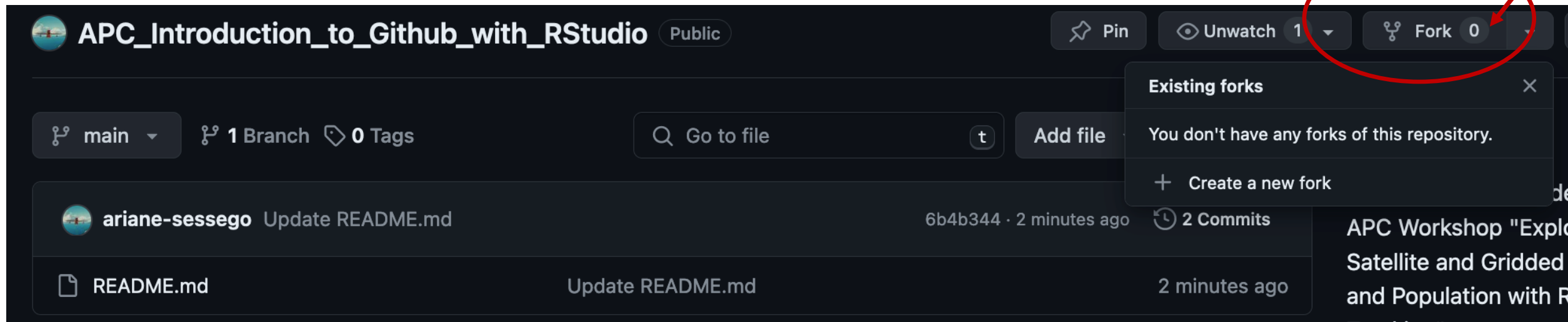
### FORKING



# Let's fork the workshop repo on your Github!

- Go to the address of our workshop repository and click on « fork »:

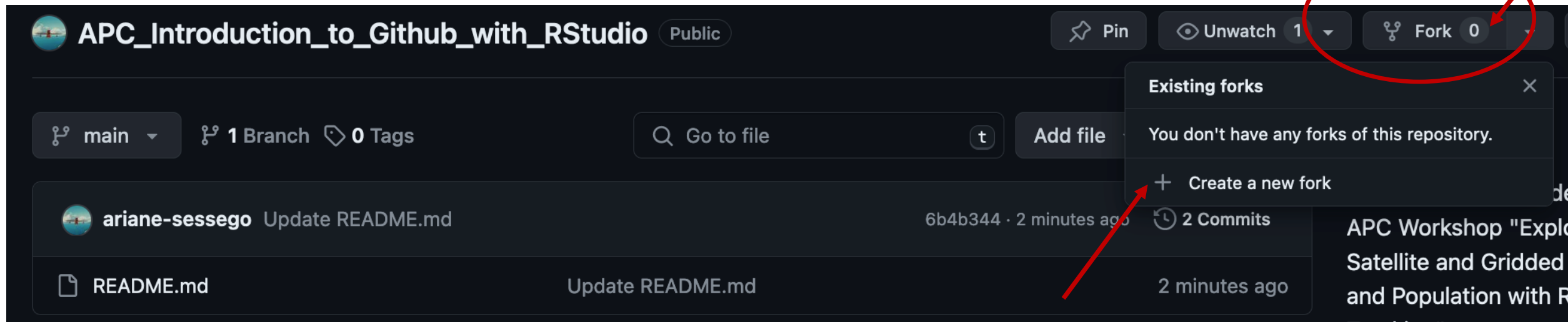
[https://github.com/ariane-sessego/APC\\_Introduction\\_to\\_Github\\_with\\_RStudio](https://github.com/ariane-sessego/APC_Introduction_to_Github_with_RStudio)



# Let's fork the workshop repo on your Github!

- Go to the address of our workshop repository and click on « fork »:

[https://github.com/ariane-sessego/APC\\_Introduction\\_to\\_Github\\_with\\_RStudio](https://github.com/ariane-sessego/APC_Introduction_to_Github_with_RStudio)

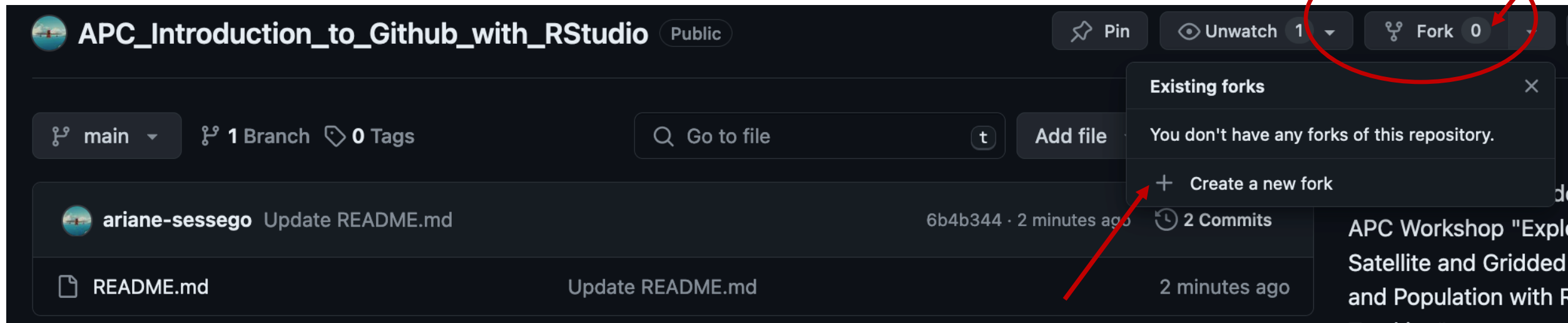




# Let's fork the workshop repo on your Github!

- Go to the address of our workshop repository and click on « fork »:

[https://github.com/ariane-sessego/APC\\_Introduction\\_to\\_Github\\_with\\_RStudio](https://github.com/ariane-sessego/APC_Introduction_to_Github_with_RStudio)



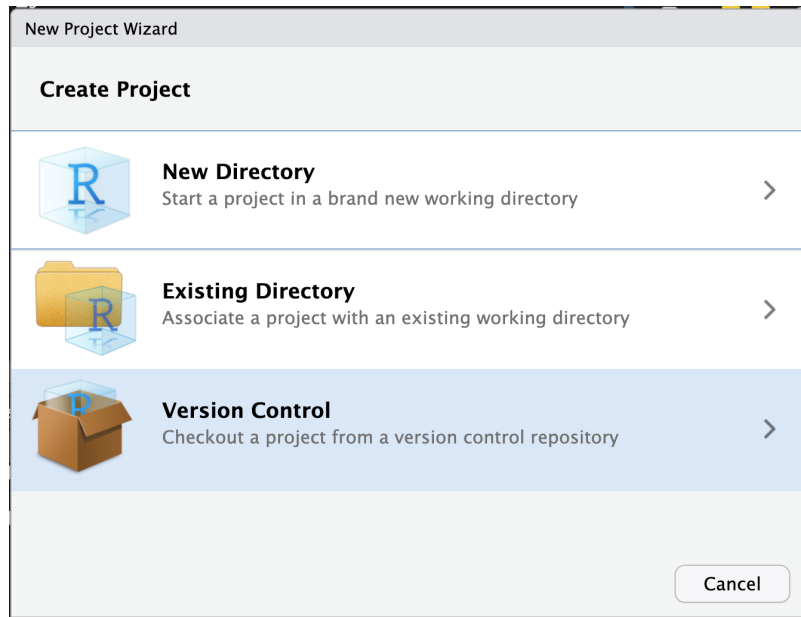
- Chose a name and it is copied on your own Github on your own!

## 2. Clone a Github repository in RStudio

- Now you can clone the your Github repository on an Rstudio project on your computer

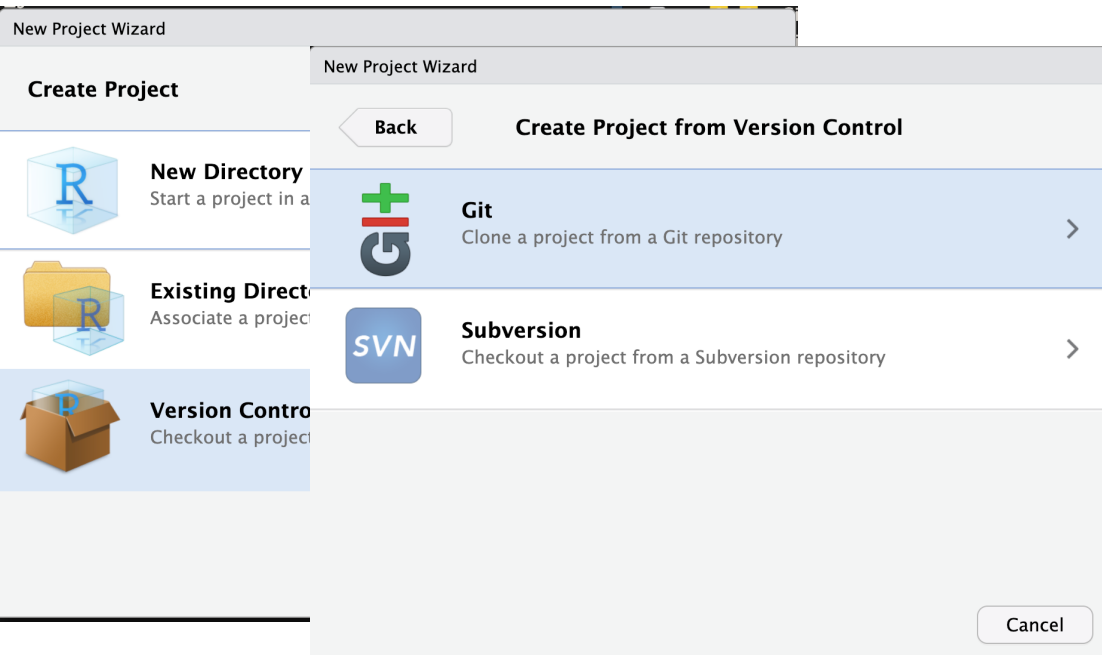
## 2. Clone a Github repository in RStudio

- Now you can clone the your Github repository on an Rstudio project on your computer
  - In Rstudio, create a new project: File> New Project> Version control > Git > Clone Git Repo



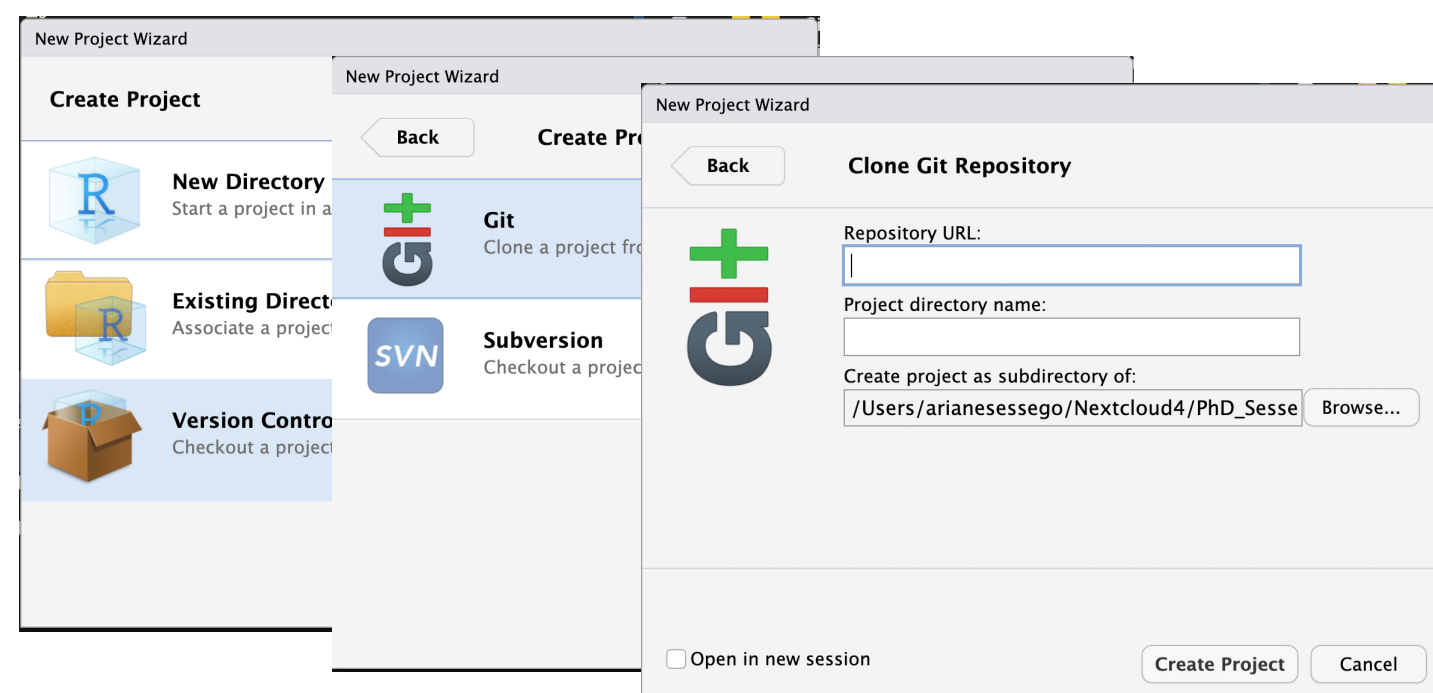
## 2. Clone a Github repository in RStudio

- Now you can clone the your Github repository on an Rstudio project on your computer
  - In Rstudio, create a new project: File> New Project> Version control > Git > Clone Git Repo



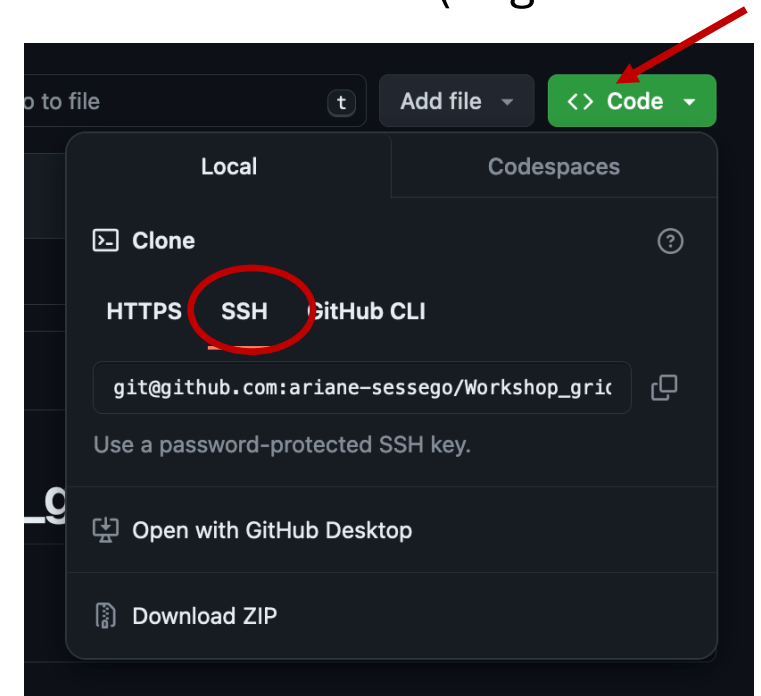
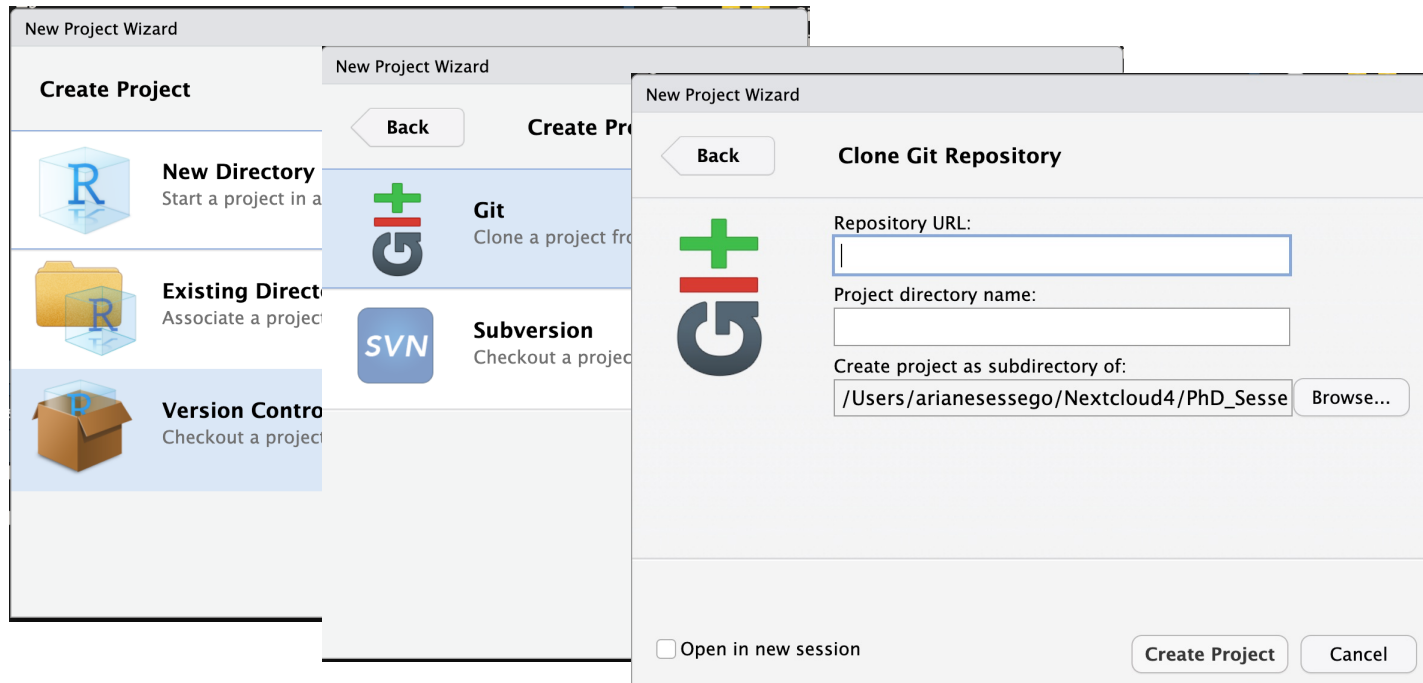
## 2. Clone a Github repository in RStudio

- Now you can clone the your Github repository on an Rstudio project on your computer
  - In Rstudio, create a new project: File> New Project> Version control > Git > Clone Git Repo



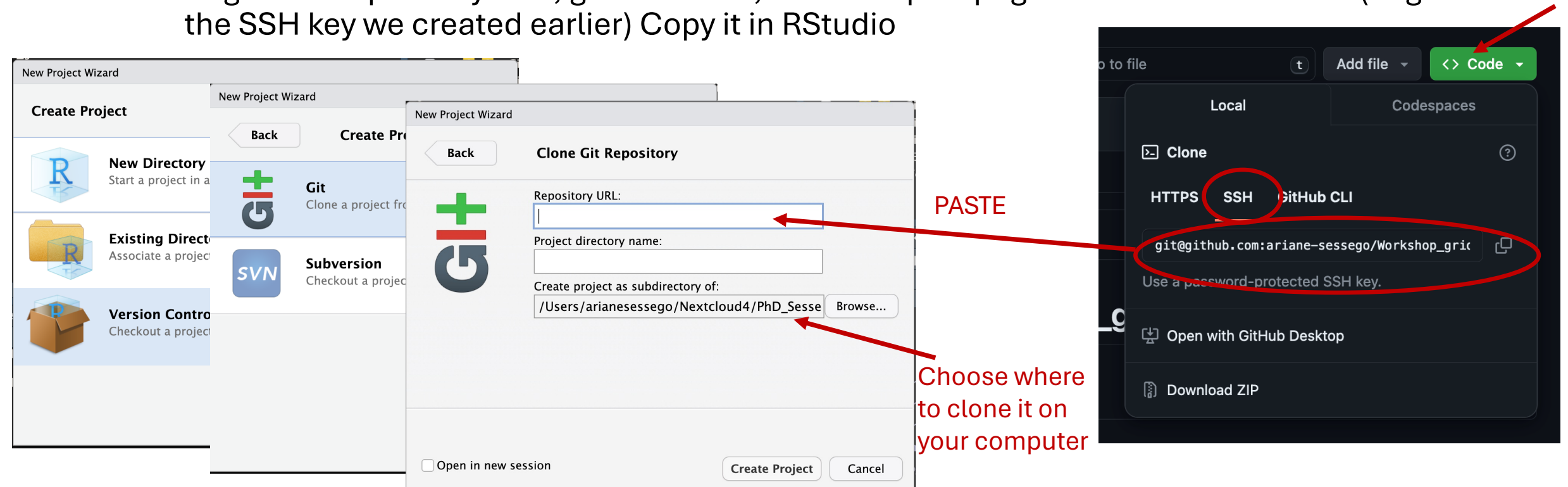
## 2. Clone a Github repository in RStudio

- Now you can clone the your Github repository on an Rstudio project on your computer
  - In Rstudio, create a new project: File> New Project> Version control > Git > Clone Git Repo
  - To get the Repository URL, go on Github, on the Repo's page: Code > Clone > SSH (to go with the SSH key we created earlier) Copy it in RStudio



## 2. Clone a Github repository in RStudio

- Now you can clone the your Github repository on an Rstudio project on your computer
  - In Rstudio, create a new project: File> New Project> Version control > Git > Clone Git Repo
  - To get the Repository URL, go on Github, on the Repo's page: Code > Clone > SSH (to go with the SSH key we created earlier) Copy it in RStudio



# 3. Use Git and Github inside RStudio

We will see how to do the following actions through the RStudio interface:

- A. Commit
- B. Push the commit(s) on Github
- C. Pull modifications from Github
- D. Handle a conflict



# 3. Use Git and Github inside RStudio

We will see how to do the following actions through the RStudio interface:

- A. Commit
- B. Push the commit(s) on Github
- C. Pull modifications from Github
- D. Handle a conflict

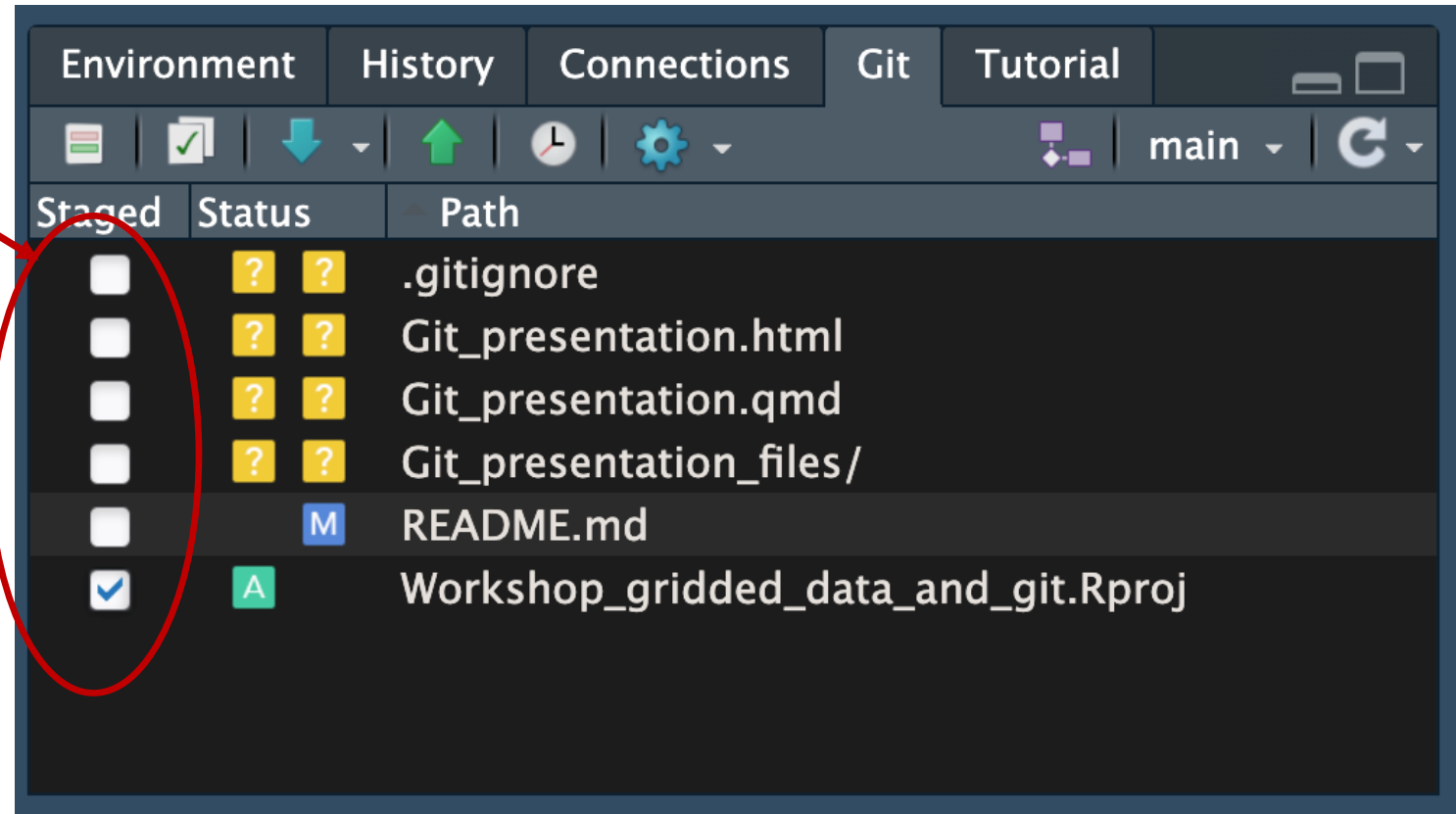
All these (and much more) can also be done through the terminal, but we will not cover that here. It can be useful to know to fix certain bugs. For more info, see:

For an introduction: <https://medium.com/@mrcherry/tutorial-using-github-through-terminal-3fc547b76c0b>

A cheatsheet of commands: <https://education.github.com/git-cheat-sheet-education.pdf>

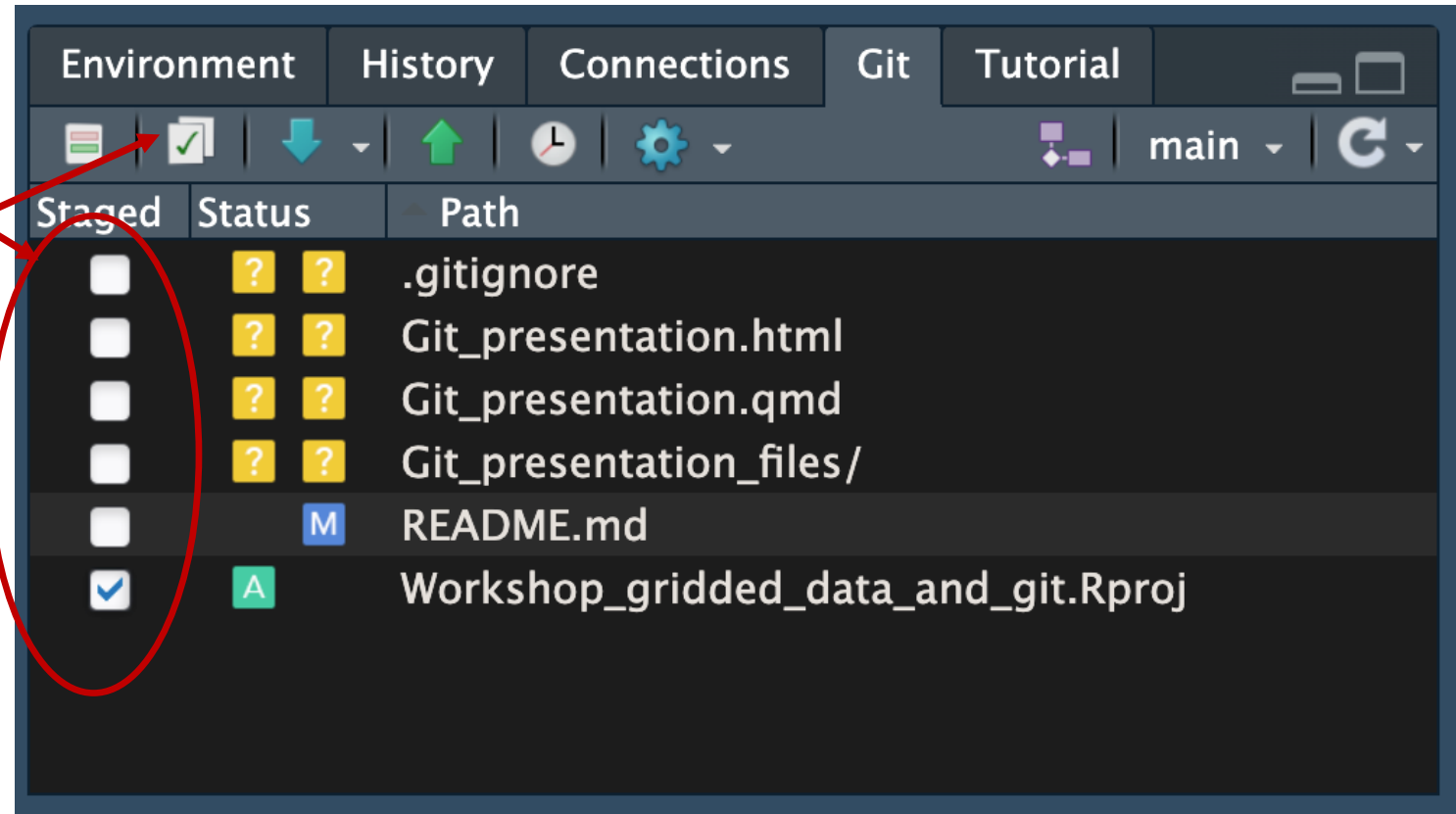
# A. Commit

1. Select what you want to commit



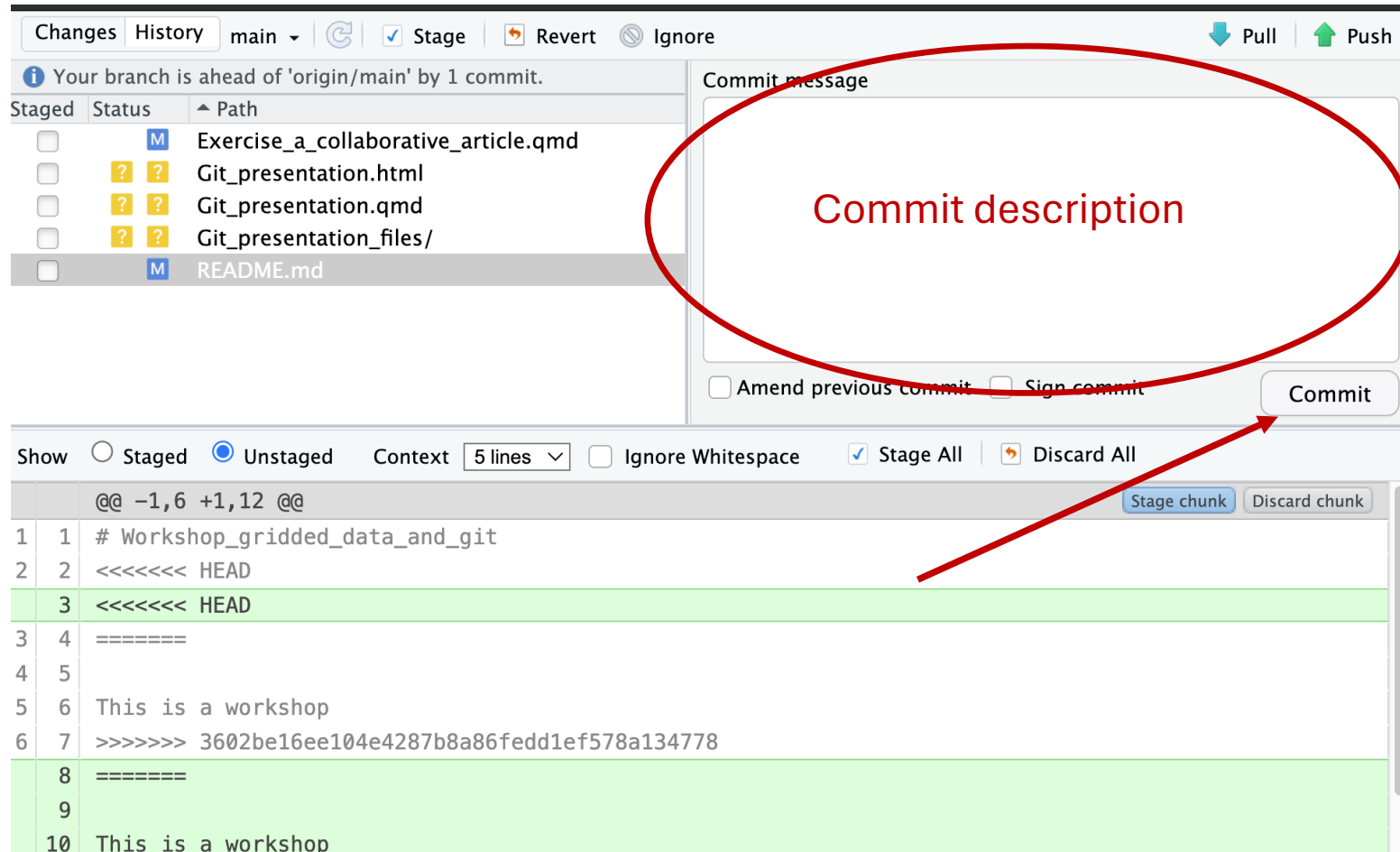
# A. Commit

1. Select what you want to commit
  - Don't forget to save your changes first
2. Go on the console to commit



# A. Commit

1. Select what you want to commit
  - Don't forget to save your changes first
2. Go on the console to commit
3. Create a commit message and commit
  - Be as descriptive as possible, this will help you to track your changes



# A. Commit

1. Select what you want to commit
2. Go on the console to commit
3. Create a commit message and commit
  - Be as descriptive as possible, this will help you to track your changes

You can then **view your commit history**

Diagram illustrating the commit process and the resulting commit history view.

**Commit History Table:**

Subject	Author	Date (UTC)	SHA
HEAD -> refs/heads/main Readme	Ariane Sessego <ariane.sesseg>	2024-05-14	310f60a2
Readme_changes Discussion, conclusion	Ariane Sessego <ariane.sesseg>	2024-05-14	4460f092
Paper structure	Ariane Sessego <ariane.sesseg>	2024-05-14	b255d83e
Paper structure	Ariane Sessego <ariane.sesseg>	2024-05-14	6c111346
made changes	Ariane Sessego <ariane.sesseg>	2024-05-14	c6d19e3a
Initial commit	ariane-sesseg <100150420+>	2024-05-06	57f56203

**Commit Details (SHA 310f60a2):**

- SHA: 310f60a2349cf3108e4640e69c36c745c9a0f9b3
- Author: Ariane Sessego <ariane.sesseg@gmail.com>
- Date (UTC): 2024-05-14 09:49
- Subject: Readme
- Parent: 4460f092a973bd0deaa549f0a457dff58a77e32b

**File Changes:**

- README.md

**Diff View (README.md):**

```
@@ -1,1 +1,6 @@
1 1 # Workshop_gridded_data_and_git
2  <<<<<<< HEAD
```

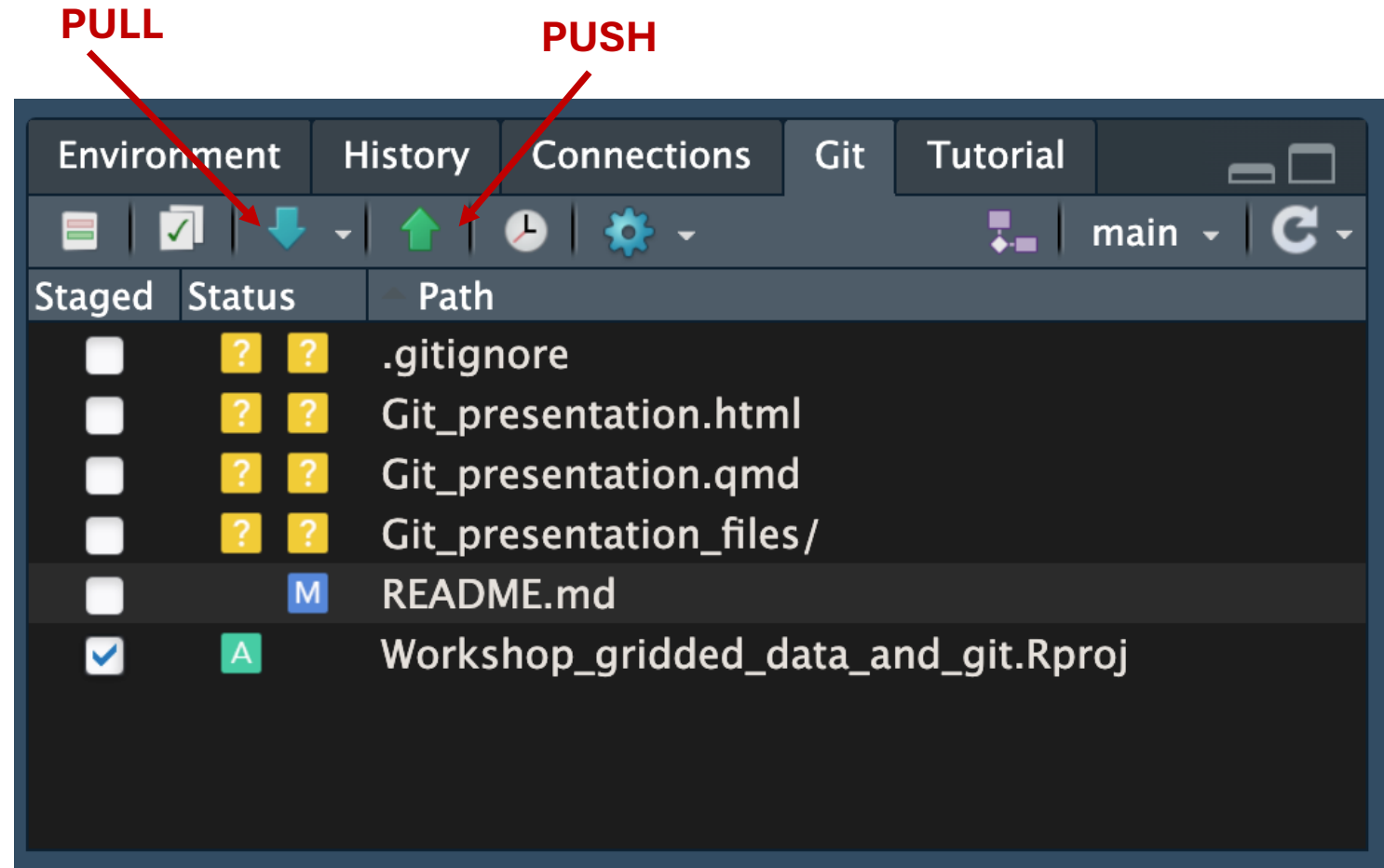
# A. Exercise

It's your turn!

1. Open the document “A\_collaborative\_article.qmd”
2. Edit the introduction:
  - Suggestion: modify “This is an introduction” to “This is a better introduction”
3. Save and commit your changes

## B. Push and Pull commits

- To push and pull your changes these use the two arrows

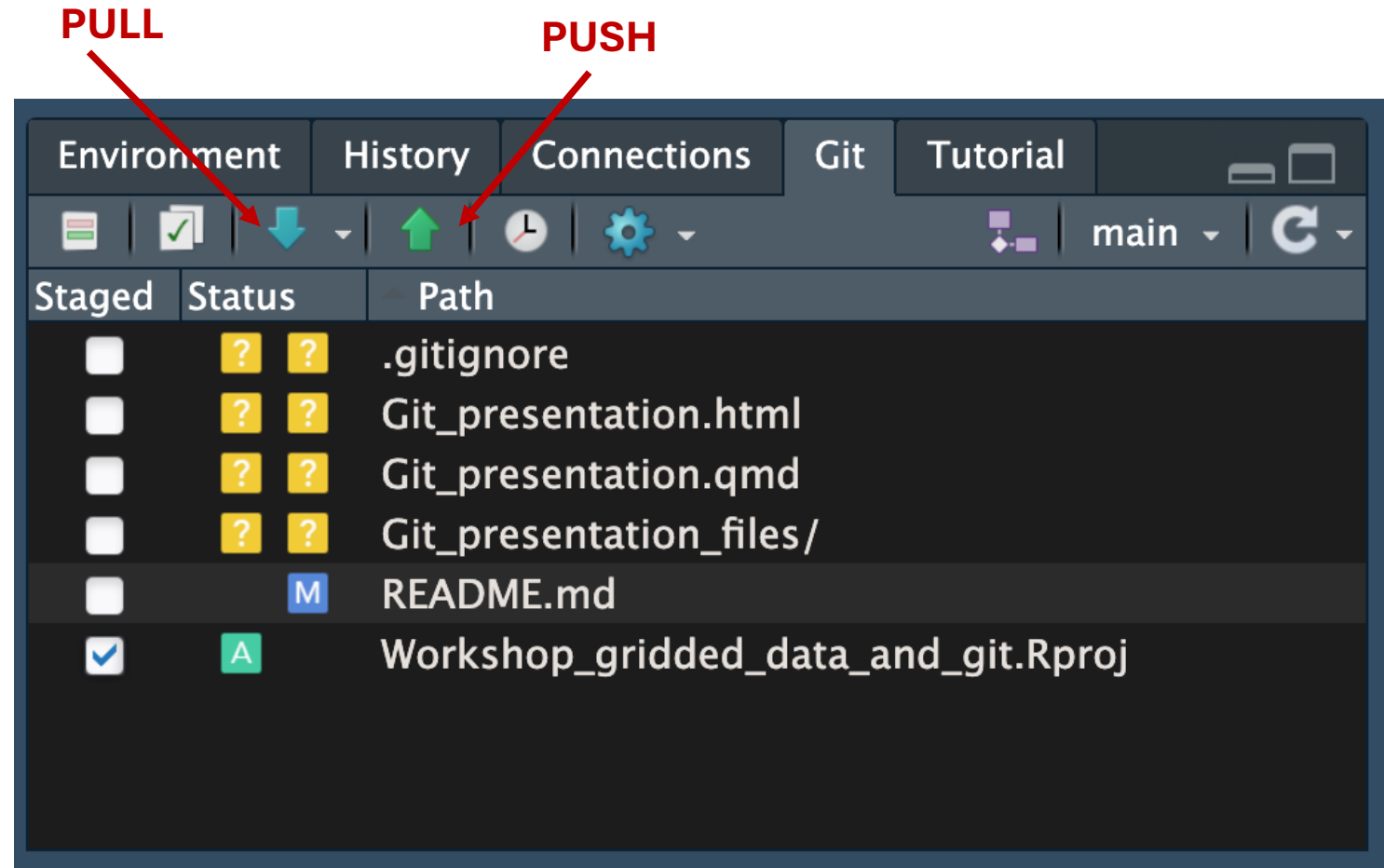


## B. Push and C. Pull commits

- To push and pull your changes these use the two arrows

### Exercise B.1 Push:

- Push your changes on Github using the arrow
- Go on your Github and look at the document: your changes have been integrated!

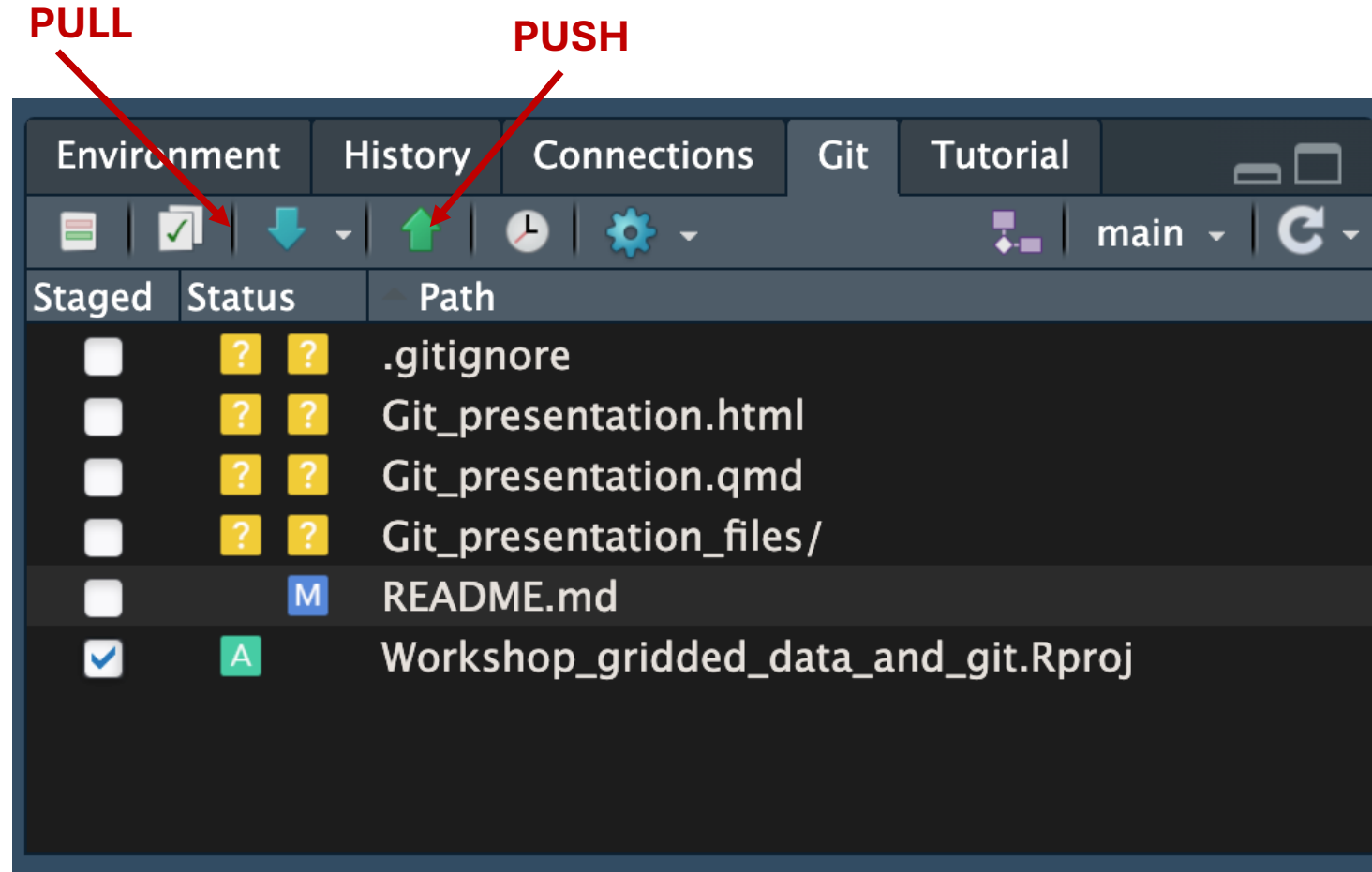




# B. Push and C. Pull commits

## Exercise B.2: Pull

- Go on your Github and edit the collaborative article:
  - Suggestion: change “These are our results.” to “These are our improved results”
- Go back to Rstudio and pull, you’ll see these changes appear in your R local R project!



# D. Handle a conflict

## Exercise D: Conflict

- We are going to create a conflict
- Go on Github and make changes to the conclusion and commit
  - Suggestion: change “This is our conclusion.” to “This is a better conclusion.”

# D. Handle a conflict

## Exercise D: Conflict

- We are going to create a conflict
- Go on Github and make changes to the conclusion and commit
  - Suggestion: change “This is our conclusion.” to “This is a better conclusion.”
- In RStudio, make different changes to the conclusion and commit
  - Suggestion: change to “This is our last words conclusion.”

# D. Handle a conflict

## Exercise D: Conflict

- We are going to create a conflict
- Go on Github and make changes to the conclusion and commit
  - Suggestion: change “This is our conclusion.” to “This is a better conclusion.”
- In RStudio, make different changes to the conclusion and commit
  - Suggestion: change to “This is our last words.”
- Try to push, you get an error message

Git Push

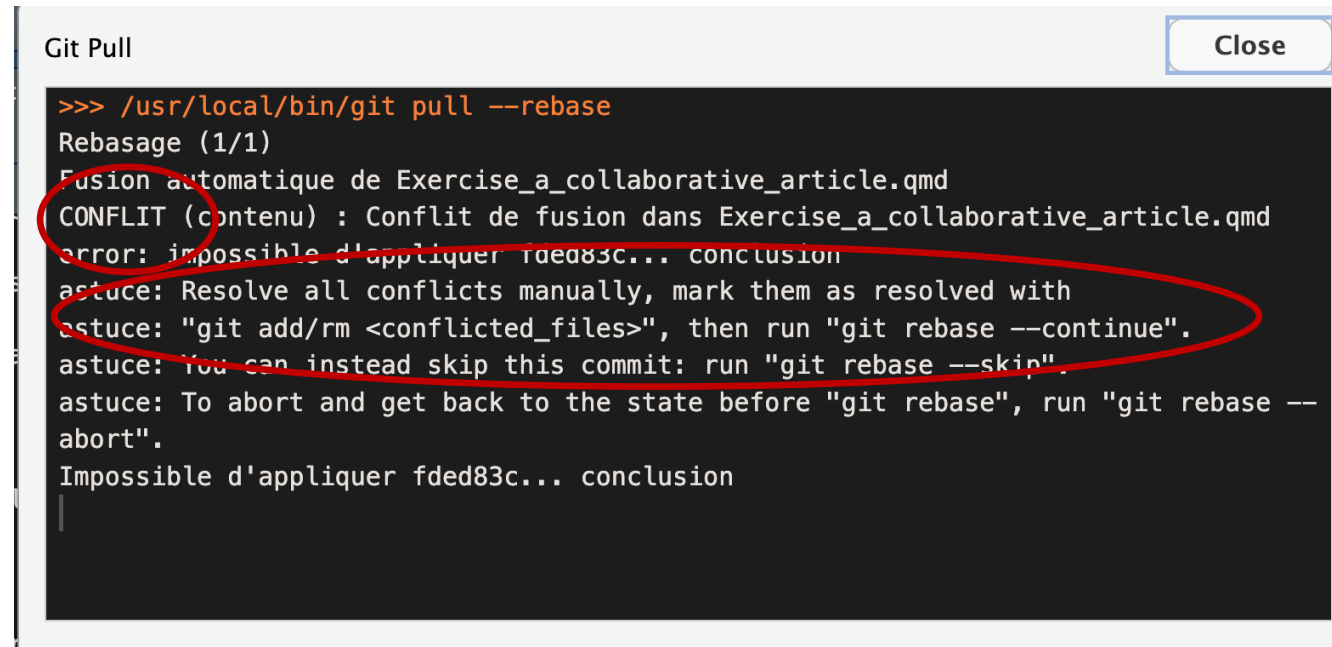
Close

```
>>> /usr/local/bin/git push origin HEAD:refs/heads/main
To https://github.com/ariane-sessego/Workshop_gridded_data_and_git.git
! [rejected]        HEAD -> main (fetch first)
error: impossible de pousser des références vers 'https://github.com/ariane-sessego/Workshop_gridded_data_and_git.git'
astuce: Les mises à jour ont été rejetées car la branche distante contient du travail que
astuce: vous n'avez pas en local. Ceci est généralement causé par un autre dépôt poussé
astuce: vers la même référence. Vous pourriez intégrer d'abord les changements distants
astuce: (par exemple 'git pull ...') avant de pousser à nouveau.
astuce: Voir la 'Note à propos des avances rapides' dans 'git push --help' pour plus d'information.
```

# D. Handle a conflict

## Exercise D: Resolving the conflict

- **You first need to pull**, to resolve the conflict locally
  - You will get a message indicating the conflict, letting you know you have to resolve it



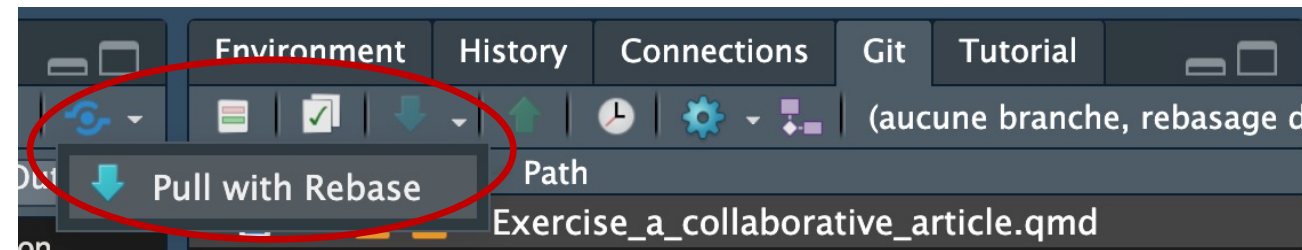
```
Git Pull Close  
  
>>> /usr/local/bin/git pull --rebase  
Rebasage (1/1)  
Fusion automatique de Exercise_a_collaborative_article.qmd  
CONFLIT (contenu) : Conflit de fusion dans Exercise_a_collaborative_article.qmd  
error: impossible d'appliquer fded83c... conclusion  
astuce: Resolve all conflicts manually, mark them as resolved with  
astuce: "git add/rm <conflicted_files>", then run "git rebase --continue".  
astuce: You can instead skip this commit: run "git rebase --skip".  
astuce: To abort and get back to the state before "git rebase", run "git rebase --  
abort".  
Impossible d'appliquer fded83c... conclusion  
|
```

# D. Handle a conflict

## Exercise D: Resolving the conflict

- **You first need to pull**, to resolve the conflict locally
  - You will get a message indicating the conflict, letting you know you have to resolve it
  - If not, while pulling you need to select "Pull with rebase" you should get the same error message
  - If you encounter more problems, come to see me, this is a configuration issue we can solve (but takes more time)

```
Git Pull
>>> /usr/local/bin/git pull --rebase
Rebasage (1/1)
Fusion automatique de Exercise_a_collaborative_article.qmd
CONFLIT (contenu) : Conflit de fusion dans Exercise_a_collaborative_article.qmd
error: impossible d'appliquer fded83c... conclusion
astuce: Resolve all conflicts manually, mark them as resolved with
astuce: "git add/rm <conflicted_files>", then run "git rebase --continue".
astuce: You can instead skip this commit: run "git rebase --skip".
astuce: To abort and get back to the state before "git rebase", run "git rebase --
abort".
Impossible d'appliquer fded83c... conclusion
```



# D. Handle a conflict

## Exercise D: Resolving the conflict

- **You first need to pull**, to resolve the conflict locally
  - You will get a message indicating the conflict, letting you know you have to resolve it
  - If not, while pulling you need to select "Pull with rebase"
- You then need to **select manually which version you want to keep**
  - The conflicts will be specified by text in between <<<< HEAD [the option from the origin] ===== [your local option >>>> xxxx (your identifier)]
  - You have to delete the option you don't want to keep and these extra indications (<<< ==)

```
## Conclusion

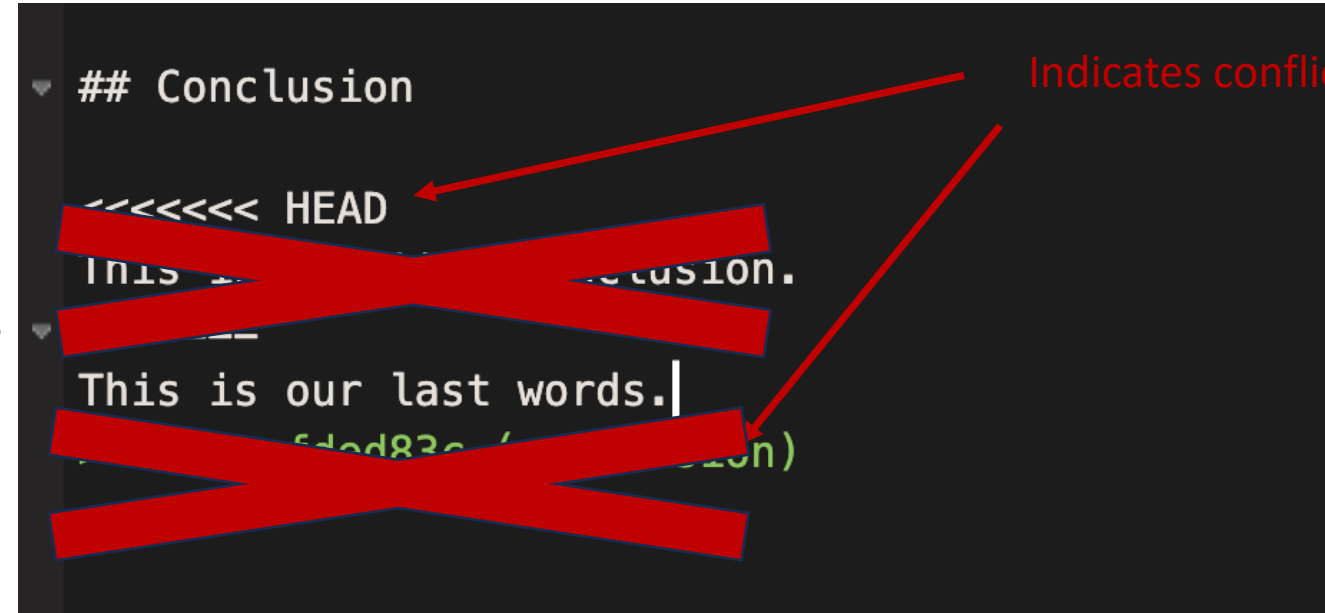
<<<<<< HEAD
This is a better conclusion.
=====
This is our last words.
>>>>>> fded83c (conclusion)
```

Indicates conflict

# D. Handle a conflict

## Exercise D: Resolving the conflict

- **You first need to pull**, to resolve the conflict locally
  - You will get a message indicating the conflict, letting you know you have to resolve it
  - If not, while pulling you need to select "Pull with rebase"
- You then need to **select manually which version you want to keep**
  - The conflicts will be specified by text in between <<<< HEAD [the option from the origin] ==== [your local option >>>> xxxx (your identifier)]
  - You have to delete the option you don't want to keep and these extra indications (<<< ==)



```
## Conclusion
<<<<<< HEAD
This is our last words.
====
This is our last words.
5ded83c / (your identifier)
```

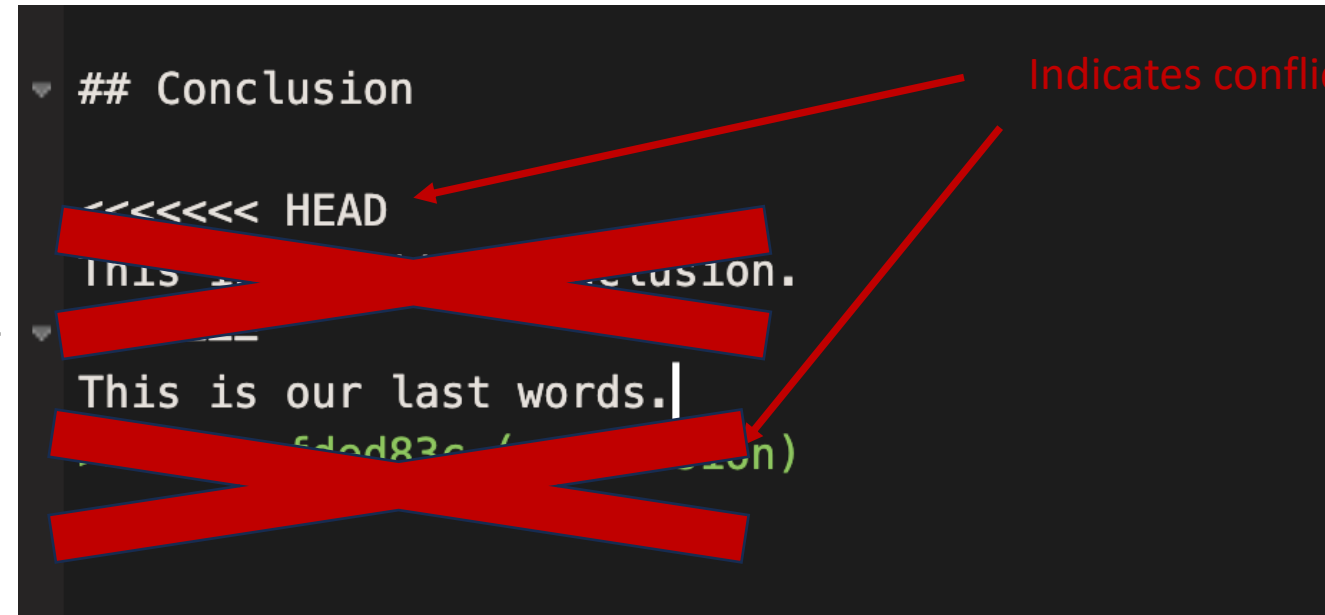
Indicates conflict



# D. Handle a conflict

## Exercise D: Resolving the conflict

- **You first need to pull**, to resolve the conflict locally
  - You will get a message indicating the conflict, letting you know you have to resolve it
  - If not, while pulling you need to select "Pull with rebase"
- You then need to **select manually which version you want to keep**
  - The conflicts will be specified by text in between <<<< HEAD [the option from the origin] ==== [your local option >>>> xxxx (your identifier)]
  - You have to delete the option you don't want to keep and these extra indications (<<< ==)



```
## Conclusion
<<<<<< HEAD
This is our last words.
====
This is our last words.
5ded83c / (your identifier)
```

Indicates conflict

- **Save, commit and push your changes: you have resolved the conflict!**

**Congratulations:**  
**You've learned how to use the main Git**  
**commands on Rstudio!**

# A lot of great resources to learn more

You will find a lot of resources via google and youtube (so much you could get lost). Here are a few I recommend looking at if you are curious:

- A series of **videos** that present these same basics and goes a little firther:

<https://posit.co/resources/videos/managing-part-2-github-and-rstudio/>

- A great in-detail guide in a **written format**: <https://happygitwithr.com/>

But as always, remember you will learn by doing, and by solving problem you encounter as you go!

# Git vocabulary quiz

Give a definition of all the following words: (I know, Git is a lot of jargon, but you'll get used to it quickly!)

- **Repository**
- **Remote repository**
- **Local repository**
- **Origin**
- **Commit**
- **Pull**
- **Push**
- **Conflict**
- **Git vs github**

# Git vocabulary quiz

Give a definition of all the following words: (I know, Git is a lot of jargon, but you'll get used to it quickly!)

- **Repository** – a folder where version are followed by Git
- **Remote repository** – the folder on the remote platform (here Github)
- **Local repository** – the folder on your computer
- **Origin** – other way to say the remote repository
- **Commit** – changes saved inside git's version history
- **Pull** - updates the local repository with new remote commits
- **Push** - updates the remote repository with new local commits
- **Conflict** – when the remote and local repository have to different versions of the same part of the code
- **Git vs github** – the software that keeps track of version vs. the website that allows you to store online and share this version control with others