

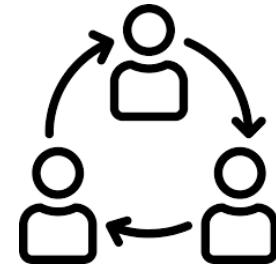
Git & GitHub Workshop

its not only for programmers



Why is git important for scientists?

Documentation Sharing / Collaborating

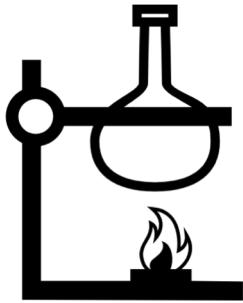


Both are important in science !

Git facilitates both!

Why is git important for scientists?

Data collection & experiment



Record parameters and changes
in lab notebook (paper/digital)

Data processing & analysis



?

Version control of code = lab notebook of experiment

"FINAL".doc



FINAL.doc!



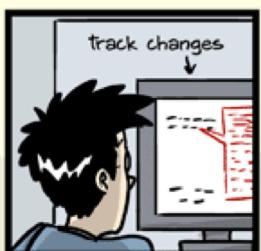
FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc



FINAL_rev.22.comments49.
corrections.10.#@\$%WHYDID
ICOMETOGRAD SCHOOL????.doc

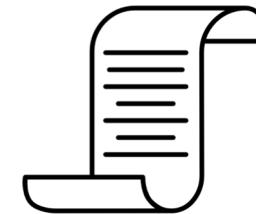


JORGE CHAM © 2012

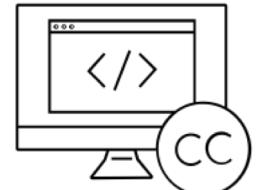
Why is learning git important for scientists?

Sharing/Collaboration

- Sharing codes are difficult through traditional means



- GitHub enables code sharing, but it uses git
 - Someone can **follow** your codes
 - Incorporate or suggest changes to others (even *strangers*!)
- Encourages open source and open science
 - Open science is a huge topic (**Kendra will talk about OSF tomorrow, Nov 17th!!**)
 - Openness to your code is generally a good idea



What is the goal of this workshop?

Git for version control



GitHub for sharing your code



Git Overview

What do we do for word documents?



Track Changes (Microsoft Word)

- Highlighted changes

DA DA DA DA DDA DA DA DA
DA DA DA DA DA D

BLA BLA BLA BLA BLA BLA BLA
B DA DA DA DA DA DALA BLA
BLA BLA BLA BLA BLA BLA BLA
BLA BLA BLA BLA BLA BLA BLA

Whale, Shady
Deleted: BLA BLA

Whale, Shady
Deleted: BLA BLA

Whale, Shady
Deleted: BLA BLA

BLA BLA BLA BLA
BLA BLA BLA BLA
BLA BLA BLA BLA
BLA BLA BLA BLA
BLA BLA BLA BLA

Version History (Google-doc)

- Snapshots

Total: 2 edits	Only show named versions
	<input checked="" type="checkbox"/>
ay, with the semi-transparent wm.mgz n emulation on a single button mouse) rrow	APRIL April 11, 1:25 PM <i>Current version</i> ● Shady Whale
	FEBRUARY

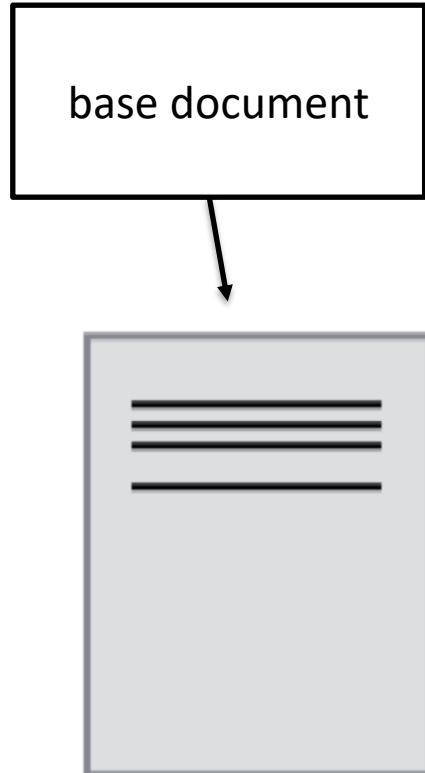
Git track changes & gives you a version history!

Git Overview

What is being tracked?



Don't think of it as multiple documents, but **base document + sets of changes**



Git Overview

Separate changes can be made, and recorded





Hands on portion of the workshop

Dark bubble with text are terminal commands (bash)

```
> echo hello
```

- Type into a terminal window or Git-Bash

OS icon will be next to the commands specific to an OS



Windows



Mac



Linux

Everything in this presentation is on a bookdown page:

<https://gitbookdown.site>



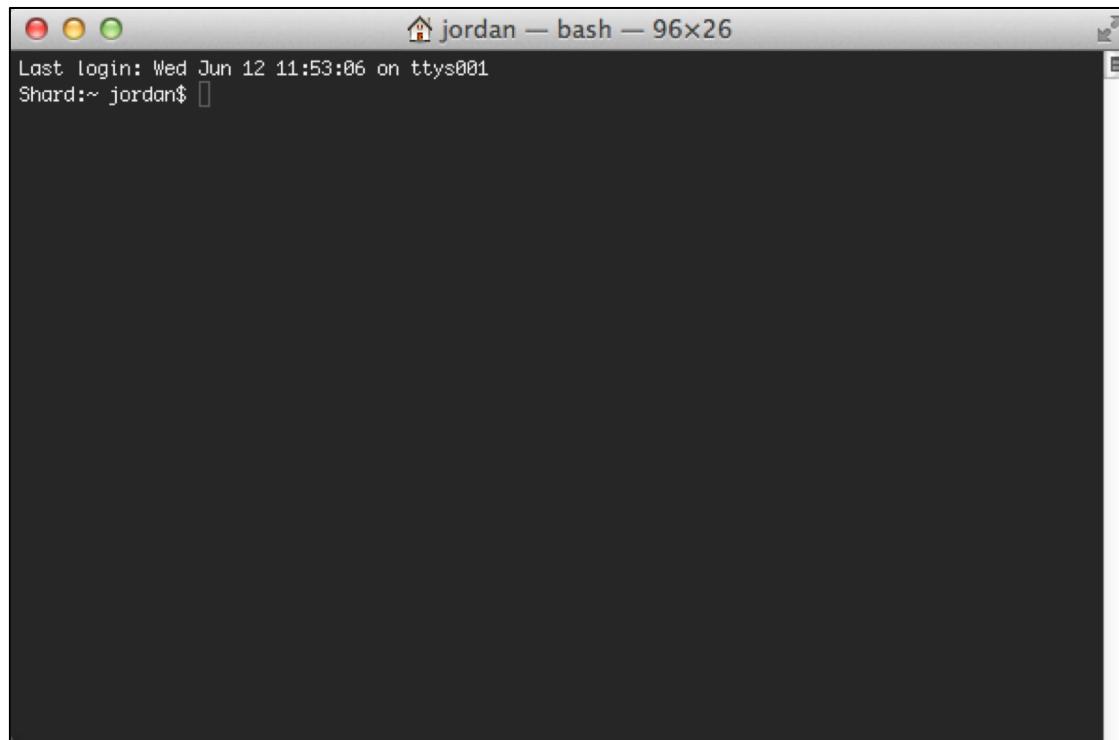
1. Setting up git

1.1 Download git

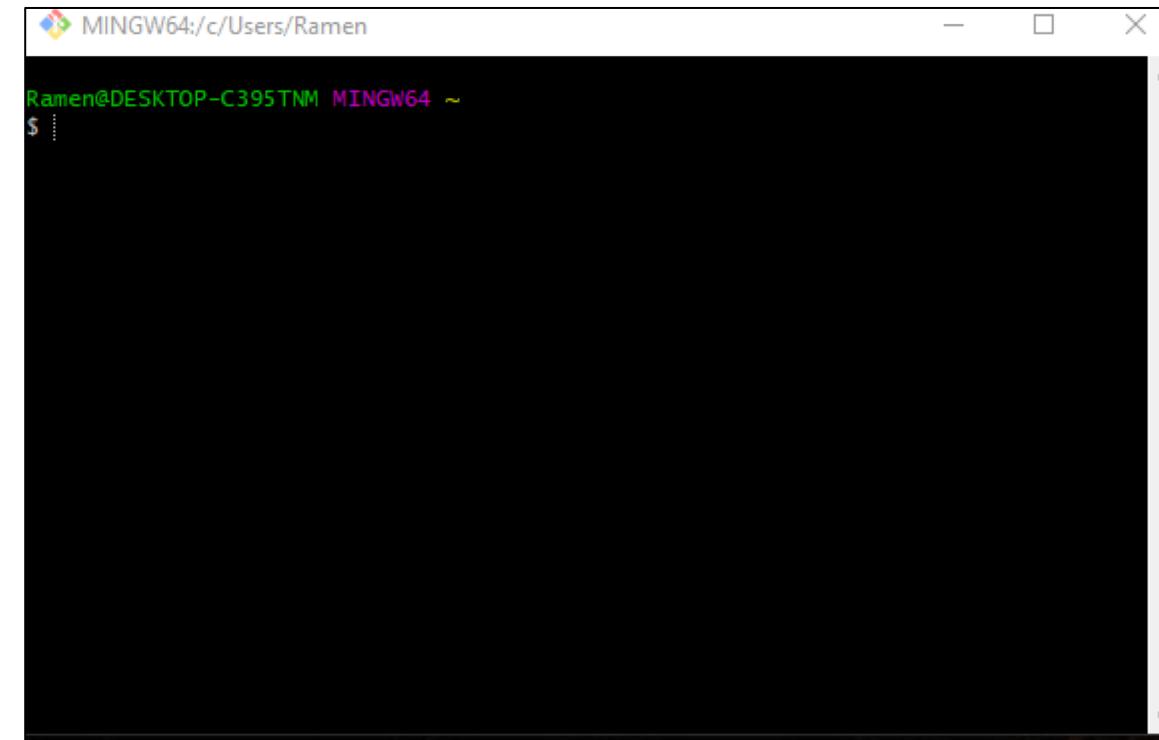
- Windows users: <https://gitforwindows.org/> The Windows logo, which is a blue circle containing four colored squares (red, green, blue, yellow).
- **Git-Bash** is a bash terminal that comes with Git for Windows.
- In this workshop, we assume Windows User have Git-Bash, so you can use functions such as “echo” or “touch”
- Linux/Mac Users: <https://git-scm.com> The Apple logo, which is a gray silhouette of an apple with a bite taken out of it. The Tux logo, which is a black penguin standing on its hind legs.

1. Setting up git

Open a terminal / Git-Bash



A screenshot of a Mac OS X terminal window titled "jordan — bash — 96x26". The window shows the user's last login information: "Last login: Wed Jun 12 11:53:06 on ttys001" and the prompt "Shard:~ jordan\$". The terminal has a dark background and a light gray border.



A screenshot of a Windows terminal window titled "MINGW64:/c/Users/Ramen". The window shows the user's login information: "Ramen@DESKTOP-C395TNM MINGW64 ~" and the prompt "\$". The terminal has a dark background and a light gray border.





1. Setting up git

1.2 Configure git

- Username & email

```
> git config --global user.name "Shady Whale"  
> git config --global user.email "shadywhale@allthewhales.com"
```

- Setup default text editor ('nano' is used in this tutorial)

```
> git config --global core.editor "nano -w"
```



2. Creating a repository

What is a repository?

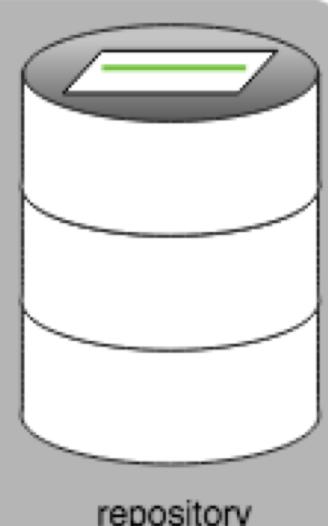
Repository (repo):

- An index that tracks changes of a directory
- a .git/ subdirectory

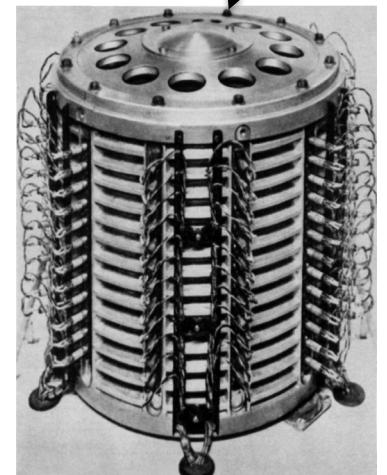
Directory



.git



Old school
drum memory



2. Creating a repository

> git init

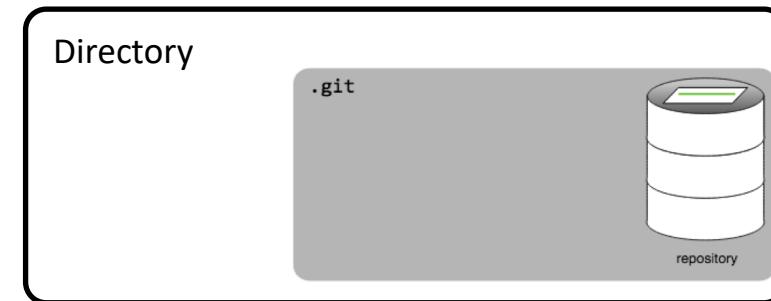


2.1 – Create a directory in your home directory

```
> mkdir ~/workdir  
> cd ~/workdir
```

2.2 – Create a **repository**

```
> git init
```



Check the status of the git repository

```
> ls -a  
> git status
```

You now have a repository (i.e., a .git/ folder) within a directory that you want to track!

3. Tracking Changes

> git add, commit

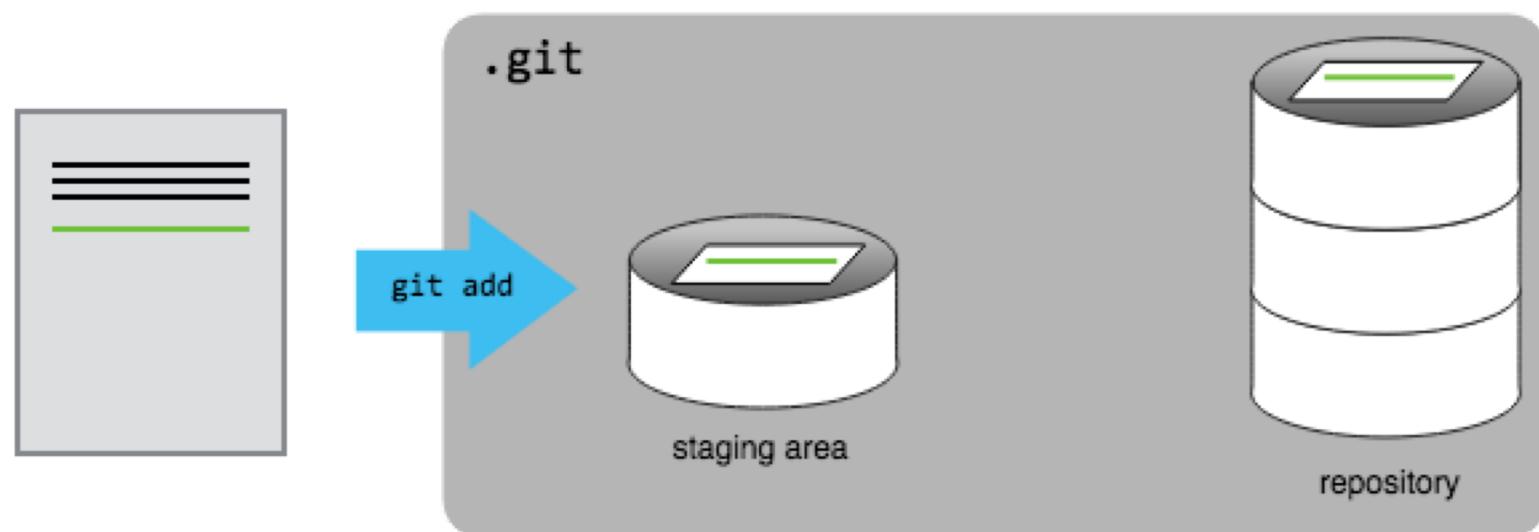


git add

- Tells git which changes you want to track
- Formally, this is called adding a file to the “staging area”

git commit

- Tell git to document the changes you “added” (i.e., the files in the staging area)
- Mandatory **commit message** that you **must provide** to explain what changes were made



3. Tracking Changes

> git add, commit



3.1 – Adding new files/modification to **staging area**

- Make a new file and check status of git

```
> touch foo.txt  
> git status
```

- Let's tell Git to track the file

```
> git add foo.txt  
> git status
```

3. Tracking Changes

> git add, commit



3.2 – Saving these changes to the **repository**

- Commit the file with a commit message

```
> git commit -m "Add foo.txt to repo"
```

- Let's check the log, which lists all the commits made so far

```
> git log
```

3. Tracking Changes

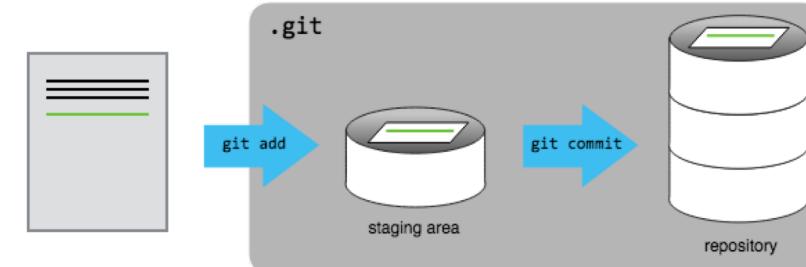
> git add, commit



PRACTICE:

- Let's add a line to the file and **git add & commit** it!

```
> echo hello >> foo.txt  
> git add foo.txt  
> git commit -m "Add hello in foo.txt"  
> git log
```



You have now used **git add & commit** to track a file. Use **git log** to read what you have done so far in this repository!

4. Exploring History

> git diff & show



4.1 – Difference between current file and N commit ago

- Edit the file and check difference between your current foo.txt and the last commit

```
> echo world >> foo.txt  
> git diff HEAD foo.txt
```

- What about compared to two commits ago?

```
> git diff HEAD~1 foo.txt
```

HEAD~1 can be replaced with
the specific identifier of a
commit. Use git log to check

4. Exploring History

> git diff & show



4.2 – What was done in _____ commit?

- Sometimes we want to check what was done during a certain commit

```
> git show HEAD~1 foo.txt
```



5. Telling git to ignore certain files

.gitignore

- Sometimes we don't want git to track a certain type of files
 - Temporary files/big data/OS files
- Adding a file to .gitignore

```
> touch ignore_this.txt  
> echo ignore_this.txt >> .gitignore
```

- Try `git add` a file we added to .gitignore, it won't let us

```
> git add ignore_this.txt
```

There should now be a .gitignore file in your directory.

Desktop Application



GitHub Desktop

FREE ↗



Sourcetree

FREE ↗



GitKraken

FREE ↗ \$



Tower

\$

The screenshot shows the GitHub Desktop application interface. At the top, it displays the current repository as "git_github_bookdown" and the current branch as "master". Below this, there are tabs for "Changes" (19) and "History". The main area shows a diff view of a file named "docs/branching-git-branch.html". The left pane lists 19 changed files, including "docs/branching-git-branch.html" which has 12 changes. The right pane shows the diff output, where lines 7 through 17 are highlighted in blue, indicating they are part of the selected commit. The commit message "It's Not Only for Programmers" is visible at the bottom of the diff. A "Commit to master" button is at the bottom left.

```
@@ -4,12 +4,12 @@
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
- <title>12 Branching (git branch) | Git/GitHub Tutorial for Scientists: It's Not Only for Programmers</title>
- <meta name="description" content="Git/GitHub Tutorial for Scientists:<br />
+ <title>12 Branching (git branch) | Git & GitHub Tutorial for Scientists: It's Not Only for Programmers</title>
+ <meta name="description" content="Git & GitHub Tutorial for Scientists:<br />
It's Not Only for Programmers" />
- <meta name="generator" content="bookdown 0.11 and GitBook 2.6.7" />
+ <meta name="generator" content="bookdown 0.13 and GitBook 2.6.7" />
- <meta property="og:title" content="12 Branching (git branch) | Git/GitHub Tutorial for Scientists: It's Not Only for Programmers" />
+ <meta property="og:title" content="12 Branching (git branch) | Git & GitHub Tutorial for Scientists: It's Not Only for Programmers" />
<meta property="og:type" content="book" />
<meta property="og:url" content="https://gitbookdown.site/" />
@@ -17,7 +17,7 @@ It's Not Only for Programmers" />
<meta name="github-repo" content="mychan24/git_github_bookdown" />
```

This ends the intro to the **version control** function of git.



Next up:

GitHub and its cousins to help you share your codes



BREAK

If you don't have a GitHub account yet,
please sign-up for an account at <https://github.com>

GitHub
and other remote hosts

A remote server that hosts repositories for free



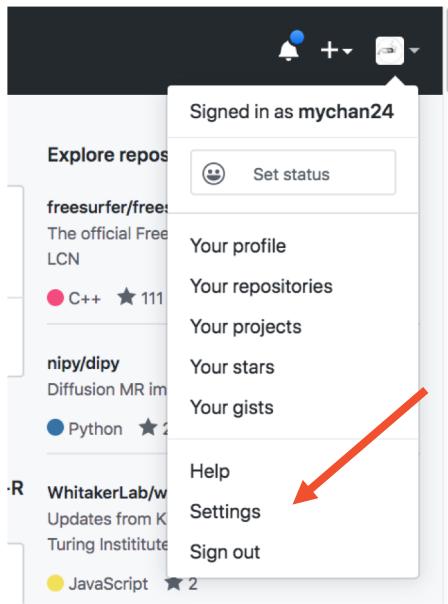
It also offers visualization of code changes and collaboration insights!
(lets take a look)

Some other choices:

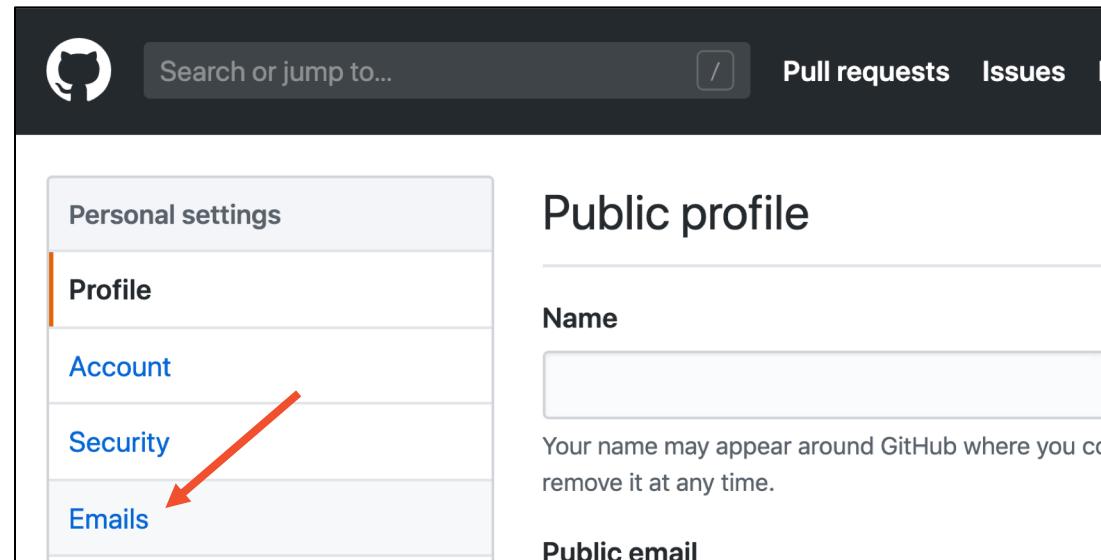


Configure GitHub's Email Setting

1) Go to Settings



2) Click Emails



3) Enter email you used for workdir

A screenshot of a 'Add email address' form. It has a text input field and an 'Add' button.

4) For this tutorial, un-check 'Block command'

Block command line pushes that expose my email

If you push commits that use a private email as your author email we will block the push and warn you about exposing your private email.

6. Create a repository on GitHub

make a repo



Create a remote repository

- Go to <https://github.com/<your username>>
- Click the **Repositories** tab
- Click **New**
- Enter **workdir** as the Repository name, click **Create repository**
- **NOTE:** We will skip adding a README
- Click **Clone or download ▾** to copy the URL of this repository

You now have an empty remote repository and some instructions to populate it!

7. Putting codes on GitHub

> git remote add & push

git remote add:

- Add a remote repository by its url so you can down/up-load from it.



git push:

- Upload a commit(s) and modifications to a remote repository.



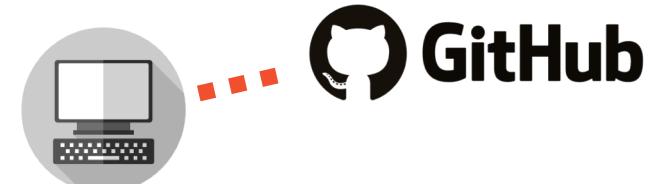
7. Putting codes on GitHub

> git remote add & push

7.1 – Push an existing repository from your local computer

- Go to workdir & add a remote repository that you want to push your data to

```
> cd ~/workdir  
> git remote add origin <copied URL>
```



<https://github.com/<username>/workdir.git>

- Push your local repository to your GitHub

```
> git push -u origin master
```



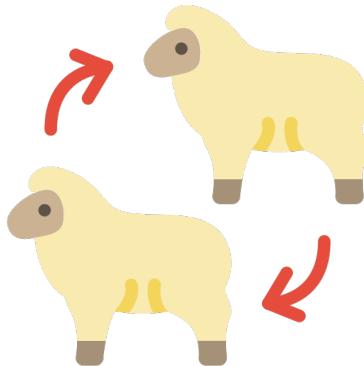
Your code is now available on GitHub!

8. Getting a repository from GitHub

git clone & fork



Clone

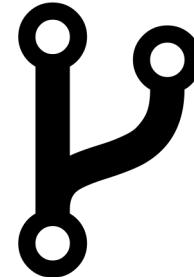


Copy a entire repository

Usually we clone from **GitHub** to our **computer**

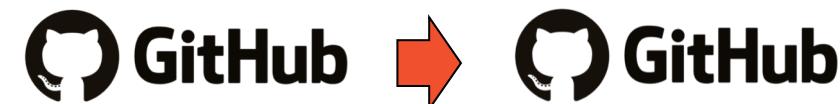


Fork



Copy a GitHub repository into your GitHub

Its essentially cloning from **GitHub** to **GitHub**

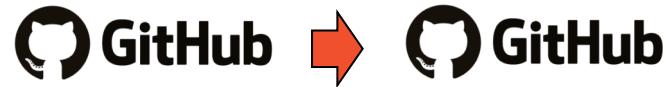


8. Getting a repository from GitHub

Practice fork & clone



8.1 – Lets Fork the repository with this presentation



- Go to https://github.com/mychan24/git_github_workshop
- Fork it by clicking A rectangular button with a thin gray border. Inside, there is a small icon of a fork and the word "Fork" in a dark blue sans-serif font.
- You'll be redirected to a copy of the same repository in **YOUR** GitHub account



8. Getting a repository from GitHub

Practice fork & clone

8.2 – Now lets clone that to your computer



- Go to https://github.com/<username>/git_github_workshop
- Click to copy the URL for cloning
- Clone it using the following command line

```
> cd ~  
> git clone <copied URL>
```

You have forked & cloned someone else's repository! This is often how collaboration starts through GitHub.

9. Collaborating

9.1 – Setup a fake collaborator (i.e., yourself in another directory)

- Go to your GitHub account's workdir repository:
- Copy the URL 
- Clone it to a new directory call at “~/workdir_fake_collab”

```
> git clone <copied URL> ~/workdir_fake_collab
```

<https://github.com/<username>/workdir.git>



9. Collaborating

- Make changes as fake collaborator and push to GitHub**

- Go into the collab's directory, make a new file foofoo.txt

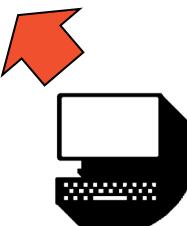
```
> cd ~/workdir_fake_collab  
> touch foofoo.txt
```

- Git add & commit foofoo.txt

```
> git add foofoo.txt  
> git commit -m "Add foofoo.txt"
```

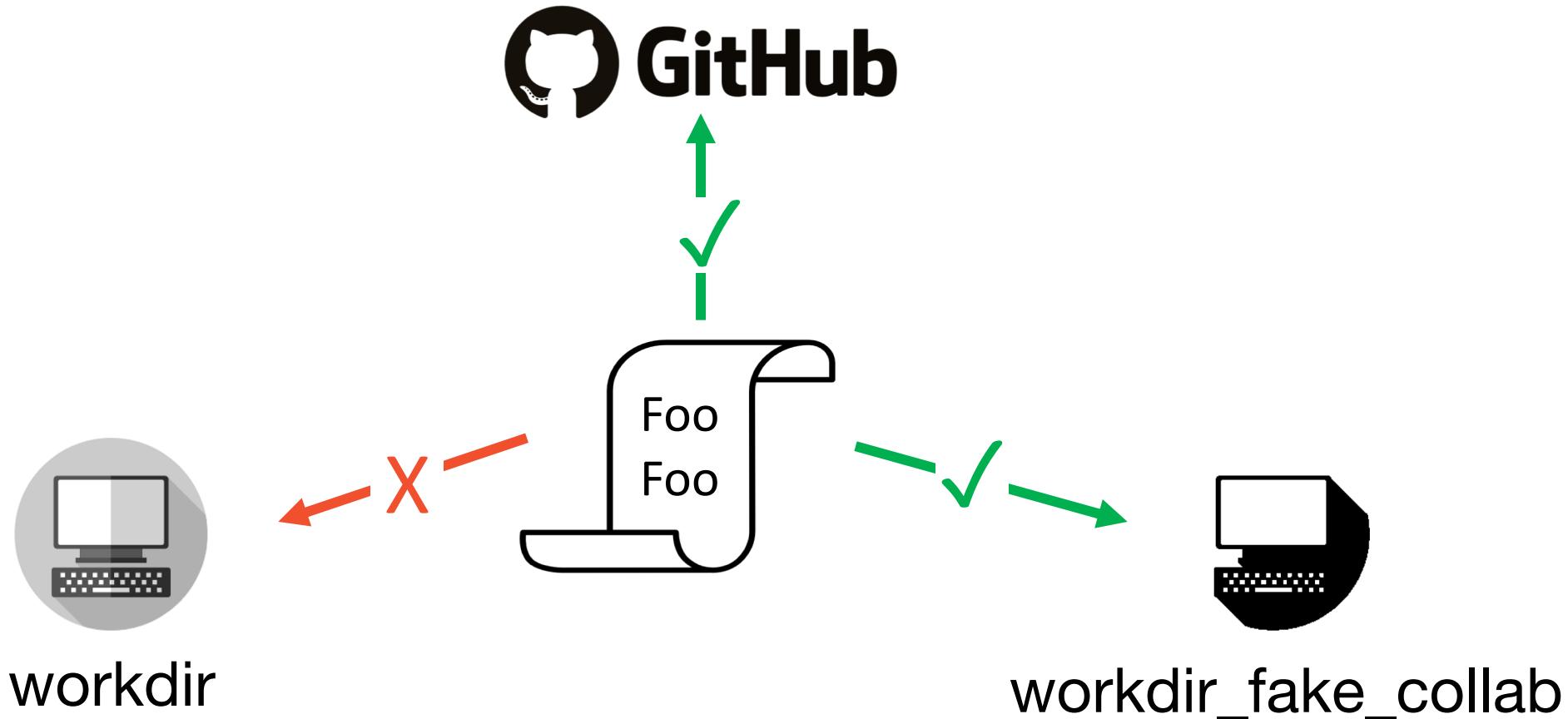
- Push this change to GitHub**

```
> git push origin master
```



9. Collaborating

Where is foofoo.txt?

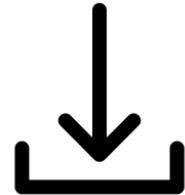


9. Collaborating

> git fetch & git merge

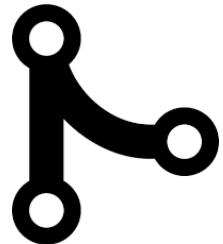


Fetch



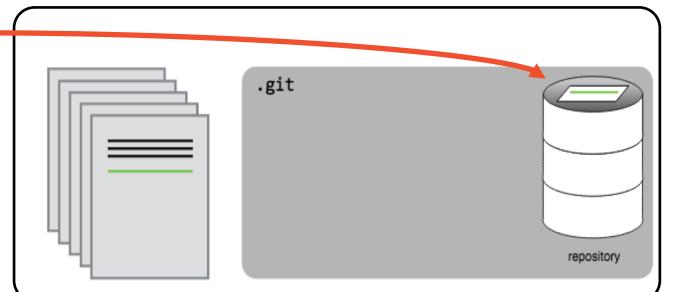
Download snapshot of remote

Merge

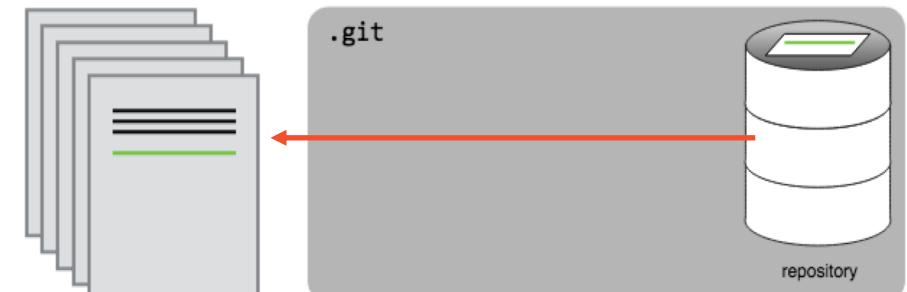


Integrate fetched changes

Downloading the changes into .git



Incorporate changes to local files



9. Collaborating

> git fetch & git merge

9.2 – Implement the changes in original workdir

- **Fetch** the changes your fake collaborator pushed to GitHub

```
> cd ~/workdir  
> git fetch
```

- Check what your collaborator has changed compared to your local version

```
> git diff master origin/master
```

Your local
master branch

Master branch on GitHub
(origin is what we nicknamed our
remote repo on GitHub)

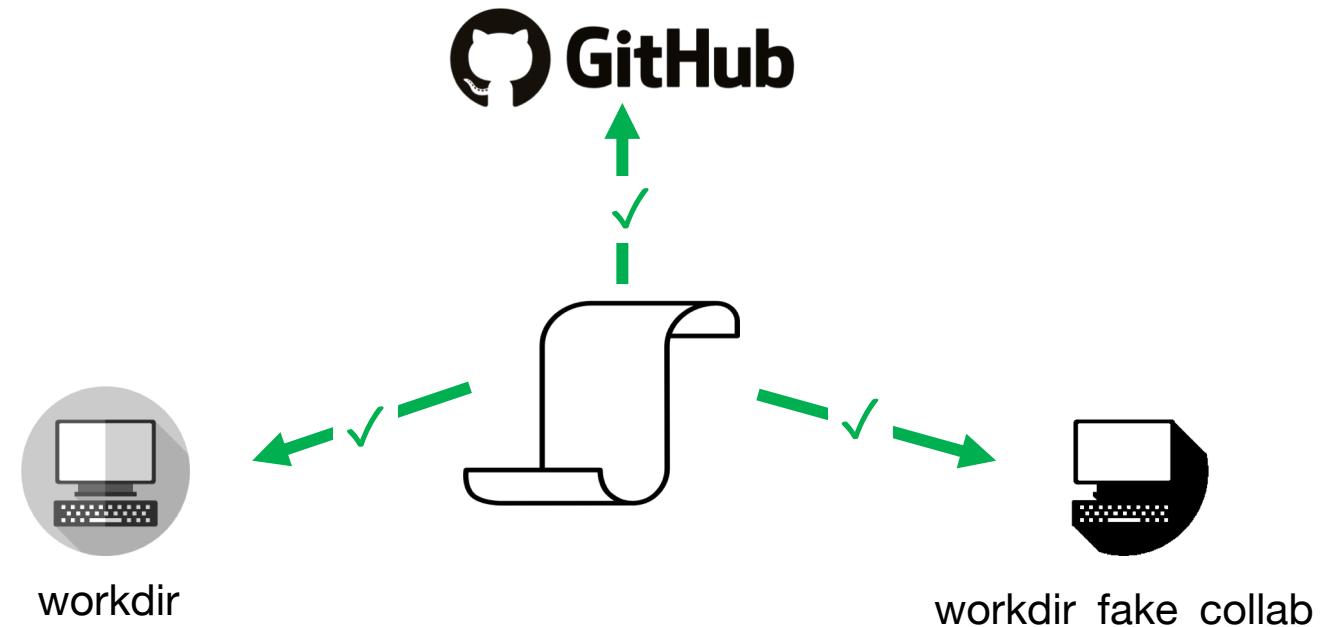


9. Collaborating

> git fetch & git merge

- If these are changes you want in your workdir, merge the changes

> git merge

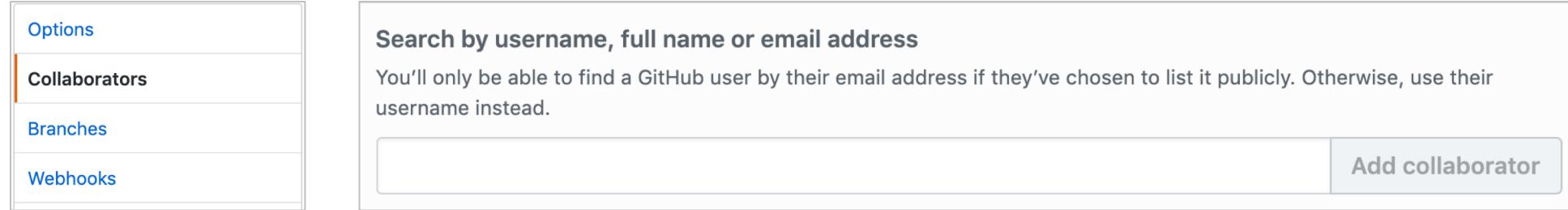


You have now fetched and merged changes from different people (or yourself on a different computer/folder)!

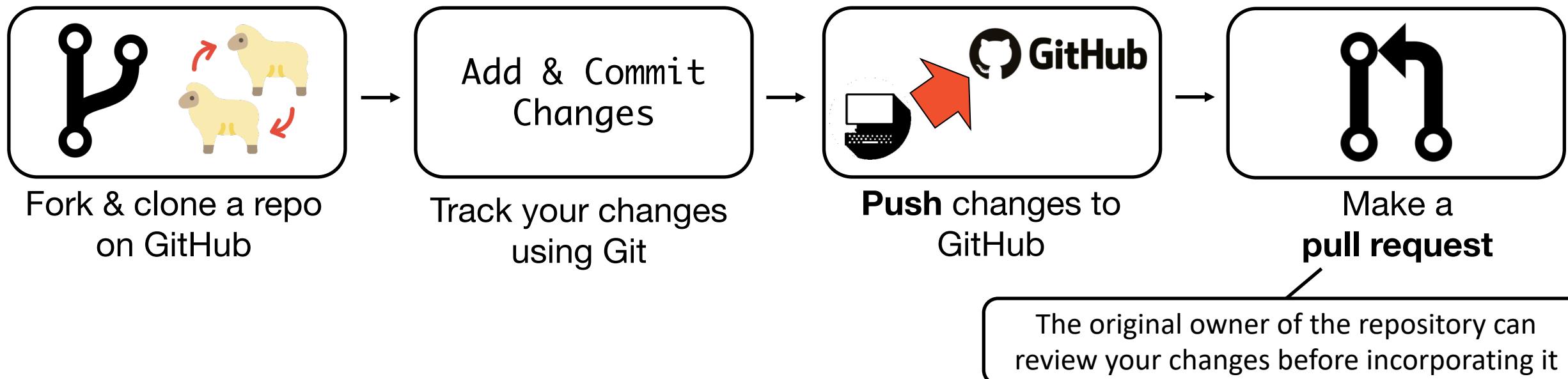
9. Collaborating

BONUS: Truly Collaborating on GitHub

1) Add collaborators in your GitHub repository



2) Using pull request (advance)



Email settings in real life

Setup the no-reply email provided by GitHub (in email setting)

```
> git config --global user.email "shadywhale@users.noreply.github.com"
```

Check the following Setting on GitHub

Block command line pushes that expose my email

If you push commits that use a private email as your author email we will block the push and warn you about exposing your private email.

Advanced Git Topics

The following topics are covered using the bookdown

<https://gitbookdown.site>

- Resolving conflicts
- Reverting to a previous commit
- Branching
- Stashing

Other resources

- GitHub Education Pack
 - <https://education.github.com/>
 - Pro account for students, academic researchers (i.e., postdoc) & faculties

- GitLab
 - <https://gitlab.com>



wiglab Group ID: 846147 | [Leave group](#)

Wig Lab (UTD)

Subgroups and projects Shared projects Archived projects

Project	Description
surface_mapping	
motion_processing	Wiglab motion processing pipeline
preprocessing	Wiglab MRI preprocessing pipeline
modules	Lab-wise modules (e.g., golgi_prepro_modules, freesurfer)
CNL_setup	Bash setup profile for WigLab

Acknowledgement

