



# Overview



## Introduction to Git and Version Control

2020-11-18

### └ Overview

- Why Git? I'll tell you what my motivations are, but what are your motivations for being here?

#### 1 Why Git?

#### 2 What is Git

#### 3 Terminal Talk

#### 4 Git basics

- Local code
- Nonlocal repos / github

#### 5 Working alone

1 Why Git?
2 What is Git
3 Terminal Talk
4 Git basics
• Local code
• Nonlocal repos / github
5 Working alone

- Version control
- Easily compare and merge changes between any version
- Organize your work items



# Why Git?

- Version control
- Easily compare and merge changes between any version
- Organize your work items



Before

After

## Introduction to Git and Version Control

### └ Why Git?

#### └ Why Git?

Imagine I asked you to remove the red sharpie marker from the left hand side?

- Difficulty finding it
- Could dive right in, but might get poked by lot of sharp things on the way in
- Or you could dump everything out and start all over

# Why Git?



A photograph showing a massive, chaotic pile of discarded wooden chairs stacked between two buildings. A small figure of a person stands at the bottom left, highlighting the enormous scale of the waste.



REPLAY

Introduction to Git and Version Control

└ Why Git?

2020-11-18

Imagine I asked you to remove this chair. What difficulties would you face?

- How to access it safely
- Can't remove it without fearing everything will fall

Why Git?



Christopher Buckley (OIST)

Introduction to Git and Version Control

November 19, 2020

4 / 21

Why Git?

# Traditional vs. Git Versioning

- What changed when
- Not limited to file name length to inform user of changes

```
christopher@christopher-ThinkPad-W541:~/git/mythesis$ ls -gtr
total 4
-rw-rw-r-- 1 christopher 51 Nov 17 18:22 thesis
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_v1
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_v2
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_v3
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_v4
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_final
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_final1
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_final2
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_final3
-rw-rw-r-- 1 christopher 0 Nov 18 12:07 thesis_finalfinal
christopher@christopher-ThinkPad-W541:~/git/mythesis$
```

```
christopher@christopher-ThinkPad-W541:~/git/mythesis$ git log --reverse
commit 839a47e257310df071ac8290cdefc04a60b86944 (master)
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date: Tue Nov 17 18:14:52 2020 +0900

    Added empty thesis template

commit e017bf79743fa7724d4c35f430e1e78064823e1a
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date: Tue Nov 17 18:19:14 2020 +0900

    Added initial title

commit 750455880517cbdd5db25054e0e9815ed67da185
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date: Tue Nov 17 18:20:38 2020 +0900

    Added initial summary section

commit a83135a00c134372a7973a879e2431008b5a466
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date: Tue Nov 17 18:21:09 2020 +0900

    Added initial bulk of main body section

commit 98c17b7472ef14bd75804d4ddb990de92f211ba
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date: Tue Nov 17 18:21:31 2020 +0900

    Added initial conclusions

commit aa3034ff24084d3f95e37ae222f265da07d3590
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date: Tue Nov 17 18:22:03 2020 +0900

    Changed conclusions to reflect new findings on Mars

commit d918fbfe43cf474a34c6cde86ccf845f63e9cb4 (HEAD -> new_versioning)
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date: Tue Nov 17 18:22:31 2020 +0900

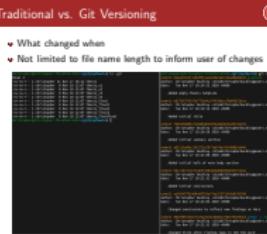
    Changed title after finding typo in teh the word
```

## Introduction to Git and Version Control

### └ What is Git

#### └ Traditional vs. Git Versioning

2020-11-18

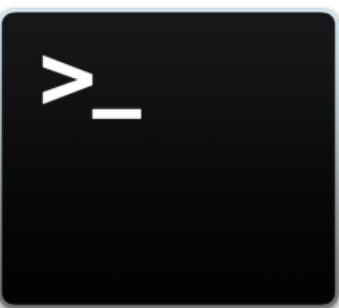


Traditional vs. Git Versioning

- What changed when
- Not limited to file name length to inform user of changes



- There are multiple GUIs available for Git, such as one from GitHub called the **GitHub Desktop**. We will not be using this for religious perfectly scientific reasons.
- These reasons primarily revolve around flexibility and improved understanding of the Git tools.
- Everything we do will be usable on Deigo.
- The **Pro Git** book is available online at [git-scm.com/book](http://git-scm.com/book)
- There is a cheatsheet for Git available here: <https://www.git-tower.com/learn/cheatsheets/git>



2020-11-18

## Introduction to Git and Version Control

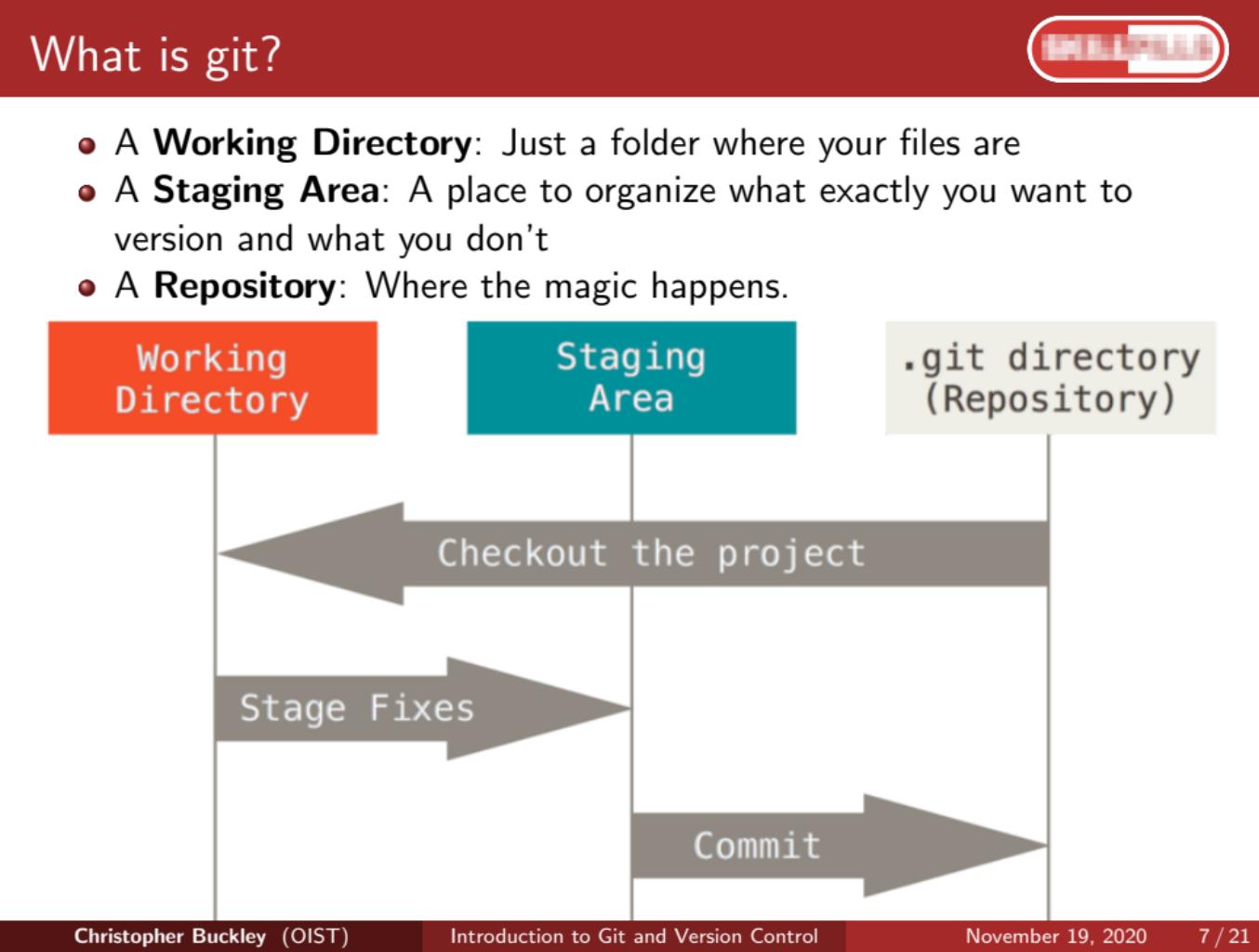
### └ Terminal Talk

#### └ Terminal Talk

- I personally struggled with the terminal interface at first because most of the man pages use so much vocab I don't know to explain terms I don't know. Hopefully by the end of this mini-course you'll have the basic vocab down so you can help yourselves more efficiently going forward.

- There are multiple GUIs available for Git, such as one from GitHub called the **GitHub Desktop**. We will not be using this for religious perfectly scientific reasons.
- These reasons primarily revolve around flexibility and improved understanding of the Git tools.
- Everything we do will be usable on Deigo.
- The **Pro Git** book is available online at [git-scm.com/book](http://git-scm.com/book)
- There is a cheatsheet for Git available here: <https://www.git-tower.com/learn/cheatsheets/git>





Introduction to Git and Version Control

└ Terminal Talk

2020-11-18

└ What is git?

What is git?

- A **Working Directory**: Just a folder where your files are
- A **Staging Area**: A place to organize what exactly you want to version and what you don't
- A **Repository**: Where the magic happens.

This chart provides a visual summary of the Git components and their interaction:

- Working Directory** (red box): Contains the files of the project.
- Staging Area** (blue box): A temporary space where changes are prepared before being committed to the repository.
- .git directory (Repository)** (gray box): Stores the history of changes and metadata.

Arrows show the flow of operations:

- From **Working Directory** to **Staging Area**: **Checkout the project**
- From **Staging Area** to **.git directory**: **Stage Fixes**
- From **.git directory** back to **Working Directory**: **Commit**

This chart might not make much sense now, but I hope it will before the end of these slides

I want to first focus on the bottom two arrows here. These are the tools we use to save exactly what changed between versions of our project.

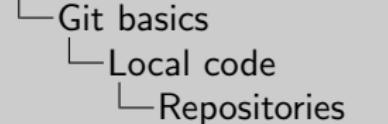
# Repositories



- A **repository** is a container for both your project data and all the items that allow interactions with git commands.
  - There are many sites to host your repository on (github, bitbucket), including your own local machine.
  - All of the essential parts of your repository can be found in the `.git` directory
  - GitHub (a website hosting Git repositories)  $\neq$  Git (a set of tools for creating and managing those repositories).



## Introduction to Git and Version Control



2020-11-18

### Repositories

- A **repository** is a container for both your project data and all the items that allow interactions with git commands.
  - There are many sites to host your repository on (github, bitbucket), including your own local machine.
  - All of the essential parts of your repository can be found in the `.git` directory
  - GitHub (a website hosting Git repositories)  $\neq$  Git (a set of tools for creating and managing those repositories).



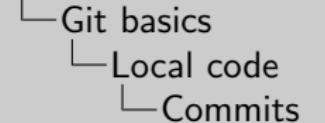
# Commits

- Conceptually similar to "versions"
- The more effort you put into crafting these using the **staging area** the more helpful they are in the future.

COMMENT	DATE
○ CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○ ENABLED CONFIG FILE PARSING	9 HOURS AGO
○ MISC BUGFIxes	5 HOURS AGO
○ CODE ADDITIONS/EDITS	4 HOURS AGO
○ MORE CODE	4 HOURS AGO
○ HERE HAVE CODE	4 HOURS AGO
○ AAAAAAAA	3 HOURS AGO
○ ADKFJSLKDFJSOKLFJ	3 HOURS AGO
○ MY HANDS ARE TYPING WORDS	2 HOURS AGO
○ HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

## Introduction to Git and Version Control



2020-11-18

Show lego photos now

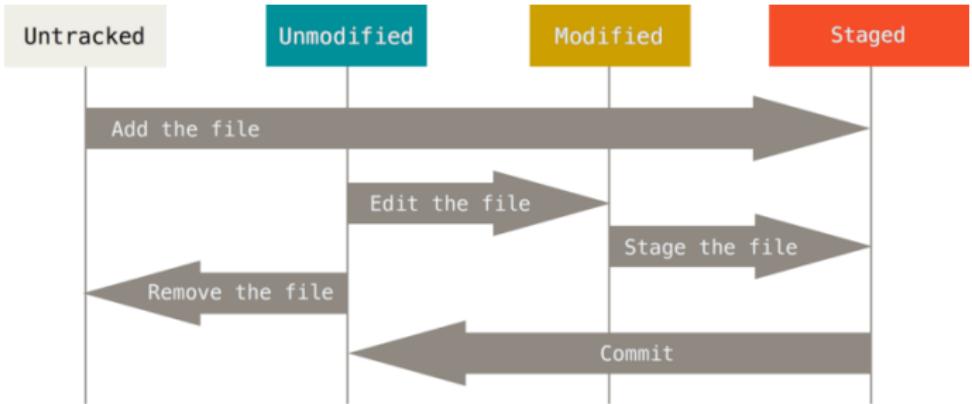
Comments		Date
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIxes	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSOKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Comments		Date
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIxes	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSOKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

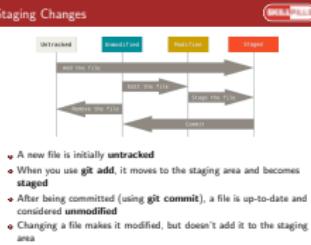
# Staging Changes



- A new file is initially **untracked**
- When you use **git add**, it moves to the staging area and becomes **staged**
- After being committed (using **git commit**), a file is up-to-date and considered **unmodified**
- Changing a file makes it modified, but doesn't add it to the staging area

Introduction to Git and Version Control  
└ Git basics  
  └ Local code  
    └ Staging Changes

2020-11-18



Go through car repo example, e.g. adding wheels and doors, but committing only one or the other.

# Currrating the Stage before Committing



- Check what is on the stage with **git status**. Anything in **green** is staged.
- If you wish to unstage all changes, simply type **git reset**. This will remove everything from the stage, but keep your working directory untouched.
- **git reset** will work for individual files as well

---

```
git reset <file>
```

---



2020-11-18

## Introduction to Git and Version Control

- └ Git basics
  - └ Local code
    - └ Currrating the Stage before Committing

Currrating the Stage before Committing

- Check what is on the stage with **git status**. Anything in **green** is staged.
- If you wish to unstage all changes, simply type **git reset**. This will remove everything from the stage, but keep your working directory untouched.
- **git reset** will work for individual files as well

```
git reset <file>
```



# Try out the Stage



Introduction to Git and Version Control

- └ Git basics
- └ Local code
- └ Try out the Stage

2020-11-18

Try out the Stage

EXERCISE

- ① Stage a commit
- ② Unstage the commit
- ③ Make a commit
- ④ Undo the commit (**DON'T DO THIS AFTER YOU PUSH!!!!!!11111!!!11!!**)

## EXERCISE

- ① Stage a commit
- ② Unstage the commit
- ③ Make a commit
- ④ Undo the commit (**DON'T DO THIS AFTER YOU PUSH!!!!!!11111!!!11!!**)



## EXERCISE

- ① Stage a commit
- ② Unstage the commit
- ③ Make a commit
- ④ Undo the commit (**DON'T DO THIS AFTER YOU PUSH!!!!!!11111!!!11!!**)

## Introduction to Git and Version Control

- └ Git basics
  - └ Local code
    - └ Quick Exercise

2020-11-18

### EXERCISE

- ① Stage a commit
- ② Unstage the commit
- ③ Make a commit
- ④ Undo the commit (**DON'T DO THIS AFTER YOU PUSH!!!!!!11111!!!11!!**)

# The local repo



Let's **git** started.

- To initialize a git repository, simply type **git init** in a directory (preferably empty for now)
- This creates a folder **.git/**, where all your repository information is held.
- Git tracks **commits**. Check these commits with **git log**.
- **git status** checks any changes since the last commit.
- **git add** adds new files.
- **git commit** commits anything in the *staging area* - git status shows these files in **green** by default.



## Introduction to Git and Version Control

- └ Git basics
  - └ Local code
    - └ The local repo

2020-11-18

- Let's **git** started.
- To initialize a git repository, simply type **git init** in a directory (preferably empty for now)
  - This creates a folder **.git/**, where all your repository information is held.
  - Git tracks **commits**. Check these commits with **git log**.
  - **git status** checks any changes since the last commit.
  - **git add** adds new files.
  - **git commit** commits anything in the *staging area* - git status shows these files in **green** by default.



# Quick Exercise



## EXERCISE

- ① Open a terminal
- ② Create a new directory and run **git init**
- ③ Create a file and run **git status**
- ④ Use a combination of **git add** and **git commit** to add a new file to the git repository.
- ⑤ Check the **git log**.

## Introduction to Git and Version Control

- └ Git basics
  - └ Local code
    - └ Quick Exercise

2020-11-18

### EXERCISE

- ① Open a terminal
- ② Create a new directory and run **git init**
- ③ Create a file and run **git status**
- ④ Use a combination of **git add** and **git commit** to add a new file to the git repository.
- ⑤ Check the **git log**.

# Ignorance is bliss

2020-11-18

- Keep your repository clean! Do your best to commit as few images and data files as possible!
- You can do this by ignoring certain file extensions in a `.gitignore` file.
- Great templates for projects of many types found at <https://github.com/github/gitignore>

---

```
# Example gitignore configuration
*.log
*.tar
*.gz
*.exe
*.dat
*.lvp
```

## Introduction to Git and Version Control

- Git basics
  - Local code
    - Ignorance is bliss

Ignorance is bliss

- Keep your repository clean! Do your best to commit as few images and data files as possible!
- You can do this by ignoring certain file extensions in a `.gitignore` file.
- Great templates for projects of many types found at <https://github.com/github/gitignore>

---

# Example gitignore configuration

```
*.log
*.tar
*.gz
*.exe
*.dat
*.lvp
```

Christopher Buckley (OIST)

Introduction to Git and Version Control

November 19, 2020

16 / 21



## EXERCISE

- ① Touch multiple files with various extensions, one of which should be **.dat**.
- ② Ignore the **.dat** file, but commit all the others.
- ③ Be sure to write a clear message describing what you did.
- ④ Check the **git log**

## Introduction to Git and Version Control

- └ Git basics
  - └ Local code
    - └ Quick Exercise

2020-11-18

EXERCISE

- Touch multiple files with various extensions, one of which should be **.dat**.
- Ignore the **.dat** file, but commit all the others.
- Be sure to write a clear message describing what you did.
- Check the **git log**

# git with it!



Now we move to the fun\* stuff: working with **online repositories**.

- For this, we will be using **github**.
- We'll begin by creating a GitHub repository using the website.
  - If we're working on a project that's already hosted on a remote Git server, we can skip this step.
- Next, we use **git clone** to download a copy.
- From here, you can do the following:
  - **git push** to push any changes you may have to the online repository.
  - **git pull** to take any changes from the repository.

\*Here, the word *fun* is subject to interpretation.



2020-11-18

Introduction to Git and Version Control  
└ Git basics  
  └ Nonlocal repos / github  
    └ **git with it!**

git with it!  
Now we move to the fun\* stuff: working with online repositories.

- For this, we will be using **github**.
- We'll begin by creating a GitHub repository using the website.
  - If we're working on a project that's already hosted on a remote Git server, we can skip this step.
- Next, we use **git clone** to download a copy.
- From here, you can do the following:
  - **git push** to push any changes you may have to the online repository.
  - **git pull** to take any changes from the repository.

\*Here, the word *fun* is subject to interpretation.





## EXERCISE

- ① Create a new GitHub repository using a browser.
- ② Clone the new repository\* to our local disk:

```
git clone git@github.com:oist/skillpill-git.git
```

or

```
git clone https://github.com/oist/skillpill-git.git
```

- ③ Make some simple commits and test the process of **pushing** and (with the help of a partner) **pulling** stuff from that repo.

\*The examples here show cloning the SkillPill Git repository - replace the links as appropriate!

## Introduction to Git and Version Control

- └ Git basics
  - └ Nonlocal repos / github
    - └ Quick Exercise

2020-11-18

EXERCISE

- ① Create a new GitHub repository using a browser.
- ② Clone the new repository\* to our local disk.

```
git clone git@github.com:oist/skillpill-git.git
```

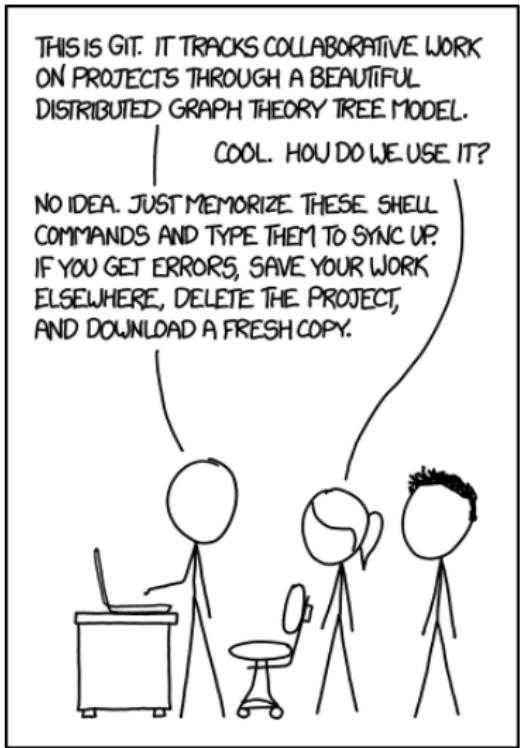
or

```
git clone https://github.com/oist/skillpill-git.git
```
- ③ Make some simple commits and test the process of **pushing** and (with the help of a partner) **pulling** stuff from that repo.

\*The examples here show cloning the SkillPill Git repository - replace the links as appropriate!

# What it will feel like...

- git is not intuitive to start with, but it's a powerful tool for storing and restoring history, and working collaboratively with other people.
- The more you use it, the more you will like it. Think Stockholm syndrome.
- Operations that you use frequently will become easy.
- Operations you use infrequently, you can Google!



## Introduction to Git and Version Control

### Git basics

### Nonlocal repos / github

### What it will feel like...

2020-11-18

What it will feel like...

- git is not intuitive to start with, but it's a powerful tool for storing and restoring history, and working collaboratively with other people.
- The more you use it, the more you will like it. Think Stockholm syndrome.
- Operations that you use frequently will become easy.
- Operations you use infrequently, you can Google!



# Checking out your versions



We now know how to work with both local and online repositories, but what about using different versions?

- **git checkout** allows you to view the repository at any commit (found with **git log**).
- You may also checkout specific files like so:

---

```
git checkout a1e8fb5 hello.py
```

---

- Note that the most recent commit is **HEAD** and the one just before that is **HEAD~1**
- This command will be used later, so keep it in mind!

## Introduction to Git and Version Control

- └ Git basics
  - └ Nonlocal repos / github
    - └ Checking out your versions

2020-11-18

Checking out your versions

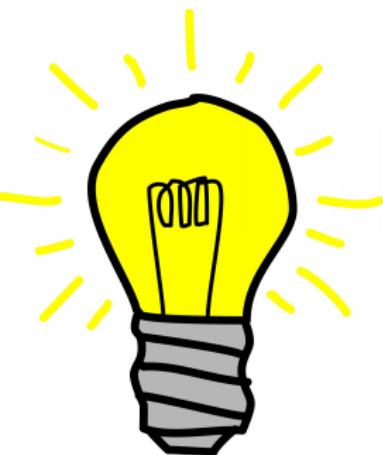
We now know how to work with both local and online repositories, but what about using different versions?

- **git checkout** allows you to view the repository at any commit (found with **git log**).
- You may also checkout specific files like so:  

```
git checkout a1e8fb5 hello.py
```
- Note that the most recent commit is **HEAD** and the one just before that is **HEAD~1**
- This command will be used later, so keep it in mind!



- git is weird. It's not intuitive, but it's the best way to collaborate with people on open projects.
- It's also great even if you don't collaborate!
- Whenever you are using git, think about other people and how they will perceive your comments. **Would you be able to understand your own cryptic commit messages?**
- You will make mistakes. Don't worry about it. Your entire history is backed up already. Learn from your mistakes and don't make them again!
- Read error messages carefully - they can be useful/informative/instructive.



### └ Working alone

#### └ Final Comments

2020-11-18

- git is weird. It's not intuitive, but it's the best way to collaborate with people on open projects.
- It's also great even if you don't collaborate!
- Whenever you are using git, think about other people and how they will perceive your comments. **Would you be able to understand your own cryptic commit messages?**
- You will make mistakes. Don't worry about it. Your entire history is backed up already. Learn from your mistakes and don't make them again!
- Read error messages carefully - they can be useful/informative/instructive.

