Skill Pill: Introduction to Git and Version Control

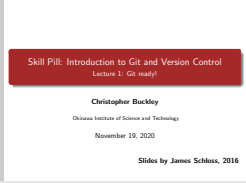
Lecture 1: Git ready!

Christopher Buckley

Okinawa Institute of Science and Technology

November 19, 2020

Slides by James Schloss, 2016



Overview

Overview
1. Why Git?
2. What is Version Control
3. Terminal Talk
4. Git basics
   - Local code
   - Nonlocal repos / github
5. Working alone

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

Why Git?

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

Before After

Why Git?
- Version control
- Easily compare and merge changes between any version
- Organize your work items
Before After

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?

Why Git?

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

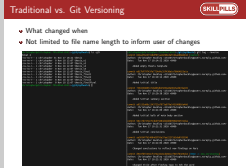
- How to access it safely
- Can't remove it without fearing everything will fall

Traditional vs. Git Versioning

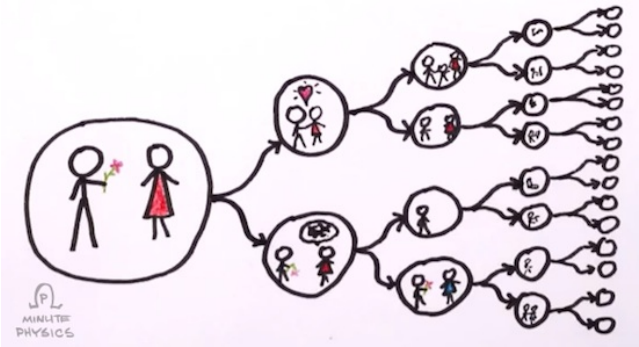
- What changed when
- Not limited to file name length to inform user of changes

```
christopher@christopher-ThinkPad-W541:~/git/nythesis$ 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/nythesis$
```

```
christopher@christopher-ThinkPad-W541:~/git/nythesis$ git log --reverse
commit 839a476257310dfd71ac8290cdefc64a60b8d944 (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 75045580517cbdd5db25054e0e9815ed67da185
Author: Christopher Buckley <15166572+topherbuckley@users.noreply.github.com>
Date:   Tue Nov 17 18:20:38 2020 +0900

    Added initial summary section

commit a83135a08c1343723a7973a879e2431008b5a466
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 aa3034ffb24084d3f95e37ae222f265da87d3590
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 d918fbfe436cf474a34c6cde86ccf845f63e9cb4 (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
```

Version Control



- Version control is a method that allows you to control different versions of things.
- Version control stores history and allows restoration to specific points in that history.

Version Control
- Version control is a method that allows you to control different versions of things.
- Version control stores history and allows restoration to specific points in that history.

Terminal Talk

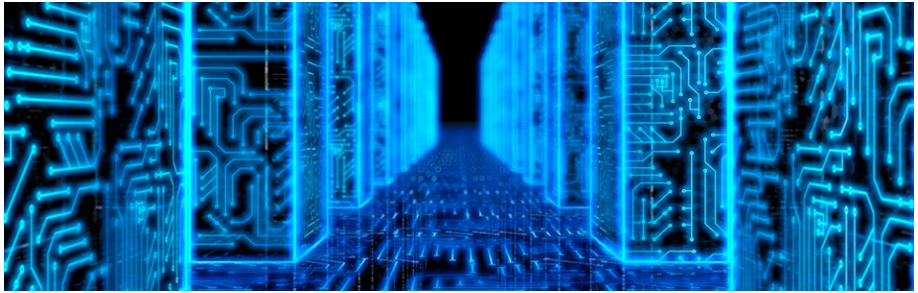
- There are multiple GUIs available for Git, such as one from GitHub called the **GitHub Desktop**. We will not be using this for ~~religious~~ perfectly scientific reasons.
- These reasons primarily revolve around flexibility and improved understanding of the Git tools.
- Everything we do will be usable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

Terminal Talk
- 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
- There is a cheatsheet for Git available here: https://www.git-tower.com/learn/cheat-sheets/git
>_

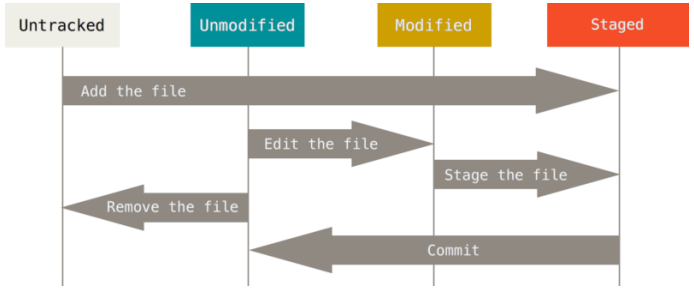
- I personally struggled with the terminal interface at first because most of the man pages use so much vocab I don't know to explain terms I don't know. Hopefully by the end of this mini-course you'll have the basic vocab down so you can help yourselves more efficiently going forward.

Your first repository

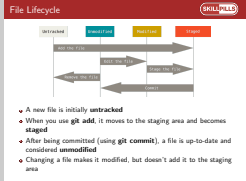
- A **repository** is a place to store code.
  - There are many sites to host your repository on (github, bitbucket), including your own local machine.
  - All of the essential parts of your repository can be found in the **.git** directory
  - GitHub (a website hosting Git repositories) $\neq$ Git (a set of tools for creating and managing those repositories).



File Lifecycle



- 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



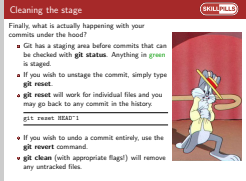
Cleaning the stage

Finally, what is actually happening with your commits under the hood?

- Git has a staging area before commits that can be checked with **git status**. Anything in green is staged.
- If you wish to unstage the commit, simply type **git reset**.
- **git reset** will work for individual files and you may go back to any commit in the history.

```
git reset HEAD~1
```

- If you wish to undo a commit entirely, use the **git revert** command.
- **git clean** (with appropriate flags!) will remove any untracked files.



Quick Exercise

SKILLPILLS

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!!**)

Quick Exercise
SKILLPILLS
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.

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.

Quick Exercise

EXERCISE

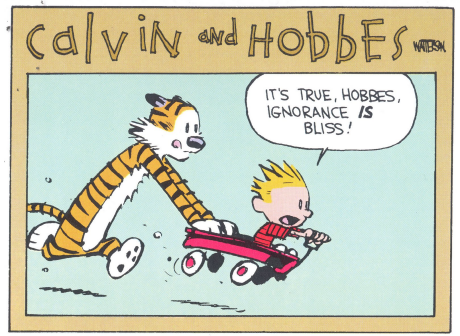
1. Open a terminal
2. Create a new directory and run **git init**
3. Create a file and run **git status**
4. Use a combination of **git add** and **git commit** to add a new file to the git repository.
5. Check the **git log**.

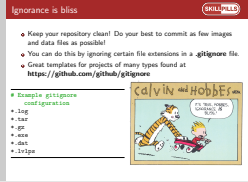
Quick Exercise
EXERCISE
1. Open a terminal
2. Create a new directory and run **git init**
3. Create a file and run **git status**
4. Use a combination of **git add** and **git commit** to add a new file to the git repository.
5. Check the **git log**.

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
*.lvlps
```





Quick Exercise

SKILLPILLS

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**

Quick Exercise
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

git with it!

SKILLPILLS

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.

git with it!
SKILLPILLS
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.

Quick Exercise

EXERCISE

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

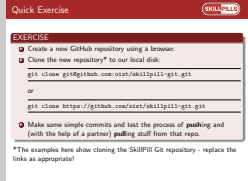
```
git clone git@github.com:oist/skillpill-git.git
```

or

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

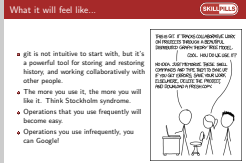
3. Make some simple commits and test the process of **push**ing and (with the help of a partner) **pull**ing stuff from that repo.

*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!



Write clear commit messages!

Write clear commit messages!
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
ADKFJSLKDFJSDKLFJ 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.

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:

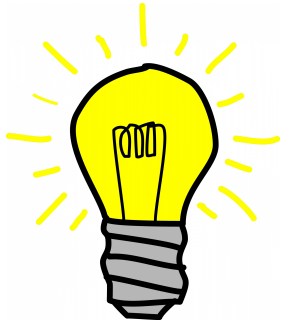
```
git checkout a1e8fb5 hello.py
```

- 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!

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!

Final Comments

- 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.



Final Comments
- 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.