Skill Pill: Introduction to Git and Version Control

Lecture 1: Git ready!

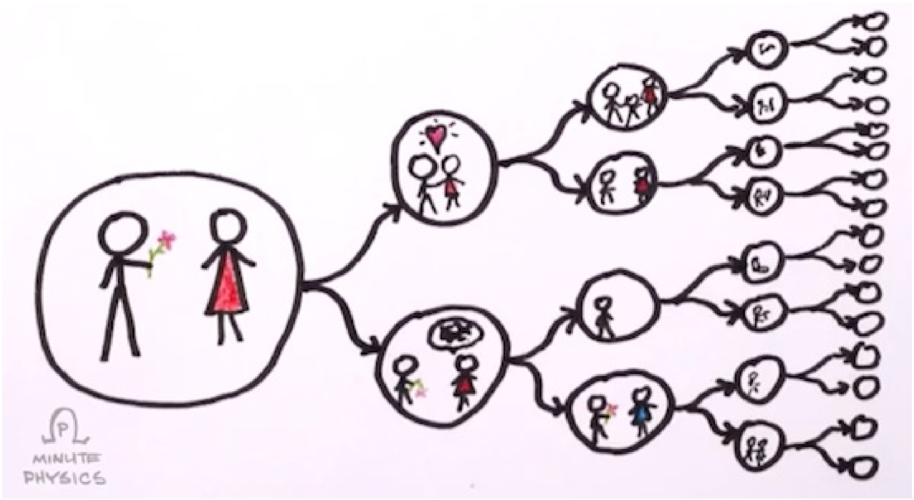
James Schloss

Okinawa Institute of Science and Technology
james.schloss@oist.jp

June 1, 2016

Overview

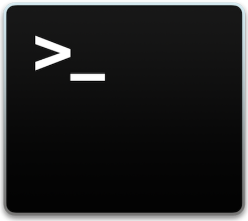
Version Control



- Version control is a method that allows you to control different versions of things (Not necessarily universes).

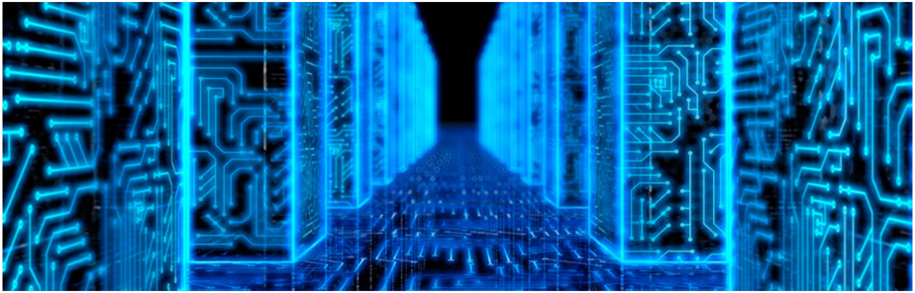
Terminal Talk

- We had a terminal skill pill and I have included the cheatsheet from that.
- There is a GUI downloadable from GitHub called the **GitHub Desktop**. We will not be using this for religious reasons.
- Everything we do will be usable on Sango.
- We will be using a cheatsheet from here: **https://www.git-tower.com/learn/cheat-sheets/git**



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 **.git** directory



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 git 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 git status shows in green.

Quick Exercise

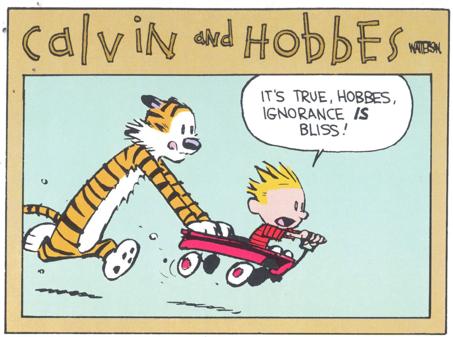
EXERCISE

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.

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



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!

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

- For this, we will be using **github**.
- To use an online repository, we need to synchronize our local machine with the master repository held elsewhere. This is done with the **clone** command.
- 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

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

Quick Exercise

EXCERCISE

1. Clone our skillpill repository (or a similar repository):

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

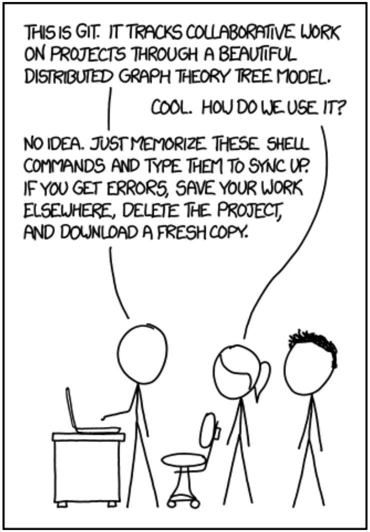
or

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

2. Working with a small group, make commits and push and pull stuff from that repo.

What it will feel like...

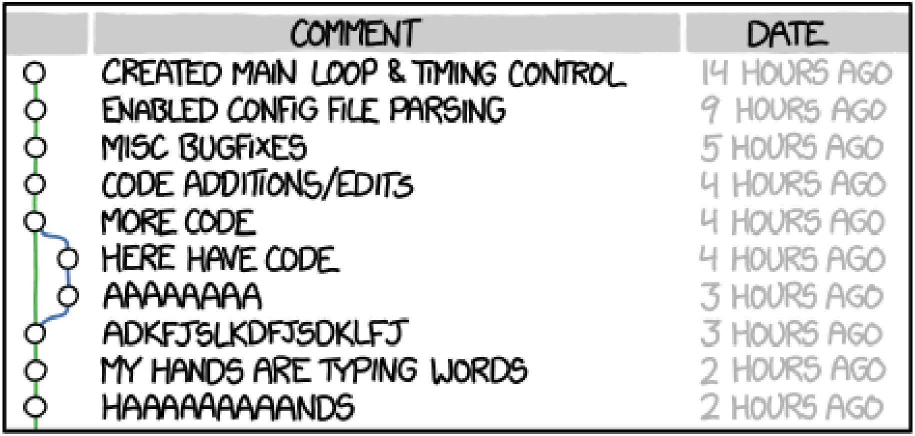
- git is not intuitive to start with, but it's the best way to work collaboratively with other people.
- The more you use it, the more you will like it. Think Stockholm syndrome.


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

A fear of commitment

Coders and Relationships:
5 Signs He'll Never Commit

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
○	HAAAAAAAANDS	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 old 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!

PRODUTOR DE GANHADORES DE MAIS DE US$ 7.000.000,00
CHECK HIM OUT
HENNESSY X CHECK HER TWICE (FIRST DOWN DASH)
Produziu ganhadores de mais de US$ 1.000.000,00 nos ultimos 4 anos seguidos: 2012/2011/2010/2009
NO BRASIL PRODUZIU UOMO VERDE:
COBERTURAS 2013
10 parcelas R$ 700,00
Informações: (11) 3226.8952 | 3868.5257

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** will remove any untracked files.

Quick Exercise

EXERCISE

1. Stage a commit
2. Unstage the commit
3. Make a commit
4. Undo the commit

Final Comments

- git is weird. It's not intuitive, but it's the best way to collaborate with people on open projects.
- 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!
- Listen to git. It's smarter than you.