Git
(and some other stuff)
Recapping last week
Any lingering administrative issues?
Recapping last week

Any lingering administrative issues?

The readings

- Biculturalism
- Command-line "bullshittery"
What's version control?
What's version control?
Why is it important?
And how might we build a version control system?
What's version control?

Why is it important?

And how might we build a version control system?

Why doesn't everyone use version control for everything?
Git-ing started (I'm so sorry)

- first — `init`
- making commits — `add`, `commit`
- viewing work/history — `status`, `log`
Git-ing started (I'm so sorry)

- first — `init`
- making commits — `add, commit`
- viewing work/history — `status, log`

Let's peek under the hood

- History is a linked list
- What's `master, HEAD`?
- Where's the old versions?
Git-ing started (I'm so sorry)

- first — `init`
- making commits — `add, commit`
- viewing work/history — `status, log`

Let's peek under the hood

- History is a linked list
- What's `master, HEAD`?
- Where's the old versions?

I'm lazy, let's make `git st == git status`

- `git config --global --edit`
  - (Fancy way of saying `editor ~/.gitconfig`)
  - (Fancy way of saying `editor /home/ppannuto/.gitconfig`)
  - (Fancy way of saying `[nano/vim/emacs/...] /home/ppannuto/.gitconfig`)

```
[alias]
  st = status
```
Using git locally

Working Directory

Staging Area

Repository

Local Machine

- `git add my_file`
- `git reset my_file`
- `git reset --soft HEAD`
- `git commit`
- `git commit my_file`
- `git reset HEAD^`
Collaborating with git

First, we have to talk just a little about branches
Collaborating with git

First, we have to talk just a little about branches

```plaintext
Working Directory

- git add my_file
- git reset my_file
- git reset --soft HEAD^ (pull = "fetch + merge")
- git commit

Local Machine

Staging Area

Repository

Remote ("bare repository")

"The Cloud"
(See the cloud with git remote -v)

- git push
- git pull
- git fetch
- git merge origin/master
- git reset HEAD^
Why is git cool?

Let's debug a real-world problem

Phony is a ruby library for pretty-printing phone numbers. It's used by a bunch of websites (AirBnB, ZenDesk, etc)

1. git clone https://github.com/floere/phony

```ruby
# US phone number
ruby -e 'require "./lib/phony"' -e 'puts Phony.format("17345551212")'
+ 1 (734) 555-1212

# French phone number
ruby -e 'require "./lib/phony"' -e 'puts Phony.format("33123456789")'
+33 1 23 45 67 89

# Emergency phone number in the UK
ruby -e 'require "./lib/phony"' -e 'puts Phony.format("999")'
```

Example borrowed from http://www.paulboxley.com/blog/2015/01/a-simple-example-of-git-bisect-run-with-ruby
ruby -e 'require "./lib/phony"' -e 'puts Phony.format("999")'

- Go back in time to when you know it worked
  - `git checkout v1.9.0`
Go back in time to when you know it worked

- git checkout v1.9.0

Search history for the problem
ruby -e 'require "./lib/phony"' -e 'puts Phony.format("999")'

- Go back in time to when you know it worked
  - git checkout v1.9.0
- Search history for the problem
  - The `git bisect` command will help keep track of progress
    - git bisect start
    - git bisect good
    - git checkout master
    - git describe
    - git bisect bad
    - (magic)
    - git describe
Ruby

- `ruby -e 'require "./lib/phony"' -e 'puts Phony.format("999")'

- Go back in time to when you know it worked
  - `git checkout v1.9.0`

- Search history for the problem
  - The `git bisect` command will help keep track of progress
    - `git bisect start`
    - `git bisect good`
    - `git checkout master`
    - `git describe`
    - `git bisect bad`
    - `(magic)`
    - `git describe`

- But I'm still lazy
- Go back in time to when you know it worked
  - `git checkout v1.9.0`
- Search history for the problem
  - The `git bisect` command will help keep track of progress
  - `git bisect start`
  - `git bisect good`
  - `git checkout master`
  - `git describe`
  - `git bisect bad`
  - `(magic)`
  - `git describe`
- But I'm *still* lazy
  - `git bisect run ruby -e 'require "./lib/phony"' -e 'puts Phony.normalize("+999")'`
- Go back in time to when you know it worked
  - `git checkout v1.9.0`

- Search history for the problem
  - The `git bisect` command will help keep track of progress
    - `git bisect start`
    - `git bisect good`
    - `git checkout master`
    - `git describe`
    - `git bisect bad`
    - (magic)
    - `git describe`

- But I'm still lazy
  - `git bisect run ruby -e 'require "./lib/phony"' -e 'puts Phony.format("999")'`

- So what changed?
  - `git log -p` — or — [github.com/floere/phony/commit/9b4234](https://github.com/floere/phony/commit/9b4234)....
  - [https://github.com/floere/phony/issues/219](https://github.com/floere/phony/issues/219)
Closing remarks

You must use git for something non-trivial this term

- You *should* use version control for all your class projects
  - And I don't mean folder-copying
- Not in any programming classes?
  - Try using it for other homework or a side project!
- Deadline for this is Week 5 — Git II (Oct 05)