http://git-scm.com/
History

- 2002: Linux + BitKeeper
- 2005: BitKeeper bombed
- Linux community + Linus Torvalds + lessons learned
Goals

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)
Basics

● Git thinks of data as a set of snapshots
● Commit = Taking a picture
● Only takes a picture once (Saves space)
Workflow

- Three main states
- Modified
  - You have changed the file but have not yet submitted them to git
- Staged
  - You have marked a modified file in its current version to go into your next commit snapshot
- Committed
  - The data is safely stored in your local database
Local Operations

working directory

staging area

commit

git directory (repository)

checkout the project

stage files
Commands

- `git init`
- `vim stuff`
- `git add`
  - `git add .`
  - `git add -p`
  - `git add myfile`
  - etc...
- `git status`
- `git commit`
  - `git commit -m "Insert message here"`

*For a full list of commands look up some cool cheat sheets.*
Branching and Merging

- Drastic changes, quick fixes
- Branch creates temporary work environment
- Merge easily combines changes with original
  - Conflicts can be tricky, sometimes
Branching and Merging
Commands

- `git checkout -b mybranch`
- `vim stuff`
- `git add .`
- `git commit -m "Change complicated things"`
- `git checkout master`
- `git merge mybranch`
Remote Git

github.com
Distributed Workflow

- Many developers, one shared repository
Distributed Workflow

- Multiple branches
Commands

- git pull
- git checkout -b mybranch
- vim stuff
- git add .
- git commit
- git checkout <develop>
- git pull
- git merge mybranch
- get latest changes
- create branch
- make changes
- stage changes
- commit changes
- switch back
- get latest changes
- merge your changes