Git

What is Git?

Git is one of the best version control tools:

- Collaborate with many users around the Internet
- Keep changes synchronized across machines
- Can manage source, documentation, text files, and more
- A safety net:
  - Roll back mistakes
  - Explore new ideas safely (in a branch)
- Backup work!
- Increasingly, the standard tool for version control
- Reproducible research
- Unprofessional not to use `git`...

Git features

Features that make Git awesome:

- SHA-1 hash:
  - Nigh uniquely identifies a commit across all copies of repo
  - Provides security from tampering
- Blazingly fast merges and diffs because git works on entire file and not incremental diffs
- Relatively painless merges
- Peer to peer:
  - Can work without Internet
  - Distributed
  - Fault tolerant

References

Two great references:

- Pro Git
- git cheatsheet

Plus, old-school help:

```
$ man git
$ git clone --help
```

Components of Git

In order to understand git, you must understand how information flows between git's components:

- *Workspace*: current version of your work
- *Stage*:
  - Temporary container for work before committing to repository
  - Also known as the *Index* and *Cache*
- *Local repository*:
  - Local storage of all versions of your work which has been committed
  - Stored in `.git` directory in root of repo
- *Upstream repository*: remote copy, possibly out of sync with local repo
- *Stash*: 'park' work here when randomized by your boss

See git cheatsheet

Install git

Check if git is installed:

```
$ which git
/usr/local/bin/git
```

If not, install Homebrew package manager and git:

```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Home
$ brew install git
```

Create an account on GitHub

Setup up an account on GitHub:

- To access course materials
- To submit your work
- To collaborate with students and colleagues
- To coordinate access to your repos on multiple machines
- To host your portfolio
- To backup your work

Configure Git

Configure `~/.gitconfig` and `~/.gitignore_global` for your preferences. At a minimum, set the following:

```
$ git config --global user.name "Eddy Merckx"
$ git config --global user.email eddy@merckx.com
$ git config --global core.editor vim
$ git config --global core.excludesfile ~/.gitignore_global
```

You might prefer a friendlier editor, such as Atom, nano, or pico

ONLY DO THIS ON YOUR OWN MACHINE

git clone *repo*

Create a local copy of the upstream repo:

```
$ git clone https://github.com/joe_student/awesome.git
```

or

```
$ git clone git@github.com:joe_student/awesome.git
```

Git workflow

Stage changes as you work and commit them to the local repo when you are ready:

```
$ atom genius.py  # Implement your great ideas
$ git status  # Check state of workspace and cache
$ git add genius.py  # Stage genius.py to commit to repo
$ git status
$ git commit -m 'Implement orbital mind control laser'
$ git push
```

Caveats

Do not put the following under version control:

- Large files: data, images, binary, .doc, .xls, .pdf, ...
- Derived files: i.e., files built from source code, markdown, LaTeX, ...

Commit message should:

- Use imperative mood to state why you made the change
- Reference JIRA item or bug
- No need to say what you did
- Be nice to your future self!

Examining changes

You have several tools to compare versions:

- Use `tig` (Install via `brew install tig`)
- `git diff 16d6758 HEAD^^^`
- `git diff master`
- `git diff --stat`
- `git log`: see manual for details
- `git blame fubar.py` to determine who broke `fubar.py`

Syncing with an upstream repo (1/2)

To copy upstream changes to a local repo:

- Use `git pull`:
  1. Performs `git fetch` to create a local copy
  2. Attempts to merge changes via `git merge`
- Will require manual intervention if there is a merge conflict
- Can optionally specify the repo and branch to pull

Syncing with an upstream repo (2/2)

Use `git push` to copy local changes to an upstream repository:

- Must merge upstream changes before pushing your changes!
- May need to set an upstream tracking branch

```
$ git remote -v
origin  git@github.com:zipfian/bss.git (fetch)
origin  git@github.com:zipfian/bss.git (push)
$ git push
Counting objects: 15, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 7.06 KiB | 0 bytes/s, done.
Total 15 (delta 3), reused 0 (delta 0)
To git@github.com:zipfian/bss.git
   16d6758..aac3438  master -> master
```

Advanced commands

Some helpful commands:

- `git stash`: save changes in workspace so you can work on something else
- `git remote`: create and examine *remotes* to access upstream repositories
- `git init`: create a local repository in current directory
- `git reset`: rollback to previous commit
- `git revert`: revert commit(s)
- `git checkout`:
  - Revert file to version in local repo
  - Switch to a branch (create it if necessary)
- `git branch`: manipulate branches
- `git rebase`:
  - Replay changes from one branch on another
  - Smash small commits into a single commit

Pro tips

To improve your workflow:

- Use ssh and not https
- Use `tig`
- Work in a branch
- Rebase your commits before requesting a code review
- Commit and push regularly so that you can roll back changes if you make a mistake
- Never put keys in your repo; e.g., keys for AWS
- Be paranoid: enable dual-factor authentication

  Keep all repositories in `~/repos` or equivalent