Intro to `git`

CS51 Code Review 1
Sam Green 2017
samuelgreen@college.harvard.edu

Intro and Norms

- About me
- Sam Green
- Junior, Cabot House (2017)
- CS and Statistics concentration
- Former rower. Happily retired.
- Fun fact: I've bicycled across the United States, from SF to DC.
- Email me: samuelgreen@college.harvar.d.edu

- I'll be there for you if you're there for me.
- I expect you to come to section prepared with your questions about last week's problem set and/or lab. I also expect you to have read the next problem set.
- In return, you can expect me to show up to section ready to answer your questions.
- Code review will (on all days but today) be discussion-based. Your participation is key.

git

/git/: noun - an unpleasant or contemptible person.

What is `git`?

- Version Control System (VCS)
- Make and backup checkpoints, revert to old versions, keep a log of your work.
- Terse but concise interface
- Easy to use, hard to use well (welcome to CS51)

Repositories

- git is based on code repositories.
- There are local repositories and remote repositories. One power of git comes from linking the two.
- BitBucket and GitHub are popular remote repository hosts. GitHub is objectively more popular, but we use BitBucket in this course, so we'll teach that.

Creating a repo

1. Create a local repo
2. Create a remote repo
3. Link them

Let's do a demo.

Key command: `git clone <remote> [dest]`

Change it up

- The whole point of `git` is to make and track changes to your code.

- Now that you have a repo, make some changes.

- Suggestions: open a text file, type your name, and save it inside your repository.

A note about commitment:

- “I can’t use git, because I have commitment issues”

Saving, `git` style

- Snapshots of code in git are called commits.
- They mark the current state of your code, assigning a “checkpoint” marker.
- git also allows you to associate messages with your commits.

Key command:

```
git commit [-a] [-m] [message]
```

Like any good relationship…

- Commitment is key.
- Commit frequently. This is save, without autosave.
- Don’t have commitment issues.
- Use commit messages. They will save your life. Be descriptive.

Let’s practice. Commit the changes you made a minute ago.

Tracking

- git *doesn't* automatically track files in our repository directories.
- This is a feature, not a bug. You don't always want *all* your files in the repository.
- We have to declare files explicitly for tracking.

Key commands:

```
git add <file[s]>
```

```
git add .
```

`git`-ing confused?

- Because git is a cli, it's easy to get confused about what's going on.
- git has the answer, though.

Key commands:

```
git status

git log

git show
```

Back it up

- All changes so far have been local.
- We need to send them to the remote repository. Why?
  - Create a remote version in case of crisis.
  - Coordinate between collaborators (like your pset partners!)

Key commands:

```
git push
```

```
git pull
```

“Those are the basics.”

–Sam Green and maybe someone else, but I didn’t check

Advanced `git`

- forking, branching, merging
- What? Why?
- Let’s draw a quick picture.

Key command:

```
git merge <source> [dest]
```

merging

- `git` is filled with conflict and commitment issues.
- When you pull in changes, things sometimes get conflicted. The fix: git will identify problems, and you fix them manually, and then commit.
- `git` is smart, but this still happens frequently.

`git`-ing it done

- Exercise:

    1. Find a partner, and ask for their remote repo URL.
    2. Clone their repository.
    3. Make a change. Commit it. Push it to your own remote repository.
    4. On BitBucket, send your partner a pull request.
    5. Approve your partner's pull request.
    6. Profit.

That's g(it)

- Time for your questions about OCaml. I know they're there. :)
  - `option` types
  - Pattern matching

- Email me for any reason at any time. I'm compulsive.
  - For section-specific stuff, email me at samuelgreen@college.harvard.edu
  - For course-wide/admin stuff, please still email me at cs51heads@gmail.com