Lecture 6
Remotes

Sign in on the attendance sheet!
Midterm Review

• Everyone did great!
What We’ve Learned So Far

• Creating and cloning repositories
  git init, git clone

• Linear commit histories and diffs
  git log, git show, git diff

• Using the working directory and staging area and making commits
  git add, git reset, git checkout, git commit

• Using branches
  git branch, git checkout, git merge

• How git’s model for commit histories works
Today

- Remotes
- `git remote`
- `git fetch`
- `git pull`
- `git push`
- Github
Last Time: Branches

- **b4e2c29**: Alice: begin work on feature A
- **8fc42c6**: Alice: more work on feature A
- **6f96cf3**: Alice: even more work on feature A
- **8277e09**: Bob: even more work on feature B
- **e167179**: Bob: more work on feature B
- **8b7d883**: Bob: begin work on feature B
- **db82ca7**: Merge branch ‘featureA’ into master

Branches:
- featureA
- featureB
From the First Lecture: Git is a DVCS

Alice’s Repo

Bob’s Repo

Charlie’s Repo

push, pull, fetch
Scenario: Alice and Bob are writing a story about squirrels

Alice

- 8fc42c6
  Add draft of chapter 2

- b4e2c29
  Finish chapter 1

Bob

- 8fc42c6
  Add draft of chapter 2

- alice-repo/
  chapter-2-draft

- b4e2c29
  Finish chapter 1
Fetching

1. Tell git to set up Alice’s repository as a “remote repository” or a “remote”. This only happens once.
2. Tell git to download the commits and branch pointers that you don’t have from the remote repository
git remote add <remotename> <remoteurl>

Example use:

```
git remote add origin https://github.com/aperley/squirrel-story.git
```

- Adds a remote repository called “origin” located at https://github.com/aperley/squirrel-story.git
- “origin” is the default name for a remote, since often times the first remote you have is the one you clone from
- If you created the repository using `git clone` (rather than `git init`), the repository you cloned from is called “origin”
git fetch <remotename>

Example use:

```bash
git fetch origin
```

- **Downloads** and **updates** all branches published by the remote
- Stores these branches as `<remotename>/<branchname>`
- Does NOT affect your own branches, like master!
Listing Remote Branches

- `git branch -r` (only remote) or `git branch -a` (all branches)
You can checkout remote branches...

But you can’t make commits on them or move them like normal!

```sh
aperley@ARRAKIS MINGw64 ~/Documents/git_stuco/midterm_q4 (master)
$ git checkout origin/chapter-2-acorn-theft
Note: checking out 'origin/chapter-2-acorn-theft'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at eb2fc1c... Add chapter 2 acorn theft story and end text
```

Because they represent the state of the remote repository and are changed by git fetch!
How do you actually bring in the remote changes?

- **Alice**
  - **8fc42c6**
    - Add draft of chapter 2
  - **b4e2c29**
    - Finish chapter 1

- **Bob**
  - **b4e2c29**
    - Finish chapter 1
  - **8fc42c6**
    - Chapter-2-draft points to: 8fc42c6
How do you actually *merge* in the remote changes?

git fetch alice-repo

git checkout -b chapter-2-draft

git merge alice-repo/chapter-2-draft
git pull <remotename>

Example use:

`git pull origin`

- Runs `git fetch <remotename>`, then `git merge <remotename>/<currentbranch>`
- Ex: runs `git fetch origin`, then `git merge origin/master`
What about giving back?

git push alice-repo chapter-2-draft:chapter-2-draft
git push

git push <remote_name> <local_branch>:<remote_branch>

• Uploads the necessary commits to <remote_name> and changes <remote_name>/<remote_branch> to point to the same commit <local_branch> points to.
Summary

• Configuring remotes:
  • git remote [-v] – lists remotes [verbosely]
  • git remote add <remotename> <remoteurl> - configure a new remote
  • git branch –r or –a – lists branches including remote tracking

• Fetching:
  • git fetch <remotename> - downloads updates to all remote-tracking branches
to match the remote
  • git pull <remotename> - runs `git fetch`, then merges in updates to the
current branch

• Pushing:
  • git push <remotename> <branchname> - uploads changes in your branches to
the remote
Activity!

Clone:
YOUR_ANDREW_ID@unix.andrew.cmu.edu:/afs/andrew.cmu.edu/course/98/174/public/lecture6-practice

Create a branch named YOUR_ANDREW_ID
And make a commit to it

Push the branch up to origin