Version Control with Git

Ben Wasserman (benjamin@cmu.edu)
15-441 Computer Networks
Recitation 3
What is version control?

- Revisit previous code versions
- Backup projects
- Work with others
- Find where things broke
Version Control Workflow

- Check for any remote updates
- Do your work
- Test your work
- Check differences, try to isolate changes
- Check for any remote updates
- Commit your work
Options

- Git
- Subversion (svn)
- Mercurial (hg)
- Bazaar (bzr)
- CVS
- Dropbox
- Others…
**svn**

Usually **remotely hosted**, shared with a team.

```plaintext
svn commit
svn checkout
svn update
```

Your **private universe**, before commit.
No notion of “working copy”—each is a full repository.
Creating a Repository (repo)

Create locally

`git init .`

Create remote

`git init --bare`

Clone local copy

`git clone git://path/to/repo`
--bare or not?

• No-bare
  • Creates a repository in your working directory
  • Don’t need to create multiple copies of your repo
  • Won’t help if you nuke the directory/disk
  • This is probably what you need if you’ll work in AFS

• --bare
  • Creates a “server copy” for hosting the project
  • Workflow more similar to svn (but still better)
  • Everyone pushes to shared bare repo (like svn)
  • You don’t work in this copy; must clone elsewhere
  • You want this to develop on your PC
Aside: network protocols

- Use different protocols to pull/push to repositories.
- If on the same computer:
  - git://path/to/repo
- If hosted on AFS
  - ssh+git://path/to/repo
- No ssh keys for AFS, sorry
Aside: Configure git

• `git config --global user.name “Ben Wasserman”`
• `git config --global user.email “benjamin@cmu.edu”`
Clone

Pull a copy of the repo to develop on

`git clone git://path/to/repo`

`git clone ssh+git://unix.andrew.cmu.edu/afs/andrew/course/15/441-641/ANDREWID/ANDREWID-15-441-project-1.git`
status

- Which files changed?
- Which files aren’t being watched?
- Which files are stashed for commit?

`git status`
Pull

- Get latest updates from remote copy

```
git pull
```

- If this fails, you probably need to commit any unsaved changes
Commit

- Merge your changes into the repository

```bash
git add foo.c ...
git commit
```
Push

- Don’t push broken code!!

```bash
git push
```

- If this fails, you probably need to pull first
Branch & Merge

- Work on something different, without disturbing master/trunk

```
  git branch branch_name
  git checkout branch_name
  do stuff...

  git checkout master
  git merge branch_name
```
Tag

- Mark a revision as “final” or “ready”

```bash
git tag tag_name
git push --tags
```
Remote Hosting

- github.com
- bitbucket.org
- svnhub.com
- AFS
- Google code
- Sourceforge
Aside: AFS Permissions

- To make a bare repo in AFS that someone else can pull/push from:

  1. Make a new directory in your home dir
  2. fs sa . ANDREWID rlidwk
  3. git init --bare
Good practices

- Small commits
- Useful messages
- Commit frequently
- Develop in branches
- Tag releasable versions
Small commits

- Only change one thing per commit
- When something breaks, easier to trace
Helpful commit messages

- Say what you changed
- Keep the first line short
- Make commits easy to find
- [www.commitlogsfromlastnight.com](http://www.commitlogsfromlastnight.com)
Commit Frequently

- Make changes, commit them
- When something breaks, go to the commit that broke it
- Only push when ready for others to get the changes
  - Don’t make your teammates hate you
Git questions?
Checkpoint 2

- Add basic HTTP server
  - Read RFC 2616
- Start by parsing and building HTTP headers
- Serve error messages
- Then HEAD requests
- Then GET
- Then POST
Wireshark

- Packet monitoring software
- Install it. Use it.
- You will want this to examine the HTTP headers you’re sending/receiving
- Do the Wireshark question on HW1
All questions?