Lab 3: Version Control

Nathan Jarus

February 8, 2016

Introduction

This lab will get you familiar with the basics of git. You will need to sign in to https://git.mst.edu and clone the repository for this lab. Your repository will be named something along the lines of `2016SP-A-lab03-nmjxv3`. Make sure to clone with the HTTPS URL (unless you've set up SSH keys).

Scenario

Your friend made a git repository for the filter program from Lab 2. They want to collaborate with you to add some features to it.

Problem 1: `.gitignorance` is bliss

Remember what I said about not committing generated files to a repository? Well, your friend managed to add `a.out` anyway. Oops.

- Use `git rm` to delete `a.out` from the repo.
- Do you need to stage this change? (Hint: check `git status`.)
- Make a `.gitignore` that ignores `a.out`. Add it to the repository and commit.

Problem 2: Peace in the repository

While you were working on this feature, your friend has added another feature: ignoring whitespace at the start of lines.

Because your friend is imaginary, you'll have to help them merge their feature into `master`:

- `git checkout whitespace` and take a look at your friend's code. (You need to `checkout` a remote branch the first time you do anything with it.)

- `git checkout master`
- `git merge whitespace`
- You got a merge conflict! Fix the code to work right.
- Add and commit your fixes to complete the merge.

Problem 3: Features grow on branches

Okay, so now you've got the repository cleaned up! Let's add a feature to this code. Instead of hardcoding the comment character as `#`, we want to have it take a command line argument that specifies the comment character.

So, for example, you could run `cat story.txt | filter %` to filter lines that start with a %.

- Make a new branch to develop your feature on.
- Write your code, add it to the repository, and commit. (Too easy? Try using `getopt()`! `man 3 getopt`)
- Push your branch to the remote repository. (You can check Gitlab to make sure it's up there.)
  The first time you `push` a new branch, you have to tell git to make a new branch on the remote. `git push` will tell you the right command to run.

Now, you'd like to merge your feature into `master` too!

- Okay so merge it already.
- Don't forget to push after you've merged!

Problem 4: Paradox-free time travel

Your friend reveals that they accidentally deleted some comments from the file.

- Figure out which commit they deleted the comments in.
- Use `git revert` to undo that mistake!
- Make sure to push your revert to the remote repo.

Epilogue: Submitting

Your repo on gitlab is your submission, so whatever is up there is what we will grade. Make sure you've pushed both your feature branch and master. You can double-check on the Gitlab website to make sure your submission looks the way you want it to.