Git and Github

Zhentao Shi and Zhan Gao

Git is a version control system useful when developing and maintaining coding projects as well as preparing long documents.

With Git, we no longer need to save historical versions `paper_v1.tex`, `paper_v2.tex` up to even `paper_v11.tex`. It is easy to track changes to code and text files and maintain a history of changes. In addition, it is safe to experiment and test
different branches of development without changing the current main branch. More importantly, Git, together with online platforms like Github, empowers collaboration. Researcher are able to work on the same project, even on the same file, separately, see what other people have done and resolve conflicts when needed.

To be further motivated, please refer to git vs. Dropbox from a researcher's perspective by Michael Stepner.

Free Git books and tutorials are available online. Self-learning is important.

- Atlassian Online Tutorial This online tutorial is a good starting point to learn the basics of Git.
- The version control module from The Missing Semester of Your CS Education, offered by MIT, is also a great reference for beginners. The online notes is accompanied with a lecture video. The course has been translated to Chinese, see Chinese version.
  Though targeted at computer science majors, this open course is also helpful for econometrics researchers. Modules other than the version control one are also recommended.
- If more elaboration is needed, online courses, Udacity Course and CodeAcademy Course for examples, and other video tutorial are helpful.
- A comprehensive reference: Pro Git.

We introduce some essential Git commands. There are two ways to interact with Git. Git provides a command line tool Git Bash, and there are many free Git GUIs available (We recommend SourceTree). Even if we use a GUI, knowing the basic commands is helpful.

Basic Commands

Identity

- `git config --global user.name <name>`
- `git config --global user.email <email>`

Local

- .gitignore
- `git help <command>` gets help for a git command

- `git init` creates a new git repo, with data stored in the `.git` directory
- `git status` inspects the contents of the working directory and staging area.
- `git add filename` adds files to the staging area from working directory.
- `git add filename1 filename2` adds multiple files to the staging area.
- If the file is changed, we can use `git diff filename` to see the difference between the file in the working directory and the staging area. Reuse `git add filename` to add the updated file to the staging area.
- `git commit` stores changes from the staging area to the repository.
- `git commit -m "Commit Message"` The commit message must be in the quotation marks.
- `git log` displays historical commits stored chronologically in the repository.
- `git log --all --graph --decorate` visualizes history as a DAG
- `git tag -a v1.0 -m 'message' [optional:commit-id]`
- `git rm --cached filename` remove

Eraser-like features

The latest commit is called **HEAD** commit

- `git show HEAD` displays view the HEAD commit.
- `git checkout HEAD filename` restores the file in the working directory to what you made in last commit.
- `git reset HEAD filename` unstages the file from committing in the staging area. This command resets the file in the staging area to be the same as the HEAD commit. It does not discard file changes from the working directory; it just removes them from the staging area.
- `git reset SHA` works by using **the first 7 characters** of the SHA of a previous commit.

Branch

- `git branch` displays the current branch.
- `git branch brach_name` creates a new branch. The branch name cannot contain white-space
- `git checkout branch_name` switches to a branch
- `git merge branch_name` merges the branch into master
  merge conflict: Git doesn't know which changes you want to keep. Modify the file in the working directory and commit it again to avoid conflict.
- `git branch -d branch_name` deletes a branch from the project.

In Git, branches are usually a means to an end. We create them to work on a new project feature, but the ultimate goal is to merge that feature into the master branch.

Remote

- `git clone https://github.com/zhentaoshi/econ5170`

- `git remote add origin` adds the origin remote's URL.
- `git remote -v` lists git project's remote copies.
- `git push origin master` uploads local commits to the remote repository.
- `git pull` downloads the remote copy and merge.
- `git fetch` fetches the remote copy to the local hard disk. **Note:** This command will not merge changes from the remote into your local repository. It inspects the changes on the remote branch.
- `git remote set-url origin https://github.com/zhentaoshi/econ5170` changes the remote's URL.

Collaboration typically works as follows:

1. Fetch and merge changes from a remote repository;
2. Create a branch to work on a new project feature;
3. Develop the feature on your branch and commit your work;
4. Fetch and merge from the remote again (in case other collaborators have uploaded new commits while you were working);
5. Push your branch up to the remote for review.

Conflict

Collaborators working separately on the same paragraph of a file may easily encounter conflicts. In such situations, the command `git pull` will not merge changes from the remote into your local repository automatically due to the conflict. You are expected to something like this in the shell:

```
git pull
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://github.com/your-user-name/repo-name
   9b02654..0462f26  master     -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

If you check the status by `git status`, you would be told that you have unmerged paths.

```
git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)
```

```
Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

To resolve the conflicts, we simply use a text editor, say *Sublime Text 3*, *Atom* or *Visual Studio Code*, to open the file with conflicts. In the illustrative example, it is the `README.md` file.

```
# Conflict-Resolving

<<<<<<< HEAD
Local changes.
=======
Remote changes.
>>>>>>> 0462f26ddfe83c35424168c2d7a0bed62c653413
```

You can see conflicts indicated by Git. The content between `<<<<<<< HEAD` and `=======` is on HEAD and content between `=======` and `>>>>>>>` is from the commit `0462f26`. To resolve the conflicts, you just remove `<<<<<<< HEAD`, `=======` and `>>>>>> 0462f26`, and keep what you want to keep in the file.

In some cases, the difference between the local branch and the remote branch is significant, or the file with conflicts is of a special format, say Lyx file. It might be cumbersome to resolve all conflicts in the text editor. Another way to resovle conflicts is simply to open two versions of the files and copy and paste new updates and then commit the updated file. It is easy to do so in *SourceTree*: select the commit -> right click the file -> click `Open Selected Version`.

In addition, we can resolve conflicts with *SourceTree* (**preferred**) or external merge tools, for example *KDiff3*. For details, it is recommended to refer to the answer on Stack Overflow.

Github

Github is so far the most popular provider of Git repository internet hosting. Github's free account offers unlimited public repositories, and university students and teachers can register for unlimited complimentary private repositories as well. Among other internet hosts such as bitbucket and Gitlab, both of which offer unlimited private repositories for free, Github boasts the largest community of code developers.

We collaborate with students and researchers on Github extensively. This is ZT's personal Github profile https://github.com/zhentaoshi.

Acknowledgment: Part of this notes is based on the course offered by CodeAcademy.