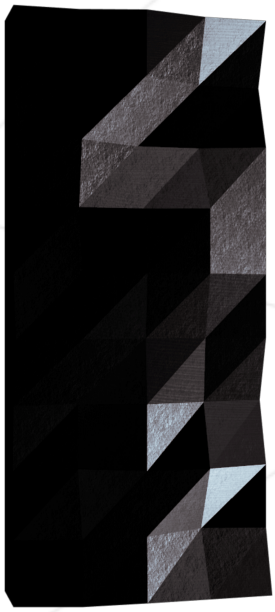


HackHighSchool: Github

How To Git Gud

Ruchen Liu ruchen@42.us.org
42 Staff pedago@42.fr

Summary: The problem with git jokes is that everyone has their own version.

Contents

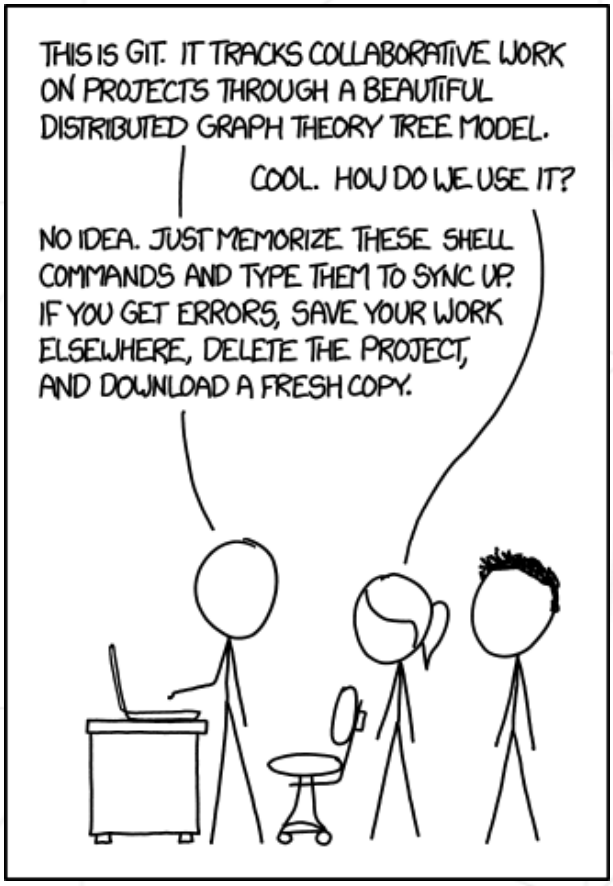
Chapter I

Foreword

```
>> git clone "https://github.com/EugeneKay/git-jokes.git" git-jokes \

Cloning into "git-jokes"...
remote: Counting objects: 185, done.
remote: Total 185 (delta 0), reused 0 (delta 0), pack-reused 185
Receiving objects: 100% (185/185), 56.69 KiB | 0 bytes/s, done.
Resolving deltas: 100% (56/56), done.
```

- git pull a day keeps the conflicts away
- If you have anything staged, commit now or stash forever
- Commit early, commit often. A tip for version controlling - not for relationships
- Fork by yourself, shame on you. Fork with a friend, now we're getting somewhere.
- Knock knock. Who's there? Git. Git-who? Sorry, 'who' is not a git command - did you mean 'show'?
- Git gets easier once you get the basic idea that branches are homeomorphic endofunctors mapping submanifolds of a Hilbert space.
- Ceci n'est pas une commit
- git out of da choppah!


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.


Has merge conflicts
ONLY DEVELOPER WORKING ON PROJECT

Chapter II

Goals

As you may be aware, Git is a version control software application created by Linus Torvalds (the creator of Linux) with the collaboration of many other developers.

A version control application keeps track of all the changes that you do in the files of your project. Every time you make changes to files of a project, you can push those changes to a repository. Other developers can pull your changes from the repository and continue to work with the improvements that you added to the project files. Version Control is kind like having a giant undo button for your project. It makes it easy to save different versions of your files at different points in time. You can also restore previous versions and compare different versions. It makes a lot easier to collaborate with other people. Which means no more sending huge zip files of your project back and forth. Learning version control is like learning how to type, it's hard at first but once you can do it, it makes you a lot more efficient. Successful sofware companies like Google and Facebook all use Version Control.

Master these Git commands:

- Git Config
- Git Basics
- Rewriting Git History
- Undoing Changes
- Git Branches
- Remote Repositories
- Git Push/Pull
- Git Log

`use git -help if you GIT stuck or forget a specific command!`

Chapter III

General Instructions

1. Create a github account.
2. Practice your git commands and go on your own github to check.
3. All your exercises you turn in must be in 1 directory named "githubPractice"
4. Ask your peers, mentor, slack or anywhere else if you need any help, and make sure to have fun

Chapter IV

Github Cheat Sheet

The simple steps to adding a file/directory to your repo is by

1. git add -A
2. git commit -m "A useful comment of what changes you made"
3. git push

`it must be in that order!`

BASICS (adding, removing, and editing files):

git config - -list : Display your local repo config settings for that project/folder.

git config - -get remote.origin.url : Show remote repository URL. It'll give you the url path of the current git project you're working on.

git status : If you want to see what you haven't git added yet. Everything in red means it hasn't been added yet, and everything in green means that it is ready for commit.

git clone <repo link> : Clone repo located at <repo link> onto local machine. Original repo can be located on the local file system or on a remote machine via HTTP or SSH.

git add -A : This will add ALL your unstaged changes in your directory to a staged version.

git commit -m "your commit message" : To commit your new changes to your files. Add a message to provide a descriptions to your team or yourself of what you changed/updated/deleted/fixed.

git rm <filename> : Deletes file from working directory and stages the deletion.

git pull : If you're working on different computers with different versions saved on the computers, or if you are working in a group, then you can use this command to pull the latest changes from the remote repo.

git log –oneline –decorate –color : Display the entire commit history with color.

git diff HEAD : Show difference between working directory and last commit.

UNDOING CHANGES

git reset : Reset staging area to match most recent commit, but leave the working directory unchanged.

git reset <filename> : Unstage a file for commit. To undo adding a file you didn't want to commit.

git revert <commmit> : Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch.

git clean -n : Shows which files would be removed from working directory. Use the -f flag in place of the -n flag to execute the clean.

ADVANCED (stashing, merging, branching):

git rm –cached <filename> : If you want to delete a file from your git repo but not your local directory.

git stash : Stash current changes.

git stash list: List all stashed change sets.

git stash apply: Apply stashed changes.

git branch : List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.

git checkout -b <branch> : Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

git merge <branch> : Merge <branch> into the current branch.

Chapter V

Set Up A Github Account

1. Go to https://github.com/join and create an account. Choose the free plan.
2. Press "start a project" and verify your email
3. Type in "gitPractice" in the repository name section and press "Create repository" (you can initialize with a README if you want, read below for more info)
4. On the next screen, press the clipboard/arrow button on the right of the github url that was created. It should pop up with a bubble saying "copied!"
5. Your link is now copied so now you can go on your terminal and use the git commands you read about earlier.

What is a README? Many projects contain a file named "README" that gives a general description of what the project does and how to use it. It's often a good idea to read this file before doing anything with the project, so the file is given this name to make users more likely to read it.

If you don't know something like "repo" or "remote vs. local repo", Google it!

Chapter VI

Make Your First Local Git Repo

	Exercise
Files to turn in : `versionControl.txt`	

1. Use the git clone command to create a directory with the url link you just copied onto your clipboard. Make sure you put that directory somewhere you will be able to find later for example on your Desktop or Home directory.

2. Inside that directory, make a file name versionControl.txt. Inside the file answer these questions:

   (a) How could having easy access to the entire history of a file make you a more efficient programmer in the long term?

   (b) As a programmer, when would you want to have a version of your code saved? and Why?

   - At regular intervals (e.g. every hour)
   - Whenever a large enough change is made (e.g. 50 lines)
   - Whenever there is a long pause in editing
   - When you choose to save a version

   (c) What is the importance of commit messages?

3. After answering the questions, save the file.

4. Now use git commands and add the new directory and file into your remote repo. (aka "Push" your local repo into your remote repo)

5. Make another file and name it deleteme.txt

6. Now use git commands and add the new directory and file into your repo.

7. Go on github.com/yourUserNameHere, go to your projects, click on Repositories, click on gitPractice. Did you directory and files show up?

8. If both files show up, then you're doing great! Now go back to your terminal and remove that deleteme.txt file using only git commands. Once done, refresh your github page and see if it got deleted.

How Often to Commit:

A good rule of thumb is to make one commit per logical change. For example, if you fixed a typo, then fixed a bug in a separate part of the file, you should use one commit for each change since they are logically separate. If you do this, each commit will have one purpose that can be easily understood. Git allows you to write a short message explaining what was changed in each commit, and that message will be more useful if each commit has a single logical change.

Chapter VII

Collaborate With A Partner

	Exercise
Files to turn in : `secretPartnerAname.txt`	

1. Find a partner to collaborate with on this project.

2. One person from the group needs to make a new and empty repository on github. Go to your github page and press Repositories and then press the green New button. Name it Teamwork.

3. Press Settings, then press collaborators, add your partner as a collaborator. Your partner will get an email and a link to the shared repo. They can now clone that repo onto their own computer.

4. Now go on your own Terminals separately, (don't forget to exit out of your last directory) and clone your new url link onto your computer.

5. Go inside that new local repo.

6. You and your partner both need to create a new file. One of you should name the file secretPartnerAname.txt and the other person should name her/his file secretPartnerBname.txt. For example, if Aditi and Lakshmi were working together then Lakshmi would name her file "secretLakshmi.txt" and Aditi would name her file "secreteAditi.txt".

7. In your own secret files, write a secret message to each other but don't show each other yet.

8. When you're done, push that file into your remote repository.

9. When you are sure that you've pushed your file into the shared remote repo, figure out how to get your partner's secret message into your local repo using ONLY git commands in your terminal.

10. Once you've successfully downloaded your partner's message, then go inside your partner's file and type in the git command(s) you used to get that message.

11. Lastly, push your changes into the shared remote repository.

Chapter VIII

Grade Your Work!

1. Go to your project page and click "Set the project as finished".

2. Next click "Subscribe to defense" and schedule two peer corrections.

3. If you run out of correction points (check your number on your profile page!), it means you need to open correction slots and correct other people in return. :)

4. When you're getting graded, have the person grading you get the git directory from your github account, you don't need to be logged into your github in order for that to be possible. You just need to know your github username.

Goodluck!