

CMPE-013/L

git

Max Lichtenstein

based on slides from Maxwell James Dunne

Containing images from:

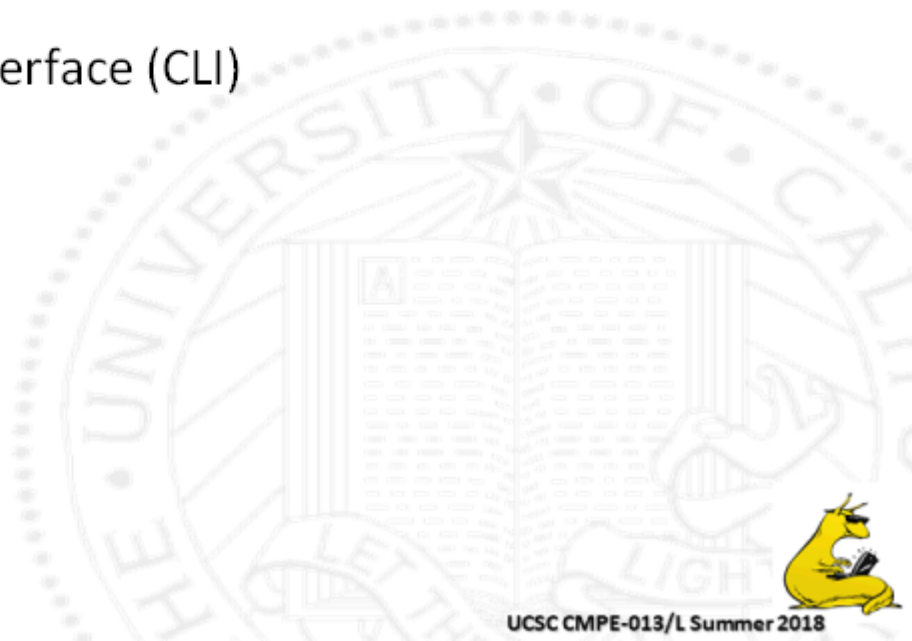
Pro Git book, written by Scott Chacon and Ben Straub

“10 Things I Hate About Git” by Steve Bennett



Roadmap

- Git overview
- Command Line Interface (CLI)
- Using git

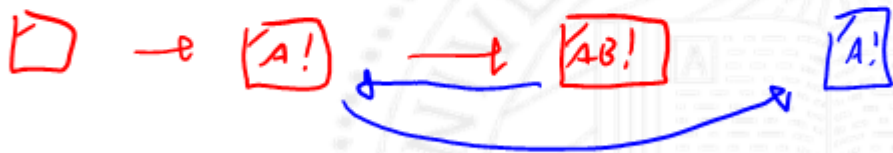


git Overview



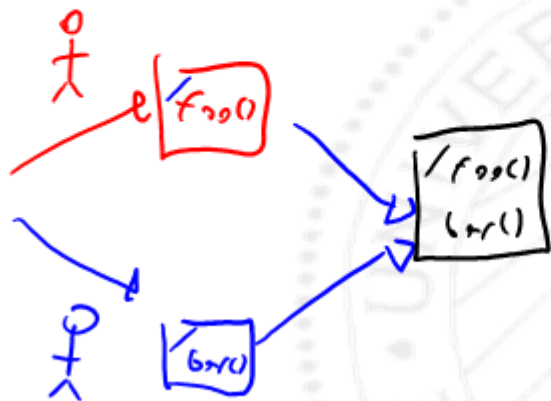
Basic Version Control (1/3)

- Need to keep track of changes in a collection of documents
- Keep the ability to “roll back” to previous working code



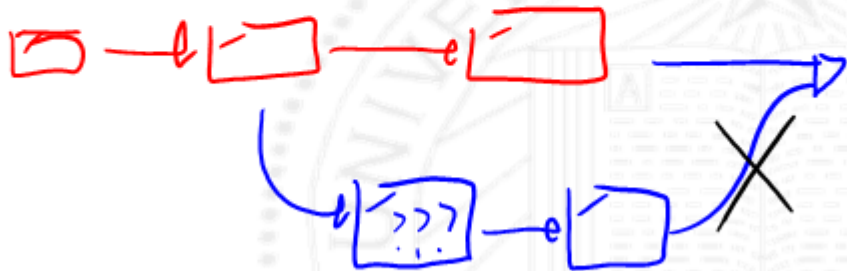
Basic Version Control (2/3)

- Coordinate with multiple programmers working on the same code.



Basic Version Control (3/3)

- Try experimenting – make a “branch”, and merge back later if it works

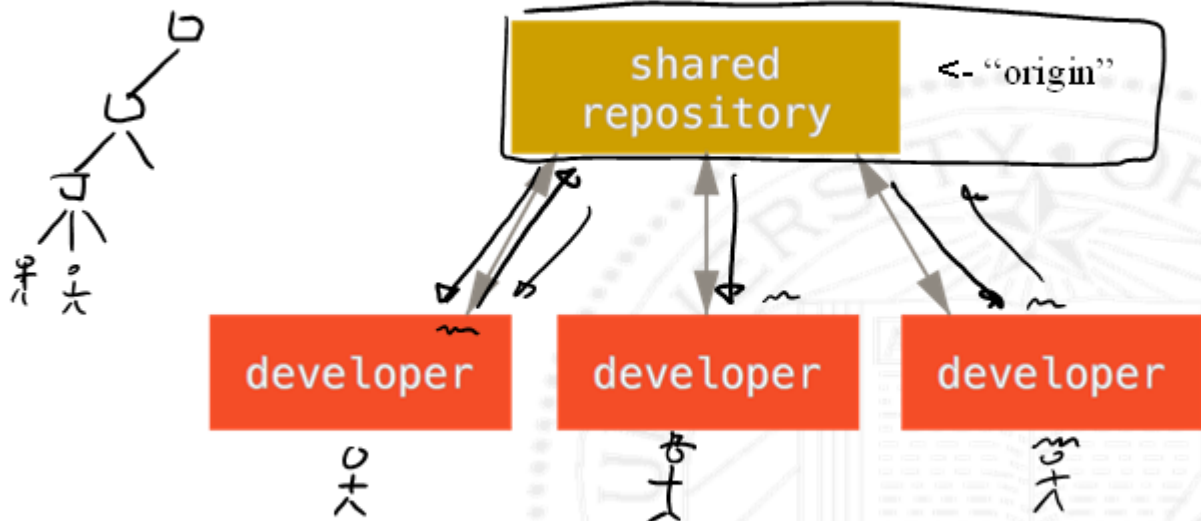


Git Theory

- Like all version control systems, git keeps tracks of *changes*.
- Except Git differs in a major way.
 - Instead of checking in changes to a server automatically, you 'commit' them locally.
 - Only 'push' your changes to server when you want
 - Only 'pull' other changes when you want



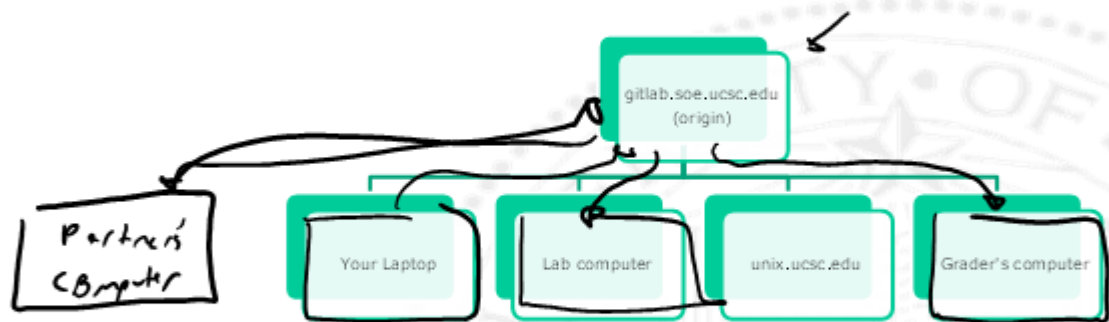
Upstream and Downstream



- <https://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows>

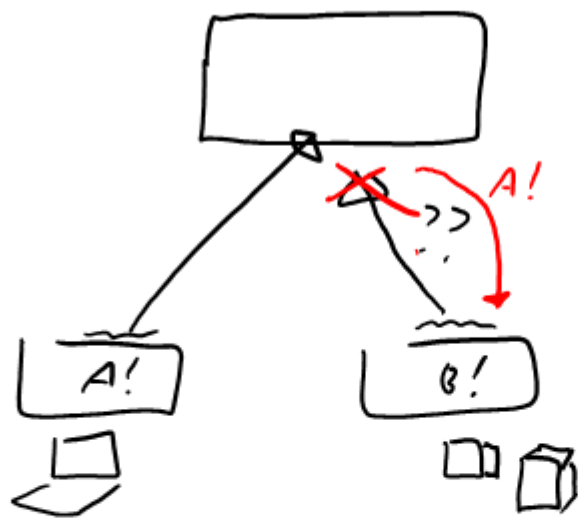


In CMPE13:

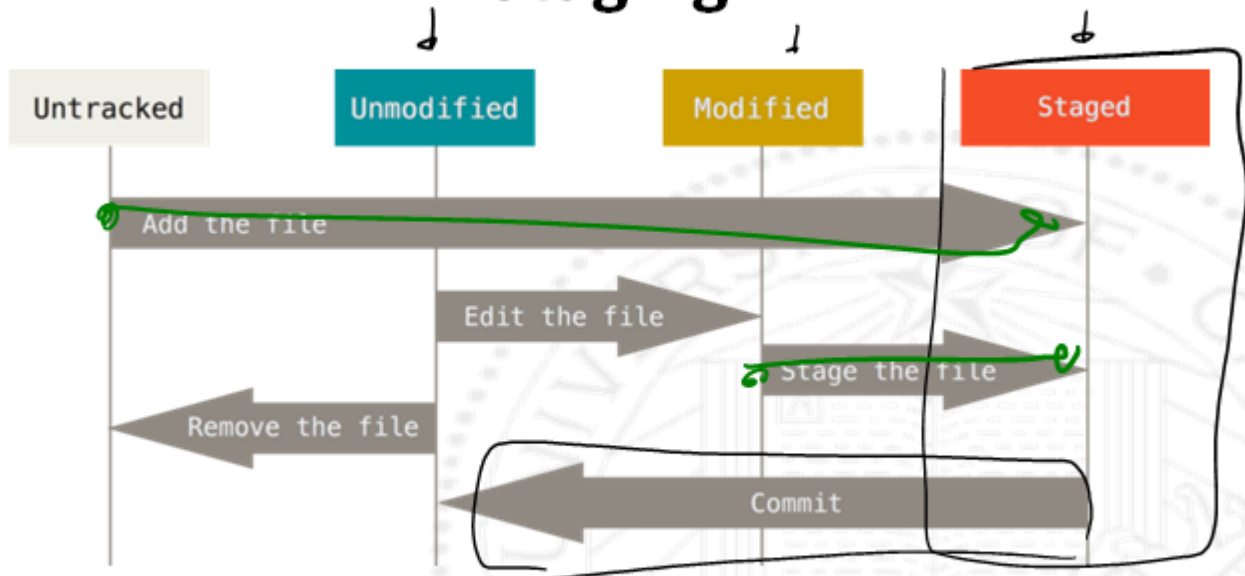


- <https://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows>

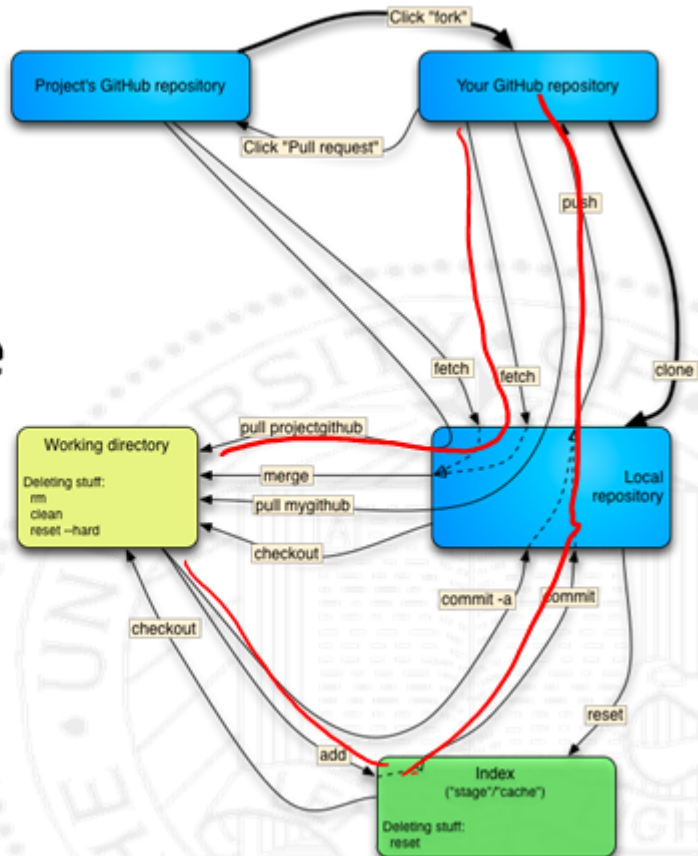




Staging



Git has a learning curve



Command Line Interface (CLI)



Basic CLI

- Command Line Interface: a means of interacting with a computer using only text
- Commands often have parameters or arguments added to them (e.g.: `ls -a`).
- Contrasts to a GUI (Graphical User Interface)
- Powerful and efficient, once you learn it
 - How we will use git



Basic File Systems

- How files are stored and organized by the operating system
- All modern OSes use a hierarchical file system
 - Top level is “root”
- CLI is “in” a directory
 - working directory



Most useful bash commands

- `$ ls` =list contents of working directory
- `$ cd` =change directory, try “`cd ..`”
- `$ mv` =move or rename a file (careful!)
- `$ cat` =print contents of a file
- `$ rm` =delete a file (careful!)
- `$ git=`



Using git



Git Basics

- The fundamental git object is a “**repository**”
 - Just a folder with some .git files in it (.git, .gitignore, etc)
 - But these files contain a history of the folder’s contents
- Git only tracks changes in files that are “added” to the repository.

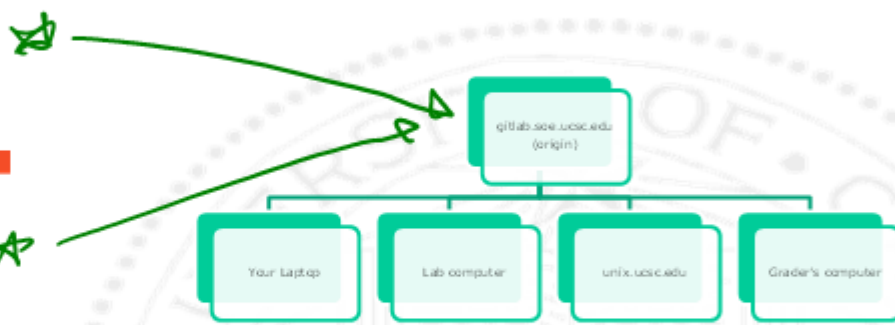


Git Basics

- At any point, a 'commit' can be made. This will take a snapshot of all tracked files.
- Once 'commits' have been made, they are 'pushed' back to the server as a separate operation



Git Flow



Git Practice

- All commands prefaced with git, eg
 - git clone < make a local copy of a repo (ie, a folder)
 - git status < see what's going on
 - git add < tell git to track a file
 - git commit < take a snapshot
 - git push < send your changes upstream (to server)
 - git pull < bring your changes downstream (to you)
 - git log < see your history (also, gitk)



Git Commands (clone)

- Copies Repository for the first time. Only used when you do not have a copy of the repository

```
$ git clone git@git.soe.ucsc.edu:/classes/cmpe013/spring16/MaxSampleStudent.git
Cloning into 'MaxSampleStudent'...
remote: Counting objects: 6681, done.
remote: Compressing objects: 100% (6395/6395), done.
remote: Total 6681 (delta 4526), reused 400 (delta 237)
Receiving objects: 100% (6681/6681), 76.70 MiB | 1.98 MiB/s, done.
Resolving deltas: 100% (4526/4526), done.
Checking connectivity... done.
```



Git Commands (status)

- Give status of the repository. Showing state of files within the repo

```
$ git status
On branch master
Initial commit
Untracked files:
  (use "git add <file>..." to include in what will be committed)
   README.txt
nothing added to commit but untracked files present (use "git add" to track)
```

```
$ git status
On branch master
Initial commit
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
   new file:   README.txt
```



Git Commands (add)

- Either adds a new file to the repository or marks a file for commit. This command has no output. Call 'status' again to verify operation.



Git Commands (commit)

- Creates a snapshot of all files in the repository at the current time. At any point you can come back to this point so commit early & often.

`$ git commit -m "committing an empty readme for a test"`

- Adding the `-a` automatically adds all tracked files

`$ git commit -am "committing an empty readme for a test"`



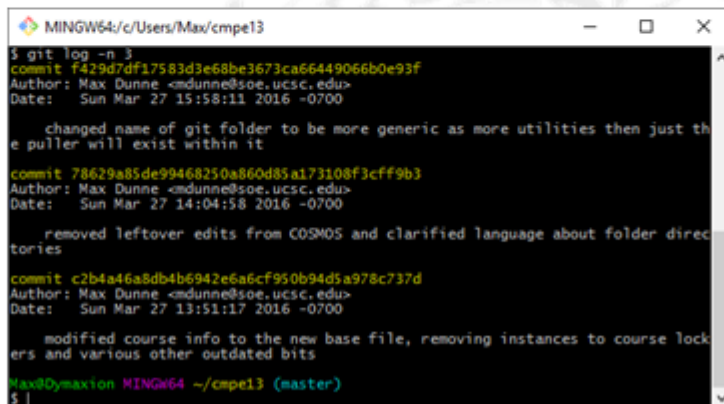
Git Commands (push/pull)

- Synchronizes the local repository with the remote server. Push should be used frequently to ensure server copy is updated.
- Once pushed to the server, the files are safe from any and all catastrophes that happen to your system.
- **WARNING:** pull will overwrite your local files. Use with discretion.
- We are not sympathetic to your data loss.



Git Commands (log)

- Gives a history of commits for the repo. Call with argument '-n 1' if only last ID is desired.

A terminal window titled 'MINGW64/c/Users/Max/cmpe13' showing the output of the command 'git log -n 3'. The output lists three commits with their IDs, authors, dates, and descriptions. The first commit is 'changed name of git folder to be more generic as more utilities than just the puller will exist within it'. The second is 'removed leftover edits from COSMOS and clarified language about folder directories'. The third is 'modified course info to the new base file, removing instances to course lockers and various other outdated bits'. The prompt '\$ |' is visible at the bottom.

```
MINGW64/c/Users/Max/cmpe13
$ git log -n 3
commit f429d7df17583d3e68be3673ca66449066b0e93f
Author: Max Dunne <mdunne@soe.ucsc.edu>
Date: Sun Mar 27 15:58:11 2016 -0700

    changed name of git folder to be more generic as more utilities than just the puller will exist within it

commit 78629a85de99468250a860d85a173108f3cff9b3
Author: Max Dunne <mdunne@soe.ucsc.edu>
Date: Sun Mar 27 14:04:58 2016 -0700

    removed leftover edits from COSMOS and clarified language about folder directories

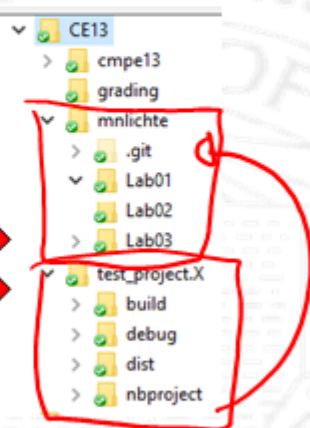
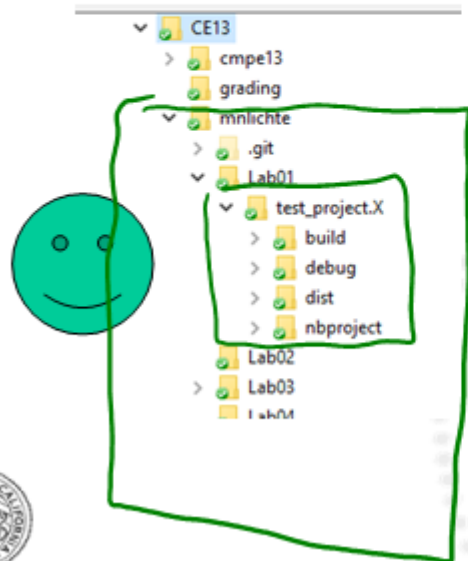
commit c2b4a46a8db4b6942e6a6cf950b94d5a978c737d
Author: Max Dunne <mdunne@soe.ucsc.edu>
Date: Sun Mar 27 13:51:17 2016 -0700

    modified course info to the new base file, removing instances to course lockers and various other outdated bits
Max@Dymaxion MINGW64 ~/cmpe13 (master)
$ |
```



Git Usage in Labs

- Work out of your repository:



Git Usage in Labs



- Commit often – Once per milestone, or $\sim 1/\text{hr}$
 - Use descriptive commit messages
 - Why not push too?
 - Committing is like saving, but better! You can only lose as much work as your last commit
- Pull when you change computers





	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Randall Munroe, xkcd.com



Git Workflow

1. Clone the repository or pull if the repository if work has been done in a different location.
2. Do work.
3. Commit your work (and why not push too?)
 - a) `git add`
 - b) `git commit -m "good message"`
 - c) `git push`
4. Repeat as necessary.



Commit through MPLABX

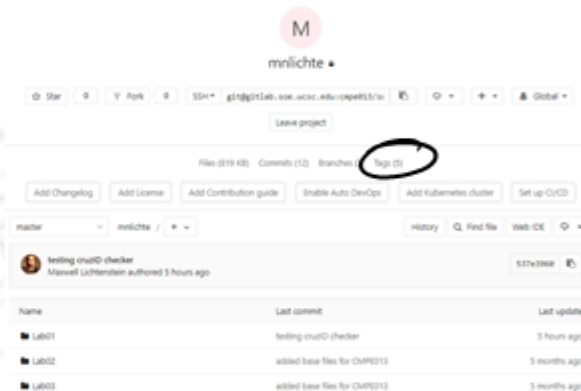
The screenshot illustrates the process of committing a file through MPLABX. On the left, the project tree shows the file 'lab8_tester.c' selected. A context menu is open over it, with the 'Commit...' option circled in red. The 'Commit - lab8_tester.c' dialog box is open on the right, showing the commit message field, author and committer information (Max Dunne), and a table of files to be committed.

File	Status	Commit Action	Repository Path
lab8_tester.c	-Modified	Commit	Lab8\Morse\lab8_tester.c



Lab Submission

1. Commit and push your work
2. Go to GitLab server
3. Click “tags”
4. Add a tag named “LabX_submission_Y”
 1. We take the largest Y, so you can resubmit by making a new tag with a bigger Y
5. To verify your submission:
 1. Git checkout <tag> / git checkout master
 2. or download as zip



A few more git Tips:

- git status is your best friend
- git log is pretty good too
- gitk is fun

- Ask for help

- Don't Panic!

