Git presentation
- what/why version control?
- main types of version control
  - centralized
  - distributed
- git conceptually
- git commands → status, diff, add, commit, pull, push
- Github brief over view
- moving forward
  - branching/merging
  - pull requests

What/why version control?
- fancy save button
  - normal save overwrites file and does not track its history
  - version control tracks history (can revert)
    - can leave a message at every save point
    - author of each save is tracked
- Good for
  - reverting to non-bugged
  - backing out work you don't want
  - figure out who last worked on problem section
- Not only used for code
  - use for notes
  - legal documents
  - artistic writers
  - group planning

Main types of version control
- centralized
  - one repo rest copies
- decentralized
  - every computer has a full repo
  - only have to be connected to push/pull

Centralized disadvantages
- requires connection to do any version controlling
- every version control step has to access remote repository
- if central repo goes down you lose the repo
  - work on individual machines are copies and do not contain the full work log

What is a repo needs to be addressed before comparison

Git conceptually
- difference between save & commit
- save is specific to file and does not contain version history
- commits are the essence of version history
  - commit is to version control as save is to individual files

Git commands (status, diff, add, commit, push, pull)
order to present
- pull / push
- status
- add
- commit

- pull / push = only commands that require connection
  - compares local repo to remote repo
  - good to start work w/ a pull
  - push can result in merge conflicts
    - git sorts out and won't let you push till you resolve conflict

moved to Git conceptually section

- status
  - compares local work to last commit on local repository
  - tells what is staged for commit
  - tells how many commits have happened since last push

- add / reset
  - stages changed files for commit
  - reset removes all staged files so you can start over staging

- commit
  - creates a commit of the staged files to the local repo
  - don't forget to use a message
  - (how to escape vim if forgot -m)

Git commands cont.

- Diff
  - what are the changes that have been made compared to last commit on local repo
  - will show all files changes unless passed a file name then just that files changes

- help
  - shows possible commands
  - if help <command> will show more detailed info on that command

slide showing which commads deal with which repo (local or remote)

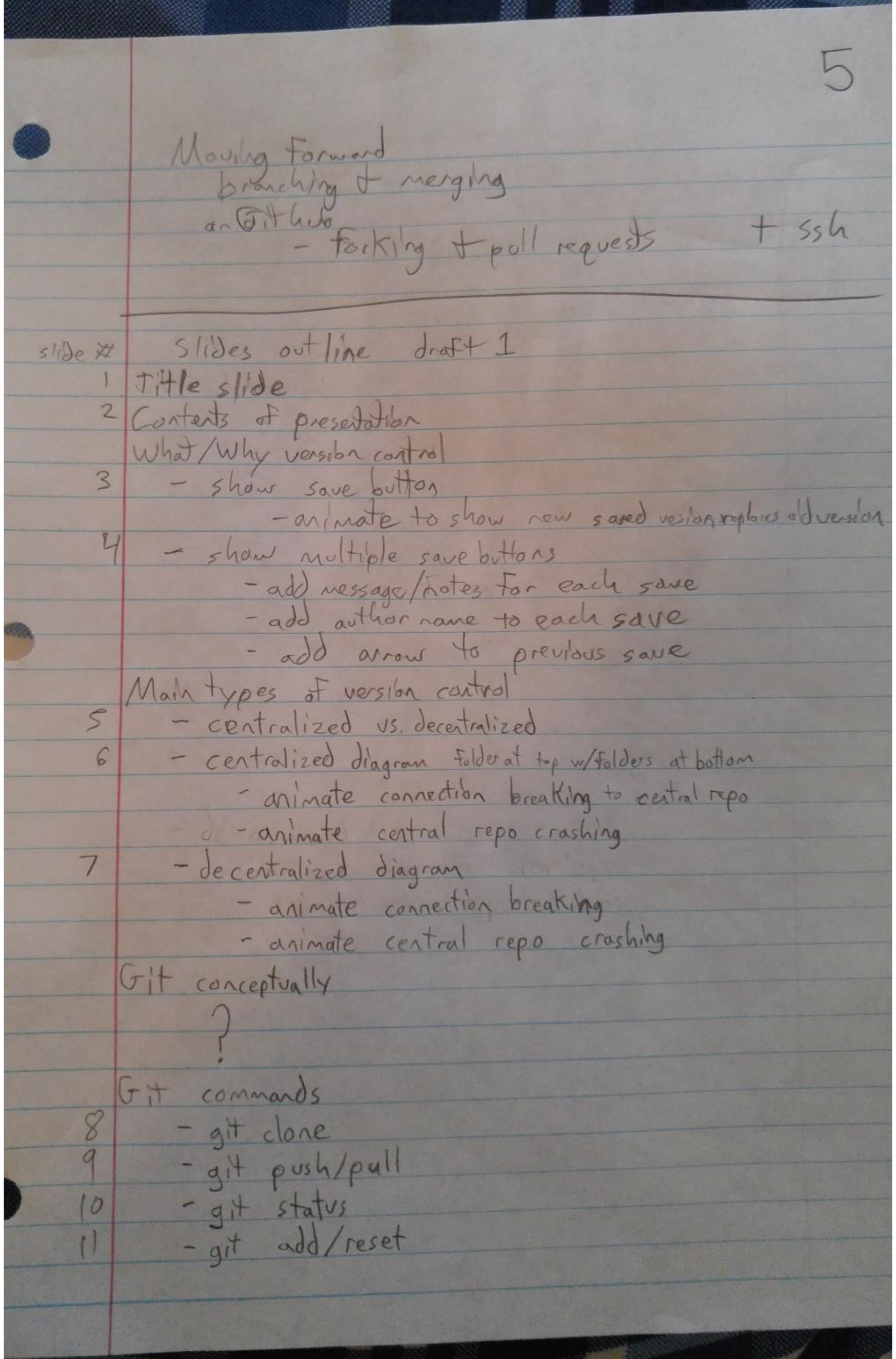
Git clone needed for commands covered

Github brief overview

github ≠ git
cloud service Free
open source

github is a webservice like a cloud service for a main repo, git is a set of version control command line tools

- Free for public repos
  - lends really well to open source projects



Moving Forward
branching & merging
on GitHub
- forking & pull requests + ssh

---

slide # Slides outline draft 1

1 Title slide

2 Contents of presentation

What/Why version control

3 - show save button
- animate to show new saved version replaces old version

4 - show multiple save buttons
- add message/notes for each save
- add author name to each save
- add arrow to previous save

Main types of version control

5 - centralized vs. decentralized

6 - centralized diagram folder at top w/folders at bottom
- animate connection breaking to central repo
- animate central repo crashing

7 - decentralized diagram
- animate connection breaking
- animate central repo crashing

Git conceptually

?

Git commands

8 - git clone

9 - git push/pull

10 - git status

11 - git add/reset

Slides outline draft 1 (cont.)

Git commands (cont.)

12 - git commit
13 - git diff
14 - git help
15 - git commands diagram
- group commands based on which repo accessed
- highlight basic workflow commands

Github brief overview

16 - Github intro slide
- highlight Github ≠ Git

17 - Decentralized diagram w/cloud for main repo

Moving forward

18 - Slide w/topics + appropriate links

19 Conclusion Slide

Slide count: 19 + ? (~22)

Problems with/changes to Slide Outline draft 1

- git conceptually slides not planned out
- Main types of version control uses some version control terminology not yet discussed
  + add discussion of what is a repo to what/why version control section
- Perhaps switch or combine Github brief overview slides
- git commands have too many slides. Maybe combine related commands to single slides
- left out slide covering extended uses of version control in the what/why version control category

How to present Git conceptually

topics needing addressing

- save vs commit
- commit staging process
- push / pull of commits

Save vs.	Commit
- per file based	- based in the repo
- can't restore previous saves	- can revert to previous commits
- changes to a single file	- can contain changes to multiple files

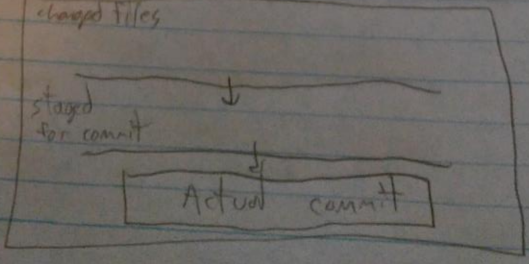
commit staging process

git knows what files have been altered (have to save after altering) compared to the last commit (repo version of save)

add the altered files to be staged for a commit

- this is not the commit itself
- allows you to choose which files are included in a single commit
  - best to only include files whose changes are related
- can reset the stage if a wrong file was added

concept drawing for this slide



How to present Git conceptually cont.
- push/pull
  - need to represent that repos can be off by commits
  - no limit to this commit difference
  - push = update remote repository w/ your commits (won't include altered files that weren't included in a commit)
  - pull = update local repo with the extra commits that the remote repo has
  - if both repos have commits on the same file have to pull before push
    - push can result in a merge conflict (two people changed the same line of code)
      - git will highlight the conflict in the file and after it has been manually fixed you can push

---

Will need to cover obtaining git commands
- on mac if you use git command but don't have git, it will show you how to install
- put link for install instructions on windows (for command prompt)
  - highlight that command line commands needs to be checked

---

add ssh to Moving Forward / Future directions

Slides Outline Draft 2

1 Title slide
2 Contents of Presentation
What/why version control
3-4 Keep slides 3 & 4
5 - Extra uses of version control
6 - Repo vs. File → what is a repo
Main types of version control
7-9 keep slides 5, 6 & 7 from draft 1
Git Conceptually
10 - Save vs. commit
11 - Commit staging process
12 - push/pull
Git Commands
13 - Git clone (obtain git commands)
14 - Git push/pull
15 - git status/add/reset
16 - git commit
17 - git diff/help
18 - git commands diagram
Github brief overview
19 - combine slides 16 & 17 from draft 1
Moving Forward
20 - keep slide 18 from draft 1
21 Conclusion

Add some where in git commands how to obtain git + add to first slide (slide 13 of draft 2)

~~Combine title slide & contents slide~~

Git Presentation topics outline (based on slides outline2)

slide 1
- Present myself and the presentation

slide 2 contents slide

What/why version control
- start with a brief overview of what version control actually is
- cover some possible reasons why you would want to use it

Main types of version control
- cover the difference between the two main types of version control
- discuss why you might want one over the other

Git Conceptually
- go over how git actually works before learning how to make it do work
- base of understanding before learning commands
- how works at higher level before diving into specifics

Git Commands
- how to actually accomplish the git concepts

Github Brief overview
- introduction to a topic often associated w/git

Moving Forward
- some more advanced things to do using git/github

Git pres topics outline (cont)

slide 3 show save button
- impart that the normal operation of saving overwrites previous version of the file
- no method of pulling back previous versions of the file

slide 4 multiple save buttons
- when using version control and you save it's not a complete overwrite of the previous version
  - still have access to previous versions
- when saving using version control you can add a message to the save so you can later remember what you changed w/ the save and why
- The save also remembers who did the saving and so people know who to contact if that portion stops working
- This concept of version control through saving isn't 100% how it works but is analogous
- V.C. is ∴ good for reverting to non-bugged code if working on a commercial product
  - good for tracking when/how a bug entered code, what that portion was supposed to address and how to fix the bug
  - good for scrapping changes you made and no longer want

slide 5 Extra uses of V.C.
- present other areas where Version control would be useful

Git Pres topics outline (cont.)
slide 6 What is a Repo
- A repo is a folder containing version history information
  - contains files
  - contains a log of changes to the files in the repo folder
- Repo has its own way of saving state separate from files (covered later)

slide 7 centralized vs. decentralized
- these are the main two categories in which version control tools fall
- main difference is the location of the repo/how many repos
- Example of centralized = SVN
- Examples of de-centralized = git, mercurial (hg)

slide 8 centralized diagram
- one repo on a server
  - this one is a full repository containing full version history
- individual work stations have a copy of the repository
  - does not have full version history
- any version control actions have to access the full remote repo
- Drawbacks
  - bc version control actions have to access remote repo can be slow
  - no connection = no version controlling
  - only one copy of the full repo = dangerous

Git Pres Outline (cont.)

slide 9 Decentralized Diagram

- individual workstations + server all contain a copy of the full repository w/full version history
  - don't technically need the server copy due to everyone having a full repo
- don't have to have a connection to do most version control actions
  - if connection breaks, can keep version controling
- if the server's copy of the repository is lost there are multiple backups with full version history
- Claims of version control commands being faster because they don't all have to access the remote repository

slide 10 Save vs. commit

- save is to file as commit is to repository
- saves affect a single file and cannot themselves be reverted
- commits can include multiple files and can be reverted
- commits are like saving the state of the whole repository

slide 11 commit staging process

- start the process of creating a new commit after a file(s) has been altered + saved
- need to add the files that you want in the commit to the stage
- If a wrong file has been added the whole stage can be cleared

Git Pres topics Outline (cont.)

slide 11 (cont.)

- when all the wanted files are staged you can create the commit
  - the commit will not include/alter or get rid of unstaged altered files
- this process allows related changes to multiple files to be commited together
- Git knows what files have been altered from the past commit

slide 12 conceptual push/pull

- local workstation contains a full copy of the repo
- it is on the local repo that commits are made
  - making a commit will not change the remote repo (on server)
- The process of transfering commits to the remote repo is known as pushing (ie. you are pushing your local commits up to the remote repo)
- pushing commits up to remote repo by itself will not include the chages to the repo on other local work stations.
- The process of transfering commits from the remote repo to a local workstation is known as pulling (ie. you're pulling the remote commits down to your local repo)
- If someone made a change to a file and pushed that change then someone else tried to change that same spot of code and tries to push their change a merge conflict will occur during person2's push

Git pres topics Outline (cont.)

slide 12 (cont.)

merge conflict

- git will block person 2's push and mark the conflicking spot
- Once the file has corrected (even if that just includes removing the marks) the file can be commited & pushed

slide 13 Git commands / git clone

- if not on mac/linux & command is attempted it will show you how to obtain
  - apt-get on linux
  - x-code command line on mac
- link w/ instructions for windows
- don't want to focus too much on acquiring git
- clone is how you make a new repository copy from another repository
- command setup → git clone <remote repo location>
- if no new location specified it will put the new copy in whatever folder the terminal is currently in
- all other git commands have to be used from SOMEWHERE inside the repo

slide 14 push/pull

- git push → pushes changes to remote repo
- git pull → pulls changes from remote repo
- concepts previously discussed

Git pres topics outline

slide 15 git status/add/reset
- status will show the difference in number of commits between local & remote repo
    - which files have been altered
    - altered files staged for commit or not
- add will add an altered file to the commit stage   git add <file name>
- reset will clear the commit stage
    - careful with additional arguements
- three commands help set the commit stage

slide 16 git commit
- will make a commit of the staged files
- add a message using -m "..."
- if forgot to use -m use :q to back out and try again
    - it puts you in a vim editor; if you know how to use just write & quit after writing your message.

slide 17 git diff/help
- diff like status but instead of showing which files are altered, will show the actual changes that had been made to the file
- help to see a list of commands available & help <command> to see even more detail about that command

Git pres topics Outline (cont.)

slide 18 Git commands diagram
- clone, push, pull interact w/ remote repository
- status, add, commit, diff, ect interact w/ local repository
- pull before starting work & push after finishing
- normal workflow commands are status, add commit, diff

slide 19 Github overview
- Github ≠ Git
- Github is a remote server provider for the central repo
    - like a cloud service
- offers web page access to repo and easy ways to view it & interact using git
- only have to pay if wanting private repos
    - all other repos are public
- naturally lends itself to open source projects where people use git tools to help and contribute to eachothers projects
- no size limit on repo

slide 20 Moving Forward
- branching & merging for advanced project control
- forking repos & pull requests on Github
- use ssh to communicate w/ Github or remote repo in general
- attach links

Git pres topics outline (cont.)

slide 21 conclusion
- went over how version control works and why it is useful
- what is the difference between the two main types of version control
- how does git work conceptually?
- how to use command line to implement Git concepts
- what is Github
- further directions for study