1. Intro
   1. Covers
      1. What is Git
         1. Version control system
         2. Free
         3. Open source
      2. What is GitHub
         1. Commercial platform to host files through GIT
         2. Allows for easy collaboration
         3. Nice visual user interface
         4. Free for public files
         5. Subscription for private files
   2. Overview
      1. Docs Like Code
      2. Version Control
      3. Getting started with GIT and GITHUB
      4. Using the command line
      5. GIT concepts
      6. Adding, renaming, and deleting files
      7. Going back in time
      8. GIT commands
         1. Push
         2. Pull
         3. Tag
         4. Stash
         5. Branch
         6. Merge
         7. Clone
         8. Rebase
      9. Handling problems
      10. Forking
      11. Documentation and next steps
2. Docs Like Code
   1. Covers
      1. What does “Docs like code” mean?
         1. Treat writing documents same way as code writing
         2. Documentation written in text format, not binary
         3. Use version control tools (Git, GitHub, etc.)
         4. Use a review process similar to code reviews
      2. Text formats for documents
         1. When a document is in text format, it can be opened and read in a text editor
         2. Some formats are designed for this:
            1. Markdown
            2. ReStructuredText
         3. Some formats are kind of readable
            1. XML
            2. HTML
         4. Binary formats (PDF for example) are not readable
      3. XML
         1. eXtensible markup language
         2. Combination of text and tags
         3. Used for DITA (Documentation tool)
         4. When software is automatically merging two versions of an XML file, it can mess up the structure
         5. Not ideal for “Docs like code”
      4. Markdown
         1. Simple Markup Language
         2. No Tags
            1. \*\*bold\*\* instead of <strong>bold</strong>
         3. Has many variations (called “flavors”)
         4. GitHub-flavored Markdown is very popular
         5. Works well for “Docs like code”
      5. ReStructuredText
         1. Simple markup language
            1. More complex than Markdown
         2. Originally came from documenting Python through a tool called Sphinx
         3. No Tags
         4. Only one variation (no flavors)
         5. Can extend its functionality
         6. Works well for “Docs like code”
      6. AsciiDoc
         1. Similar to reStructuredText
         2. No tags
         3. Only one variation (no flavors)
         4. Can extend its functionality
         5. Works well for “Docs like code”
      7. Gitbook.com - \*\*Gitbook hosts files\*\*
3. Version Control
   1. Covers
      1. What is version control?
         1. Modifications/Edits to documents is creating a new version
         2. Useful to keep copies
         3. Each file req a new name
         4. Gets confusing when multiple people are working on a document
         5. With VCS, multiple users manage the versions as if they all had the same name
         6. Also called revision/source control systems
      2. Collaboration
         1. Often work on documents as part of a team
         2. Different people make changes to the document
         3. Version control systems can help manage this
         4. Documents are hosted remotely, but worked on locally
      3. Older styles of source control
         1. Earliest: one person checks out at a time
         2. Later: multiple people check out simultaneously – Merge capabilities
      4. Git
         1. Created by Linus Torvalds in 2005 (Linux creator)
         2. Name doesn’t mean anything
         3. Rather than storing every version, it stored the differences between versions
            1. Much smaller
         4. Every computer stores all versions
            1. Older systems only stored what was being worked on
      5. GitHub and other platforms
         1. Git manipulates filed using the file system
            1. File content changes with Git commands
         2. Separates commit and push
            1. Older systems: check in marks and uploads files
            2. Git: marking and uploading are two separate things
         3. You can create a git remote repository on any server
         4. However, commercial platforms make it easier
         5. GitHub:
            1. Most popular
         6. Others:
            1. Provide better security, better integration with other tools, etc.
            2. BitBucket, GitLab, Beanstalk, etc.
4. Getting started with Git
   1. Covers
      1. Command line vs GUI
         1. Command Line
            1. Type commands in terminal
            2. Gives you full power of Git
            3. Most people (even non-technical) eventually use command line
         2. GUI
            1. An app with a user interface
            2. Not that much less confusing than command line
      2. Installing Git on Mac and Windows
         1. Text Editor
            1. Definitely want a text editor working with files
            2. Some ideas (free editors)

Mac: Atom (http://atom.io/)

Windows: Notepad ++ (https://notepad-plus-plus.org/)

For Windows, download and install before installing Git

* + - * 1. Mac

Git already installed

Only need to use terminal app

* + - * 1. Windows

Requires installation

* + - 1. Use special terminal app called “git bash shell” (https://git-scm.com/downloads)
    1. Making sure you have it installed
       1. Easiest way to test is to have git show you it’s version
          1. git --version

1. Getting started with GitHub
   1. Covers
      1. What is GitHub
         1. A way to host content remotely
         2. Multiple users can read and/or write versions
         3. Everyone uses Git to access it
      2. Creating an account
         1. Accounts on GitHub are free
         2. It’s free to host files if those files are open to the public to view
         3. Most companies will pay a subscription for privacy
         4. Open-source projects are often public
      3. Creating a repository
         1. Repository (Repo)
            1. Repo is a place that holds a directory structure (folders) with files
            2. Contains not just current versions, but all versions and history
            3. Typically exists both remotely and locally, although not necessarily in sync
         2. Can be created locally with Git and then “pushed" to the remote platform (GitHub)
         3. Easier to create in GitHub and then “clone” it locally
      4. Cloning a repository
         1. Once repo is created, GitHub shows you it’s path
            1. Click the Clone or download button
         2. Use this in a git clone command
            1. git clone https://github.com/user/repo
2. GitHub authorization
   1. GitHub has changed their authorization, so username and password no longer works
      1. Personal access token must be created
         1. Full description @: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens
3. Command Line
   1. Covers
      1. What is the command line?
         1. Before 1973, computers did not have GUI’s
            1. Xerox invented the GUI, and Apple ran with it
         2. Instead, you interacted with a computer through text
            1. Type commands
            2. Read responses
         3. This is called the command line
      2. Useful Unix commands
         1. Unix commands move around directory structure (folders and files)
            1. pwd: tells which directory user is in

Stands for “Present working directory”

Folders in Windows and Mac OS are called directories in Unix

Returns the path to which directory you are working in

Note that levels are separated with forward slashes, even on Windows

* + - * 1. ls: tells which files and folders are in current directory

Short for “List”

* + - * 1. cd: moves you to a new directory

Short for “Change Directory”

Changes your current directory to a new directory

Special directory names:

. (one period): your current directory

.. (two periods): up one directory

~ (tilde): home directory

* + - * 1. mkdir: creates a new folder

Short for “make directory”

Creates a new directory

Does not automatically move user to new directory (need cd command)

* + 1. Command line shortcuts
       1. Tab: fills in rest of a name
          1. Stops if it doesn’t have enough information to finish
       2. Up arrow: repeats previous command
          1. Multiple up arrows keep going back in history and vice-versa
          2. You can edit the line to make changes
    2. Structure of Unix commands
       1. General Command Structure
          1. Command –o --option argument(s)
       2. Options modify the command in some way
       3. One dash for one-letter options
       4. Two dashes for multiple-letter options
       5. Argument is for additional data
          1. Options before arguments
          2. Because elements are separated by spaces, sometimes arguments need to be in quotes
    3. Example git commands
       1. Git commands start with git and then a command
          1. git config --list
          2. git config --global user.name “first name last name”

Quotes around name so Unix knows it’s one argument (arguments seperated by spaces)

* + - 1. Example one-letter option
         1. git commit –m “Marketing changes”

1. Git Concepts
   1. Covers
      1. Git’s four file stages
         1. Unstaged
            1. Local change made to file (or added/deleted a file)

Used when user not sure changes need to be kept

* + - 1. Staged
         1. A local change is marked as something to commit and save in Git

Used when pretty sure but not fully sure of changes

* + - 1. Committed
         1. File committed and version saved in Git

Used when changes definitely want to be kept

Each commit has a description to help recall of reason for version

* + - 1. Pushed
         1. File has been uploaded to the server so others can access it

Used when user wants to share changes with others

* + - 1. \*\*Analogy\*\*: writing a book
         1. Unstaged: pages written in notebook as preliminary draft
         2. Staged: Putting a checkmark on each page writer wants to save
         3. Committed: once reviewed pages scanned and stored with brief description
         4. Pushed: upload scanned pages to server where others can access
      2. How files moved through stages
         1. Unstaged: just make changes on computer
         2. Staged: Use *git add* command
         3. Committed: Use *git commit* command

Add a description

No need to tell which files, because it commits the staged ones

* + - * 1. Pushed: Use the *git push* command

Command automatically pushes the committed files that haven’t been pushed already

* + - 1. Text and Binary Files
         1. Git works on both text and binary files

Text: documents

Binary: images, video, audio, etc.

* + - * 1. Both are treated the same
        2. The only difference: merging different versions

Text: software can reliably figure out how to merge

Binary: extremely difficult to figure out how to merge

Git will require one file chosen over another