1. Rebasing
   1. Covers
      1. What is rebasing?
         1. Rebasing reorganizes your history so that it’s as if you branched later
      2. When would it be used?
         1. Eg – branch created to update documentation before spelling changes from US to UK spelling
         2. Pull request will show both changes
            1. Making it more difficult to see user changes from first branch
         3. Rebased branch will pick up the US to UK changes and pull request show accurate changes
            1. Much easier to understand
      3. Rebase vs Merge
         1. Merging from original branch will have the same effect
            1. History is simpler with rebase
            2. Rebase “rewrites history” to make it look like user branched later than user has
      4. Git Command
         1. From working branch, use git rebase command followed by name of branch rebasing from
            1. Eg. Command: git rebase master
2. Handling Problems
   1. Covers
      1. How Git can create difficult situations
         1. With all the branching and merging, sometimes git won’t let user do something needed
            1. Won’t commit, push, pull, etc.
         2. Try to figure out how to fix it
            1. Git provides suggestions
            2. Google for internet can help

Git doesn’t have an undo function

Git doesn’t have an trash folder – deleted items are gone

Unless they are committed, then the user can revert to that commit before deletion

* + - 1. Sometimes it’s faster to not try to fix the problem
         1. Copy the changed files to folder outside of repository
         2. Revert changes to point where git works again
         3. Manually merge the changes back in
    1. Overall strategy for handling problems
    2. Reverting unstaged changes
       1. Git reset --hard command gets rid of uncommitted changed
       2. Git reset --hard/branchName and Git reset origin helps to update local version to remote version
          1. branchName is name of remote branch
          2. If local messed up but remote good
    3. Reverting unpushed commits
    4. How to start over if needed
       1. If nothing works, here’s a last resort
          1. Copy the changes to files outside the repository
          2. Clone the repository again into a new folder
          3. Manually merge changes back in
          4. Delete local repository

1. Ignoring Files
   1. Covers
      1. Why would files be ignored
         1. Software consists of source and build files
         2. Tools take source files and create build files, which are the final product
         3. For documentation source files may be Markdown but build files would be HTML
            1. Only source files in Git
         4. Build files should be ignored b/c they take up too much space and are redundant with source files
            1. User may also have local files they would like to keep local

e.g. text files that have nothing to do with the build

* + 1. The .gitignore file
       1. Create file named .gitignore in top level directory of local repository
          1. List any files to ignore (“wildcard” characters can be used)
          2. List any folders to ignore
          3. Add file to repository and commit

Note that because it starts with a period, it will be considered a hidden file

* + - 1. File structure
         1. Any line starting with # is a comment and doesn’t do anything
         2. List any file to be ignored using a path
         3. \* can be used for a wildcard (meaning anything)
         4. List any folder by ending with a /

e.g. - # Ignore the file notes.txt in the tmp folder : temp1/notes.txt

# Ignore all .txt files : temp2/\*.txt

# Ignore everything in the tmp folder : temp3/

File is not part of repository – DON’T add . Then don’t git commit

* + - 1. Create .gitignore file
         1. Mkdir e.g. - html

Create file

Save in created directory

Git status to see file

\*\*Don't add . Or git commit (not part of repo)

* + - * 1. New document

First line is # comment (ignore files)

Second line directoryName/

Save in top level of repository (same as directory) as .gitignore

* + - * 1. Git add .
        2. Git commit
        3. Git push
        4. Refresh GitHub and file .gitignore shows up
        5. Git status command shows untracked file .gitignore

Shows directories/files not part of repository

1. Forking
   1. Covers
      1. What is forking?
         1. Fork is a copy of repository – allows free experimentation without affecting the original project
         2. Similar to branching
            1. Branching occurs in same repo. Forking occurs across GitHub accounts
         3. Branching is part of Git. Forking is part of GitHub
         4. Pull requests created to marge changes back into the original GitHub project
      2. When to fork
         1. Most common in open-source projects
            1. Anyone can copy repository and make improvements
            2. Pull requests created to incorporate improvements
            3. Project owner can decide whether in incorporate changes
         2. Also used for non-public projects
            1. Team members fork and create pull requests
      3. How to fork
         1. To Fork:
            1. Find project to fork on GitHub
            2. Click Fork button
            3. Choose account
         2. To create pull request
            1. Click “Create pull request” button in GitHub
            2. Follow instructions
         3. Fork process (in GitHub)
            1. Search project and select
            2. Click “Fork” button
            3. Edit on read.me (example)
            4. Commit changes
            5. Click “Pull Requests” tab
            6. Create pull request
2. Creating documentation and next steps
   1. Covers
      1. Documentation systems that can be used with Git
         1. GitHub, GitBook, Jekyll, Sphinx, AsciiDoctor
            1. Some projects are documented in GitHub itself
            2. The README.md file is displayed when looking at GitHub project
            3. Additional documentation can be created w/ Markdown or Restructured Text (reST) files
            4. Very bare-bones
            5. No sophisticated navigation
         2. GitBook
            1. designed specifically for documentation
            2. Can choose own domain
            3. Free to two users (Public/Private)
         3. Host files in GitHub
            1. Webmaster uses local copy of repo for files
            2. Runs scripts to turn into HTML

Markdown: Jekyll

ReStructured Text: Sphinx

AsciiDoc: AsciiDoctor

Some tools like ReadMe can pull from GitHub and generate documentation

* + 1. How to keep learning Git and GitHub
       1. Next Steps
          1. No advance courses for writers but have enough knowledge to take developer course
          2. Read up on “Docs Like Code”