1. Tags
   1. Covers
      1. What is a tag
         1. Sometimes commits need to be labeled as special
            1. This is called a tag
         2. E.G. - Tag a commit that corresponds to the files that are part of a release
         3. A tag is a label attached to a commit to make it easier to find
         4. Tags have a name and a description
         5. Tags are applied locally. Use git push to upload it to GitHub
            1. Git Commands

Git tag –a: adds a tag with name and description (space followed by –m for message)

Git tag: lists all tags

Git checkout tags /<tagname>: to checkout a tag

Git push origin <tagname>: to push a tag

* + 1. How to add a tag
       1. git log --oneline
       2. git checkout : to change HEAD
       3. ls to verify file t/b tagged
       4. git tag –a (name of file) -m “message”
          1. Adds the tag
       5. git tag: lists/displays tag
       6. git checkout master : returns HEAD
       7. ls: shows no tag
    2. How to see what tags exist
       1. Git checkout tags/ <tagname>
       2. Ls: shows tag
    3. Pushing a tag
       1. Git push origin: moves tag to github
          1. Go to github click on branches/tags and find tag

1. Pull
   1. Covers
      1. Collaboration and version control
         1. Main advantages of using version control
            1. Going back in time
            2. Collaboration
         2. Version control lets multiple people work on a file at the same time
            1. But each have their own copy
         3. User can control how those separate changes are incorporated
      2. What is a pull?
         1. git pull downloads all commits that have been pushed to the repository
         2. git pull can only be used if files committed
            1. No merging unless user is at a good stopping point
      3. Handling changes
         1. Remote file hasn’t changed since last pull
            1. Leaves local file as is, whether user changed it or not
         2. Remote file has changed. Local file hasn’t
            1. Replaces local file with remote file
         3. Both remote and local files have changed
            1. Remote file is merged into local file

Bad practice to edit files directly in GitHub

* + 1. Merging
       1. Git compares line-by-line, noticing changes
       2. If not changed locally or remotely line stays
       3. If line changed, either locally or remotely, line is changed
       4. Conflict if line changed both remotely and locally
    2. Conflicts
       1. If line changed both remotely and locally, Git doesn’t have enough information to know which is correct
          1. Git will notify of conflict
          2. Marks old and new lines with <<<and>>> symbols
          3. Edit the files and remove <> symbols
          4. Tell git conflict has been resolved
    3. Binary files
       1. Git can only merge text files
          1. Git goes line by line
       2. Binary Files (images, audio files, video files, etc.) cannot be merged
       3. A version must be chosen
       4. If merge needed, it can be done manually
    4. Fetch
       1. Git fetch command
          1. Will download remote files but not merge them
          2. Will need git merge command to merge
          3. Git pull does both automatically, but git fetch and git merge offer finer control

1. Branches
   1. Covers
      1. What are branches?
         1. Sometimes it’s useful to create set of versions for particular purpose
         2. New branch needed for:
            1. Adding a new feature
            2. Fix a bug
            3. For documentation
         3. Create new branch when changes made for that purpose
         4. Once everything complete, branch can be merged into original branch
      2. The master branch
         1. Created when repository created
            1. Root branch
         2. Typically don’t work on master branch unless working on small project
         3. Typically work on own branch and merge back to master
         4. Can add more branches from branch being worked on
      3. Branch naming
         1. Branch names can use letters, numbers, dashes, periods, and forward slashes
         2. Case sensitive
         3. Most common is to use lowercase with dashes between words
            1. e.g. api-docs
         4. Slashes used to create groups of branches
            1. e.g. v1.2/api-docs, v1.2/login-fix
      4. Creating a branch
         1. Git branch *branch-name* : creates a branch named *branch-name*
            1. Branches off current branch
            2. But user stays on current branch
         2. Git checkout –b branch-name : creates a branch and moves user to that branch
            1. Handy shortcut
      5. Changing branches
         1. Git checkout branch-name : moves to the branch named branch-name
         2. Can only move to another branch if all files are committed
            1. User should be at a good stopping place
            2. If not ready to commit user can use git stash command
         3. Git branch command displays all branches and which one user is currently on
      6. Deleting and pushing
         1. Git branch –d branch-name command deletes the branch named branch-name
            1. Only do this once changes are merged, or user is sure branch not needed anymore
         2. Git push command uploads everything to the server, including new branches
            1. Until user pushes, all new branches are local only
         3. Once pushed, branch can be deleted on GitHub
            1. Changes can be viewed with git pull command
2. Stash
   1. Covers
      1. What if user not ready to commit?
         1. If not ready to commit, Git provides a way to “stash” your changes
         2. User reverts to last commit
         3. User can switch to another branch
         4. When ready, user can come back and “pop” stash
            1. The Stack

When stash created, it’s at top of list

When “popped” it is removed from list

Now second most recent stash is top

\*Like a stack of dishes\*

* + 1. Stash commands
       1. Git stash : create a new stash and revert to most recent commit
       2. Git stash list : list most recent stashes
       3. Git stash pop : restore changes to most recent stash on stack
          1. Changes are unstaged
          2. Removes stash from top of list
    2. Listing stashes
    3. Popping a stash
    4. Naming stashes
       1. By default, stashes don’t have names
       2. Good idea to name stashes, especially if doing more then one
       3. Git stash save “stash-name” will save stash under a name
          1. Will see the name with *git stash list* command

1. Merging Branches
   1. Covers
      1. What does it mean to merge a branch
         1. Create new branch in order to make changes
            1. Could be main, or could be another branch
         2. Once working branch is correct, it can then be merged back to branch it came from
         3. Now original branch has changes made
      2. Merging through command line vs. GitHub
         1. User can merge locally using Git command line
         2. Can merge remotely using GitHub
            1. GitHub is easier and provides visual tools to lead through merge process

Most recommended

Changes need to be pushed

* + 1. Pull requests
       1. When ready to merge, a pull request needs to be created
       2. Changes are reviewed (if working on massive project with a supervisor) before approved
       3. Merge automatically if possible
       4. If there is a conflict, GitHub will display them before merging
    2. Conflicts
    3. Squashing commits
       1. When user merges, they can either
          1. Merge pull request

Every commit on new branch becomes a commit on original branch

* + - * 1. Squash and merge pull request

All commits that were in the new branch are squashed down into one commit on the original branch

Squash and merge simplifies history

* + 1. Merging with git command line
       1. The git diff command compares two branches
          1. Git diff branch1...branch2 (three dots needed, not an ellipse)
       2. The git merge command can merge the branch locally
          1. Change to the branch to merge *into*
          2. Git merge newBranch

1. Cloning
   1. Covers
      1. What is cloning?
         1. Making a local copy of remote repository
         2. Creates copies of all files and history locally
         3. Creates both visible and hidden files
      2. How to clone a project on GitHub
         1. Click green “Code” button on GitHub repository
         2. Copy URL (under HTTPS)
         3. Use git clone command with URL
      3. HTTPS vs SSH cloning
         1. Can clone with either secure HTTP or secure shell (SSH)
         2. HTTPS is simpler
         3. Some feel SSH is more secure
            1. Requires generating a special SSH key and providing the passphrase
      4. What is the origin in Git commands?
         1. Origin in Git is just a shortcut for the URL where you cloned the repository from
            1. Git push origin command same as:

Git push (e.g. HTTPS URL)

\*\*In theory pushes can be used to change other repos\*\*

Very unusual