Instructor Notes

By: Brian Hankins

Outline:

1. Git / GitHub Basics - Non-Forked Repos
   1. Create GitHub repositories (aka: repos)
   2. Clone repo to local computer
      1. git clone <your repo clone link>
   3. To add files into staging
      1. git add \* or <file name>
      2. Remove file from staging: You can use *git reset <filename>* to remove a file or files from the staging area.
      3. git status
         1. File will be green if staged, red it not.
   4. To commit all files that are staged
      1. git commit -m “message here”
      2. To show logs of commits
         1. git log --summary
   5. git push origin master (Note: This reads like: git push ‘yourGithubRepo’ from the ‘masterBranch’
      1. Origin == GitHub & master == master branch
      2. git push -u origin master : saves the -u origin master parameters so next time you can do just: git push
      3. If you you receive an error: ‘could not read from remote repository’, you can add the origin manually by: git remote add origin <paste repo clone link>, then retry git push origin master
2. Forked Repos
   1. Go to <https://github.com/qccoders/Git_Good>
   2. Click on the clone button and copy the clone file path
   3. git remote add upstream <file path copied from the step above>
      1. git remote -v
         1. Shows a list of remote upstreams
   4. git pull upstream master
      1. git stash: Sometimes when you go to pull you may have changes you don't want to commit just yet. One option you have, other than committing, is to stash the changes. (Note: changes must be in staging - so do a git add \* first.
      2. 'git stash pop' to re-apply your changes after your pull.
   5. Push changes to GitHub when ready: git push origin master
3. Branch
   1. Create new branch
      1. git branch <branchname>
      2. Switch to that branch
         1. git checkout <branchname\_from\_above>
         2. git status
            1. Shows what branch you’re currently on
            2. git branch or git branch -a will show branches
         3. Hints: git status
            1. Staged: Files are ready to be committed
            2. Unstaged: Files with changes that have not been prepared to be committed.
            3. Untracked: Files aren't tracked by Git yet. This usually indicates a newly created file.
            4. Deleted: File has been deleted and is waiting to be removed from Git.

*KEY: [I] - instructor performs action; [A] - audience performs these steps*

Interactive Setup:

1. Have everyone **fork** and then **clone** repo to a new directory so we start fresh
   1. Git\_Good on repo ( <https://github.com/qccoders/Git_Good> )

Interactive:

1. Setup
   1. [I] Use randomquotes.txt file
      1. [A] Setup upstream and verify they are on master branch
         1. git remote add upstream https://github.com/qccoders/Git\_Good.git
      2. [A] Pull repo down using proper commands
         1. git pull upstream master
      3. [A] Push to their personal github repos.
         1. git push origin master
   2. [I] Make a change to a file
      1. Show what's on GitHub so [A] can see they are behind..
      2. [A] Have them pull the change down.
      3. [A] Push to Github so they are now even with origin
   3. Unstage a file:
      1. [A] Make changes to a file
         1. Stage those changes (don't commit!)
      2. git reset <filename>
      3. Go back to original: git checkout <filename>
      4. Undo file changes in editor
2. Bad Commits to GitHub
   1. [A] Make change to file and commit to GitHub
   2. Remove commit by:
      1. git log
         1. Find 2nd to last commit
      2. git push origin +[ref]:[branchName]
      3. git reset --hard origin/master
         1. Removes all changes
   3. If all else fails: delete the directory and re-clone the forked GitHub repo
3. Branching
   1. [A] git checkout <branch name>
      1. See branches by: git branch (or include upstream branch by git branch -a)
      2. Switch branches: git checkout <branch name>
      3. Make changes to a file
      4. git diff <file\_name>
      5. Stage and commit file on test branch
      6. git diff <branch 1> <branch 2>
      7. git status
   2. [A] Merge branches
      1. Change to master branch: git checkout master
      2. git diff <branch 1> <branch 2>
      3. git merge <branch\_name\_not\_master>
   3. [A] Committed code to wrong branch
      1. Undo last commit, but leave the changes
         1. git reset HEAD~ --soft
            1. Puts file back into staging

To remove from staging: git reset

If errors, may have to do a: git rm --cached -r

* + - 1. git stash
    1. Move to correct branch
       1. git checkout <name of correct branch>
       2. git stash pop
    2. Add files as normal and commit
       1. git add \* (or individual file name)
       2. git commit -m “message here”
       3. git push origin master
  1. [A] Delete branch
     1. git branch -d <branch\_name>

1. [A] Conflicts
   1. [I] Create a line for each person
      1. Have everyone pull down the changes and push back to their own repo to make even.
         1. git pull upstream master
         2. git push origin master
   2. [A] Make change to line number of a file.
      1. [M] Note: Each person will have their own line number.
   3. [I] Make changes to each lines number above of the master repo (same file)
   4. [A] git pull upstream master
      1. Shows error and aborts - need to either commit changes, stash changes, or discard changes.
   5. [A] Stage and commit changes
      1. git pull upstream master
         1. Shows conflict needing to be resolved in editor.
      2. git status
      3. Add file to staging and commit, and have them push.
   6. [I] Merge PR’s
      1. Now their repos are 1 commit behind...
         1. git pull upstream master
         2. git push origin master
2. Questions?
   1. Pro Tips:
3. Keep bugs, features, etc. on different branches.
4. Push commits often. Typically after a bug fix, end of the business day, feature completion, etc.