Git Version Control

Objectives

- Create a Git repository
- Work with another person and a Git repository
- Understand the basic functionality of Git/version control software
- Able to connect to Git via Eclipse
- Add another plug-in to Eclipse

PRE-LAB work:
If you are not familiar with Git, then run through the Interactive (15 Minute) Git Tutorial (https://try.github.io/levels/1/challenges/1) .

Eclipse now comes with Git!

1. Download the latest version of Eclipse at www.eclipse.org
   (You may need to update your Java version)
2. Unpack the compressed file

If you need more info on using git, see the Git Docs. Google is also your friend.

Git

Complete the following actions:

1. If you do not already have one, create an account on either GitHub or GitLab.
2. Create a PUBLIC git repo on any of the git hosting services listed in Step 1
3. Clone the repo to your local machine
4. Add and commit a simple README file with your name and the name of this assignment
5. Add and commit a few other files (anything you like) to the repo
6. Created a tag named "Practice_Tag_1"
7. Push all changes and tags to your remote repo (note that tags require a special flag to be included in push)
8. Create and checkout a new branch named "Testing_New_Files"
9. Add and commit a few more files to this branch
10. Checkout the master branch
11. Edit and commit at least one of the files in this branch
12. Merge the "Testing_New_Files" branch into the master branch
13. Checkout the "Testing_New_Files" branch
14. Edit and commit at least one of the files in this branch
15. Push all of your changes in both branches to your remote repo (make sure that both branches show up on your remote host)
16. Find a partner or teammate
17. Grant this person write access to your remote repo

18. Clone your partner's remote repo to a new directory outside your original repo
19. Edit the README file in your partner's master branch to list your name as 'Partner: <Name>'
20. Commit your changes
21. Push your changes to your partner's remote repo
22. Return to your original repo (but do not pull from your remote branch yet)
23. Checkout your Master branch
24. Edit your README file to add a line that says "This might cause a merge conflict"
25. Commit your changes
26. Make sure your partner has completed Step 21 (e.g. pushed their changes to your README)
27. Attempt to pull your partner's changes to your repo
28. Resolve the merge conflict if one occurs
29. Commit the merge (assuming a conflict occurred)
30. Push the merged changes back to your remote repo

To get credit for this lab exercise, show the TA and sign the lab's completion log.

Using Git with Eclipse

You can do the following lab exercise connecting Eclipse with Git [optional]. Eclipse is a great IDE used a lot for Java development and also has versions for C++, PHP, etc. We will use EGit here.

Setting up the Remote Repository in Eclipse

31. Goto Help>Install New Software..
32. In "Work with" type in http://download.eclipse.org/egit/updates-1.0 and click Add
33. Give a name (example: EGit) to the Repository and select OK.
34. Now you get two options: Eclipse Git Team Provider and JGit. Select Eclipse Git Team Provider.
35. Click "Next" and then Accept to the terms and conditions and click "Finish".
36. Now, in your web browser visit your GitHub repository/
37. Click on the little clipboard icon. You should find it in the lower righthand side of the window next to a text box entitled HTTPS clone URL. When you click on the icon, it should say "copied!"
38. Open Eclipse and go to the 'Window" menu and choose Open Perspective>Other… then choose "Git Repository Exploring."
39. Click on the little yellow icon in the button bar that says git and has a small blue arrow. When you mouse over it the tool tip should say "Clone a Git repository and add the clone to this view."
40. In the URI field past the URL copied from GitHub
41. Add your GitHub username and password and, if you are on your personal computer, check "Store in Secure Store"
42. Click "Next"
43. Click "Next" again
44. Click Finish. You have just setup the remote repository. You only have to do this once.

Sharing your Project in Eclipse

1. Return to Eclipse and go to the "Window" menu and choose Open Perspective>C/C++
2. Right-click on the project you want to add to git
3. Choose Team>Share Project… from the contextual-menu.
4. Click "Git" and "Next"
5. Choose csc260 as the repository
6. Use the assignment name for the path within repository (e.g. hw1)
7. Click Finish. You have now created a local repository for your project and linked it to GitHub as the remote repository.

Committing and Pushing Changes in Eclipse

1. When you want to commit and push changes to the server. Right click on your project
2. Choose Team>commit...
3. If you have created any new files since the last commit the will be listed in the bottom of the window. Make sure that you check of all the files you want to add the the repo.
4. Add a comment that describes the state of your code.
5. Click "commit" if you just want to commit your code locally or commit and push if you want you changes uploaded to GitHub (I recomend doing a commit and push all the time.)
6. A dialog will pop up when files are done being pushed. Click ok.

Tagging the Repository for Grading in Eclipse

1. When you want to "turn something in" right-click on project
2. Choose team>advanced>tag
3. Use the name of the assignment as the tag name e.g. hw1
4. Have the message something "The final version of hw1 for submission"
5. Right-click on project (yet again)
6. Goto Team>Remote>push
7. Click next
8. Click "Add all Tag spec"
9. Click finish. Eclipse will push the tag to github so I know which version of the repository I should evaluate.

More information about version control

- RCS: http://www.gnu.org/software/rcs/rcs.html
- CVS: http://savannah.nongnu.org/projects/cvs
- SVN: http://subversion.apache.org/
- Git: http://git-scm.com/ Interactive Tutorial Illustrated Tutorial
- Mercurial: http://mercurial.selenic.com/
- Bazaar: http://bazaar.canonical.com/en/
- Comparison of VC systems
- *Read the History of Bazaar: http://www.stationary-traveller.eu/pages/bzr-a-retrospective.html*
- *Read about security in DCVS & Git: http://mikegerwitz.com/papers/git-horror-story*