Redi Cako

CSC 475 Distributed and Cloud Computing
Introduction to Git and GitHub

[redson87 redson87@gmail.com]

1. **What is a Version Control System?**
   Version Control System is a system that records changes to a file or set of files over time so that we can recall some specific versions later if needed. Basically as we progress on a project we make numerous changes to reach our final product and Version Control System helps us keep track of the changes we constantly make. A VCS also allows us to revert certain files back to a previous state, revert an entire project back to a previous state, review changes made over time, see who last modified something etc.

2. **Define the following terms used in Git.**
   a. **Git Bash**: Is a command line application of the VCS.
   b. **Git GUI**: Is a user interface based version of the VCS.
   c. **Working directory**: It's the directory where our files and data will be. The ones we are currently working on.
   d. **Index (staging area):** Before committing the data, this is where Git will look to get a snapshot. After you we use the "add" command our files are in the index are.
   e. **Local repository**: Is where all the files will be after we have used the "commit" command to get a snapshot. The commit or the snapshot will be saved here. your local computer
   f. **Remote repository**: Remote repositories are versions of your project that are hosted on the Internet or a network somewhere. A place where files can be accessed from any computer, they are not just in your local one.
   g. **SSH Keys**: Is essential into uploading code in our GitHub accounts. This key basically sets up a trusted relationship between our computer and our online account, for us to upload code.
   h. **Branching**: Every project will have a master copy. Branching can help us have copies of it and edit them as we go. Different teams can work on different part and those would be the branches.
   i. **Merging**: before uploading our version we make a "pull" and sync all the data and later we basically add our data to the project in order to merge it all together.
   j. **Cloning**: is when we download a project into our local computer from a remote repository.

3. **Use the online resources to provide a brief description for each of the following Git Bash commands.**

   a. **git init <project>**

Create an empty Git repository or reinitialize an existing one. This command creates an empty Git repository - basically a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files. An initial HEAD file that references the HEAD of the master branch is also created.

b. git add <file>

Add file contents to the index. This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

The "index" holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the contents of the next commit. Thus after making any changes to the working directory, and before running the commit command, you must use the add command to add any new or modified files to the index.

c. git add .

The important point about "git add ." is that it looks at the working tree and adds all those paths to the staged changes if they are either changed or are new and not ignored, it does not stage any 'rm' actions

d. git commit –m "msg"

Record changes to the repository. Stores the current contents of the index in a new commit along with a log message from the user describing the changes.

-m <msg>

Use the given <msg> as the commit message. If multiple -m options are given, their values are concatenated as separate paragraphs.

e. git status

Show the working tree status. Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked by Git

f. git log

Shows the commit logs. The command takes options applicable to the *git rev-list* command to control what is shown and how, and options applicable to the *git diff-** commands to control how the changes each commit introduces are shown.

g. git diff

Show changes between commits, commit and working tree, etc. Show changes between the working tree and the index or a tree, changes between the index and a tree, changes between two trees, changes between two blob objects, or changes between two files on disk.

h. git diff –cached

To see what you've staged so far and the differences.

i. git remote add origin …

To add a new remote Git repository as a short name.

j. git push –u origin master

This is the command that commits your data to the remote repository.

k. ssh-keygen –t rsa –C …

To generate a new SSH key. This command is followed usually by an email address.

l. **ssh –T git@github.com**

This command sets up and authenticates the SSH key.

4. **What is your experience in using (1) GitHub with Git Bash vs. (2) GitHub Windows Client (GUI)?**

   In Git Bash we are not limited by an interface, meaning that we have more control over what we want to do. Although both can get the same result in the end, in Git Bash we learn more in depth of how to manage and control the data for our VCS. The GUI version hides a lot of commands.