Version Control with Git
ME 461 Fall 2018

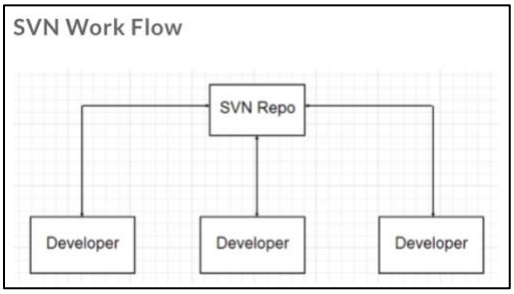
0. Contents

Introduction
Definitions
- Repository
- *Remote* Repository
- *Local* Repository
- Clone
- Commit
- Branch
- Pushing
- Pulling

Create a Repository
Clone a Repository
Commit
Push
Pull
Log

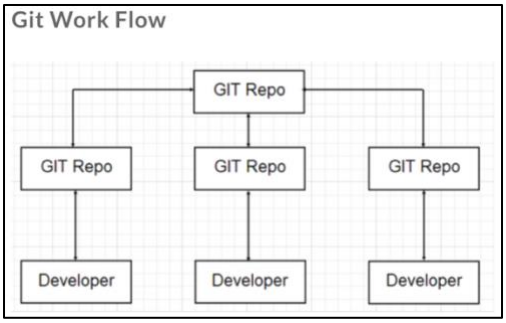
1. Introduction

Version control (also known as source control) is a method for tracking changes to a set of files. It is software that runs on your PC recording all the changes made to these files over time such that you can view the history and progress of your files. Three widely used version control systems (VCS) are Subversion (SVN), Git, and Mercurial (Hg). SVN is a centralized VCS where you will likely need internet access to view the file change history.



'Repo' meaning Repository is the location of all your files and file history. An SVN Repo is usually a remote online repository hosted on a web server. Git and Mercurial are a more decentralized VCS as they use local repositories to store your files and file history (meaning you won't need internet access to view your history of file changes).

Back to Top



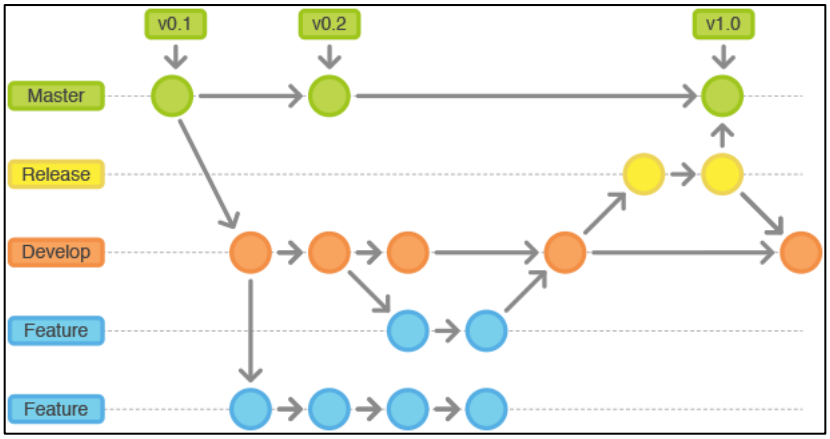
The difference between Git and Mercurial is mostly the user-interface and choosing between the two usually comes down to personal preference. For this course, we will be using Git to *commit* program file changes to our local repositories, *push* commits to our remote repositories, and *pull* all commits to stay up-to-date with our current programs.

2. Definitions

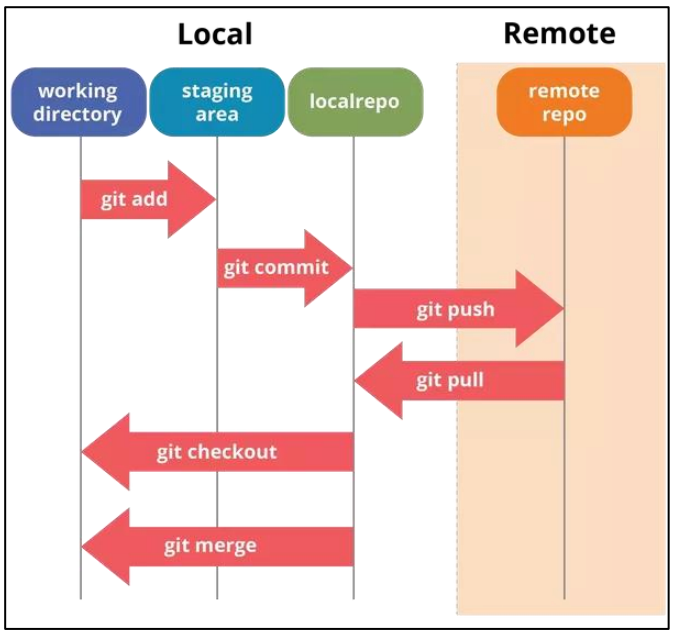
- **Repository**
  - Repository is a fancy name for a folder. It is the hierarchy of all subfolders and files that you wish to use with version control. For this course, all the program files for each lab will be saved to a repository (shared between you and your partner).
- ***Remote* Repository**
  - Remote meaning not stored on your PC. These repositories are the backups of all your files stored on some online server (like GitHub). These repositories can be accessed by any permissible users.
- ***Local* Repository**
  - These repositories are stored directly on your personal PC with which you directly interact with and make the changes to your files. A local repository is often a *clone* of a remote repository.
  - Example: Google Drive: You can think of your school Google Drive account as a remote repository. You can view your files online and download them as you please. If you've used Google's Backup & Sync where you store those Google Drive files directly on you desktop, that is like a local repository. Google's Backup & Sync is constantly and automatically syncing your local repository with the remote repository.
- **Clone**
  - To clone is to make a local copy of an existing repository. For example, you may clone your remote repository on GitHub to create a full copy locally on your PC. Cloning copies all files and folders as well as all branches with the default branch selected as the working tree.
- **Commit**

- Committing saves your files with the VCS.
- Committing is usually done on your PC (local repository) and it creates a 'snapshot' or checkpoint of your current repository noting all the changes made since the last commit.
- The more commits you make with your local repository, the more documented progress you have in your version history. If you commit often, you give yourself the benefit of being able to go back in time to any commit and revert all files to that point in time.
- How often? One rule of thumb is to commit after you've completed a section of code (e.g. a function).

- **Branch**
  - To branch is to isolate your files in order to make several changes and commits without affecting the original set of files.



  - In this example there are five branches: Master, Release, Develop, and two Feature branches. The idea is that idea branch can progress with their own code without affecting other users work. Then once each users work is complete, the branches can merge together to get a complete working branch (usually Master).
  - Branching is not a requirement for this course, but you are welcome to experiment with branching on your lab code.
- **Push**
  - Pushing sends all your recent commits at the local repository level to your remote repository, thus allowing other users to access those commits.

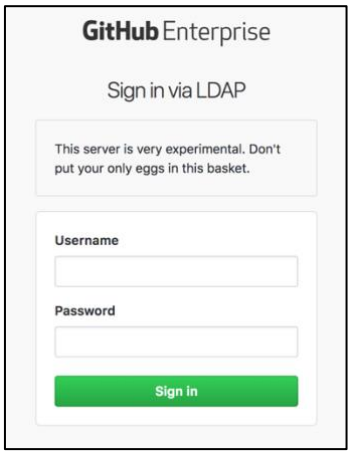


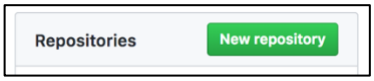
- Pushing will likely require credentials to access the remote repository. For this course, you will need you netID and password to push your commit to your remote GitHub repository.

- **Pull**
  - Pulling 'syncs' all recent changes made to the remote repository with your local repository.
  - To see all the changes collaborative users have made to your shares files, you have to pull from the remote repository to ensure your local repository is up to date.
  - If you have made a critical push to your remote repository and would like to inform other users, you can submit a pull request which will prompt them to pull and sync their own local repositories.

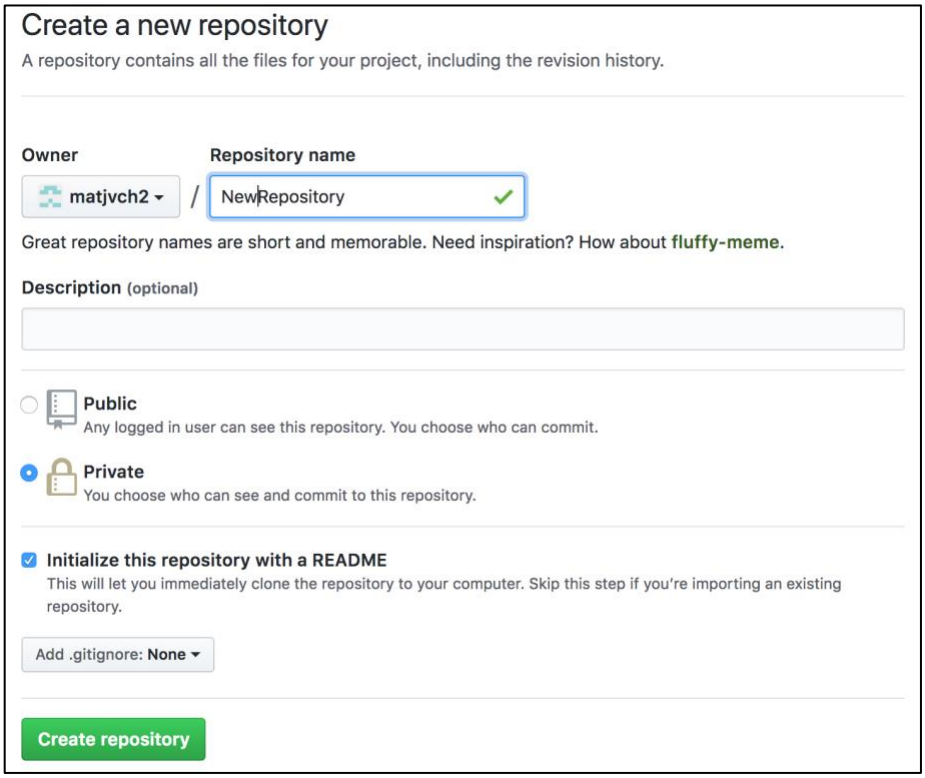
3. Create a Repository

- To create a new repository, use the cs department's GitHub server and login with your netID and password: https://github-dev.cs.illinois.edu/login



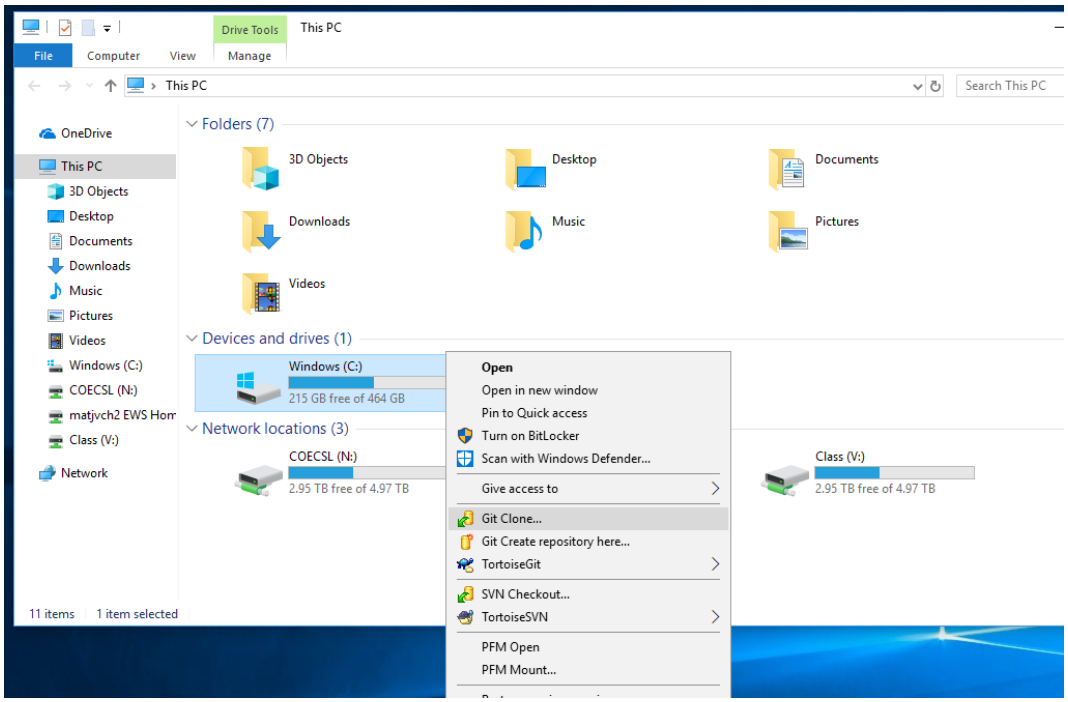


- Name the repository with you and your partner's netIDs: netID1_netID2
- Select private repository and initialize with ReadMe

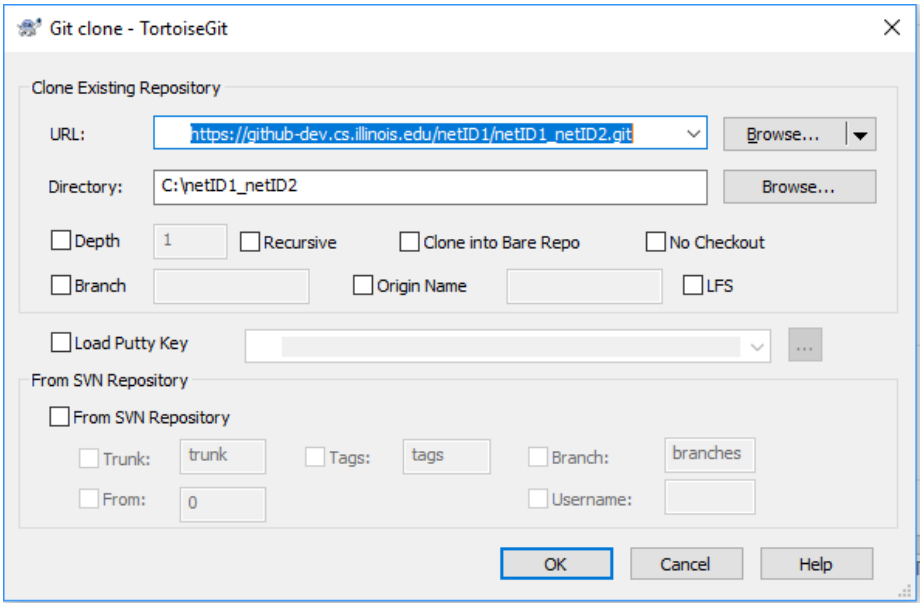


4. Clone a Repository

- Once you've created a remote repository following the directions in the previous section, you can clone it to your PC in the Mechatronics Lab.
- Clone the repository directly on the C:\ drive.



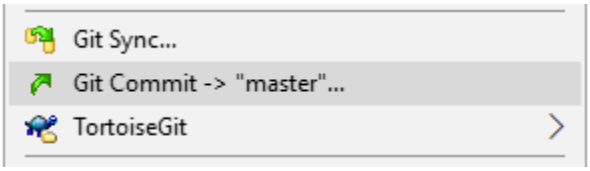
- You should see something like this:



- Click OK.
- You will be prompted for your netID and password.
- You should see text spill out in the progress window, once you see “Success” in blue with a time stamp you can click Close.
- Congratulations, you have now successfully created a local repository on your C:\ drive.

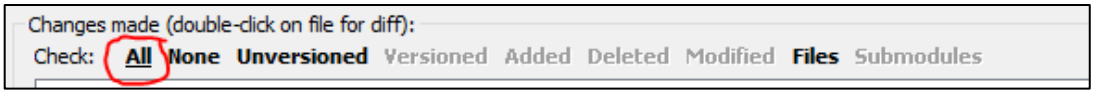
5. Commit

- You may commit as often as you'd like. Generally you want to commit after you've completed a section of the lab.
- To commit, in File Explorer, right-click on your repository "netID1_netID2" and select "Git Commit -> "master" … ".



- Add a commit message, e.g. "Completed through Lab2 exercise 1".

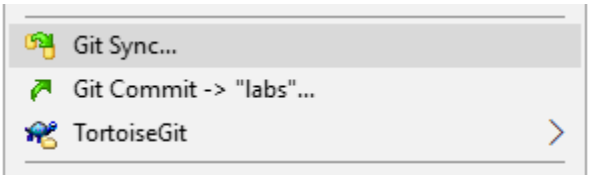
- Check All.



- Click Commit.
- Once you see "Success" in blue, your commit has be registered successfully.
- Click Close.

6. Push

- To push, in File Explorer, right-click on your repository "netID1_netID2" and select "Git Sync…".



- All you new commits should be listed in the Out Commits tab.
- Click Push.
- Enter your netID and password.
- Once you get “Success” in blue, you commits have been successfully pushed to the remote repository on the CS GitHub server.
- Click Close.

7. Pull

- To Pull, follow the same directions as for Push, but click Pull instead.
- You will only need to Pull if you or your partner works on the lab code on another PC, e.g. if you switch benches in the Mechatronics Lab you can pull your code to get the most up to date versions.

8. Log

- The log is the history of all you commits from all PCs/users.
- To view the log, right-click on your repository and click “Git Sync…”.
- In the bottom left corner, choose “Show log”.
- This will list all the commits, messages, authors, and time-stamps that you and/or your partner have made with this repository.