Git Guide

Sections:

* Overview & Usage
* Setup
* Commands
* Examples

**Overview & Usage**

Git is a distributed version control system. What does that mean?

* **Code Base.** Just as with SVN, we all have the same code base. We can get other people’s changes (pull) and share our changes with our team (push).
* **Work Simultaneously.** If two people work on the same file at the same time, no problem. Git will automatically merge the changes multiple people made to the same program.
  + If Git could not automatically merge, one person will manually resolve the “merge conflict.**”**
* **Commit History.** We have a record of our commit history and comments, which means can see what changes were made, to what files, and why we made them.
* **Safety Net.** We can view or revert to old versions of programs whenever we want.

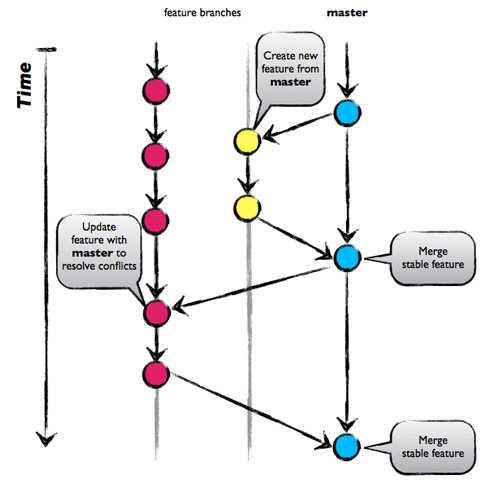
Note – typically Git is not used for CAD models. With GIT, we *can* view or revert to old versions of CAD models and we *cannot* merge file differences (only one person can make changes at a time to one file).

Here are some helpful concepts:

**Commit.** A commit is like clicking save. Each commit creates a snapshot of the repository which you will be able to view later. Commit often, it prevents disasters! I recommend committing after any feature is added, every 2-4 hours, and before you stop working.

**Branch & Merge.** To work on a separate copy of a repository, create a new branch. More than one person can work on a branch at a time. In this image we have 3 branches. Each circle is a commit.

**Figure : Branching. https://www.nicoespeon.com/en/2013/08/which-git-workflow-for-my-project/**

* The yellow branch is made from the master branch.  
  
* If major changes are made while you are working on a branch, get the changes (aka pull) and merge them to resolve conflicts. The pink branch does this.
* Once work is done on a branch, merge it back, like the yellow and pink branches do.

We have a master branch, which is the pristine copy of the repository.

* Do not work in the master branch. Do create new branches to work in *from* the master branch ☺
* Do not merge changes to master. I’m working on restricting the Master branch so I’m the only person who can merge changes to Master.

You can make a branch from master or any other branch. Branches are used liberally and have a dedicated purpose – make one in order to fix a bug, add a feature, or anything else.

Other concepts are described in the Commands & Example section.

**Setup (Clone Repository)**

Cloning a repository creates a copy of the repository on your computer which you can make changes to and share with our team. We currently have 1 repository – FeasibilityStudies. You will clone FeasibilityStudies in this guide.

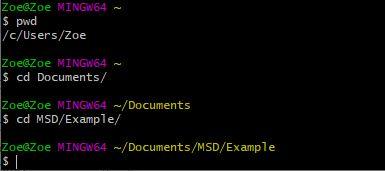
We will add more repositories as the need arises (e.g. for the GUI, master processor, other MATLAB simulations).

**Download Git Bash**

* <https://git-scm.com/downloads>
* In the section on configuring the line-ending conversions, choose the option **Checkout as-is, commit as-is** to avoid converting any line endings in files.

**Prepare the Repository Folder**

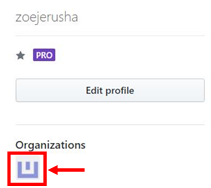
* Open File Explorer.
* Create the folder you want the repository to be in or decide on an existing folder.
* Open Git Bash.
* Go to the location where you want to make the repository.
  + Use pwd to see your current directory and cd <folder> to go to a new one.
    - Anthony, use dir instead of pwd if you use Git CMD…
  + For example, I want to make my git repository in my Example folder. The yellow text displays the current directory.



* + Additional basic information on using Git Bash is the Git Bash section.

**Copy the Repository Link**

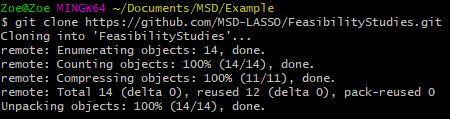
* Go to the email with an invite to our GitHub Organization (sent on 10/1).
* Accept the invitation to MSD-LASSO. If you need to make an account, use your RIT email.
  + Once you join MSD-LASSO, let me know and send your GitHub username so I can add you to the FeasibilityStudies repository.
* Go to the MSD-LASSO organization.
  + Go to Your profile from the dropdown in the top right.
  + Open MSD-LASSO by clicking the icon under Organizations on the bottom left.



* Go to the Respositories tab.
* Click FeasibilityStudies.
* Click Clone or download (green button).
* Copy the link.

**Clone the Repository**

* Go back to Git Bash.
* Use pwd to make sure you are in the correct directory.
* Use git clone <repo link> to clone the repository



You are done!

**Commands**

Remember to use the extension at the end of a filename if its referenced in a command.

**Add.** The first time you *add* a file, Git will start tracking it. (If you never add a file, its changes won’t be shown on the repository.) All subsequent times, adding a file prepares it for *commit* by adding it to the staging area (unneeded if you use git commit -a)  
 git add *file*

**Commit.** Similar to “save.”*Committing does not make your changes public.*Commit a file:  
 git commit *file*  
Commit a file and add all tracked files, so after you add a file once, you don’t need to add a file every time you change it:  
 git commit -a  
Commit a file and write the message in line:  
 git commit -m “This is my commit message”

**Pull.** Fetches other people’s changes on your branch and merges that into your current program. Do this if you know major changes were made that affect your code, and before you push.  
 If there is a merge conflict (the automatic merge failed), it will give you a message and insert dividers into your code around the conflict area, where you will need to manually fix it. To avoid merge conflicts, pull often and pull before pushing.   
 git pull

**Push.** Make your changes public. *Always pull before pushing.*  
If this is the first push on the branch since it’s been created, this reveals your branch to the team:  
 git push --set-upstream origin *myBranch*All subsequent pushes:  
 git push

**Status.** Shows you the status of your repository.  
 git status

**Remove.** Use this before deleting a file that was tracked by git. If you don’t, git still keeps a copy of the file in remote branches.  
 git remove *file*

**Move.** Use this before moving a file that is tracked by git. Or instead, you can git remove the file, rename it, then git add the file.  
 git mv *oldFileName newFileName*

**Diff.** See the changes (differences) you made before you commit. Red lines with a - were removed, and green lines with a + were added.  
For whole directory:  
 git diff  
For a specific file:  
 git diff *file*

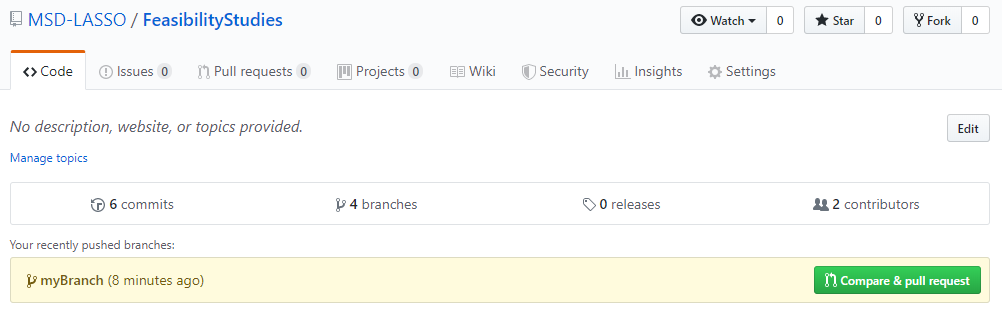
**View Log.** Displays previous commits.  
 git log

**View an Old Commit.** Before doing a git commit to save recent changes, you can use this view and run the files from an old commit. Eventually, you will need to return to the head (aka most recent commit) of your branch.  
View old commit. The hash can be found from git log and is a very long number (e.g. 6144d22f7235a81c868ce8f707cbdf09b54ddbe4):  
 git checkout *commitHash*  
Return to head:  
 git checkout *myBranch*

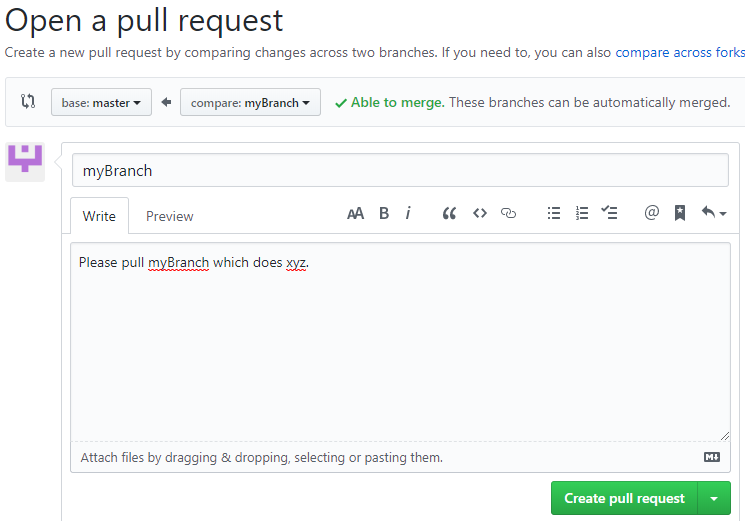
**Discard Changes.** You want to discard your changes and return to the last commit. *This will lose all your changes after the last commit!* Please temporarily save your files elsewhere, then run this command if you’re sure you want to undo your changes.  
 git reset --hard

**Make a New Branch.** Before working on a new program or updating anything in master, make a new branch to work in.  
 git checkout -b *branchName*

**Pull Request.** After you are done working in your branch and pushed, you can request a branch (probably master) pulls your code in. You are asking for your changes to be merged *into* master. Go to the repository on GitHub and click the green button, **Compare & pull request**.



Make sure your branch is the compare branch, and set master as the base branch. Fill out the Title and description information then click **Create pull request**.



You can now see your pull request in the Pull Requests tab on GitHub and check its status. You do not have permissions to accept a pull request into master. Send me a message when you’ve submitted a pull request so I’ll take care of it.

**Examples**

**Example: Edit code & make it public**

1. pull  
   git pull
2. Edit code
3. Either
   1. git add, then git commit (must choose this option if the file was never added before)  
      git add *myFile*  
      git commit -m “I did something”
   2. git commit and add simultaneously  
      git commit -a -m “I did something”
4. pull  
   git pull
5. push  
   git push

Use git status sprinkled in between, to make sure everything is working as it should.

**Example: Make a new branch and make it public.**

1. create a new branch  
   git checkout -b *myBranch*
2. git push (first time on this branch)  
   git push --set-upstream origin *myBranch* (lets others see what I have done)
3. Edit code, adding & committing throughout, then push (see other examples)
4. Submit a pull request on GitHub

**Git Bash**

* Here are some basic Unix commands (use in Git Bash).

****

<http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html>

* You can use tab to autocomplete directory or file names.
* Use a backslash to escape special characters like a space. Here is an example of changing directories in our SVN repository to a folder with a space (“Photo Gallery”).

