What is version control? You can find explanations in this chapter of the GitHub website: <https://git-scm.com/book/id/v2/Getting-Started-About-Version-Control>

It is important that you know how to use at least one Version Control System (VCS) given that it is common practice to work collaboratively on projects in many companies. In some cases, collaborative workers can be in different countries and operating according to different time zones so using VCS tools effectively is essential.

To encourage you to read the chapter on VCS and Git, answer the following:

* Why is Git more efficient at managing branch data (don’t worry about what branch data it yet) compared with say CVS?
* How are changes stored in Git?
* How does Git know about changes you make to files and directories - what does it use to determine this?
* What are the three main states that your files go through in Git?
* What does Git cloning do?

It is recommended that you learn the Git commands before choosing a GUI. This is because you can run all the Git commands from the command line but not from the GUIs.

You can run the commands from PowerShell in Windows or a Terminal window on a Mac or Linux system.

We can work through this in groups if easier. But, hopefully this will get you started.

# Installing

To install Git on Linux it’s a simple case of using command: apt-get install git (as superuser). There are specific downloads for Windows and Macs (part of Xcode) containing simple installers. There are also many GUIs available to run Git outside of the command line.

# Configuring

To be able to push changes to a Git repository you need to specify your user name and email, as this data will be sent with your changes.

To do this use the following commands:

$ git config --global user.name “Your name”

$ git config --global user.email “user@localMBP”

When testing, you might not want to use a real email. But, for more serious projects, you will want to add your email address.

You might also wish to setup your default text editor as the default Unix one is often used (not sure about Windows). The command to set the editor to emacs for editing Git message files, etc., is:

$ git config --global core.editor emacs

To see what you have configured, use the following command:

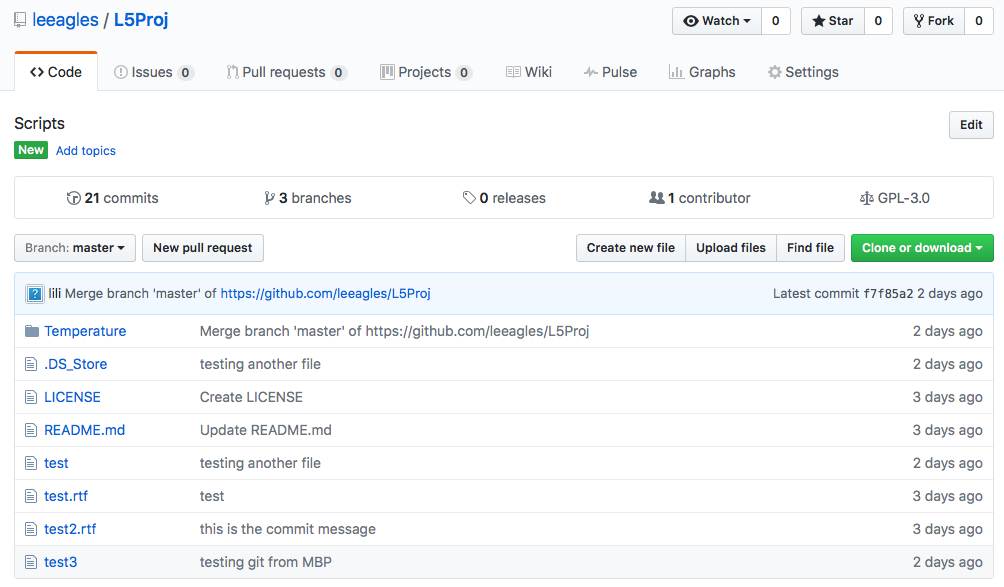
$ git config --list

# Git commands – working with a cloned local repository

This chapter contains the basic commands (and explanations) to get you started. See: <https://git-scm.com/book/id/v2/Git-Basics-Getting-a-Git-Repository>

In summary, you will want to know how to obtain data from the repository which corresponds to your project, how to create a branch from which you will work from and how to apply your changes to the main project when ready. There are other commands you will need to know later, but for now, this is all you need to focus on.

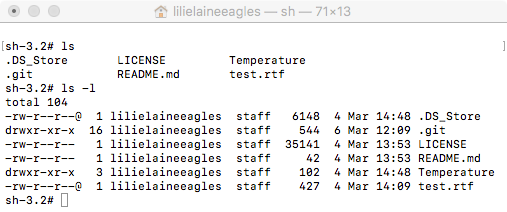
Imagine your team has created a repository for your project – named L5TP1, e.g.



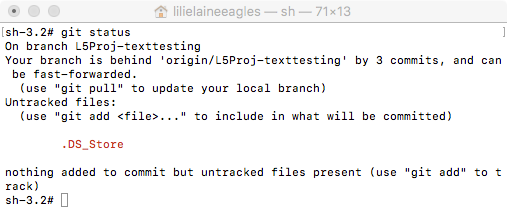
each team member would run the following command on their client machine, i.e. Linux, Windows or Mac system:

$ git clone https://github.com/L5TP1/L5TP1

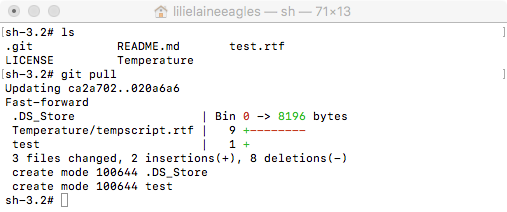
This creates a directory named L5TP1 and initialises a .git directory inside it. All the data from the repository is also pulled down, providing you with a working copy of the latest version. For more details, read this page: <https://git-scm.com/book/id/v2/Git-Basics-Getting-a-Git-Repository>



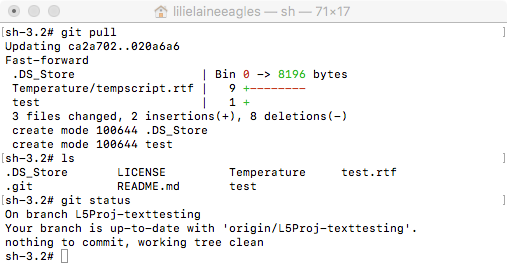
A useful command to use frequently is git status, which allows you to check what stage files are in, e.g. unmodified, modified, staged (changes to commit to project). When you first run the command, you will receive the message stating that there is nothing to commit.



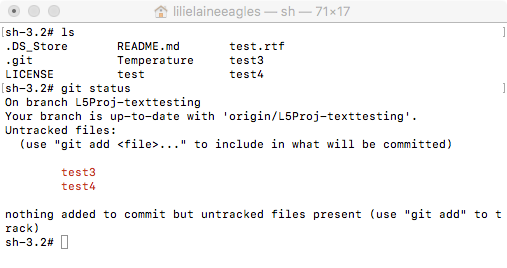
as advised by Git, used git pull to update my local branch



After running: git status again:



To include a new file in the project, run the status command and you will see that the file is being tracked and staged to be committed. Always add a file using git add when you have changed it to make it part of the latest version.

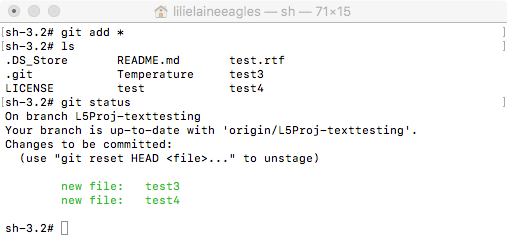


You edit it and then use the following command to stage it for future upload to the main repository:

$ git add new\_file

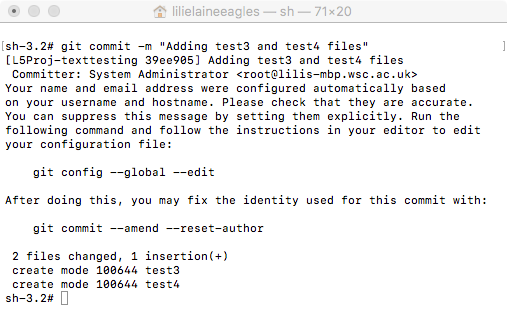
Note that on the main repository, there is already a file named test3 (on the master branch – which will be covered soon), so a conflict will occur when changes are pushed up to it. This will not be in issue while it is local.

After running git add, the following will be displayed when you run git status:



To commit your changes, you use the git commit command. This will commit only staged data to your local project. Remember to add all files that you want to be committed.

$ git commit –m “Adding test3 and test4 files”



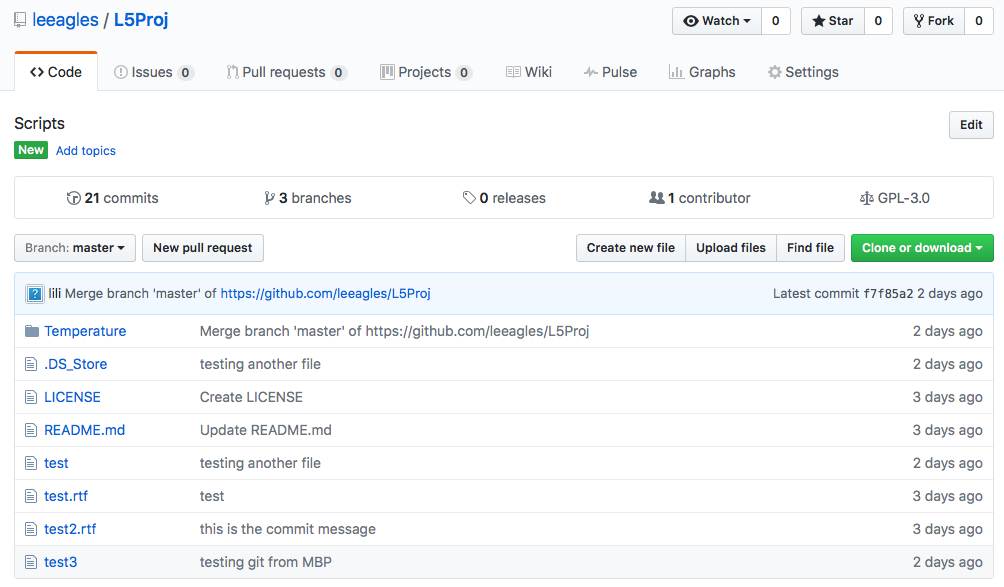
To view the commit history, use the git log command. This is useful when you are learning to use Git and you want to keep track or remember what you did, etc.



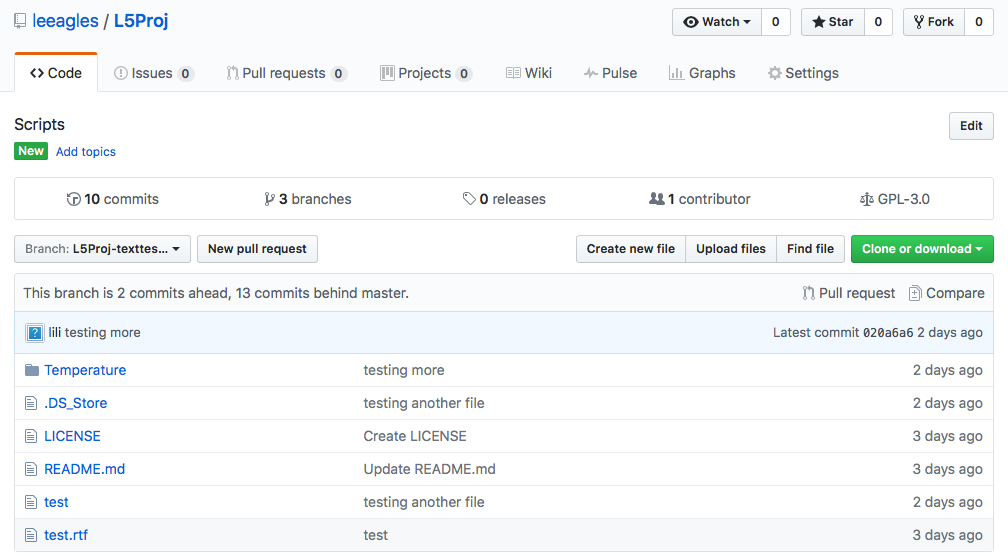
There is a way to ignore certain files too, which you can read about on the online tutorial. Another useful command is git diff, which shows you what you’ve changed but not yet staged.

The examples and screen captures you have seen so far have been based on a branch. I’ve created a potential conflict that I am aware of by adding test3, because the master branch already has test3. When I merge the branches into one project, this will show up. But, until I do this, it will not be an issue. So, before you go too far, this really is the time to get to know about branching: <https://git-scm.com/book/id/v2/Git-Branching-Branches-in-a-Nutshell>.

The following is the Master branch:



Compare with another branch: L5Proj-texttesting:



You will see that you don’t have all the files related to the Master in your directory if you are working from the branch L5Proj-texttesting instead of Master.

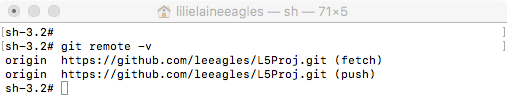
# Git commands – working with a remote repository

Now that you’ve worked on a local cloned repository, you need to know how to work with remote repositories, i.e. to collaborate with team members.

So, you have cloned a repository using:

$ git clone https://github.com/L5TP1/L5TP1

Run the git remote –v command to see which remote servers have been configured. If you don’t have all the data expected then you can fetch it using the following command:



$ git fetch origin

If you are working on a branch, then you can use git pull to fetch and merge with your current branch.

To change branch, you can use the following command:

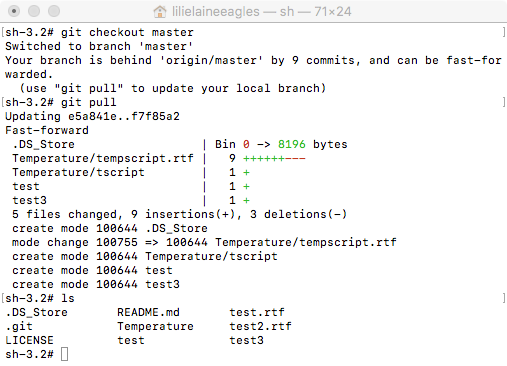
$ git checkout -b L5Proj-texttesting

or

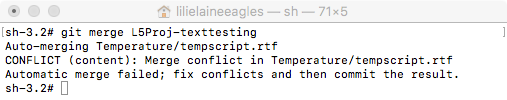
$ git checkout master

An example:

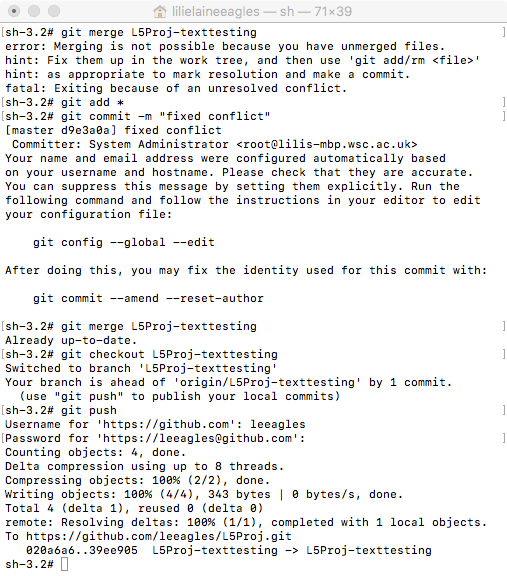
First I ran git checkout master and there was a complaint from Git. So, I ran Git pull to update my directory with the Master branch files as shown:



When the L5Proj-texttesting is merged with Master you will see a conflict, as mentioned:

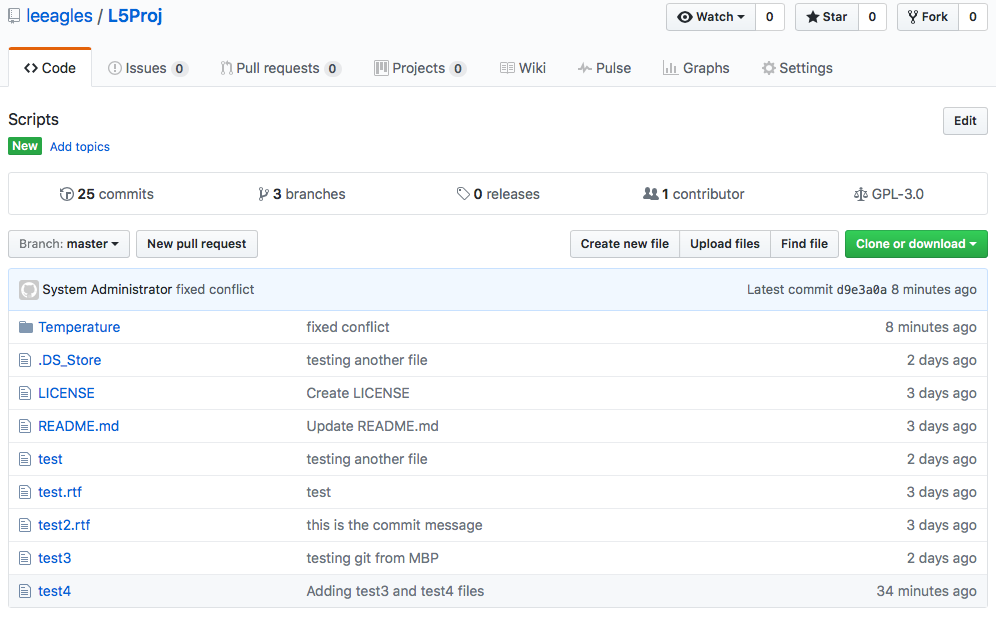


It is important to resolve the conflict, i.e. remove my test3, then redo the merge.



As a file (file 3) was removed and there were changes, git commit was needed to make the local project changes. A git push was required to push changes to the remote repository (to synchronise master and L5Proj-texttesting) before they could be merged.

After merging the branch with Master, you should see the additional file (test 4) that was created via the L5Proj-texttesting branch:



In summary, the following commands are useful when working with remote repositories:

$ git checkout -b <your branch name>

or

$ git checkout master

$ git pull

$ git push

$ git merge <your project name>

There are many other commands of course, but the above should give you a good starting point. Don’t worry, it can be confusing at first. But, it is important that you learn this tool as it, or one like it, is likely to be used by a company you work for in the future (not just for programming).

Create a help table for your reference.

We can work through this in groups if easier.