

Summary: All students who have formed a group for their ENCE260 project and logged into the [eng-git](#) server should have received an email from our Administrator stating that they can now log into git and use this as part of their code development. This note provides information on how you and your project partner can start using your newly created git repository. It is by no means meant to be a comprehensive guide. This note is designed to get you started so you can at least push your source files to your **eng-git** repository and keep this updated.



1. Introduction

As stated on our [ecewiki](#) website, git is a distributed revision control system. This means every working directory is a full-fledged repository with complete history and full revision tracking capabilities, and is not dependent on network access or a central server. However, if you haven't used a distributed revision system before the process can be a little intimidating. This guide and tutorial will hopefully take you through most of the pitfalls and at least get you started.

Here are a few motivations reasons why you should use git for your assignment:

1. Both you and you project partner can work collaboratively on code development.
2. Updates can periodically be pushed to the **eng-git** server. Local commits are saved to your local drive or server. Past revisions are never lost and can be easily retrieved.
3. Two levels of file management and revision control are available - local and server based.
4. Use of a repository such as git prepares students for revision control used in industry.
5. Last, but most importantly, your lecturer will only be able to access and mark your code if it is in your team git repository.

2. Command line tutorial

So, you have received an email from gitlab saying something like: "You have been granted Developer access to group ence260-2017. If you are in the lab your PC has git installed. If you are using your own PC or Laptop, you may need to install git. Various versions and variants exist.

The following tutorial assumes you have: i) an internet connection, ii) you have a command line version of git installed and available, and iii) you are logged into your terminal program and in your `../ence260-ucfk4/assignment` subdirectory. This last requirement is particularly important.

1. Type in: "`git clone https://eng-git.canterbury.ac.nz/ence260-2017/teamXX.git`", where XX is your group number. Note if you only have a single digit for your group number, e.g., 0, your repository will be "team0". If all is well you should see a message in your terminal window saying something like "Cloning into 'teamXX.git' ..." and a warning message that you have cloned an empty repository. This warning is fine. We have just created your teamXX repository so there will be nothing in it. Pushing your source files into this will be up to you!
2. Check that you have a newly created "teamXX" directory by typing "`ls`" at the command prompt. You should see: "README final teamXX".
3. Now copy some files into your newly created teamXX directory. To get you going, you may want to copy the Makefile and game.c source files from the /final directory. These are just very simple example files in an example directory. The source, game.c, should compile in your "/final" directory, and if you move these files they should also compile in your "/teamXX" directory. Make sure they do compile by typing in "`make`". They won't do much but, again, they're not meant to.
4. We are now going to add these files to your git repository. **You only need to do this once.** To do this simply type: "`git add game.c`". You may not get any feedback from this. Then type in: "`git add Makefile`".
5. Use "`git status <command>`" to ensure your source files have been added and updated.
6. The next step is to commit your source files to your repository. Do this by typing in the following:

```
git commit game.c -m "make a comment on the major changes you have made"
```

Note that the double quotes **are required**!

7. Also do a commit for your Makefile. Remember, your Makefile is also a source file. Note that you should receive feedback here, such as "one file changed ..."
8. The last thing for you to do is to push your last commits on to your teamXX repository on our eng-git server. To do this type the following:

```
git push origin master
```

Once, you have typed and entered this in the command window you should see some dialogue from git stating something like:

```
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 961 bytes | 0 bytes/s, done.
Total 6 (delta 1), reused 0 (delta 0)
To https://eng-git.canterbury.ac.nz/ence260-2015/team0.git
 * [new branch]    master -> master
elec-sweddell-3:team0 sweddell$ pwd
/Users/sweddell/Courses_2015/ENCE260/ence260-ucfk4/assignment/team0
```

9. That's it for now. At this point I suggest you check your git repository and verify that your files have indeed been pushed onto our eng-git server. You may even like to download your files, which will be nicely wrapped in a git wrapper! More importantly, your project partner will be able to clone these files onto their PC or Laptop. This is where you start to work co-operatively together on code development. Both of you can push your work to update your shared repository, however, and most importantly, execute a git pull first in order to ensure your code is updated and synchronised with your code repository before you submit your changes. If no changes have been made by your project partner, you should receive the message, "Already up-to-date".

3. Traps and pitfalls

1. Use the recommended directory for your game program development, such as:

```
../ence260-ucfk4/assignment/teamXX
```

where XX stands for your group number. If you don't then the Makefile paths will need to be modified. This is fine if you know what you are doing, however I typically get a lot of emails from students who move this directory to some obscure location, where the path settings in your makefile are set to a very specific level.

2. Commit regularly, i.e., every hour or so. Push every session. Again, you have a two tier revision tracking system however you will only update your partner's code when you pull his or her code from the eng-git server first.

4. Documentation

Documentation for the UCFK4 can be found at:

<http://ecewiki.elec.canterbury.ac.nz/mediawiki/index.php/UCFK4>.

Also, try out the [code school tutorial](#), as outlined in your tutorial notes.