Version Control with Git and GitLab

Chris Price January 2018

In this practical, you will learn:

- How to make a Git repository
- How to set up your GitLab account
- How to clone a remote GitLab project
- How to make changes and send them back to a remote GitLab project
- How to make a group including other people
- How to work together on a project and how to resolve conflicts when they occur

1. Setting up Git on your computer

You should only need to do this the first time that you log on a computer system, but it needs to know who you are, your email and the editor you want to use.

On the command line (use GIT CMD on Windows), give the following commands, substituting your user name for "cjp":

```
git config --global user.name "cjp"
git config --global user.email "cjp@aber.ac.uk"
```

If you wish to use a different editor from vi to give details of the changes you make, also do the following command, putting in your favourite editor (e.g. emacs) if you don't like gedit either.

```
git config --global core.editor "gedit"
```

2. Setting up a local repository with existing files

This shows you how to set up the group project (although only one member of your group will need to do this - the rest will *clone* the project, see later). If you already have some files that you wish to make into a repository, then it is also a good way to do that.

a. In a web browser, go to https://gitlab.dcs.aber.ac.uk/cjp/GroupProj.git

Click on the above icon, and download a zip file containing the basic structure of the group project repository. Unpack the zip file if it is not done automatically for you. What you have is just a set of folders and files, not a repository yet.

b. Make into a repository
Name the root directory something sensible, e.g. GP22, and changing directory at the command line so you are in the GP22 directory. You can then set up the local repository by the following commands.

```
git init
git add .
git commit -m "Setting up repository"
```

The init command makes the repository, the add command marks all changed or added files as ready to be added to the repository (all files in this case). This brings them under source control. The commit command will prompt you to edit a commit file recording why you are doing the commit unless you use the "-m" option. When doing commits because you have made a change to the repository, you will want to give sensible reasons, such as "adding my research on UI done this week" or "fixing problems with design spec found in review".

Another useful command is

```
git status
```

It will tell you when you have files that you have files that you need to add or to commit.

3. Setting up GitLab for your use

In a web browser, go to https://gitlab.dcs.aber.ac.uk. Log in using your Aberystwyth University user id and password.

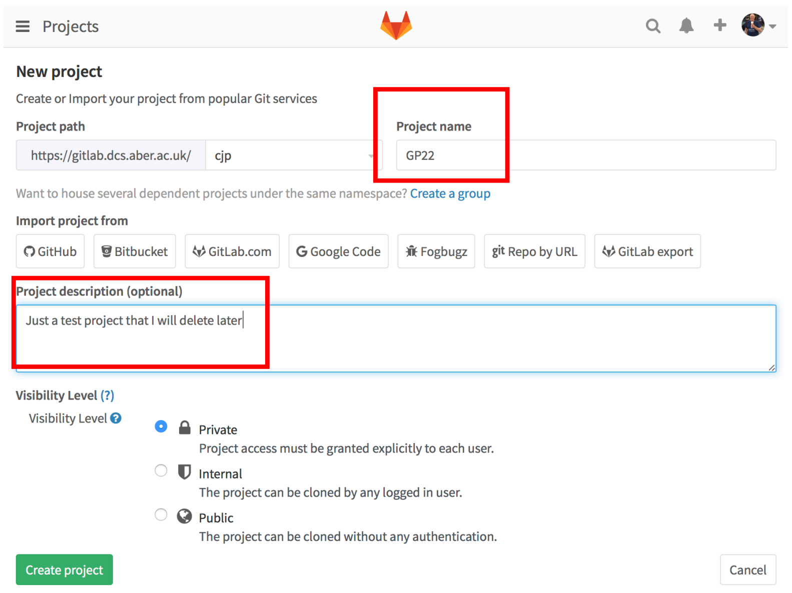
You will be logged into GitLab and your user ID should then be recognised as a GitLab user (e.g. when someone wants to add you to your group project group).

Much of our use of GitLab will be done from the command line, but setting up new projects, changing who can access a project, and raising issues are all done from this web page. It also gives a summary of the latest changes that have been committed to the repository.

4. Making a GitLab repository and linking your local repository to it

In the browser, make a new project in GitLab. If the "New Project" button is not visible, click on the orange icon shown below, and you should get a page showing it.

You will get the screen below - fill in details in the circled areas, and click "Create Project".



Projects

New project

Create or Import your project from popular Git services

Project path: https://gitlab.dcs.aber.ac.uk/ cjp

Project name: GP22

Want to house several dependent projects under the same namespace? Create a group

Import project from: GitHub | Bitbucket | GitLab.com | Google Code | Fogbugz | git Repo by URL | GitLab export

Project description (optional): Just a test project that I will delete later

Visibility Level (?)

Visibility Level

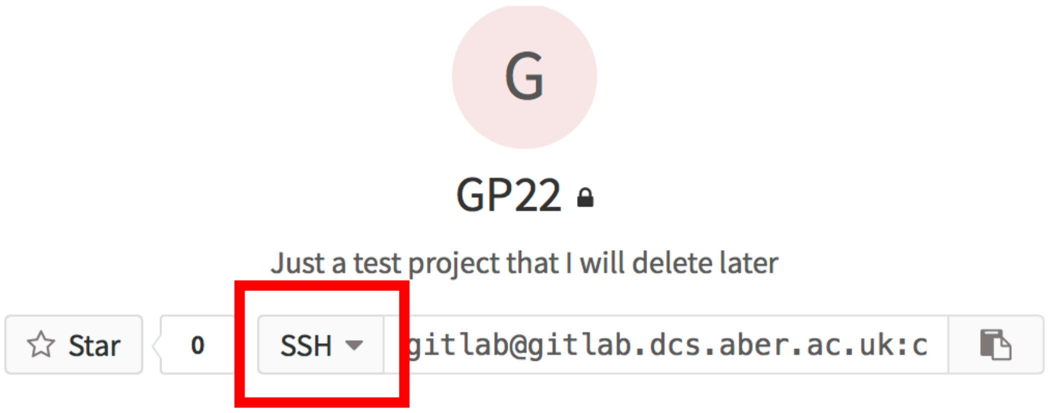
[✓ Private] Project access must be granted explicitly to each user.

Internal The project can be cloned by any logged in user.

Public The project can be cloned without any authentication.

Create project | Cancel

To link your local repository to the GitLab repository, you need the link shown towards the top of the next page in the browser. Click on where it says SSH (as below) to see a https link to the repository. Copy the link (it will look like https://gitlab.dcs.aber.ac.uk/cjp/GP22.git, but with your user ID instead of cjp).



G

GP22

Just a test project that I will delete later

Star | 0 | SSH | gitlab@gitlab.dcs.aber.ac.uk:c

At the command line, give the following instruction using the copied link

```
git remote add origin <copied link>
```

It will look something like:

```
git remote add origin https://gitlab.dcs.aber.ac.uk/cjp/GP22.git
```

You can now copy the state of the project on your computer to the GitLab repository by:

```
git push -u origin master
```

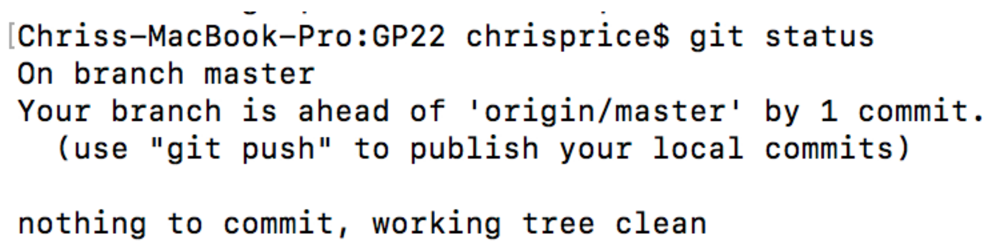
5. Edit your local repository and push the changes to GitLab

a. Edit the README.md file to say this is project 22 not project 99, and save your changes.

b. Do "git status". It will show that you have changed README.md, but it has not been "staged" (i.e. you have not done "git add .") or "committed" (you haven't done "git commit").

c. Do "git add .", then do "git status" again. Now it just says that you have not committed the change yet.

d. Do "git commit" and add a commit comment as prompted, and then "git status" again. You should see something like the following:



```
[Chriss-MacBook-Pro:GP22 chrisprice$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

The first part (ahead of 'origin/master) tells you that you have changes on your computer that have not been copied to the remote repository in GitLab. The second part says that all of the changes you have made have been staged and committed locally, so to get your changes onto GitLab, you just need to push them

e. Do "git push -u origin master". Do "git status" again, and it will say your working tree is clean. In the browser, look at your GitLab project details (you may need to refresh the screen). It should say that README.md has been changed, and you can click on it to see that your local change has now been propagated to GitLab.

6. Clone a remote repository

Delete your local GP22 directory and all its contents.

Using the http link to your GitLab repository, give the command below (it will differ at least by not being{"cjp":

```
git clone https://gitlab.dcs.aber.ac.uk/cjp/GP22.git
```

This will make a new version of the repository on your local computer, linked to the GitLab repository, and you can make changes, add them, commit them, and push them to the remote repository as if you had never deleted the local version.

TAKE A PAUSE - YOU'VE EARNED IT

You just about have all of the commands that you need if you are working on your own on a project. You will have a remote copy in case your local computer ever crashes. You can just go to a computer that doesn't have your project on, clone the project, work on it, push the changes, and delete it from that machine.

If you keep a copy of the project on your own computer, but work on it elsewhere, then you will also need the following command to keep the copy on your own computer in sync with the changes made elsewhere:

```
git pull
```

GitLab - working in a team

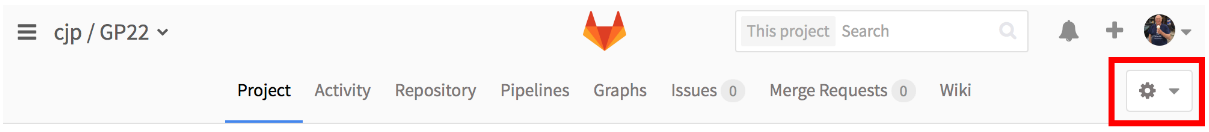
Problems tend to start occurring with Git when more than one person is accessing a repository - as will be the case on the group project. These problems occur when more than one person is accessing the same file at the same time. If Jim and Jane both work on a file on their own computers, then Jim pushes his version to the shared GitLab repository, what happens when Jane wants to do the same? If her copy is written to GitLab, it will overwrite Jim's changes and they will be lost.

You now need to work with another person to see how this problem is avoided.

1. Find a partner and share a repository

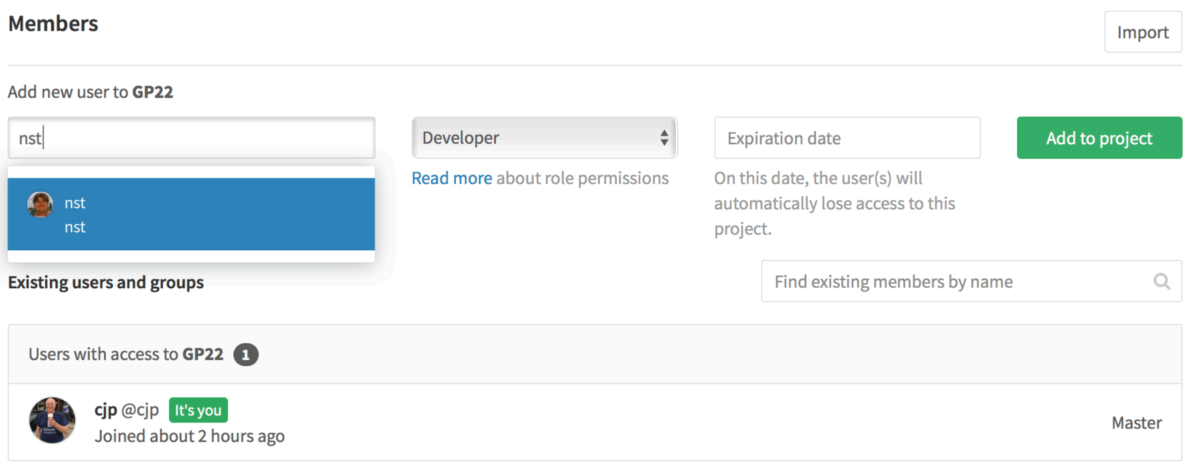
You need to work with someone else at this point. You only need one repository, and so decide whose repository to use and delete the other one, so you don't get confused.

a) The unwanted repository on GitLab can be deleted by going to settings for the project (shown below) and choosing "Edit Project". At the bottom of a long list, you will see "Delete repository" - do it!



b) Delete the local repository for you and your partner.

c) The person who still has a GitLab repository should add their partner as Developer on that repository. Again in the "Edit project" menu, select "Members". The screen below shows where you put the partner's user ID (I'm adding "nst" as a partner), and where you choose "Developer". Then click Add to project and they can also access this repository.


Members
Import
Add new user to GP22
nst
nst
nst
Developer
Read more about role permissions
Expiration date
On this date, the user(s) will automatically lose access to this project.
Add to project
Existing users and groups
Find existing members by name
Users with access to GP22 1
cjp @cjp It's you
Joined about 2 hours ago
Master

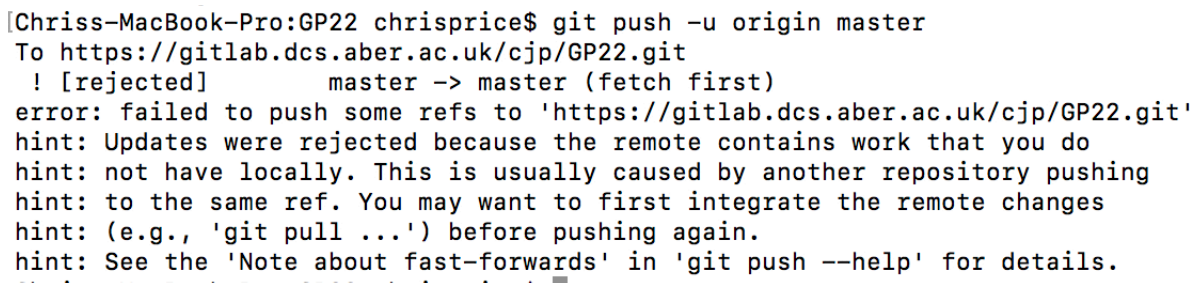
d) Clone the project to your local computer (as done in section 6 above, but if you are not the person who made the repository, the link will have your partner's user ID.

2. Both make changes and try to commit them

a) Each of you edit the README.md to list your name and then your partner's name.

b) One of you add and commit the file, and push to master. This should work fine.

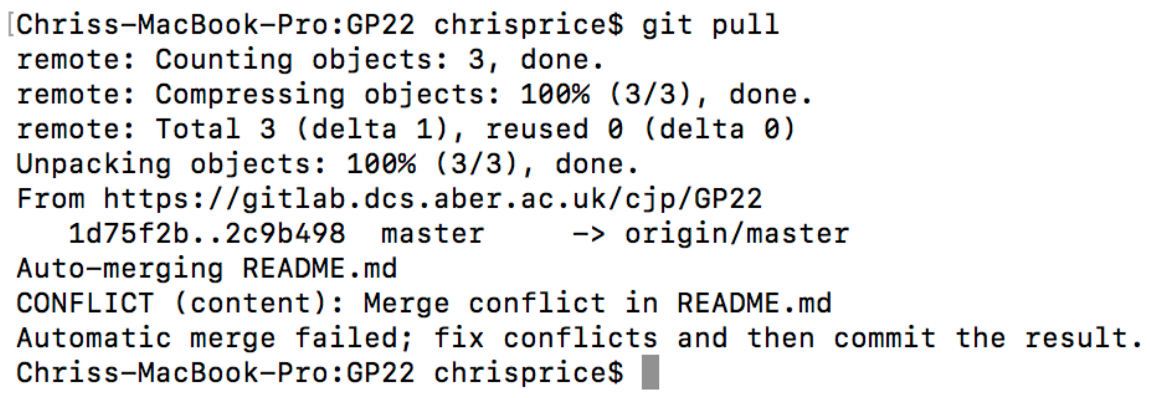
c) Second person should now try to add, commit and push to master. They should get a message like the following:



```
Chriss-MacBook-Pro:GP22 chrisprice$ git push -u origin master
To https://gitlab.dcs.aber.ac.uk/cjp/GP22.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://gitlab.dcs.aber.ac.uk/cjp/GP22.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

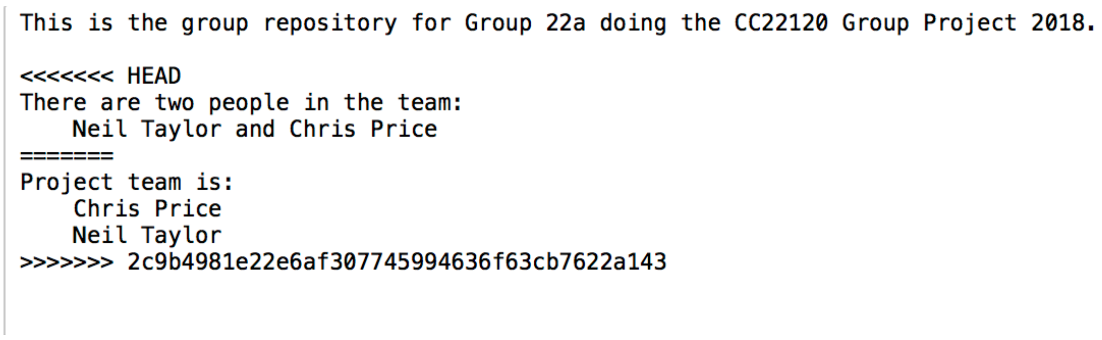
This tells you that someone else has added work to the remote since you last pulled a copy of it. You need to do "git pull" to get those updates. If you were lucky, they would have been working on other files, and you would then be able to do your push with no problems. However, we deliberately edited the same file so we know that is not going to be the case.

So when we do "git pull", we get the following error message saying that there is a clash in the file READING.md that we need to fix:



```
[Chriss-MacBook-Pro:GP22 chrisprice$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://gitlab.dcs.aber.ac.uk/cjp/GP22
   1d75f2b..2c9b498  master     -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
Chriss-MacBook-Pro:GP22 chrisprice$
```

So there is a problem with README.md. If the person who did the second push now edits the file again, they will find that extra info has been added to it similar to below.



```
This is the group repository for Group 22a doing the CC22120 Group Project 2018.

<<<<<<< HEAD
There are two people in the team:
    Neil Taylor and Chris Price
=======
Project team is:
    Chris Price
    Neil Taylor
>>>>>>> 2c9b4981e22e6af307745994636f63cb7622a143
```

The text between "<<<<<<< HEAD" and "=================" is what was in my version that is different from what is in the other version (shown between "===========" and ">>>>>>> 2c9b4981e22e6af307745994636f63cb7622a143")
You need to edit the file again, so that looks like it should. Ideally this would include your changes AND the changes by the other person. You also need to delete all the "<<<< HEAD" type markers.

You can then add, commit and push and it should work.

END OF PRAC

Useful extra: Set up a .gitignore file

Sometimes you do not want files to be under source control. A good example is files that are created by the compiler.

If you add a file named ".gitignore" to the GP22 directory, it will ignore all files that meet the pattern listed there.

For example, if your .gitignore file contained the following lines, git would ignore all files that ended in ".o" or in "~".

```
*.o
*~
```

Further examples of .gitignore files are in the ProGit book (http://git-scm.com/book/en/v2)

Git cheat sheet

Once you have done and understood the prac above, this page should act as a reminder for what you need to be able to do.

Cloning a repository from GitLab

```
git clone <url for GitLab repository>
```

Getting latest updates done by other group members

```
git pull
```

Do this at the start of each day and when your push fails.

Making sure your changes are included in your local repository

```
git add .
git commit -m "meaningful comment here"
```

Updating the remote repository from your local repository

```
git push -u origin master
```

Tools to make life easier

There are GUI tools to help you with Git. Indeed, you will find that you can change the remote repository from the GitLab web page, and get information on what changes people have made.

The University computers have a tool called GitEye - this allows you to do your adding and committing and pulling interactively, but you still need to be able to understand what is happening when you do these things or you will get in trouble. It is also available for Linux and Mac.

There is another tool on Windows called TortoiseGit which is quite popular.

Several of the IDEs can also be linked to your repository.