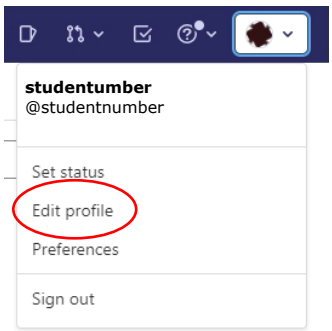
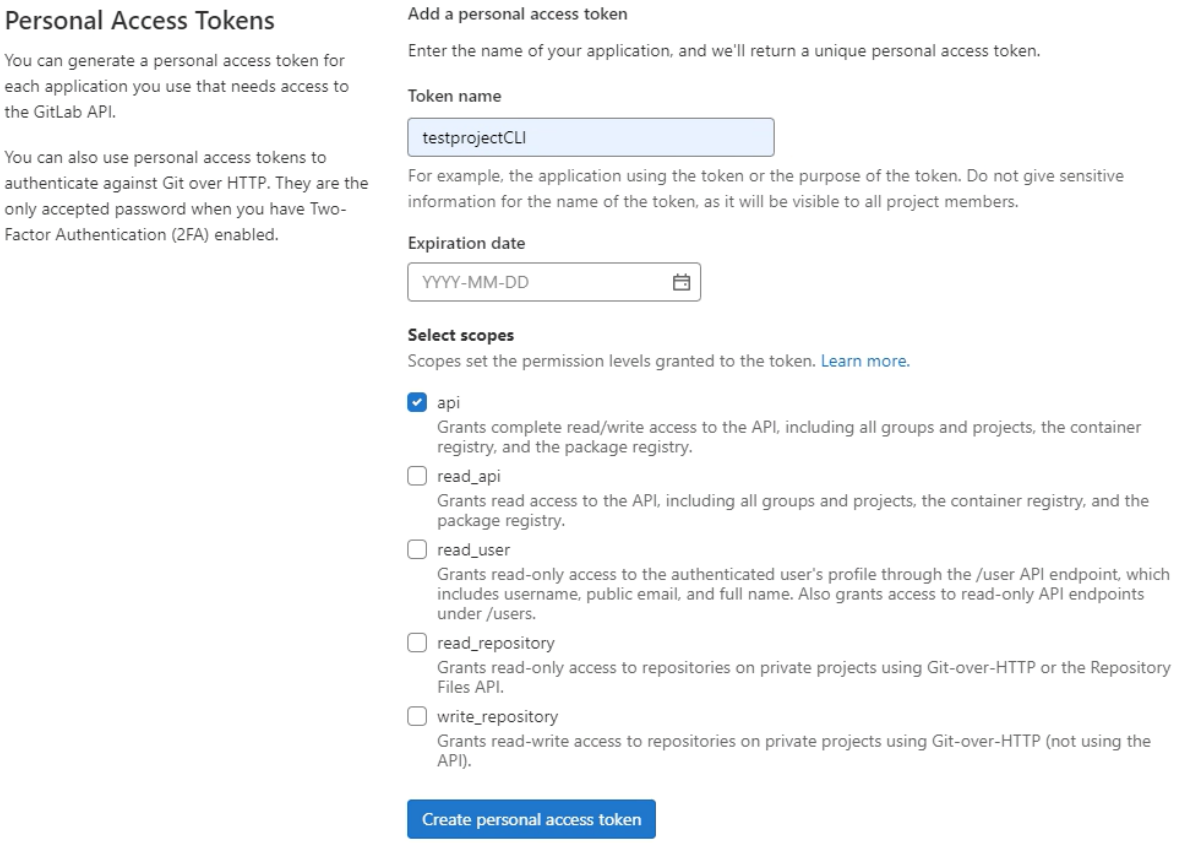
EEECS Git Command Line Guidance

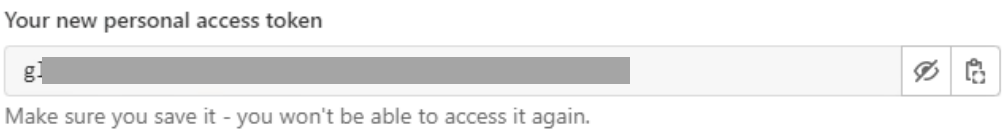
Git can be operated in two ways – via a graphical user interface (GUI) or a command line interface (CLI). This guide will teach you how to set up Git using the CLI. Git is already installed on the EEECS lab computers. To install it on your home computer, follow the download instructions at: https://git-scm.com/downloads

To read/write from/to the repository from outside you will need to create a personal access token with 'API' scope. Do this by logging into EEECS GitLab and from your GitLab home page via your profile icon, click **Edit Profile** and then **Access Tokens** from the menu. Choose "Add a personal access token" and choose the '**API'** option and click **create**.

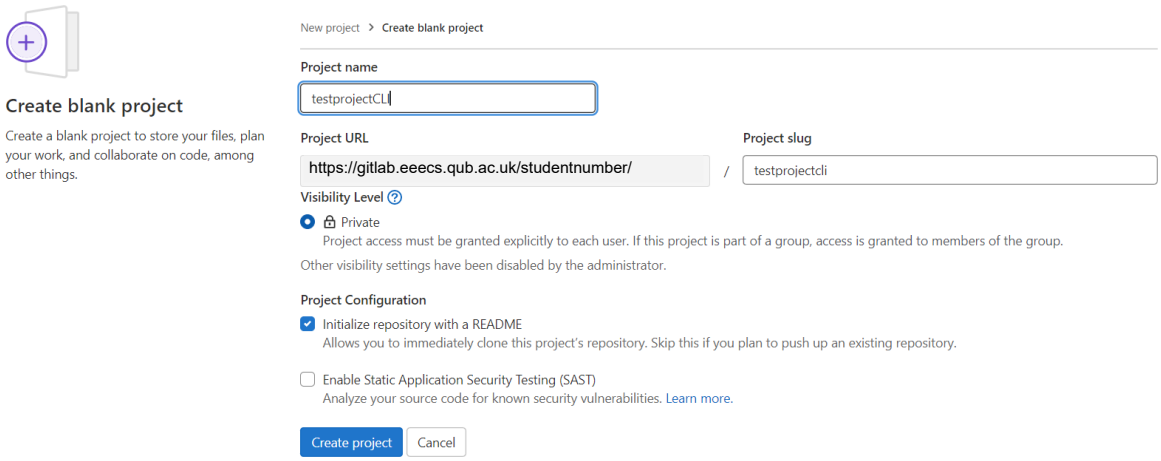




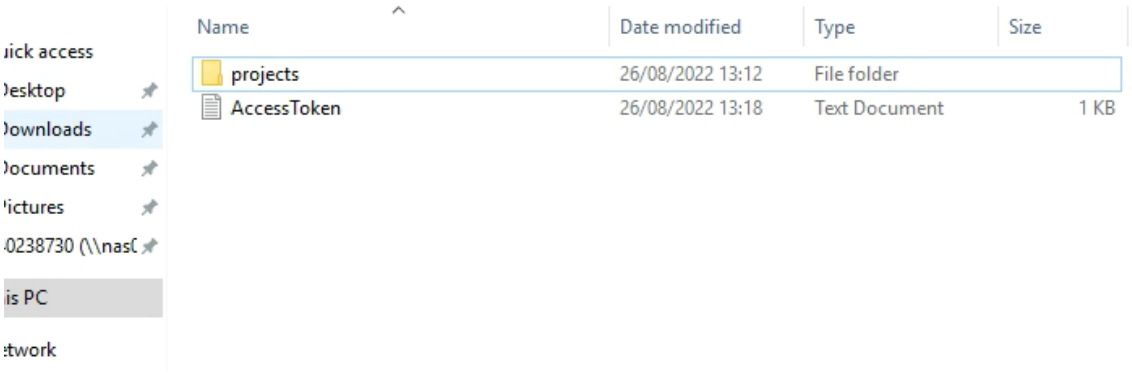
This will generate a token at the top of the page. Take a note of the access token by copying the token into notepad and saving it.



Once you have saved that, create a new blank project on GitLab.



Then create a directory to store your project on your local machine. In my case, I have created a folder called 'projects'. The project we have created on GitLab will be stored in there.



Open the command prompt window from the Start menu button and type in the following commands to configure git:

1. **To set your username**
   git config --global user.name "FIRST_NAME LAST_NAME"

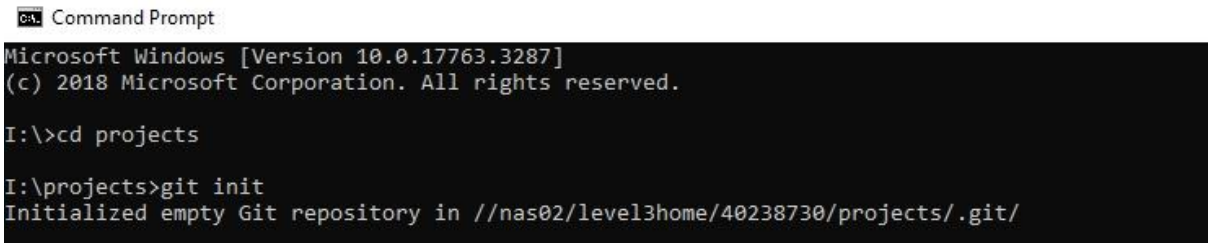
   e.g., git config --global user.name "Bob Smith"

2. **To set your email**
   git config --global user.email "MY_NAME@example.com"

   e.g., git config --global user.email "bsmith01@qub.ac.uk"

Then navigate to the folder you just created by typing **cd x** where **x** is your folders name. Followed by typing the command **git init** as shown below.

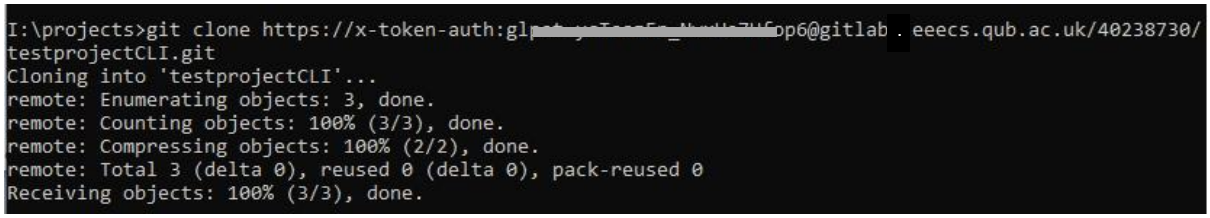
This will initialize the local directory as a Git repository.



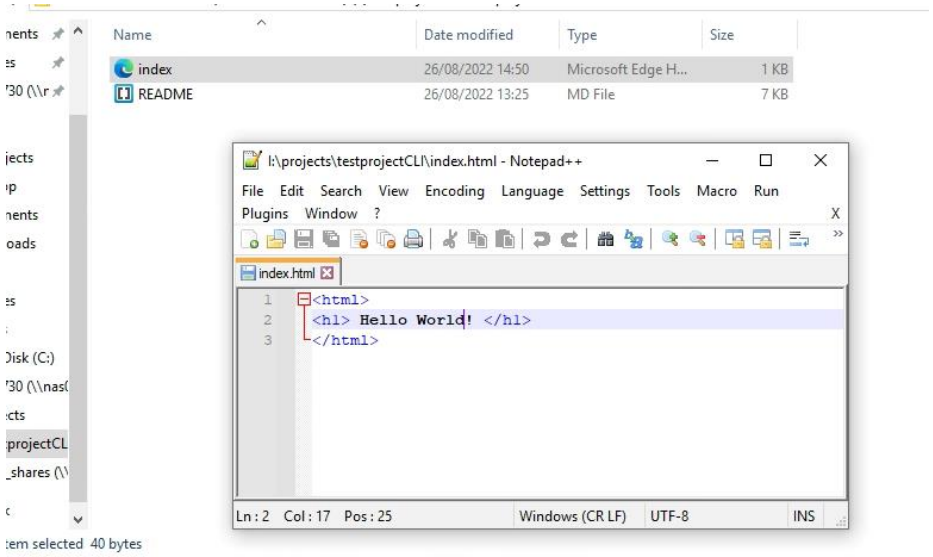
You will then need to clone the existing project from GitLab. Type in the following command:

git clone https://x-token-auth:xyz@gitlab.eeecs.qub.ac.uk/studentNumber/GitProjectName.git

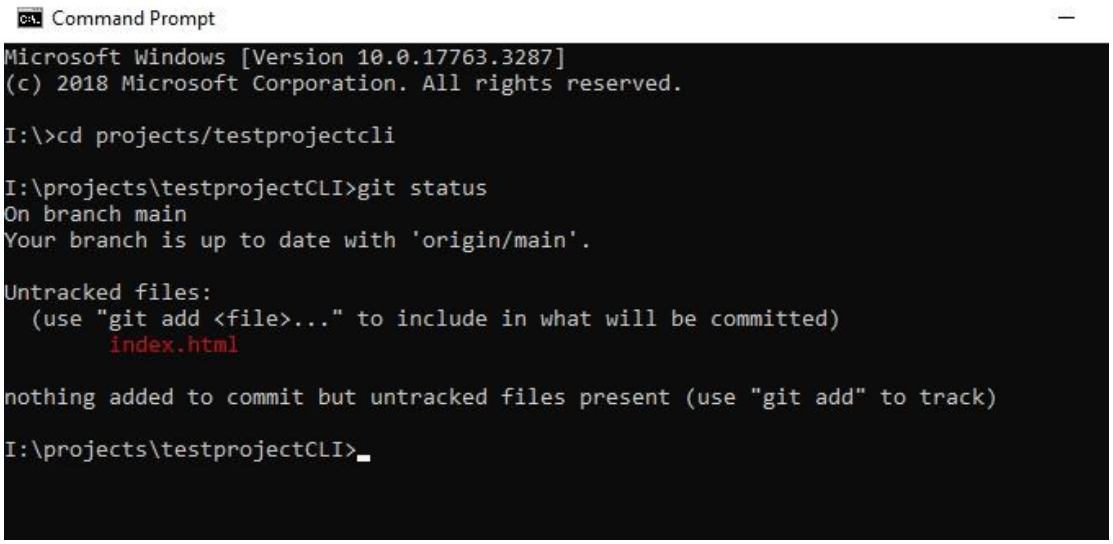
Replace **xyz** with the personal token you generated earlier, **studentNumber** with your own student number and **GitProjectName** with your EEECS GitLab project name as shown in the example below:



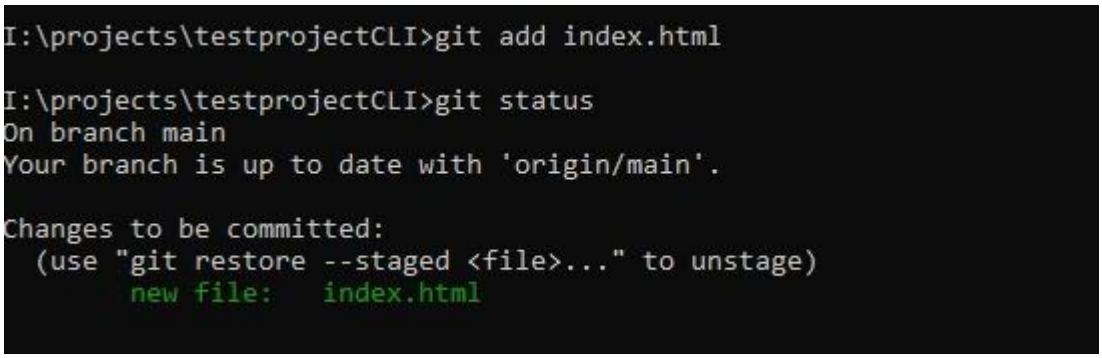
If you check your projects folder, the GitLab project should have copied into that folder. Now that the repository has been initialized, it's time to add some code to the project. Below I have created an HTML file as an example.



Git notices when you add or modify files in the folder containing the git repository but doesn't track the file unless instructed. Git saves the changes only for the files it tracks, so you need to let Git know you want to track changes for a specific file. You can check which files Git is tracking by running **git status**



Git notifies you if you have any untracked files. If you want Git to start tracking a file, run the following command: **git add [filename]** to add one particular file or **git add .** to add all the files that has been changed.



If you rerun the **git status** command it will show that Git is tracking the specified file. After adding the files to the staging enviroment, we need to instruct Git to package the files into a commit using the **git commit** command. Git then stores that file version and you can review the stored version at any given time.

git commit -m "message"

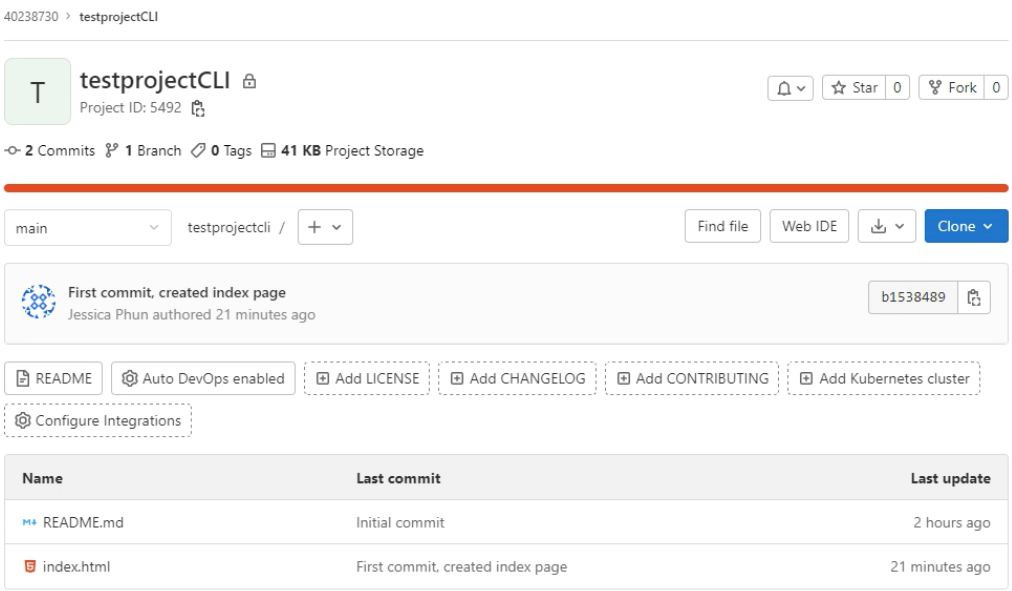
Add a message at the end of the commit to state whether it's a new feature, a bug fix, or anything else. Commits remain in the repository, and they are rarely deleted, so an explanation of what you changed helps other developers working on the project or help you keep track of all the changes.

```
I:\projects\testprojectCLI>git commit -m "First commit, created index page"
[main b153848] First commit, created index page
 Committer: Jessica Phun <40238730@eeecs.qub.ac.uk>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 3 insertions(+)
 create mode 100644 index.html
```

To push the changes in your local repository to GitLab, use the command **git push**

```
I:\projects\testprojectCLI>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 333 bytes | 41.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://gitlab2.eeecs.qub.ac.uk/40238730/testprojectCLI.git
   e382708..b153848  main -> main
```

To see the if the changes has been pushed, go to your GitLab and you should see that the file has now been added to your project. If you create any new files or modify files, you will need to **git add**, **git commit** and **git push** again so that GitLab can be up to date with your local repository.



Uncommit changes

Suppose you made some error in your code or committed the wrong files, you can remove the most recent commit by using the command **git reset HEAD~1**

```
I:\projects\testprojectCLI>git reset HEAD~1
Unstaged changes after reset:
M       index.html
```

Team working on GitLab

If you are working on another machine and set up the Git repo on there, you will need to get the up-to-date code copied into your local repository.

Git clone remote_repository_URL

If you are collaborating on a project with others, each time someone makes a change and pushes it to the remote repository, you will have to pull the changes that they made into your local repository to make sure you are working on the latest version of the Git repo. To do this, use **git pull**