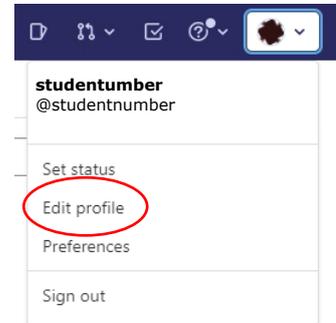


## EEECs Git Command Line Guidance

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**.



### Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

#### Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

##### Token name

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

##### Expiration date

##### Select scopes

Scopes set the permission levels granted to the token. [Learn more](#).

- api**  
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.
- read\_api**  
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- read\_user**  
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- read\_repository**  
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- write\_repository**  
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

Create personal access token

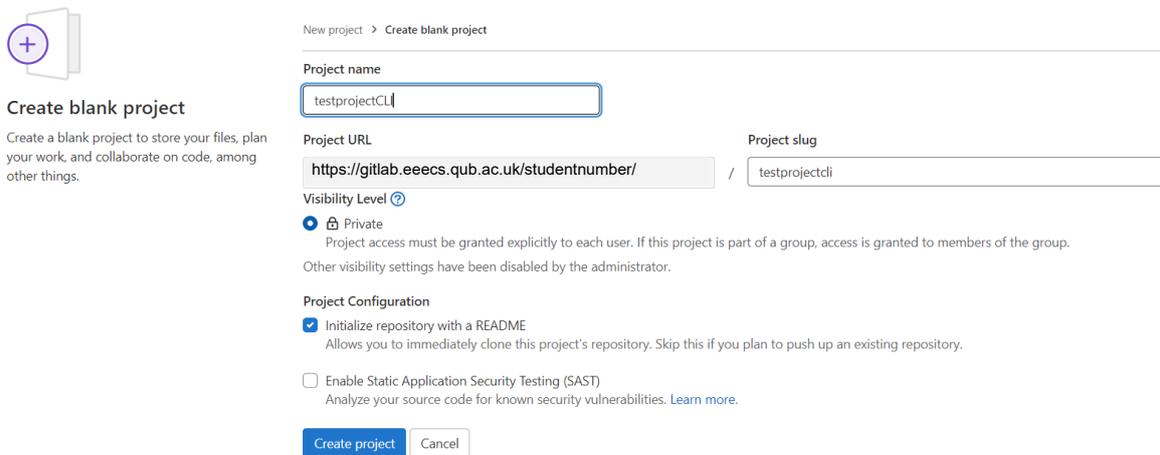
This will generate a token at the top of the page. Take a note of the access token by copying the token into notepad and saving it.

#### Your new personal access token

g: [REDACTED]

Make sure you save it - you won't be able to access it again.

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



New project > Create blank project

**Create blank project**  
Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name: testprojectCLI

Project URL: https://gitlab.eeecs.qub.ac.uk/studentnumber/ / Project slug: testprojectcli

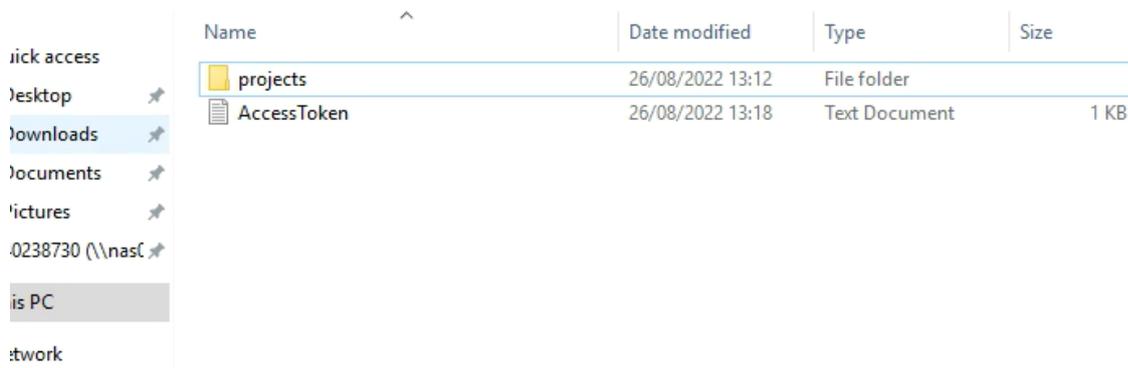
Visibility Level: Private  
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group. Other visibility settings have been disabled by the administrator.

Project Configuration

- Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
- Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more.](#)

Buttons: Create project, Cancel

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.



Name	Date modified	Type	Size
projects	26/08/2022 13:12	File folder	
AccessToken	26/08/2022 13:18	Text Document	1 KB

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"`



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**

```
Command Prompt
Microsoft Windows [Version 10.0.17763.3287]
(c) 2018 Microsoft Corporation. All rights reserved.

I:\>cd projects/testprojectcli

I:\projects\testprojectCLI>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
       index.html

nothing added to commit but untracked files present (use "git add" to track)

I:\projects\testprojectCLI>
```

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.

```
I:\projects\testprojectCLI>git add index.html

I:\projects\testprojectCLI>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       new file:   index.html
```

If you rerun the **git status** command it will show that Git is tracking the specified file. After adding the files to the staging environment, 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 if the changes have 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.

The screenshot shows the GitLab interface for a project named 'testprojectCLI'. At the top, it displays the project name, ID (5492), and statistics: 2 commits, 1 branch, 0 tags, and 41 KB of project storage. Below this, there are buttons for 'Find file', 'Web IDE', and 'Clone'. The main content area shows a commit titled 'First commit, created index page' by Jessica Phun, authored 21 minutes ago. Below the commit, there are several buttons for adding project files like README, LICENSE, CHANGELOG, and CONTRIBUTING. At the bottom, a table lists the files in the repository:

Name	Last commit	Last update
README.md	Initial commit	2 hours ago
index.html	First commit, created index page	21 minutes ago

## 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**