



Gitlab User Manual

The following document is a manual which aims to be a guide for agile work space GitLab

Introduction to GitLab

GitLab aims to bring teams together and tackle all aspects of the agile development in one free Development Operations Platform.

As the name entails the main software that GitLab runs off of would be Git, aka the Global Information Transformation. Git is a open source software that tracks file changes and branching to allow for programmers to collaborate with eachother while developing source code.

Accounts

A gitlab account is needed to access all the features of the agile development platform.

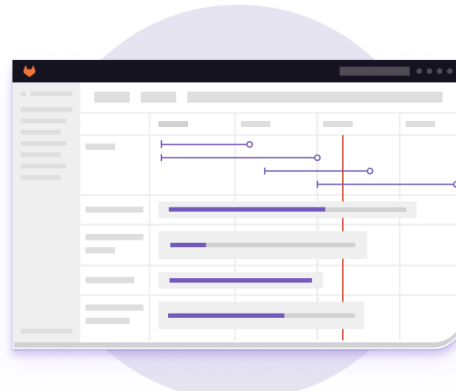
Account Creation

The home page for <https://gitlab.com> is just information about what the platform is capable of and features that you would recieve when you sign up. The portal to logging in and account creation is on the top right of the homepage with the button being `Login`.



The One DevOps Platform

From planning to production, bring teams together in one application. Ship secure code more efficiently to deliver value faster.



After hitting the **Login** button it will take you to the following screen where you can login if you already have an account, register for a dedicated gitlab account, or sign in using a third party account like google.



GitLab.com

Username or email

Password

☐ Remember me

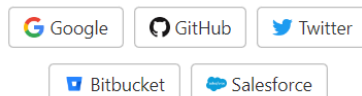
[Forgot your password?](#)

Sign in


By signing in you accept the [Terms of Use](#) and acknowledge the [Privacy Policy](#) and [Cookie Policy](#).

Don't have an account yet? [Register now](#)

Sign in with



To create an account the navigation you would go would either be the [Register now](#) hyperlink or signing in with the third party. After hitting the [Register now](#) hyperlink the following screen prompts to create an account as well as gives you another chance to sign in with a third party account.


GitLab.com

First name

Last name

Username


Email

We recommend a work email address.

Password

Minimum length is 8 characters.


☐ I'm not a robot



reCAPTCHA
[Privacy](#) - [Terms](#)


Register


By clicking Register or registering through a third party you accept the GitLab [Terms of Use](#) and acknowledge the [Privacy Policy](#) and [Cookie Policy](#)


Register with:

 Google

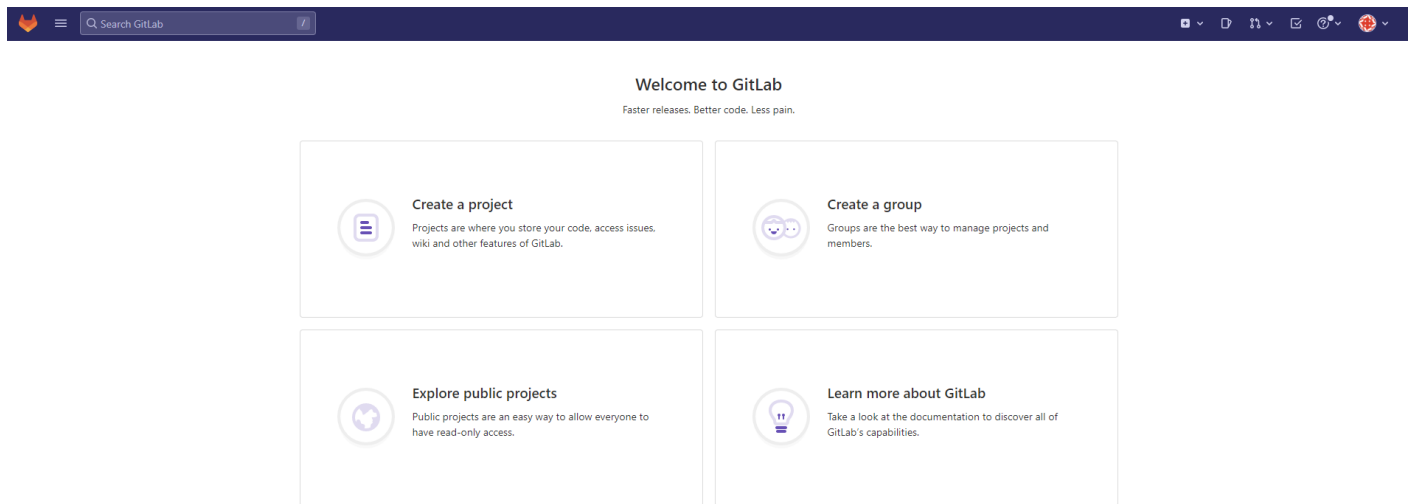
 GitHub

 Twitter

 Bitbucket

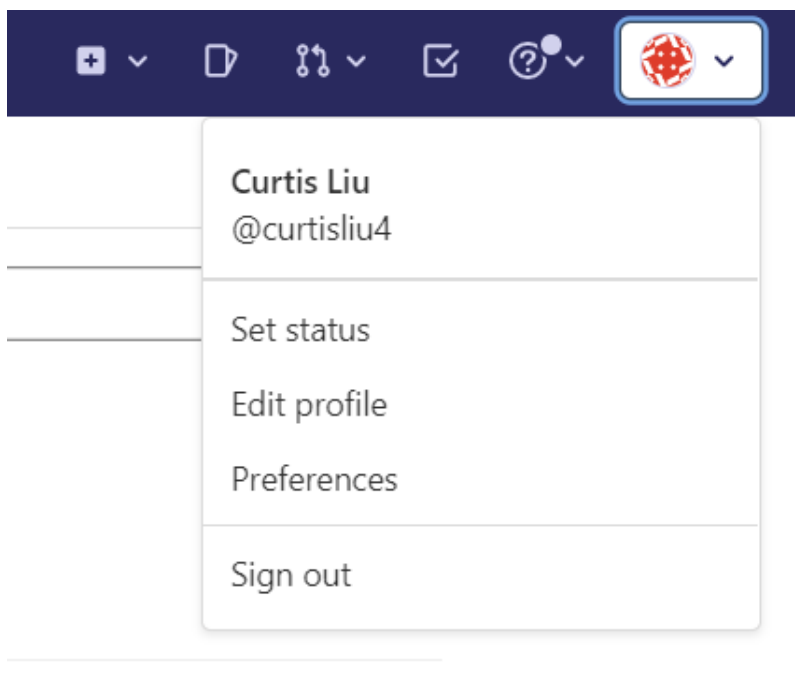
 Salesforce

There will be a verification for your account but once that is done you are logged in your homepage for [GitLab](#) should change to a completely different page where you have access to all of GitLab's functionality.

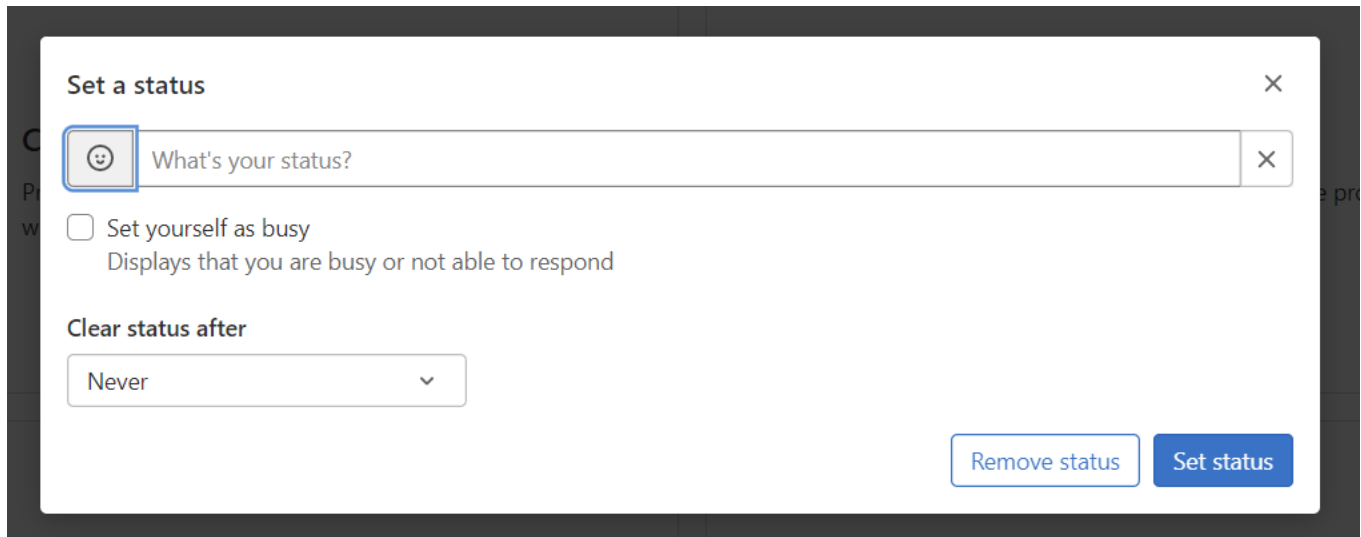


Basic Settings and Preferences

Just like any good collaboration space for developers, GitLab allows for you to change account information as well as preferences to add a little bit of uniqueness to your GitLab experience. The way to access this is from your homepage and hit the profile drop down on the top right.

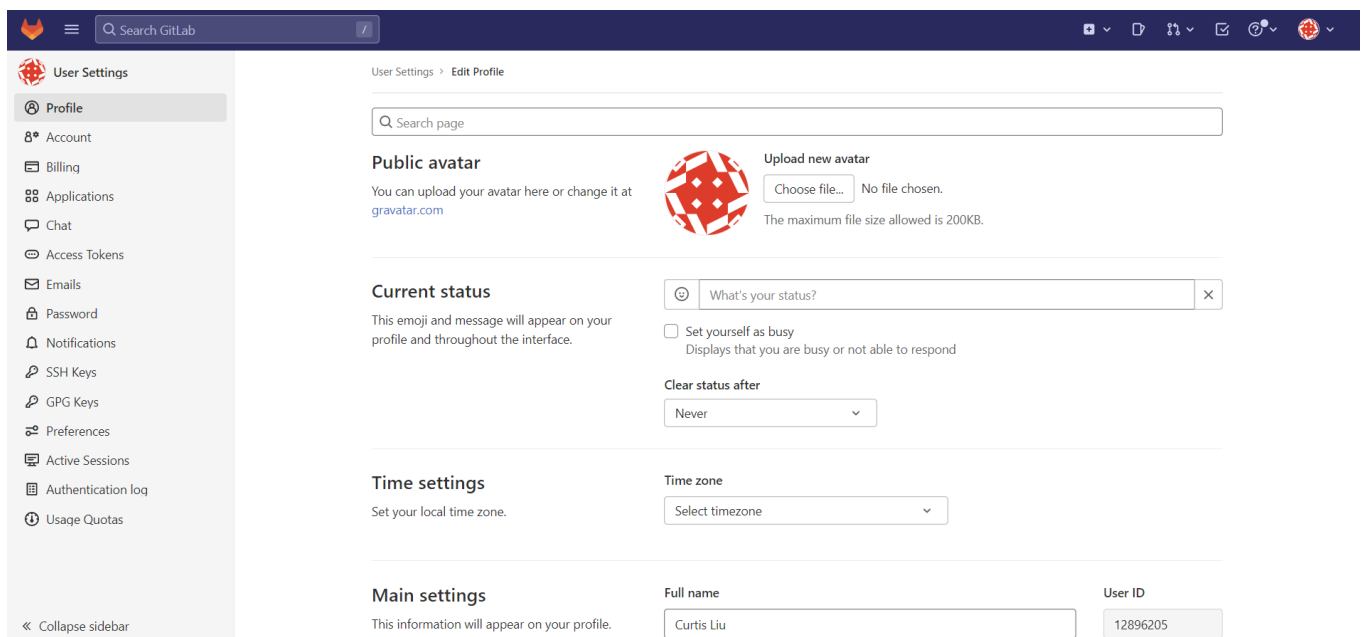


There are four options on this dropdown: **Set status**, **Edit profile**, **Preferences**, and **Sign out**. The most obvious one would be **Sign out** where it would take you back to the original GitLab page prior to being signed in to your account. You can also set your status with the **Set status** button which allows for you to broadcast to your fellow collaborators how you are currently feeling and if you are currently busy.



The screenshot shows a modal titled "Set a status" with a close button (X) in the top right corner. Inside the modal, there is a text input field with a smiley face icon and the placeholder text "What's your status?". Below the input field, there is a checkbox labeled "Set yourself as busy" with the subtext "Displays that you are busy or not able to respond". Underneath, there is a section titled "Clear status after" with a dropdown menu currently set to "Never". At the bottom right of the modal, there are two buttons: "Remove status" and "Set status".

The last two drop down options both take you to the User Settings in different tabs, with **Edit profile** going to the profile tab of User Settings. This page is where you can change your main profile settings like TimeZone, Avatar, and little about yourself in the User Information.



The screenshot shows the "Edit Profile" page in GitLab. On the left is a sidebar with "User Settings" and a list of tabs: Profile, Account, Billing, Applications, Chat, Access Tokens, Emails, Password, Notifications, SSH Keys, GPG Keys, Preferences, Active Sessions, Authentication log, and Usage Quotas. The "Profile" tab is selected. The main content area has a search bar at the top. Below it, the "Public avatar" section shows a red and white geometric avatar and an "Upload new avatar" button. The "Current status" section includes a status input field, a "Set yourself as busy" checkbox, and a "Clear status after" dropdown set to "Never". The "Time settings" section has a "Time zone" dropdown set to "Select timezone". The "Main settings" section shows the "Full name" as "Curtis Liu" and the "User ID" as "12896205".

As expected the **Preferences** button takes you to the Preferences tab of the User Settings and the following is what you can do in each tab:

- Profile: Houses your main profile settings
- Account: Allows for you to change your username and manage other accounts with 2FA (Two-Factor-Authentication)
- Billing: Current User Git Plan and monthly charge
- Applications: Place to manage applications that can use GitLab as an OAuth provider
- Chat: Shows your account's chats
- Account Tokens: Where you would create personal account tokens for your API's
- Emails: Place for you to link more emails to your Git Account
- Password: Where you change your password for your Git Account
- Notification: Manage your notifications for your emails and what notifications Git sends you
- SSH Keys: Key to access git from your computer
- GPG Keys: Key to allow for verification of commits
- Preferences: Customize how your Git looks through Themes and Styles
- Active Sessions: Shows GitLab session ran on which devices
- Authentication Log: Security log of authentication events
- Usage Quotas: GitLab usage charts

SSH Keys

All SSH Documentation comes directly from [here](#)

Prerequisites

To use SSH to communicate with GitLab, you need:

- The OpenSSH client, which comes pre-installed on GNU/Linux, macOS, and Windows 10.
- SSH version 6.5 or later. Earlier versions used an MD5 signature, which is not secure.

To view the version of SSH installed on your system, run `ssh -V`.

Generate an SSH key pair

If you do not have an existing SSH key pair, generate a new one:

1. Open a terminal.
2. Run `ssh-keygen -t` followed by the key type and an optional comment. This comment is included in the `.pub` file that's created. You may want to use an email address for the comment.

For example, for ED25519:

```
ssh-keygen -t ed25519 -C "<comment>"
```

For 2048-bit RSA:

```
ssh-keygen -t rsa -b 2048 -C "<comment>"
```

3. Press Enter. Output similar to the following is displayed:

```
Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/user/.ssh/id_ed25519):
```

4. Accept the suggested filename and directory, unless you are generating a deploy key or want to save in a specific directory where you store other keys.

You can also dedicate the SSH key pair to a specific host.

5. Specify a passphrase

```
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:
```

A confirmation is displayed, including information about where your files are stored.

A public and private key are generated. Add the public SSH key to your GitLab account and keep the private key secure.

GPG

All GPG documentation comes directly from [here](#)

Create a GPG key

If you don't already have a GPG key, create one:

1. [Install GPG](#) for your operating system. If your operating system has `gpg2` installed, replace `gpg` with `gpg2` in the commands on this page.
2. To generate your key pair, run the command appropriate for your version of `gpg`:

```
# Use this command for the default version of GPG, including  
# Gpg4win on Windows, and most macOS versions:  
gpg --gen-key  
  
# Use this command for versions of GPG later than 2.1.17:  
gpg --full-gen-key
```

3. Select the algorithm your key should use, or press Enter to select the default option, `RSA` and `RSA`.
4. Select the key length, in bits. GitLab recommends 4096-bit keys.
5. Specify the validity period of your key. This value is subjective, and the default value is no expiration.

6. To confirm your answers, enter **y**.
7. Enter your name.
8. Enter your email address. It must match a verified email address in your GitLab account.
9. Optional. Enter a comment to display in parentheses after your name.
10. GPG displays the information you've entered so far. Edit the information or press O (for **Okay**) to continue.
11. Enter a strong password, then enter it again to confirm it.
12. To list your private GPG key, run this command, replacing **<EMAIL>** with the email address you used when you generated the key:

```
```\ngpg --list-secret-keys --keyid-format LONG <EMAIL>\n```
```

13. In the output, identify the **sec** line, and copy the GPG key ID. It begins after the **/** character. In this example, the key ID is **30F2B65B9246B6CA**:

```
```\nsec   rsa4096/30F2B65B9246B6CA 2017-08-18 [SC]\n      D5E4F29F3275DC0CDA8FFC8730F2B65B9246B6CA\nuid           [ultimate] Mr. Robot <your_email>\nssb    rsa4096/B7ABC0813E4028C0 2017-08-18 [E]\n```\n
```

14. To show the associated public key, run this command, replacing **<ID>** with the GPG key ID from the previous step:

```
```\ngpg --armor --export <ID>\n```\n
```

15. Copy the public key, including the `BEGIN PGP PUBLIC KEY BLOCK` and `END PGP PUBLIC KEY BLOCK` lines. You need this key in the next step.

## Add a GPG key to your account

To add a GPG key to your user settings:

1. Sign in to GitLab.
  2. In the top-right corner, select your avatar.
  3. Select **Edit profile**.
  4. On the left sidebar, select **GPG Keys** ().
  5. In **Key**, paste your *public* key.
  6. To add the key to your account, select **Add key**. GitLab shows the key's fingerprint, email address, and creation date:
- 

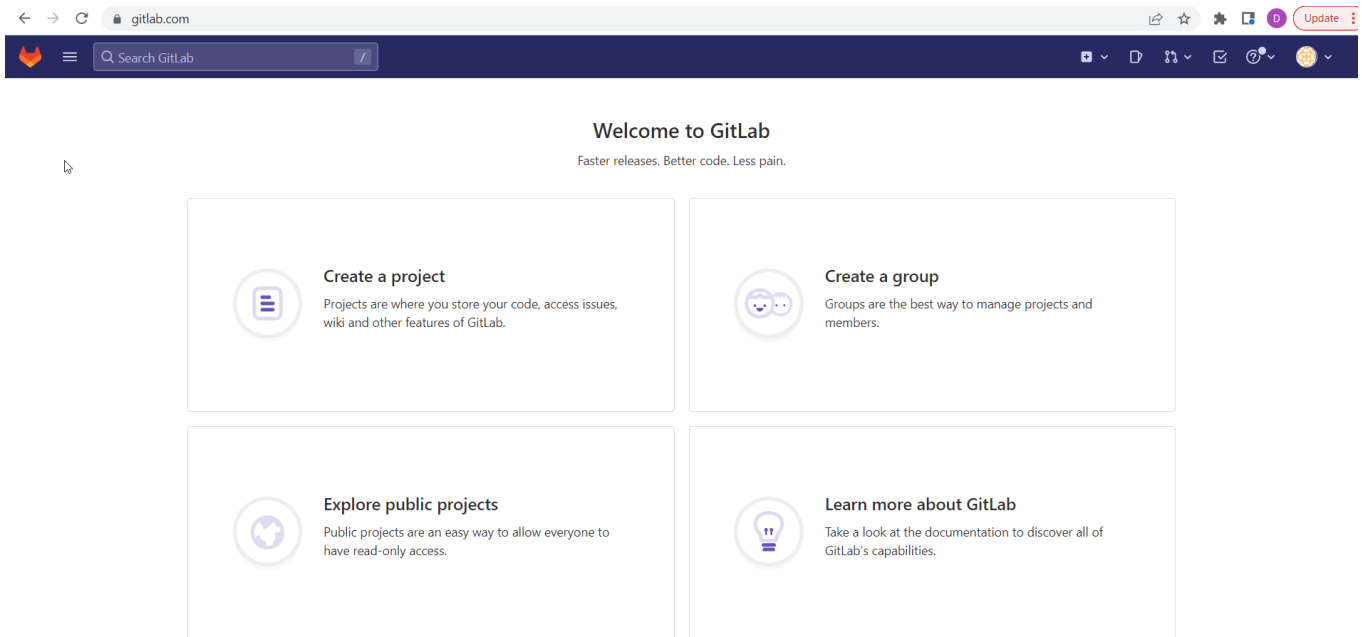
## Repositories

A **repository** is a directory or folder that is stored in your GitLab account. GitLab acts as a version control system for your repository, which means that GitLab allows changes to your repository to be easily moderated. In this **repository** section of the manual, you will learn how to create a remote repository in GitLab, clone that repository to your local machine, and push changes from your local repository to your remote GitLab repository.

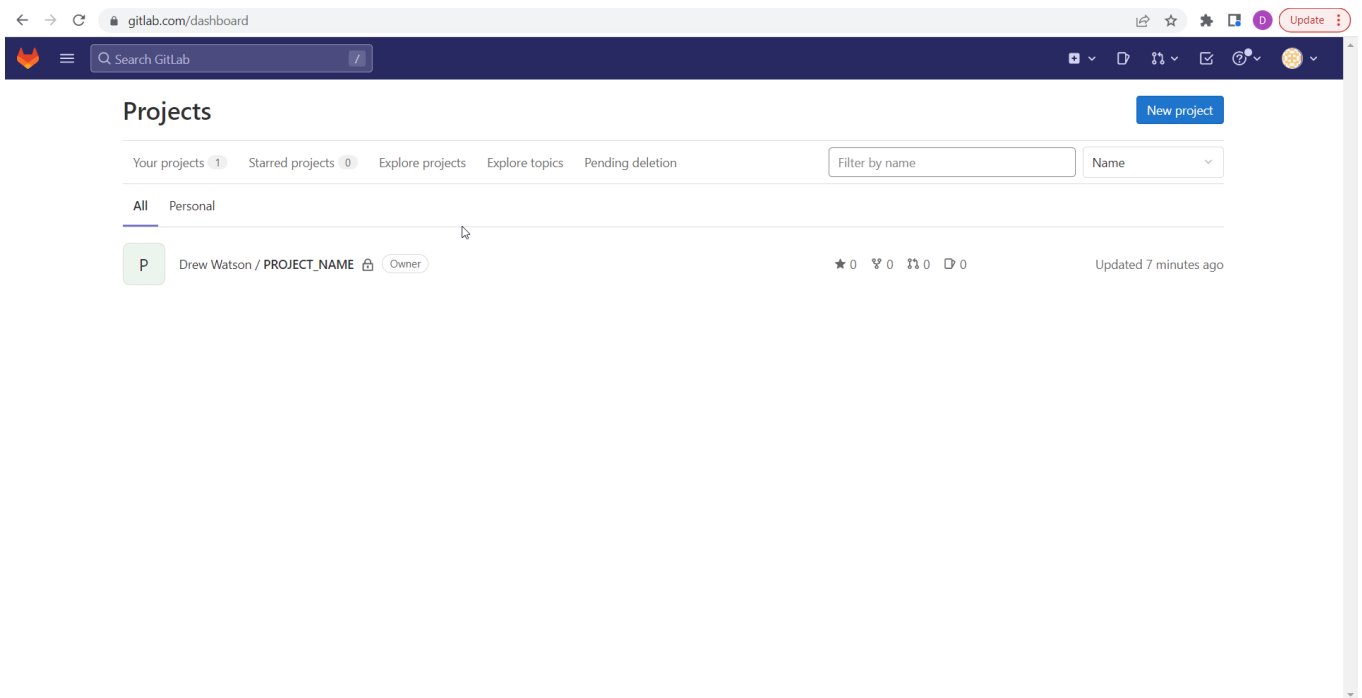
### Creating a Remote Repository in GitLab

GitLab provides a way to very easily create a repository on your account. To create a repository, you must have a GitLab account. For information on how to create a GitLab account, refer to the "Creating Account" section above. The repository hosted on your GitLab account is referred to as a **remote** repository, because it is stored remotely on the web.

1. Navigate to your GitLab Dashboard. If you are currently signed in to GitLab, then navigating to <https://gitlab.com/> should take you to the dashboard. The dashboard page looks like this if you don't have any projects:



If you do have projects, the dashboard looks like this:



2. Click "Create a Project" (or "New project"). This should navigate you to <https://gitlab.com/projects/new>.
3. Click on "Create Blank Project". If you wish, you may also explore available templates by clicking "Create from Template".
4. You should see a page that has text boxes corresponding to "Project name" and "Project slug". The URL of the webpage is [https://gitlab.com/projects/new#blank\\_project](https://gitlab.com/projects/new#blank_project). Here's what it looks like:

gitlab.com/projects/new#blank\_project

Search 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**

My awesome project

**Project URL**

https://gitlab.com/drewzidrew1/

**Project slug**

my-awesome-project

Want to organize several dependent projects under the same namespace? [Create a group.](#)

**Project deployment target (optional)**

Select the deployment target

**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.

☐ Public  
The project can be accessed without any authentication.

**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.](#)

Create project Cancel

5. Type the name of your project into the text box that says "My awesome project". This should automatically populate the "Project slug" text box. If you would like to have a different project slug, replace the automatically generated text in the text box with whatever you would like your slug to be.
6. Make sure the check box that says "Initialize repository with a README" is checked.

gitlab.com/projects/new#blank\_project

Search 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**

PROJECT\_NAME

**Project URL**

https://gitlab.com/drewzidrew1/

**Project slug**

project\_slug

Want to organize several dependent projects under the same namespace? [Create a group.](#)

**Project deployment target (optional)**

Select the deployment target

**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.

☐ Public  
The project can be accessed without any authentication.

**Project Configuration**

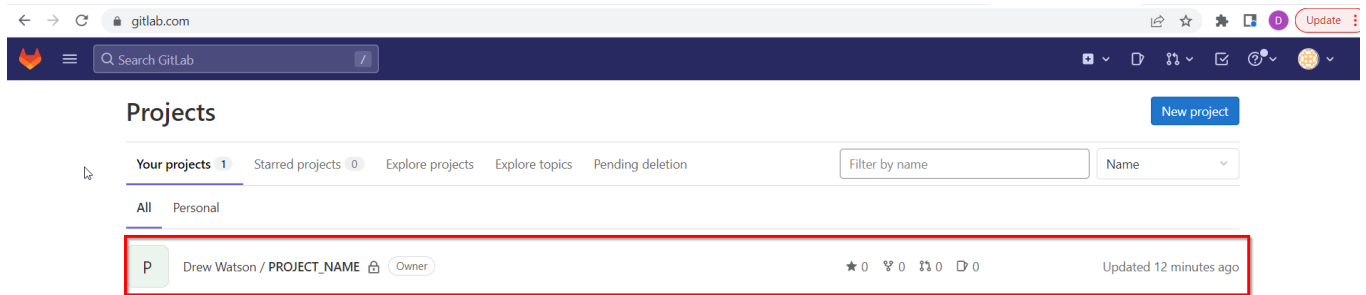
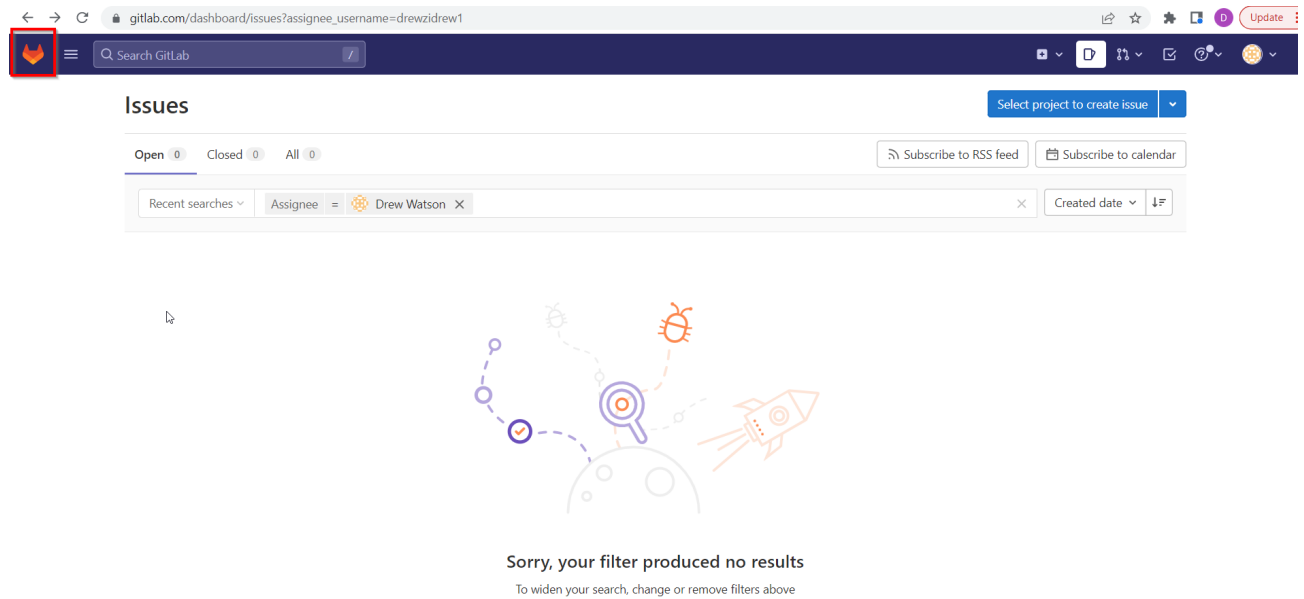
☒ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more.](#)

Create project Cancel

7. (OPTIONAL) If you would like to set a deployment target or change the Visibility Level of your repository, you may do so now.
8. Click the button that says "Create project".

- This will create the repository, and navigate you to the webpage for your new repository.
- To find your repository from any point on the GitLab website, just click on the fox icon in the top left, look for your repository in the list, and click on it:



## Cloning a Repository

To be able to make edits to the code in your repository, you have to be able to **clone** your remote GitLab repository onto your local machine. Cloning a repository will allow you to have access to the code in your remote repository on your local machine. To clone a repository, you will need to have access to a terminal on your computer. For this demonstration, Git Bash will be the terminal used. If you would like to download Git Bash, consult <https://git->

[scm.com/downloads](https://gitlab.com/downloads). After you have a terminal on your machine that you are comfortable using, you can follow this guide to clone a repository onto your local system.

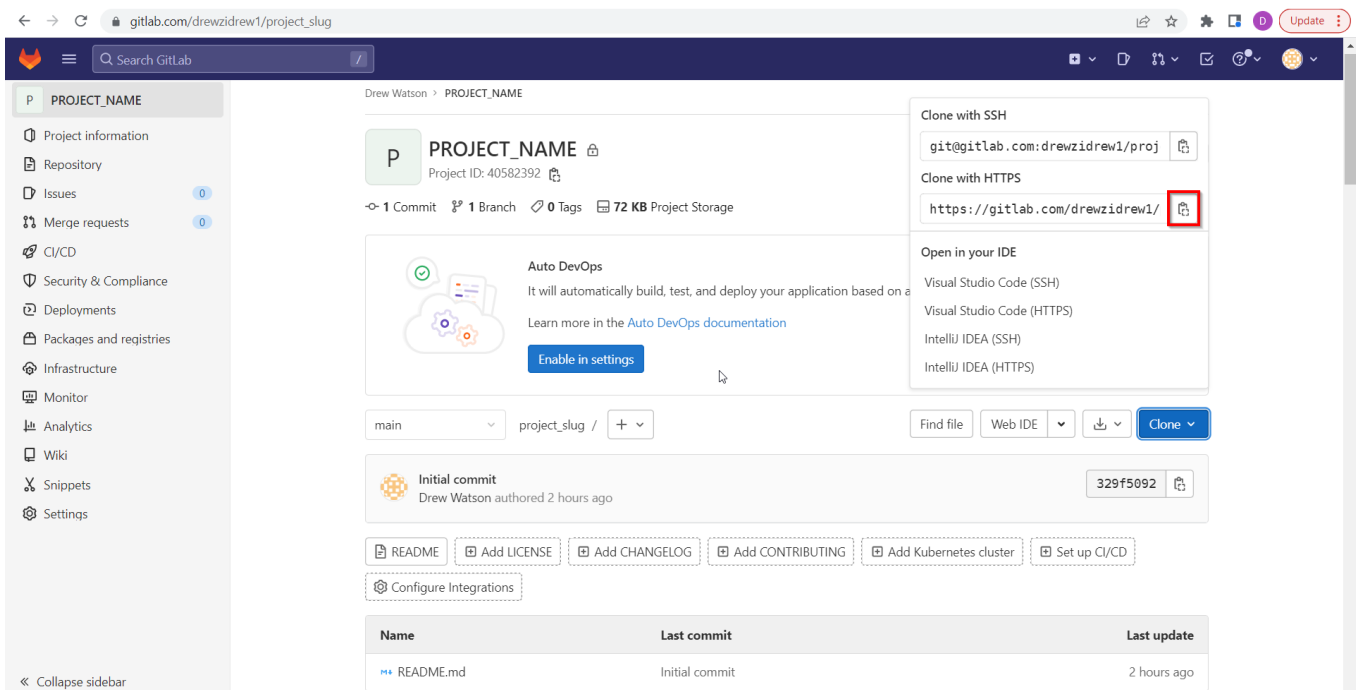
There are two primary methods of cloning a repository: Using HTTPS, and using SSH. Both methods will be discussed in this guide.

## Cloning with HTTPS

1. On GitLab, navigate to your desired repository. Navigating to an existing repository in GitLab is outlined in Step 10 of the "Creating a Repository in Gitlab" section above, or by typing a link of the form [https://gitlab.com/{YOUR\\_USERNAME\\_HERE}/{YOUR\\_PROJECT\\_SLUG\\_HERE}](https://gitlab.com/{YOUR_USERNAME_HERE}/{YOUR_PROJECT_SLUG_HERE}).
2. Click on the button that says "Clone":

"Pasted image 20221104111859.png" is not created yet. Click to create.

4. Click the clipboard icon next to the text box labeled "Clone with HTTPS":



This will copy the HTTPS address of the repository to your clipboard.

4. Open the terminal of your choice, and navigate to the directory you would like to put your repository in using the `cd` command. If you need to create a directory to house your repository, you can use the `mkdir` command to create one.

```
drewz@LAPTOP-3CSTM3SD MINGW64 ~
$ cd School_Stuff/Fall_2022/Technical_Communication/

drewz@LAPTOP-3CSTM3SD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication
$ mkdir Cloning_Directory

drewz@LAPTOP-3CSTM3SD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication
$ cd Cloning_Directory

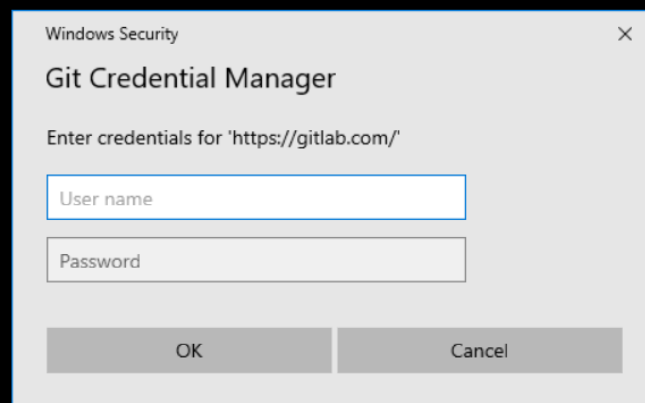
drewz@LAPTOP-3CSTM3SD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory
$
```

5. Input the following command:

```
$ git clone {Paste URL you copied in Step 3}
```

Note that in Git Bash you can paste text by right clicking with the mouse and then selecting "Paste". Press enter to execute the command. If prompted, enter your Gitlab username and password. If you were prompted for your username and password, the output should look like this:

```
drewz@LAPTOP-3CSTM3SD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory
$ git clone https://gitlab.com/drewzidrew1/project_slug.git
Cloning into 'project_slug'...
```



A Windows Security dialog box titled "Git Credential Manager". It prompts the user to "Enter credentials for 'https://gitlab.com/'". There are two input fields: "User name" and "Password". At the bottom, there are "OK" and "Cancel" buttons.

Otherwise, the output should look something like this:

```
$ git clone https://gitlab.com/drewzidrew1/project_slug.git
Cloning into 'project_slug'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

6. Enter the command `$ ls` into your terminal. This will list directories and files inside your current working directory. You should see a directory whose name matches the **project slug** of your repository. Navigate into that directory using `$ cd {Directory Name}`.

```

drewz@LAPTOP-3CSTMJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory
$ ls
project_slug/

drewz@LAPTOP-3CSTMJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory
$ cd project_slug/

drewz@LAPTOP-3CSTMJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory/project_slug (main)
$

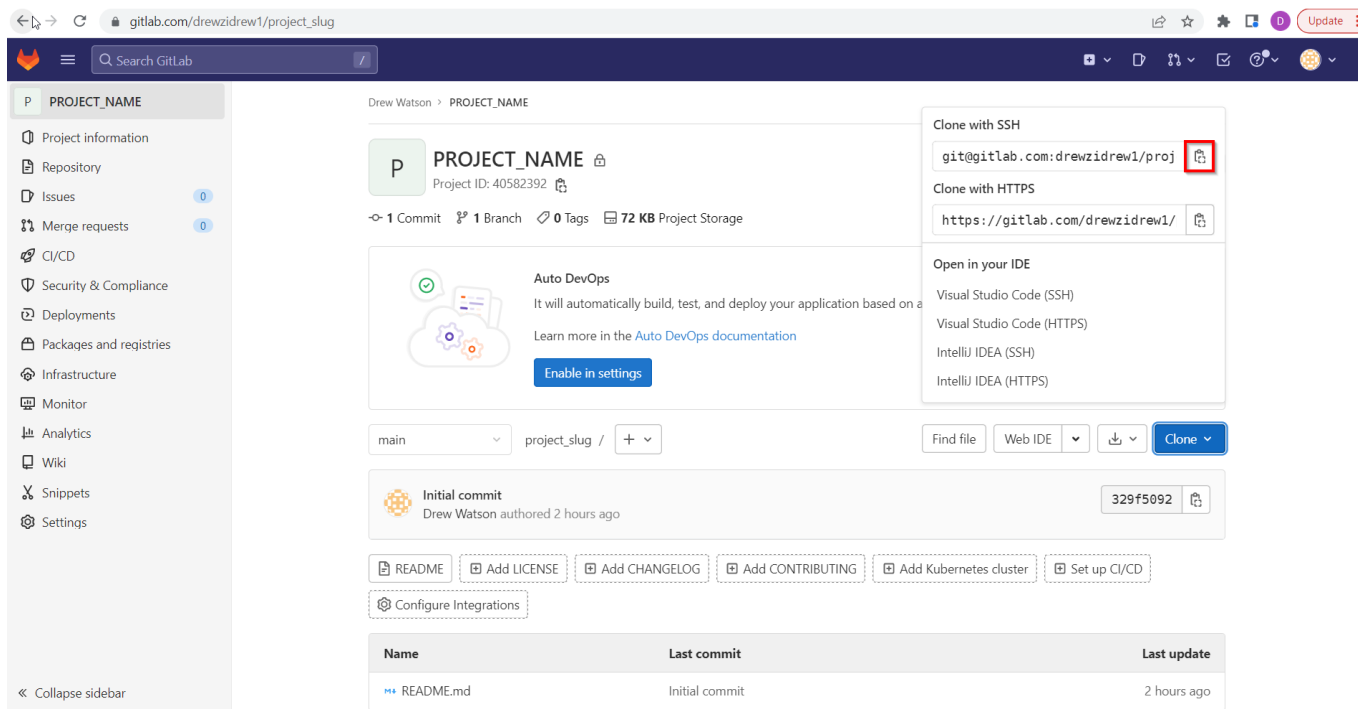
```

7. The directory which has the same name as your **project slug** is the cloned repository. In Git Bash, there will be a word in parenthesis (probably "main"), which indicates that you are inside your repository!
8. Congratulations! You have successfully cloned your remote Gitlab repository using HTTPS!

## Cloning with SSH

If you would prefer cloning the repository using SSH, you must have an SSH key associated with your machine on your GitLab account. For instructions on how to do this, consult the "Creating an SSH Key" guide in the "Creating Account" section above.

1. On Gitlab, navigate to your desired repository. Navigating to an existing repository in GitLab is outlined in Step 10 of the "Creating a Repository in GitLab" section above, or by typing a link of the form [https://gitlab.com/{YOUR\\_USERNAME\\_HERE}/{YOUR\\_PROJECT\\_SLUG\\_HERE}](https://gitlab.com/{YOUR_USERNAME_HERE}/{YOUR_PROJECT_SLUG_HERE}).
2. Click on the button that says "Clone"
3. Click the clipboard icon next to the text box labeled "Clone with SSH":



4. Follow Steps 4-7 in the "Cloning with HTTPS" guide above. Some of the screenshots will look slightly different from your output; don't be alarmed by this.



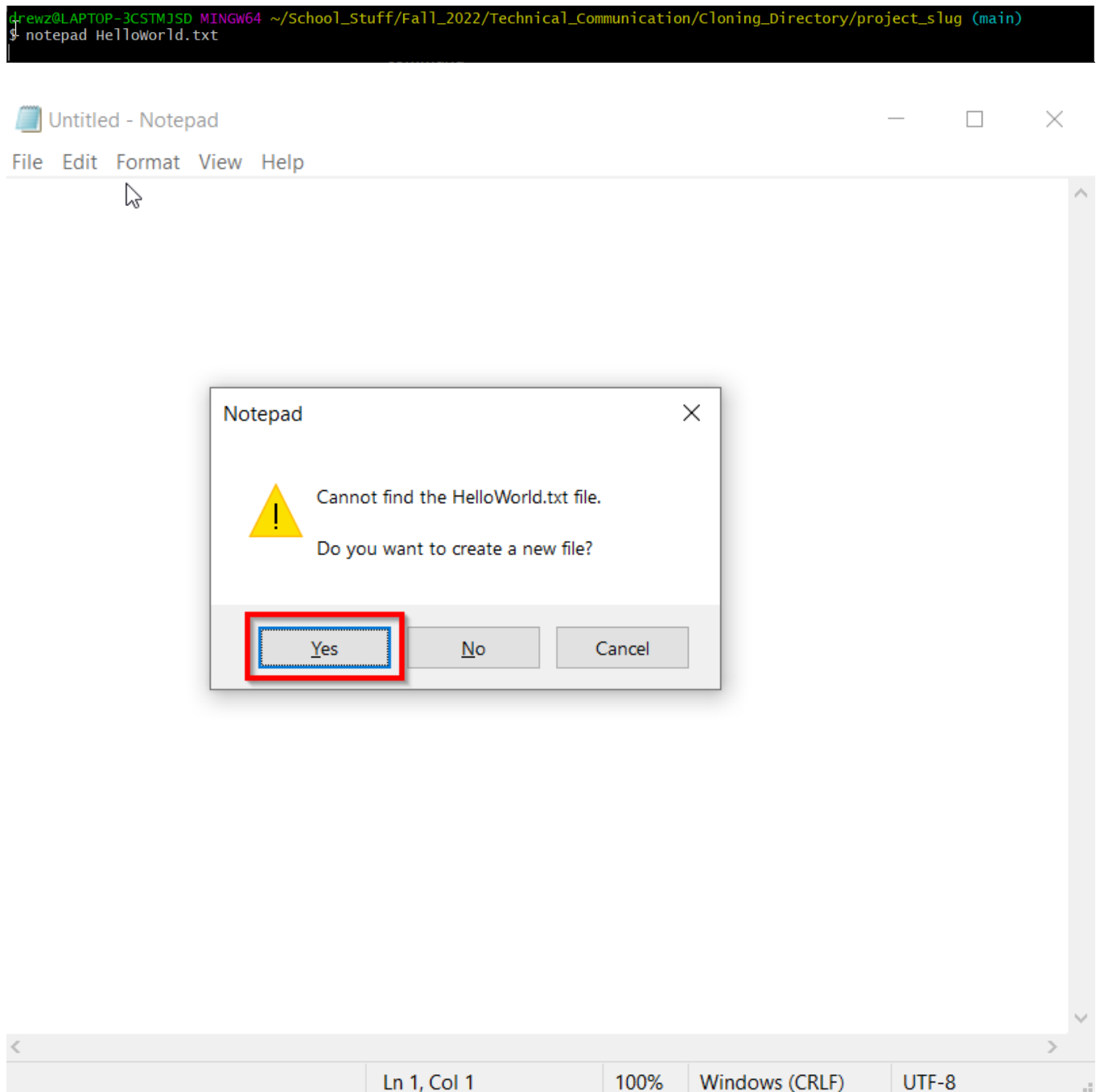
# Pushing Code to a Repository

To **push** to a repository means to take changes you've made to your local repository and put them into your repository on GitLab. This requires that you have your repository created in GitLab, and cloned on your local machine. For the purposes of this demonstration, we will just add a `.txt` file with some text. However, pushing will usually be done with code or documentation files in practice.

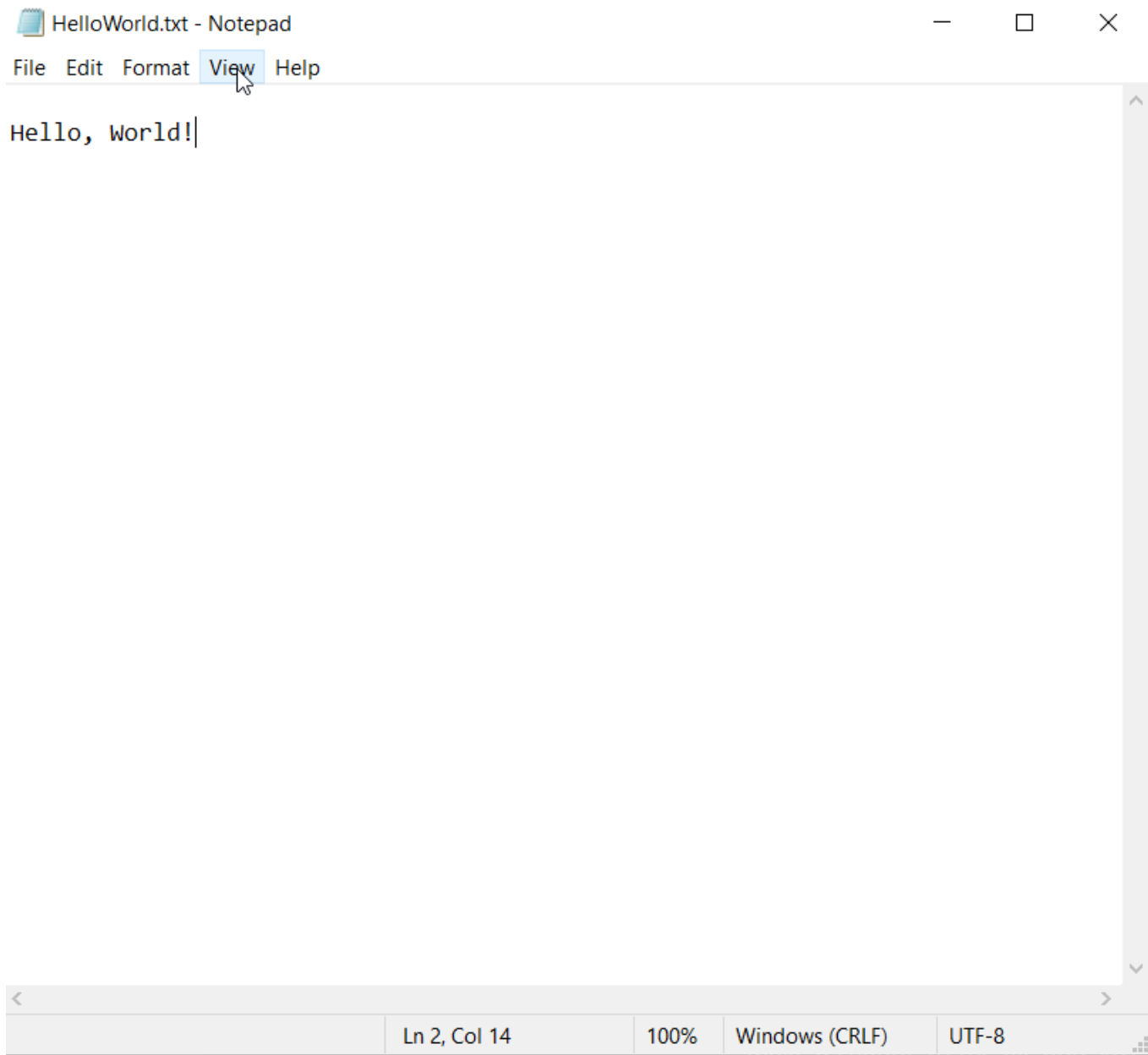
1. Create a file named "HelloWorld.txt" and open it. On Windows, this can be done by typing the command

```
$ notepad HelloWorld.txt
```

and then clicking "yes" if prompted to create a new file.



2. Make a change to the file and save it. You can now exit your text editor.



3. You now have a change on your local repository that is not yet on the remote repository on GitLab. To view your changes, enter the command `$ git status` :

```
drewz@LAPTOP-3C51HJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory/project_slug (main)
$ 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)
 HelloWorld.txt

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

4. Sets of changes are pushed to the remote GitLab repository in groups. A group of changes to be pushed to the remote repository is called a **commit**. To add a file to a commit, use the command `$ git add { file name(s) }`, or use `$ git add .` to add everything in your

current working directory. You can then do another `$ git status` to verify that your file has been added to the commit:

```
drewz@LAPTOP-3CSTMJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory/project_slug (main)
$ git add HelloWorld.txt

drewz@LAPTOP-3CSTMJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory/project_slug (main)
$ 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: HelloWorld.txt
```

5. Now that you have added the files you wish to commit and push to GitLab, its time to make the commit. This wraps your changes up nice and neatly so that you can push them as a group, and come back to them later if needed. To commit your added changes, input this command:

```
$ git commit -m "{Enter a message about your commit here}"
```

```
drewz@LAPTOP-3CSTMJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory/project_slug (main)
$ git commit -m "Created a text file called HelloWorld.txt"
[main e4abc02] Created a text file called HelloWorld.txt
1 file changed, 2 insertions(+)
create mode 100644 HelloWorld.txt
```

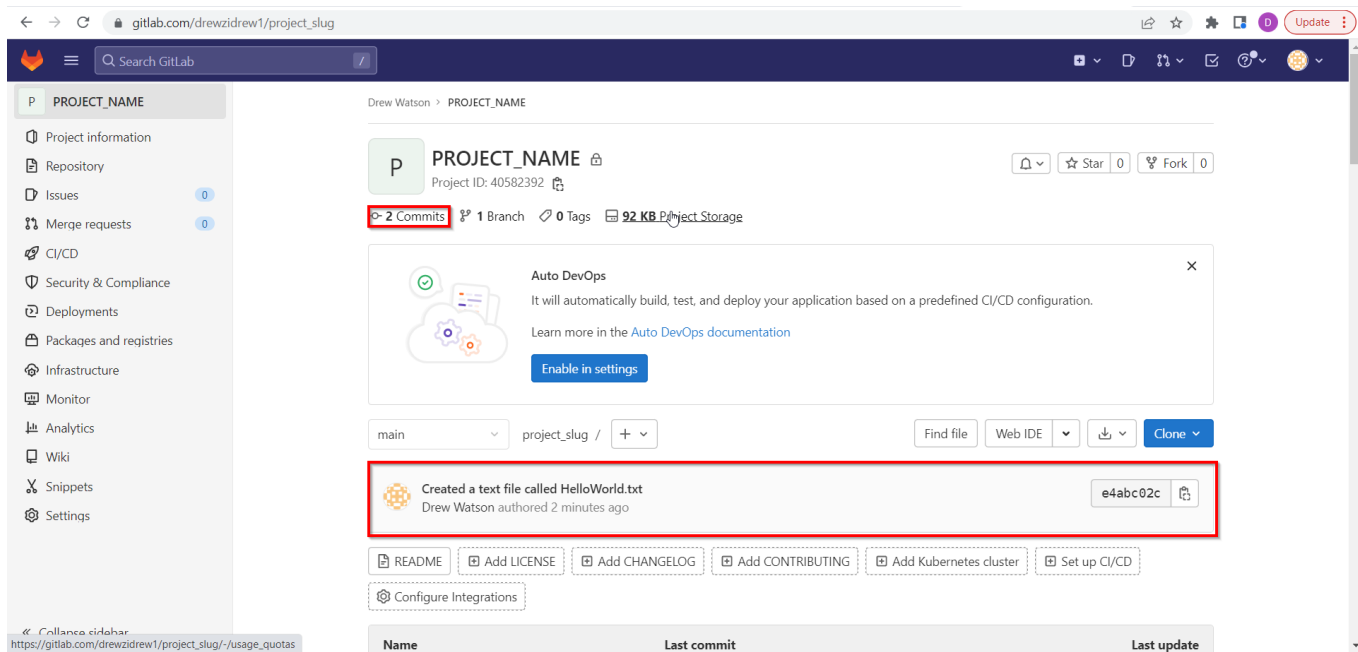
6. Now that you have made your commit, you can push that commit to GitLab. To do this, use the command :

```
$ git push
```

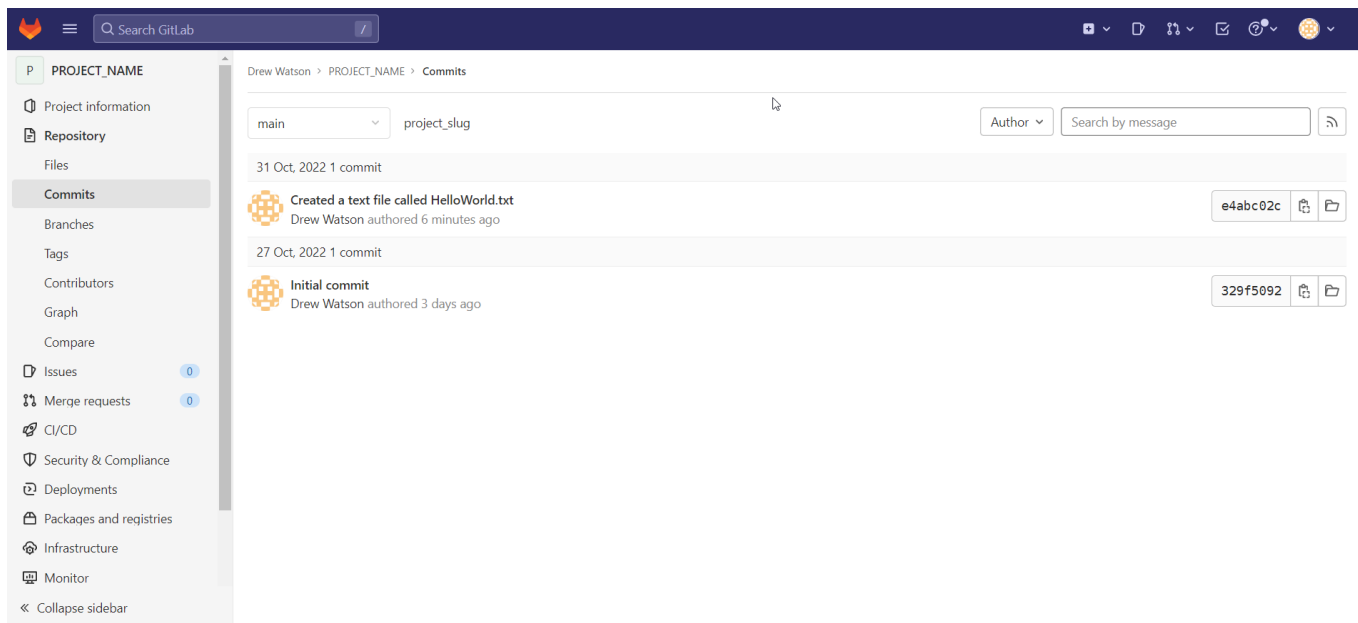
If you've done things correctly, your output should look similar to the screenshot below:

```
drewz@LAPTOP-3CSTMJSD MINGW64 ~/School_Stuff/Fall_2022/Technical_Communication/Cloning_Directory/project_slug (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 322 bytes | 161.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To gitlab.com:drewzidrew1/project_slug.git
 329f509..e4abc02 main -> main
```

7. Now that you've pushed your commit, you should be able to see your commit on your GitLab repository. It is a good idea to verify that your commit was pushed to GitLab correctly. To find your commit, navigate to your GitLab repository. You should be able to see the latest commit in a card right underneath the button labelled "Clone":



If not, you can click on the "Commits" button (highlighted in red above). This will take you to a page that contains all commits to your repository.



The bolded text on the commit should match the commit message you inputted in Step 5. Congratulations! You've successfully pushed changes you made on your local machine to your remote GitLab repository.

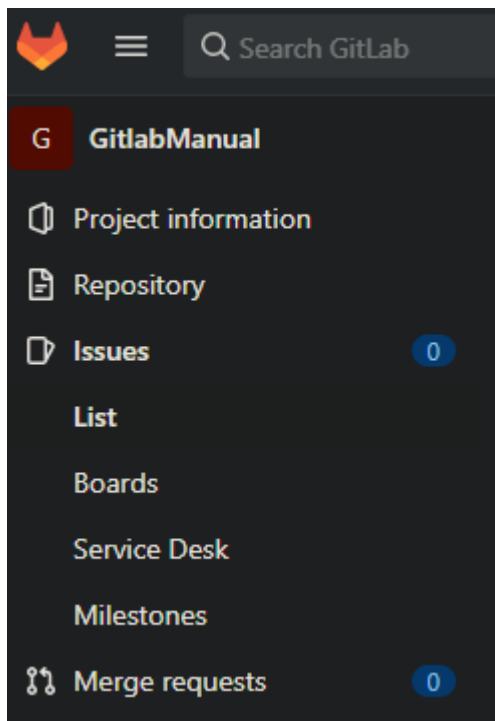
# Collaboration

## Issues

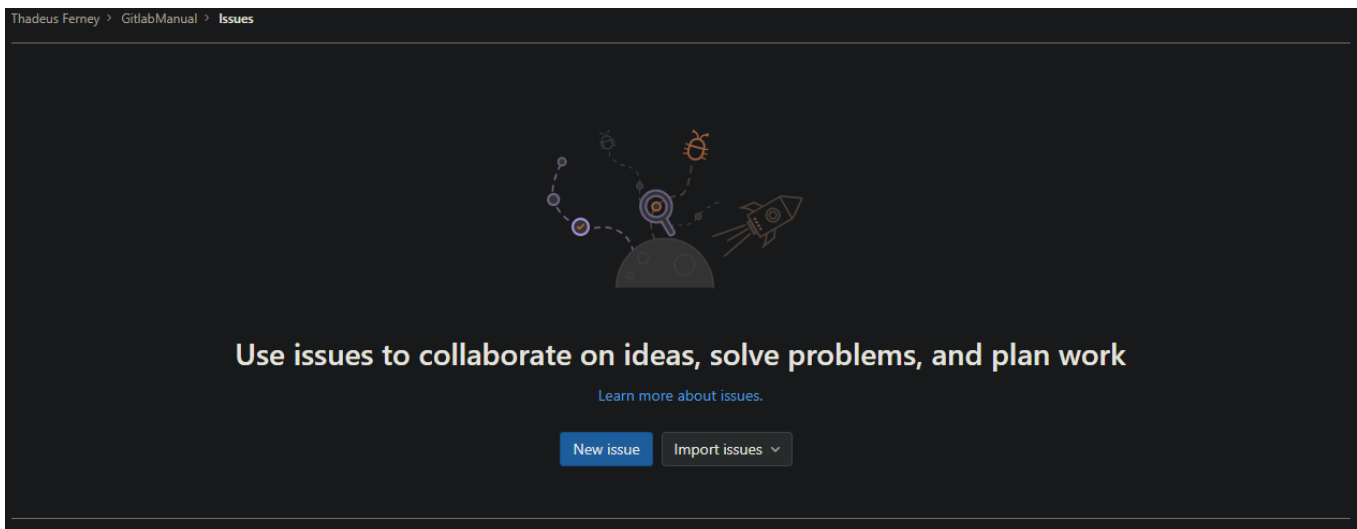
A useful feature of GitLab is the advanced issue tracking. An issue is any bug or feature request that is to be added. These can be created by the end user through the service desk or by the developers themselves. This can be used as a Scrum board (similar to kanban board) to better track what each team member is doing. Each issue has multiple properties to aid in sorting and visualize.

## Issue Creation

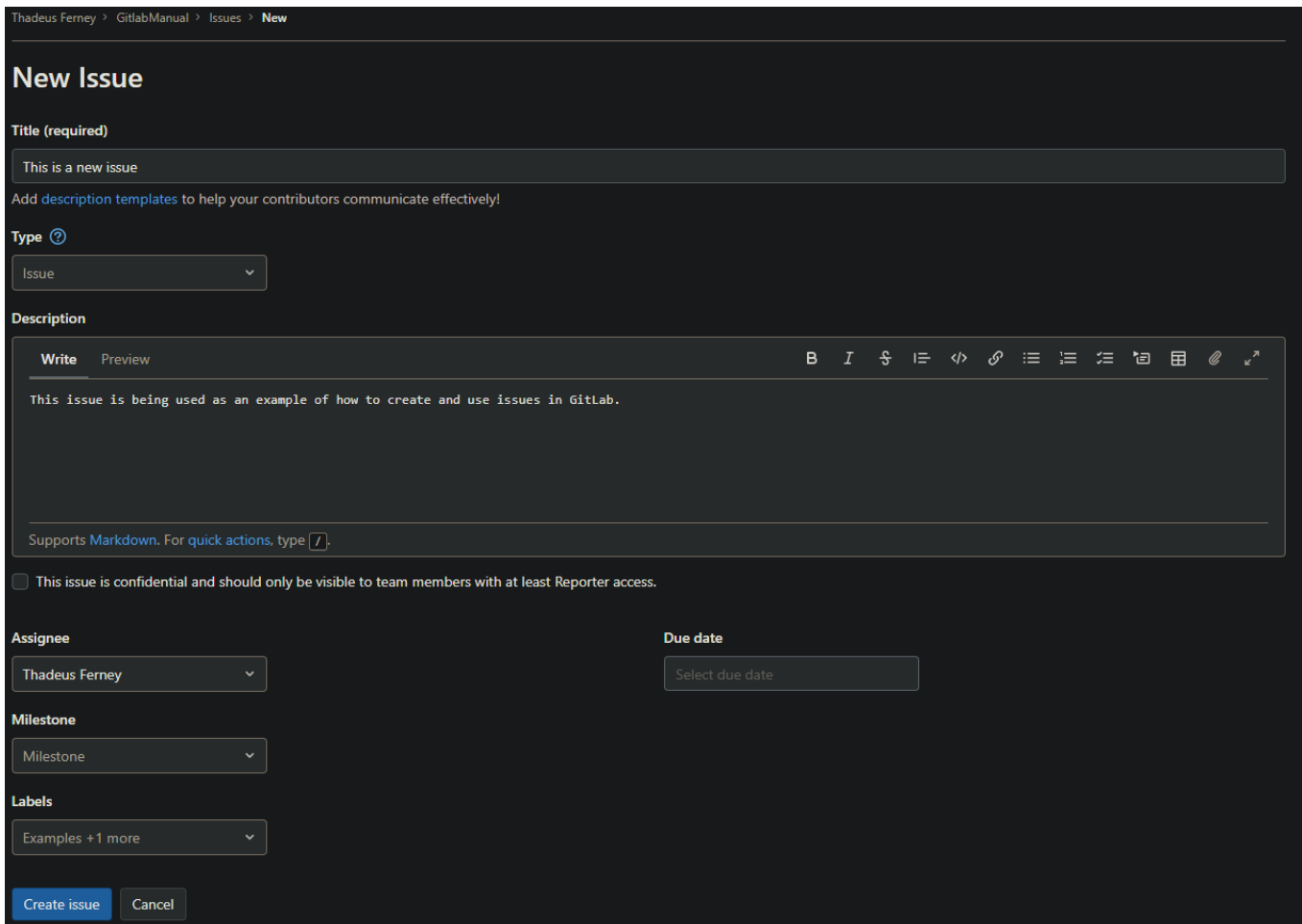
1. Navigate to either List or Boards under the issues tab. In this example we will use the List tab.



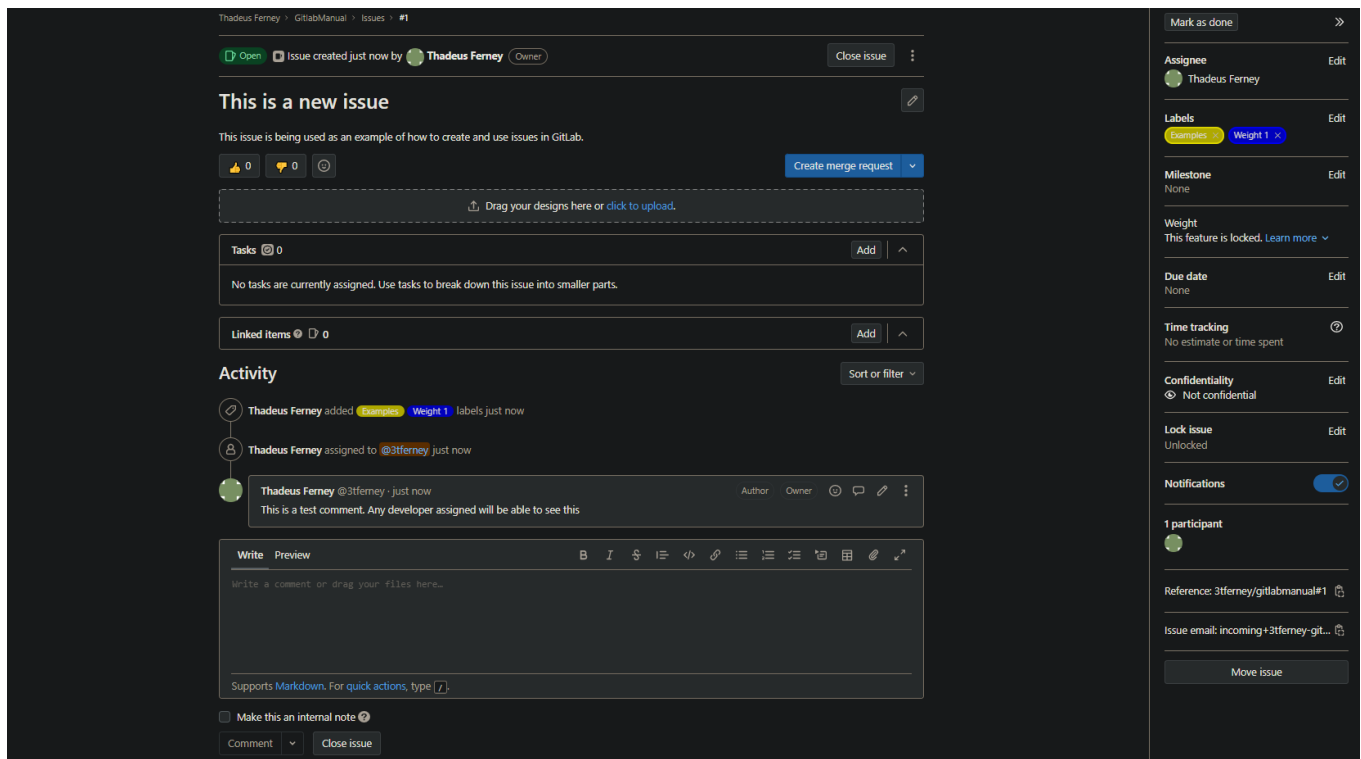
2. Then select the New Issue button to bring up the Issue creation page.



3. The issue needs a name. To better use issues, it is recommended to give each issue a type, description, and applicable labels. For collaboration you can assign any developer in the repository quickly. We will see this on the issue board later.

The image displays the 'New Issue' form in GitLab. The breadcrumb trail at the top is 'Thadeus Ferney > GitlabManual > Issues > New'. The form has a title 'New Issue'. Under 'Title (required)', there is a text input field containing 'This is a new issue'. Below the title, a note says 'Add [description templates](#) to help your contributors communicate effectively!'. The 'Type' section has a dropdown menu set to 'Issue'. The 'Description' section features a rich text editor with a 'Write' tab selected and a 'Preview' tab. The editor contains the text 'This issue is being used as an example of how to create and use issues in GitLab.' and a note 'Supports Markdown. For quick actions, type [Z]'. There is a checkbox labeled 'This issue is confidential and should only be visible to team members with at least Reporter access.' which is currently unchecked. Below this, there are four sections: 'Assignee' with a dropdown set to 'Thadeus Ferney', 'Due date' with a 'Select due date' button, 'Milestone' with a dropdown set to 'Milestone', and 'Labels' with a dropdown set to 'Examples +1 more'. At the bottom left, there are two buttons: 'Create issue' and 'Cancel'.

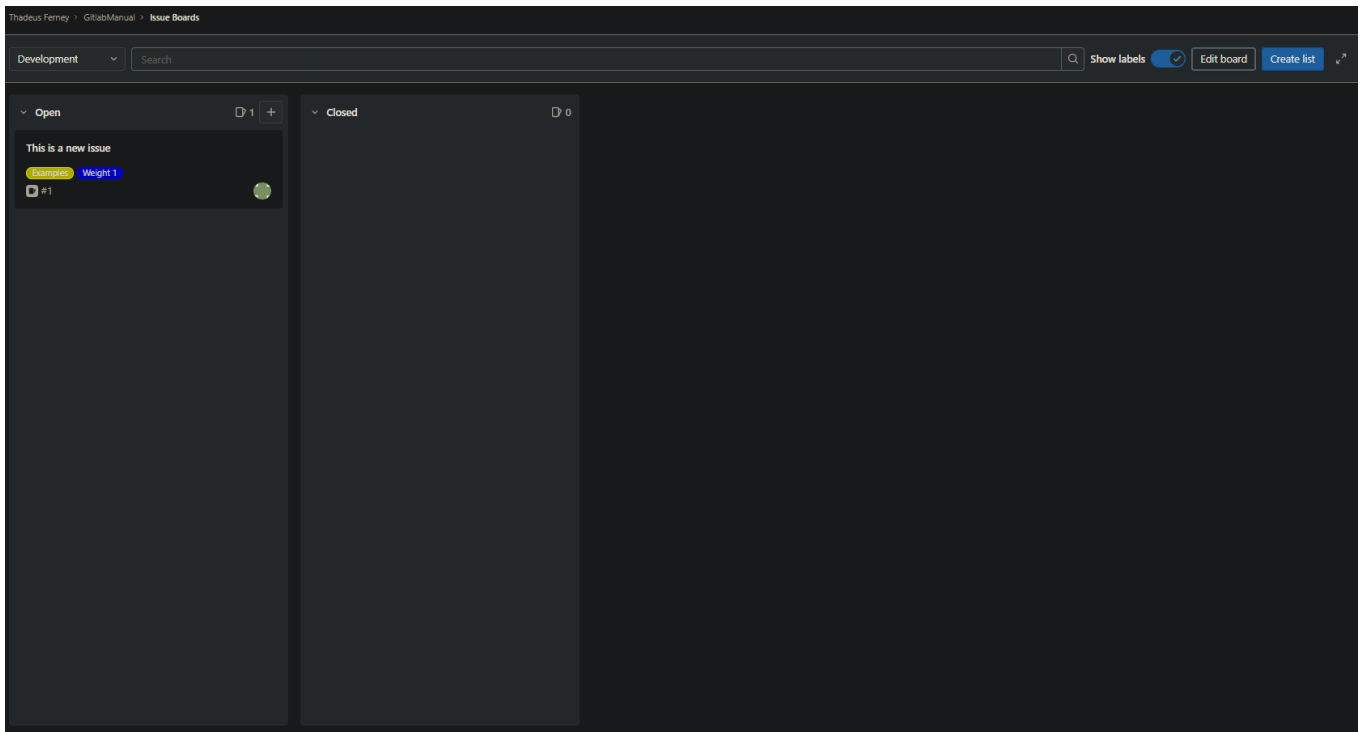
4. Finally select the create issue button to finish creating your issue. This will take you to the issue's new page. This is a great place to see all of the detail relating to the issue and update any information that is missing.



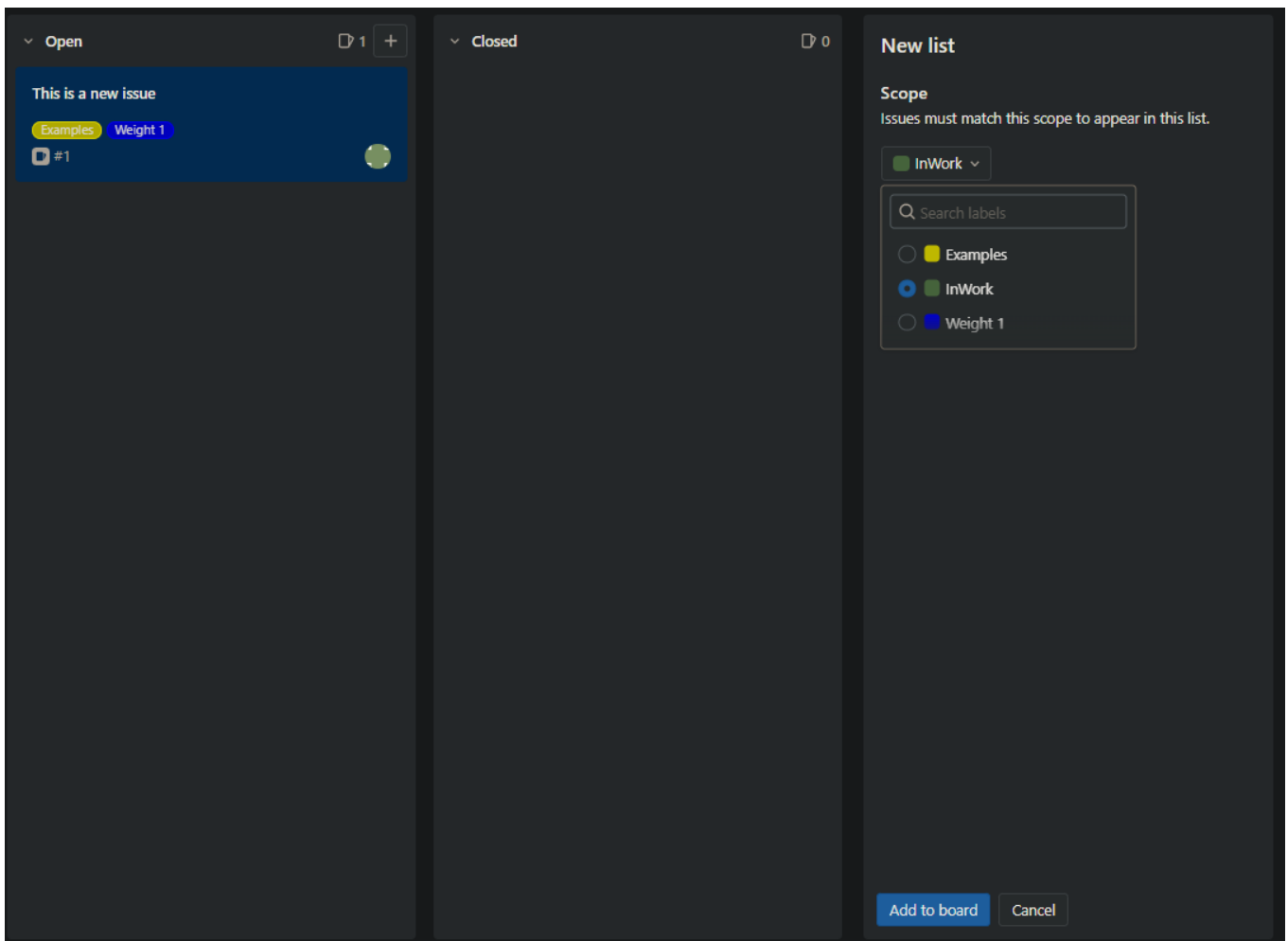
Returning to the original list page you can now see the issue you have created. While the list view gives a good overview, it is difficult to navigate. This is where the boards page comes in.

## Boards

Boards help organize the teams work flow and provides a great visual for both the developers and the project managers. This creates a central place to track the design and implementation of new features. Selecting this page under issues, you can see the issue you just created under open. By default there is only the open and closed column, but these can be expanded to meet your needs. We will walk through creating a in progress column to be able to show what is actively being worked on:

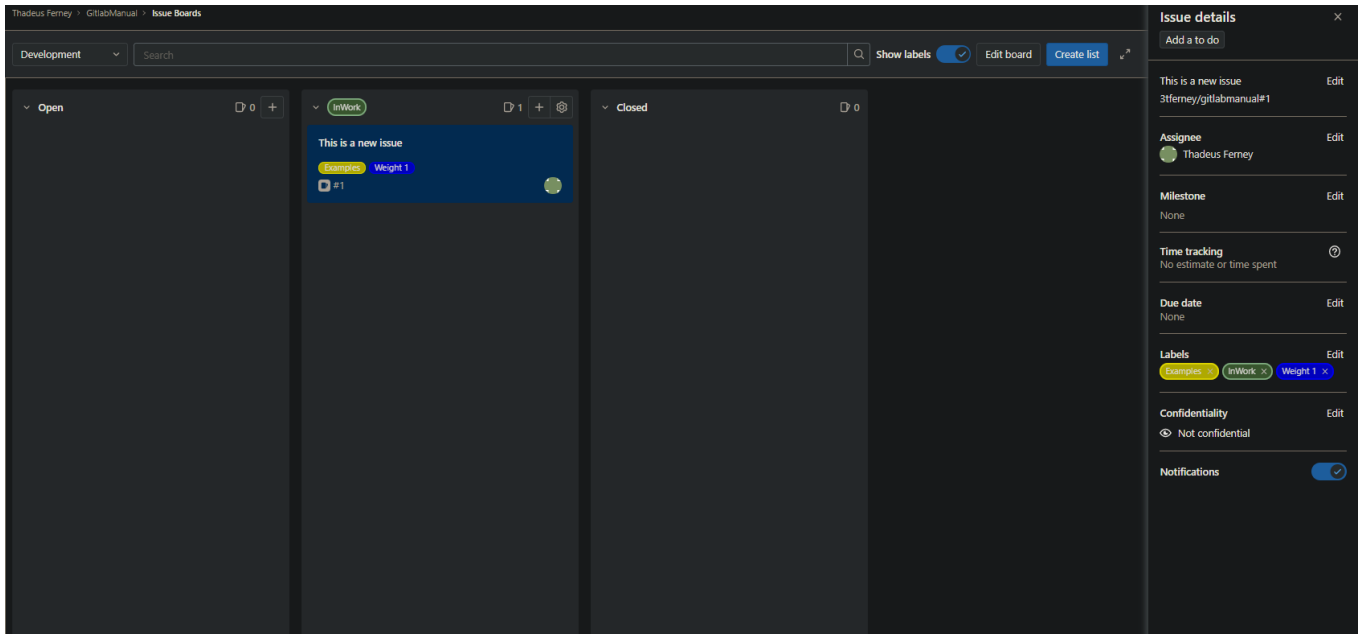


1. First start by selecting Create List and select which tag this list should track. Here we have created a new tag called InWork. This can be done under the issue page mentioned before.





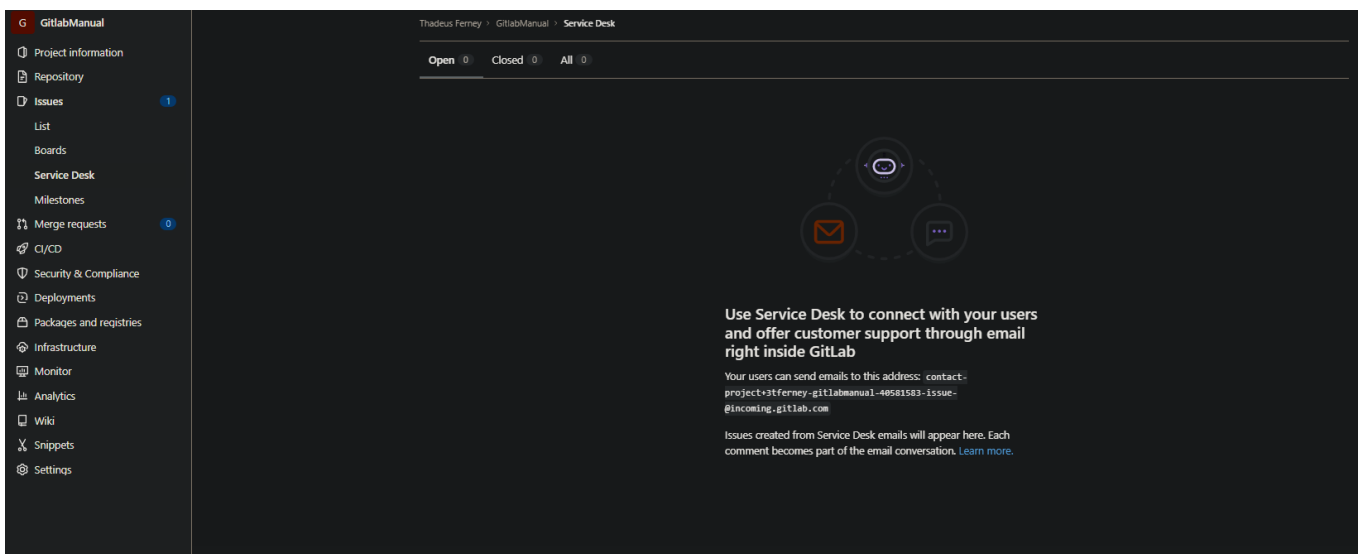
2. After adding this to the board you will now be able to drag and drop issues between lists to organize your work flow and edit the tag automatically.



As you can see this has allowed us to shift each issue individually between steps in the development process. Clicking each issue will open a side bar summary of the issue, allowing for quick edits. This also displays each developer assigned icon by the issue so at a glance it is clear what each person is working on.

## Service Desk

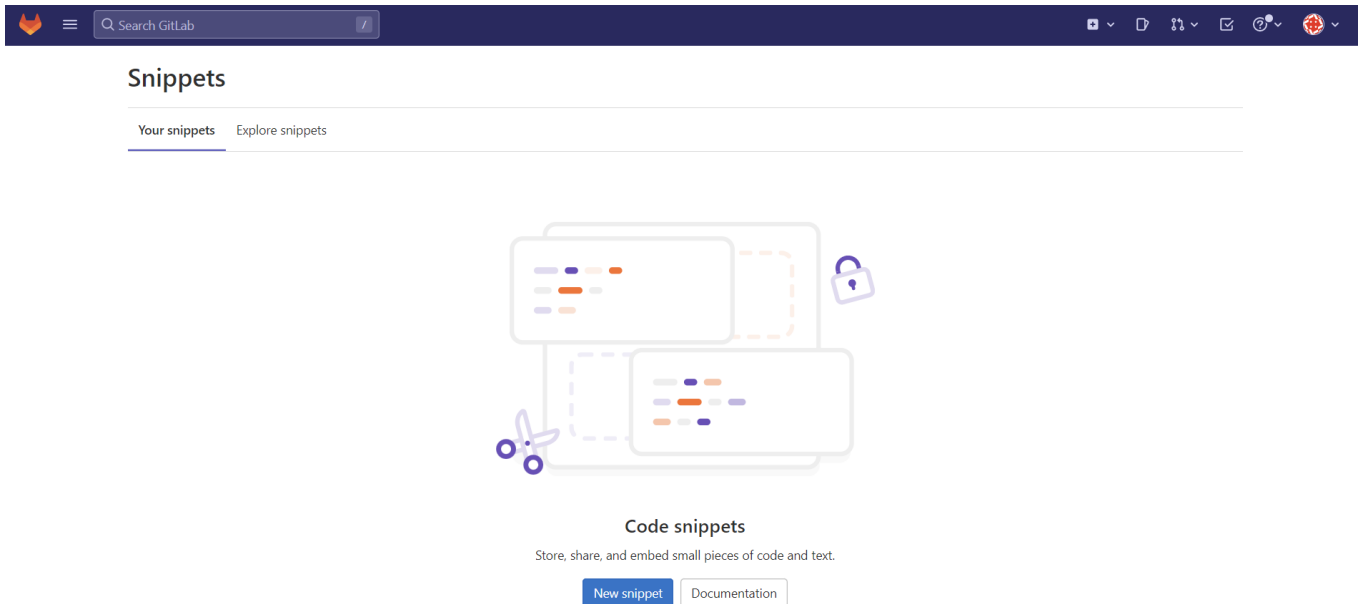
Until now we have only covered how the developers can interact with issues. GitLab provides a way for the end user to send a direct issue to the board via the Service Desk feature. This automatically creates an email the user can submit issues to.



1. Selecting the service desk will provide you with this email. This email can be embedded into a built in help features or provided directly to users. This can be customized to the repository.

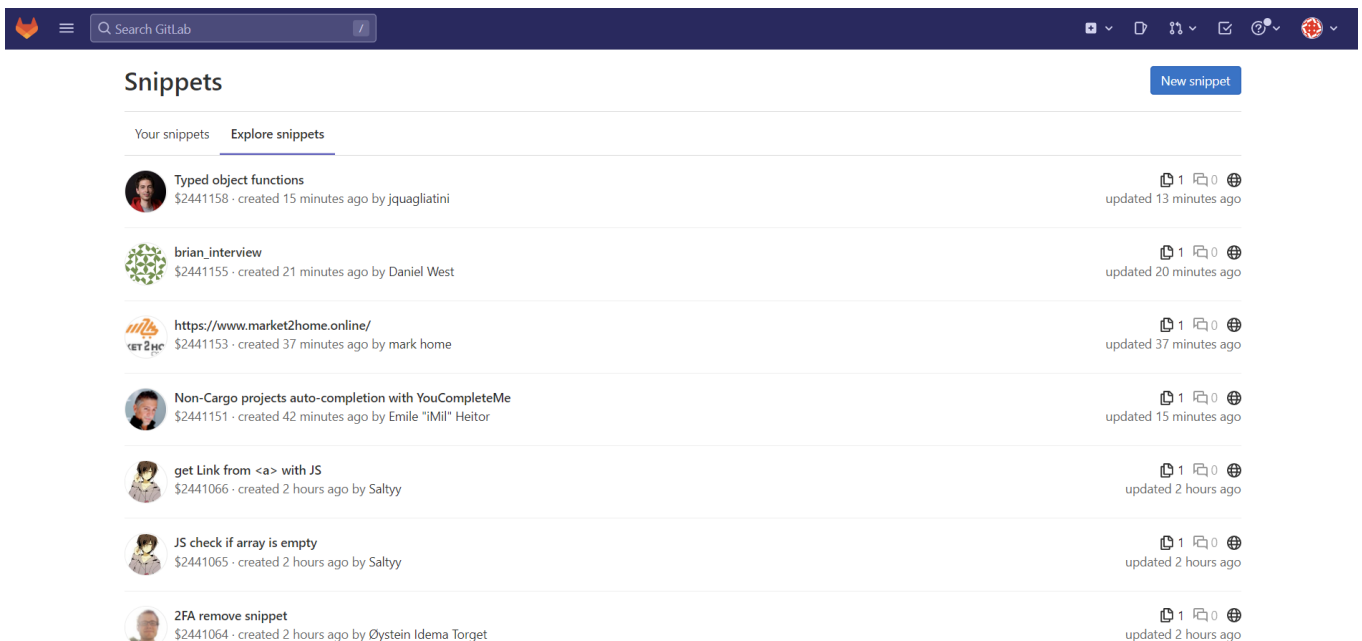
## Snippets

The snippets tab is a place to store your code snippets. Small pieces of code that have it's independent functionality that you can share with the rest of your team.



There are two tabs here, one for your current snippets and the other is for Exploring other snippets.

On the Explore snippets you can see the most recently created public snippets.



To create a snippet all you have to do is hit `New snippet` on the "Your snippets" tab, there is also a button that links to the documentation behind how code snippets work if that is what you are looking for.

## Code snippets

Store, share, and embed small pieces of code and text.

New snippet

Documentation

When you press `New snippet` you it allows for you to enter a Title, Description, File Name, the code you want in that snippet, optional more files, and whether the code snippet is private or public.

# New Snippet

Title

Description (optional)

Optionally add a description about what your snippet does or how to use it...

Files

Give your file a name to add code highlighting, e.g. example.rb for Ruby

Delete file

1

Add another file 1/10

Visibility level 


☒  Private

The snippet is visible only to me.

☐  Public

The snippet can be accessed without any authentication.

The following is what it looks like after you have created a code snippet and the options it gives you to fiddle with it.

Authored just now by  Curtis Liu

Edit



Delete



New snippet

## The Law Of Coding




Edited just now

Clone ▾

 **Let it Be**  292 bytes









  

```
1
2 // You may think you know what the following code does.
3 // But you dont. Trust me.
4 // Fiddle with it, and youll spend many a sleepless
5 // night cursing the moment you thought youd be clever
6 // enough to "optimize" the code below.
7 // Now close this file and go play with something else.
```

 0  0 

Write

Preview

Write a comment or drag your files here...

The Snippets page will have also added that as well and increase tabs for Private, Internal, Public, and All of you code snippets.

## Snippets

New snippet

Your snippets Explore snippets

All 1 Private 1 Internal 0 Public 0

**The Law Of Coding**

\$2441229 · created 1 minute ago by Curtis Liu

 1  0   
updated 1 minute ago