# Git + Sourcetree with STM32 Projects

**Course:** DESN2000 Computer Engineering

**Author:** Riley Haydon

**Date:** T2 2024

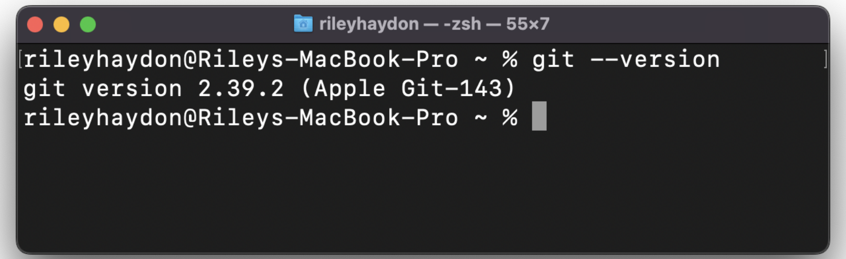
**Version:** 1.0

# 🍺 Installing Git

## Check if you already have it installed

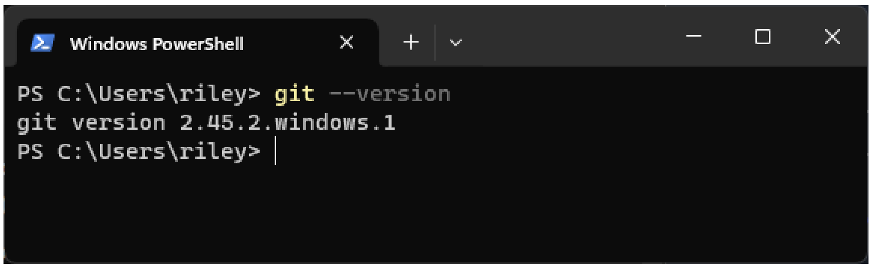
You probably already have this still installed on your system if you have done COMP1531. To check, open a terminal and type `git –version` (two dashes in front of version).

If you have it installed, you should see something like the following. If you don't have it installed, you'll get some error saying that it's not installed or that the command 'git' is unknown.



rileyhaydon — -zsh — 55×7

```
rileyhaydon@Rileys-MacBook-Pro ~ % git --version
git version 2.39.2 (Apple Git-143)
rileyhaydon@Rileys-MacBook-Pro ~ %
```



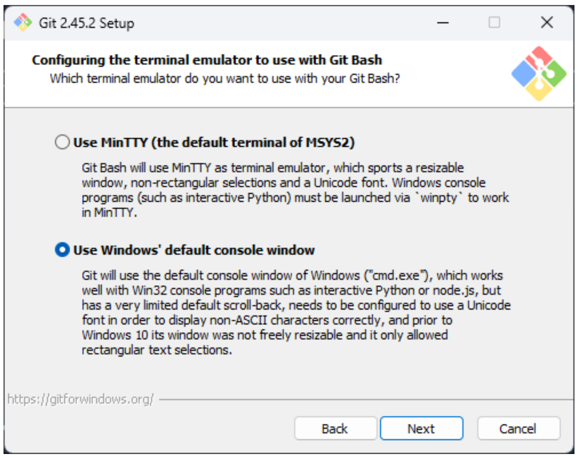
Windows PowerShell

```
PS C:\Users\riley> git --version
git version 2.45.2.windows.1
PS C:\Users\riley>
```

## Installing Git

This is pretty self explanatory, and you can just follow the install wizard. Just some things to note:

- If you're on windows:
  - During the 'select components': ensure "Windows Explorer Integration" is selected.
  - During the 'Configuring the terminal emulator to use with git bash': I suggest using windows default console.
- On the Adjusting your PATH environment, make sure you select the second option 'Git from the command line and also from 3rd-party software'
- Every other page of the installer you shouldn't have to change any settings - unless you know what you're doing.



Git 2.45.2 Setup

**Configuring the terminal emulator to use with Git Bash**

Which terminal emulator do you want to use with your Git Bash?

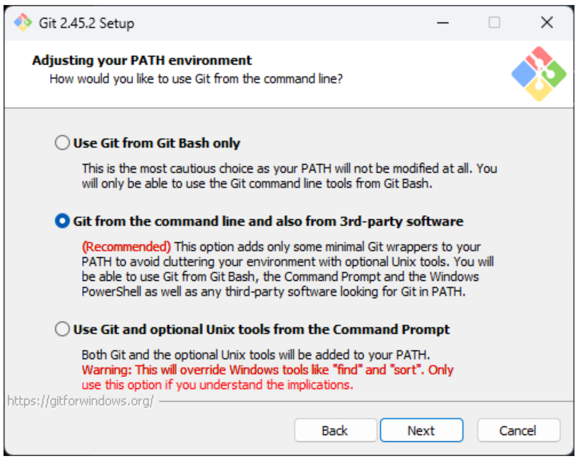
○ **Use MinTTY (the default terminal of MSYS2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

[✓ **Use Windows' default console window**]

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

https://gitforwindows.org/



Git 2.45.2 Setup

**Adjusting your PATH environment**

How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

[✓ **Git from the command line and also from 3rd-party software**]

(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

○ **Use Git and optional Unix tools from the Command Prompt**

Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

https://gitforwindows.org/

You can donwload git from here: https://git-scm.com/

Follow the installation wizard until you see a version number in your terminal similar to above.

Note: you may need to open a new terminal instance if you have installed git and nothing comes up when you type `git —version`.

---

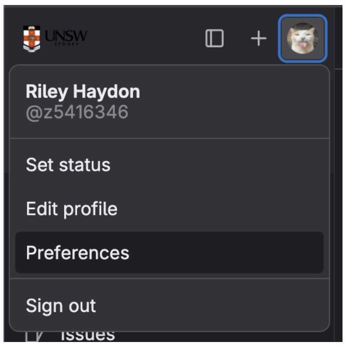
## ⚒️ Setting up a Gitlab Project

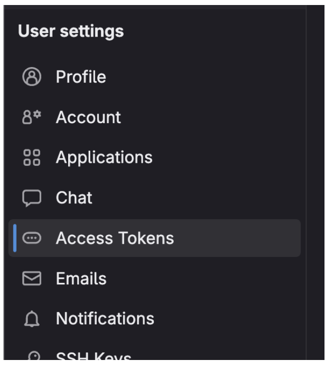
### Creating an Access Token

Since many (if not all) of you will have used gitlab at some point at UNSW, we will be using it to store the repository for project.

Once you're logged into gitlab, navigate to the preferences page by clicking on your profile picture.

On the preferences page, click 'Access Tokens' from the sidebar menu.





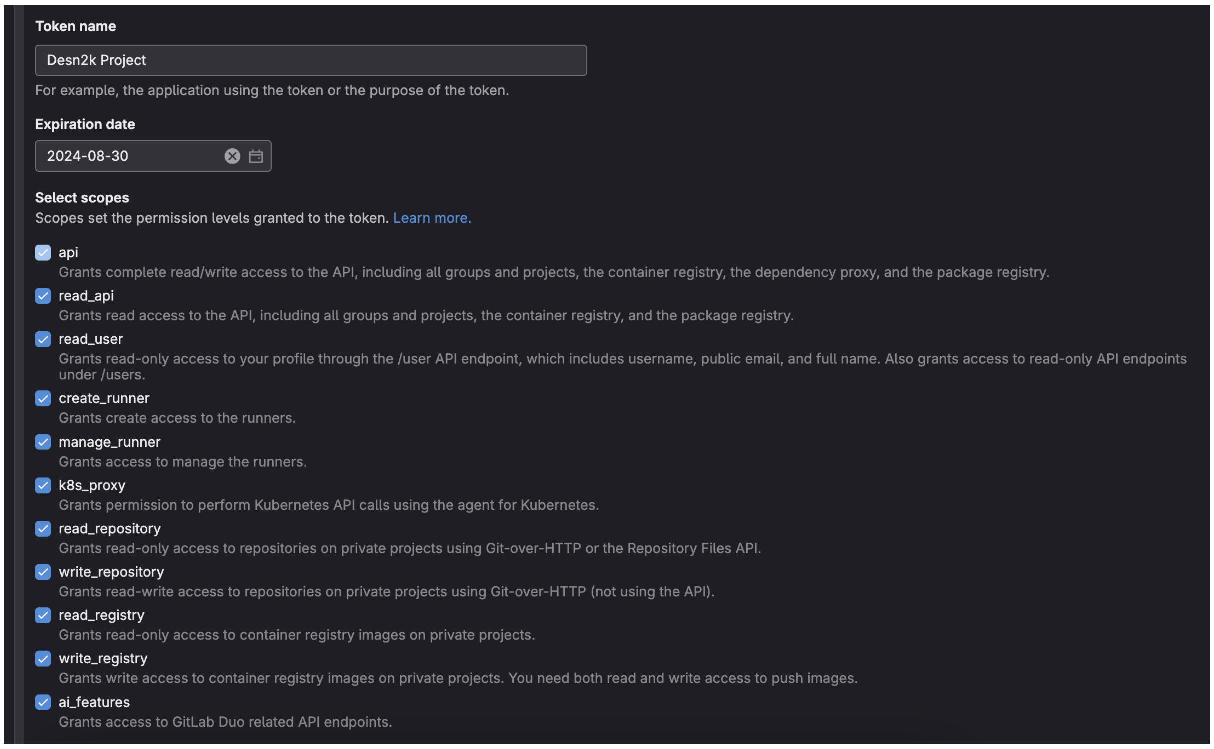
These access tokens will allow us to connect gitlab to our git repo over the HTTP protocol.

(I did try extensively to use the SSH protocol, but couldn't get it to work properly/consistently - I believe due to the way UNSW has setup our gitlab accounts)

💡 Even though your team will only have 1 project/repo, each team member must create their own personal access token.

In the table, click '**Add new Token**'.

- **Name**: This can be whatever you want e.g "Desn2k Project"
- **Expiration date:** I suggest setting it to the end of the current term + 1 week.
  - It doesn't really matter what the date is, but you'll have to make a new token and do this process again if you set it too early.
  - Additionally, you're able to delete any created tokens at any time so setting it a few months makes no difference. Deleting the access token **does not delete any projects** using it. It will just prevent you from pushing your project to gitlab.
- **Scopes**: Select **ALL** of them
  - We only technically need a few of them, but during testing, my ability to push/fetch/pull etc was flakey at best so enabling all of them worked consistently for me.


**Token name**

Desn2k Project

For example, the application using the token or the purpose of the token.

**Expiration date**

2024-08-30

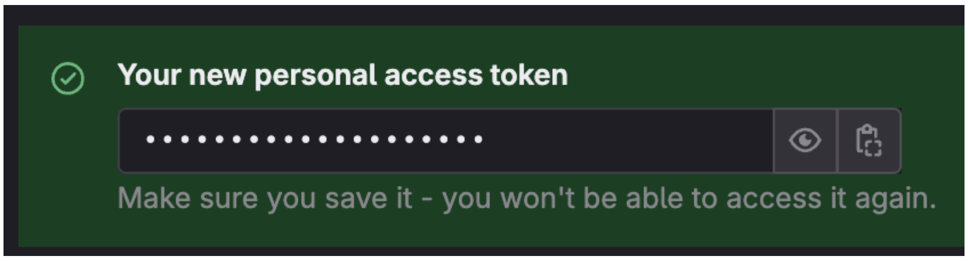
**Select scopes**

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, the dependency proxy, 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 your profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ✓ create_runner
  Grants create access to the runners.
- ✓ manage_runner
  Grants access to manage the runners.
- ✓ k8s_proxy
  Grants permission to perform Kubernetes API calls using the agent for Kubernetes.
- ✓ 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).
- ✓ read_registry
  Grants read-only access to container registry images on private projects.
- ✓ write_registry
  Grants write access to container registry images on private projects. You need both read and write access to push images.
- ✓ ai_features
  Grants access to GitLab Duo related API endpoints.

After you've set those fields, press 'Create Personal Access Token'. You'll then see a green box that contains your new token. This token will not be able to be viewed again even if you return to this page. If you accidentally refresh the page or lose access, you'll have to make a new token.

🚨 You **MUST** copy the generated now to a notepad/notes/text file **now**.

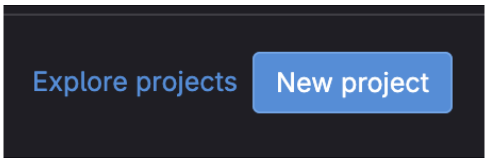

**Your new personal access token**

••••••••••••••••••••

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

## Creating a New Project

Head back to the main gitlab homepage and in the projects menu select ‘New Project’ in the top right.

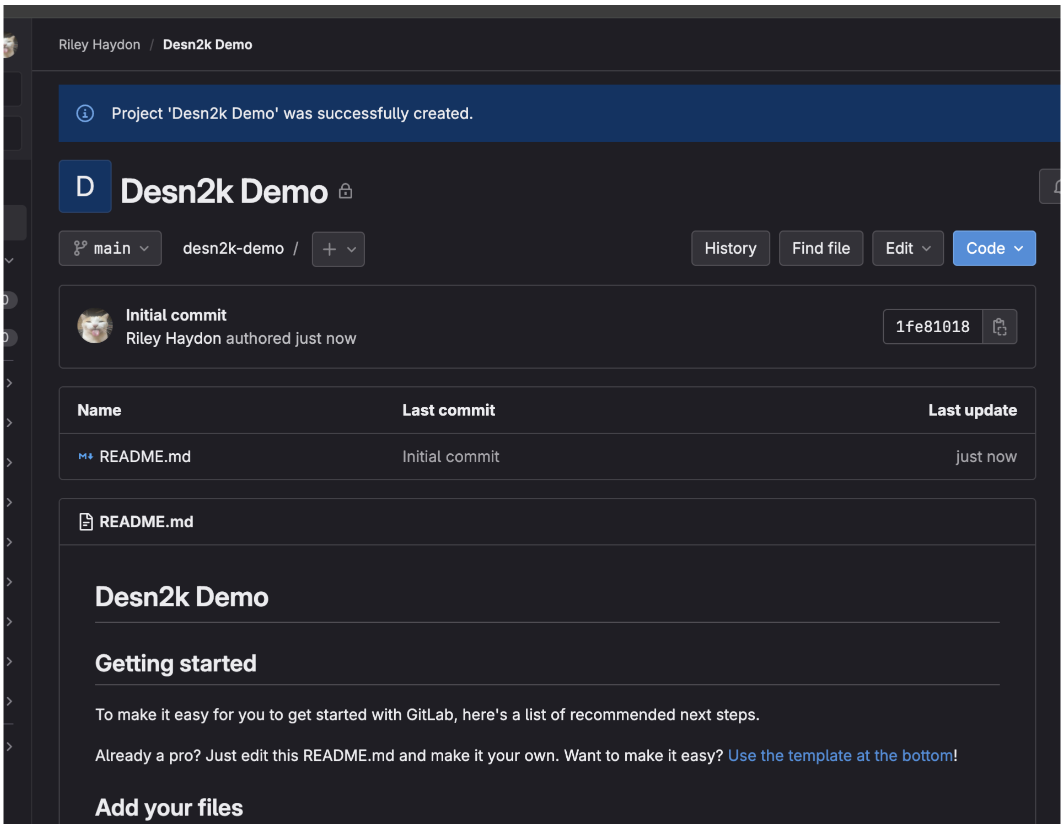


Press ‘Create a blank project’ on the next page.

On the following page, you’ll fill out the following fields:

- **Name**: Whatever you want e.g “Desn2k Project”
- **Project Slug**: Just adds more detail to the project URL and is generated from the project name. (Modify if you wish)
- **Visbility Level**: Private
- **Project Configuration**: Select ‘Initialize repository with a README’. Leave the other field blank.

After pressing ‘Create Project’, you’ll see the main homepage for your project.



Project 'Desn2k Demo' was successfully created.

Desn2k Demo

Initial commit
Riley Haydon authored just now
1fe81018

| Name | Last commit | Last update |
|---|---|---|
| README.md | Initial commit | just now |

README.md

Desn2k Demo

Getting started

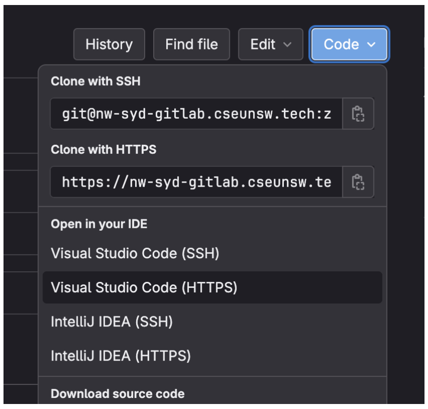
To make it easy for you to get started with GitLab, here's a list of recommended next steps.

Already a pro? Just edit this README.md and make it your own. Want to make it easy? Use the template at the bottom!

Add your files

## Cloning our Gitlab Repository

Click the 'Code' dropdown in the top right and select 'Visual Studio Code (HTTPS)'. You may get some popup asking/allowing gitlab to open Visual Studio Code - Press accept/open.



Once in Visual Studio Code, you'll be prompted to set the destination for the repo locally. This can be wherever you like - chose a good location where you can find it easily and it wont be deleted accidentally e.g Desktop.

Press 'Select as Repository Destination' once you've nagivated to your desired directory.

### Entering Gitlab Credentials

This is the same for both windows/mac, just slightly different interfaces.

**Windows**: Expect to see a popup window appear with inputs. ***Do not use the password tab***.

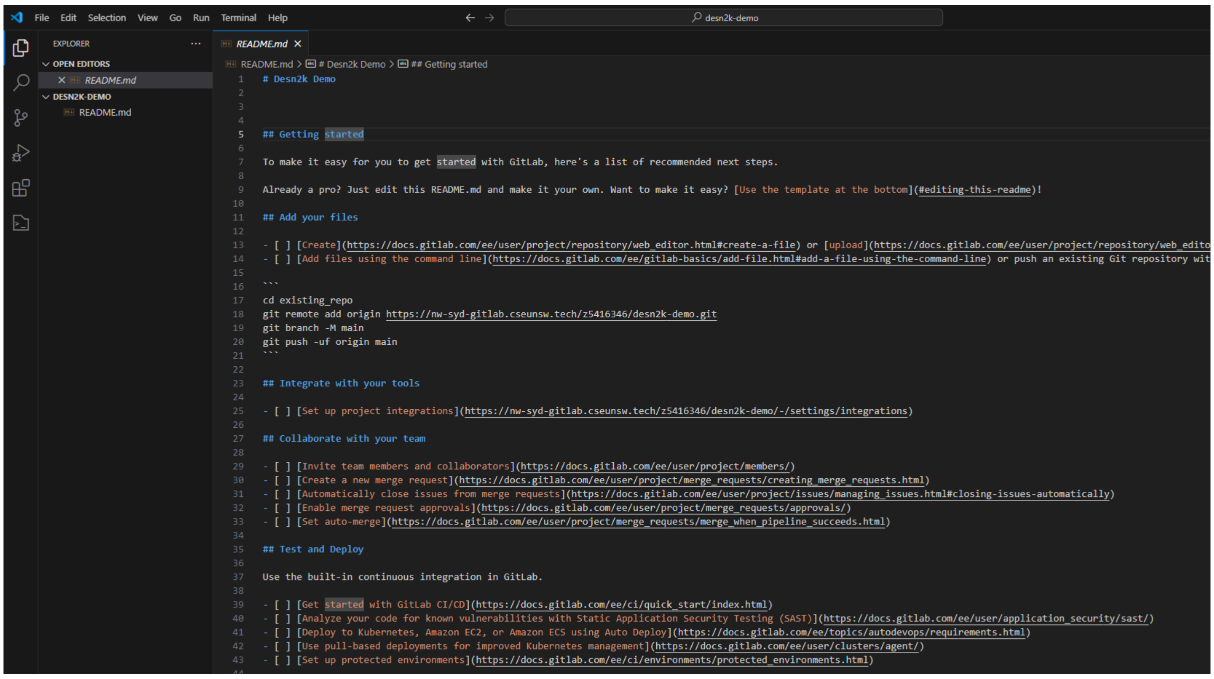
**Mac:** Expect to see a input field at the top of your screen - the same interface if you've used the ssh extension to connect to unsw servers.

**Username:** Enter your gitlab student email e.g 'z1234567@ad.unsw.edu.au'

**Personal access token:** Enter the token we made earlier.

For Mac users, you'll get the user field first, enter it and press enter. You'll then see the 'Password' field, **do not use your gitlab password** - It will not work, enter the personal access token here.

You should now be able to see the `README.md` file thats in our repo.



~~~
# Desn2k Demo



## Getting started

To make it easy for you to get started with GitLab, here's a list of recommended next steps.

Already a pro? Just edit this README.md and make it your own. Want to make it easy? [Use the template at the bottom](#editing-this-readme)!

## Add your files

- [ ] [Create](https://docs.gitlab.com/ee/user/project/repository/web_editor.html#create-a-file) or [upload](https://docs.gitlab.com/ee/user/project/repository/web_edito
- [ ] [Add files using the command line](https://docs.gitlab.com/ee/gitlab-basics/add-file.html#add-a-file-using-the-command-line) or push an existing Git repository wit

```
cd existing_repo
git remote add origin https://nw-syd-gitlab.cseunsw.tech/z5416346/desn2k-demo.git
git branch -M main
git push -uf origin main
```

## Integrate with your tools

- [ ] [Set up project integrations](https://nw-syd-gitlab.cseunsw.tech/z5416346/desn2k-demo/-/settings/integrations)

## Collaborate with your team

- [ ] [Invite team members and collaborators](https://docs.gitlab.com/ee/user/project/members/)
- [ ] [Create a new merge request](https://docs.gitlab.com/ee/user/project/merge_requests/creating_merge_requests.html)
- [ ] [Automatically close issues from merge requests](https://docs.gitlab.com/ee/user/project/issues/managing_issues.html#closing-issues-automatically)
- [ ] [Enable merge request approvals](https://docs.gitlab.com/ee/user/project/merge_requests/approvals/)
- [ ] [Set auto-merge](https://docs.gitlab.com/ee/user/project/merge_requests/merge_when_pipeline_succeeds.html)

## Test and Deploy

Use the built-in continuous integration in GitLab.

- [ ] [Get started with GitLab CI/CD](https://docs.gitlab.com/ee/ci/quick_start/index.html)
- [ ] [Analyze your code for known vulnerabilities with Static Application Security Testing (SAST)](https://docs.gitlab.com/ee/user/application_security/sast/)
- [ ] [Deploy to Kubernetes, Amazon EC2, or Amazon ECS using Auto Deploy](https://docs.gitlab.com/ee/topics/autodevops/requirements.html)
- [ ] [Use pull-based deployments for improved Kubernetes management](https://docs.gitlab.com/ee/user/clusters/agent/)
- [ ] [Set up protected environments](https://docs.gitlab.com/ee/ci/environments/protected_environments.html)
~~~

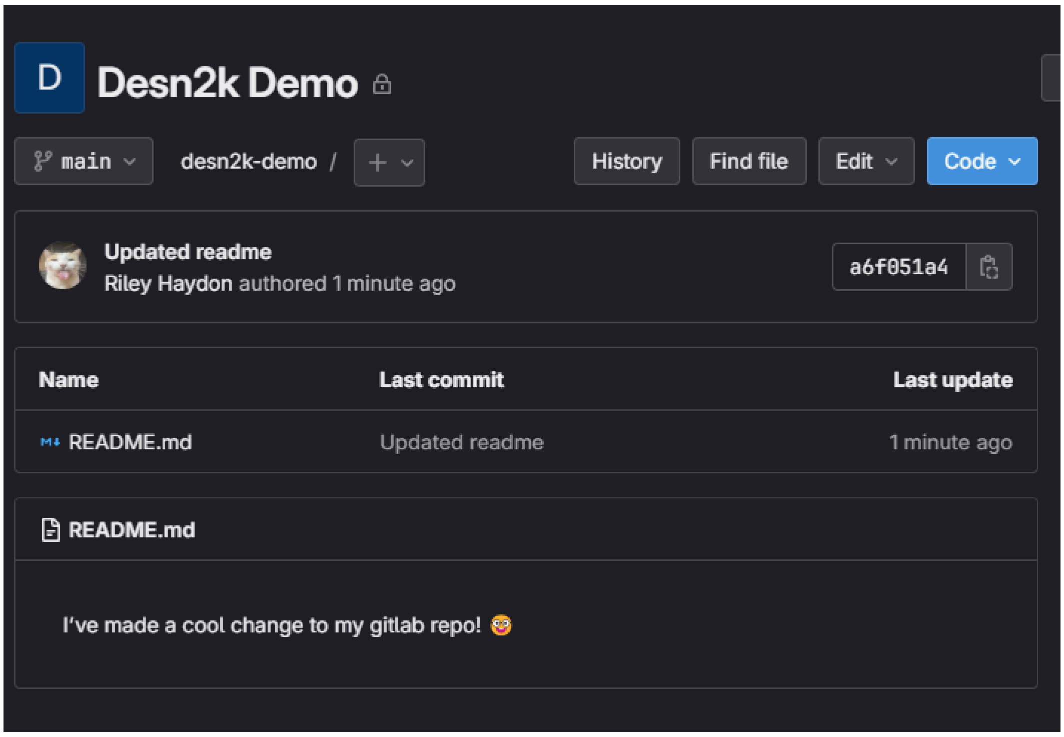
## Pushing and Pulling Changes

To verify that we did everything correctly, we're going to push a change to our repo. In the `README.md` , remove the existing content and put something in the file e.g "I've made a cool change to my gitlab repo! 🤓"

Following the standard git protocol, save your files, make a commit with a message, and push. If you've forgotten the commands, you can use these.

```
git status
git add -all
git commit -m"Updated Readme
git push
```

If you return to your gitlab project page and refresh, you should see our updates to the readme file.



Updated readme
Riley Haydon authored 1 minute ago
a6f051a4

| Name | Last commit | Last update |
|---|---|---|
| README.md | Updated readme | 1 minute ago |

README.md

I've made a cool change to my gitlab repo! 🤓

---

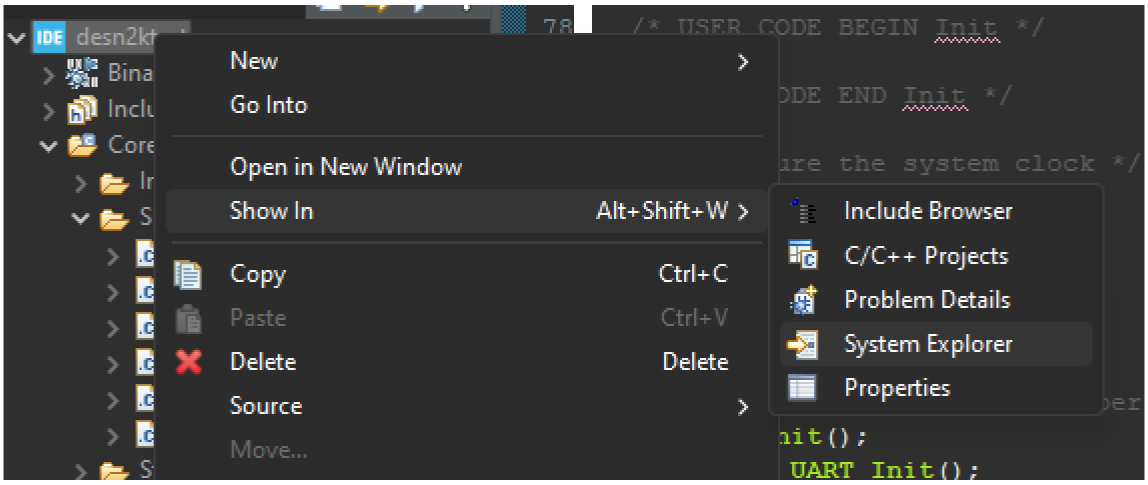
## 📁 Adding Project Files to our Repository

For this, I'll be making a brand new stm32 project using the cubeIDE - you can use your project files if you like. Make sure you do the standard steps for creating a new project like generating the code and pressing build before continuing with this guide. I will be adding some code to make the LED flash.

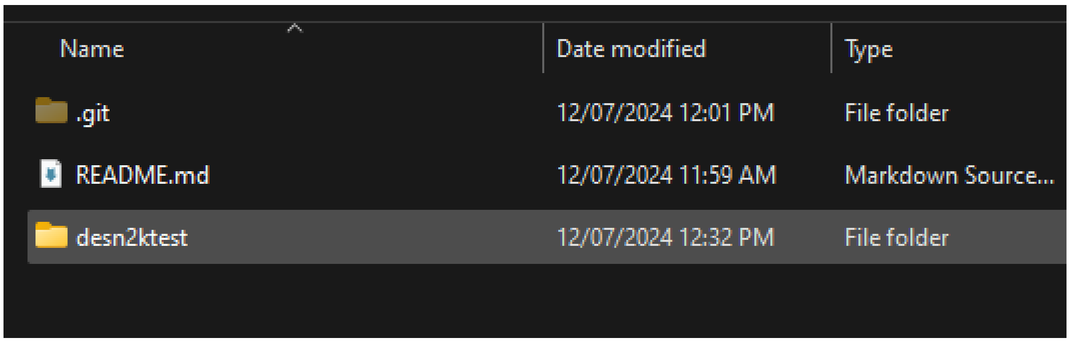
```
while (1)
  {
    /* USER CODE END WHILE */
      HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
      HAL_Delay(500);
    /* USER CODE BEGIN 3 */
  }
```

If you want to use your exist project files, make sure to make a copy of the entire project files first just incase something goes wrong - i.e just duplicate the directory its located in.

Open your project in the STM32CubeIDE, right-click → show-in → system explorer.



This will open up windows explorer/finder, copy the root folder of the project - should be whatever the project name is. Paste that folder in the same directory as we cloned our gitlab repository earlier.



| Name | Date modified | Type |
|---|---|---|
| .git | 12/07/2024 12:01 PM | File folder |
| README.md | 12/07/2024 11:59 AM | Markdown Source... |
| desn2ktest | 12/07/2024 12:32 PM | File folder |

Return to vscode where we edited the README file, you should see the folder we just added; and in the source control icon on the left (looks like the git icon), you should see many files that want to be commited - I have 198 for a blank project.

## Adding .gitignore File

As most of the files that are created by the cubeIDE will be common driver files and executables, we're going to omit them from being sent to the repository - but keep them in our directory so the project can still be built.

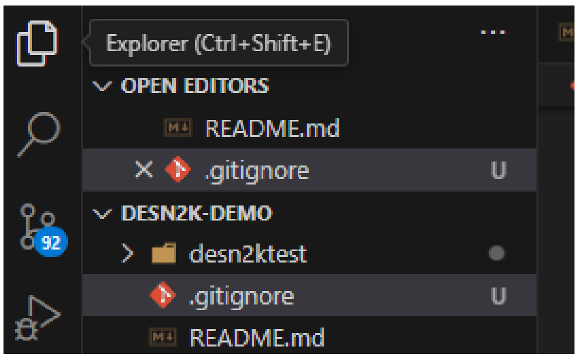
In vscode, and in the root of the project - the same directory as the README, not in the stm files.

Create a new file with the name `.gitignore`. Inside that file, paste the following:

```
#settings
.settings/

#metadata
.metadata/

#Debug
Debug/
debug/
Release/
release

# Prerequisites
*.d

# Object files
*.o
*.ko
*.obj
*.elf

# Linker output
*.ilk
*.map
*.exp

# Libraries
*.lib
*.a
*.la
*.lo

# Shared objects (inc. Windows DLLs)
*.dll
*.so
*.so.*
```

```
*.dylib

# Executables
*.exe
*.out
*.app
*.i*86
*.x86_64
*.hex
*.bin

# Debug files
*.dSYM/
*.su
*.idb
*.pdb
```

Save the .gitignore file. On the source control icon, if you created the file correctly, you should see the number of files to commit significantly reduce. Mine went from 198 to 92.



After verifying the .gitignore file is working, git add, commit, and push to the repo.

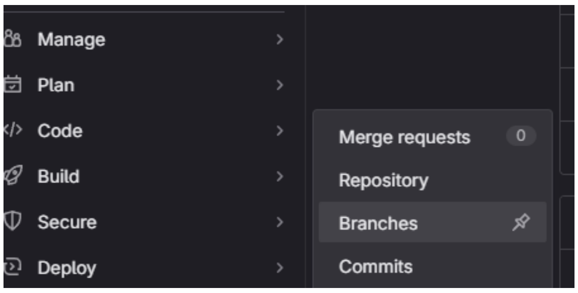
## Making a Working Branch + Verifying that the Project Still Builds

If you dont recall from COMP1531, directly editing on the main/master branch is bad practice. Gitlab by default will actually prevent others you invite from this repo directly pushing to main.

As a reminder, you should always make a merge request on gitlab whenever you want to add code to the master branch. We will create a new branch, make a change, and then make a merge request.

I will be making the branch remotely on gitlab instead of in th terminal to ensure that I am able to fetch from my repository.

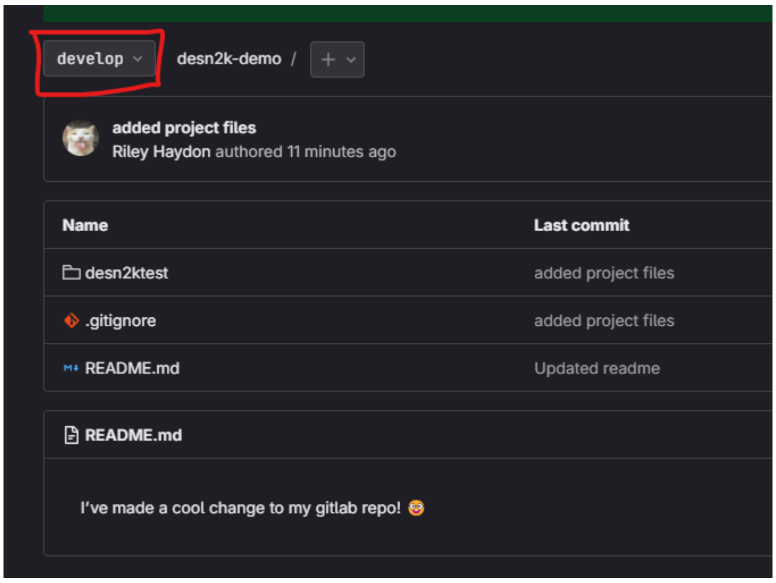
On gitlab, go to Code → Branches.



Click 'New Branch'.

Name the branch 'develop' or something similar for the purpose of this tutorial. Select `main` as the branch to create from.

This will return you to the homepage where you can see that we're now viewing the develop branch - that contains identical code to main.



added project files
Riley Haydon authored 11 minutes ago

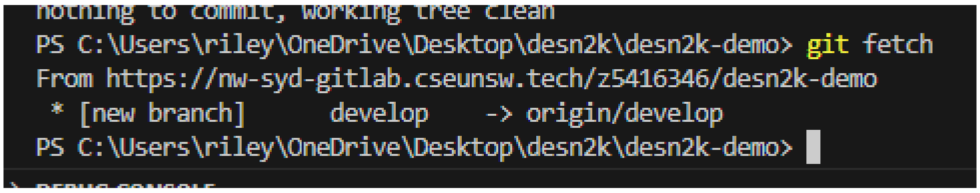
| Name | Last commit |
|---|---|
| desn2ktest | added project files |
| .gitignore | added project files |
| README.md | Updated readme |

README.md

I've made a cool change to my gitlab repo! 🤓

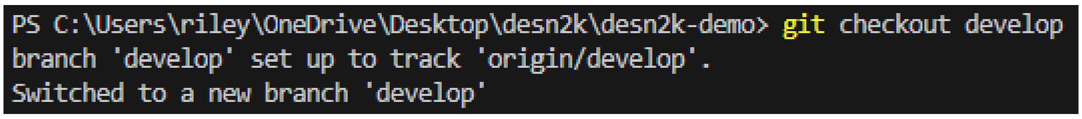
💡 It is important to fetch before making commits to ensure your code/directory is as up to date as possible to avoid later merge conflicts.

Back in vscode, in the terminal, type `git fetch` . You should see the new branch. Fetching doesn't add the changes to our files/branches, to do that type `git pull`.



```
nothing to commit, working tree clean
PS C:\Users\riley\OneDrive\Desktop\desn2k\desn2k-demo> git fetch
From https://nw-syd-gitlab.cseunsw.tech/z5416346/desn2k-demo
 * [new branch]      develop    -> origin/develop
PS C:\Users\riley\OneDrive\Desktop\desn2k\desn2k-demo>
```

Because our local git hasn't seen the develop branch before, we need to `git checkout develop` in order to add it to our index of branches. This will automatically switch us to the develop branch.



```
PS C:\Users\riley\OneDrive\Desktop\desn2k\desn2k-demo> git checkout develop
branch 'develop' set up to track 'origin/develop'.
Switched to a new branch 'develop'
```

## Verifying Project Build

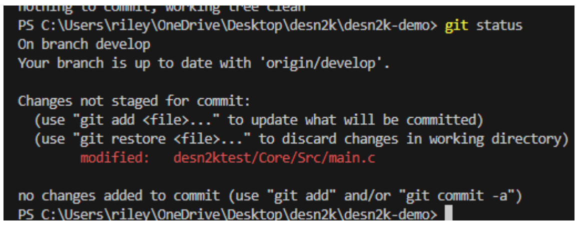
Navigate to the directory of your project, and in the root folder, double click the `.project` file to open it in the STM32CubeIDE.

I will make a change to the program such that the LED will flash every 1 second instead.

```
while (1)
{
  /* USER CODE END WHILE *
    HAL_GPIO_TogglePin(LD2
    HAL_Delay(1000);
  /* USER CODE BEGIN 3 */
}
```

If you imported the project correctly, you should be able to build/run the program as you usually would. 🥳

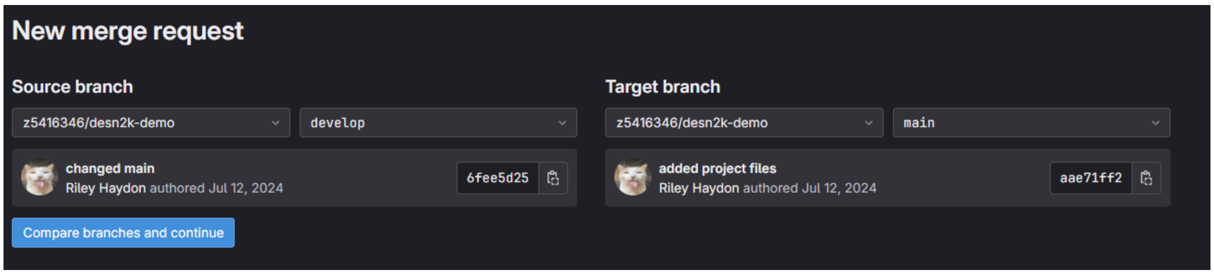
Back in the terminal, running `git status` will let me know of the file I modified.



```
nothing to commit, working tree clean
PS C:\Users\riley\OneDrive\Desktop\desn2k\desn2k-demo> git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   desn2ktest/Core/Src/main.c

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\riley\OneDrive\Desktop\desn2k\desn2k-demo>
```

Like before, I will git add, commit, push my changes to the develop branch.

On gitlab, head to Code → merge requests. Then 'New merge request'. On the next page, select `develop` as the **source branch**, and `main` as the **target branch**.


New merge request

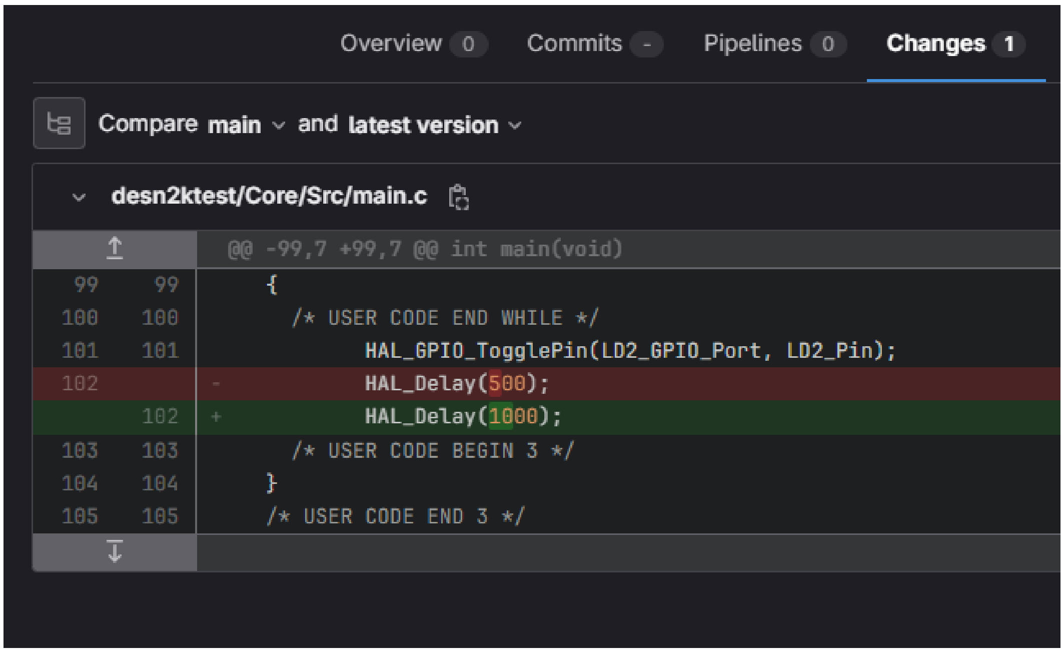
Source branch: z5416346/desn2k-demo · develop
changed main — Riley Haydon authored Jul 12, 2024 — 6fee5d25

Target branch: z5416346/desn2k-demo · main
added project files — Riley Haydon authored Jul 12, 2024 — aae71ff2

On the next page, you can assign the merge request to a member, add a description, title etc. These can also be changed later.

In the merge options section, there is an option to delete the source branch. Generally, I dont like to delete branches as I reuse them to develop new features after they've been merged to main, and there is no consequence to having 'stale' branches. This is circumstantial, so you can proceed as you wish.

Once you're happy with your merge request, press 'Create merge request'. On the next page, you'll be able to approve, merge and manage the request. **It is always a good idea to get another member to approve your request before you merge.** In the 'Changes' tab, you'll see the file we modified.



desn2ktest/Core/Src/main.c

```
@@ -99,7 +99,7 @@ int main(void)
  {
    /* USER CODE END WHILE */
        HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
-       HAL_Delay(500);
+       HAL_Delay(1000);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

Once you've had a poke around the merge request, press 'Merge'. Back on the repository home page, and on the main branch, you'll see that our work was successfully merged to main!

This is all you need to need at a basic level, below is a quick guide for setting up Sourcetree with your repo. It doesn't take very long, and makes doing all the basic git stuff **significantly easier, simplier and more managable**.

---

## 🌲Installing Sourcetree (A git GUI)

> 💡 This step is optional if you're **quite comfortable** using git through terminal commands. However, learning how to use sourcetree is very easy and makes managing a git repo **significantly easier** - it is also very useful for other subjects like COMP2511.

> ⚠ Sourcetree is only available for MacOS and Windows

### Install the sourcetree app

Download the latest version of sourcetree.

Download: https://www.sourcetreeapp.com/

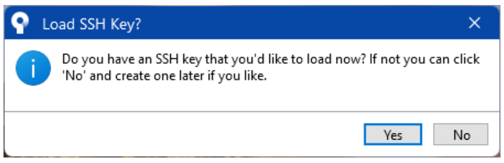
After Opening, proceed with the following options on the installer:

- **Registration**: Skip
- **Install Tools:** Make sure Git and Mercurial are both enabled/configured/preinstalled. You can leave any option in the 'Advanded Options' section blank.
- **Preferences:** These are just the default name/email that will be attached to your commits if you dont set up a custom one for the project. These can

also be changed later.

- **Author Name:** Just your name is fine e.g 'Riley H'
- **Author Email Address:** Probably best to use your unsw email, personal one is also fine.

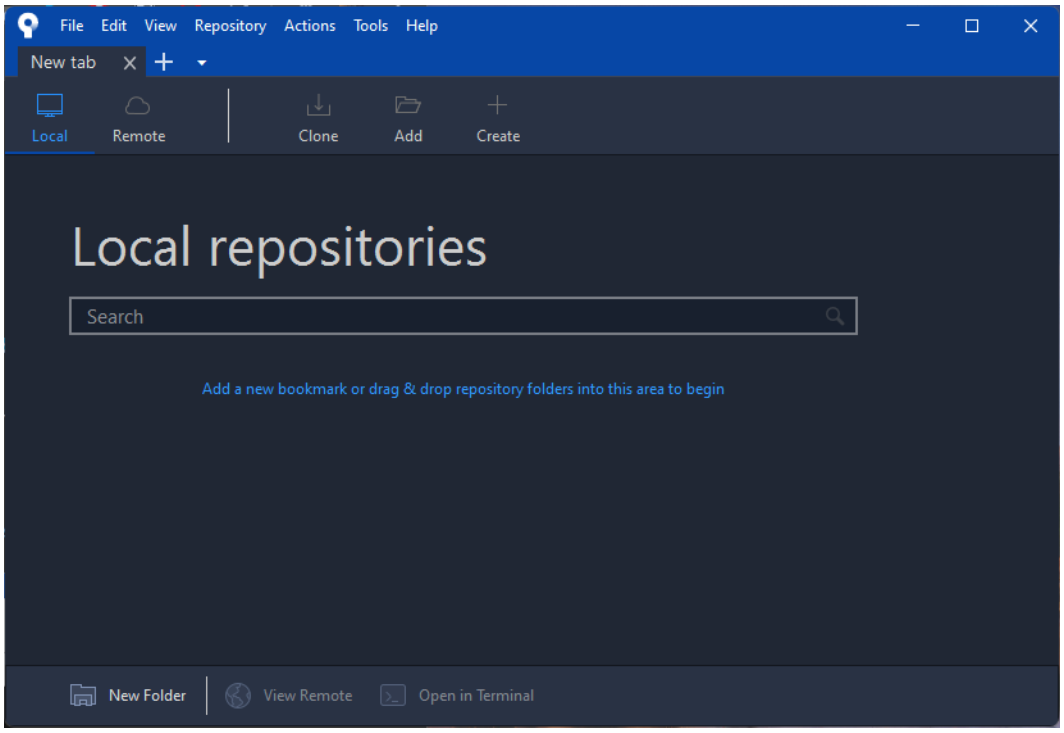
After pressing next, you might be prompted to 'Load SSH Key', select **No**.



Do you have an SSH key that you'd like to load now? If not you can click 'No' and create one later if you like.

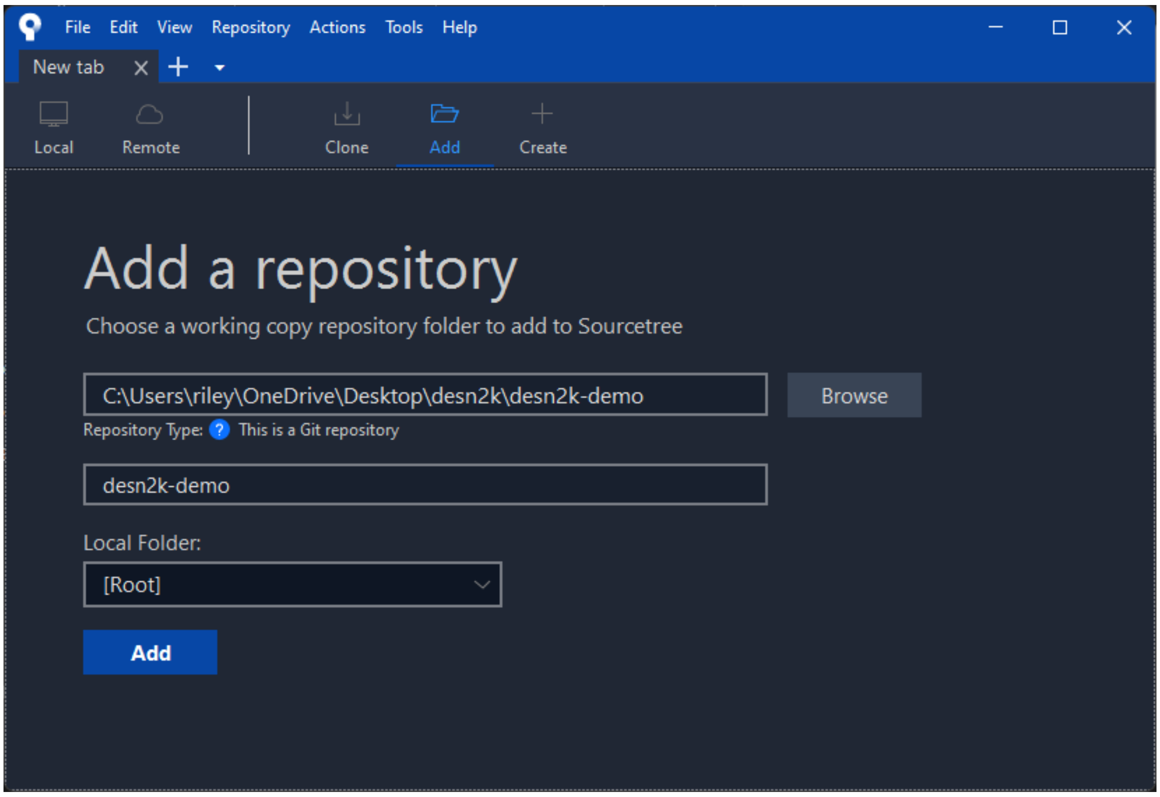
## Adding Our Existing Repository

Once installed, you should be greeted with an app that looks like the following.



Add a new bookmark or drag & drop repository folders into this area to begin

Click 'Add', and for the directory, add the root of the project folder we created earlier - i.e the folder that contains the readme and .gitignore files. You can name the repo here if you like - this wont affect the project name on gitlab, just the name you see on sourcetree. When you're happy, press 'Add'.



Add a repository

Choose a working copy repository folder to add to Sourcetree

C:\Users\riley\OneDrive\Desktop\desn2k\desn2k-demo

Browse

Repository Type: This is a Git repository

desn2k-demo

Local Folder:

[Root]

Add

You'll then be taken to the repository screen and can see all sorts of details about the commits, branches and files associated with your repo. The first thing you'll see is all the commits and the merge we made to the `main` branch earlier - as well as the changes made, author and date.

You'll also see a visualisation of how and when the branches we're split from the trunk (main).

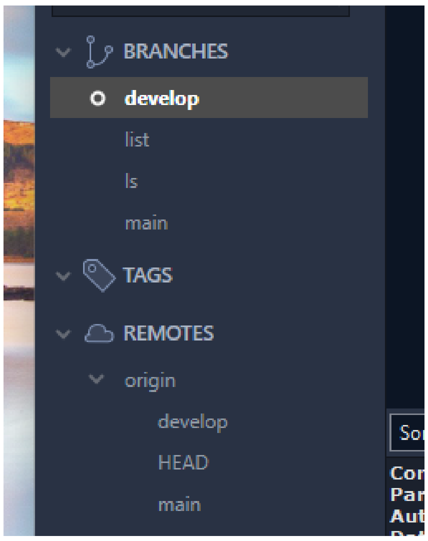
Drawing your attention to the left hand column, you'll see the 'Branches' and 'Remotes' dropdowns.

BRANCHES:

- These are the branches that you've checked out **locally**
- The white dot and name indicates the branch you're currently on. To change branches, simply double click another branch in the list.

REMOTES:

- These are branches that exist **remotely**
- Instead of doing git checkout, simply right-click a branch in remotes and press 'checkout'



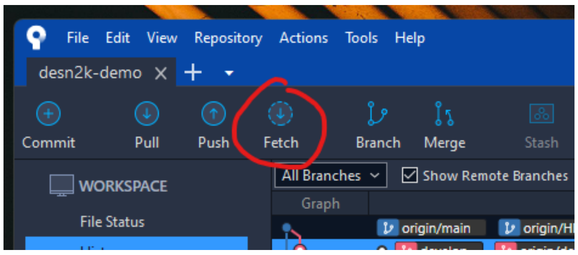
## Sourcetree Example

We're going to do a quick example of using sourcetree to fetch, pull, commit and push changes to our repo - in a fraction of the time of the terminal. I'll add the equivalent commands along the way.

⚠ Although sourcetree does have a 'Merge' button, you shouldn't use it to merge branches, make a merge request on gitlab.

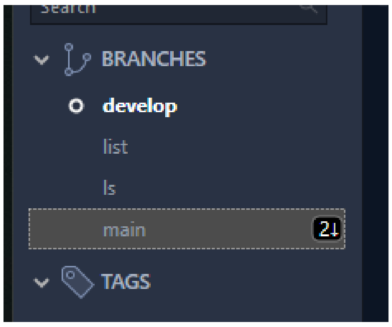
Press 'Fetch' at the top. You dont need to select/diselect any fields that popup, just press 'OK'.

(`git fetch`)



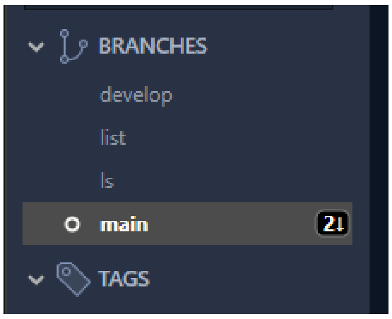
Observe that the main branch is indicating that it has changes that exist remotely, but not locally.

Switch to the main branch by double clicking it.



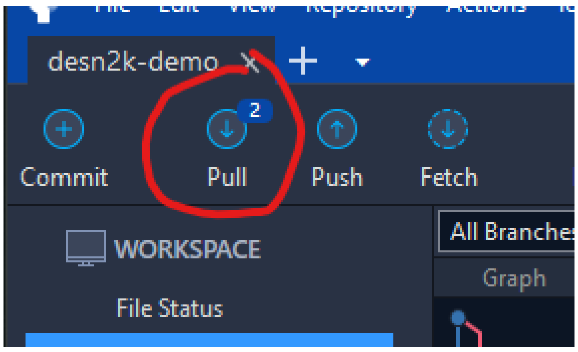
Pull the changes into the branch by pressing ‘Pull’ at the top. Dont change anything in the popup that appears, just press ‘Pull’

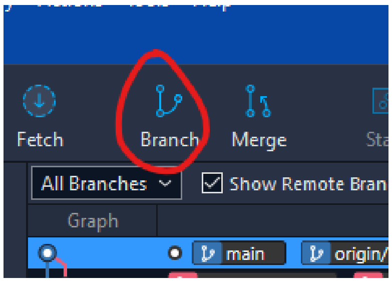
(`git pull`)



Create a new branch from main by ensuring you’re currently on the main branch, and pressing ‘Branch’ at the top.

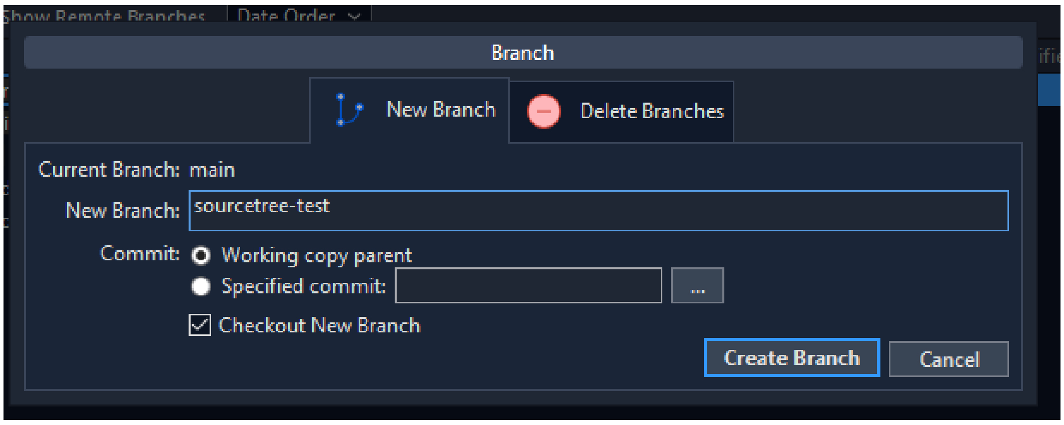
(`git branch`)





Name the branch whatever you want and make sure ‘Checkout New Branch is selected’. Press ‘Create Branch’.

To make gitlab aware of our new branch, we need to push it to the remote.



Branch

Current Branch: main

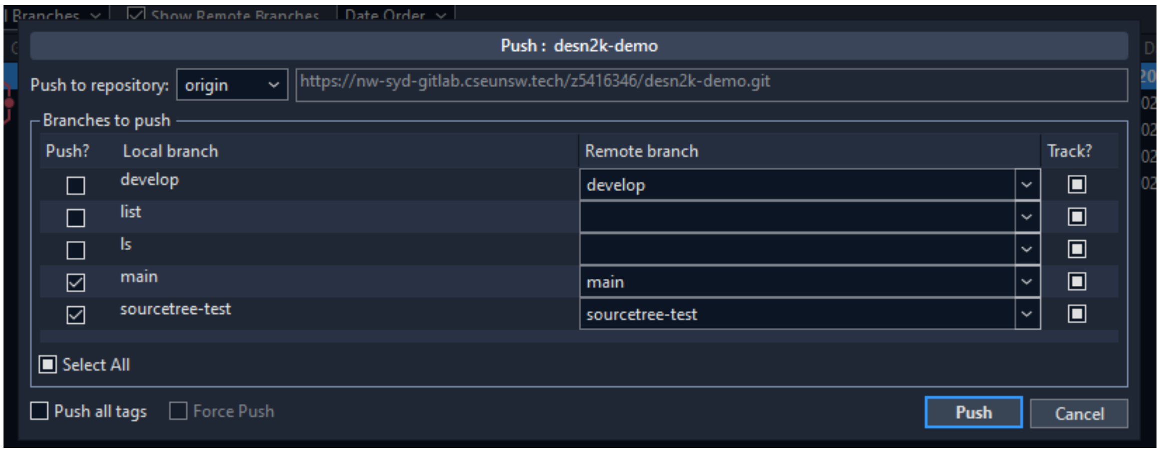
New Branch: sourcetree-test

Commit: [✓ Working copy parent]
Specified commit:

☑ Checkout New Branch

Create Branch | Cancel

Press 'Push' in the top. In the popup, make sure that your new branch is slected on the left ,and that the 'track' column on the right is solid. You should see your new branch on gitlab.



Push : desn2k-demo

Push to repository: origin https://nw-syd-gitlab.cseunsw.tech/z5416346/desn2k-demo.git

Branches to push

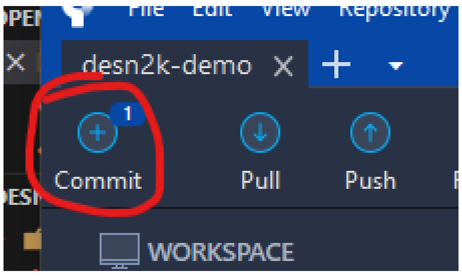
| Push? | Local branch | Remote branch | Track? |
|---|---|---|---|
| ☐ | develop | develop | ■ |
| ☐ | list | | ■ |
| ☐ | ls | | ■ |
| ☑ | main | main | ■ |
| ☑ | sourcetree-test | sourcetree-test | ■ |

■ Select All

☐ Push all tags ☐ Force Push

Push | Cancel

Make some changes to a file in your project or just the readme file if you want. Press 'Commit' in the top menu. (It may have already seen your changes and the number indicates how many files were changed).

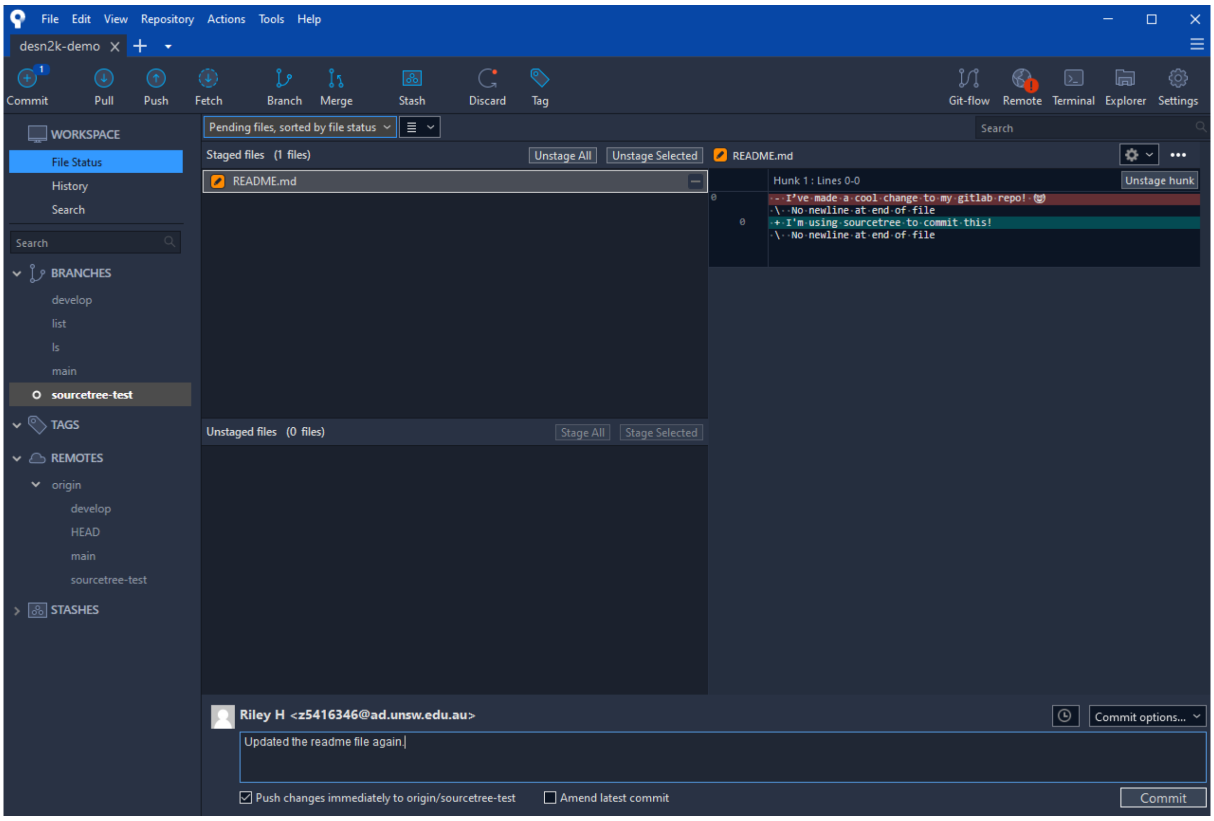


On the commit page, you'll see the files that changed. Press 'Stage All'. (`git add –all`).

In the box at the bottom, write a commit message. (`git commit`)

Below that, select the option that says 'Push Changes immediately to ….' (`git push`)

Press 'Commit' in the bottom right.



Staged files (1 files): README.md

Hunk 1 : Lines 0-0

```
- I’ve made a cool change to my gitlab repo! 😺
\  No newline at end of file
+ I'm using sourcetree to commit this!
\  No newline at end of file
```

Riley H <z5416346@ad.unsw.edu.au>

Updated the readme file again.

[✓] Push changes immediately to origin/sourcetree-test
[ ] Amend latest commit

🎉 That's it! All your changes are in both your local directory and in the remote repository.