Unity + GIT

WORKSHEETS 2024

ADAPTED BY IAN SMITH FROM ORIGINAL RESEARCH BY DR. GLENN JENKINS

Problem: How do I use GIT?

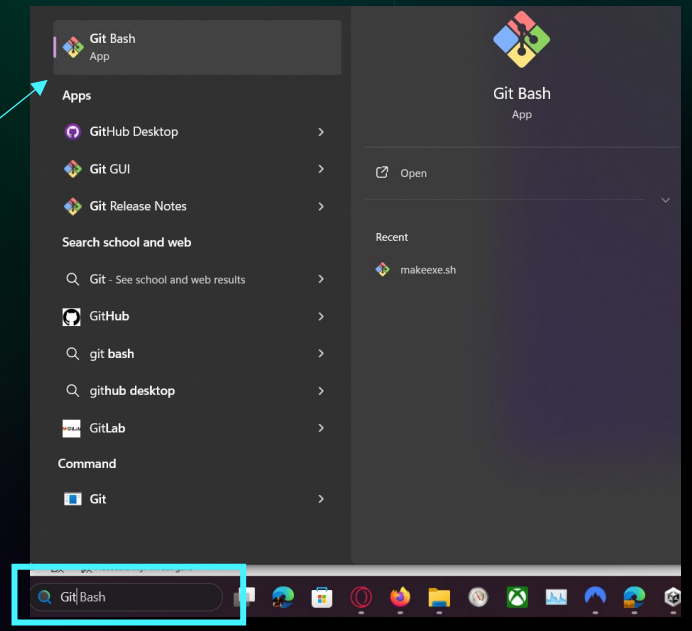
- *Hint: Read through this in full before attempting any steps. Don't try to do it on-the-fly.*

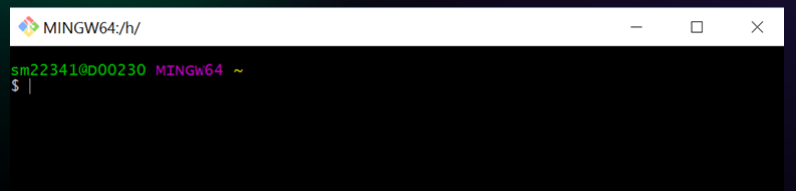
Git Bash – GIT does have GUIs available (GIT Hub, GitKraken, etc.), but it can be beneficial to understand what it is doing under-the-hood, hence why we will be using the Command Line-based interface.

The PCs in the Games Lab and the Learning Lab have Git Bash installed on them. This is essentially a text-based interface that communicates with the OS. For anyone familiar with using the Command Prompt/DOS Prompt on Windows, the logic is the same. You are accessing folders and files on the hard drive through text commands, rather than clicking icons and navigating tabs.

Git Bash can be found by typing into the Windows Search Bar – It should come up at the top.

Double-click to open.



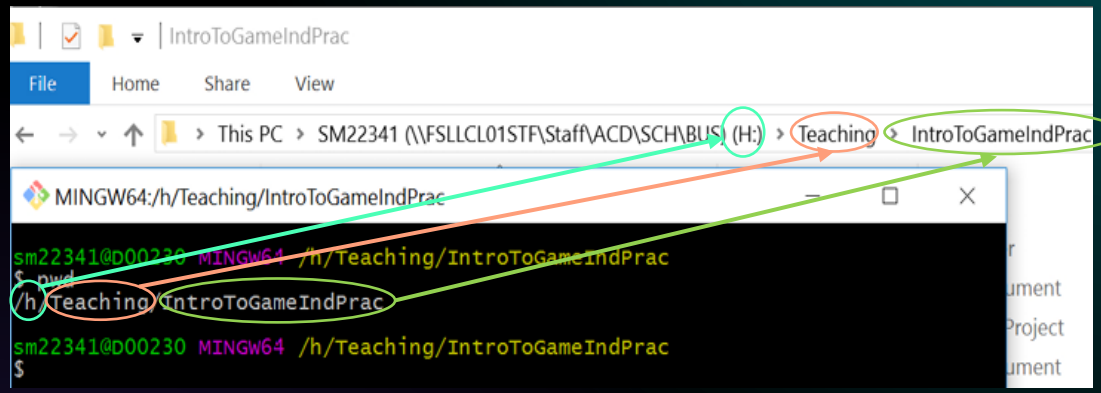


You should see a window like this.

Print Working Directory (pwd)

Our first command is going to be one that tells us where in the directory structure of the system we are. In Bash this is pwd or Print Working Directory.

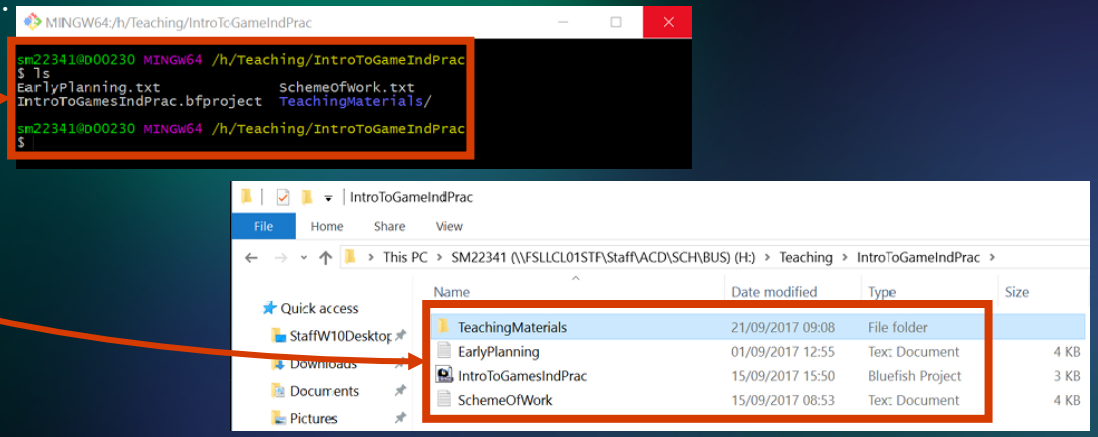
The screenshot below is a comparison of the directory location in Windows File Explorer (Windows 10) and the same location in Git Bash.



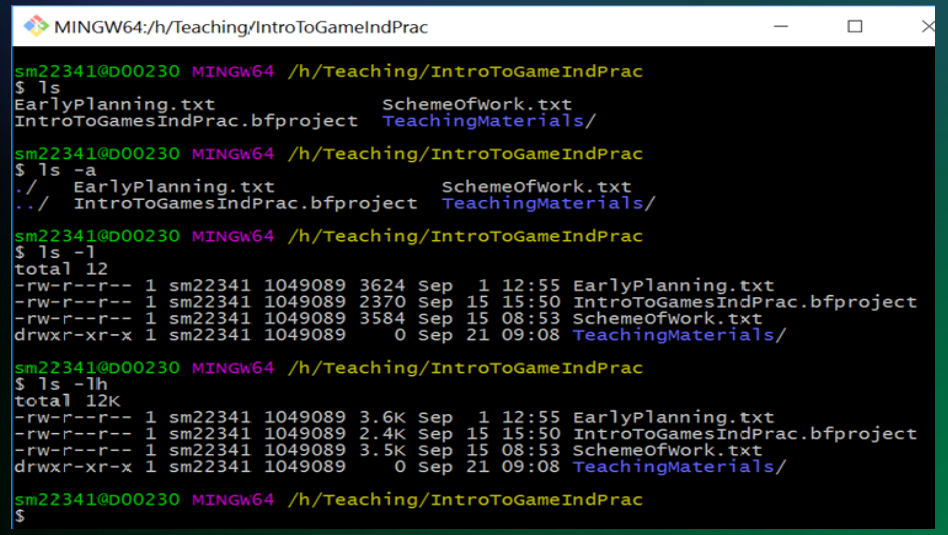
List (ls)

The List command (ls) tells us which files and folders are present in the directory. We have a huge number of flags (options) we can pass to ls – we can instruct it to point at a different directory, operate on the current directory, and to display more or less information using flags.

As before, the screenshot is a comparison between Windows File Explorer and Git Bash.



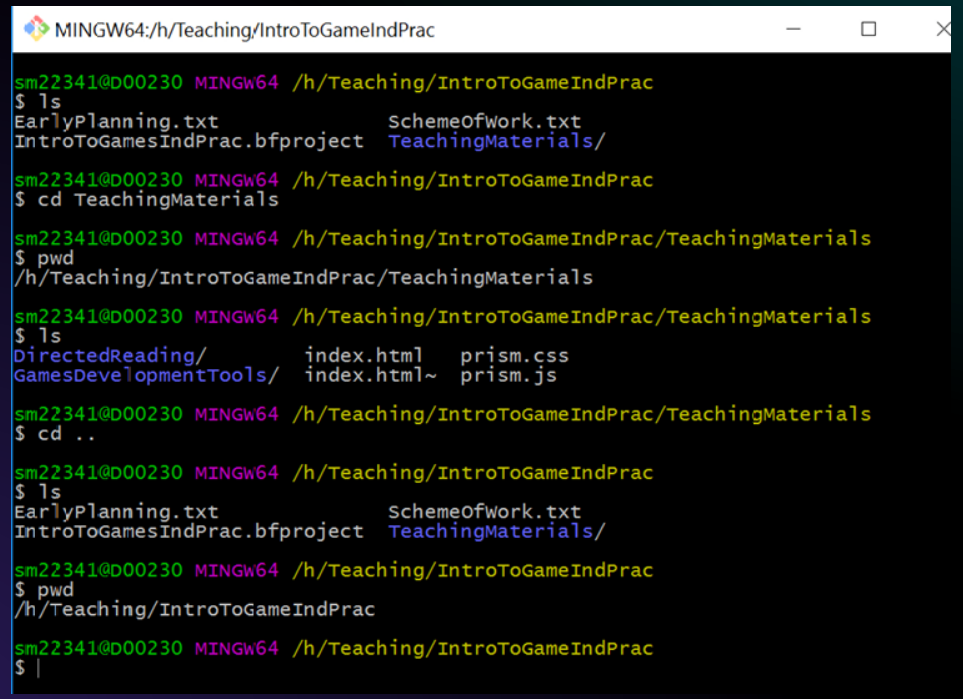
Command	Description
ls	Plain old ls, gives the names of files and folders (but not which are which).
ls <directory>	List the content of a given directory.
ls -a	All, includes .. (one directory down) and files which start with . (hidden files in Unix).
ls -l	Long format, includes information on owner, group and access rights.
ls -h	Human readable format, shows files sizes in GB/MB/KB rather than bytes.
ls -alh	Flag combinations are fine, I tend to use this one.



Change Working Directory (cd)

Our last command is Change Working Directory, which, you guessed it – changes the working directory. This allows us to move through the hierarchy of the file structure. In the screenshot below, pwd has been used to print the current directory, and ls to list its contents.

Note: 'cd .. ' takes us down a directory. So, cd ../ ../ would take us down two directories.



GIT

This isn't going to cover everything GIT-related, but it should be enough to get you started putting a one-developer project on Bitbucket. We will go as far as putting a file into a repository.

Using GIT specifically with Unity will be in its own section.

We will start by reviewing the GIT work-flow and the commands we will use, and then we'll head to Bitbucket to set up an account.

Workflow Overview

Before we dive into this, let's take a moment to look at what we're doing and how.

1. Create a new repository on Bitbucket (this is the remote or central repo).
2. Create a copy (or clone) of that repository on our local machine.
3. Create or copy files into the directory and make a change.
4. Add them to the staging area
5. Commit them (store the changes) with a comment in the local repository
6. Push the change to the remote repository.

At some point in the future we may return to this local repository, having done work (cloned, committed, and pushed) on the project elsewhere.

GIT

Workflow Overview (Cont.)

At some point in the future we may return to this local repository, having done work (cloned, committed, and pushed) on the project elsewhere. In this case we do the following:

1. Pull from the origin (the remote repo), merging changes into this copy.
2. Edit files, write new code, create new files in the directory.
3. Add them to the staging area.
4. Commit them (store the changes) with a comment in the local repository. This can be done multiple times.
5. Push the change to the remote repository.

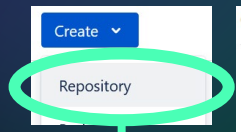
Setting up a Bitbucket Repository

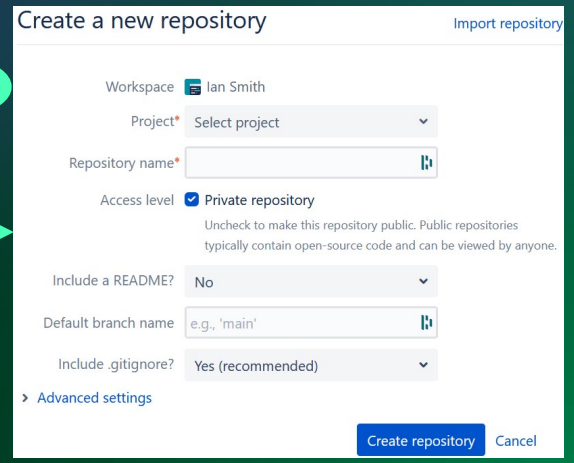
Prerequisites:

- Install GIT (already on the Games Lab & Learning Lab PCs)
- Sign up for a Bitbucket account - https://bitbucket.org/ This should work with your Cardiff Met email.

Step 1: Create the Repository

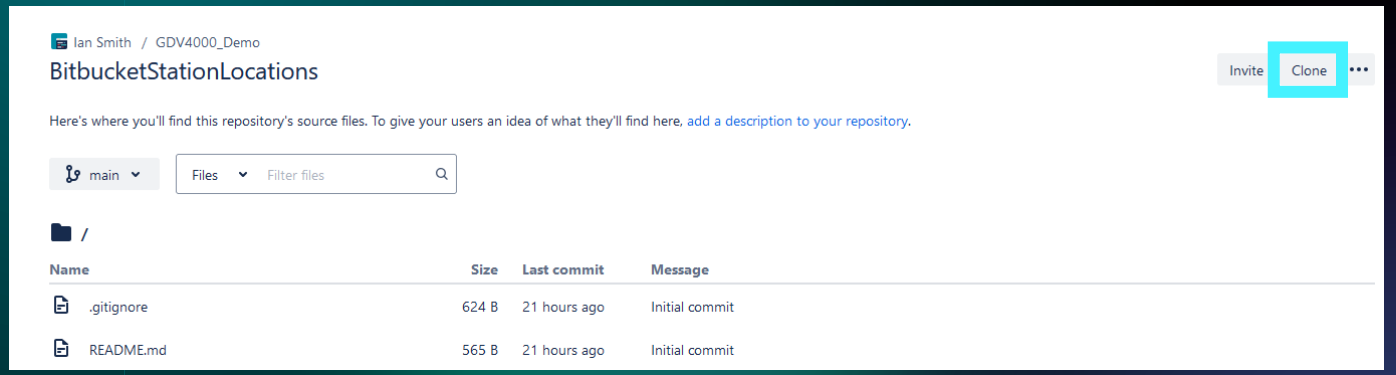
- On Bitbucket, select the Create button and select Repository
- Most of the fields on the Create Repository page can be altered later on.
- On the Project drop-down menu, select 'create new project' and give your Project a name. The project is the container for your Repos for this particular project. For instance, if you wanted to keep all of your work during the first year in one place, you could name your project 'CST_Year_One' and then have a separate Repository for each Module ('GDV4000, GDV4001, etc.)'



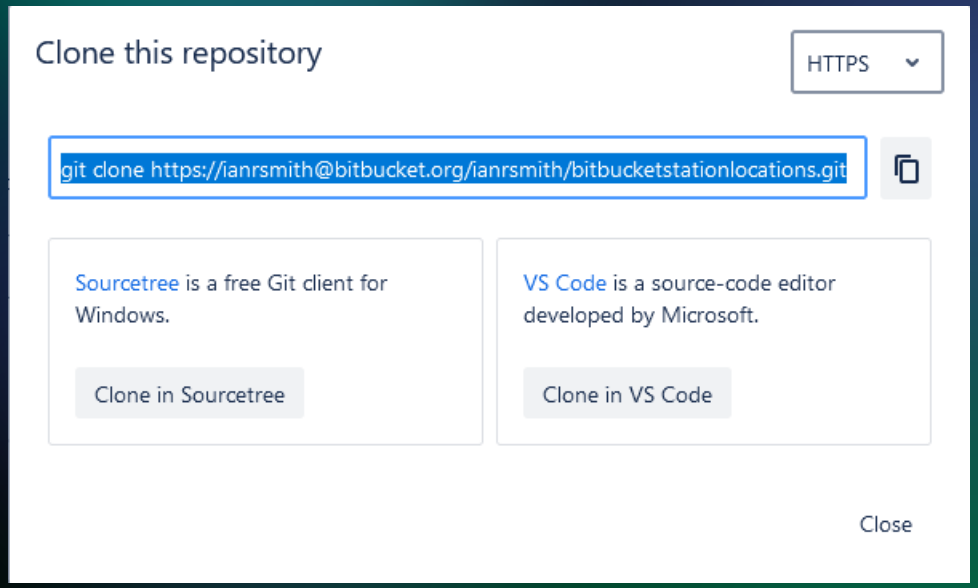


Now I have created a Repo (BitbucketStationLocations) within a Project (GDV4000_Demo).

This is our origin. Next, we need to make a local **copy** (Clone) on our own machine. This is the one we will be doing our work on.

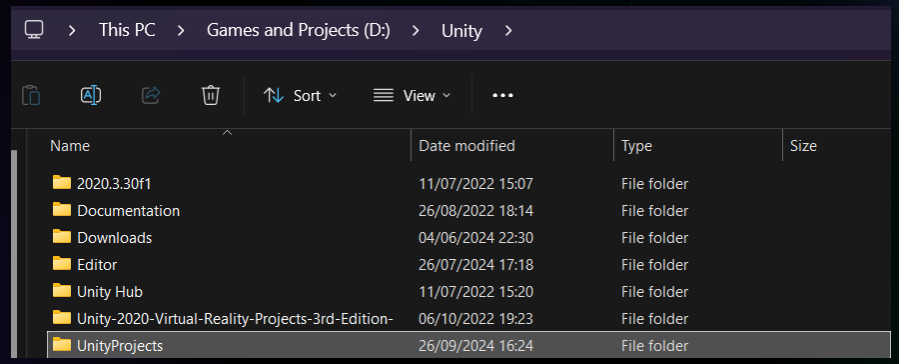


Click the Clone button in the top right of the screen. This pre-generates the command we will need to use in Git Bash to clone this to our local machine. Keep the setting as 'HTTPS' and click the Copy icon



Now we need to navigate to where on our local machine we want to keep our copy of the repo.

I have created a folder within the Unity folder called 'Unity Projects'



Open the folder and Right-Click in the Window.

If you are using Windows 11, you may need to click Show More Options to see the full menu (shown right).

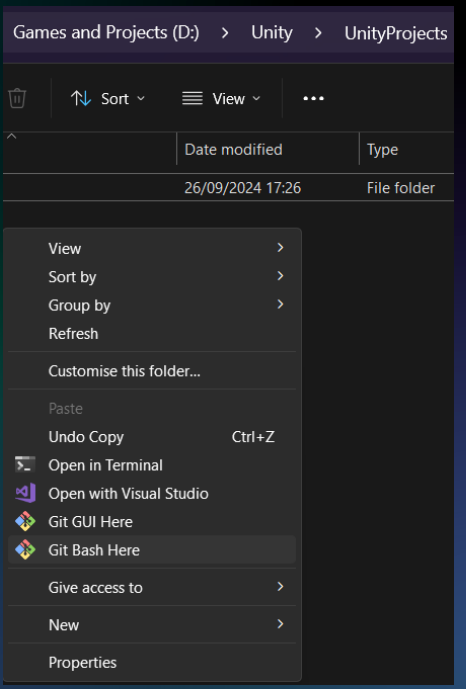
Hopefully you should see an option to open Git Bash Here. This will automatically point Git Bash to this directory without having to find it manually…

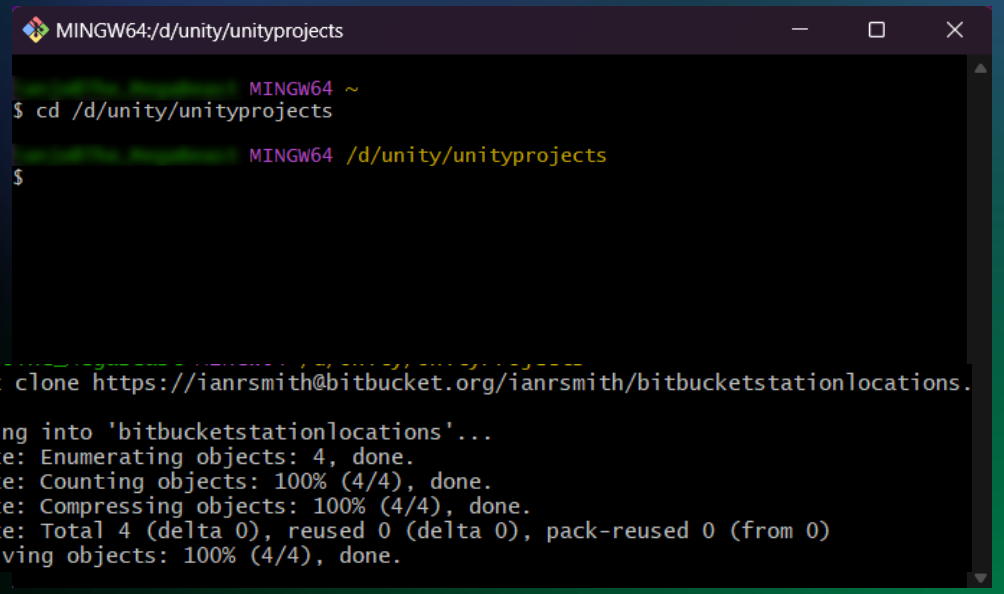
But if you don't have the extension installed, search for Git Bash in the Windows Search Bar and click on the icon to start Git Bash (as on the 1st slide).

We know the location we want is (in this specific case) on the D drive and inside a folder called Unity. Remember our command for Call Directory?

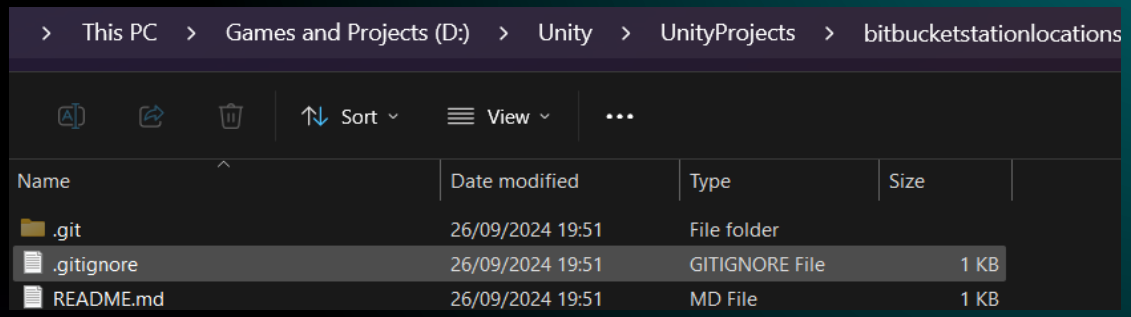
cd /d/unity/unityprojects

If you don't get an error and the address now appears after the identity of the PC in yellow, then all is well. We can now paste the HTTPS address we copied from Bitbucket into Git Bash. The shortcut to paste in Git Bash is Shift + Ins(ert).
If all has worked, then we will have a local copy of our repo.





Now we have our own local **Clone**/Copy of our repo, we can make a change and **Push** it back to the **Origin**



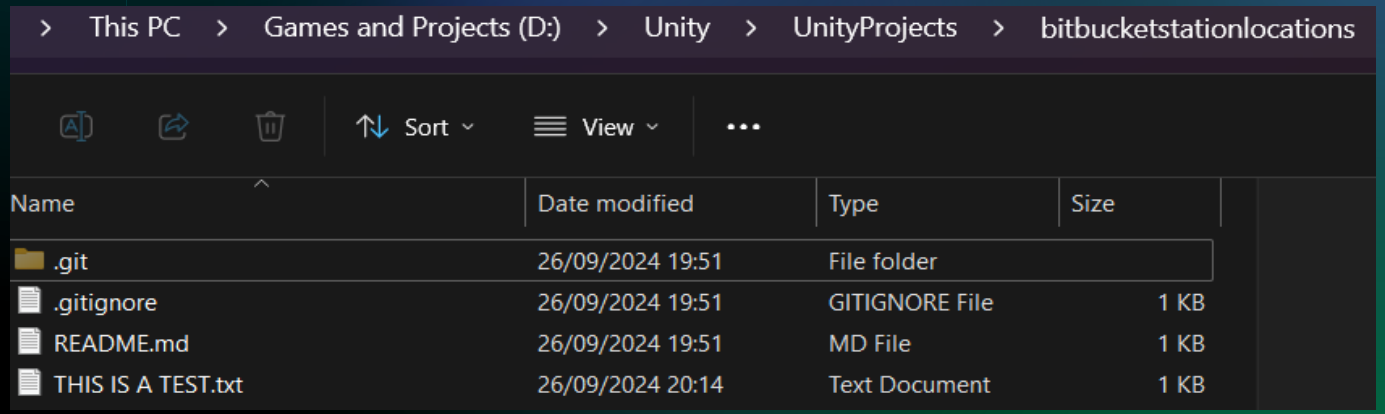
Open up the repo that we just copied to our local machine. We should see a 'hidden'' folder called '.git', a .gitignore file (we will come back to this), and a README.md file. These are auto-populated by Bitbucket.

Open Word or Notepad and text document. Write anything in it and then save it to the repo. Call it something like 'Test'.

Back to Git Bash!

1. We now want to **Push** our new file to Origin. To do this we need to tell Git Bash that we are Tracking these files. We do this using the command 'git add .' – The space before the '.' is important. We are saying that we want to add all the untracked files to the repo. You can use other tags to specify particular files to add.

```
[redacted] MINGW64 /d/unity/unityprojects/bitbucketstationlocations (ma
in)
$ git add .
```



2 .The next step is to **Commit** the files we tracked. Using the 'git commit –m *"insert message here"*' command enables us to add a message afterwards to note what changes were made. This can be invaluable if you need to remember where you left off, if you have things to fix, or if you're communicating with other team members.

3 . Once we have committed the current contents of the index, the next step is to **Push** the changes to the **Origin**.

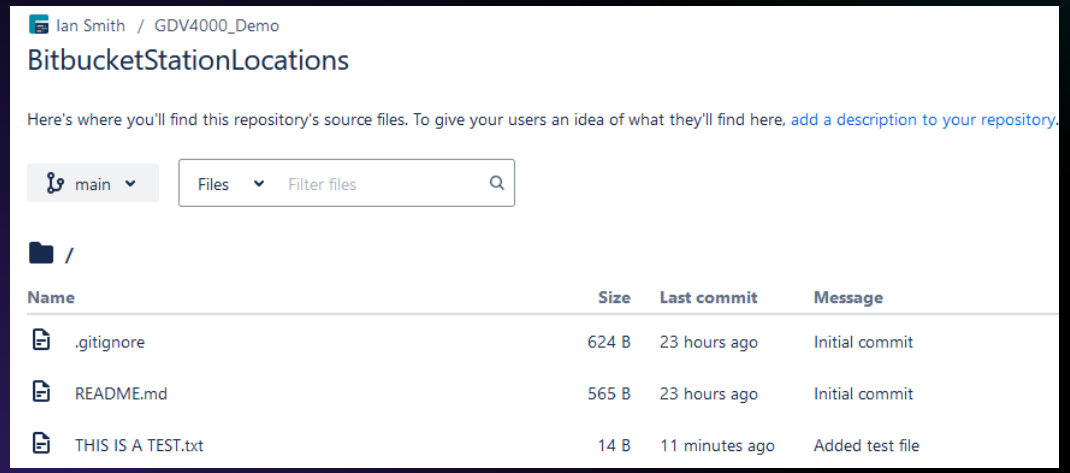
This is done with the 'git push' command. As we can see from the last line, it is pushing our changes back to the HTTPS address that we originally cloned the repo from.

Now back to Bitbucket…

```
MINGW64 /d/unity/unityprojects/bitbucketstationlocations (ma
in)
$ git commit -m "Added test file"
[main 861fbaa] Added test file
 1 file changed, 1 insertion(+)
 create mode 100644 THIS IS A TEST.txt
```

```
MINGW64 /d/unity/unityprojects/bitbucketstationlocations (ma
in)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 24 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 354 bytes | 354.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://bitbucket.org/ianrsmith/bitbucketstationlocations.git
   0365237..861fbaa  main -> main

MINGW64 /d/unity/unityprojects/bitbucketstationlocations (main)
$
```



Our new file has been successfully pushed to the Origin, which can be cloned onto a new machine with all the files up to date. You can carry on working and adding more files to the local repo and **Push** them.

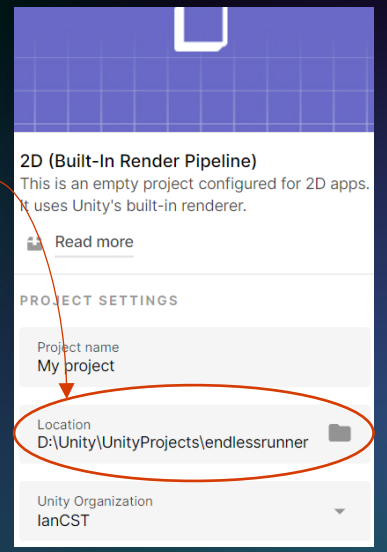
That is a very brief and basic overview of how we will use Git. Now to add Unity into the mix…

Problem: How do I use GIT?

Creating a Repo for Unity

The process is the same as above – We create repo, clone it onto our local machine and then we need to create a Unity project and save it *inside* wherever we have created our local repo.

The process is the same as above – We create repo, clone it onto our local machine and then we need to create a Unity project and save it *inside* wherever we have created our local repo.



One thing we need to do is create a Unity-specific .gitignore file and place this in the repo (you'll find one already prepped on Moodle). If there is already one in the repo that was made when it was created, overwrite it with the Unity one.

The .gitignore tells Git what files we need to include or exclude from our Repo. Unity has a lot of files that we do not need to keep track of or include.

Name your new Unity project and save it. Then open GitBash and Add, Commit, and Push it to the Origin.

Note – The first push may take a while, be patient.