UNIVERSITY OF ALBERTA

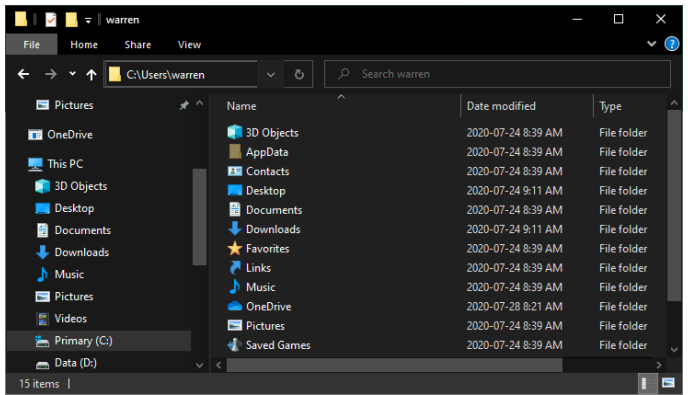
Git Bash (Terminal)

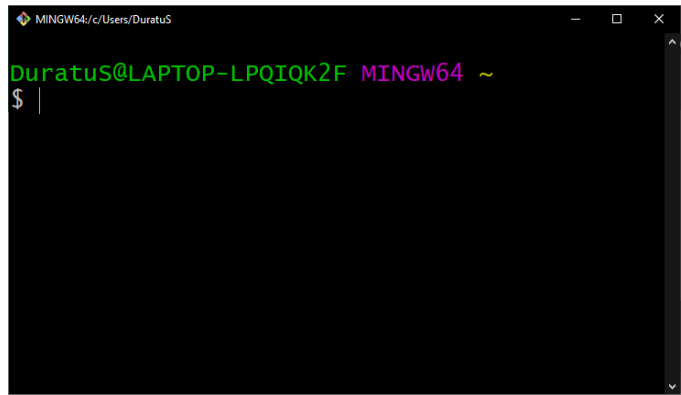
Using a **c**ommand-**l**ine **i**nterface (CLI.)

What *is* Git Bash, or any terminal for that matter?

Most users are familiar with File Explorer, and use it every day to navigate to files and folders on their computer. This is considered a GUI (**g**raphic **u**ser **i**nterface) file manager.

An alternative, is to use a CLI (**c**ommand-**l**ine **i**nterface.) That is to say, instead of using visual buttons and icons, we type commands to get where we want and do what we'd like!

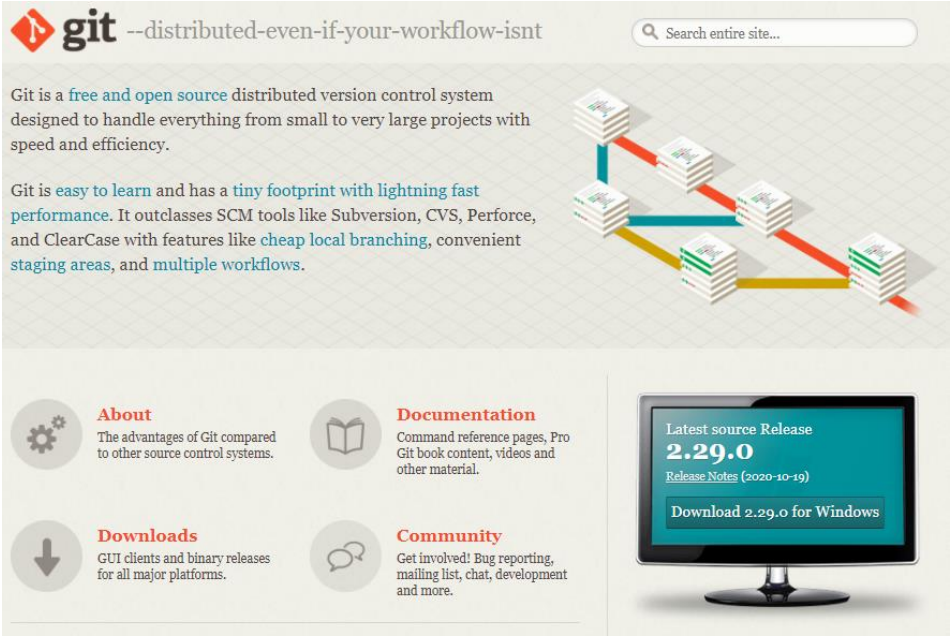




Downloading and installing the git and Git Bash programs.

Navigate in a web browser to the official website for git: https://git-scm.com/

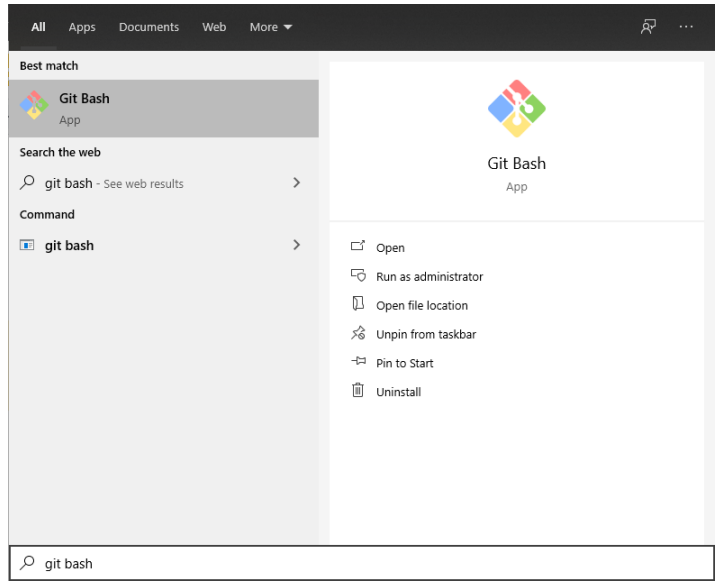
Download and install the version of the software most appropriate for your system.



Opening Git Bash.

If you're using a recent version of Windows, hold the Windows key and press the "S" key to open the Windows Search interface.

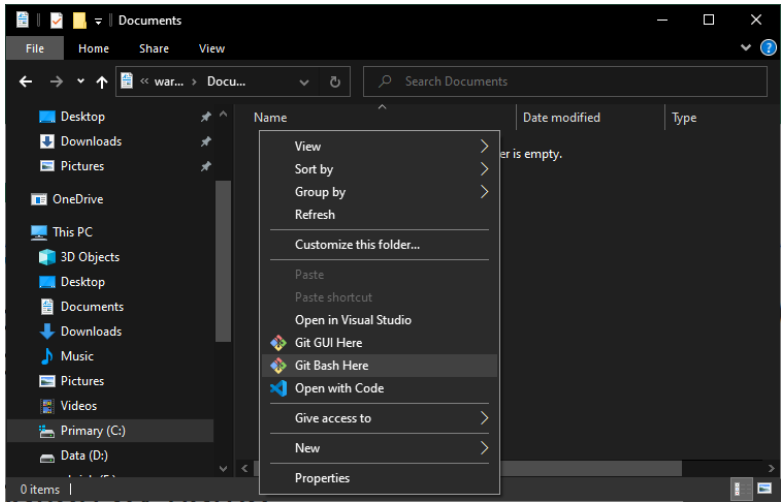
From here, you can easily search "Git Bash" and select the application by name.



Another way to open Git Bash.

Depending on your installation choices, you may also have new options available in your “right-click” context menu.

If this option is enabled for you, you may navigate using File Explorer to a folder of your choosing, right click in an empty space, and select “Git Bash Here” to open the current directory in the Git Bash program.



Navigating using Git Bash.

There are a number of commands you can make use of to navigate to different files and folders on your computer using Git Bash.

Some of note include:

- pwd
- ls
- cd

Other commands:

- mkdir
- touch
- rm

Present Working Directory (PWD).

It is important to know where you are currently looking before you do anything!

To check where you are right now, use the “pwd” command.

After you type the command, hit enter, and it should display the path representing where you are right now.

```
$ pwd
/c/Users/warren
```

List current directory contents (LS).

Okay, so you know where you are now!

What if you'd like to see what files or folders are available here?

Type in "ls" and hit enter, you'll be presented with a list of all available files and folders at your current path.

```
$ ls
'3D Objects'/
 AppData/
'Application Data'@
 Contacts/
 Cookies@
 Desktop/
 Documents/
```

Change Directory (CD).

You know where you are, you know what is available…

Now for the cool part—how to navigate from your current location, to somewhere else.

Let's say we want to enter the "Documents" folder.

To get yourself there, type:

cd "Documents"

```
$ cd "Documents"
```

Did I get inside that folder?

If you navigate successfully, you won't see a success message. There is a way to check where you are now, though!

Remember "pwd"? It can be useful to run that after changing directory to make sure you're where you think you are.

```
$ cd "Documents"
```

```
$ pwd
/c/Users/warren/Documents
```

Make Directory (MKDIR).

Say we're in the "Documents" folder and we'd like to make a new directory inside for some text documents.

We can create this new folder by typing the command "mkdir", followed by the new folder's name:

mkdir "text-docs"

On success, there won't be a message, but you can always use the "ls" command to check if your new folder is here.

```
$ pwd
/c/Users/warren/Documents
$ mkdir "text-docs"
```

```
$ ls
 desktop.ini   'My Pictures'@   text-docs/
'My Music'@    'My Videos'@
```

Touch.

We can make use of the “touch” command to create a new, empty, file.

```
$ pwd
/c/Users/warren/Documents
```

Perhaps we’d like to go inside our new “text-docs” folder and create a “my-text.txt” file, how would we get that done?

```
$ cd "text-docs"
```

Go inside your new folder, if that’s where you want to work:
cd “text-docs”

Use “touch” to make the new file:
touch “my-text.txt”

```
$ pwd
/c/Users/warren/Documents/text-docs
$ touch "my-text.txt"
$ ls
my-text.txt
```

Remove file (RM).

We created a file! What if we didn't mean to, or we're done with it? How do we delete it!?

Use the "rm" command, followed by the file's name:
rm "my-text.txt"

```
$ pwd
/c/Users/warren/Documents/text-docs
$ ls
my-text.txt
$ rm my-text.txt
$ ls
```

Navigating “Up a Level.”

We ended up inside of the “text-docs” directory we made… But how do we get back out? Back into the “Documents” folder?

We can go to a parent folder by typing:

cd ..

The “..” represents the folder that contains the one we’re already in; very handy!

```
$ pwd
/c/Users/warren/Documents/text-docs
```

```
$ cd ..
```

```
$ pwd
/c/Users/warren/Documents
```

Remove Directory (RM -d).

Now that we're back in the "Documents" folder. Let's clean up that "text-docs" folder we made earlier.

If we want to remove a folder, instead of a file, we type the "rm" command like before, but with the "-d" option and the folder name like so:

rm -d "text-docs"

```
$ pwd
/c/Users/warren/Documents
```

```
$ ls
 desktop.ini   'My Pictures'@   text-docs/
'My Music'@    'My Videos'@
```

```
$ rm -d "text-docs"
```

```
$ ls
 desktop.ini   'My Pictures'@
'My Music'@    'My Videos'@
```

Cheat Sheet.

Commands we've covered:

- Present Working Directory (**PWD**)
- List (**LS**)
- Change Directory (**CD** "directory name")
- Make Directory (**MKDIR** "directory name")
- Touch (**TOUCH** "file name")
- Remove (**RM** "file name")
- Remove Directory (**RM -d** "directory name")

Relative Paths.

So far, we've been using what are called "relative paths."

This means, if we're in the right place, we can just refer to a file or folder name directly in our commands.

Example)
I open a new Git Bash window, and run "ls". I see "Documents" in the listing, and want to go there. I can get there by simply typing:

cd "Documents"

Absolute Paths.

The strength of relative paths is that if we are already close to the folder or file, it is easy and quick to get there!

The alternative are “absolute paths.” These are the **full address** to the file or folder. Here is the previous example adapted to an “absolute path” solution.

Example)
cd “/c/Users/warren/Documents”

The strength, here, is you could be *anywhere* on your PC and an absolute path will work! Note they start with a slash.

Recommended Reading

If you'd like to read more on common commands, and gain more context on the history and purpose of the Bash terminal, have a look through the following:

- Kirkbride, P. (2020). *Basic Linux Terminal Tips and Tricks: Learn to Work Quickly on the Command Line*. Berkeley, CA: Apress.
  (Look for the section in Chapter 1 called "Common Commands" for more commands like we covered today!)


UNIVERSITY OF
ALBERTA