



# Where Do We Start?





# Git Is (Primarily) A Terminal Tool

Git was created as command-line tool. To use it, we run various git commands in a Unix shell. This is not the most user friendly experience, but it's at the very core of Git!





# The Rise of GUI's

Over the last few years, companies have created graphical user interfaces for Git that allow people to use Git without having to be a command-line expert.

Popular Git GUI's include:

- Github Desktop
- SourceTree
- Tower
- GitKraken
- Ungit





# GUI Clients



## Pros

- Way lower barrier-of-entry for beginners compared to the command-line.
- Friendlier to use. Can be a much better experience (when it works)
- Some people prefer the visual experience, even those who can use the command-line

## Cons

- At times, there is lots of "magic" involved. The inner-workings of Git are obfuscated and hidden away with GUI's.
- Often leads to dependance on a particular piece of software.
- When things go seriously wrong, it can be very challenging to fix without the command-line
- The interfaces, buttons, and menus vary between different GUI's.





# The Command Line



## Pros

- Git is a command-line tool. All the documentation and resources online will refer to the command-line
- The command-line can be way faster once you get comfortable with it!
- Some of the more advanced Git features are only available on the command-line
- The commands are always the same, no matter what machine you are on!

## Cons

- Not beginner-friendly. At All. Can be difficult to learn and remember the commands at first.
- Even for some practiced users, the command-line interface is just not a good experience. It's really a matter of preference.



There is a lot of stupid  
GUI gate-keeping



Are you a developer? Do you plan on becoming a developer?

**Learn the command-line!**

you'll need to use the command-line any way

Learning Git for other purposes?

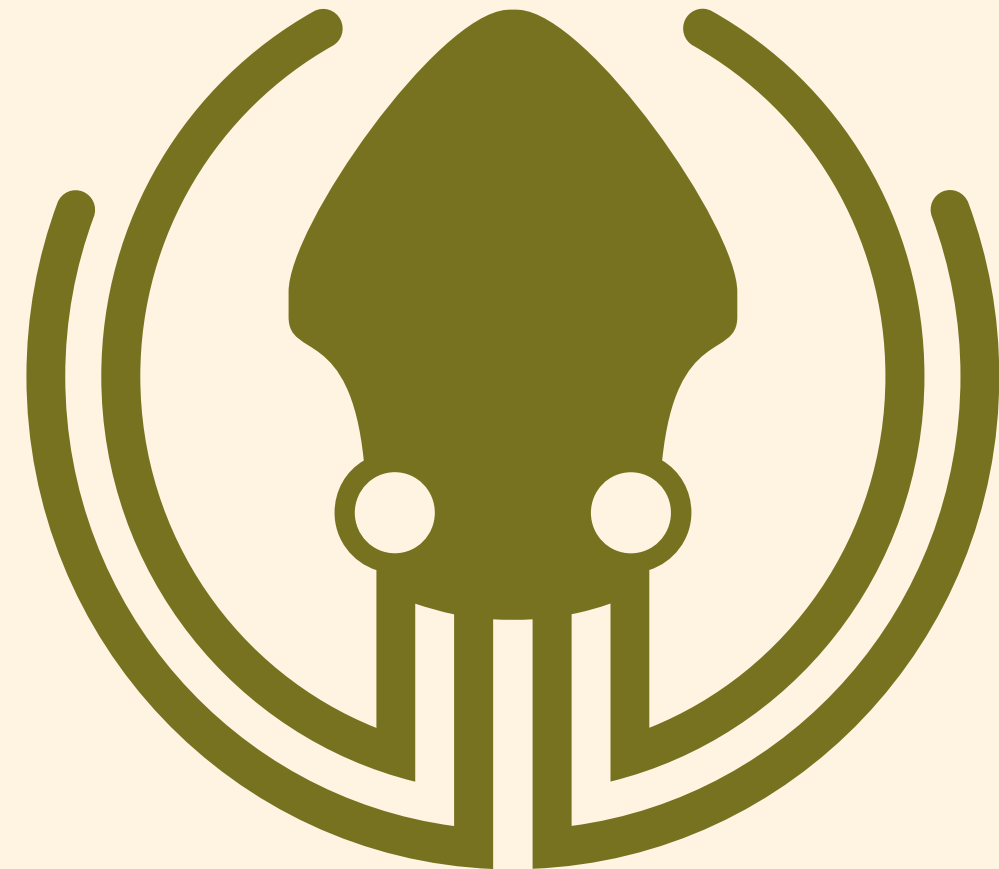
**Try a GUI!**  
(or don't!)



# Installing a GUI

There are many options to choose from depending on your operating system, and most of them are incredibly simple to install!

I will be using **GitKraken** throughout the course, and I recommend you install it too. It's free, though there is a paid tier that we don't need.







# Installing Git

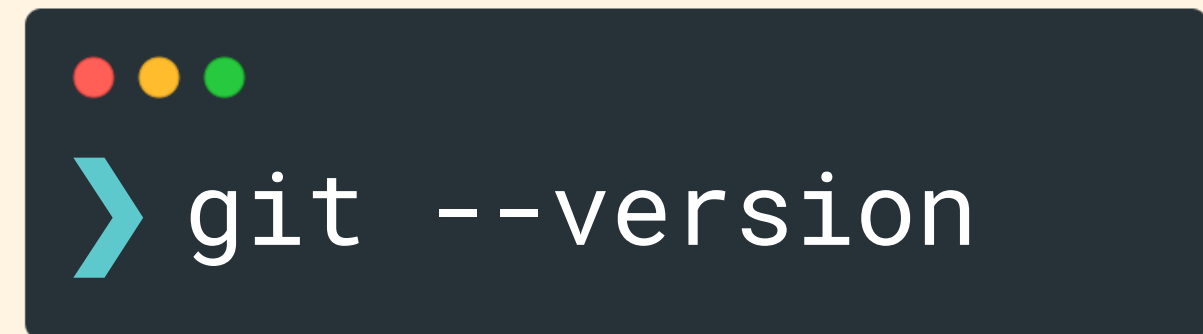
Installing Git locally is slightly trickier, depending on your specific operating system.

Git is intended to run on Unix-style systems like Linux and MacOS, so if you're on a Windows machine you may need to take a couple extra steps.



# Mac Install

- First, check to see if you have Git installed already using the command **git --version**
- If not, or if you have an old version, download the latest Git installer package using the link in the description
- Verify your install worked by running **git --version** again afterwards

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top-left corner. A light blue prompt character is followed by the text `git --version` in white.

```
> git --version
```

# Windows Install

Git Bash emulates the unix-based Git command-line experience for Windows machines, and it's super easy to install!

- Download **Git For Windows**
- Find the downloaded .exe file and open it to execute Git Bash.

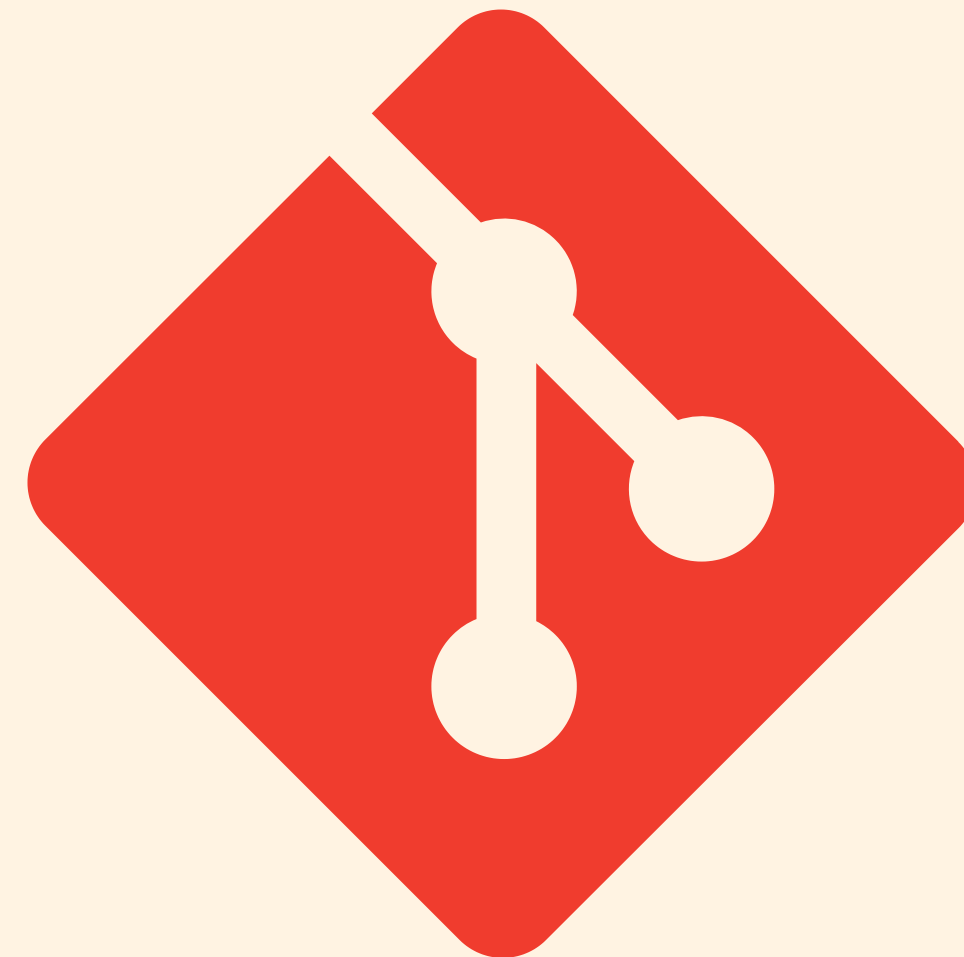


# Configuring Git

Now that Git is hopefully installed, it's time to configure some basic information. You do not need to register for an account or anything, but you will need to provide:


- Your name
- Your email

If you are using a GUI, it should prompt you for your name and email the first time you open the app.



# Configuring Git

To configure the name that Git will associate with your work, run this command:

```
 > git config --global user.name "Tom Hu1ce"
```

# Configuring Git

Do the same thing for your email using the following command. When we get to Github, you'll want your Git email address to match your Github account.

```
  
> git config --global user.email blah@blah.com
```

# Let's Get Started!

