



WSL 2 for Windows 10

Microsoft  Linux



WSL 2 is relatively new, and potentially buggy. It is still under very active development. If you are uncomfortable with configuring and troubleshooting Windows and Linux, this may not be a good solution for you. If you run into problems, be prepared for there to not currently be a fix for them. While many students do run WSL 2 as their sole development environment during the course, we highly recommend running a WSL 2 installation in parallel with another development environment such as macOS or Ubuntu that you can fall back on if you run into an error that cannot be resolved. Proceed at your own risk.

Contents

[Contents](#)

[What You Need to Begin](#) (*YOU MUST READ THIS DO NOT SKIP THIS SECTION THIS IS IMPORTANT*)

[Troubleshooting](#)

[Notion](#)

[Slack](#)

[Zoom](#)

[Microsoft PowerToys \(Optional\)](#)

[A Note On Copying Commands](#)

[Copying code blocks from Notion](#)

[Run Windows PowerShell as Administrator](#)

[Install WSL](#)

[Restart your computer](#)

[Handling Errors](#) ❤️

[Virtual Machine Platform Error](#)

[The Ubuntu Application Does Not Start Upon Login](#)

[I Do Not Have Either of the Above Errors](#)

[Creating a User Account](#)

[Handling Errors](#) ❤️

[Virtual Machine Could Not Be Started Error](#)

[Install Windows Terminal from the Microsoft Store](#)

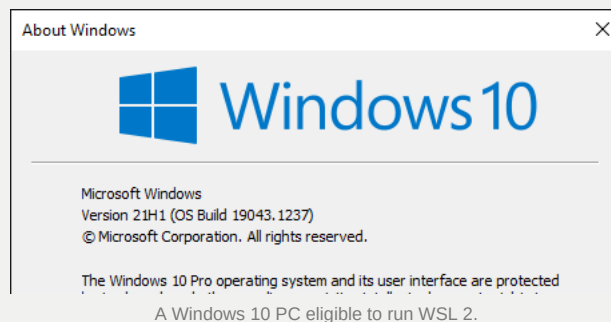
[Launching Ubuntu 20.04 LTS](#)
[Updating and Upgrading Packages](#)
[zsh](#)
[Oh My Zsh](#)
[Visual Studio Code](#)
 [Install the Remote WSL extension](#)
[Git in Windows 10](#)
[Git in Ubuntu](#)
 [Configuring a Global Git Ignore File](#)
 [Here is a .gitignore_global file for you to use.](#)
[Github](#)
 [GitHub CLI](#)
 [Handling Errors](#) ❤️
 [GitHub CLI Login](#)
 [Generating a GitHub Personal Access Token](#)
[So much done! Just a little more now...](#)
[Node.js](#)
 [Handling Errors](#) ❤️
 [command not found: nvm Error](#)
[Heroku](#)
[Being More Productive By Using the Keyboard Instead of the Mouse in Windows](#)
 [Launching Apps with Search](#)
 [Switching Between Applications](#)
[Taking Screenshots](#)
 [Uploading Screenshots and Images to \[imgur\]\(#\)](#)
[OH WOW YOU DID IT!](#)
[Level Up](#) 🚀
 [A Password Manager](#)

What You Need to Begin *(YOU MUST READ THIS DO NOT SKIP THIS SECTION THIS IS IMPORTANT)*

- *A device running Windows 10 updated to version 20H2 (build 19042 or greater). We do not support Windows 10 versions prior to this as they have reached [End of Service](#). WSL 2 WILL NOT RUN on versions before this.*



To find your Windows version and build number use [Windows Logo Key + R](#) on your keyboard, type **winver**, and select OK. You'll see a dialog window like the one below.



- A familiarity with your system's BIOS may be required. This is extremely important as you may need to adjust BIOS settings to complete the WSL install, particularly if your machine uses an AMD processor. You are not able to screen share within the BIOS environment and it will be different depending on the device you use. Therefore, it is ON YOU to be able to enter this

environment and find the settings you will need to change. *Modifying the incorrect settings in the BIOS can cause permanent hardware damage to your device, which we are not liable for.*

- A non-Windows 10 S Version of Windows 10
- A user account with administrative privilege to your local installation of Windows 10.
- A Microsoft Account with access to the Microsoft Store application. (All requirements are free, but some are only available from the Microsoft Store)
- At least 15GB of free hard drive space
- At least 8GB of RAM (16GB of RAM or more is GREATLY preferable and will vastly improve your coding experience)
- A modern processor capable of running virtual environments - specifically processors with Intel Virtualization Technology (Intel VT) or AMD Virtualization (AMD-V) technology
- A fundamental understanding of Windows and Linux system administration and debugging.

Troubleshooting

If you run into issues during this process Microsoft has a collection of errors and their suggested fixes [here](#). If your issue is not included in that documentation you can also search for your specific error on the [WSL GitHub Issues](#) page. If you've exhausted those resources try searching Google for a solution. If your issue remains unresolved, reach out to an instructor for further guidance.

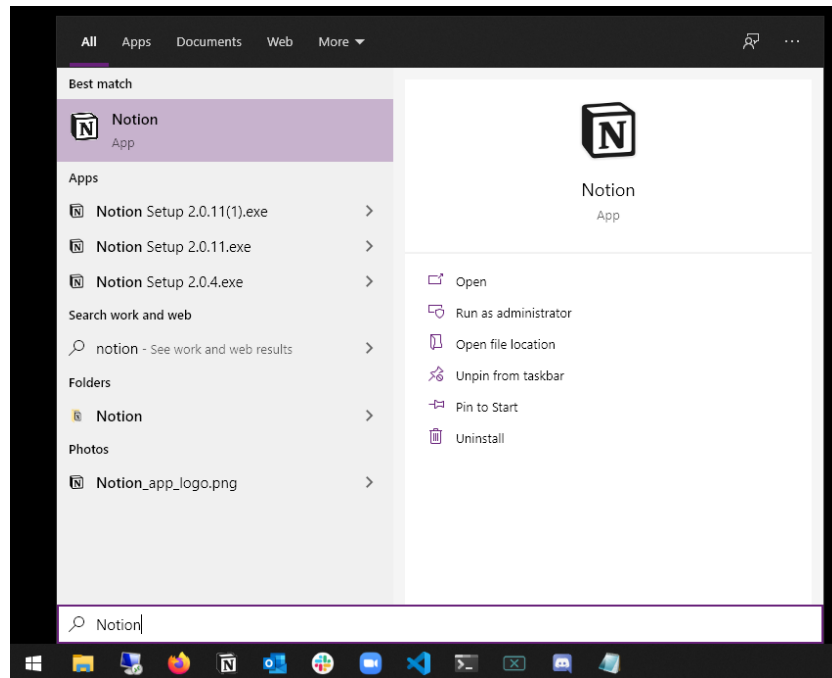


Be prepared for there to not currently be a fix for your problem.

Notion

Notion is an all-in-one workspace which all the course content will be hosted on. You can download Notion [here](#). Be sure to select the **Download for Windows** option when it is given. Install it by opening the downloaded .exe file and completing the installation steps.

When the installation is complete open the Notion application. To do so, press the **Windows Key** to launch Windows Search and type **Notion**, then select the Notion application by pressing **Enter** when it appears, as shown in the screenshot below.



Launching the Notion application using Windows Search. Get used to seeing this often; it's the fastest way to start applications on Windows!

First thing's first, when the application launches, **pin it to your task bar**. We'll be using this application throughout the day every day. You will be presented with a login screen. **DO NOT** select **Continue with Apple** or **Continue with Google**. Instead, **enter the email you used to apply to General Assembly** in the provided area and select **Continue with email**.

After doing so, you will be prompted to check your email for a login code. Do so and enter the code to create a new account.

Upon signing up, you should have access to the course content, including this installfest! Continue on from here in the app.




Don't have access to the course content? Let your instructor know immediately so that you can complete the installfest and have access to the course content.



Welcome to Notion

Log in to sync your content.

 Continue with Google

 Continue with Apple

Email

alargecaribou@gmail.com

There is no existing account under this email address. We just sent you a temporary sign up code.

Please check your inbox.

Sign up code

Enter login code

Create new account

You can also [continue with SAML SSO](#)

The Notion create an account page - an account tied to an email is being created.

Slack

We will be using slack to communicate throughout the course. While you can technically log in via the web browser, installing the app is highly recommended. Download Slack [here](#) and install it.

Zoom

If you haven't already, [download the Zoom client](#) and install it.

Microsoft PowerToys (Optional)

From Microsoft:

Microsoft PowerToys is a set of utilities for power users to tune and streamline their Windows experience for greater productivity.

Inspired by the Windows 95 era PowerToys project, this reboot provides power users with ways to squeeze more efficiency out of the Windows 10 shell and customize it for individual workflows.

Among its many useful tools, PowerToys includes one of the best window managers for Windows: FancyZones - which allows you to customize window layouts and get the best setup just for you.

Get PowerToys [here](#). Learn more about FancyZones [here](#).

A Note On Copying Commands

When possible, ***please copy the commands from this page***. You will use most of the commands here once and never again.

Typing them out will only introduce the possibility for you to make errors. Certain commands will require you to alter portions of them - this is specifically called out when they appear. There are no bonus points for doing work that has already been done for you.

Copying code blocks from Notion

If the code you're trying to copy is only on a single line it is easiest to triple-click the line you are trying to copy and use `ctrl + c` to copy it and `ctrl + v` to paste it.

If the code you're trying to copy spans multiple lines you'll want to click and drag to highlight *only the text* that you intend to copy.



Never highlight an entire code block to copy it as shown in the image below:

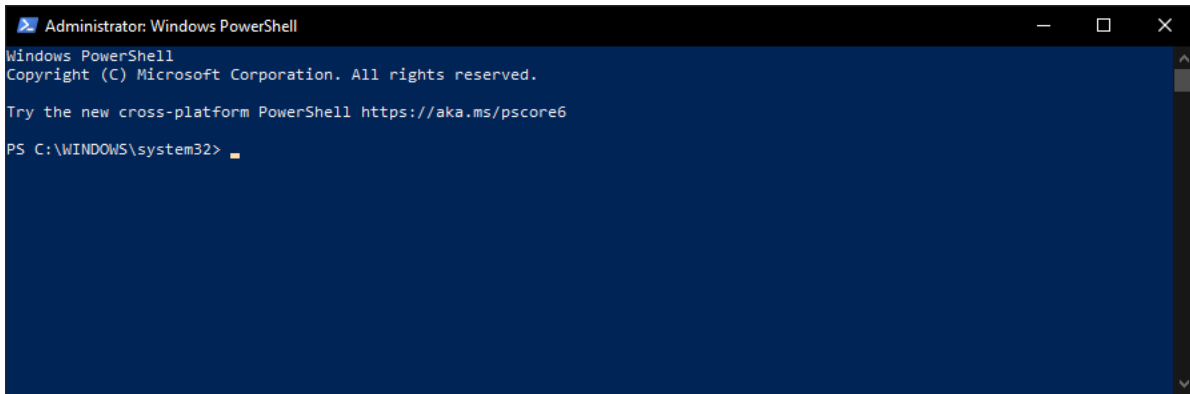
```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

Do not do this!!

This will result in extra text being added to the code you are attempting to copy.

Run Windows PowerShell as Administrator

Use `Windows Logo Key + R` on your keyboard, type **powershell**, and use `Ctrl + Shift + Enter` on your keyboard to run as administrator. You will be prompted to allow elevated permissions - do so.

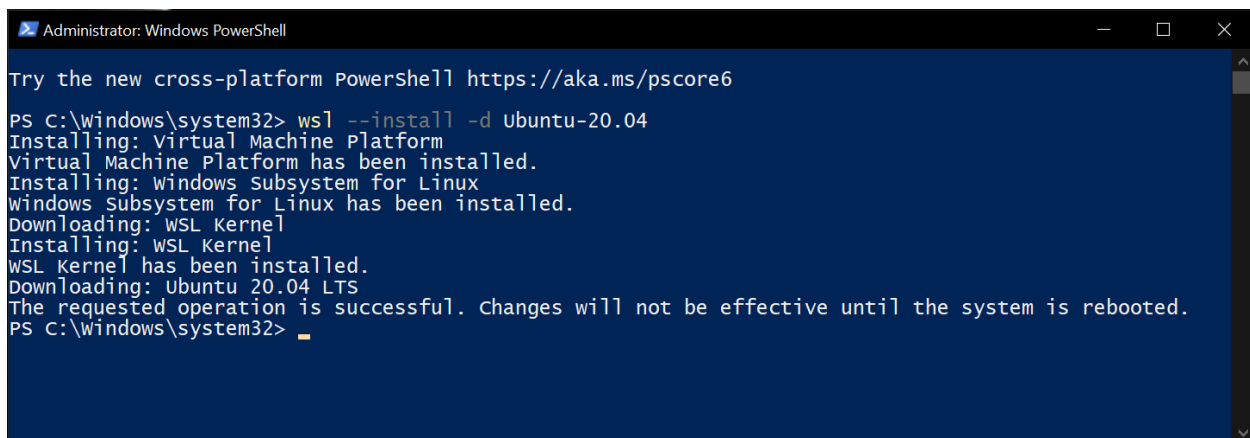


PowerShell running with administrative access. Note the title bar indicating that PowerShell is running with elevated permissions!

Install WSL

Use this command to install Ubuntu 20.04 in WSL2.

```
wsl --install -d Ubuntu-20.04
```

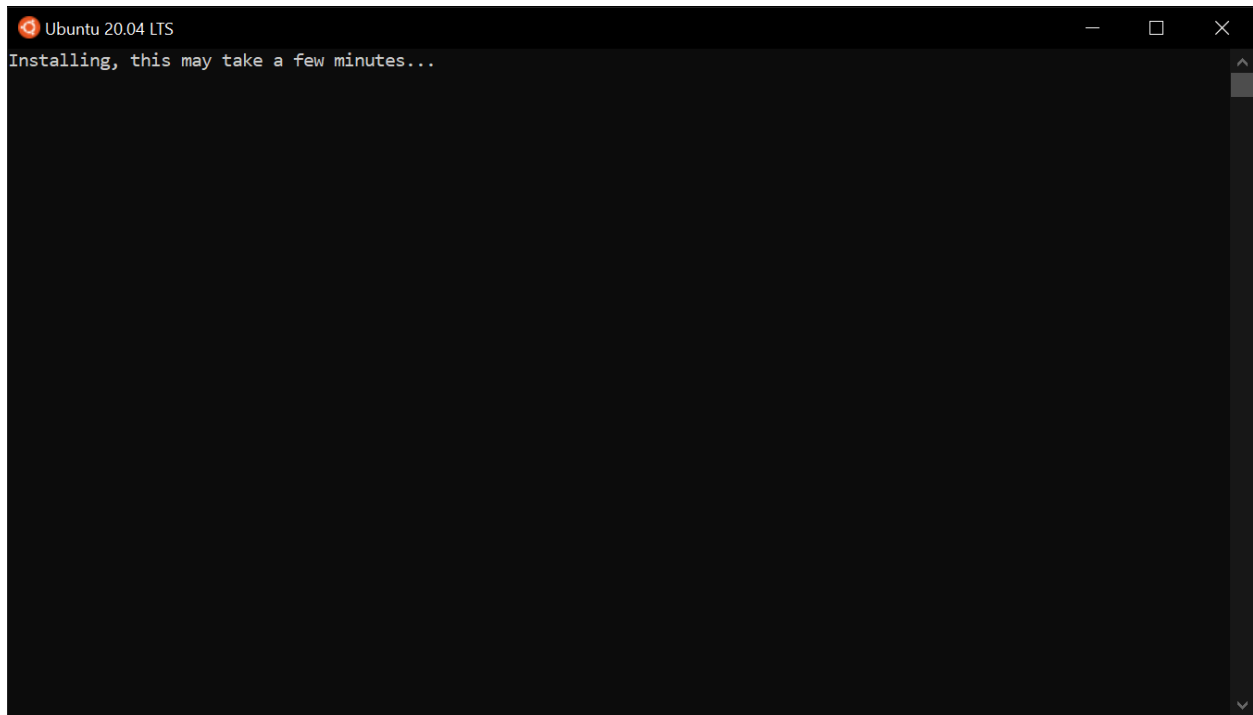


The expected output shown in PowerShell after running the above command.

Restart your computer

Save any work you want to keep, including this page and restart your computer to continue!

Upon restarting the Ubuntu Installer will launch automatically then finalize the installation, which may take a few minutes. If you get an error message or run into another issue check out the **Handling Errors** ❤️ subsection below.



The Ubuntu Installer auto-running after a system restart.

Handling Errors 💔

Virtual Machine Platform Error

You may see this message after your machine restarts:

```
Please enable the Virtual Machine Platform Windows feature and
ensure virtualization is enabled in the BIOS.
```

If this occurs, run the command below in PowerShell with Administrator permissions (reference the above instructions to launch PowerShell in as the Administrator):

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

and restart your machine.

If the error persists after restarting you likely need to enter your BIOS and enable Intel Virtualization Technology (Intel VT) or AMD Virtualization (AMD-V) technology to continue. This error is most common with AMD processors.

The Ubuntu Application Does Not Start Upon Login

Launch the Ubuntu 20.04 LTS with a search for `Ubuntu` in the start menu.

I Do Not Have Either of the Above Errors

Reach out to an instructor for further guidance.

Creating a User Account

You will then be prompted to create a username and password. The username should not have any spaces in it. *The password will not be visible as you type it. This is common in many command line applications.* It is **extremely important** that you do not forget

this username and password - preferably you will store it in a password manager such as [Bitwarden](#) or [1Password](#).

```
david@hyperion: ~
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: david
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.10.16.3-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Sep 21 20:36:01 CDT 2021

System load:  0.0          Processes:            8
Usage of /:   0.5% of 250.98GB   Users logged in:    0
Memory usage: 0%            IPv4 address for eth0: 172.25.17.68
Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

This message is shown once a day. To disable it please create the
/home/david/.hushlogin file.
david@hyperion:~$
```

The Ubuntu application after successfully finalizing installation



You're now running WSL 2! Congrats! Go ahead and close the Ubuntu 20.04 application - we won't use it again. Don't close this guide though, you're not quite done yet...

Handling Errors 💔

Virtual Machine Could Not Be Started Error

You may see the following error after launching Ubuntu:

```
Select Ubuntu 20.04 LTS
Installing, this may take a few minutes...
WslRegisterDistribution failed with error: 0x80370102
Error: 0x80370102 The virtual machine could not be started because a required feature is not installed.

Press any key to continue...
-
```


If this occurs, run the command below in PowerShell with Administrator permissions (reference the above instructions to launch PowerShell in as the Administrator)

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

and restart your machine.

Then re-launch Ubuntu using the Ubuntu application. If the error persists you likely need to enter your BIOS and enable Intel Virtualization Technology (Intel VT or Intel VMX) or AMD Virtualization (AMD-V) technology to continue. This error is most common with AMD processors.

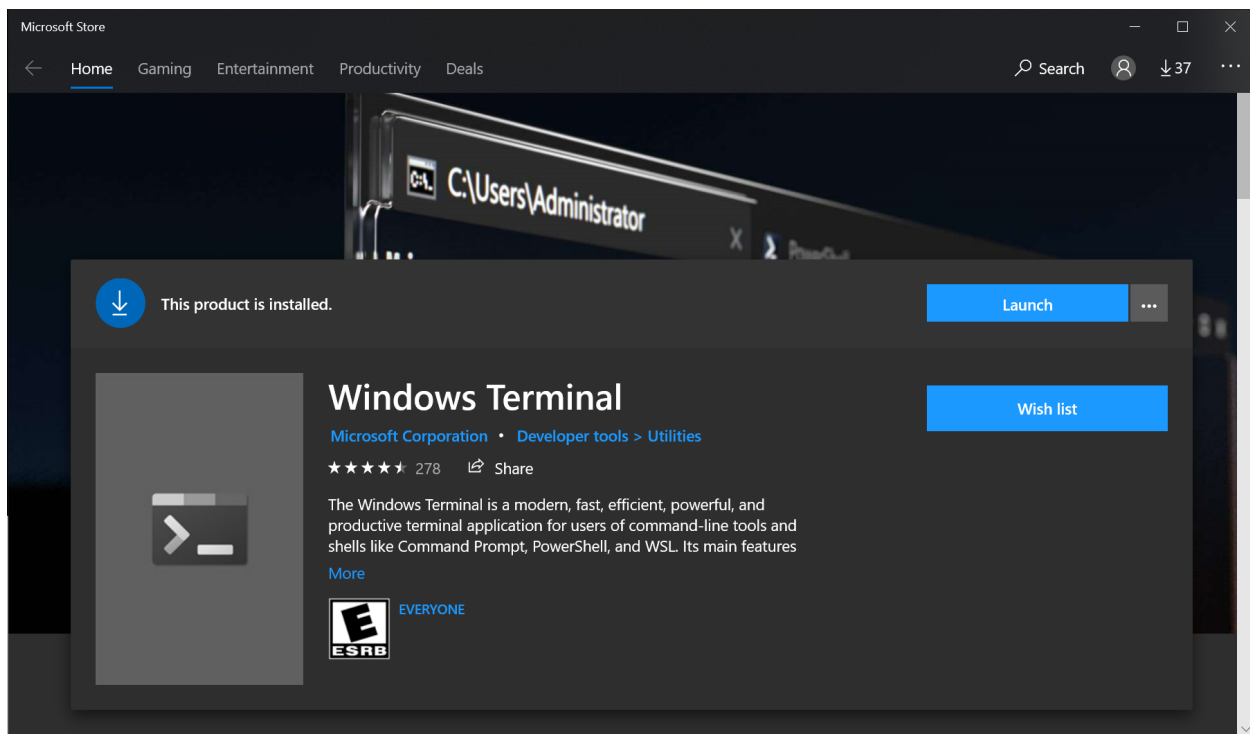
Install Windows Terminal from the Microsoft Store

Follow [this link](#) to **Windows Terminal** on the Microsoft Store. Select the **Get** button which will launch the Microsoft Store application. Select the **Get** button again on the page inside the Microsoft Store application.



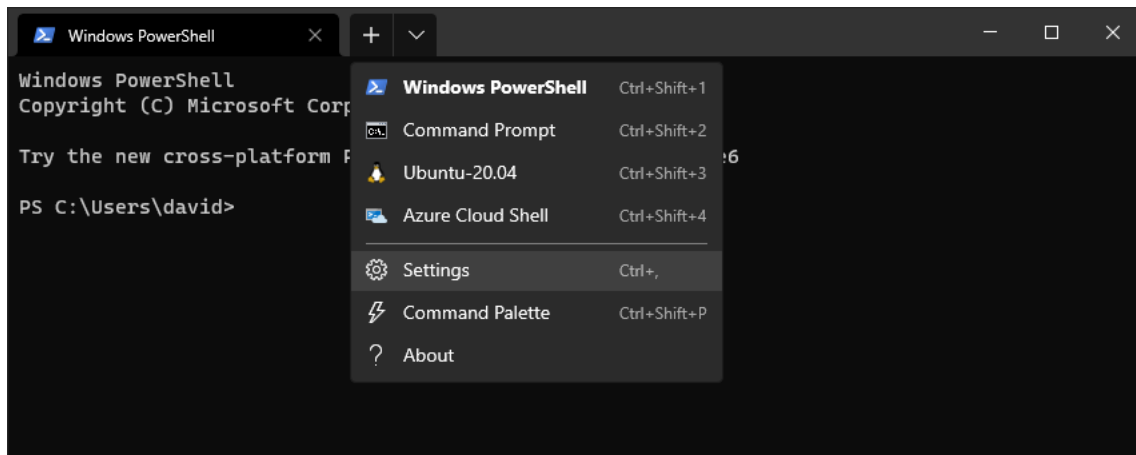
If the Microsoft Store application doesn't launch when you click on the **Get** button in your browser, you may need to manually launch the Microsoft Store application and find **Windows Terminal** in the store through a search.

Windows Terminal will download and install and can then be started by selecting the **Launch** button.



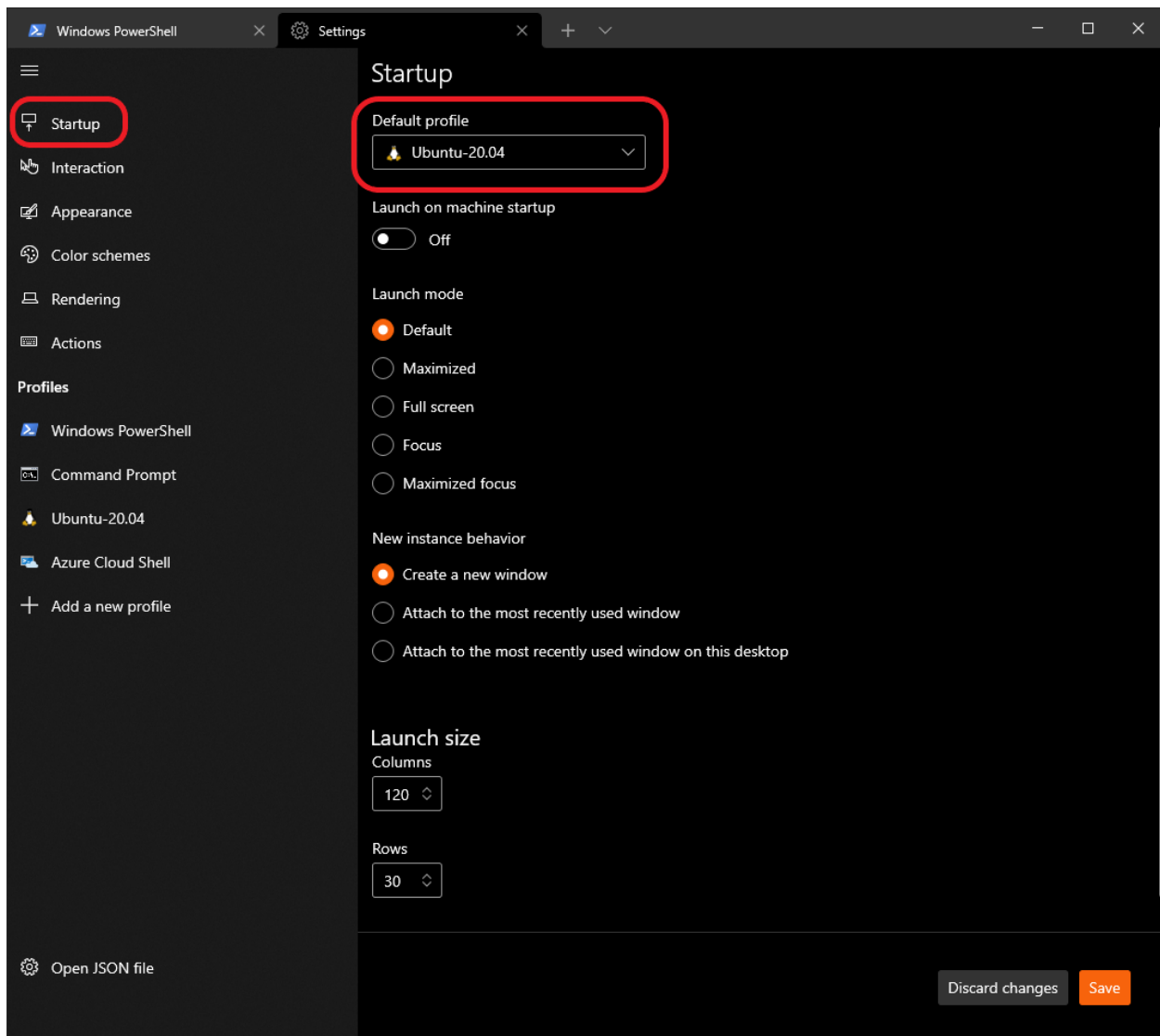
When the Windows Terminal is launched **pin it to your task bar. From now on you will launch Ubuntu from here.**

Windows Terminal will initially launch with only a Windows PowerShell tab. Let's configure it so that it launches into your Ubuntu installation by default. Select the dropdown arrow in the title bar, then select **Settings** to open the Windows Terminal settings.



Accessing the **Settings** tab in Windows Terminal.

The **Settings** tab should open with the **Startup** section already selected. The first order of business is changing the **Default profile** setting in this section to be **Ubuntu-20.04**.

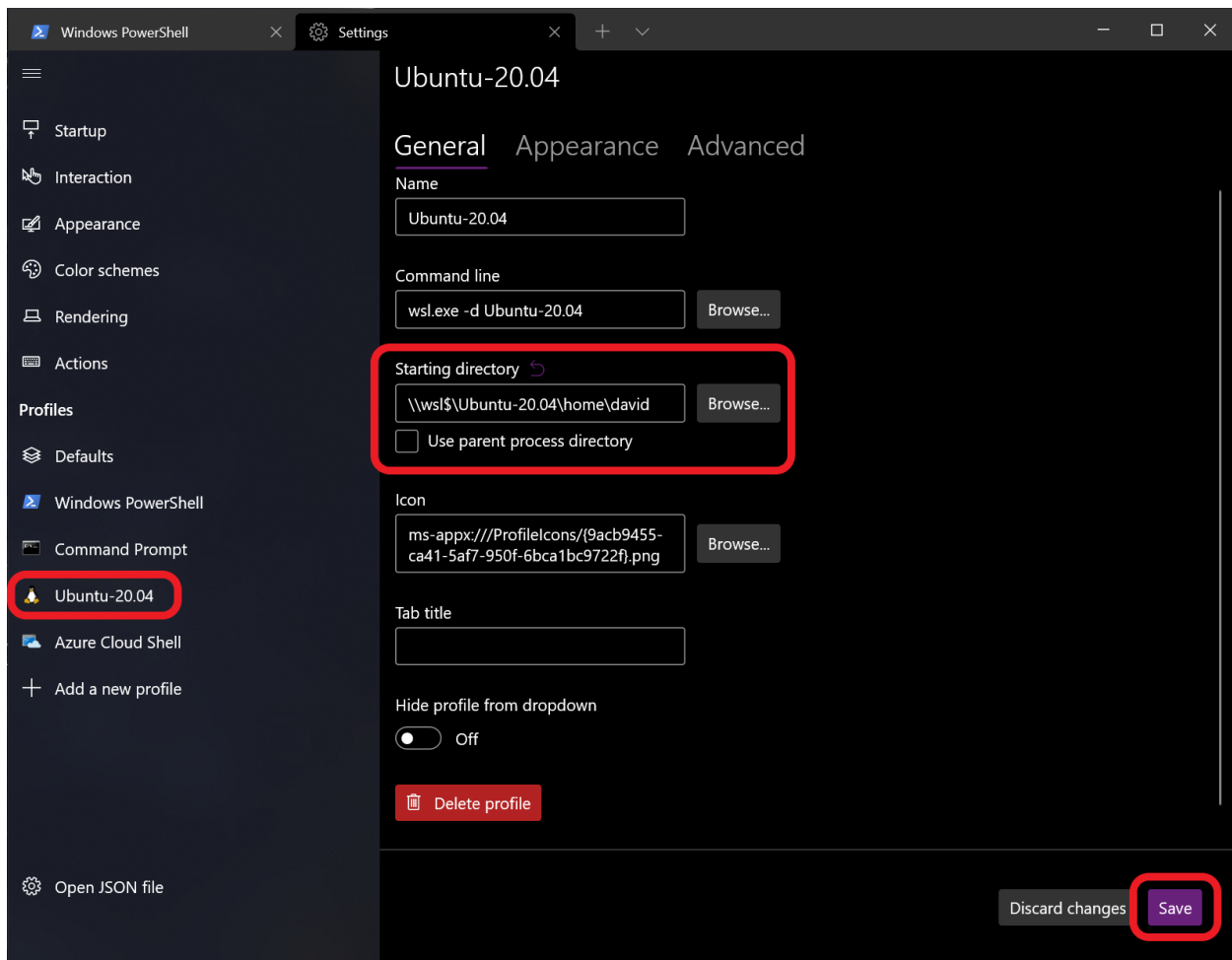


In the **Settings** tab in Windows Terminal, the **Startup** section has been selected and the **Default profile** has been changed to **Ubuntu-20.04**.

Next, navigate to the **Ubuntu-20.04** section. Change the **Starting directory** to the following, replacing `<Your Ubuntu Username>` with the Ubuntu username that you created above:

```
\\wsl$\\Ubuntu-20.04\\home\\<Your Ubuntu Username>
```

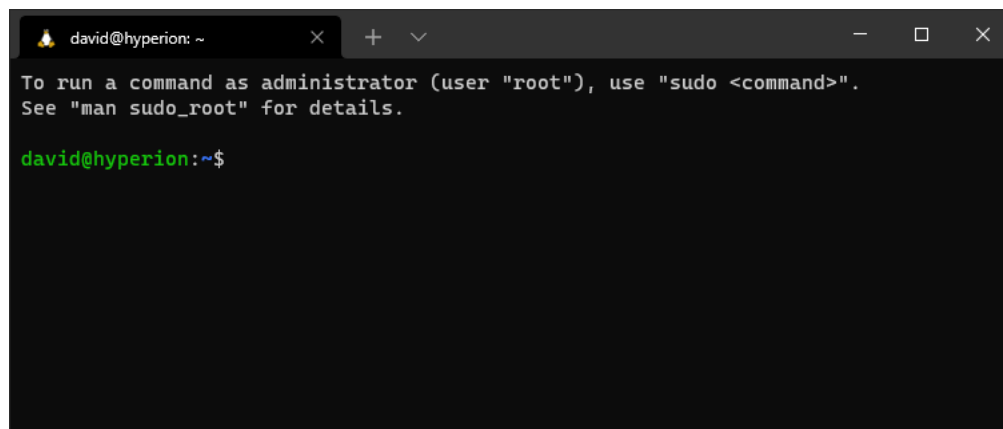
After doing this, click **Save** in the bottom right of the window and close the Windows Terminal application (ensure you have pinned it in your task bar first).



In the **Settings** tab in Windows Terminal, the **Ubuntu-20.04** section has been selected and the **Starting directory** has been changed to `\\wsl$\\Ubuntu-20.04\\home\\david`. Your **Starting directory** won't match this exactly unless your Ubuntu username is also `david`. The **Save** button in the lower right is also highlighted.

Launching Ubuntu 20.04 LTS

Drum roll! Launch Windows Terminal application one more time and if everything has been successful so far you should see a window very similar to the one below.



Windows Terminal on launch after following the above steps.

A couple items to note:

- Windows Terminal should launch directly into the Ubuntu-20.04 environment.
- The command line prompt should read `<Your Ubuntu Username>@<Your device name>:~$`. As shown above, the Ubuntu Username is `david`, and the device name is `hyperion`. Yours will be different. The `~` represents that we are in the *current user's home directory*.

Updating and Upgrading Packages

Windows **does not** manage your Linux installation and will not automatically perform updates for it. To manually do this use this command:

```
sudo apt update && sudo apt upgrade
```

Do this now. Enter your Ubuntu password when prompted, and accept the changes to be made by entering `y` when prompted to continue.

zsh

Bash is the default shell (command interpreter) of Ubuntu, but Z shell is the shell of taste and class so that's what we're going to use. Install it with this command, and accept the changes to be made by entering `y` when prompted to continue:

```
sudo apt install zsh
```

Verify the installation with this command:

```
zsh --version
```

The version number should be 5.8 or greater

Make zsh the default shell with this command. Enter your Linux password when prompted.

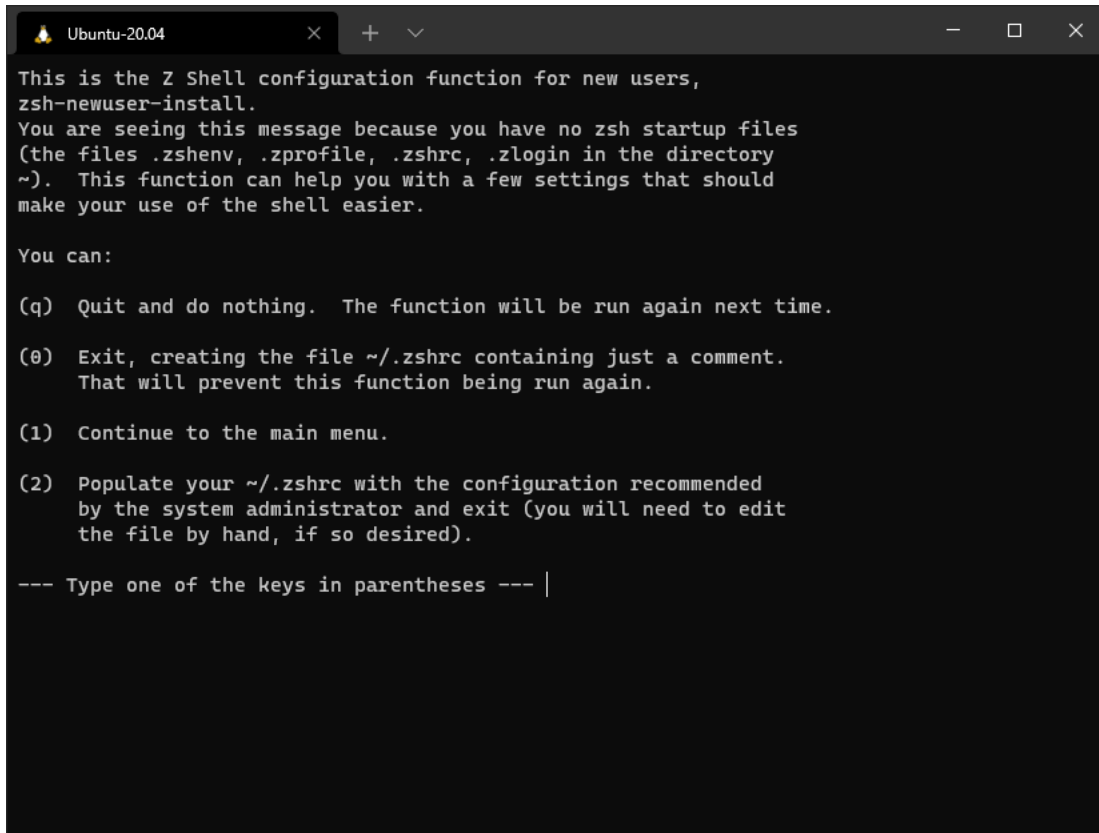
```
chsh -s $(which zsh)
```

Close Ubuntu and the Windows Terminal session with this command:

```
logout
```

Launch the Windows Terminal application again.

Windows Terminal will launch and should present you with this screen:

A terminal window titled 'Ubuntu-20.04' with standard window controls. The text inside reads: 'This is the Z Shell configuration function for new users, zsh-newuser-install. You are seeing this message because you have no zsh startup files (the files .zshenv, .zprofile, .zshrc, .zlogin in the directory ~). This function can help you with a few settings that should make your use of the shell easier. You can: (q) Quit and do nothing. The function will be run again next time. (0) Exit, creating the file ~/.zshrc containing just a comment. That will prevent this function being run again. (1) Continue to the main menu. (2) Populate your ~/.zshrc with the configuration recommended by the system administrator and exit (you will need to edit the file by hand, if so desired). --- Type one of the keys in parentheses --- |'.

```
This is the Z Shell configuration function for new users,
zsh-newuser-install.
You are seeing this message because you have no zsh startup files
(the files .zshenv, .zprofile, .zshrc, .zlogin in the directory
~). This function can help you with a few settings that should
make your use of the shell easier.

You can:

(q) Quit and do nothing. The function will be run again next time.

(0) Exit, creating the file ~/.zshrc containing just a comment.
    That will prevent this function being run again.

(1) Continue to the main menu.

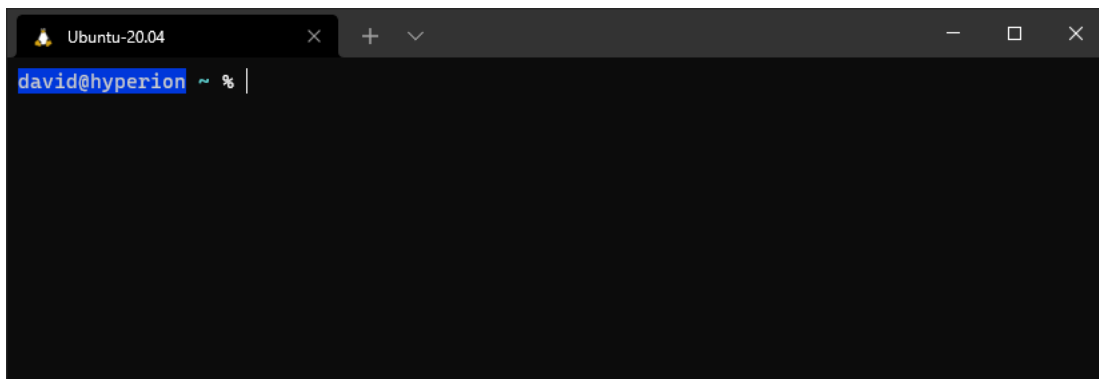
(2) Populate your ~/.zshrc with the configuration recommended
    by the system administrator and exit (you will need to edit
    the file by hand, if so desired).

--- Type one of the keys in parentheses --- |
```

Windows Terminal after launching zsh for the first time

Enter **2** to accept the default configuration.

Your terminal should look different now!

A terminal window titled 'Ubuntu-20.04' with standard window controls. The prompt 'david@hyperion ~ %' is displayed in blue text on a black background.

```
david@hyperion ~ % |
```

Zsh in action.

Let's confirm it worked with this command:

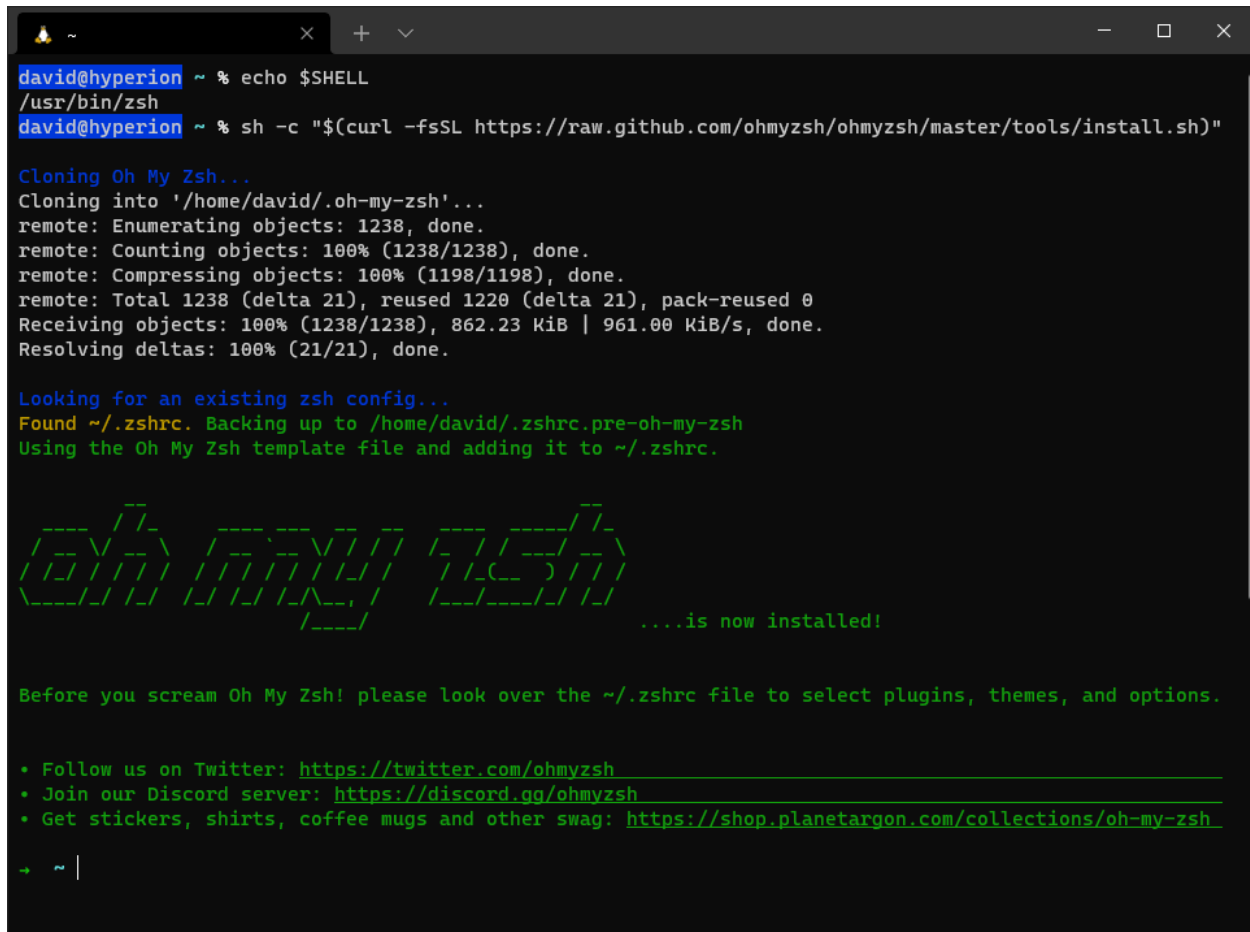
```
echo $SHELL
```

This should return `/usr/bin/zsh`.

Oh My Zsh

We're also going to install Oh My Zsh - an "open-source, community-driven framework for managing your zsh configuration." Use this command:

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

A terminal window with a dark background and light green text. The prompt is 'david@hyperion ~'. The user enters 'echo \$SHELL' and the output is '/usr/bin/zsh'. Then the user enters the command to install Oh My Zsh. The terminal shows the cloning process, progress bars for enumerating, counting, and compressing objects, and the resolution of deltas. It then checks for an existing zsh config, backs up the current .zshrc, and adds the Oh My Zsh template. A large ASCII art logo for 'Oh My Zsh' is displayed. Below it, a message says '....is now installed!'. A warning message follows: 'Before you scream Oh My Zsh! please look over the ~/.zshrc file to select plugins, themes, and options.' At the bottom, there are links to follow on Twitter, join a Discord server, and get merchandise. The prompt returns to '~ |'.

A successful installation of Oh My Zsh

The vanilla configuration of Oh My Zsh is great for our needs, but you can further customize it if you want to - check out [their website](#) and [their documentation](#) to see how.

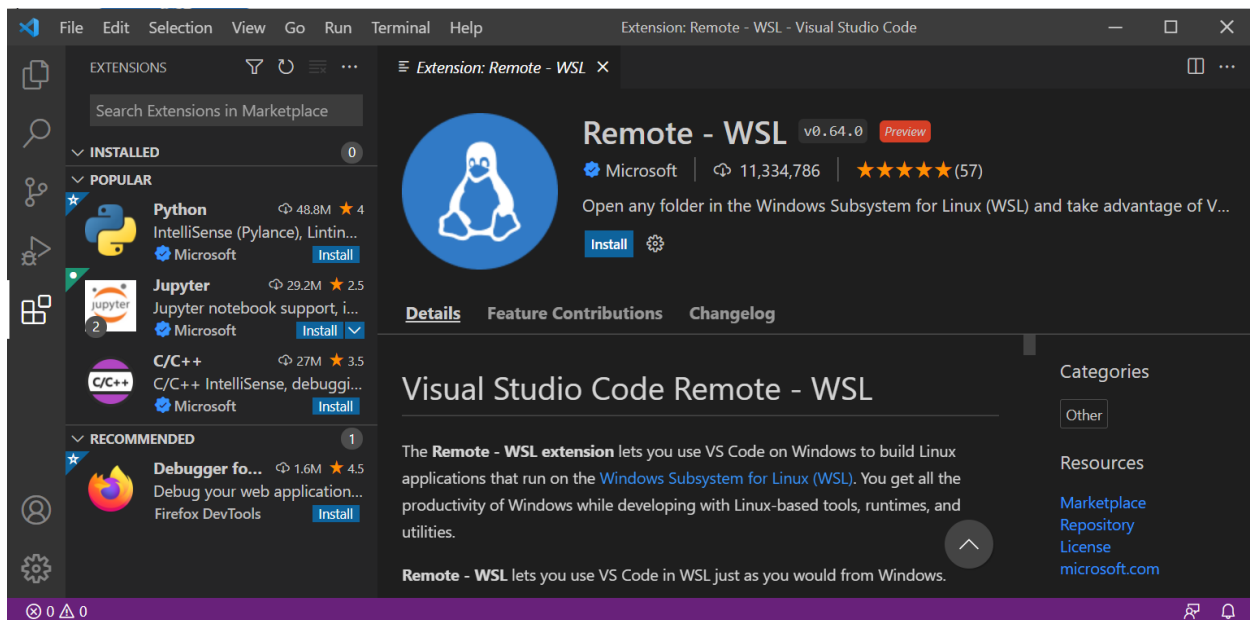
Visual Studio Code

Install Visual Studio Code from [the Visual Studio Code site](#). Install Visual Studio Code on Windows - not in the WSL file system.

! When prompted to **Select Additional Tasks** during installation, ensure that the **Add to PATH (requires shell restart)** option is checked so that you can easily open a folder in WSL using the **code** command.

Install the Remote WSL extension

Once VS Code is installed continue by installing the [Remote - WSL extension](#) in Visual Studio Code. On the page for the extension, click the **Install** button. A dialog box will likely appear informing you that you must have Visual Studio Code installed to install extensions (congrats, you do!). Select **Continue**. If your browser has any additional security prompts asking if you want to use the link to open VS Code, accept them.



The page for the Remote - WSL extension in the VS Code Extension Marketplace.

VS Code will finally open. Select the **Install** button in VS Code. After doing so you should see this appear in the bottom left of your VS Code window:



Microsoft has this to say about running extensions on WSL:

The Remote-WSL extension splits VS Code into a “client-server” architecture, with the client (the user interface) running on your Windows machine and the server (your code, Git, plugins, etc) running remotely.

When running VS Code Remote, selecting the 'Extensions' tab will display a list of extensions split between your local machine and your WSL distribution.

Installing a local extension, like a theme, only needs to be installed once.

Some extensions, like the Python extension or anything that handles things like linting or debugging, must be installed separately on each remote WSL distribution. VS Code will display a warning icon ⚠️, along with a green "Install in WSL" button, if you have an extension locally installed that is not installed on your WSL Remote.

! This concept is **VERY IMPORTANT**. You are now running two separate operating systems simultaneously on your machine. When something is installed in one operating system, the other will not know about it! Let's explore this concept more.

Git in Windows 10

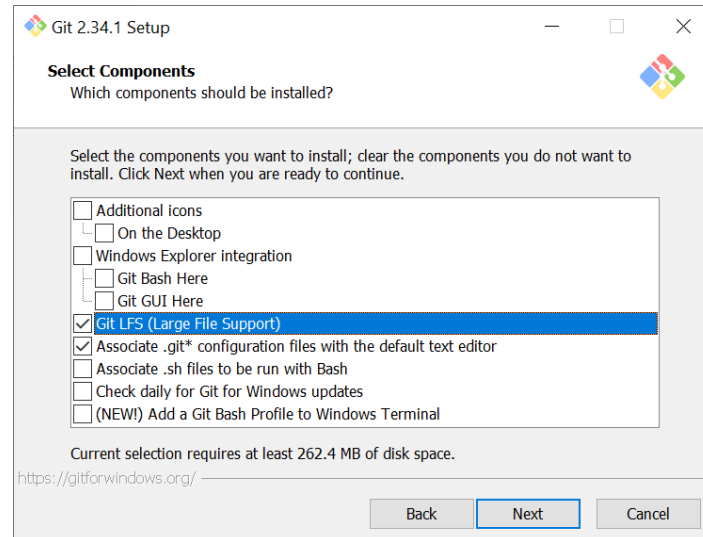
Download Git for Windows 10 from [here](#). Follow the installation instructions below.



While you could use git on Windows as a source control manager, we won't be for our purposes - instead we will be using it mainly for its credential management features. Unless you have a specific need for the features unchecked it's recommended you use only install the checked components below:

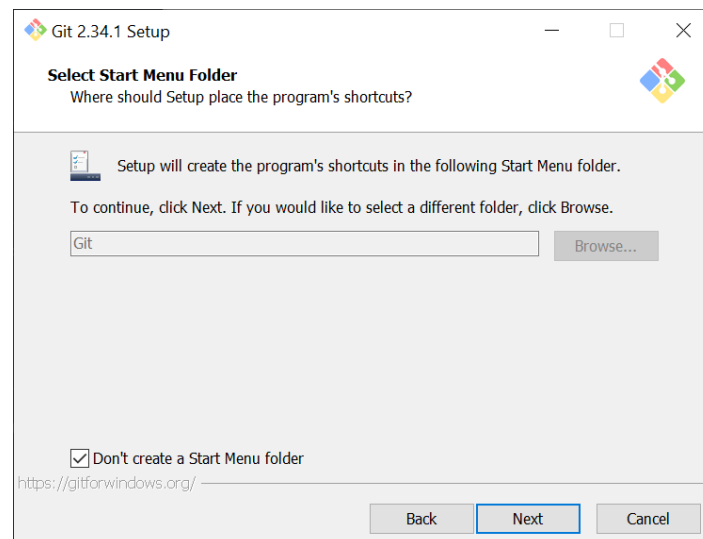
Do not change the install location.

You will be given many prompts on features to install and choices to make while installing Git. All of these may be left as their default, except for the ones below.



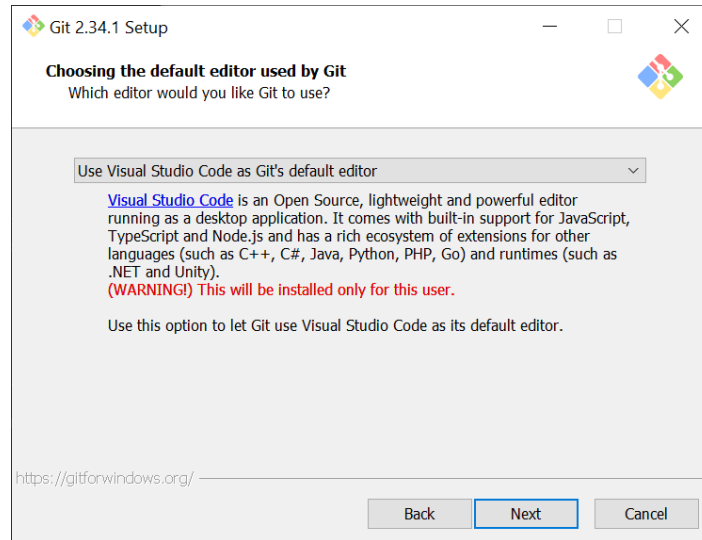
Note we have unchecked some features not required for our purposes that may only be confusing, get in the way, or waste hard drive space.

Again, we won't be using Git features on the Windows side, so having a Start Menu folder would only be confusing and create clutter - select the option for **Don't create a Start Menu folder**.



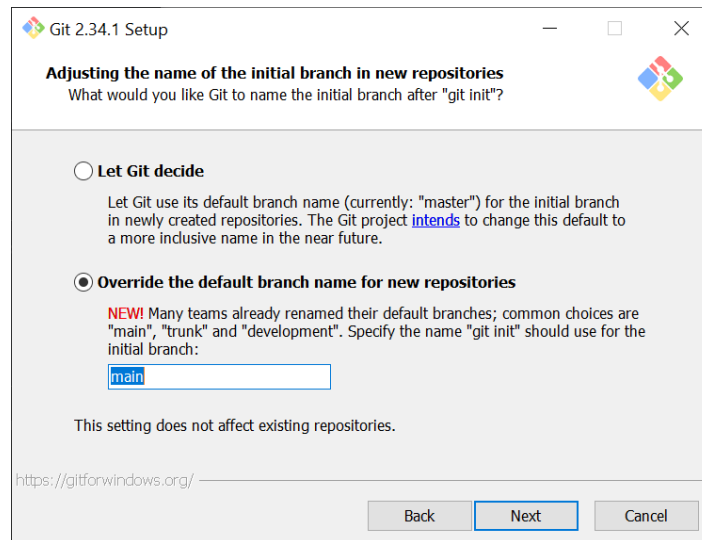
When prompted to Select a Start Menu Folder, we are opting to not create a Start Menu folder.

When prompted, you should select Use Visual Studio Code as Git's default editor.



We are using VS Code as Git's default editor (just in case you ever use Git in Windows for any reason in the future, this will already be set up for you)

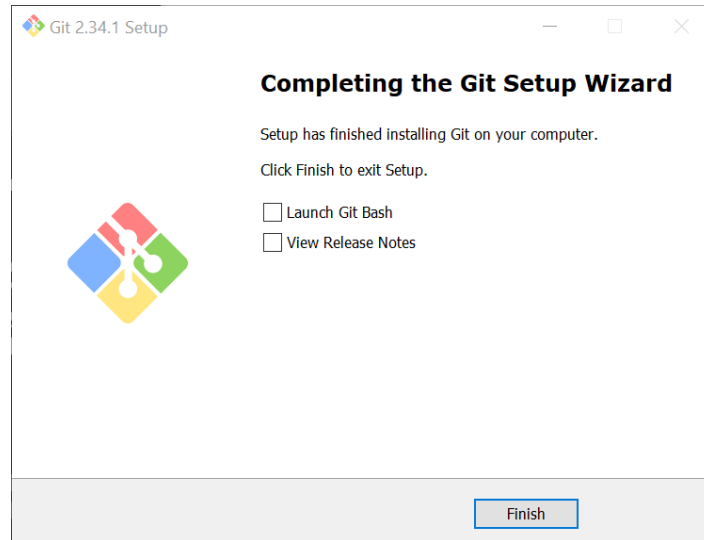
We'll be using the `main` branch for all our work. Go ahead and set this up now. Again this setting won't directly impact us as we use Ubuntu, but if you ever do use Git for Windows in the future you won't have to dig through a bunch of configuration files to change this setting later



`main` as the default branch name is the future

Again, continue on from this point leaving all other settings as their default.

Finally, when installation has completed you can uncheck View Release Notes and hit Finish!



You've completed the installation of Git on Windows, great work. Now on to Ubuntu!

Git in Ubuntu

Close any open Windows Terminal sessions that you have running, and re-launch the Windows Terminal application.

Git comes pre-installed with Ubuntu, but ensure you have the most recent stable version with:

```
sudo add-apt-repository ppa:git-core/ppa
```

You may be prompted for your Ubuntu password. If you are, enter it. When prompted to continue, press **Enter**.

Then enter:

```
sudo apt-get update
```

and

```
sudo apt-get install git
```

Enter **y** when prompted to continue. Once it has been installed, add a user name and email, which will be used to identify your commits, to your git configuration:

```
git config --global user.name "User Name"
```

Replace **User Name** with a name of your choice. Make sure you leave the quotes surrounding your username. Keep the name somewhat professional - this will be used to identify your commits on GitHub.

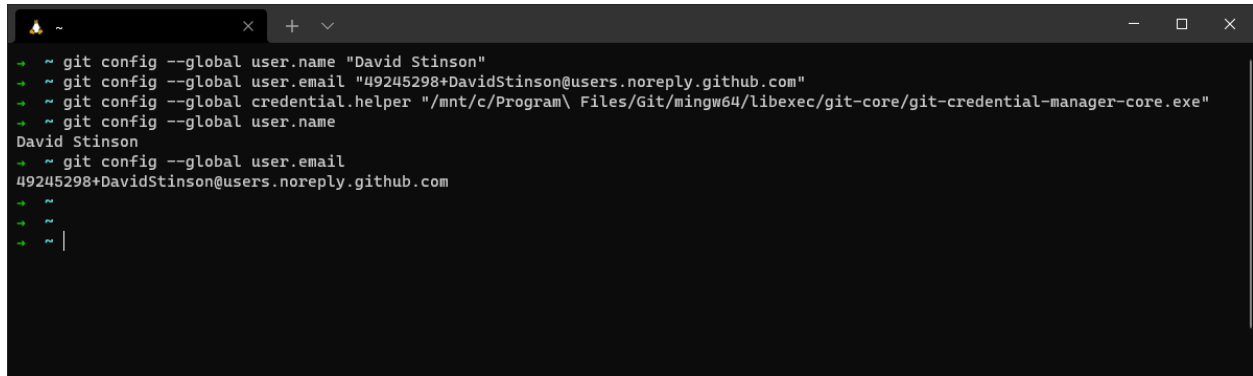
```
git config --global user.email "user@email.com"
```

Replace **user@email.com** with the email address associated with your GitHub account. Make sure you leave the quotes surrounding your email.

All that work we just did to get Git up and running on Windows is about to pay off: Make the Ubuntu installation of Git use the Windows Git Credential Manager Core:

```
git config --global credential.helper "/mnt/c/Program\ Files/Git/mingw64/libexec/git-core/git-credential-manager-core.exe"
```

Now any git operation you perform within Ubuntu will use the credential manager.

A terminal window with a dark background and light text. It shows a series of git config commands being executed. The first command sets the global user name to "David Stinson". The second sets the global user email to "49245298+DavidStinson@users.noreply.github.com". The third sets the global credential helper to the path of the git-credential-manager-core.exe file. The fourth command checks the global user name, returning "David Stinson". The fifth command checks the global user email, returning "49245298+DavidStinson@users.noreply.github.com". The terminal shows the prompt character "~" and the command history on the left.

```
+ ~ git config --global user.name "David Stinson"
+ ~ git config --global user.email "49245298+DavidStinson@users.noreply.github.com"
+ ~ git config --global credential.helper "/mnt/c/Program\ Files/Git/mingw64/libexec/git-core/git-credential-manager-core.exe"
+ ~ git config --global user.name
David Stinson
+ ~ git config --global user.email
49245298+DavidStinson@users.noreply.github.com
+ ~
+ ~
+ ~ |
```

As shown here you can use `git config --global user.name` to check your stored username and `git config --global user.email` to check your stored email.

Set the default branch name to main with this command:

```
git config --global init.defaultBranch main
```

Set the default git editor to VS Code with this command:

```
git config --global core.editor "code --wait"
```

and finally turn off rebasing as the default behavior when making a pull:

```
git config --global pull.rebase false
```

Configuring a Global Git Ignore File



Note: This step is vital to you getting a job after the course. If you do not complete these steps exactly, it will look extremely bad to a future employer when they look over your GitHub repos.

Proper code, utilities, and the use of Git ignore files prevent us from uploading private secrets to the internet.

A global Git ignore file (`.gitignore_global`) will prevent us from uploading private secrets to the internet across all of your projects so that you don't have to worry about making the appropriate entries in every project's Git ignore file.

Use this command to create a `.gitignore_global` file in the user directory:

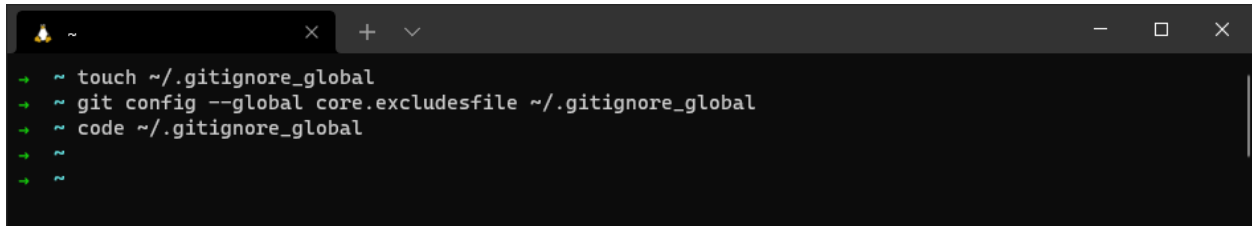
```
touch ~/.gitignore_global
```

Next, configure Git to use this file:

```
git config --global core.excludesfile ~/.gitignore_global
```

Open the new `.gitignore_global` file in VS Code:

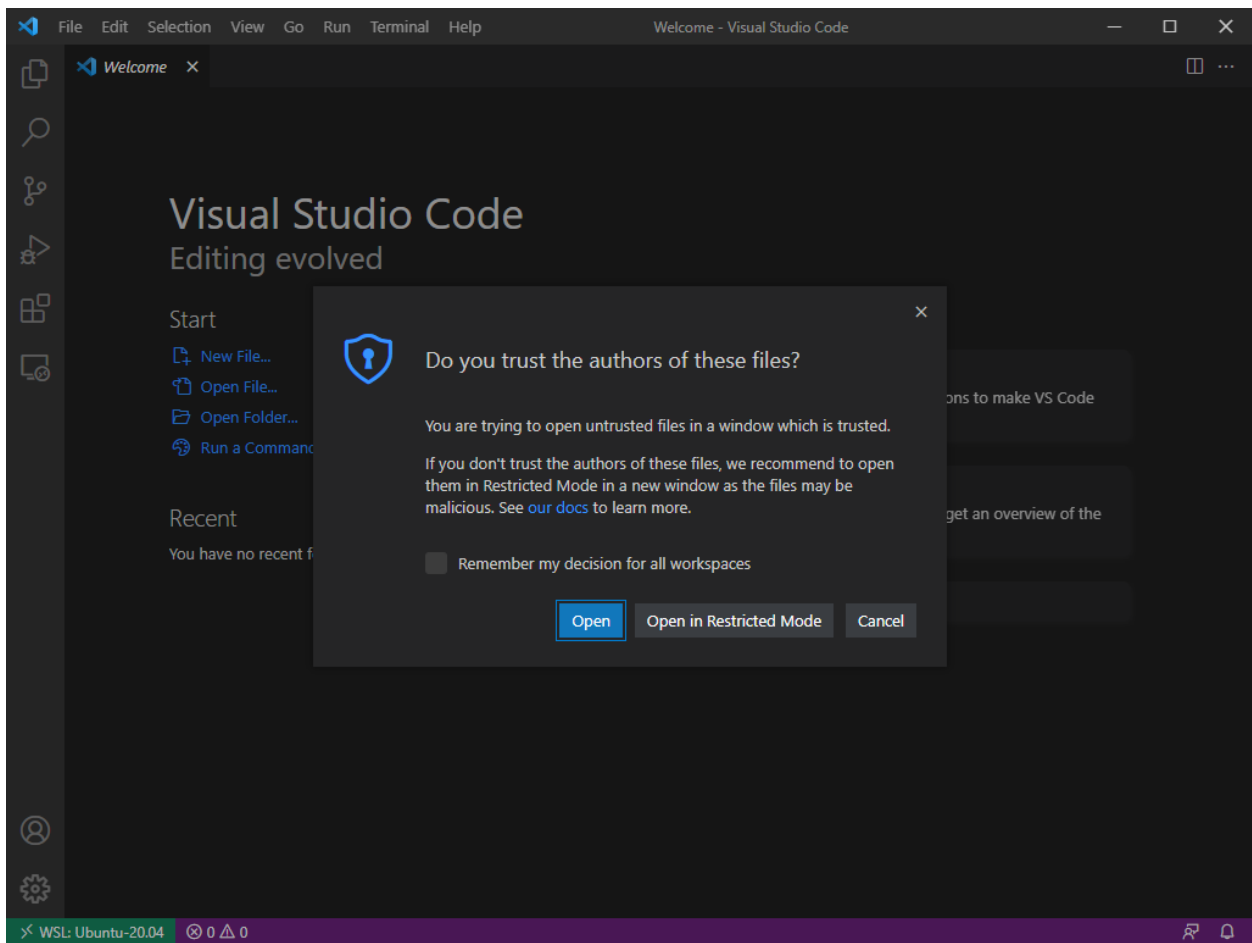
```
code ~/.gitignore_global
```



```
~ touch ~/.gitignore_global
~ git config --global core.excludesfile ~/.gitignore_global
~ code ~/.gitignore_global
~
~
```

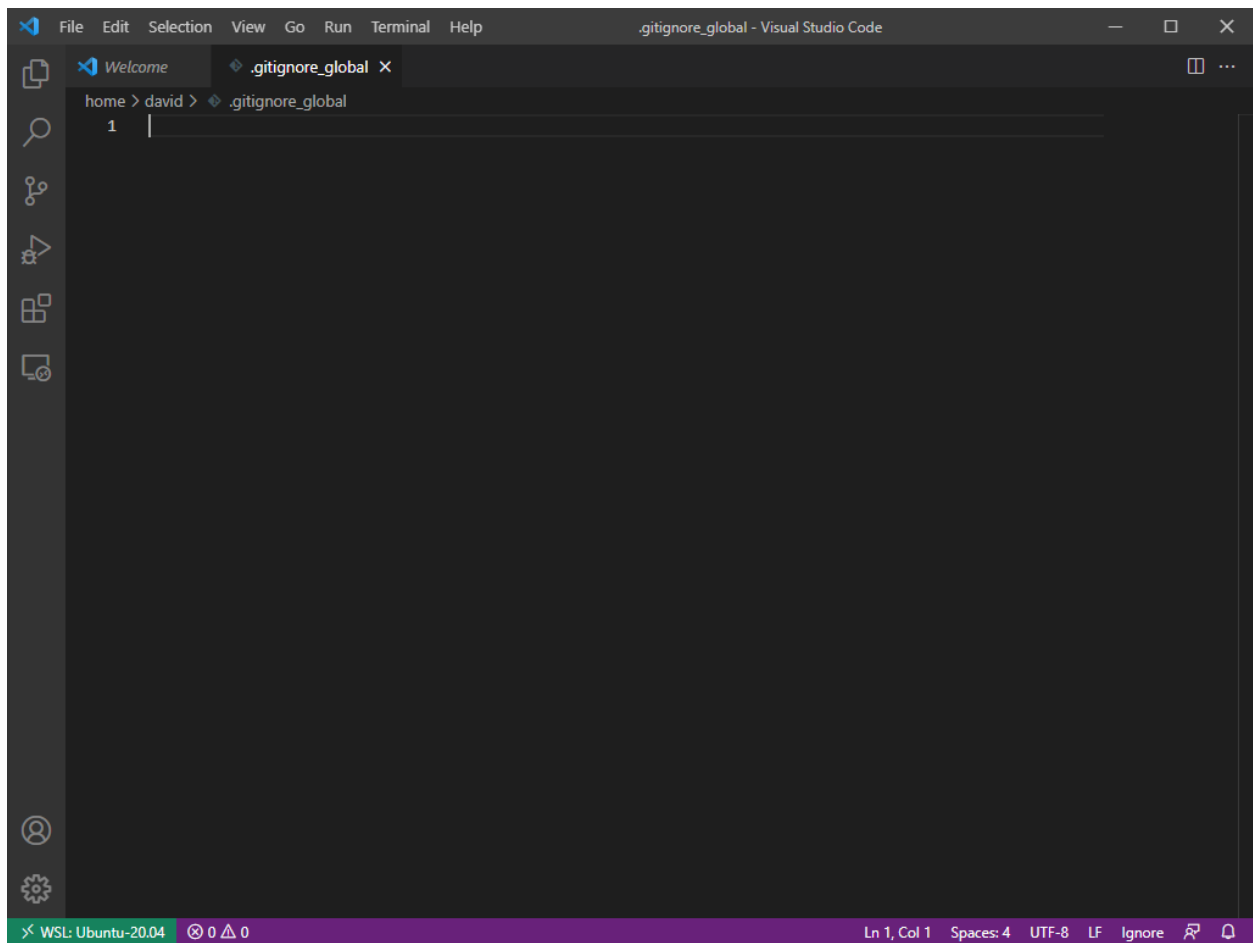
Creating and opening `~/.gitignore_global` in VS Code

This may be your first time launching VS Code to work with an actual file. If so, congrats! You'll be greeted with a prompt asking you if you trust the author of the file. Since we just made this file ourselves, yes, we do.



VS Code, prompting us to trust the author of the workspace we're opening.

We'll finally arrive at a page that should look a lot like this:

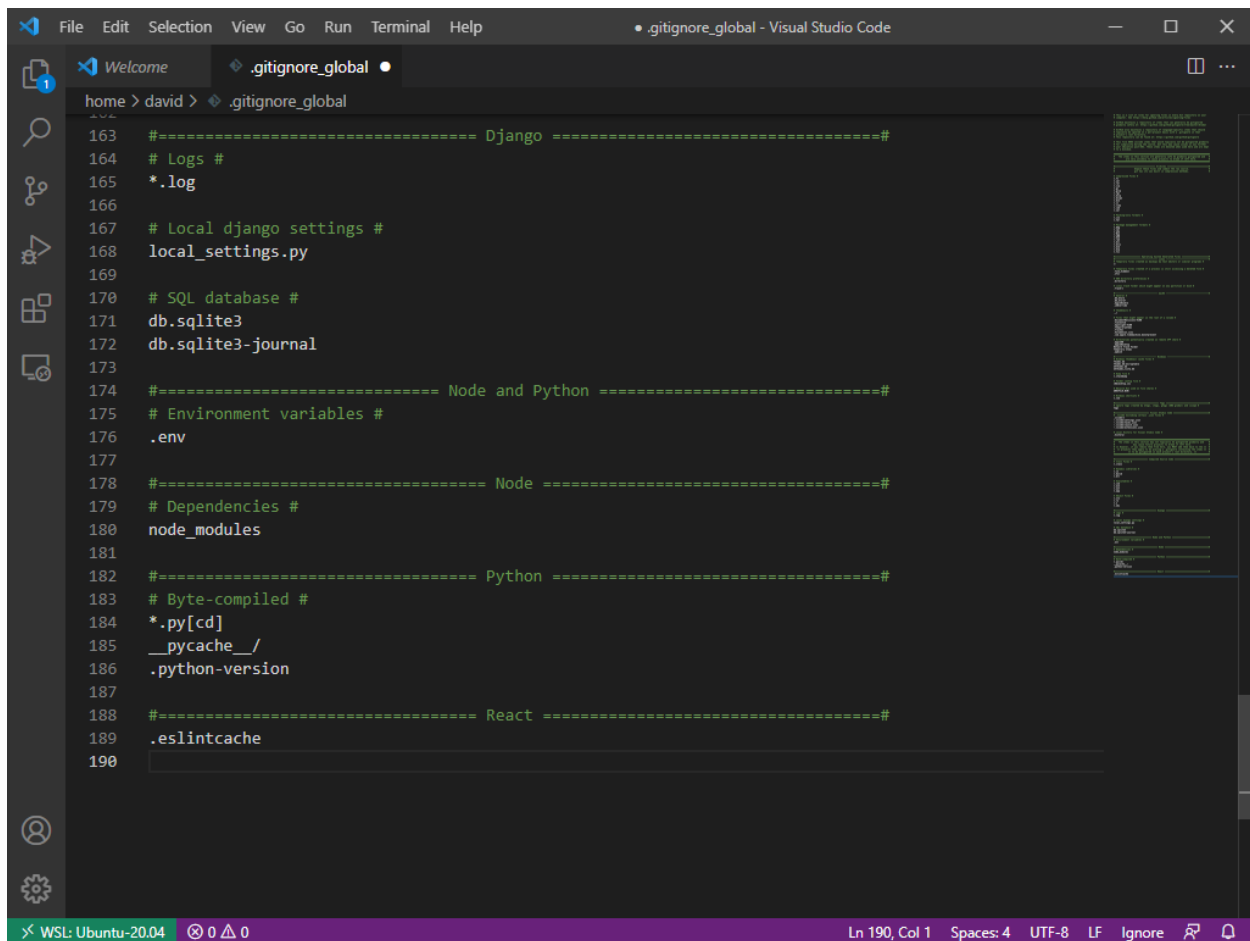


The new `.gitignore_global` file open in VS Code. Note the **WSL: Ubuntu-20.04** icon in green in the lower left corner.

Here is a [.gitignore_global file for you to use](#).

Open the above page and copy all of its contents.

Paste the contents of the file you copied into VS Code.



```
163 #===== Django =====#
164 # Logs #
165 *.log
166
167 # Local django settings #
168 local_settings.py
169
170 # SQL database #
171 db.sqlite3
172 db.sqlite3-journal
173
174 #===== Node and Python =====#
175 # Environment variables #
176 .env
177
178 #===== Node =====#
179 # Dependencies #
180 node_modules
181
182 #===== Python =====#
183 # Byte-compiled #
184 *.py[cd]
185 __pycache__/_
186 .python-version
187
188 #===== React =====#
189 .eslintcache
190
```

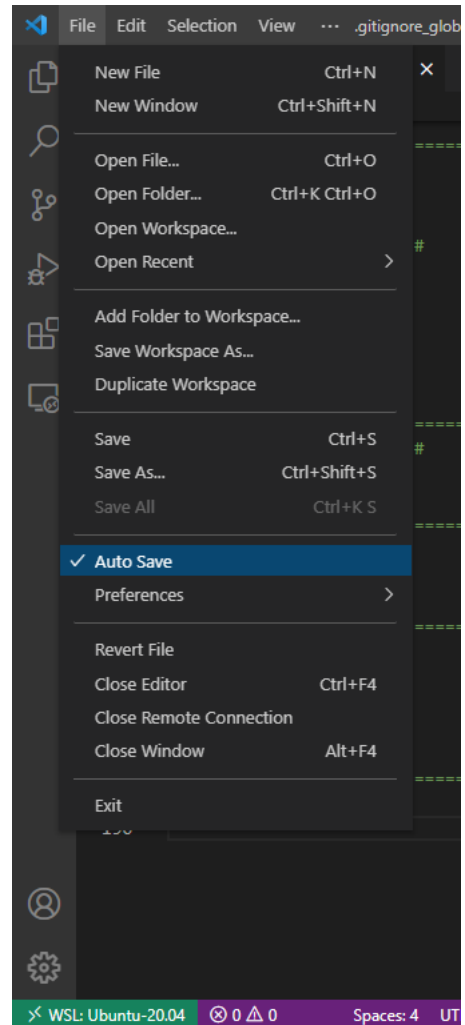
The end of the new `.gitignore_global` file.



This is a great time to turn on **Auto Save** as well! This setting is in the **File** menu - select it, then re-open the **File** menu to ensure that there is a check mark next to the **Auto Save** option.

This should save the file but make sure it gets saved by also manually saving, either by using **Save** in the **File** Menu or by pressing **Ctrl + S**.

You can close VS Code for now.



Auto Save checked in the File menu, indicating that Auto Save is enabled.

Github

[Github](#) provides a way to host Git repos in the cloud. It enables collaboration and is wildly popular. If you have not already created an account there, do so now.

GitHub CLI

We'll be using the GitHub command line utility to perform some actions on GitHub as well. Install it with this command:

```
curl -fsSL https://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo gpg --dearmor -o /usr/share/keyrings/githubcli-archive-keyr
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/githubcli-archive-keyring.gpg] https://cli.github.com/packages s
sudo apt update
sudo apt install gh
```

Handling Errors 💔

If you get this error:

```
gpg: failed to start the dirmngr '/usr/bin/dirmngr': No such file or directory
```


Try installing the `dirmngr` package with this command:

```
sudo apt-get install dirmngr
```

and repeat the install steps above.

GitHub CLI Login

Login with this command:

```
gh auth login
```

You will be prompted to login to a github.com account or a GitHub Enterprise account. Select the **github.com** option.

The Second prompt will ask you to choose whether you want to use HTTPS or SSH. Select the **HTTPS** option.

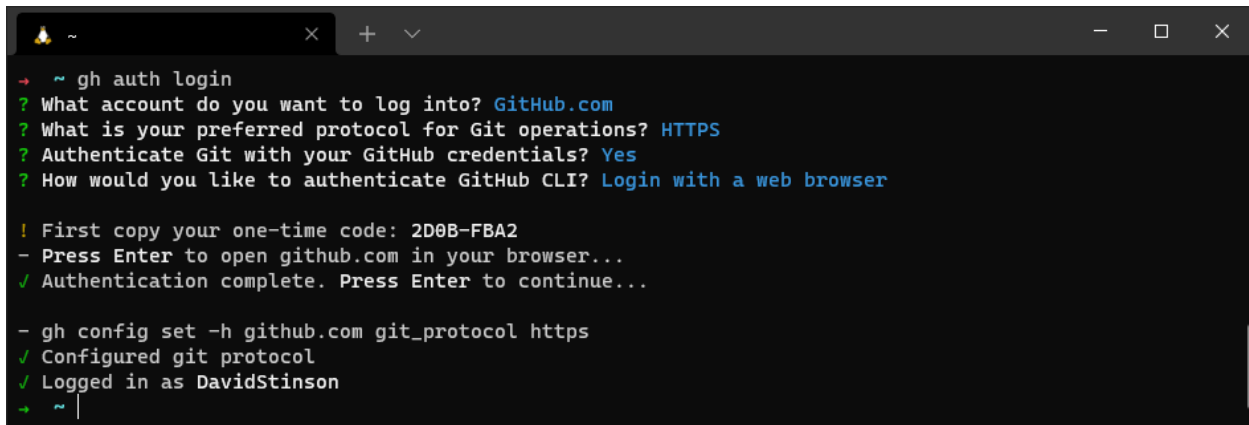


A third prompt may appear asking you if you want to authenticate with your GitHub credentials. If you are given the option to Authenticate Git with your GitHub credentials, do so - this allows you to skip the next step: *Generating a GitHub Personal Access Token*.

The fourth prompt will appear asking how you would like to authenticate. Select the **Login with a web browser** option.

You will then be prompted to copy the one time code from the terminal. Do this now. Then press the `Enter` key to open the github.com login page in your browser.

Complete the login process, authorize the GitHub CLI, and return to your terminal. If you were successful, you will receive a message that says authentication is complete. Press `Enter`.



The completed `gh auth login` process



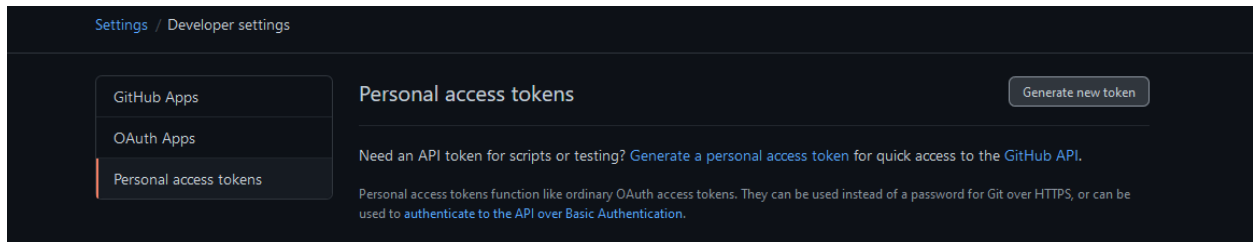
If you were given the option to Authenticate Git with your GitHub credentials, and said yes - you may skip the next step: *Generating a GitHub Personal Access Token*.

Generating a GitHub Personal Access Token

GitHub deprecated the use of password authentication via the command line on August 13, 2021, as detailed in [this GitHub blog post](#). This means, we must authenticate using GitHub's preferred authentication method: Personal Access Tokens (PATs).

First, visit <https://github.com> and ensure that you are signed in. Also, ensure that you have [verified your email address](#) with GitHub. After doing so, navigate to <https://github.com/settings/tokens>.

On the **Personal access tokens** page, click **Generate new token**.



The **Personal access tokens** page in **Developer Settings**. The **Generate new token** button is highlighted.

You will be taken to a page prompting you to create a **new personal access token**. Fill the **Note** field with the name of the device you are using the token with. Also set an expiration date. We recommend setting a custom expiration date for one year from today's date, but you may choose to set it to never expire. Select all the **repo** scopes - ensure your selections match what is in the screenshot below. When you have done so, click the **Generate token** button.

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

Main Desktop

What's this token for?

Expiration *

Custom... 01 / 17 / 2023

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo:deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input checked="" type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_runners:enterprise	Manage enterprise runners and runner-groups
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input type="checkbox"/> admin:gpg_key	Full control of public user GPG keys (Developer Preview)
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys

Generate token **Cancel**


You will be taken back to the **Personal access tokens** page, and the token you just created will be visible:

Personal access tokens

Generate new token **Revoke all**

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ 8f155d0597592f85e6d6367789a48cfac7b86bd0 

Delete

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Click the clipboard button to copy the newly created token.



You will only see the token on this page **ONCE**. You **MUST** copy it now and paste it in a secure and private place (preferably in a password manager). Treat this PAT as you would a password! **The PAT will be used in place of a password to interact with GitHub on the command line!**



Using multiple machines? It is best practice to create a new PAT for each device requiring command-line access to GitHub - this way, if you need to revoke access to any single device, none of your other devices are impacted.

Place the token in a secure place!

So much done! Just a little more now...



Although all the files you have stored in Windows are available on the Ubuntu side in `/mnt/c`, accessing them comes at a significant performance cost. Therefore the code you write will be stored in the Ubuntu storage space.



The Ubuntu storage space can be accessed from Windows at: `\\wsl$Ubuntu-20.04`.

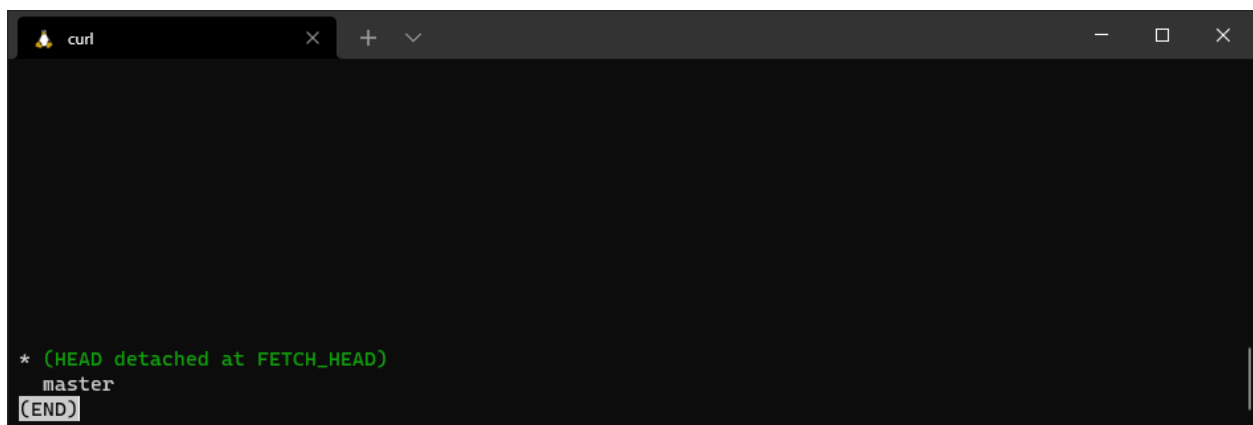
You shouldn't have to bring files from Windows to Ubuntu or from Ubuntu to Windows often, but the above information may be useful to you in certain circumstances.

Node.js

Use this command to install `nvm` which we will use to install node. `nvm` stands for [Node Version Manager](#), and can be used to rapidly hot-swap between different versions of Node.js:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
```

You may see this prompt part way through the install process:



```
curl
```

```
* (HEAD detached at FETCH_HEAD)
master
(END)
```

If you do, just hit `q` - that will exit this screen and return you to the below install process. If not, continue on!

```
~ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 14926  100 14926    0     0  69423      0 --:--:-- --:--:-- --:--:-- 69747
=> Downloading nvm from git to '/home/david/.nvm'
=> Cloning into '/home/david/.nvm'...
remote: Enumerating objects: 348, done.
remote: Counting objects: 100% (348/348), done.
remote: Compressing objects: 100% (297/297), done.
remote: Total 348 (delta 39), reused 157 (delta 26), pack-reused 0
Receiving objects: 100% (348/348), 200.30 KiB | 359.00 KiB/s, done.
Resolving deltas: 100% (39/39), done.
=> Compressing and cleaning up git repository

=> nvm source string already in /home/david/.bashrc
=> bash_completion source string already in /home/david/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
-> ~ |
```

The completed installation of nvm.

Restart the Windows Terminal application now.

After starting up the Windows Terminal again run the `nvm --version` command . If you do not get a version number, check out the **Handling Errors** ❤️ subsection below, otherwise, continue.

Use nvm to install node version 16 with this command:

```
nvm install 16
```

```
~ nvm install 16
Downloading and installing node v16.13.2...
Downloading https://nodejs.org/dist/v16.13.2/node-v16.13.2-linux-x64.tar.xz...
##### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v16.13.2 (npm v8.1.2)
-> ~ |
```

A successful install of node v16.13.2

With node installed install nodemon globally with this command:

```
npm i -g nodemon
```

```
→ ~ npm i -g nodemon

added 116 packages, and audited 117 packages in 3s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.1.2 -> 8.3.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.3.1
npm notice Run npm install -g npm@8.3.1 to update!
npm notice
```

nodemon successfully installed!

Disregard any prompts to update `npm` for now.

Handling Errors 💔

command not found: nvm Error

Copy this command block and run it in the terminal, which will point to the nvm directory in your `~/.zshrc` file:

```
cat << EOF >> ~/.zshrc

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
EOF
```

Restart your terminal. you should now be able to run the `nvm --version` command and get a version number in response. If you do not alert an instructor.

Heroku

We will deploy full stack applications to Heroku. Install it with the below command. You may be prompted for your Ubuntu password.

```
curl https://cli-assets.heroku.com/install.sh | sh
```

```
~ curl https://cli-assets.heroku.com/install.sh | sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 1894  100 1894    0     0  8648      0 --:--:-- --:--:-- --:--:-- 8609
This script requires superuser access.
You will be prompted for your password by sudo.
[sudo] password for david:
Installing CLI from https://cli-assets.heroku.com/heroku-linux-x64.tar.xz
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 18.7M  100 18.7M    0     0 8431k      0 0:00:02 0:00:02 --:--:-- 8427k
v12.21.0
heroku installed to /usr/local/bin/heroku
> Warning: Our terms of service have changed: https://dashboard.heroku.com/terms-of-service
heroku/7.59.0 linux-x64 node-v12.21.0
~ |
```

The Heroku install process completing successfully.

Being More Productive By Using the Keyboard Instead of the Mouse in Windows

Launching Apps with Search

Developers avoid using the mouse whenever possible because they are more productive when their hands are on the keyboard. Windows lets us do this by opening applications using *Search* instead of the mouse by:

1. Pressing **Windows Key** to open *Search*
2. Start typing the name of the app until the app is highlighted
3. Press **Enter** to open the app!



PowerToys includes a macOS-like launcher as well that you can access by pressing **Alt + Space**

Switching Between Applications

Quickly switch between running applications by pressing **Alt + Tab**.



Note that it's best to minimize how many windows/applications you have open when developing to make switching between applications quicker and minimize distractions to the job at hand.

Taking Screenshots

You'll periodically need to take screenshots. Use **Windows Key + Shift + S** to take a screenshot of an area of your screen.

The screenshot you take will be placed on your clipboard, and you can paste it wherever you would like to use it.

Uploading Screenshots and Images to [imgur](https://imgur.com)

Often you will need to share images with others or use them in your applications, notes, readme files, etc. Unfortunately, if an image exists only on your computer, you lose the ability to use it anywhere but on your computer. To get around this, we can upload images to a cloud service like [imgur](https://imgur.com), one of the most popular image hosting services on the internet.

Feel free to open an account there, so you can keep track of what you upload, but you can also use their service without an account.

OH WOW YOU DID IT!

You now are set up to start developing in Linux on Windows! Be very proud of yourself, that was quite the process!

Level Up

[Mac Keyboard Shortcuts Every Dev Should Know](#) - Most of these commands are also available in Windows as well - just replace instances of `Command` with `Control`.

A Password Manager

While this is optional, we recommend using a password manager to help keep track of the various accounts and logins you will be creating throughout the course and in the rest of your digital life. [Bitwarden](#) is free, open-source, and provides a great user experience, but if you're using a different one (or not using one at all), that is no problem.