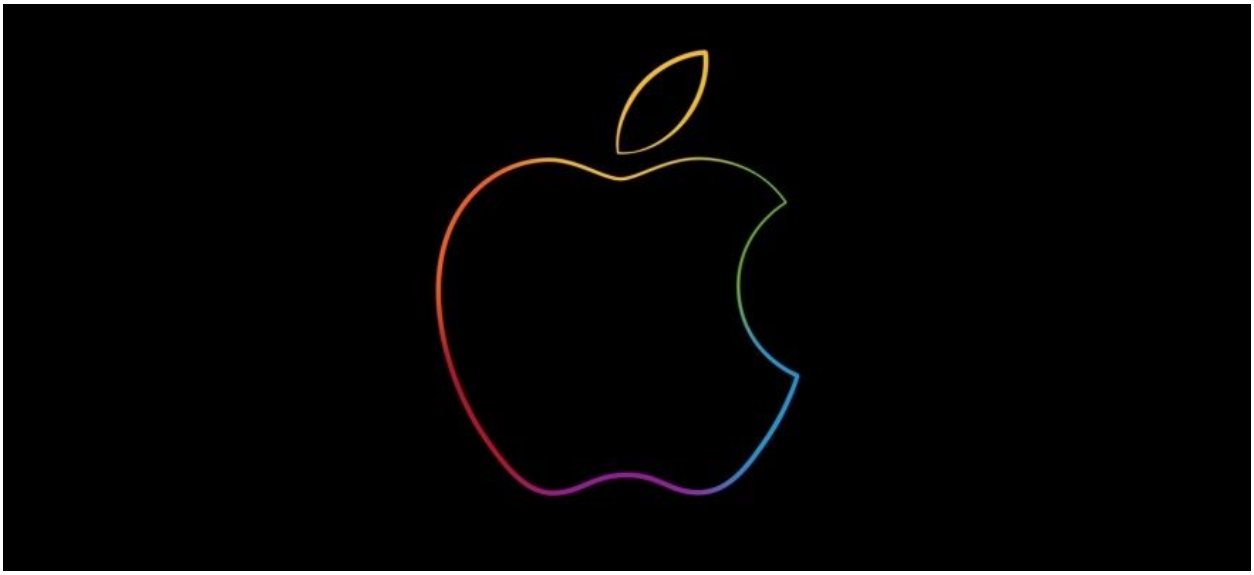
macOS 12 Monterey for Apple Silicon Macs



⚠️ This guide is for Macs with the Apple Silicon chips using a public release of macOS 12 Monterey. As all Apple Silicon Macs are able to run macOS 12 Monterey we very highly recommend ensuring that you have upgraded to this version before proceeding.

🚨 ***This guide is NOT for Macs with Intel processors!***

Contents

Checking Your Processor Type in Monterey

Check your processor type in macOS by selecting the logo in the top left of your screen and select the **About This Mac** option. Macs with the Apple Silicon will have a chip type of

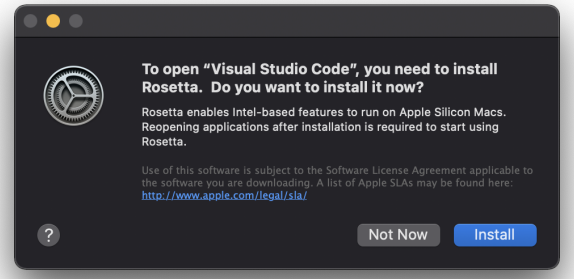
Apple, like in the screenshot to the right.

!! If your chip type is Intel you need to switch to the macOS 12 Monterey and Earlier for Intel Macs install guide now.

A Mac running macOS Monterey with an Apple M1 chip.

Running Applications Built for Intel Macs on Macs Equipped with Apple Silicon Processors

The first time you run an application that was uses Intel-based features you will see something like the prompt to the right telling you to install Rosetta. When this happens, select **Install**, and re-launch the application.



The prompt for installing Rosetta.

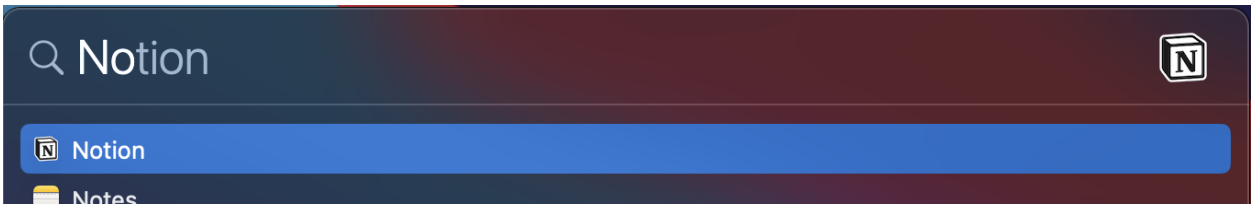
Notion

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 **For Macs with Apple M1** option when it is given. Install it by opening the downloaded `.dmg` file and drag the **Notion** application into the `Applications` directory.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/714d7367-96fa-43e3-9678-6c7a447795cd/Screen_Recording_2021-02-02_at_10.35.13_AM.mov

Moving the Notion application into the Applications directory.

Open the Notion application from your `Applications` directory now. To do so, press `Command ⌘ + Space` to launch **Spotlight** and type **Notion**, then select the Notion application by pressing `return` when it appears, as shown in the screenshot below.

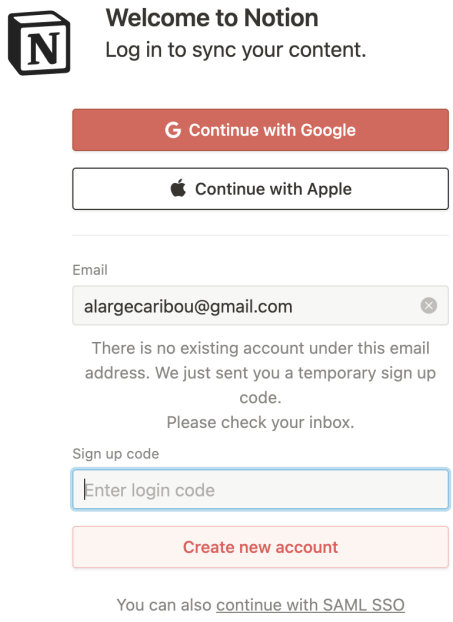


Launching the Notion application using Spotlight. Get used to seeing this often; it's the fastest way to start applications on the mac!

When the application launches, 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.



The Notion create an account page - a non-OAuth account is being created.

🚨 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!

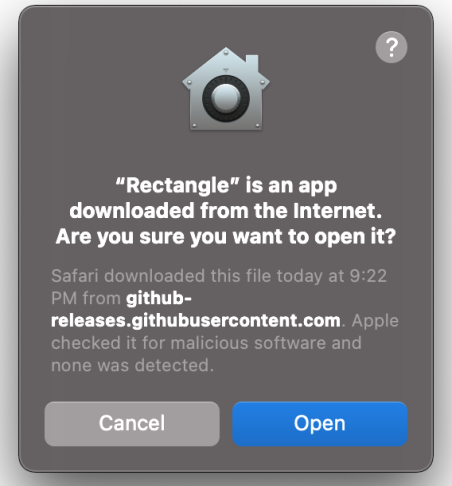
Rectangle

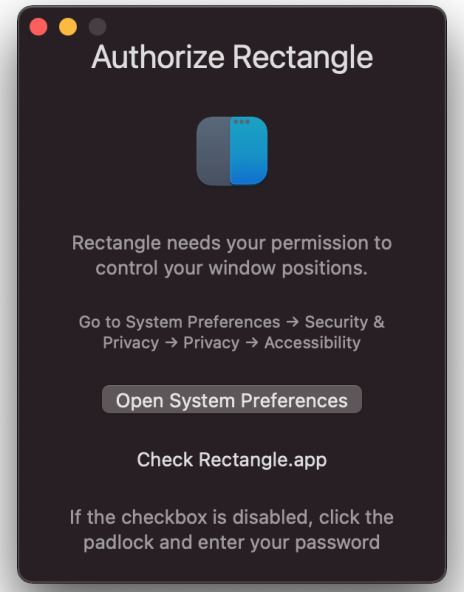
💡 We highly recommend against using the built-in macOS window management features, unless you have extensive experience using them and are already comfortable with manipulating windows without using a mouse. This is where Rectangle comes in.

Rectangle is an open-source window management tool that offers extensive customization. No more fiddling with window position in macOS! Install Rectangle from **here**. Read more on the **project's GitHub**. Once it is installed by moving it into the `Applications` directory, launch it with **Spotlight** (with `Command ⌘ + Space`).

The first time you launch it, you will be prompted to confirm that you would like to run the program because it is downloaded from the internet. Give it permission to open by clicking the **Open** button.

You'll also be prompted to authorize Rectangle to control your window positions. Allow this by clicking the **Open System Preferences** button in the dialog box.

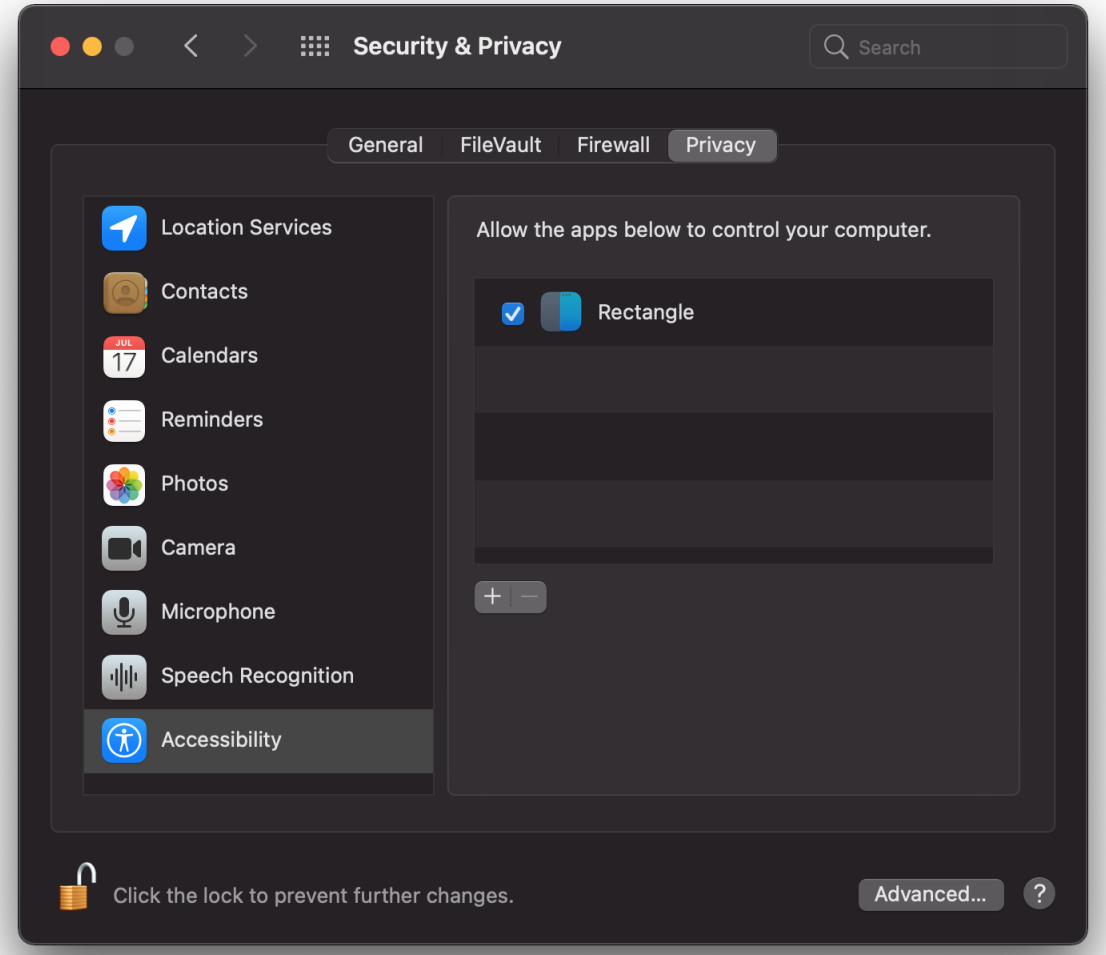




Rectangle, prompting for accessibility permission.

macOS prompting for confirmation to launch the Rectangle application.

System Preferences will open and take you to the **Security & Privacy** pane. Select the checkbox next to the **Rectangle** app. If the checkbox is disabled, click the padlock in the bottom left corner of the window and enter your user account password. This will enable you to make changes. After unlocking the settings, you should be able to select the checkbox next to **Rectangle**, as shown below.



The Security and Privacy pane, after Rectangle has been given the appropriate system permissions.

Immediately after you have given Rectangle the appropriate permissions, you will be asked which default shortcuts and behavior that you prefer. Opt for the **Recommended** control scheme (the Spectacle scheme conflicts with multiple programs we use in class - that's no good).

Try getting familiar with Rectangle today and the rest of this week. Here is a cheat sheet for all the different keyboard shortcuts that are pre-configured. The most useful commands for your use at first will likely be `Ctrl ^ + Option ⌥ + Left Arrow ←` to move windows to the left half of the screen, `Ctrl ^ + Option ⌥ + Right Arrow →` to move windows to the right half of the screen, and `Ctrl ^ + Option ⌥ + Return` to make maximize windows.

The default keyboard shortcuts for the window layouts provided by Rectangle.

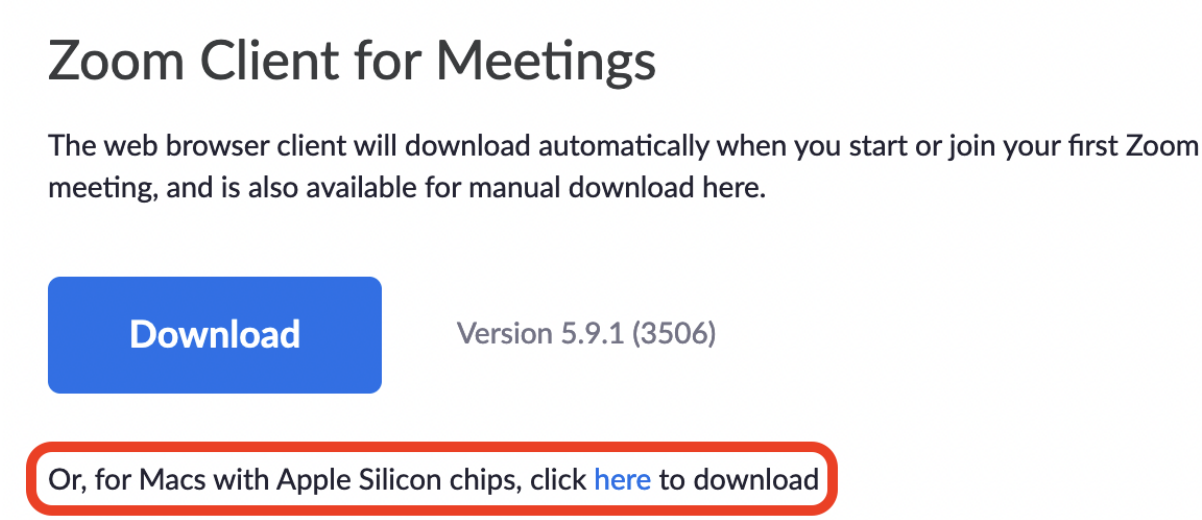
Slack

We will be using slack to communicate throughout the course. Download Slack **here**.

Remember to drag the Slack app into the Applications folder when you open the downloaded archive.

Zoom

If you haven't already, **download the Zoom client** and install it. Even if you have installed Zoom before, double check that you have installed the version for Apple Silicon chips, highlighted below.



The Zoom client requires certain permissions to access your camera, microphone, and screen - let's enable those now.

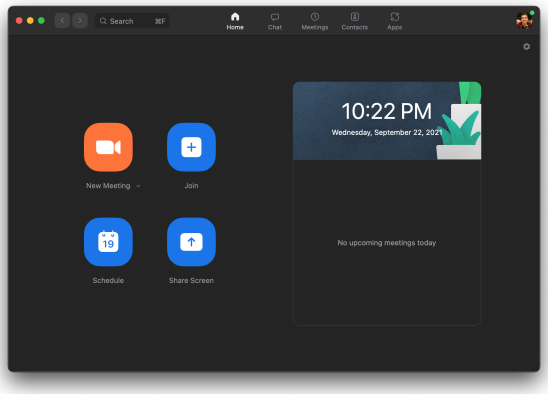
After it is installed, open the program and login to a zoom account. You'll need to create an account if you don't already have one by clicking the Sign Up link that is on the Sign In page.

Upon signing in, you should immediately be prompted to allow access to the microphone. Grant this permission by selecting **OK**.

Zoom, prompting for access to your Mac's microphone.

⚠️ If a prompt didn't appear, that's ok, continue on!

You should arrive at the main Zoom window. Click on the **New Meeting** button.



The main Zoom window.

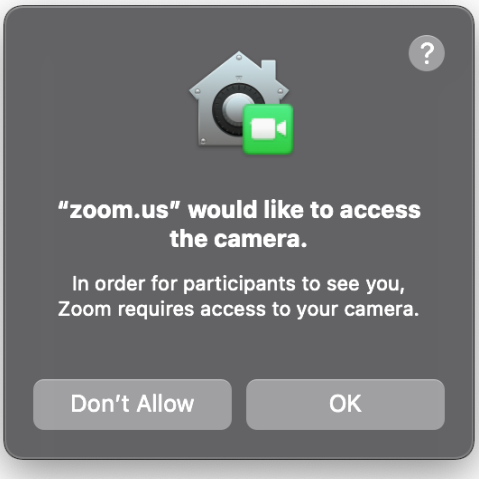
This will launch a new meeting with you as the only participant. You'll be prompted to allow access to your Mac's webcam. Do so by selecting **OK**.

> ⚠️ If a prompt didn't appear, that's ok, continue on!

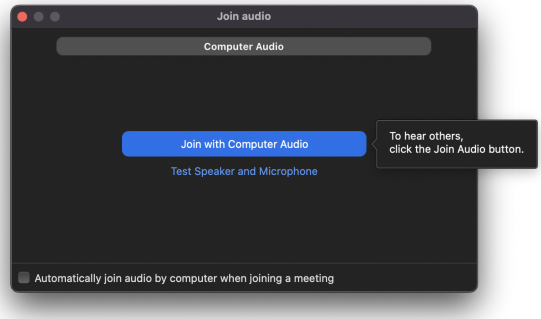
Next, you'll likely be asked to join your audio to the zoom room. Do this. If you're not prompted to join your audio to the zoom room that is ok! Share your screen by clicking the green **Share Screen** button at the bottom of the window.

You'll be shown a screen that looks similar to the one on the right. Select **Desktop 1** and click the **Share** button in the bottom right of the window.

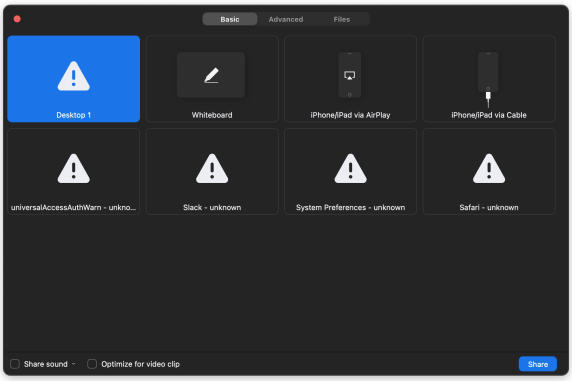
After clicking share, you'll see the below dialog box appear. Select **Open System Preferences**.



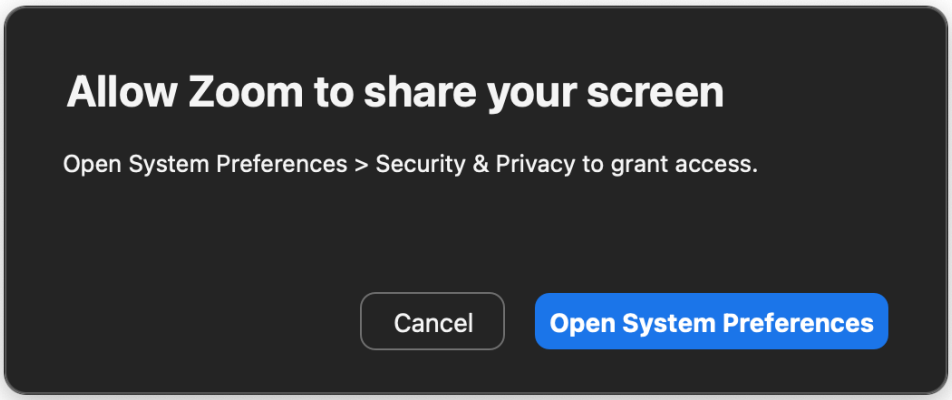
Zoom, prompting for access to your Mac's camera



The Join audio dialog box. When prompted, select Join with Computer Audio.

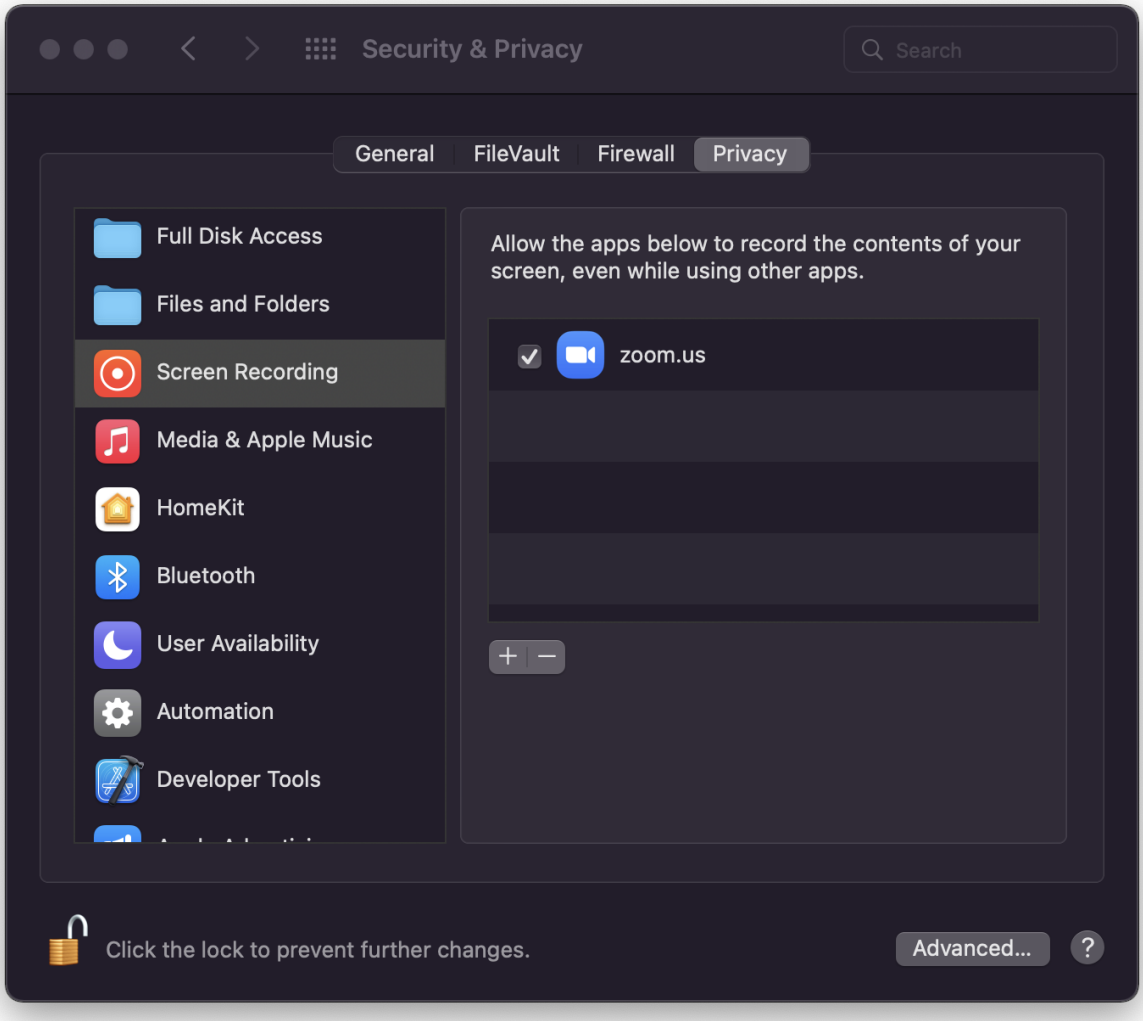


The Screen Share pane. Get used to seeing this quite often!



The dialog box asking you to allow Zoom to share your screen.

You'll be taken to the **Security & Privacy** pane in the **System Preferences** application. The **Screen Recording** subsection should be highlighted. Select the checkbox next to the **zoom.us** app. If the checkbox is disabled, click the padlock in the bottom left corner of the window and enter your user account password. This will enable you to make changes. After unlocking the settings, you should be able to select the checkbox next to **zoom.us**, as shown below.



The Security and Privacy pane, after Zoom has been given the appropriate system permissions.

Immediately after you have given Zoom the appropriate permissions, you will receive a notification saying that Zoom will not be able to record the contents of your screen until it is quit. Select the **Quit & Reopen** option. You'll also be asked to end the current Zoom meeting. Do so.

A Note On Copying Commands

When possible, ***please copy the commands from this page***. Most of the commands here will be used 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

To copy text from code blocks in Notion click and drag to highlight *only the text* that you intend to copy then use `Command ⌘ + C` to copy it and `Command ⌘ + V` to paste it.

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

Do not do this!!

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

zsh

We will use zsh as the default shell. Open the Terminal application from your **Applications** directory now. To do so, press `Command ⌘ + Space` to launch **Spotlight** and type **Terminal**, then Select the Terminal application by pressing `return` when it appears. After the Terminal application starts, check if zsh is already your default shell with this command:

```
echo $SHELL
```

🥳 If this command outputs `/bin/zsh` please skip to the **Xcode Command Line Tools** section below.

If zsh is Not Your Mac's Default Shell

If the `echo $SHELL` command outputs anything other than `/bin/zsh`, you will need to make zsh your default shell with this command:

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

End your terminal session by closing the terminal window.

Open a new terminal window. You may be prompted to run a configuration setup for new users. If you are, populate the `~/.zshrc` with the configuration recommended by the system administrator.

After doing that, rerun this command:

```
echo $SHELL
```

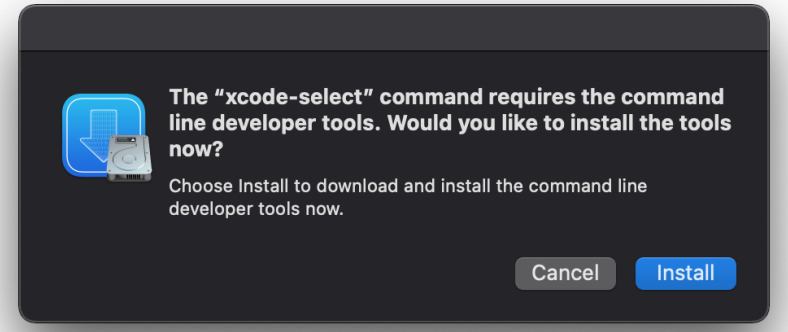
It should now output `/bin/zsh`.

Xcode Command Line Tools

We do not use Xcode in class, but some other command line applications that we use do require some Xcode libraries. Install them with this command:

```
xcode-select --install
```

You should be prompted with the below dialog box. Select **Install**.



This will begin a rather large (~1.3 GB) download. Please wait for it to complete before moving on.

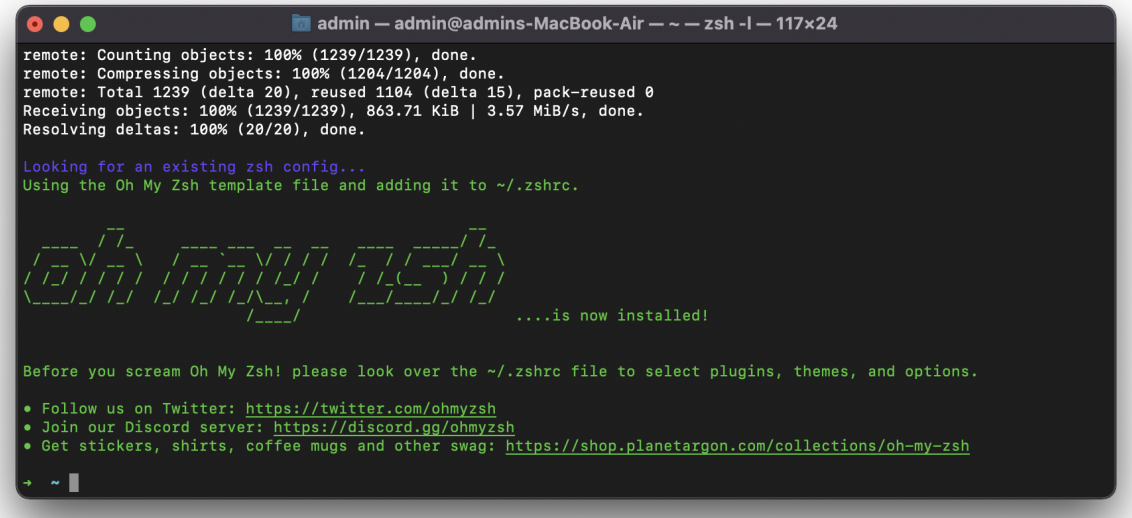
> ⚠️ Under certain circumstances, you may be required to install Xcode from the Mac App Store in order to get command line tools - if you are still getting errors that the command line tools are not installed even after running the above command this may be required.
> You can get Xcode from **here**.
>
> Under even more rare circumstances, this may not solve your issue. Pair with an instructor if this happens to you.

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.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

Upon successfully installing Oh My Zsh, you should be greeted with the following screen:

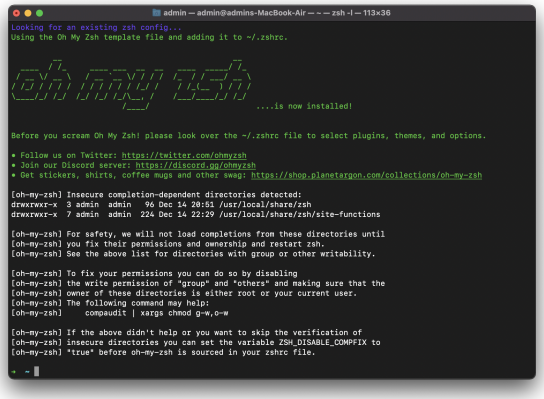


The vanilla configuration of Oh My Zsh is great for our needs, but you can further customize it if you want to - check out [their website] and [their documentation] to see how.

Handling Errors 💔

You are likely to see an error like the one to the right after installing Oh My Zsh. It is possible to fix this by running the command that is specified:

```
compaudit | xargs chmod g-w,o-w
```



A common minor error when launching zsh

Visual Studio Code

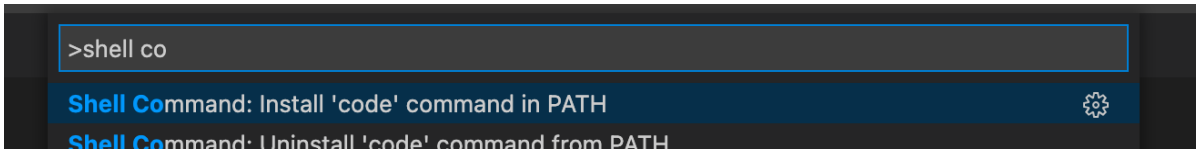
We will use VS Code as our editor in class. Download VS Code [here].

🚨 ***Extremely Important:*** To ensure you can properly execute code be sure that **Visual Studio Code** is in your Mac's `Applications` directory. ***It will not be placed in the Applications directory by default!*** Open the **Finder** application and navigate to the `Downloads` directory. With it open, drag the freshly downloaded **Visual Studio Code** application into the `Applications` directory. A video on how to accomplish this is below!

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/3d695688-0e7b-4764-970c-bb452e18b562/VS_Code_Install_macOS.mp4

Install the `code` Command in your PATH

1. Launch VS Code using spotlight (`Command ⌘ + Space` - then start typing **Visual Studio Code** until you see the app, then press enter). You may be asked to allow it to run the first time you use it since it is an application downloaded from the internet. If you are asked, allow it.
2. Type `Command ⌘ + Shift + P` to open the command palette.
3. Start typing **shell command** and when you see the **Shell Command: Install 'code' command in PATH** command - click it!



The command palette, with the **Shell Command: Install 'code' command in PATH** option highlighted.

4. You will see a dialog box that reads "Code will now prompt with 'osascript' for Administrator privileges to install the shell command." Select **OK**.
5. You'll then be given a prompt to enter your user account password to continue. Do so.
6. You'll be given a message the "Shell command 'code' successfully installed in PATH. Select **OK**.
7. Quit VS Code and the Terminal application.
8. Relaunch Terminal

Check **this link** for troubleshooting if you run into issues.

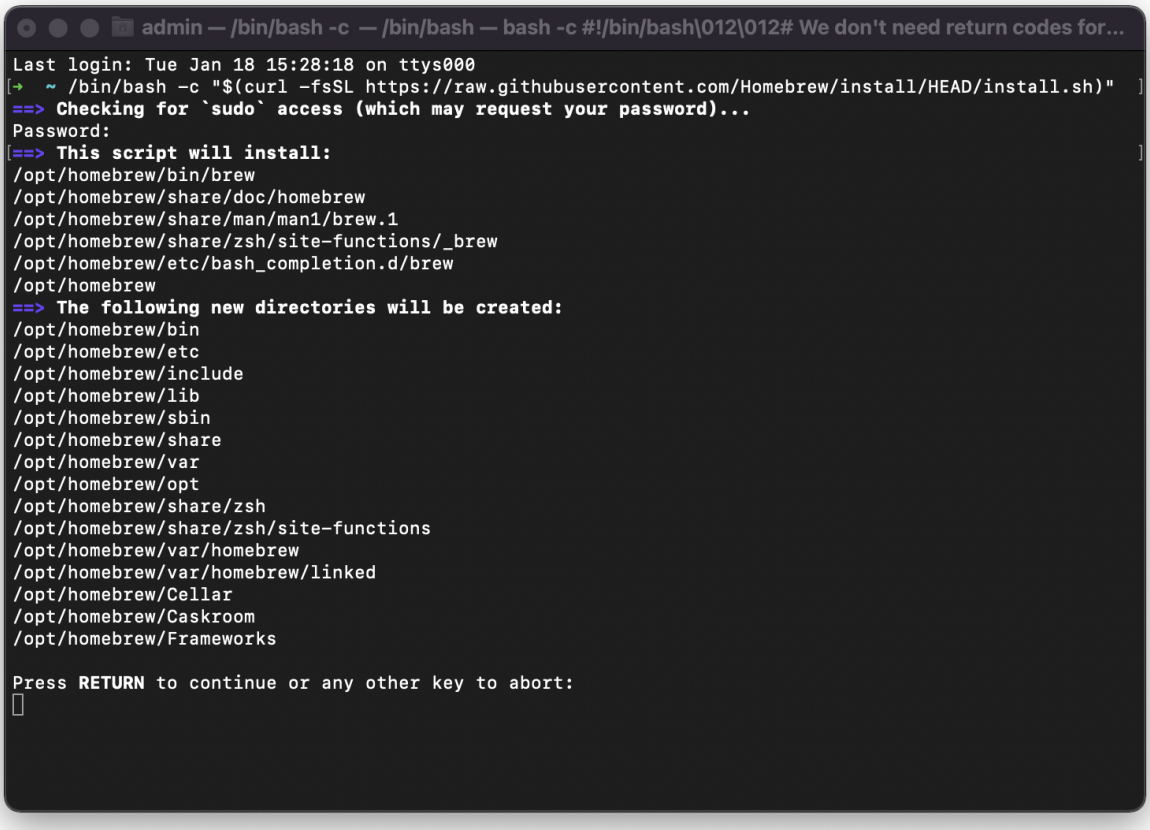
Homebrew

Homebrew is a package manager that we will use to install various command-line tools in our class. Learn more **here**.

Open up Terminal, and paste the following command to install Homebrew. You might be prompted to install XCode Command Line Tools during the install process.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

You will be prompted to enter the user password for your device. Do so. It will not be displayed on the screen in any form as you type it - this is common for command-line password entry. After entering it, you will be prompted to allow the script to install various applications and create various directories, as shown in the screenshot below. Press `Return` to allow this.



⚠️ If you are prompted to install the XCode CLI, say yes.

Next Steps

After the installation has been completed you may be prompted to enter further commands to finalize the installation. **You must complete the actions in these prompts before proceeding.**

Handling Warnings ⚠️

You may encounter a warning at the end of your installation that says something like this:

```
Warning: /opt/homebrew/bin is not in your path
```

If this occurs, you will need to add the directory specified to your PATH. You can do so with this command:

```
export PATH=$PATH:/opt/homebrew/bin
```

Git

Git is the version control software we will be using - it's an extremely popular tool among developers.

```
brew install git
```

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. Ensure you leave the quotes surrounding your email.

Set the default branch name to main with this command:

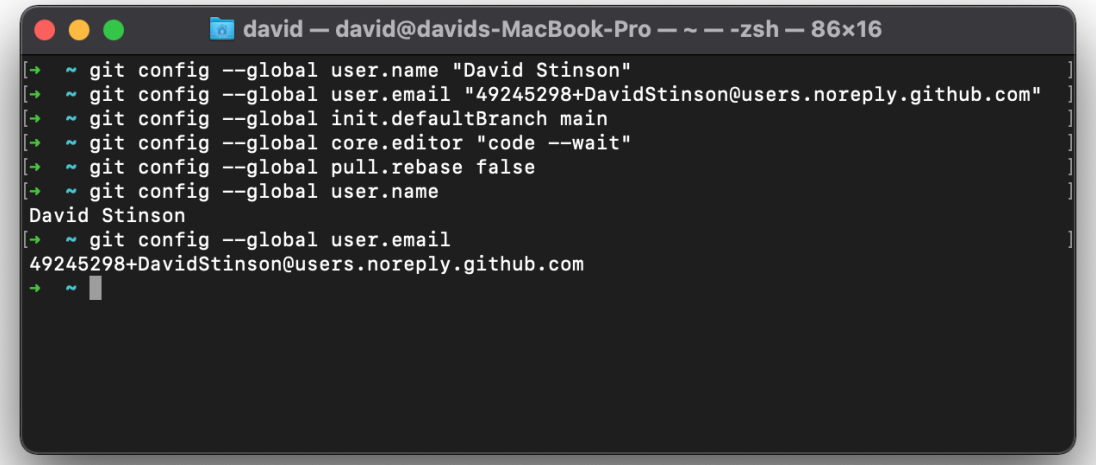
```
git config --global init.defaultBranch main
```

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 from a repo:

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



The result of the above commands being run. Note you can use `git config --global user.name` to check your stored username and `git config --global user.email` to check your stored email.

Configuring a Global Git Ignore File

> 🚨 ***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:

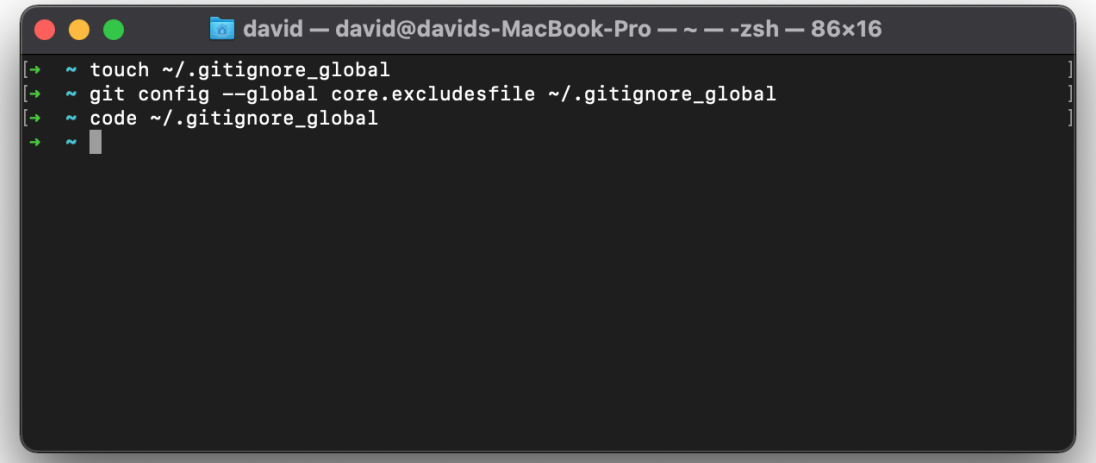
```
touch ~/.gitignore_global
```

Next, configure Git to use this file:

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

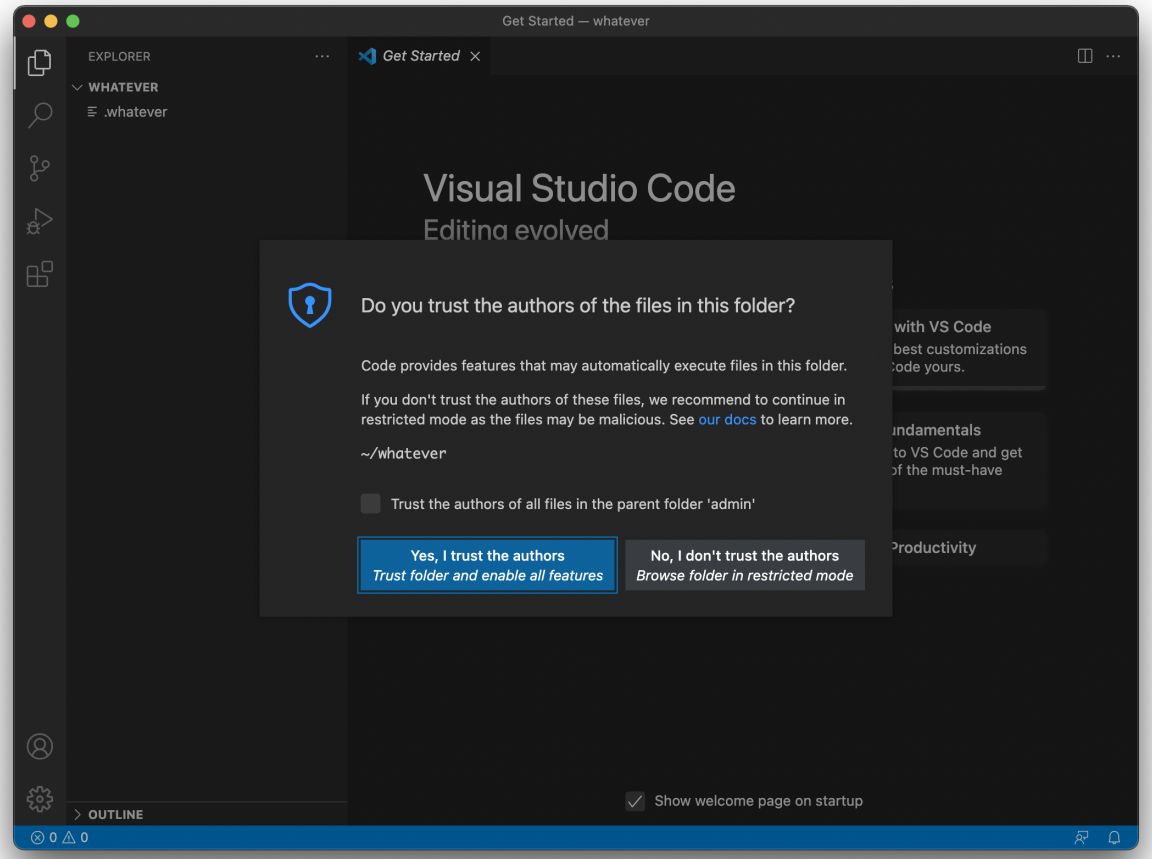
Open the new `.gitignore_global` file in VS Code:

```
code ~/.gitignore_global
```



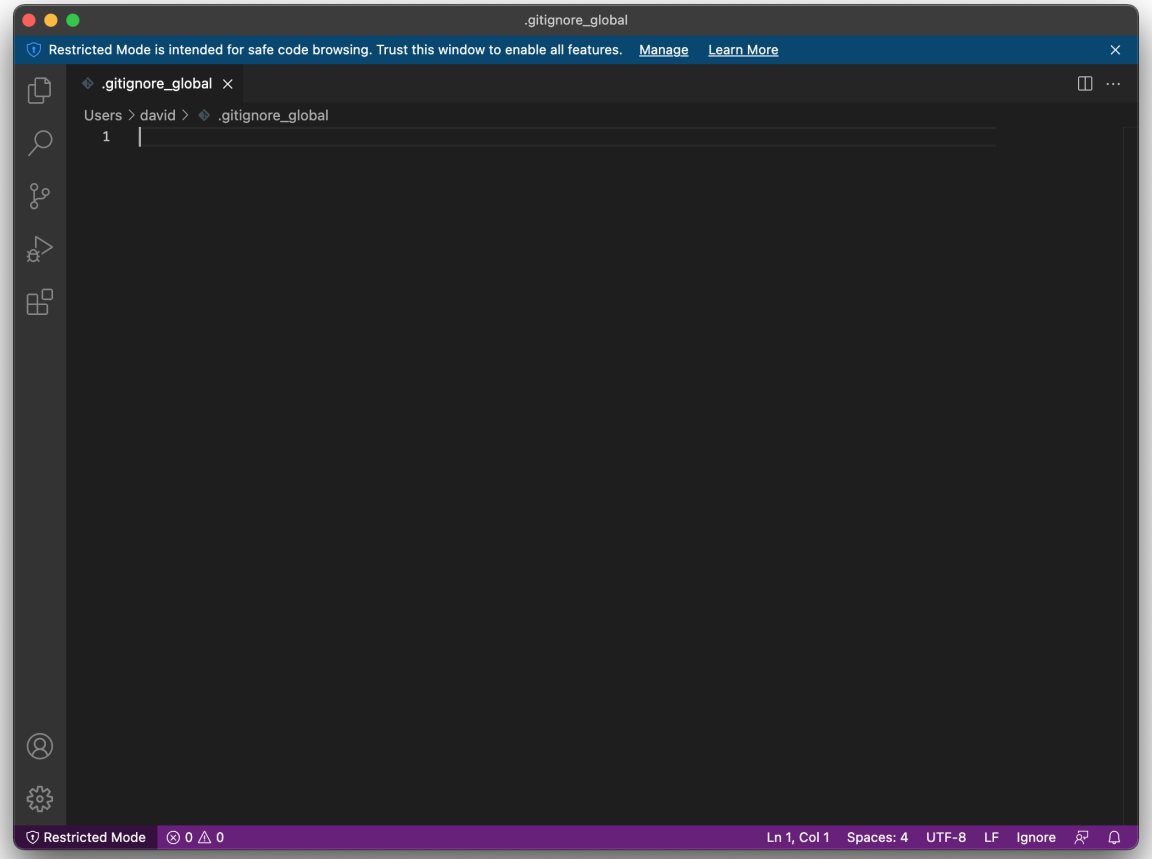
Creating and opening `~/.gitignore_global` in VS Code

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:

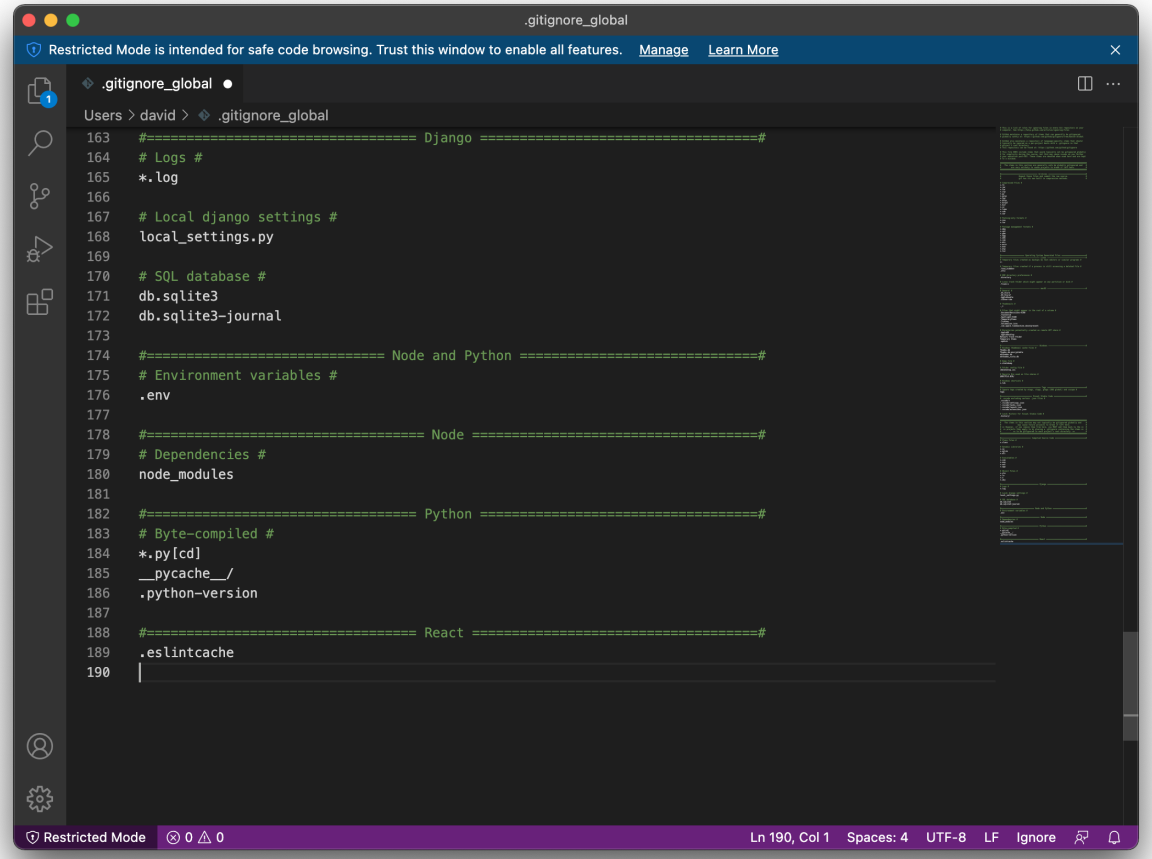


The new `.gitignore_global` file open in VS Code.

Here is a .gitignore_global file for you to use.

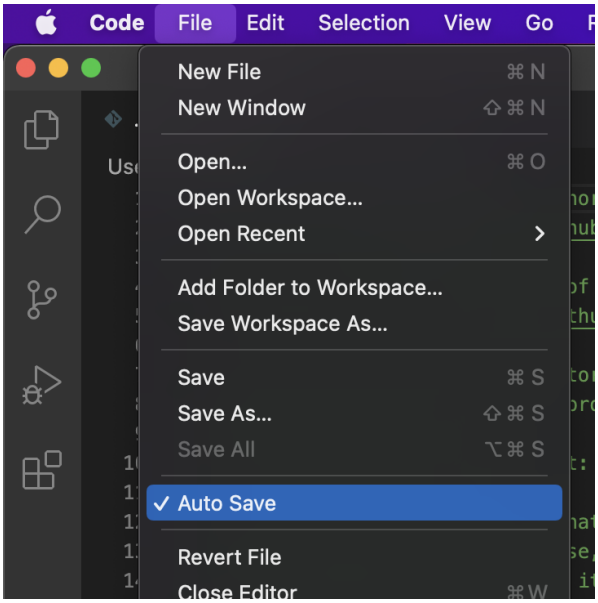
Open the above page and copy all of its contents.

Paste the contents of the file you copied into VS Code, then save the file.



The end of the new `.gitignore_global` file.

This is a great time to turn on **Auto Save** as well! As shown to the right, this setting is in the **File** menu - select it, then re-open the **File** menu to ensure a checkmark is next to the A**uto 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 `Command ⌘ + S`.

You can close VS Code for now.

Auto Save turned on in the File menu.

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:

```
brew install gh
```

Then 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.

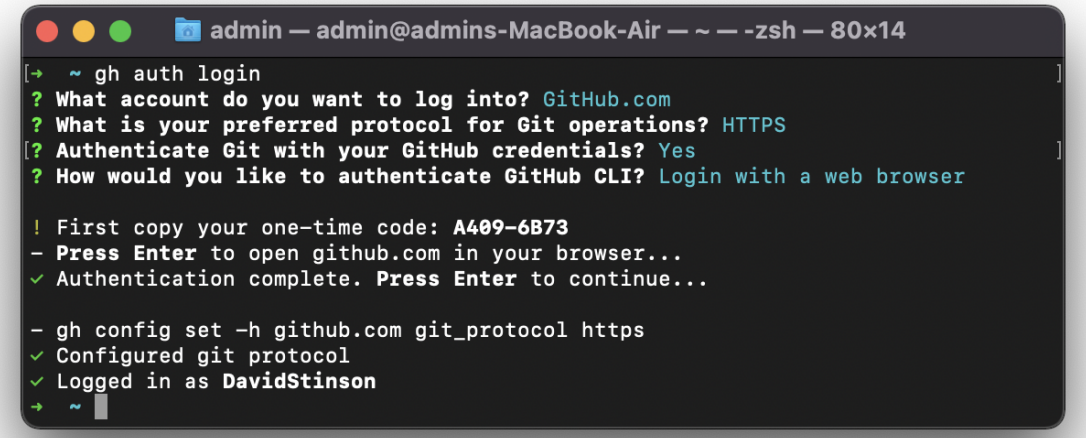
The Second prompt will ask you to choose whether you want to use HTTPS or SSH. Select the **HTTPS** option.

> 🧠 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 `return` 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 `return`.



A demonstration of the completed `gh auth login` process.

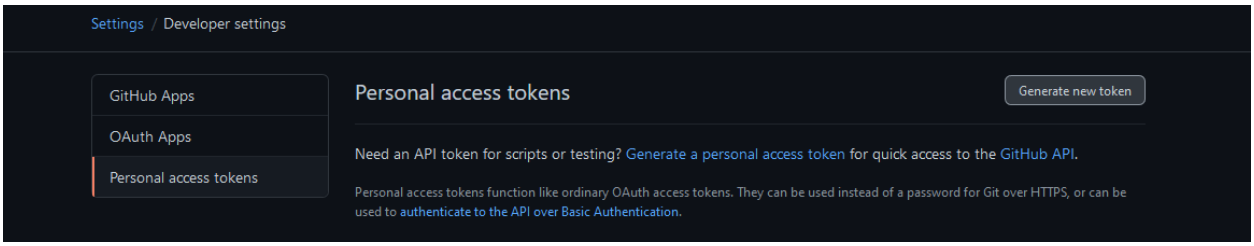
> 🥳 If you were given the option to Authenticate Git with your GitHub credentials, and said yes - you may skip the next step: *Generating a GitHub Personal Access Token.*

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).

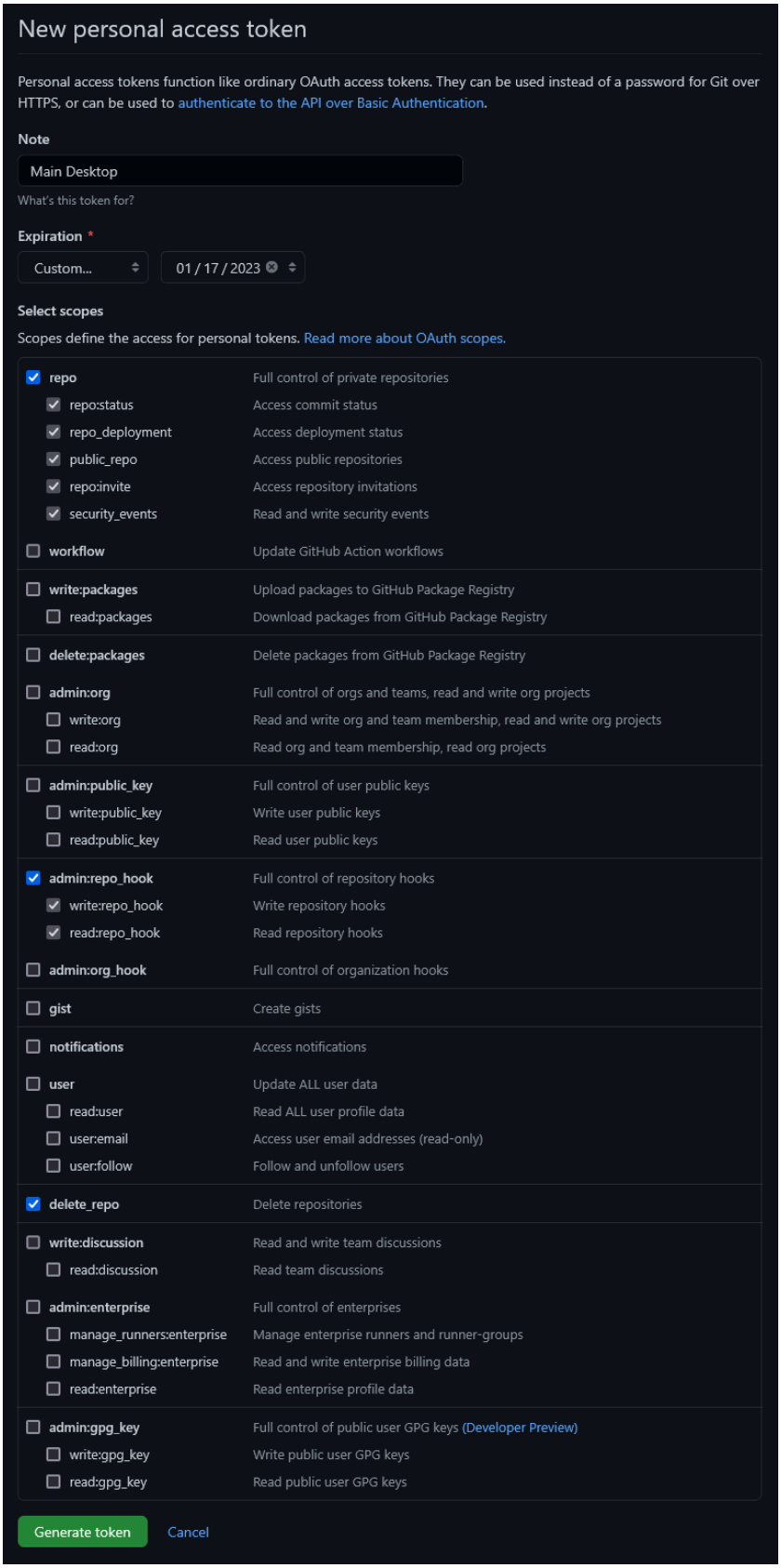
First, visit `https://github.com` and ensure that you are signed in. Also, ensure that you have **verified your email address** with GitHub. After doing so, navigate to `https://github.com/settings/tokens`.

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

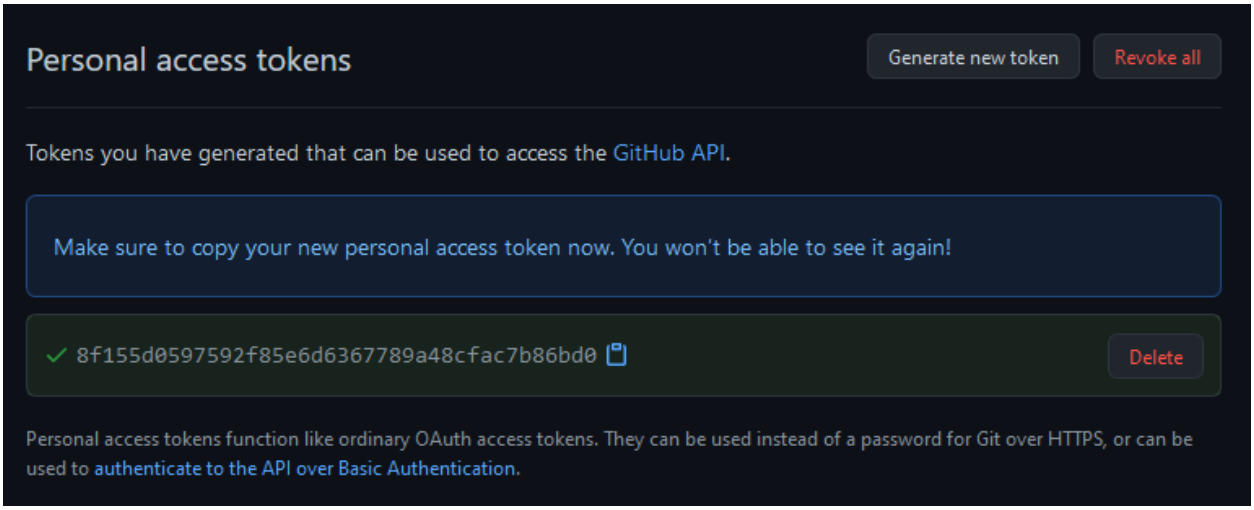


The **Personal access tokens** page in **Developer Settings**. The **Generate new token** button is highlighted.

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



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 will 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!

Node.js

Node is a JavaScript engine for the backend. We use it to power our web servers and connect to our databases.

```
brew install node@16
```

Then ensure you can execute the `node` command by running:

```
brew link --force --overwrite node@16
```

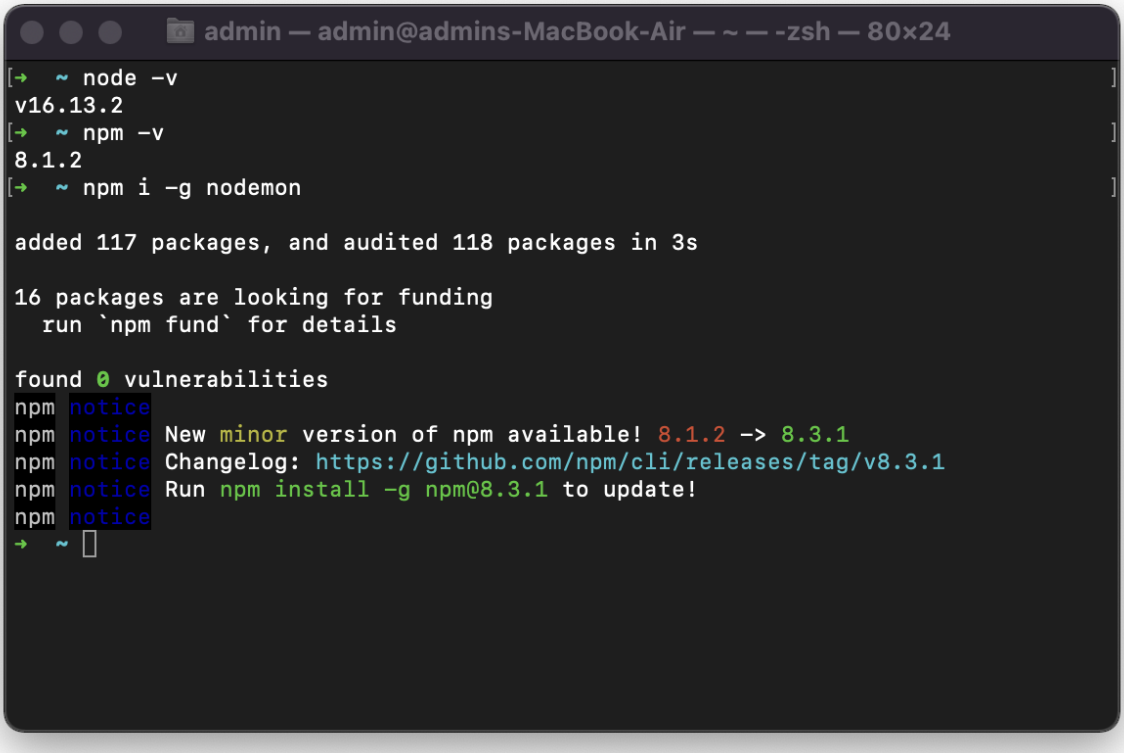
Close your terminal, and open a new one.

In the new terminal, verify the installation by running:

```
node -v
npm -v
```

The above commands should display versions without any errors. To verify that all the required permissions are set correctly, try to install *nodemon* globally:

```
npm i -g nodemon
```



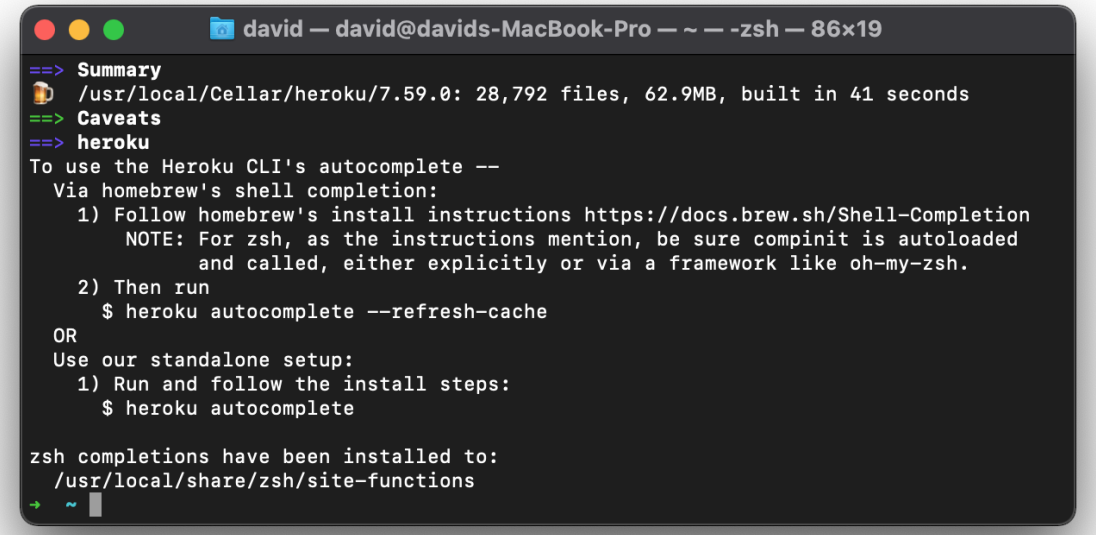
node, npm, and nodemon all correctly installed.

Disregard any prompts to update `npm` for now.

Heroku

We will deploy full stack applications to Heroku. Install it with this command:

```
brew tap heroku/brew && brew install heroku
```



The Heroku install process completing successfully.

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

Launching Apps with Spotlight

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

1. Pressing `Command ⌘ + Space` to open *Spotlight*

2. Start typing the name of the app until the app is highlighted
3. Press `Return` to open the app!

Switching Between Applications

Quickly switch between running applications by pressing `Command ⌘ + Tab`.

If a minimized application does not display after tabbing to it with `Command ⌘ + Tab`:

1. Continue to hold down `Command ⌘` and release `Tab`
2. Press `Option ⌥`, then release `Command ⌘`

Switching Between Instances of an Application

You can switch between multiple windows of the same application using `Command ⌘ + ` ` (that's a back-tick character above the `Tab` key).

> 💡 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 `Command ⌘ + Shift + 3` to take a screenshot of your entire screen space, or use `Command ⌘ + Shift + 4` to take a screenshot of a selected area instead. `Command ⌘ + Shift + 5` will allow you to view more advanced screenshot options.

By default, screenshots will be saved to your desktop.

Uploading Screenshots and Images to imgur

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, 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.

Level Up 🚀

Check out **Mac Keyboard Shortcuts Every Dev Should Know**

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 decide against using one entirely), that is no problem.