# 

# 

# 

# 

# **Open Sourcing Tutorial**

By: Adam Sarabia, Andrew Lambe, Cameron Porembski, Darius Herman, Dominic Grandsard, Lin Reyes, and Parker Libby

# **Table of Contents**

**Installations**  **2**

Docker Desktop Installation 2

HeidiSQL Installation 5

Visual Studios Code Installation 6

**Duplicating the GitHub Repository** **7**

Method 1 Steps 7

Method 2 Steps 12

**Setting Up Applications** **14**

Docker Desktop/Visual Studios Code Set Up 14

HeidiSQL Set Up 34

**Starting the Website for Testing Purposes** **39**

**How to Change Colors/Themes** **43**

**How to Replace Images** **46**

**Branding Consistency** **52**

**How to Make it a Website** **54**

**General Tips/Information** **55**

**Additional Resources** **56**

**Thank You! 56**

# **Installations**

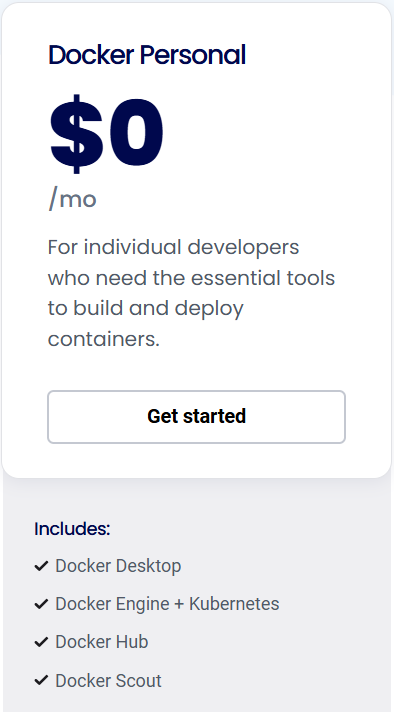
## Docker Desktop Installation:

* What is this: This is the main piece of software that you will use to start modifying the codebase
* Steps:

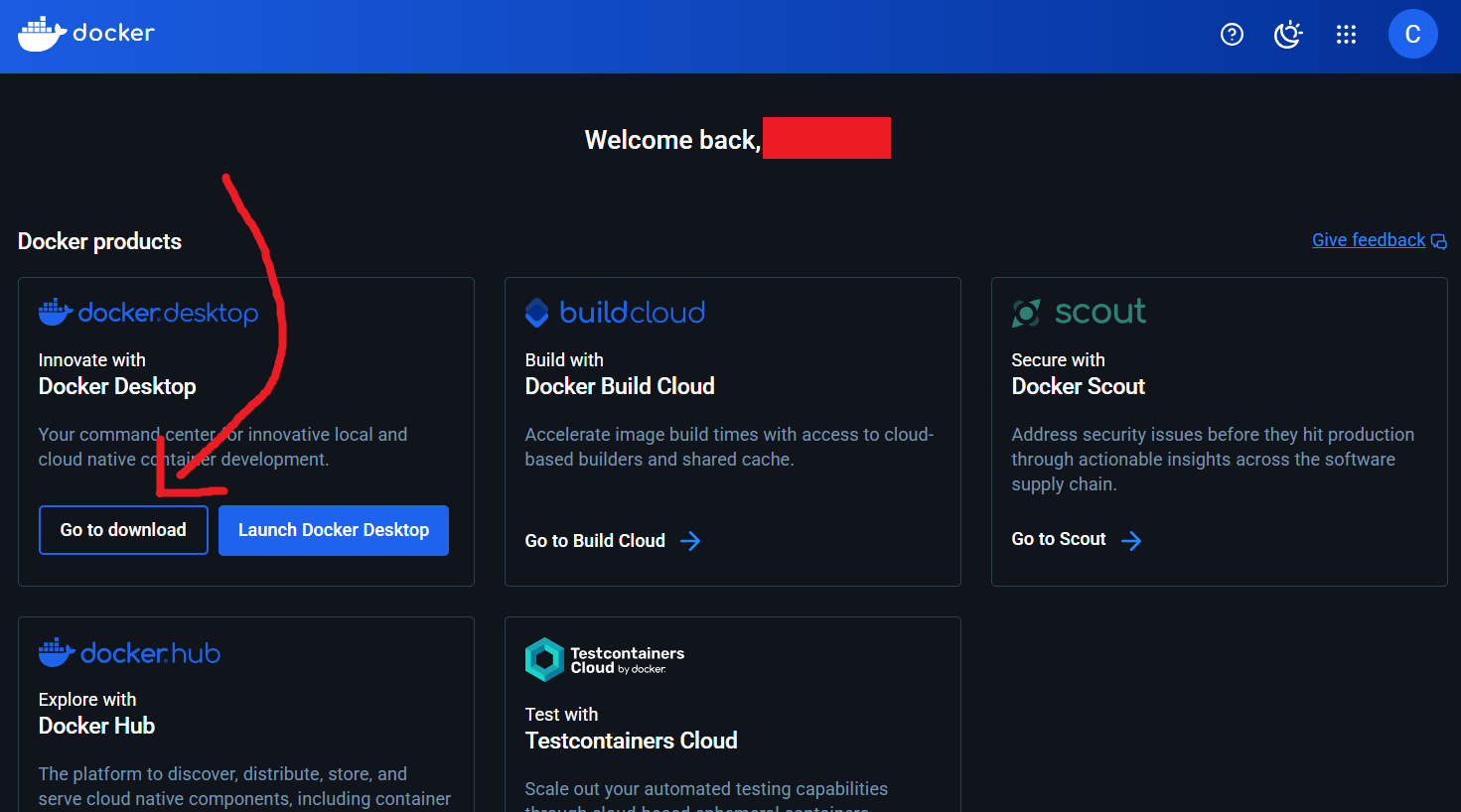
1. Go to this Website: <https://www.docker.com/products/docker-desktop/>
2. Click the button that says: “Choose plan” (this will be free don’t worry)



1. Choose “Docker Personal” by clicking Get Started



1. Sign in using any email you wish. After that it will redirect to this page, click “Go to Download”:



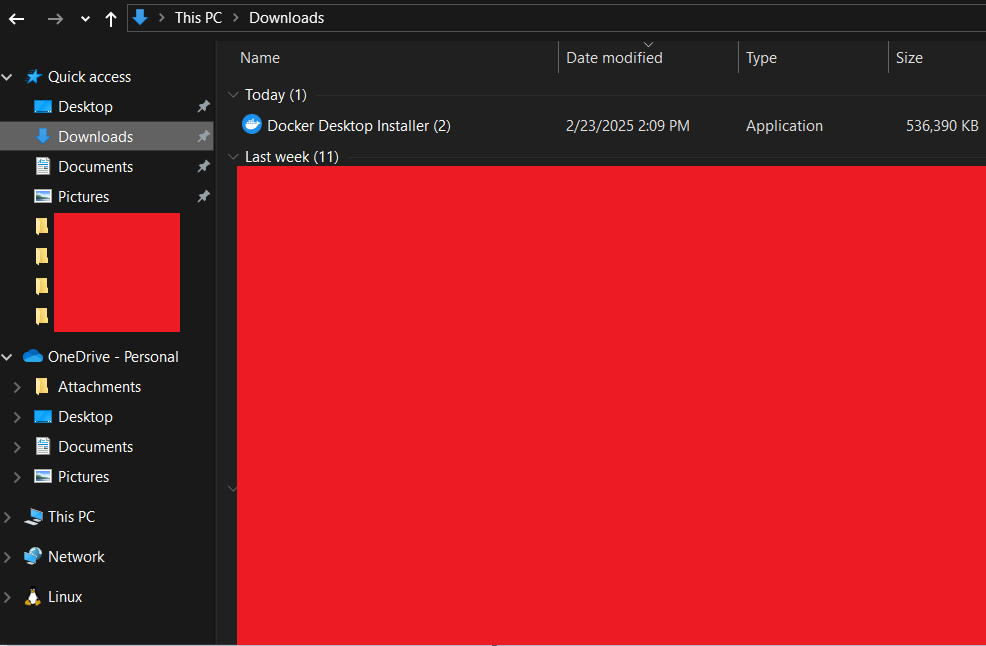
1. It will take you to another page where you can choose the version of Docker Desktop that works with your OS. It currently supports Mac, Windows, and Linux



1. If you’re using Windows, click that, and then choose this version:



1. Look for the file in your Downloads folder and then run it



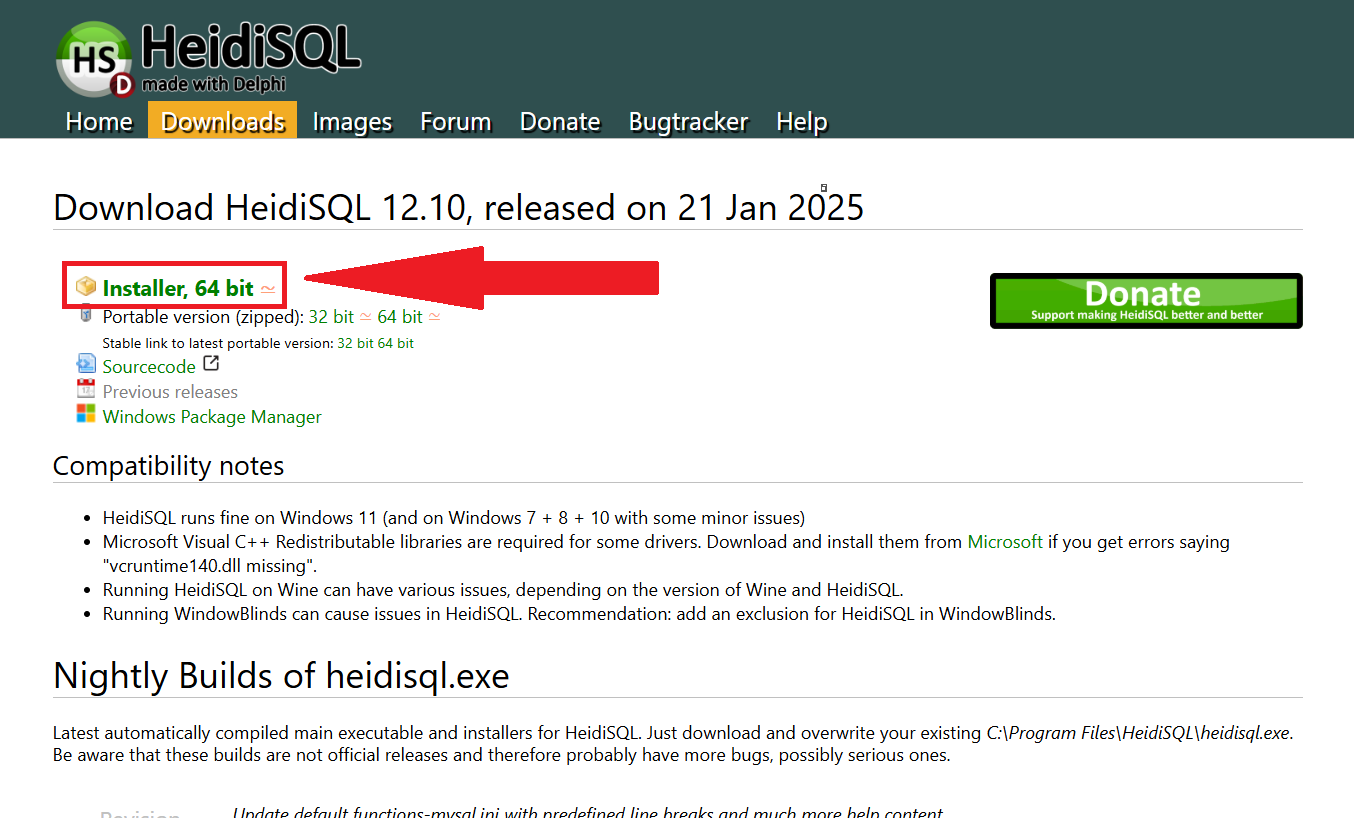
1. Then it should be installed, you will need to set up the application though, and information regarding that is below in this document.

## 

## HeidiSQL Installation:

* What is this: This is the software that holds the Database for Peoria Fresh
* Steps:

1. Go to this website: <https://www.heidisql.com/download.php>
2. Click this button



1. Once the installer is downloaded, run the installer.
2. Then you’re done installing the software

## 

## Visual Studios Code Installation:

* What is this: This is the software that houses all of the Front-end and Back-end code. You will access this through Docker Desktop
* Steps:

1. Go to this website: <https://code.visualstudio.com/>
2. Click this button



1. Once the installer is downloaded, run the installer
2. Then you’re done installing the software

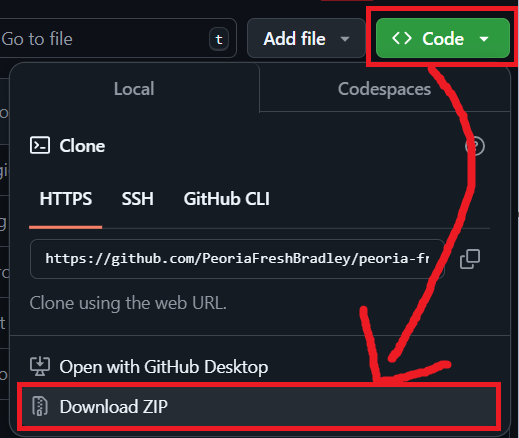
# 

# **Duplicating the GitHub Repository**

* Before you do anything else, you’re going to create a new GitHub Repository that is a direct copy of the Peoria Fresh GitHub Repository. This way, your repository is entirely independent and cannot break the original codebase. There are two methods to do this, the second one is a bit faster, but we haven’t completely verified that it’s a perfect duplication like we have with the first method.
* NOTE: You’re going to need to do this for both the front end and back end

## Method 1 Steps:

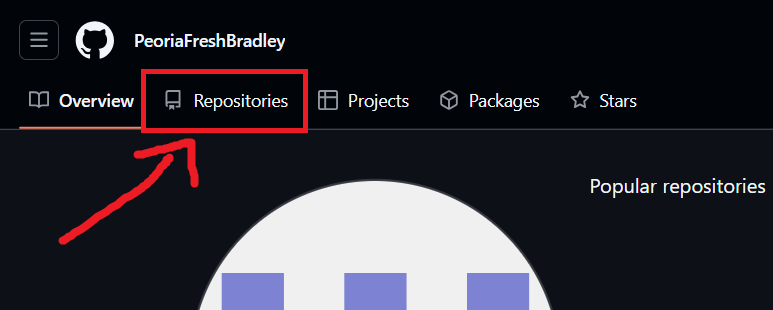
1. You will need to do this whole process twice, for the Front End and the Back End. If you’re duplicating the…
   1. **Front End**, go to this GitHub Repository: (link)
   2. **Back End**, go to this GitHub Repository: (link)
2. Download a Zip file of the entire Repository by clicking the green “ < > Code” button:



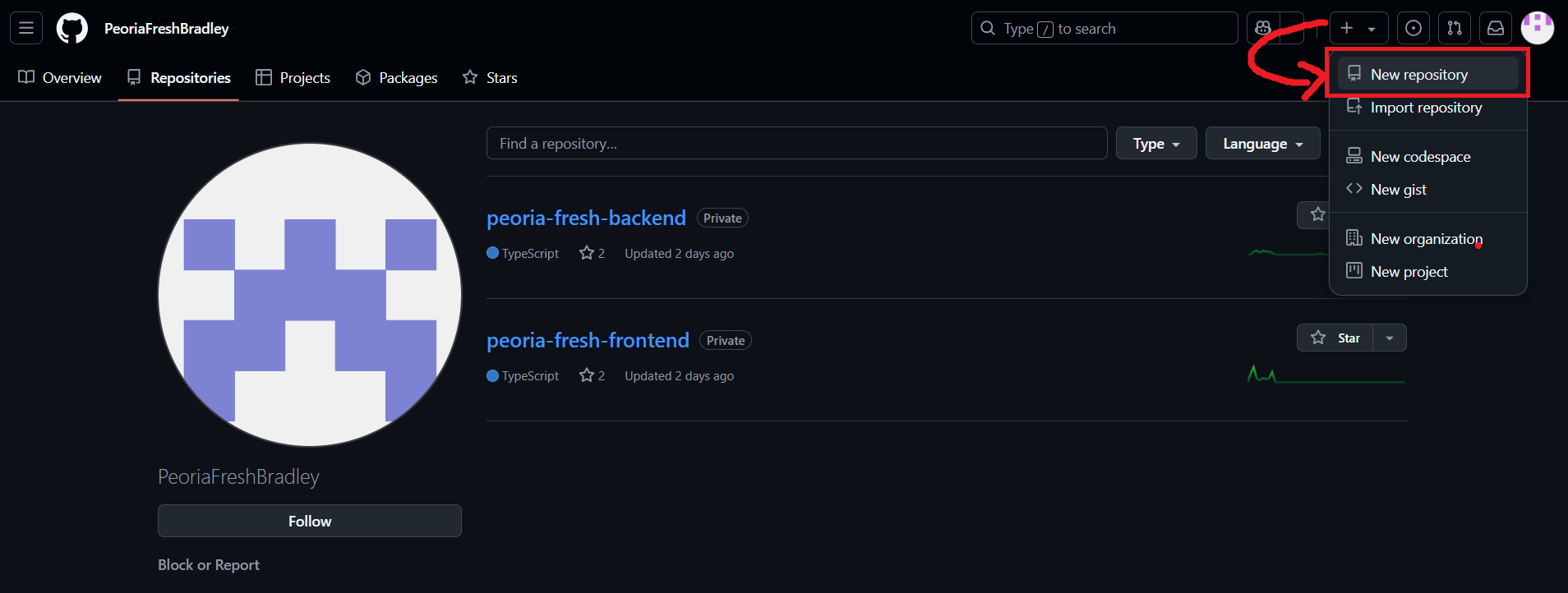
1. Next we’re going to actually make a new repository. Click on the Text next to the GitHub Logo



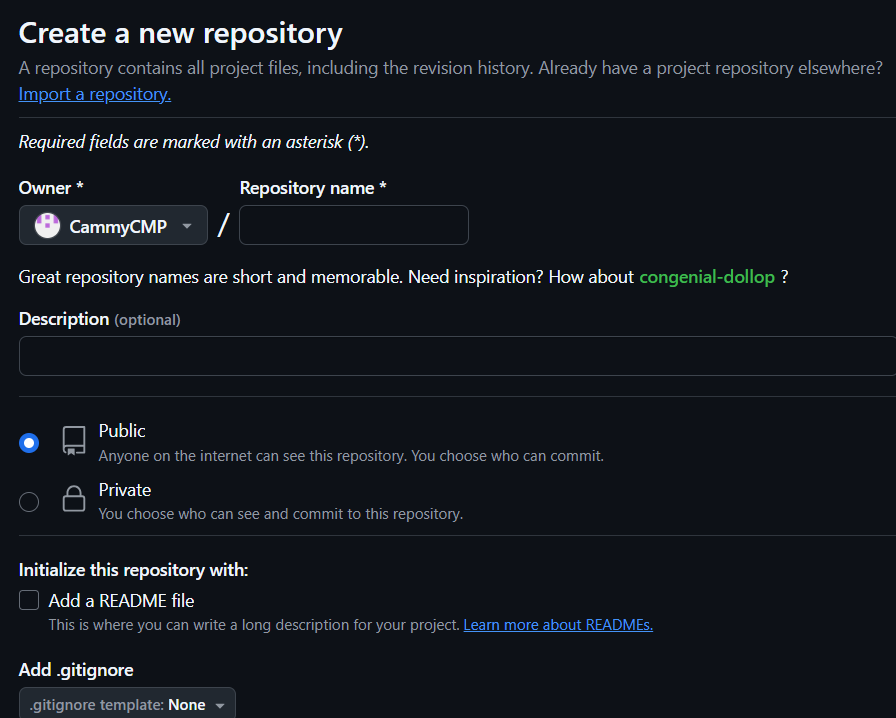
1. The click the “Repositories” button:



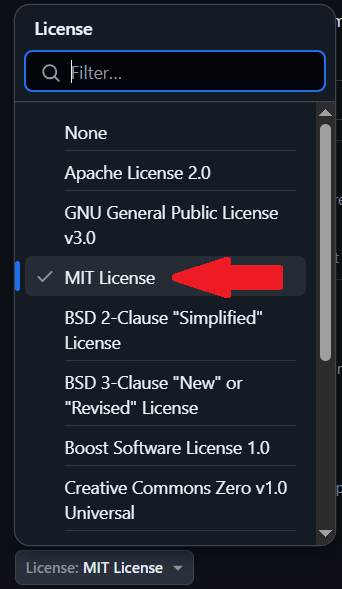
1. Then Click the Add Button in the Upper Right Corner (it looks like a + sign) and then New Repository



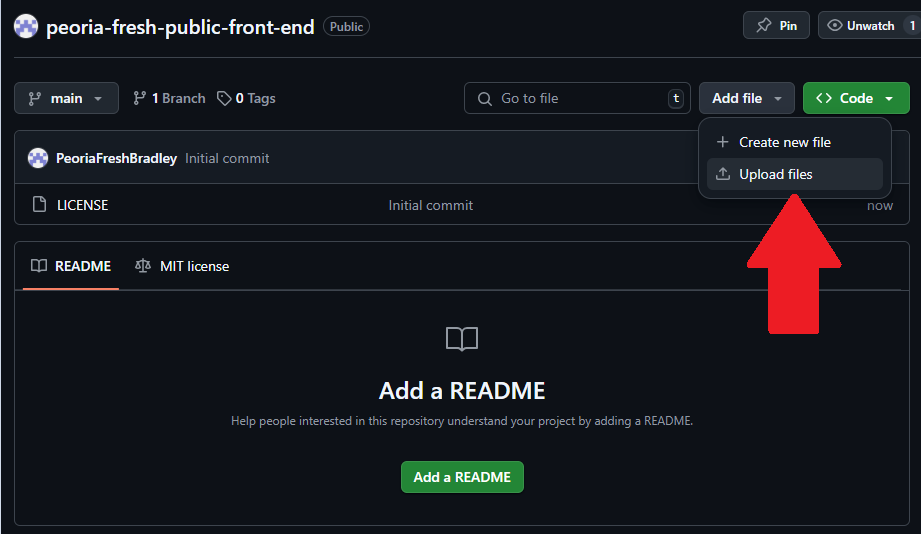
1. You should land on a page like this:



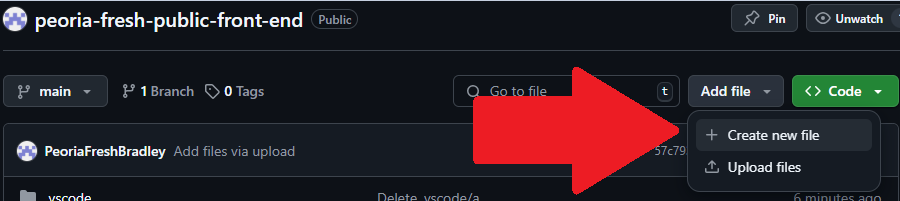
1. From there, you’re going to name your Repository whatever you want. We suggest you name it to be your city name and then “Fresh”
2. Next set your repository to be public or private depending on your preference.
3. Do NOT add a README file
4. Now in the License menu, select the “MIT License”



1. Then click the green “Create Repository” button
2. Once the repository is created, you’re going to extract that zip file you downloaded in step 2 into their own folder on your computer
3. Once you have those files extracted, click the “Add File” dropdown and then click “Upload Files”



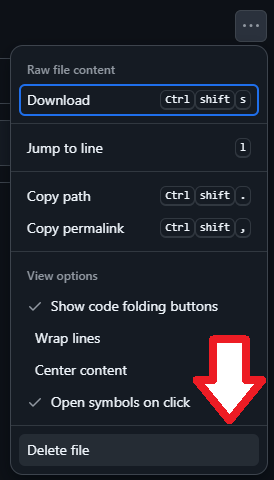
1. You will need to go through and slowly update your new GitHub to get it to match the file you extracted earlier. GitHub has a cap of 100 files per upload so you will need to break it up into chunks.
   1. If you want to create a new folder (which you will need to do):
      1. Click the “Add file” drop down and then click “Create new file”



* + 1. In the name bar, type the name of the folder and then add a slash at the end of the name and it will create a new folder

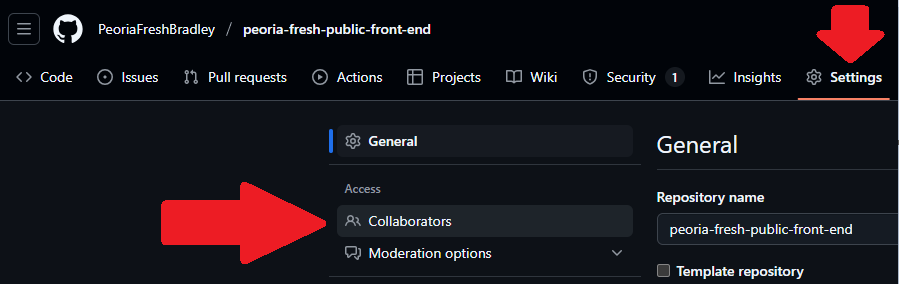


* + 1. You still have to add something to that folder so just put a temp name and then click the green “Commit changes” button on the right
    2. Navigate back to the folder you just created and click on the temp file you created
    3. Add a file you want to be in this folder. If you don’t do this and delete the temp file you created, it will delete the folder.
    4. Click the drop down menu at the top right corner and then click “Delete file”

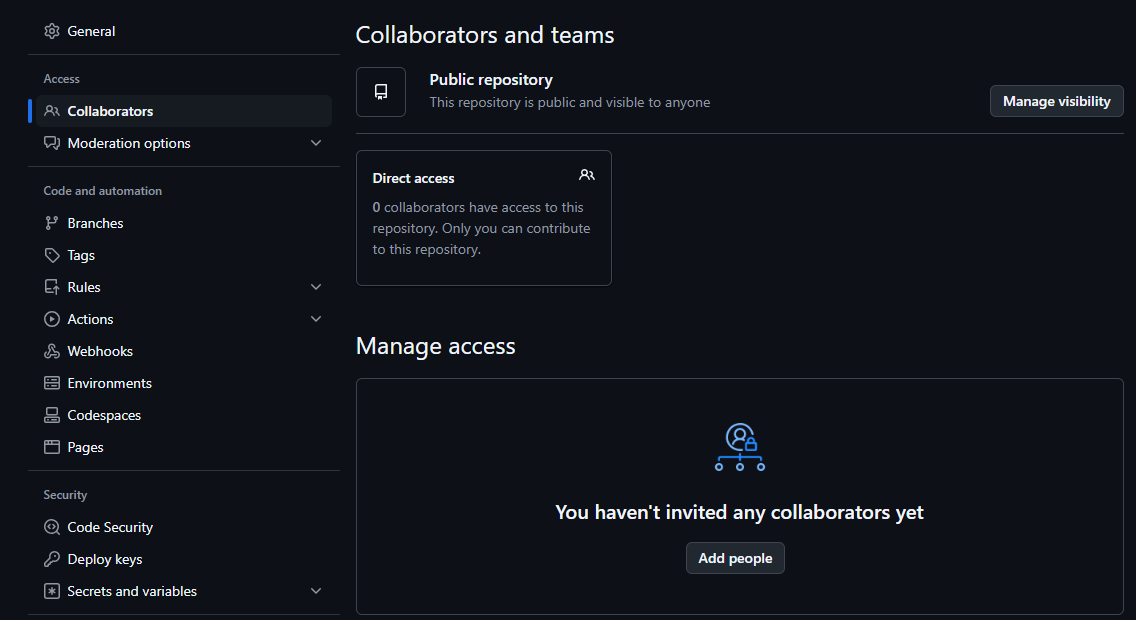


* + 1. Then click the green “Commit changes” button in the top right corner.
    2. Now you should have a new file with nothing in it, you can start uploading files to it.

1. Once your GitHub matches that file you extracted, then you should be done with setting up your Repository.
2. Now here’s how you share it with other people. Click on the “Settings” button on the top middle section of the screen.



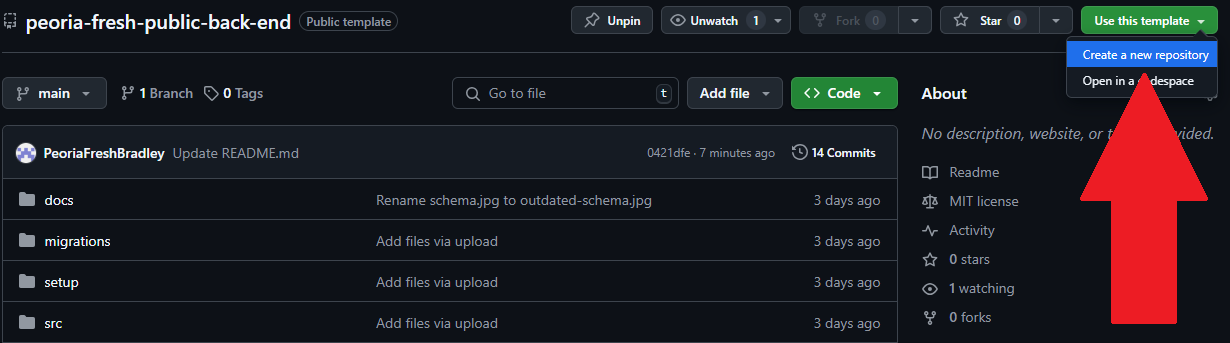
1. From that page, you can add any contributors you want.



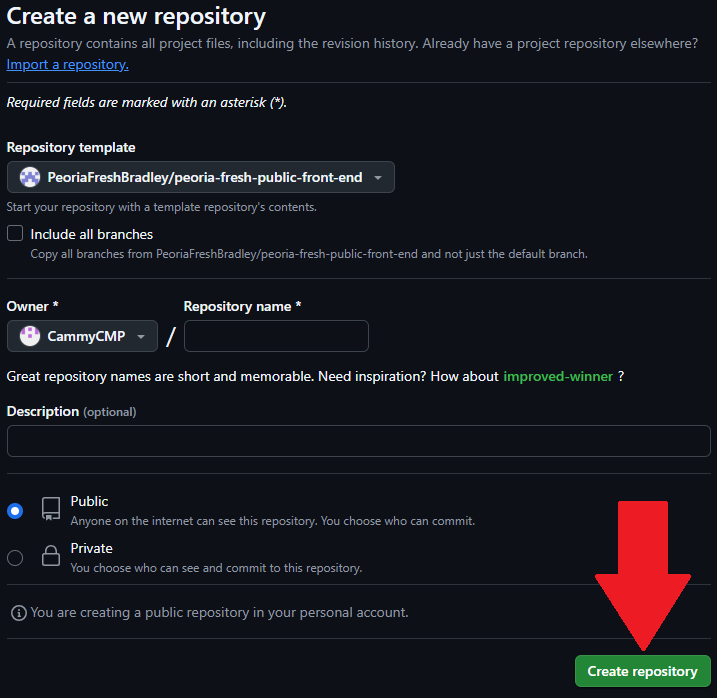
1. From there, you should be all set! Make sure to do this for both the frontend and backend!

## Method 2 Steps:

1. Another method that you can use to duplicate our GitHub Repositories is to use GitHub’s Template System.
2. On our GitHub, in the upper right corner, there is a green “Use this Template” Button. Select that and then click “Create a new repository”



1. From there, fill out the rest of the information and then click “Create Repository”



1. And then you should be done!

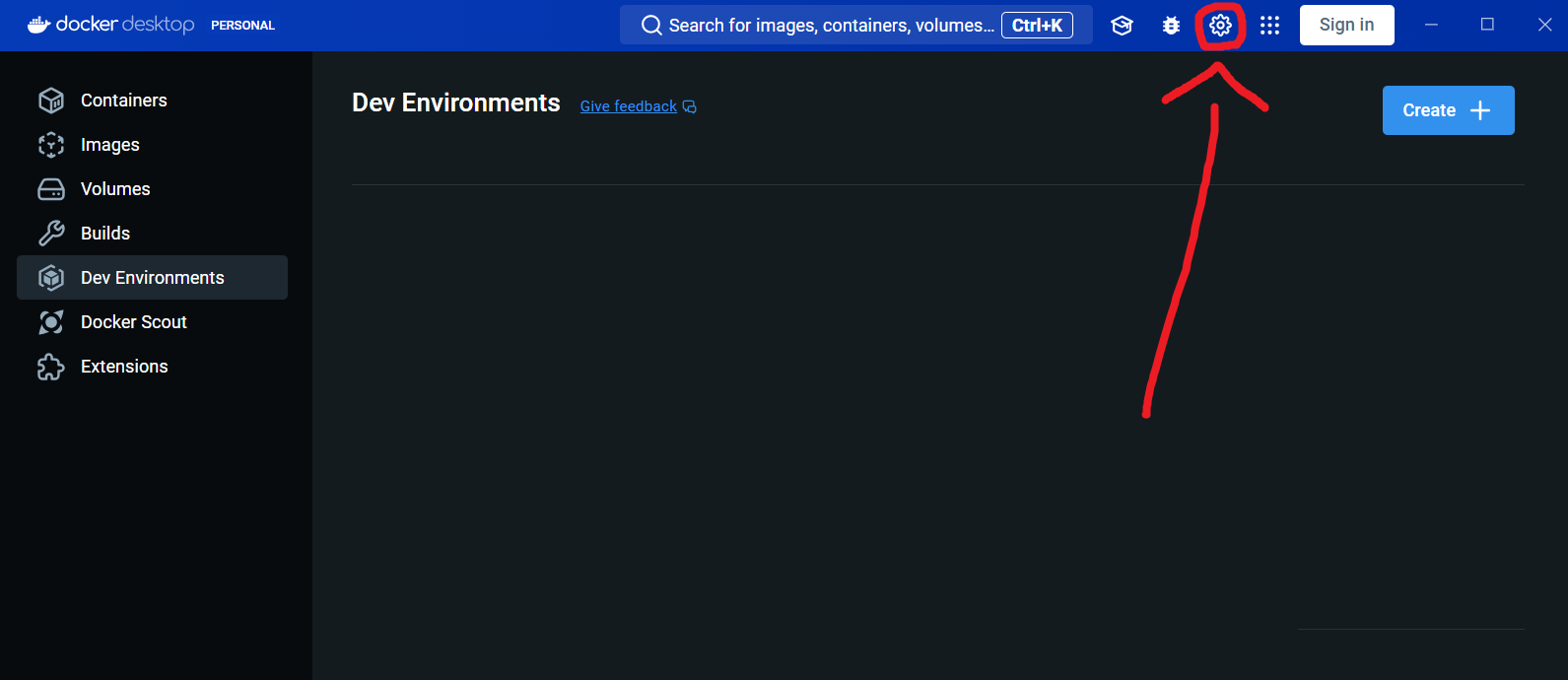
# 

# **Setting Up Applications**

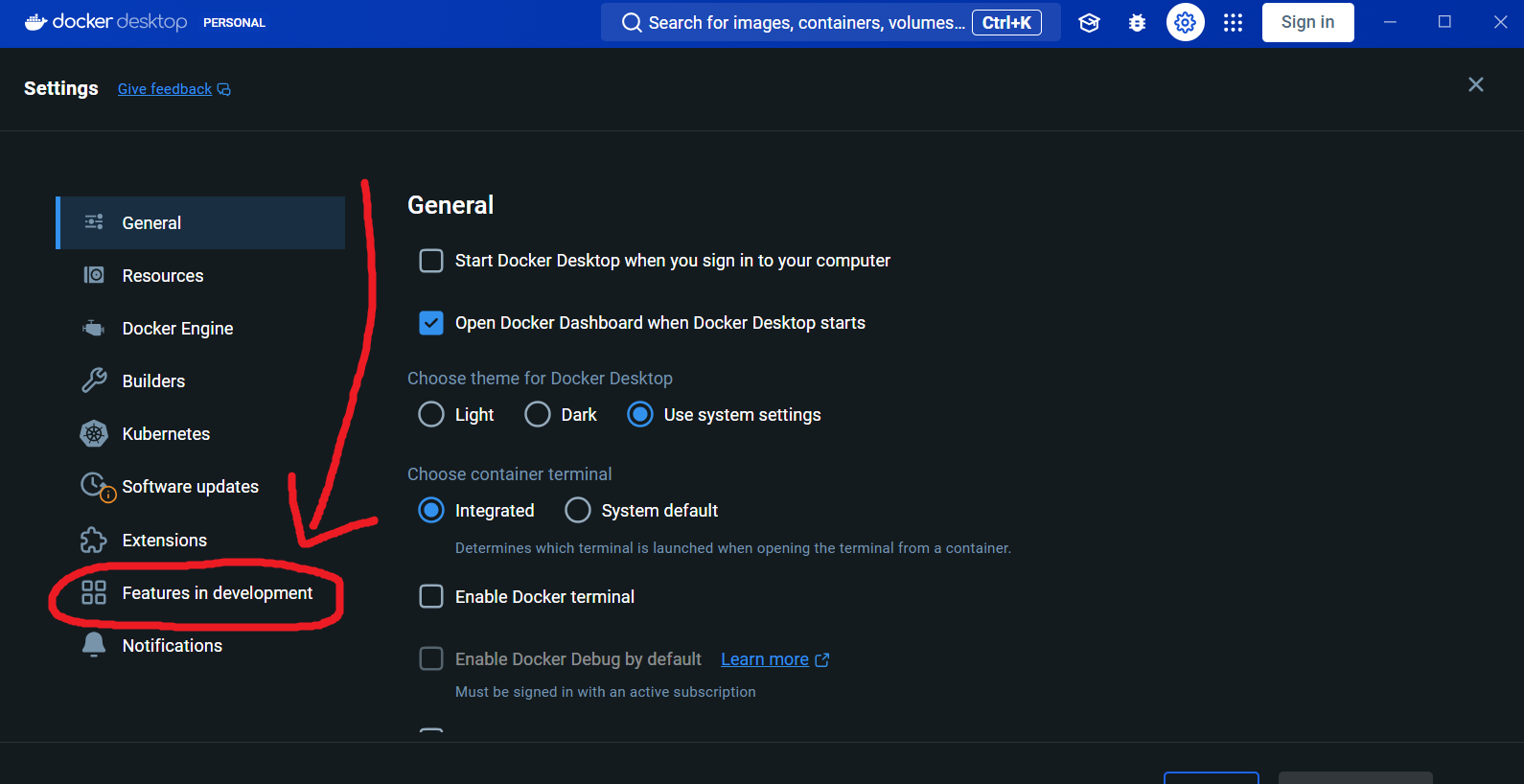
## Docker Desktop/Visual Studios Code Set Up:

* Steps:

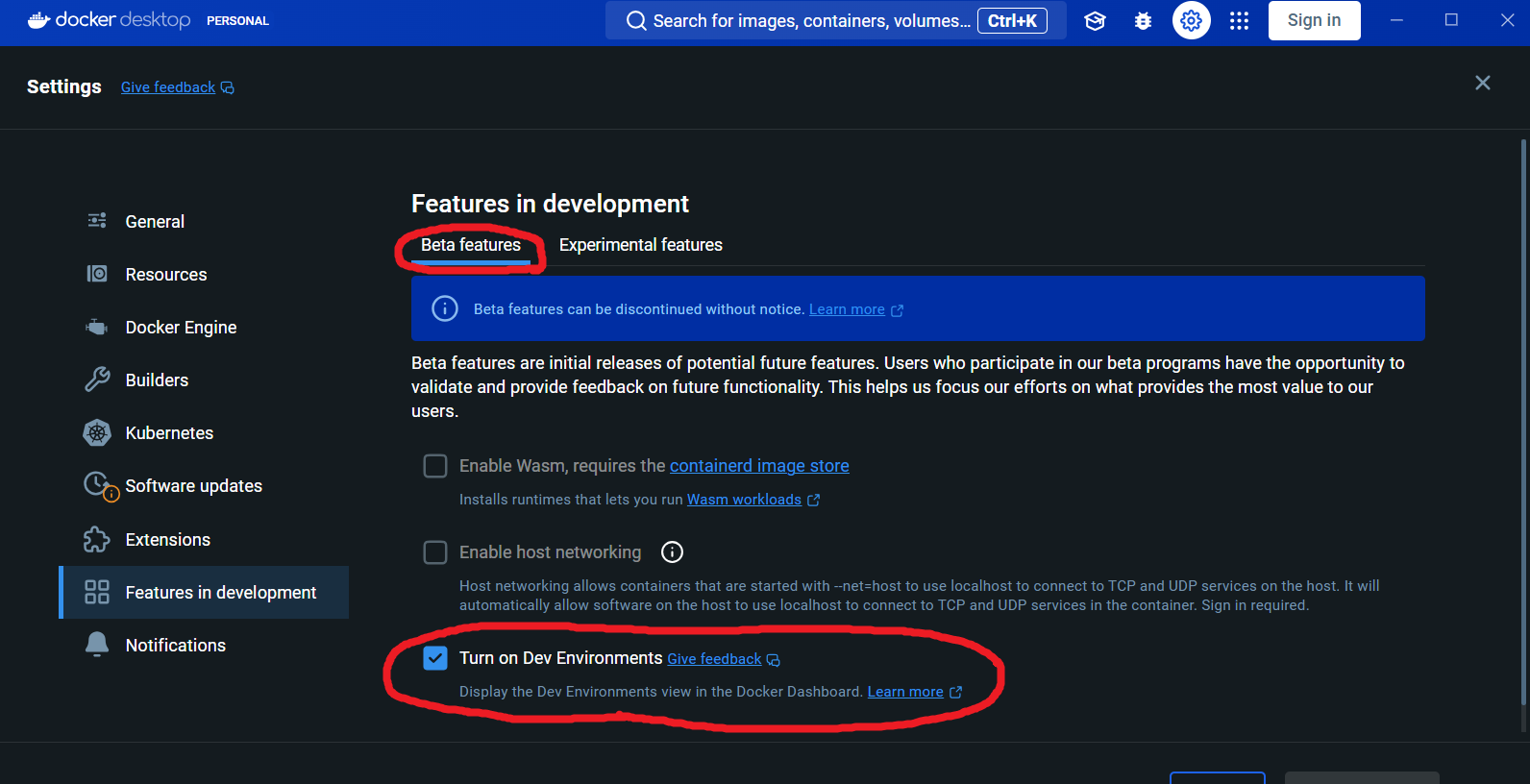
1. When you first launch Docker Desktop, you may get a pop-up asking for admin privileges, click yes
2. When you have the software open, click the settings button



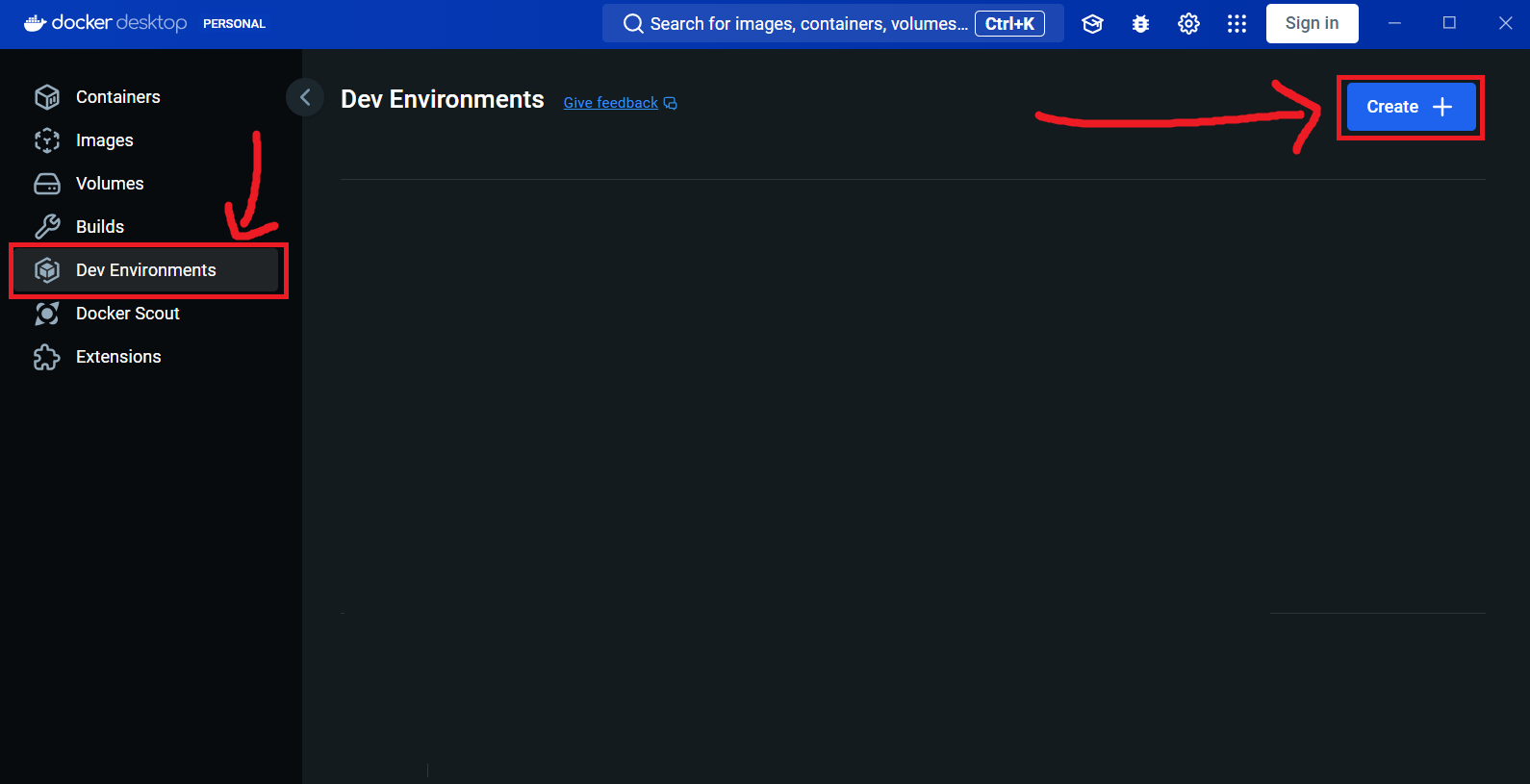
1. Then you’re going to click “Features in development”



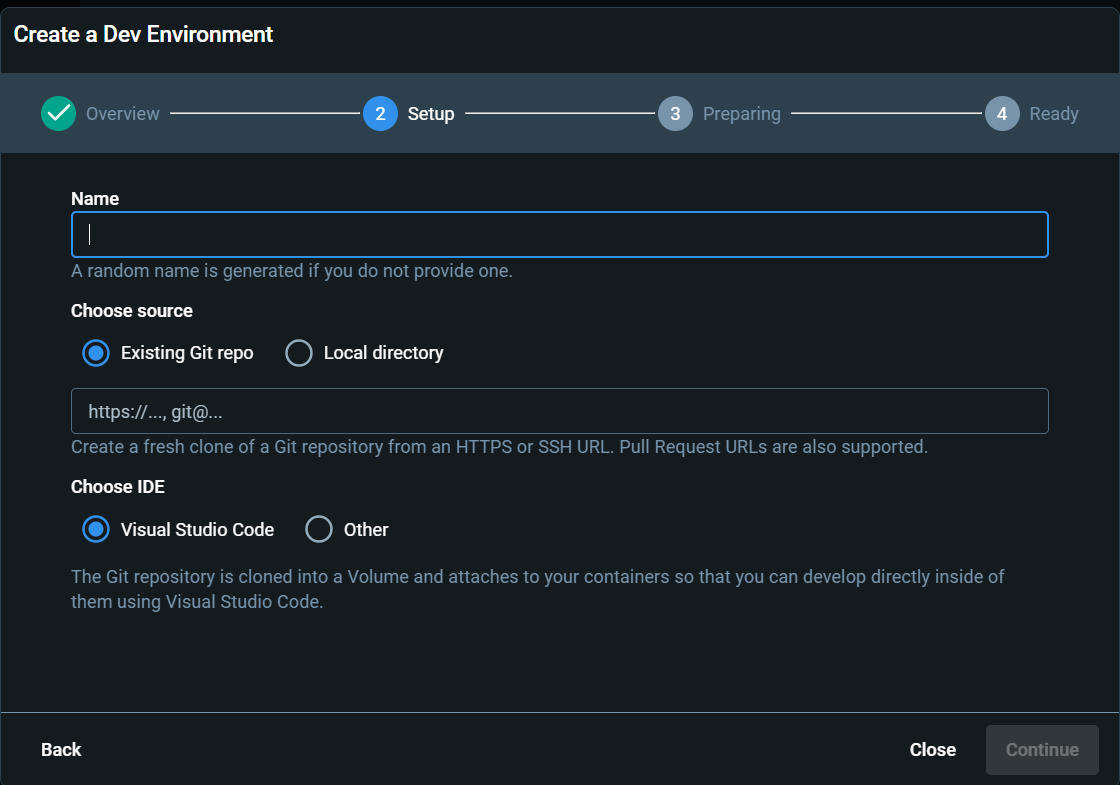
1. Next, click turn on the **Dev Environments** feature in the Beta Feature Tab, this will cause a new ribbon on the left side when you are not in the settings menu



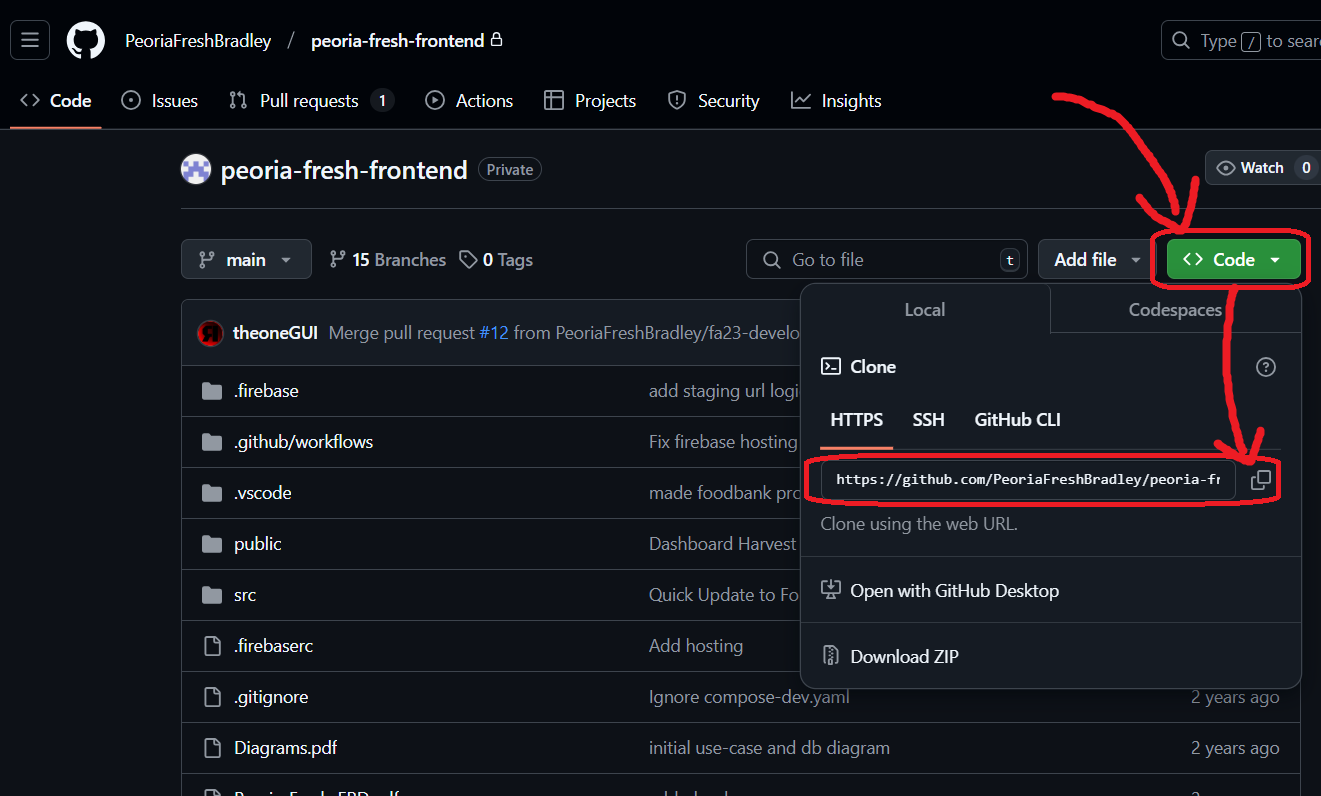
1. Once you’re in the Dev Environments window, click on the blue “Create” button



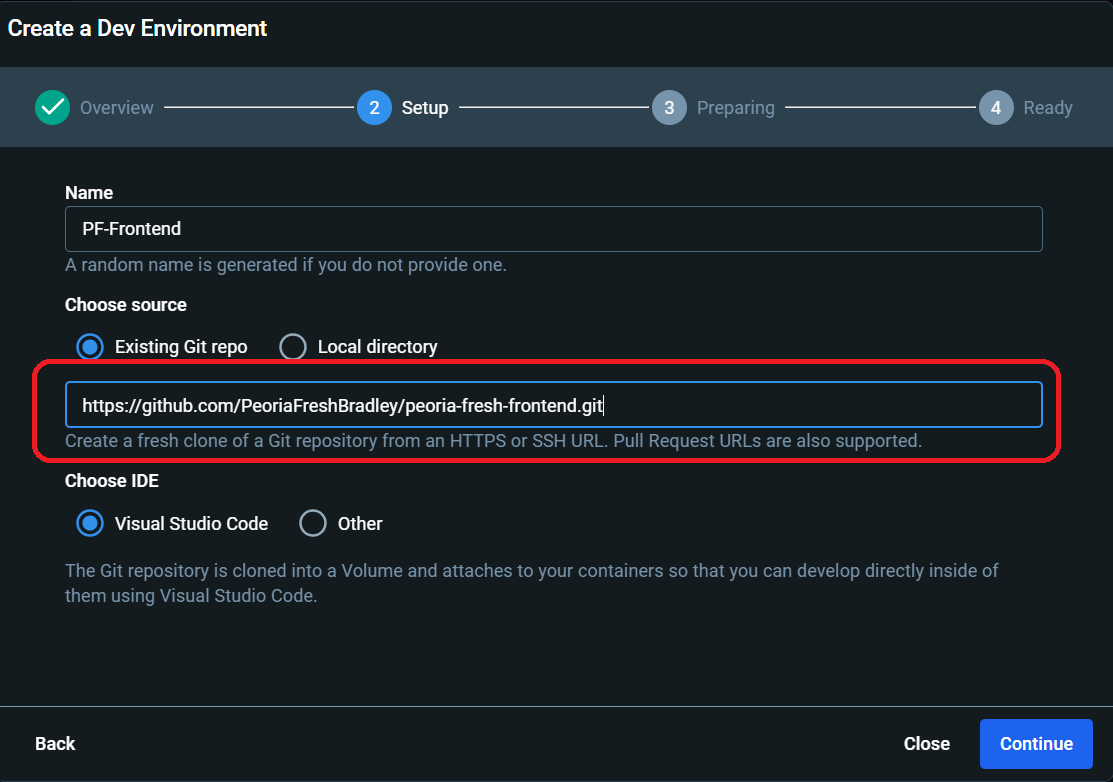
1. A new window will pop up. Click “Get Started” and then you’ll see this page. Type in any name you’d like, as long as you can tell the difference between the frontend and backend



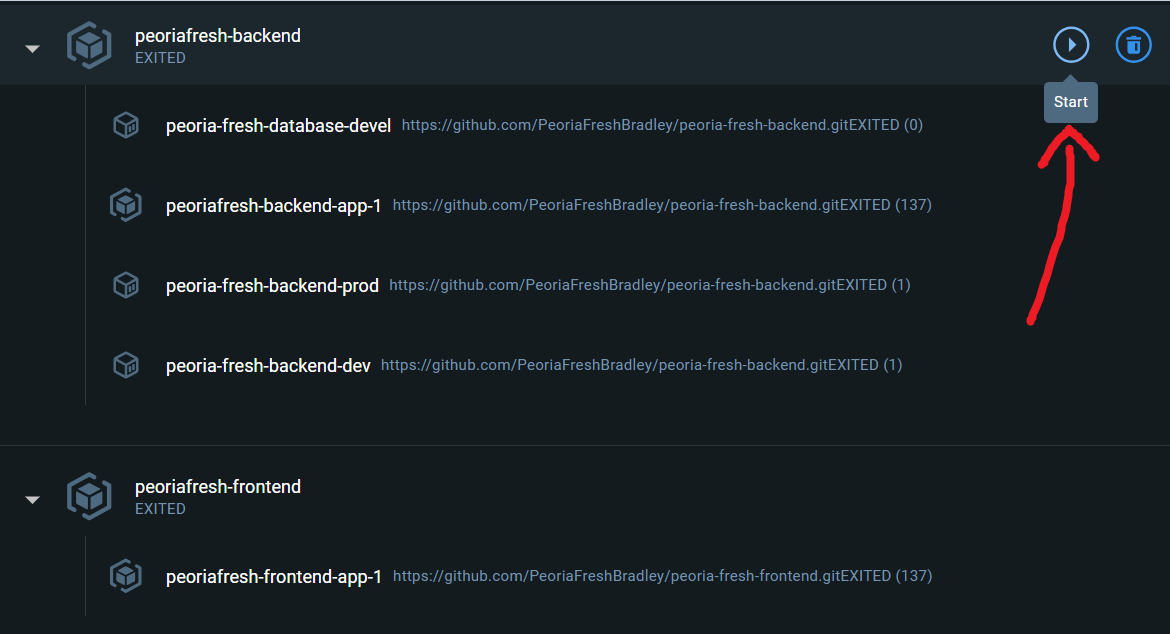
1. Go to your cloned GitHub Repository for the frontend/backend and click the Green “Code” Button. A drop-down menu will appear with an HTTPS link, copy that link.



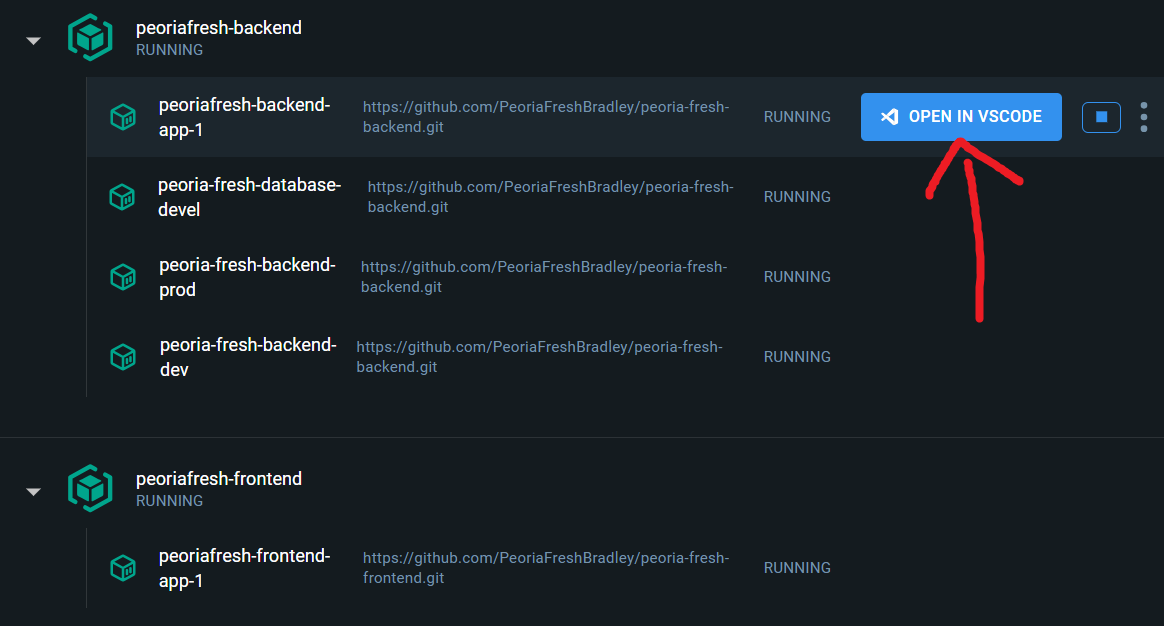
1. Paste that link into this box and make sure that Visual Studio Code is selected for your IDE



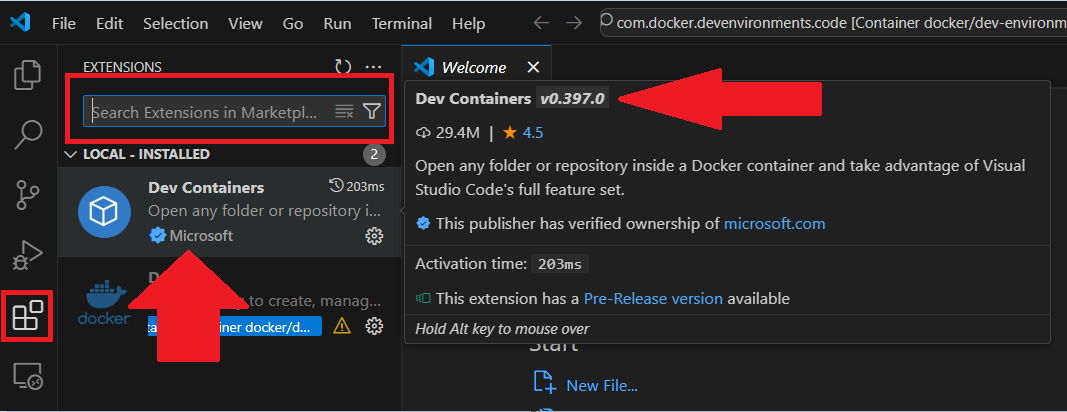
1. Click continue and wait until the process is done. This may take a little while.
2. Follow steps 5-9 again but with the other end of the database (if you just did the frontend, then do it for the backend and vice versa)
3. Wait about 5 or 10 minutes to give the Dev Environments a bit of time to process. If you jump into them too soon after installing them, you will need to restart the process starting at step 5.
4. Click the start button left of the Environment’s header for both the frontend and the backend



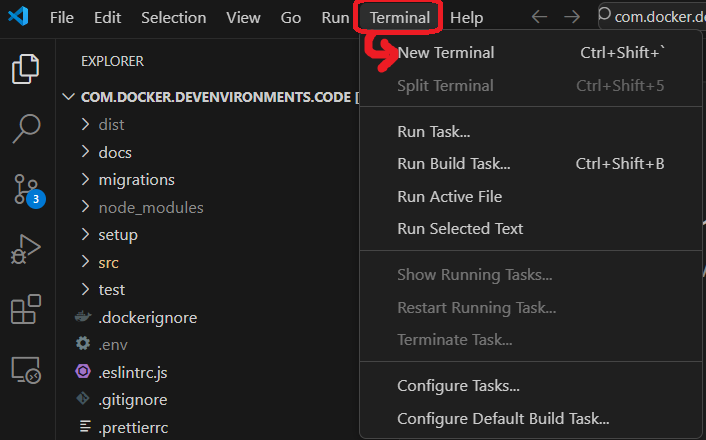
1. All of the Environment apps should turn orange and then eventually green
2. On the apps labeled “app-1,” click the blue button that says “Open in VSCode”



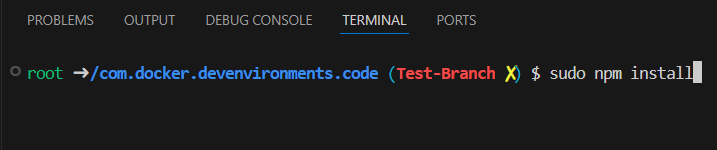
1. Once Visual Studios is open, click this icon on the left side of the screen and look up “Dev Containers” and make sure to install it.



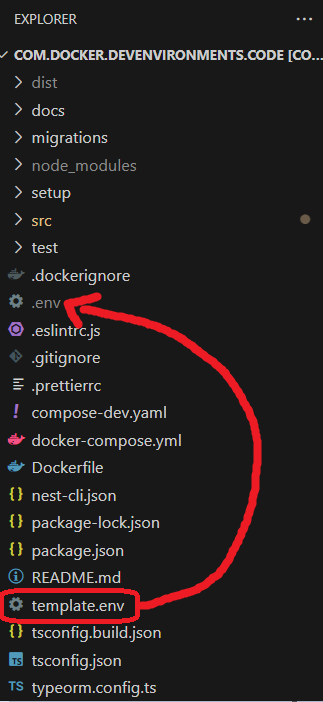
1. After that, on both sides (the frontend and backend) click the button on the top ribbon that says “Terminal” and then click “New Terminal.” This will open a new terminal at the bottom of VS Code



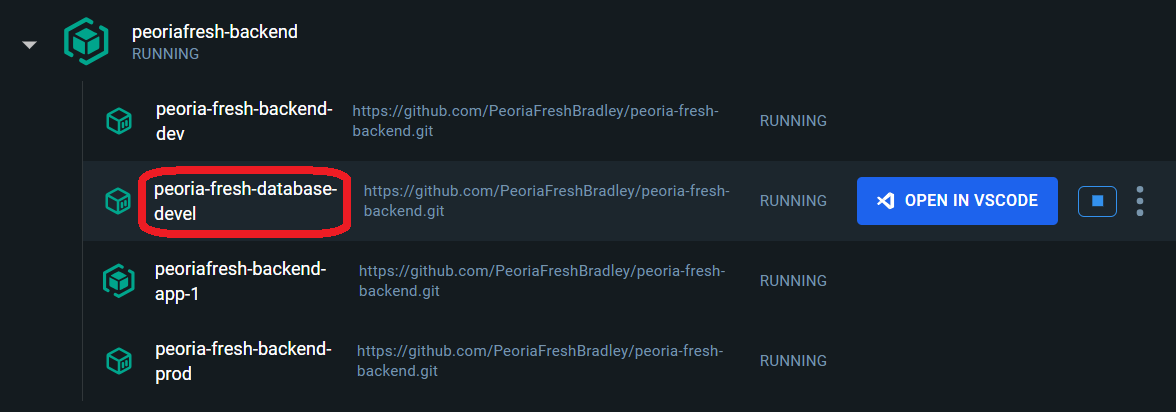
1. For both the frontend and the backend, type “sudo npm install” and wait. It may take a while.



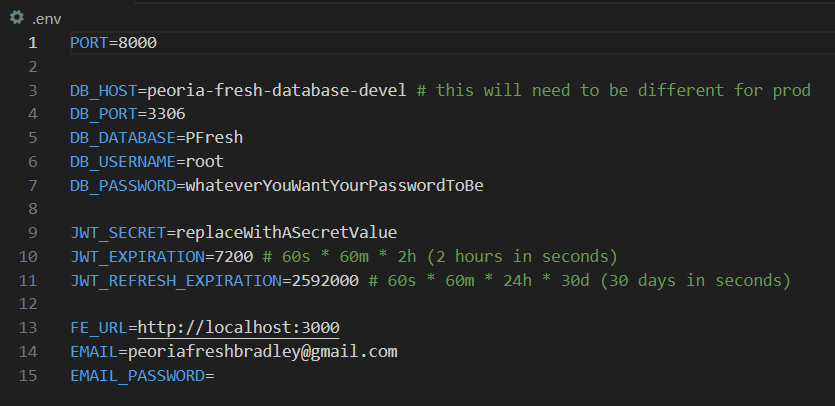
1. After that process is done for both ends, go to your backend window
2. On the explore tab on the left side of VS Code, copy the file called “template.env” and paste it. Rename it as just “.env”



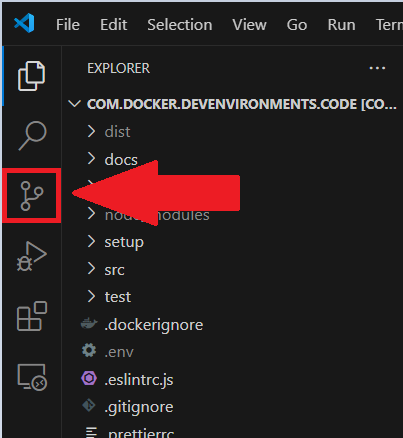
1. In your .env file, change the DB\_Host line to match the name of this file from Docker Desktop:



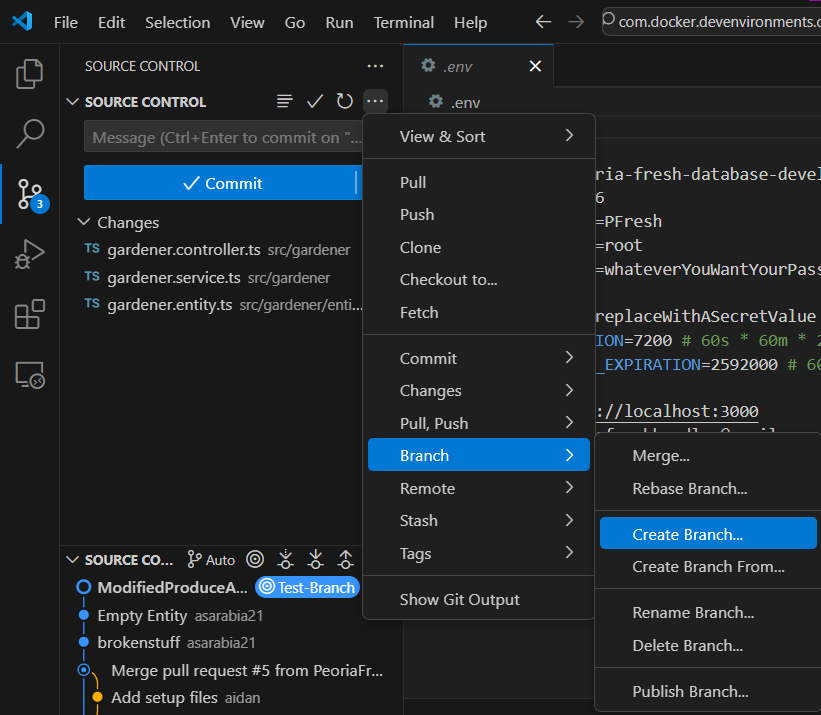
1. Your .env file should look like this image below (excluding the change made to DB\_Host), however, you can change your Database Password (or DB\_Password) to be whatever you want it to be. This password will be used later when we’re setting up HeidiSQL.



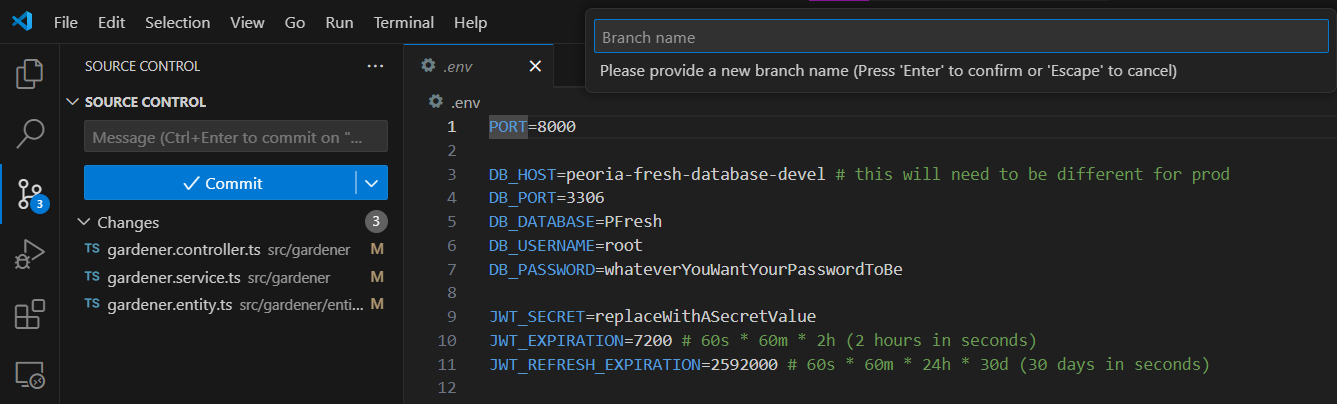
1. The next step is ***extremely important***, we’re going to be making a new branch for both ends. This is a GitHub term that refers to basically copying all of the code into its own section so that you can modify it. If you mess up this copy, you can go **BACK** to the original (referred to as “main”) and then make another branch. This is an ***extremely important*** safeguard if you mess up your code and have no idea what is wrong. Only one person needs to make the new branch and then everyone else can join.
2. On the far left ribbon of VS Code, click this icon:



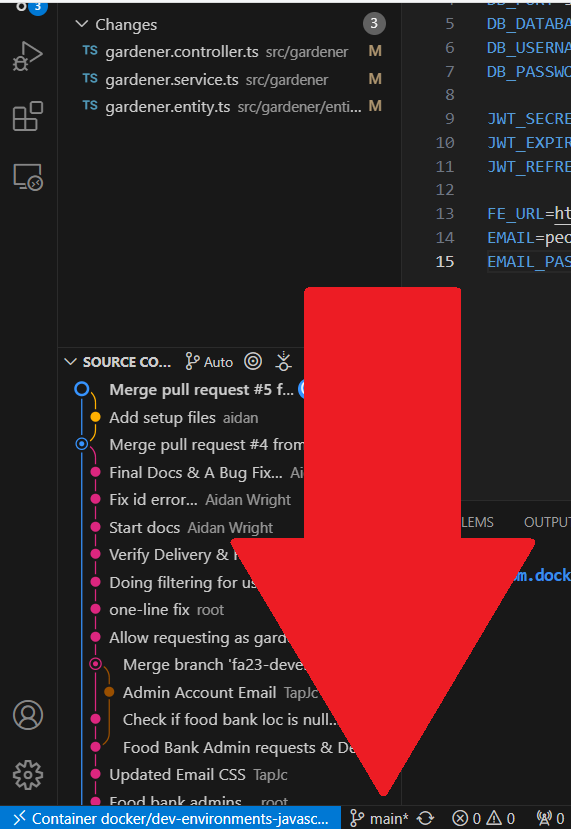
1. From there, click the 3 dots next to the second time Source Control is listed. Go down the pathing of Branch > Create Branch



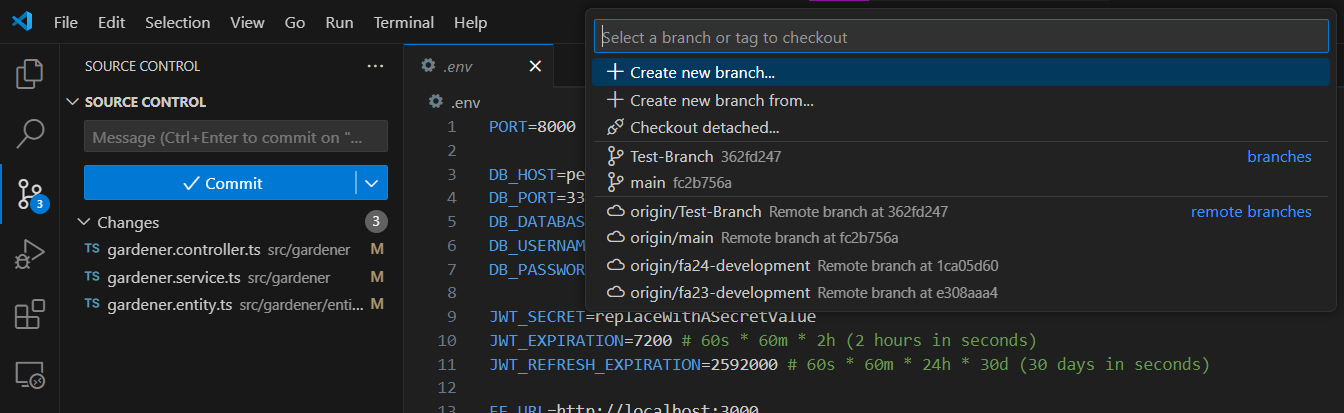
1. You will be prompted to give it a name at the top. Something like “Test-Branch” will suffice:



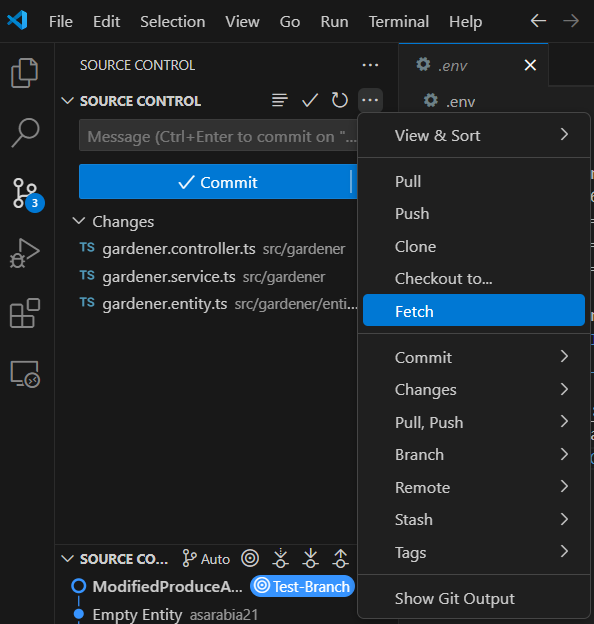
1. Only one person needs to make this new branch, once it is created your team members can switch over to it. To do this, click on the bottom button that says “main”



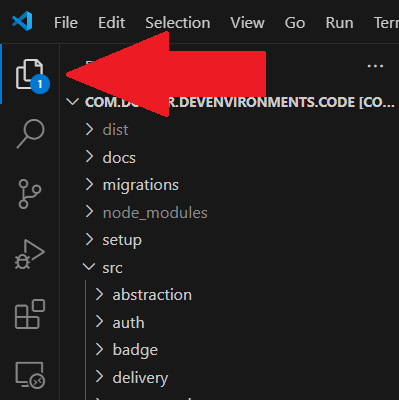
1. A pop-up at the top of the screen will appear and then you will select the new branch you created:



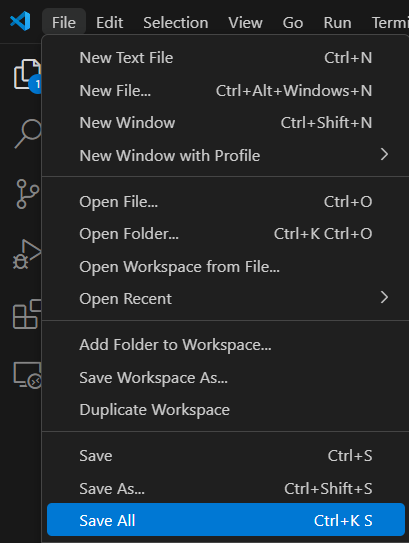
1. If you do NOT see the new branch, go back to the three dots next to source control and then click “Fetch.” After it’s done loading, repeat step 27 to see if it is there.



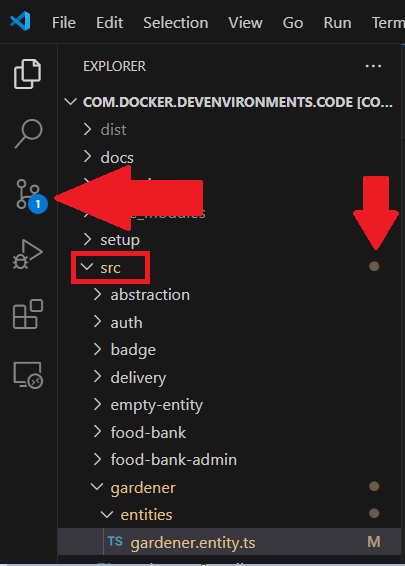
1. Make sure to create a branch for BOTH ends and make sure that every member of your team joins that new branch for BOTH ends. From there, people should be able to start modifying the code in any way they please.
2. To share changes made on one device to another, you need to know how to PUSH and PULL changes. These are both GitHub terms for sharing and downloading changes other people have made.
   1. A small tip, before modifying any code on your end, make sure you PULL everyone else’s changes first. If you start modifying a page someone else has already edited and PUSHed, when you attempt to PUSH your changes you will likely get an error message. If you proceed with the PUSH, ignoring the error message, the other person’s changes will likely get deleted. You will learn how to PULL code shortly.
3. Make sure you are in the correct branch BEFORE PUSHing, you can see which branch you’re in by following step 26
4. PUSHing code - Whenever you make a change to the code you will see a number appear next to this page icon, this indicates the number of pages you’ve changed.



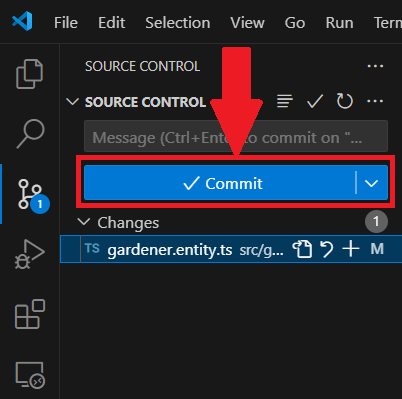
1. To PUSH those changes to other people, the first thing you have to do is click File > Save All



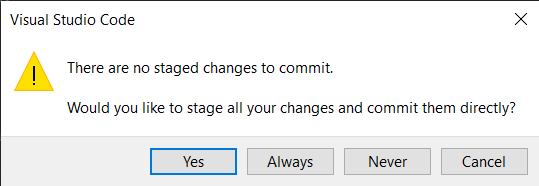
1. Once you do that, the source control icon on the left will update to have that number instead. Additionally, any pages you’ve changed will now highlight yellow



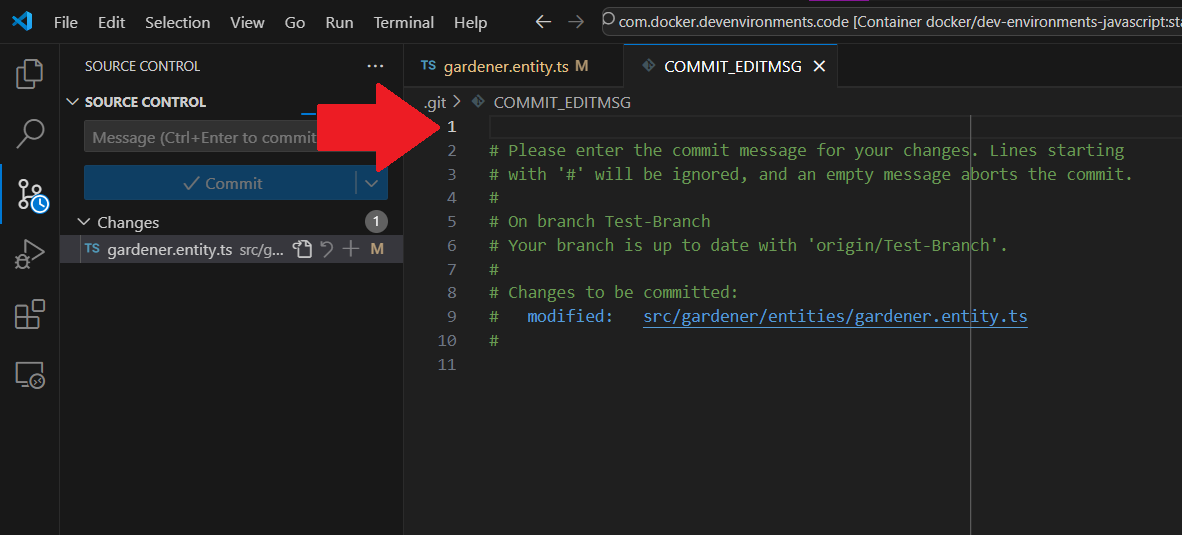
1. Click the source control icon and then you’ll see this screen. To PUSH your changes, click the blue “Commit” button



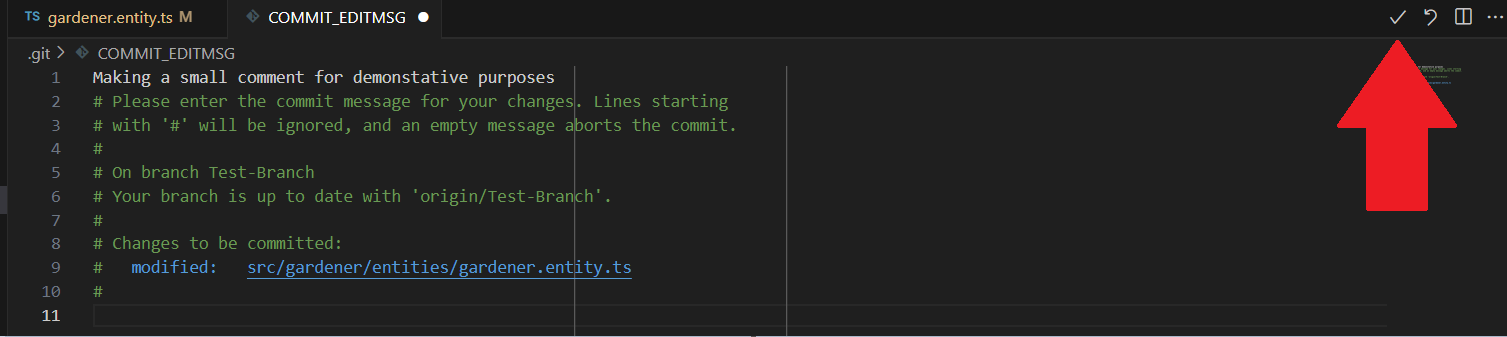
1. You will be prompted with the following menu. If you want to push every change you make, click “Yes.” This makes the changes “staged,” which means that they’re ready to be PUSHed. If you only want to push specific changes, click “Cancel” on this menu and click the + icon on every change under the blue “Commit” button. This will specifically “stage” all of the changes you’d like to PUSH



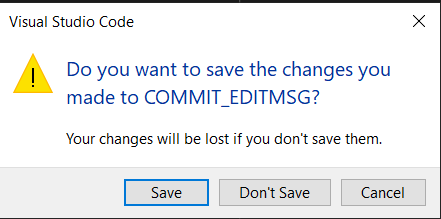
1. From there, you will be prompted to write a description for every change you made that you are PUSHing. You will write it in the area above the green text.



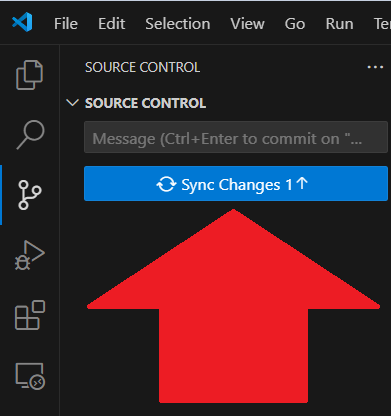
1. After you write that out, click the white check mark at the top right of this page



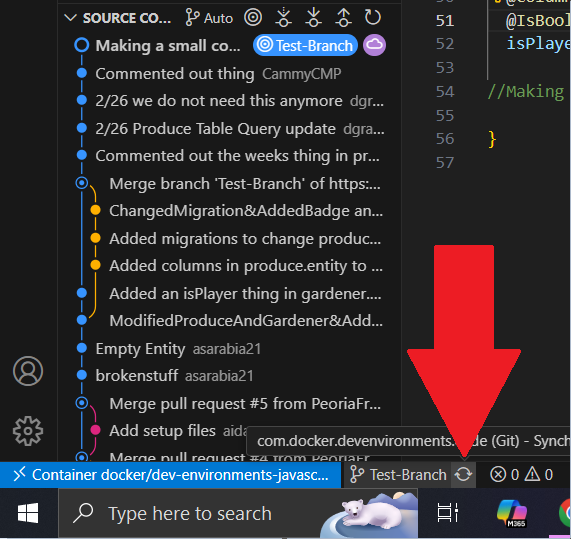
1. After that you will be prompted to save your update message, click “Save”



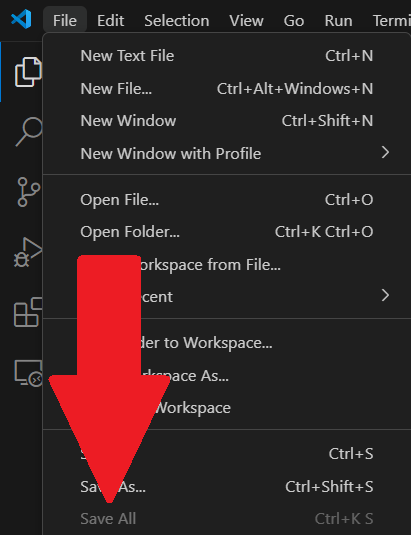
1. The source control menu should now have a “Sync Changes” button, press that, press “Ok” when prompted, and your changes will be PUSHed.



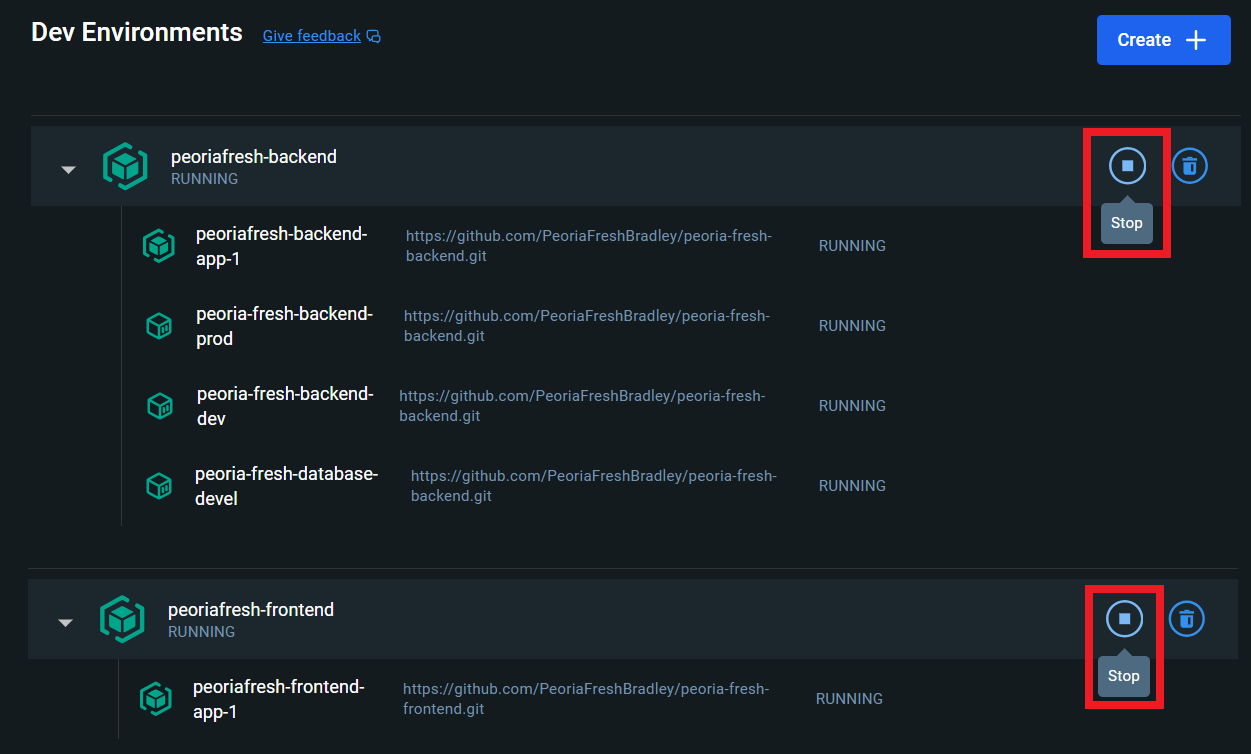
1. To PULL those changes on a different device, click this button next to the tab where you can see what branch you’re on



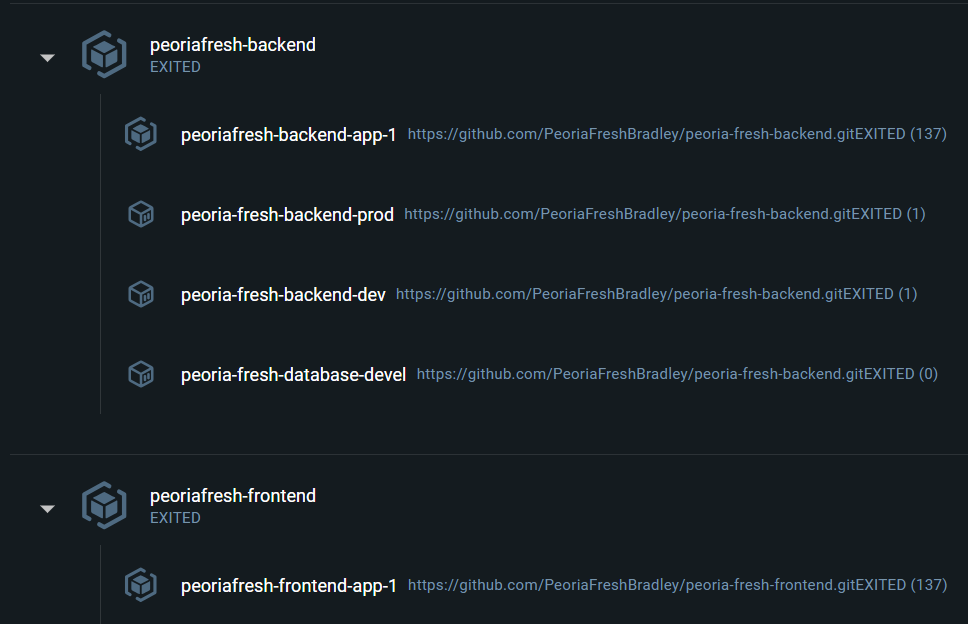
1. If the changes do not appear, try clicking the fetch button shown earlier. If the changes are still not syncing, make sure you are in the correct branch and that the person who PUSHed the changes was in the correct branch
2. With all of that said and done, you should be ready to modify the code! Now we’re going to show how to close VS Code and Docker Desktop completely.
3. First, make sure to save your work by clicking the File tab and then Save All. Note: You don’t need to push your code every single time you close your code; we recommend that you do that when whatever you’re working on is done or if you need to PUSH it so you can work on it on another computer.



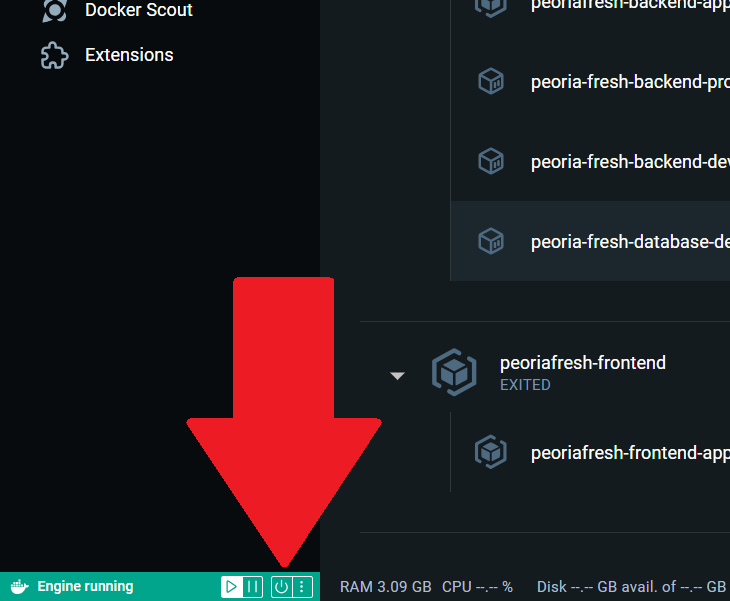
1. Next close the tabs for VS code so that you only have Docker Desktop open.
2. After that, click the “Stop” buttons for both the frontend and backend. The buttons are to the right of the overall containers as seen below:



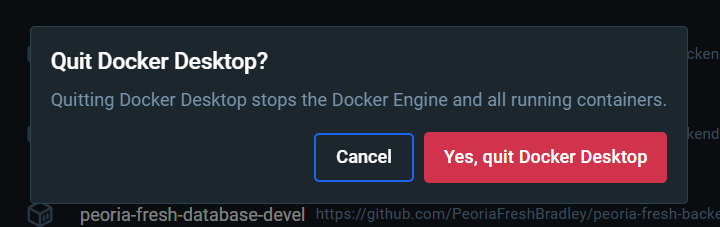
1. When they’re closed, the containers should look like this:



1. In the bottom left corner, click the power button icon



1. It will ask if you if you want to close Docker Desktop, click the red button that says “Yes, quit Docker Desktop”



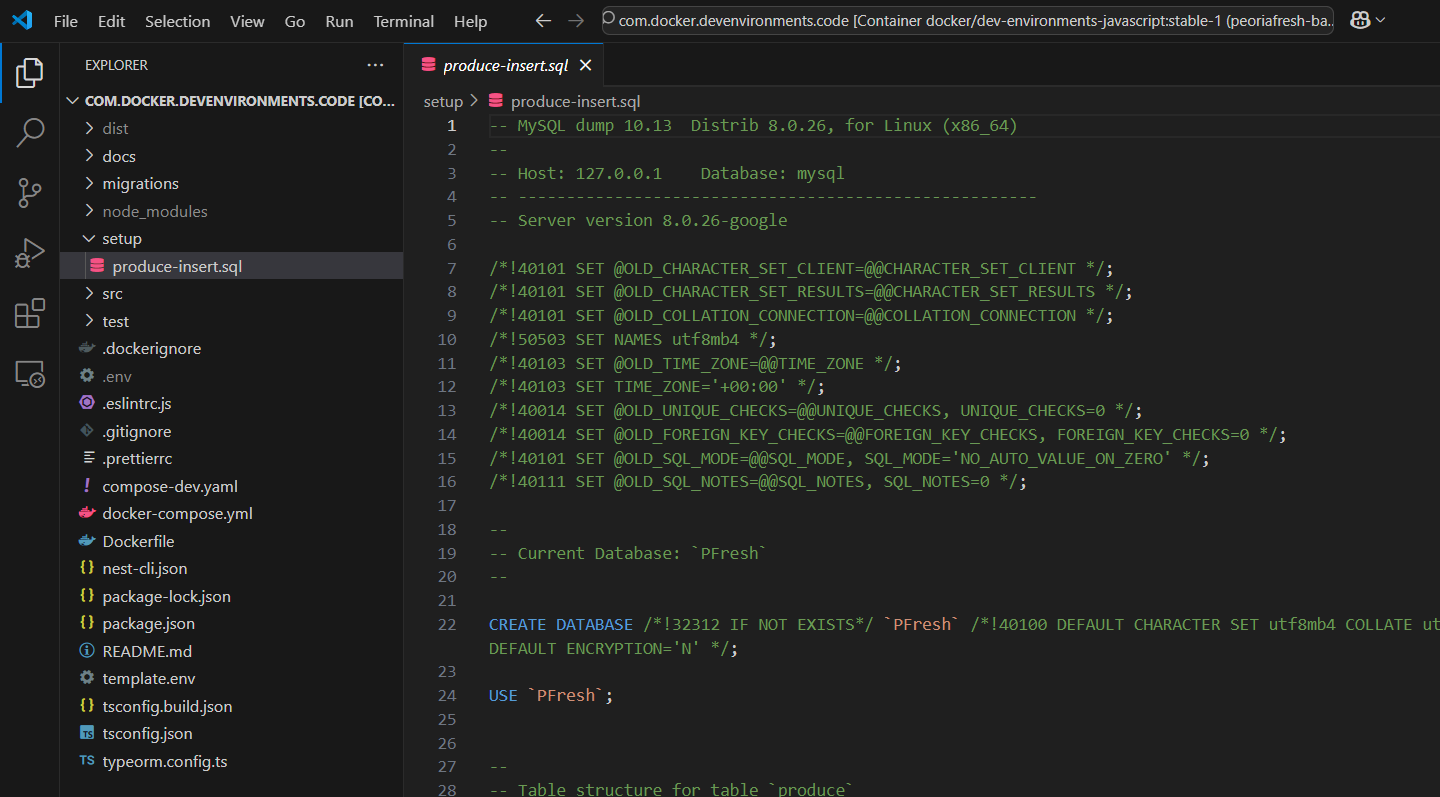
1. From there, Docker Desktop will completely shut down. If you just click the X in the top right corner to close the Docker Desktop Tab, it will still be running a bit in the background and use some of your computer’s resources. Doing it the way we’re telling you to will completely shut it down.

## 

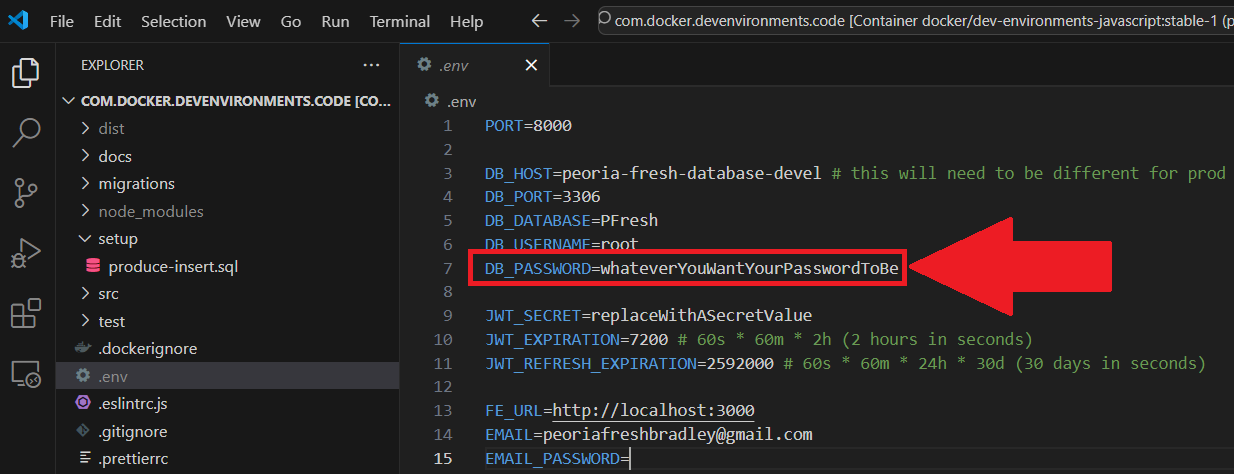
## HeidiSQL Set Up:

* Steps:

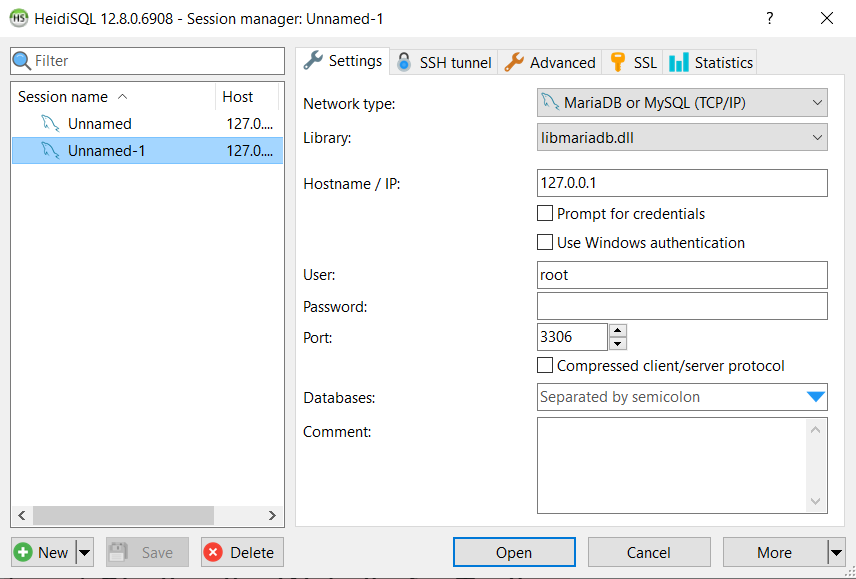
1. First, open the backend and locate the produce-insert.sql file. It’s in the produce folder and looks like this:



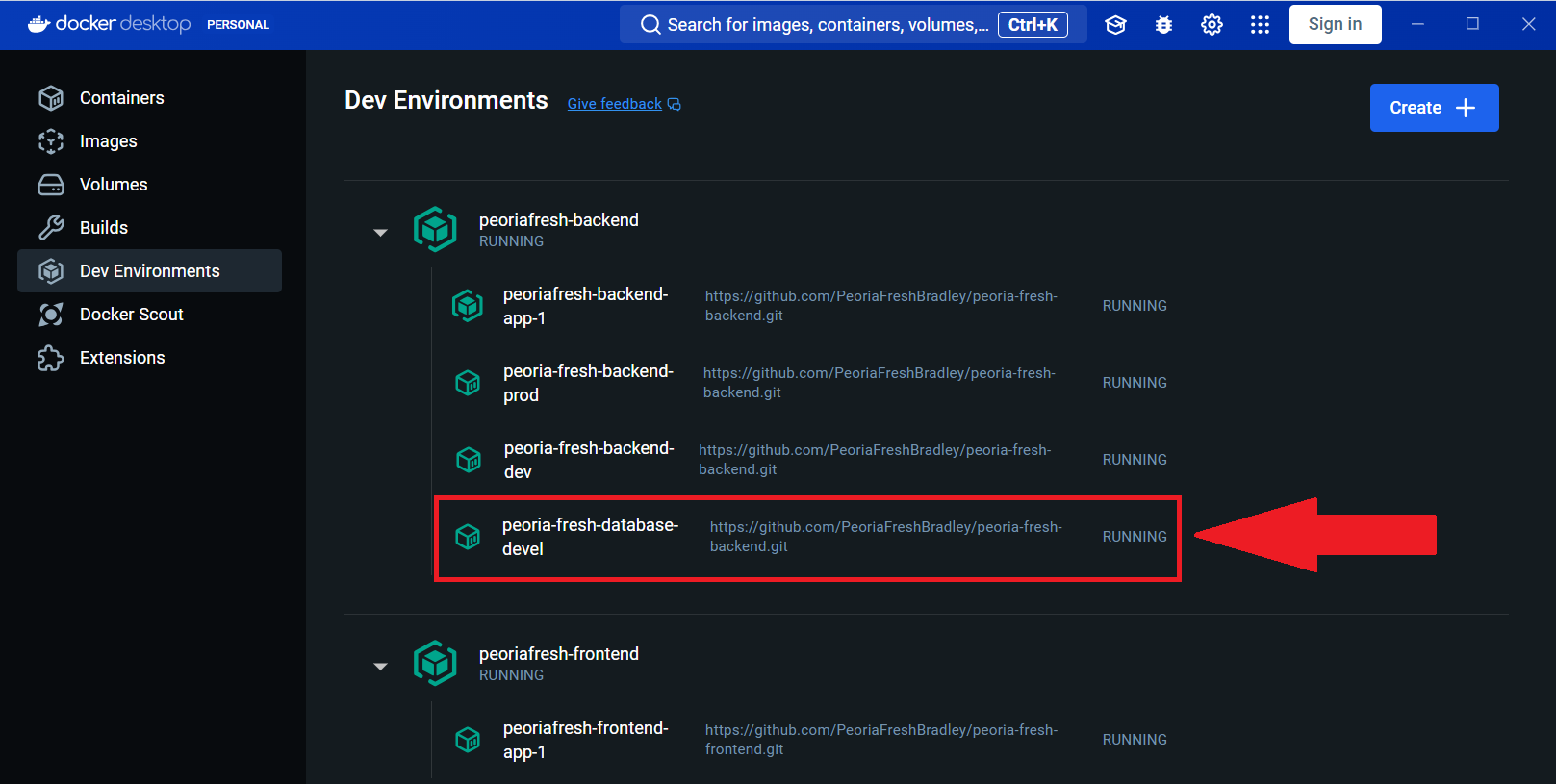
1. Additionally, you need to locate the .env file that you created earlier. It shouldn’t be in any folders. It looks like this (ignore the red box, that will be used in the next step):



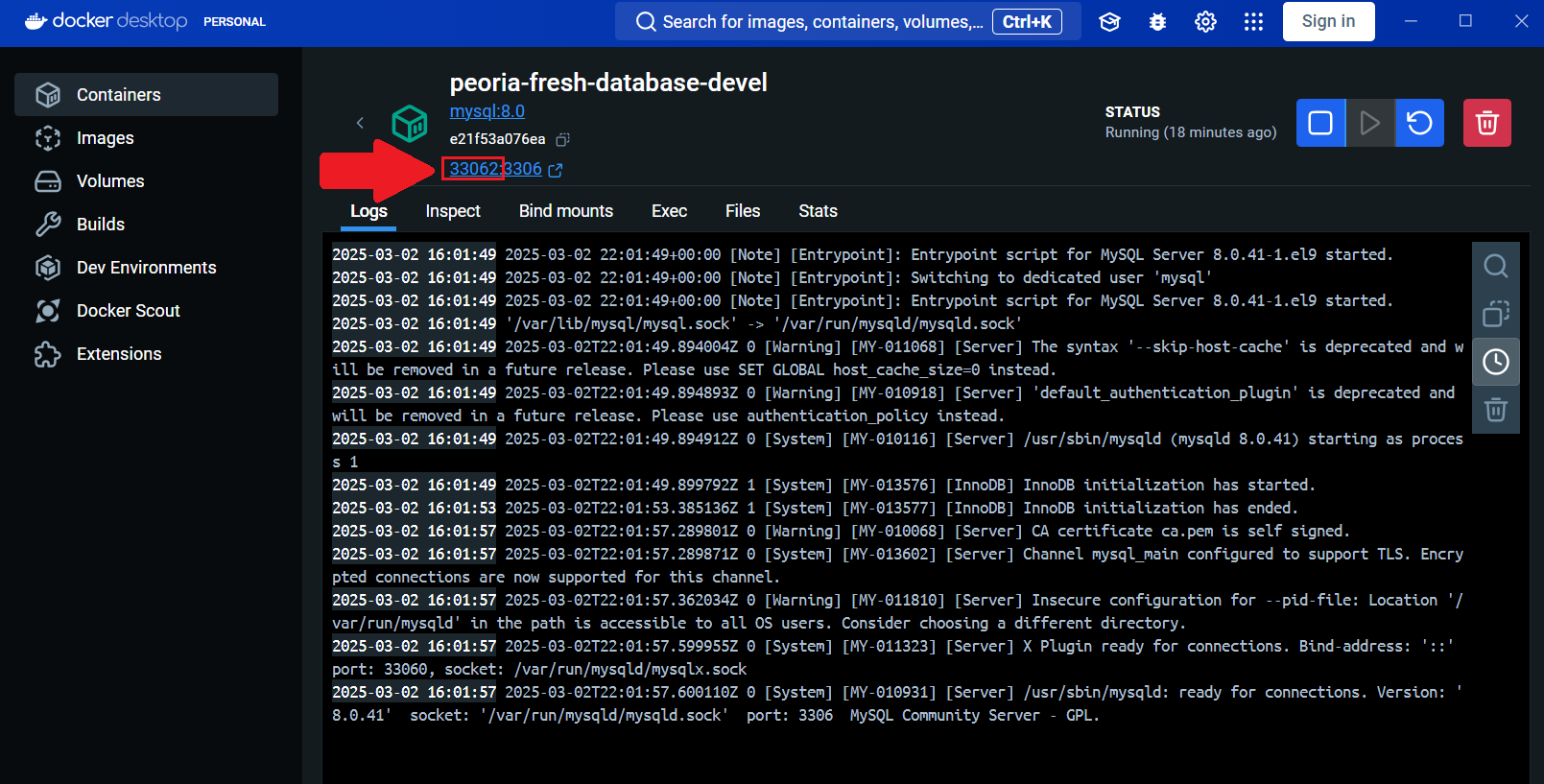
1. Open HeidiSQL and it should look like the following image. For the username, type “root” and for the password, type in what you have for DB\_PASSWORD from the .env file.



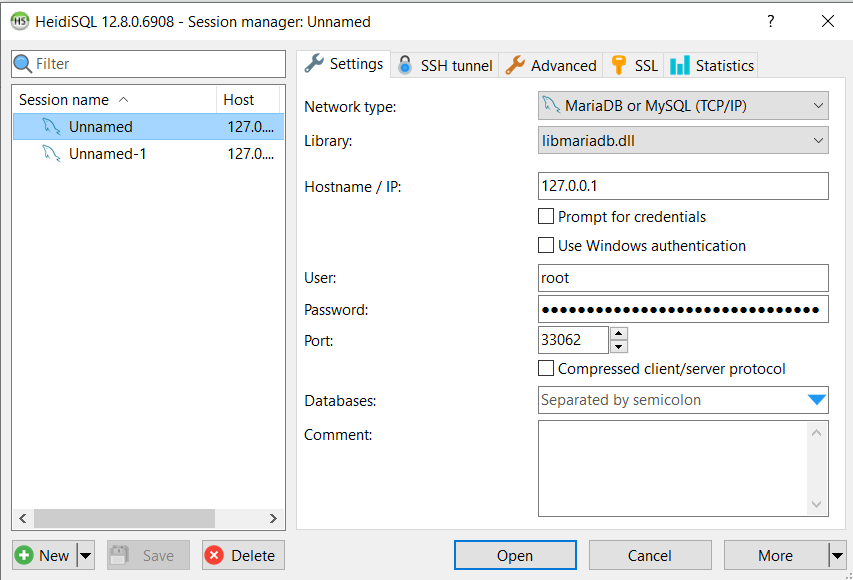
1. Next, we’re going to find the correct port number to put in the port section. On Docker Desktop, open the backend file that has the word “devel” at the end of it. Specifically, just click on text for it, we are NOT opening it in VS Code:



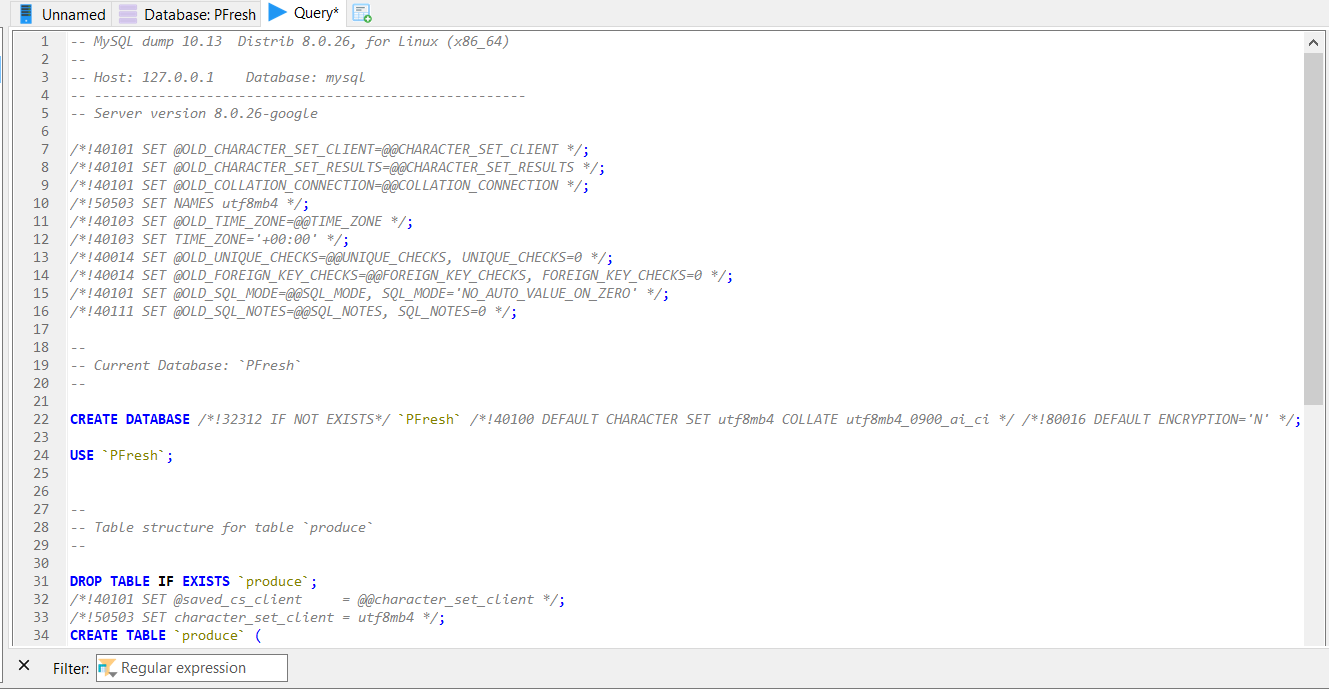
1. Docker will now open a new page. On that page we’re going to copy the boxed number in the image below, for me it is 33062, but it could be something different for you:



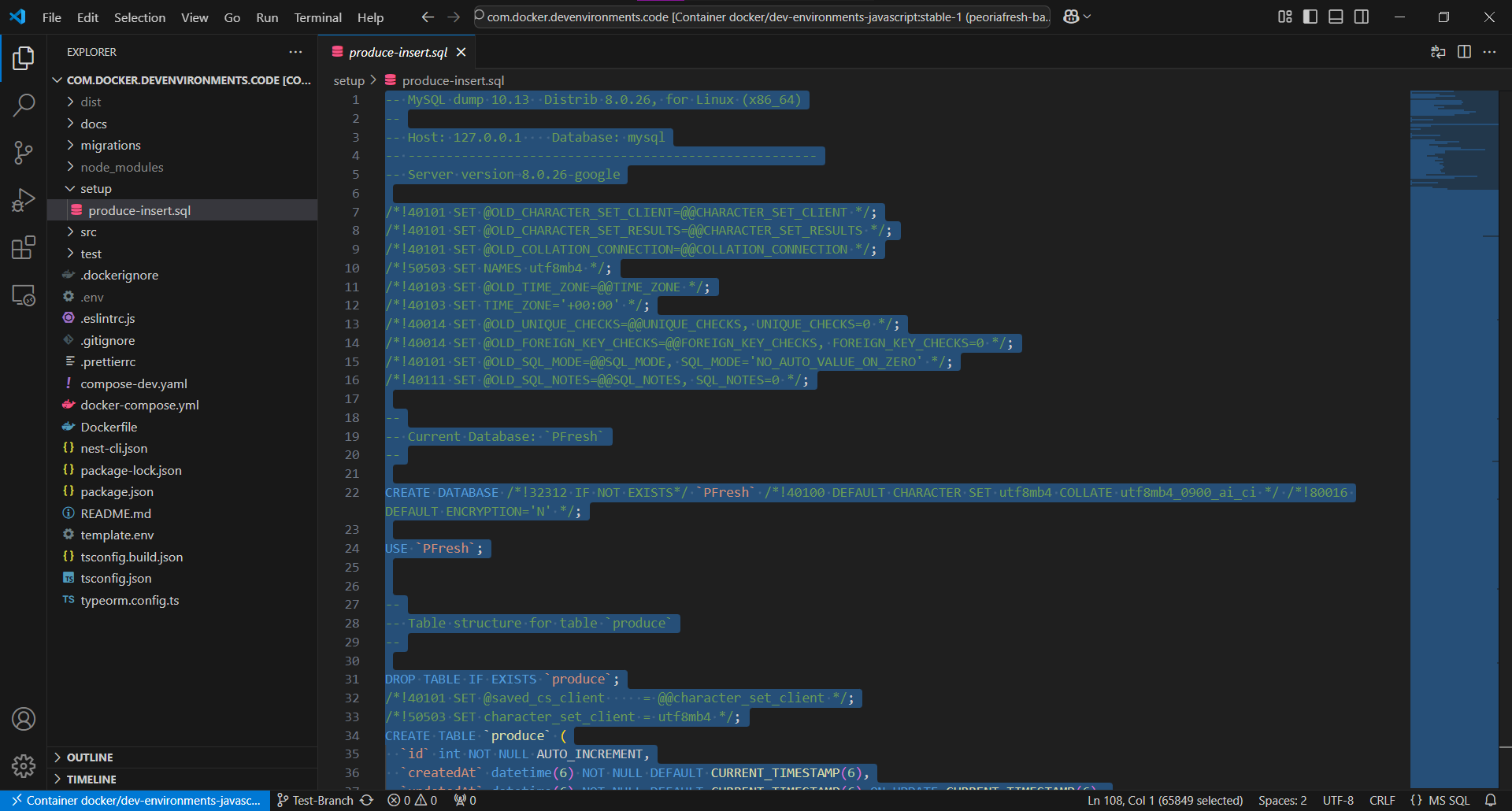
1. Put that number into the port section in HeidiSQL:



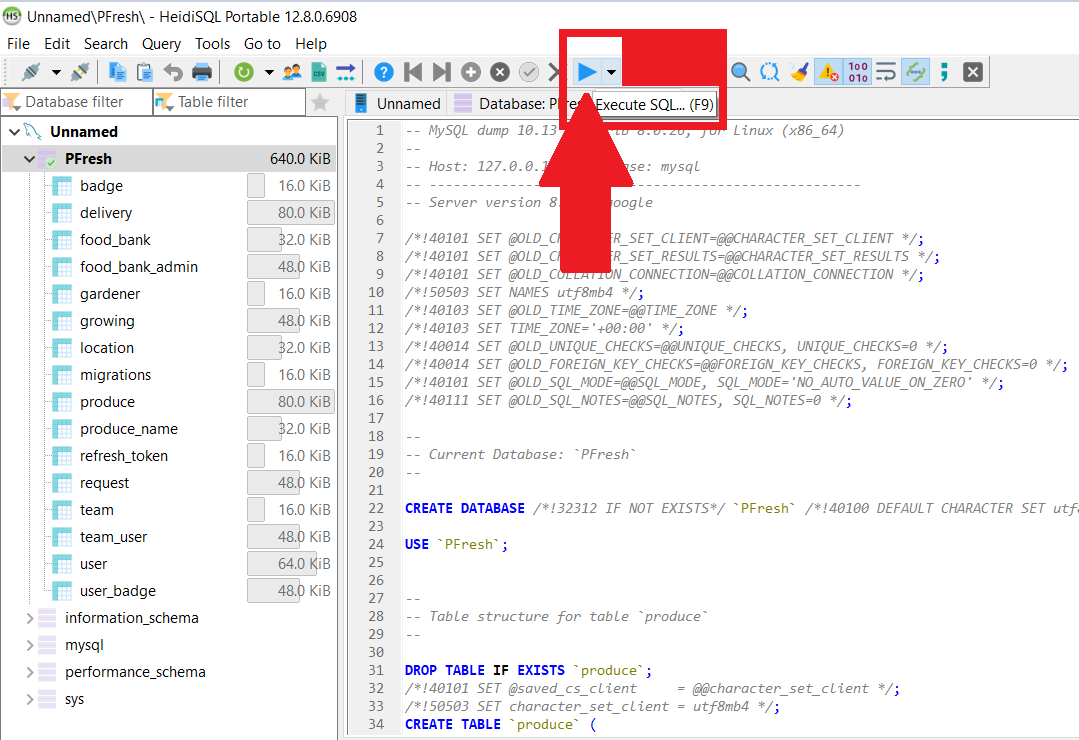
1. Click “Open” and then navigate to the “Query” tab. It should be completely empty for you.



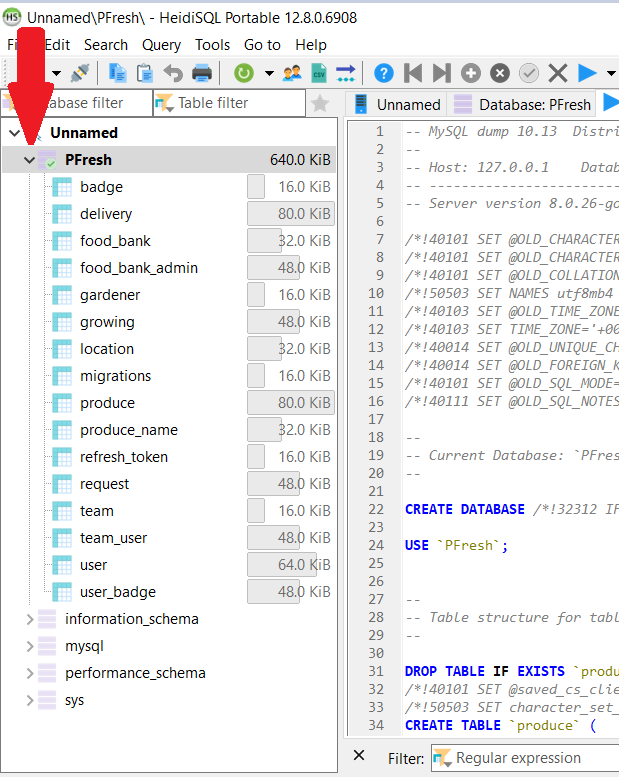
1. Go back to the produce-insert.sql in VS Code, click into the document, and press (ctrl + a), that should highlight the entire document. Copy the text by right clicking or pressing (ctrl + c).



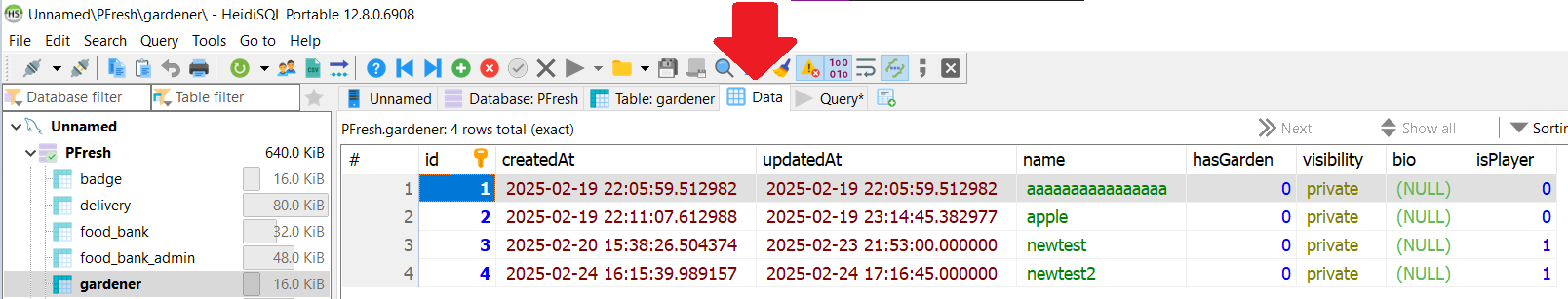
1. Paste this text into the Query page on HeidiSQL
2. Click the blue arrow button. This executes the SQL code

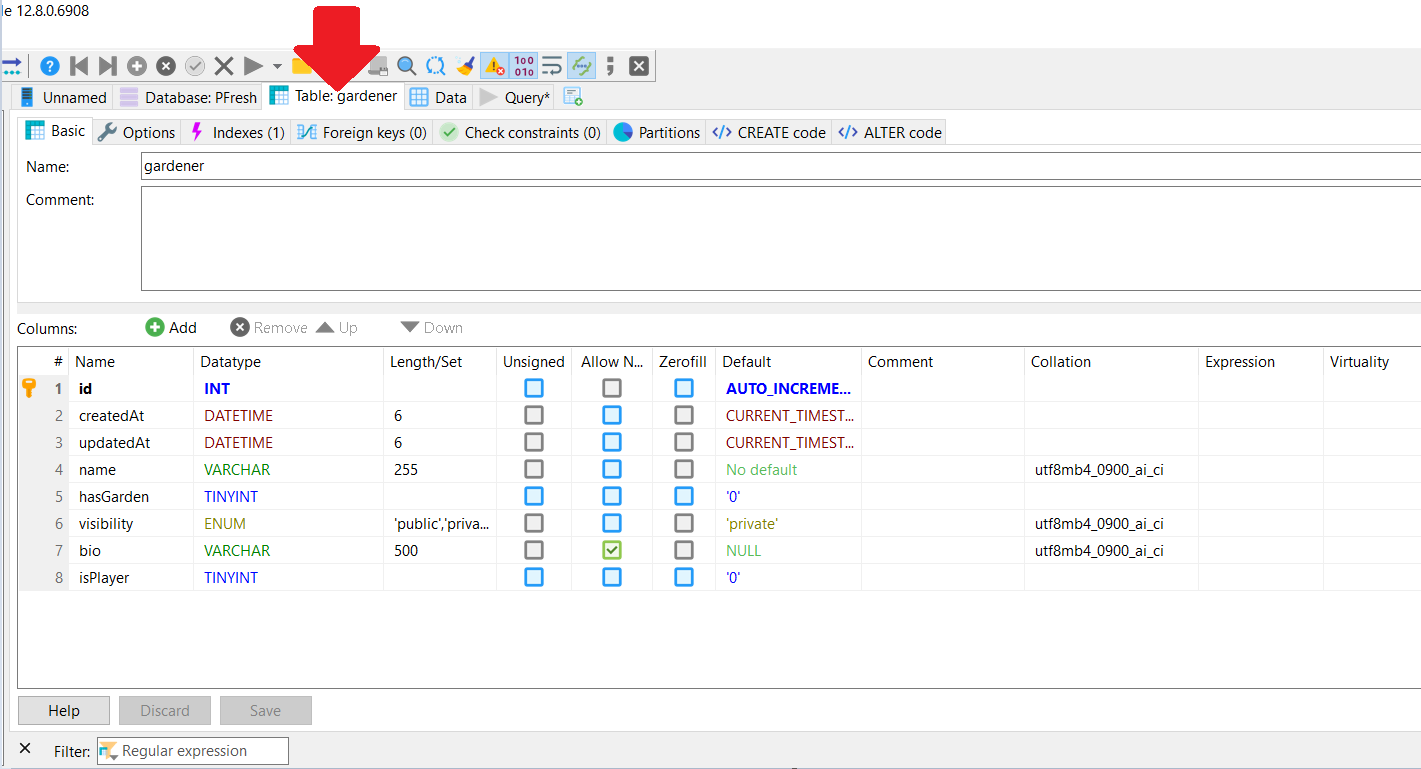


1. This will populate the entire database and then you should be done setting up Heidi. The following steps are a few tips you can use to view or manipulate the database on Heidi. These changes will ONLY affect your device’s database.
2. To view entries in a schema, click the drop down arrow for the database and select the table whichever schema you want to view. For example sake, we’re going to show the gardener schema.

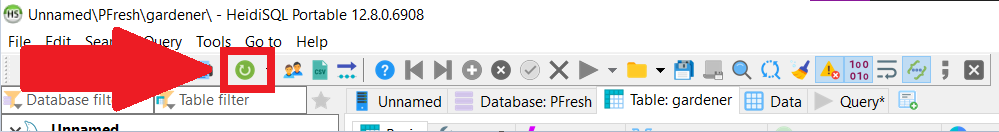


1. After you select the schema you want, two new tabs will appear. The “Data” tab directly shows you the data inside of that schema. The “Table: Gardener” tab will show what variable





1. In the “Table: Gardener” tab, you can directly add or change columns to this specific schema. You can click directly on the boxes to change them if you need to.
2. If you make changes to the database through the frontend, you may need to click this refresh button:

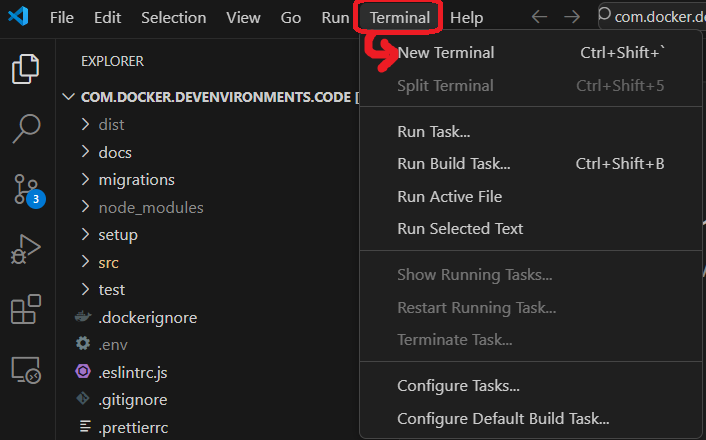


# 

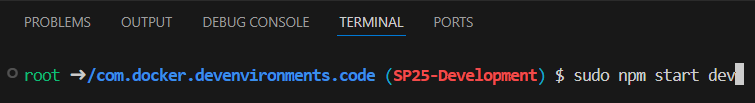
# **Starting the Website for Testing Purposes**

* When you want to test the changes you’ve made in the frontend and backend code, you’ll want to run the website on your local device. Here are some steps on how to do that:

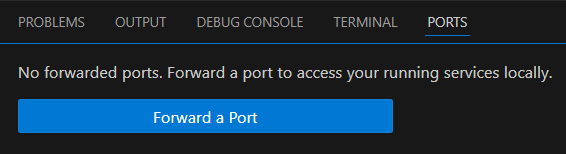
1. First, you need to have both the frontend and backend open in VS Code.
2. Next open a terminal in both of them like how we did when we were setting up VS Code:



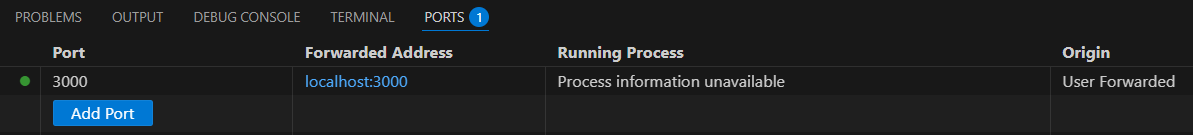
1. In both terminals, you’re going to type “sudo npm start dev,” this will cause the website to start running



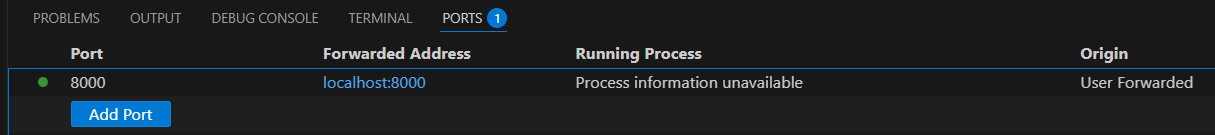
1. If you get an error, the terminal will directly tell you what is wrong. If it’s a frontend issue, the terminal there will tell you what the error is. If it’s a backend issue, then the terminal in the backend will tell you what is wrong. If you have no errors, continue forward.
2. Next, click the “Ports” tab right above the terminals. We are going to “Port Forward” to two different ports depending on which end you are in
3. Click the blue “Forward a Port” button on both ends



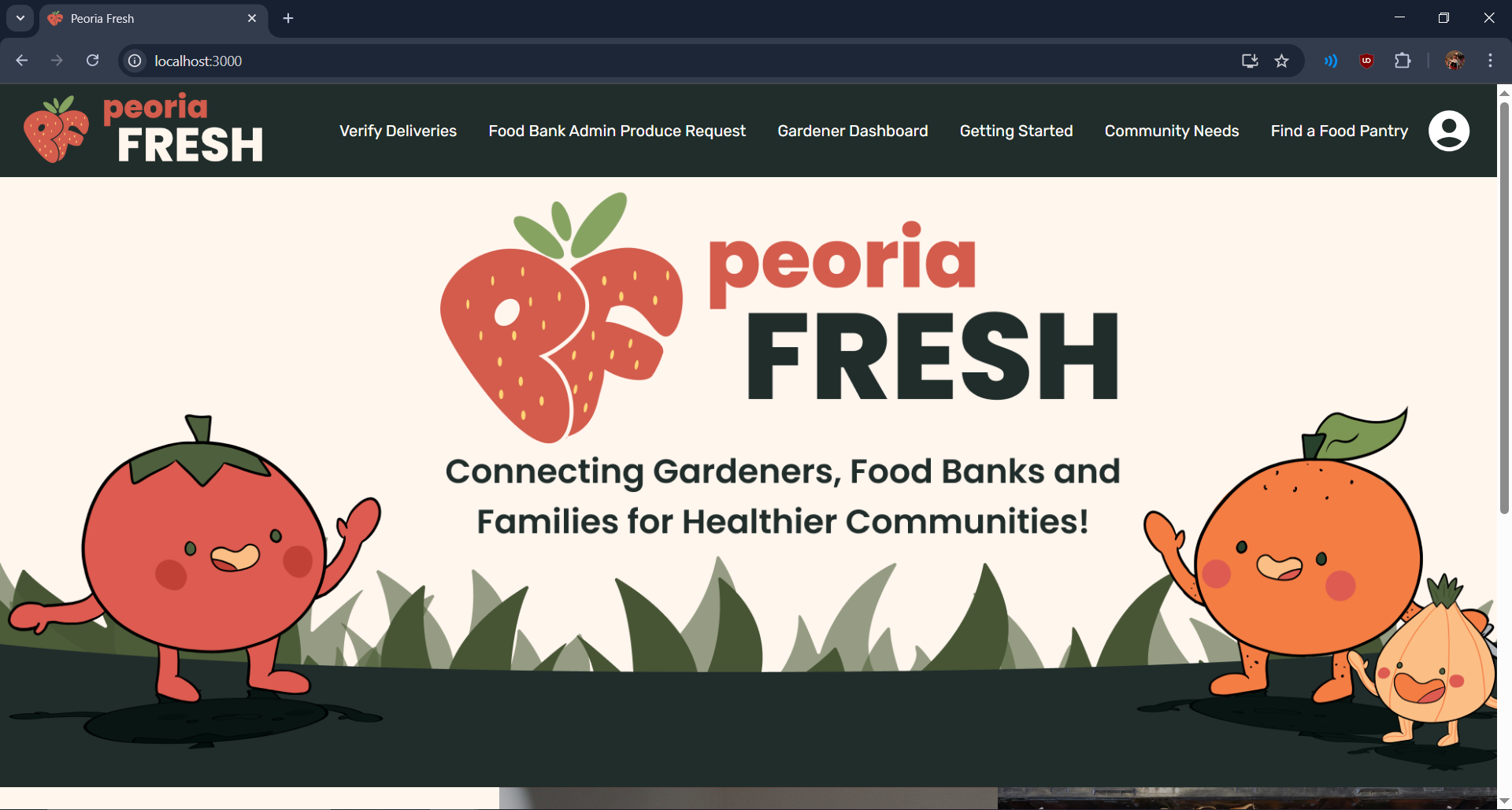
1. In the frontend, type into the Port column “3000”



1. In the backend, type into the Port column “8000”

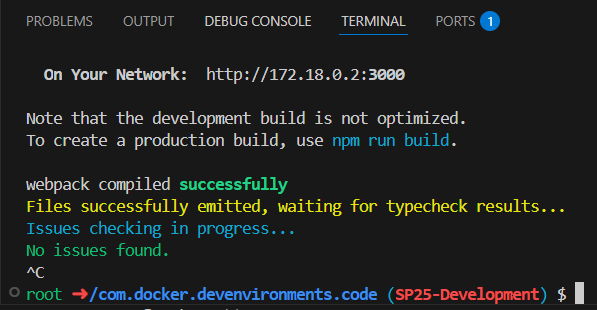


1. Next, in a browser of your choosing, type “<http://localhost:3000/>” into the search bar. It should open up the Peoria Fresh Website on your local device:

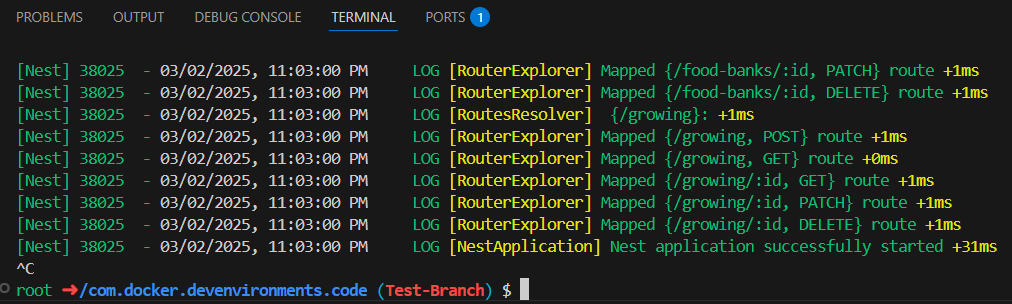


1. When you’re done testing things, go back to the terminal on both ends and press (ctrl + c). This shuts down the local hosting. It will look something like this:

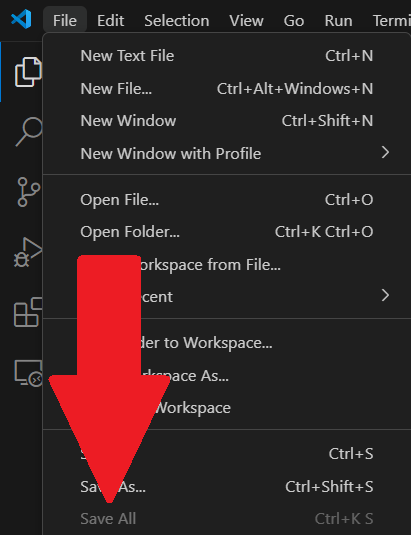
(frontend example)



(backend example)



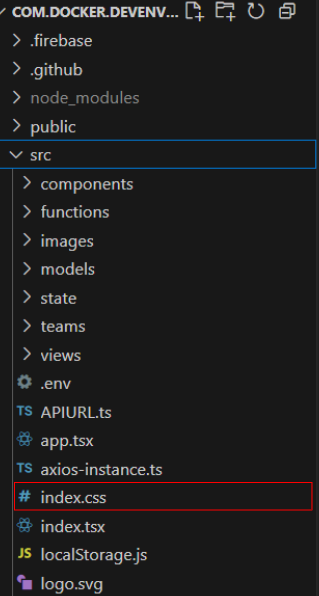
1. While the tab on your browser looks like it will still work, it won’t fully load any pages.
2. If you make any changes to the frontend code and you want it to appear in the locally hosted version in the browser, click the File tab and then click Save All. All changes you make in the frontend code will update immediately to the UI in your browser AFTER you do this.



1. If you make any changes to the backend code and you want it to appear in the locally hosted version in the browser, click the File tab and then click Save All. After you do this, shut down the backend code with (ctrl + c) like in step 10. Then type “sudo npm start dev” in the terminal to restart the backend for the locally hosted version. For the backend specifically, you MUST restart the code in the terminal after saving your changes.

# 

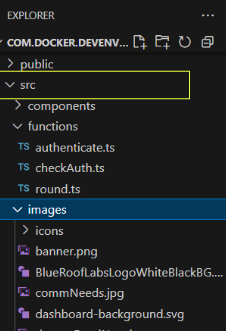
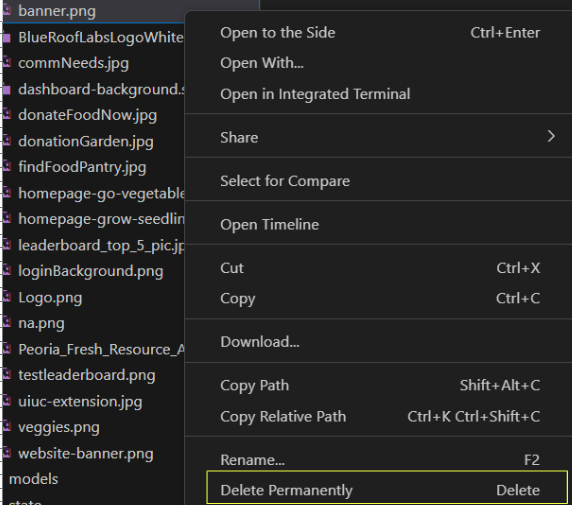
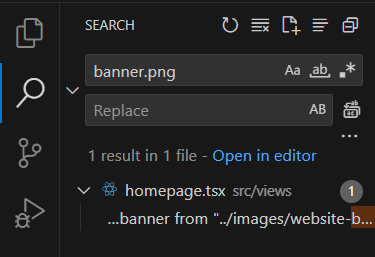
# **How to Change Colors/Themes**

* What is this: most of the main colors on the site are sourced from one file. This makes it simpler to change colors throughout the website by changing one color code.
* In the file path go to src/index.css
* 
* At the top of this file, a number of color variables will be listed. Here are a few of the aspects of this file:
  + Comments above the variable will describe what each of the colors do throughout the site.
  + The –<name> is the variable name. If you are to do a deeper dive into the frontend, you will find instances of this name to refer to the color listed.
  + Colors will be listed as a hex code (ex: #FFFFFF) or an RGB code (ex rgba(22, 33, 44, 0.70)). These values are what you can change which will thereby change the color where it appears on the site.
  + Finally, the comment next to the color value lists a “default.” This is simply what the PeoriaFresh site used originally.
    - These defaults are also hard coded into the front end as a fallback color. If any colors are changed but formatted incorrectly, the color will return to this default.
* Further color changes such as separating colors or more specific color changes not made possible by the index.css file will require delving into the frontend further. Users who intend to do this will need further knowledge of colors to do so. Please check the documentation for further assistance.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Hex Code | Color | Description |
| --base\_site\_bgColor | #fff6ee |  | The background color for most of the website, main theme color |
| --base\_site\_MTColorLight | #fffbf8 |  | Used as text color for onset on darker backgrounds mainly |
| --base\_site\_activePopupColor | #ea7f3a |  | Used in request badge popup |
| --base\_site\_MTColorDark | #202d2b |  | Text color for lighter backgrounds and buttons on lighter backgrounds |
| --base\_site\_inactiveUIColor | rgba(44, 65, 44, 0.69) |  | Inactive UI elements, or inactive buttons |
| --base\_site\_buttonHoverColor | #50603b |  | Color buttons typically become when users hover over (usually confirm buttons) |
| --base\_site\_destructiveButtonColor | #a44433 |  | Color for buttons typically for cancelling or negative answers |
| --base\_site\_destructiveButtonHoverColor | #d35c4c |  | Color for buttons for negative answers when users are hovering over them |
| --base\_site\_backdropColor | #f0c6a6 |  | Layered background for certain tables (ex Leaderboard) |
| --base\_site\_backdropAccentColor | #eeba94 |  | Accent for the backdrop for certain tables |
| --base\_site\_light | #ffffff |  | Instances of the color white on the site |
| --base\_site\_dark | #000000 |  | Instances of the color black on the site |
| --base\_site\_neutral | #808080 |  | a couple buttons use this for a hover color |
| --base\_site\_neutralLight | #D3D3D3 |  | a couple buttons use this as a neutral color |
| --base\_site\_success | #008000 |  | Success messages, typically for donation pages |

# 

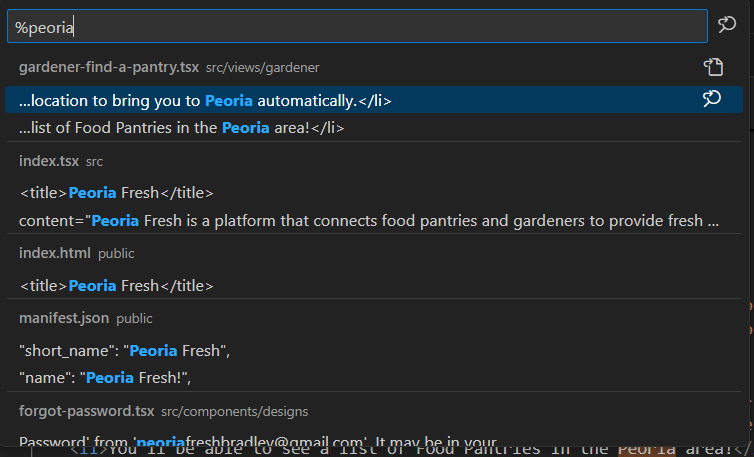
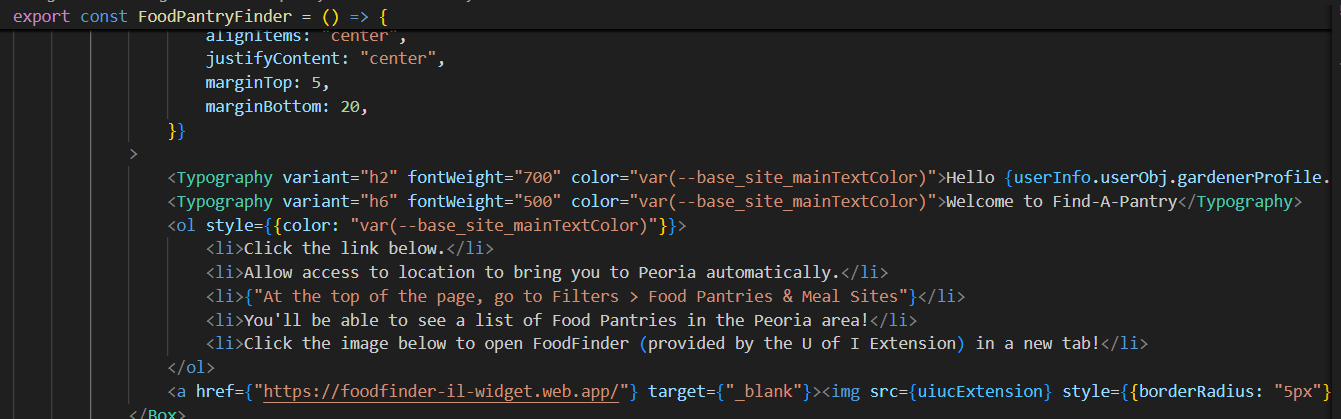
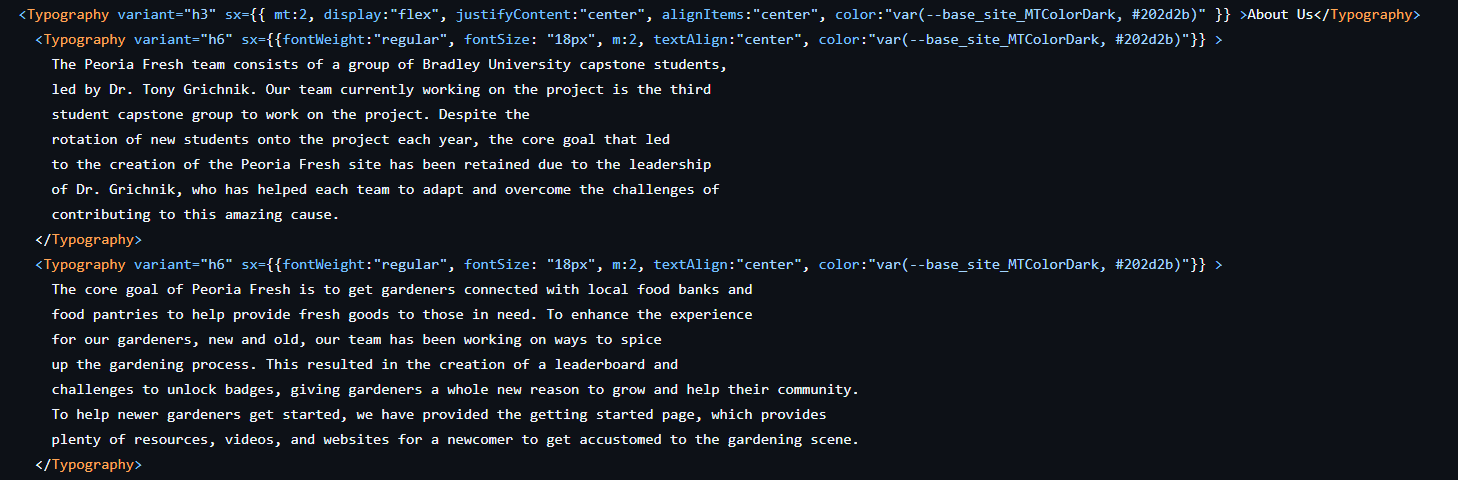
# **How to Replace Images**

* What is this: Images are easy to plug and play into the site if dimensions are consistent between old and new assets.
  + To find the image assets, go to src/images to see the image files used on the site
  + If you wish to replace an image where it appears on the site, you can simply remove the image and replace it with an image of the same file type and the same name.
  + Further explanation:
    - Image name: if you are wanting to replace the banner and logo images (these are the only PeoriaFresh specific images, you will likely need to replace these for branding), you want to keep them the same so the code can retrieve them. Below is a list of steps that you should follow for each image you want to replace if you wish to do it without touching the frontend code.
      * Ex: banner.png
      * Design a new banner with the same dimensions as the original banner.png (all image dimensions listed in table below).
      * Store the image as a png
      * Remove the original banner.png
      * Next, simply insert your new image into this file. Verify that it is a png and rename it to “banner” if you haven’t already so it is listed as “banner.png” in the file path.
      * If all of this is done correctly and the image is kept to the same criteria as the last, the frontend will be able to retrieve it and display it without causing formatting problems.
    - WHEN DOING THIS: keep in mind that image width and height (dimensions), name and file type must be the same as the original file to not cause issues. If you wish to do dimension changes it will require you to do a bit more work in the frontend to verify its function. If you are looking to do file type or name changes, you may need to change every instance of its import.
      * You can, for example, use the VSCode search and replace function to do this
  + Additions of extra images, image dimension changes, and anything else of the like will require you to make changes to the frontend code. Check the documentation of the frontend and look through where these images are located in the frontend if this interests you.

|  |  |  |
| --- | --- | --- |
| File Name | Width | Height |
| donationGarden.jpg | 2778 | 3161 |
| donateFoodNow.jpg | 2731 | 2822 |
| website-banner.png | 2400 | 800 |
| veggies.png | 2723 | 4096 |
| uiuc-extension.jpg | 300 | 296 |
| testleaderboard.png | 1792 | 1024 |
| na.png | 760 | 571 |
| loginBackground.png | 2048 | 396 |
| leaderboard\_top\_5\_pic.png | 1051 | 717 |
| homepage-grow-seedling.jpg | 1920 | 1280 |
| homepage-go-vegetables.jpg | 1708 | 1137 |
| findFoodPantry.jpg | 2579 | 3868 |
| commNeeds.jpg | 2516 | 2731 |
| banner.png | 2000 | 338 |
| Peoria\_Fresh\_Resource\_Asset\_4.png | 522 | 521 |
| Logo.png | 505 | 150 |
| AboutUsBanner.png | 2400 | 800 |
| BlueRoofLabsLogoWhiteBlackBG.svg | 500 | 191.6 |
| Peoria\_Fresh\_Resource\_Asset\_1.png | 780 | 720 |
| Peoria\_Fresh\_Resource\_Asset\_.2.png | 780 | 720 |
| Peoria\_Fresh\_Resource\_Asset\_3.png | 780 | 720 |
| Peoria\_Fresh\_Resource\_Asset\_4.png | 780 | 720 |
| 1st Place.png | 500 | 500 |
| 2nd Place.png | 500 | 500 |
| 3rd Place.png | 500 | 500 |
| 4th Place.png | 500 | 500 |
| 5th Place.png | 500 | 500 |
| Apple Angel.png | 500 | 500 |
| Aspiring Asparagus.png | 500 | 500 |
| Blueberry Baron.png | 500 | 500 |
| Bok Champion.png | 500 | 500 |
| Broccoli Boss.png | 500 | 500 |
| Carrot Crusader.png | 500 | 500 |
| Cherry Bomb.png | 500 | 500 |
| Gracious Grape Giver.png | 500 | 500 |
| Leaderboard Legion.png | 500 | 500 |
| Melon Lord.png | 500 | 500 |
| Peach Princess.png | 500 | 500 |
| Pepper Pal.png | 500 | 500 |
| Potato Prince.png | 500 | 500 |
| Scurvy Free.png | 500 | 500 |
| Spinach Sultan.png | 500 | 500 |
| Sprite Sipper.png | 500 | 500 |
| Strawberry Sage.png | 500 | 500 |
| Top Banana.png | 500 | 500 |
| Locked 1st Place.png | 500 | 500 |
| Locked 2nd Place.png | 500 | 500 |
| Locked 3rd Place.png | 500 | 500 |
| Locked 4th Place.png | 500 | 500 |
| Locked 5th Place.png | 500 | 500 |
| Locked Apple Angel.png | 500 | 500 |
| Locked Aspiring Asparagus.png | 500 | 500 |
| Locked Blueberry Baron.png | 500 | 500 |
| Locked Bok Champion.png | 500 | 500 |
| Locked Broccoli Boss.png | 500 | 500 |
| Locked Carrot Crusader.png | 500 | 500 |
| Locked Cherry Bomb.png | 500 | 500 |
| Locked Gracious Grape Giver.png | 500 | 500 |
| Locked Leaderboard Legion.png | 500 | 500 |
| Locked Melon Lord.png | 500 | 500 |
| Locked Peach Princess.png | 500 | 500 |
| Locked Pepper Pal.png | 500 | 500 |
| Locked Potato Prince.png | 500 | 500 |
| Locked Scurvy Free.png | 500 | 500 |
| Locked Spinach Sultan.png | 500 | 500 |
| Locked Sprite Sipper.png | 500 | 500 |
| Locked Strawberry Sage.png | 500 | 500 |
| Locked Top Banana.png | 500 | 500 |
| add.svg | 30 | 30 |
| caretDown.svg | 36 | 36 |
| caretLeft.svg | 36 | 36 |
| caretRight.svg | 36 | 36 |
| caretUp.svg | 36 | 36 |
| checked.svg | 18 | 18 |
| delivery.svg | 20 | 20 |
| edit.svg | 12 | 12 |
| editBlack.svg | 12 | 12 |
| filter.svg | 24 | 24 |
| growing.svg | 16 | 14 |
| harvest.svg | 31 | 31 |
| minus.svg | 24 | 24 |
| minusButton.svg | 27 | 27 |
| plant.svg | 20 | 20 |
| plus.svg | 24 | 24 |
| plusButton.svg | 27 | 27 |
| question.svg | 16 | 16 |
| redX.svg | 28.9 | 28.9 |
| search.svg | 24 | 24 |
| seedPacket.svg | 18 | 17 |
| star.svg | 25 | 25 |
| trash.svg | 24 | 24 |
| unchecked.svg | 18 | 18 |
| x.svg | 24 | 24 |

# 

# **Branding Consistency**

* What is this: if you are making a version of this website for your own city, you likely do not want the website to be about Peoria. This section will refer to how you can replace any references to the city with your own.
  + Logos: discussed in the replacing images section, there are a few images that are named specifically after the city of Peoria, or are specific to the PeoriaFresh brand:
    - Banner.png
    - Logo.png
    - website-banner.png
    - favicon.ico (this one is located in the public folder)
    - Honorable mention: uiuc-extension.jpg and its links
  + Text references: References to “Peoria” and “Peoria Fresh” will need to be replaced as well. To find references to Peoria, the state of Illinois, and the like, you can use %<search> to find these references.The website contains information about the Peoria branch specifically that will need to be changed beyond the words Peoria and Peoria Fresh. This would include references to the state of Illinois or resources specific to Illinois, as well as the UIUC links.
  + Find-a-pantry: If you are out of the state of Illinois, the find a pantry will need an overhaul (ex.: gardener-find-a-pantry.tsx). 
  + About Us: At file path ‎src/views/gardener/aboutUs.tsx, you may want to change the about us to something about yourself
    - The text listed in typography is plaintext on the site. You can use this to describe yourself, or describe where your <City>Fresh originated!

# **How to Make it a Website**

There are a number of different ways to turn this project into a website. The Peoria Fresh Team did this using Google Cloud to host it. Because of the many ways in which people can adapt our project, we’re just going to link to a few different website tutorials that you can use.

* Google Cloud Tips (Recommended by us)
  + [How to Host a FREE Website on Google Cloud Platform](https://www.youtube.com/watch?v=f56PG7QxjFI)
  + [Connect to a GitHub repository | Cloud Build Documentation](https://cloud.google.com/build/docs/automating-builds/github/connect-repo-github)
  + [Connect to GitHub | Developer Connect | Google Cloud](https://cloud.google.com/developer-connect/docs/connect-github-repo)
  + [Connecting GitHub Repo with Cloud Source Repository](https://www.youtube.com/watch?v=PD83mmyAbs4)
* Amazon Web Services (AWS) (Recommended by us)
  + [Create a connection to GitHub - Developer Tools console](https://docs.aws.amazon.com/dtconsole/latest/userguide/connections-create-github.html#:~:text=to%20Step%202.-,To%20create%20a%20connection%20to%20GitHub,Authorize%20AWS%20Connector%20for%20GitHub).
  + [Connect a GitHub Repo with AWS | FREE Beginner AWS x DevOps Project](https://www.youtube.com/watch?v=4NPjgpOdeFY)
* GitHub Pages
  + [How to Host a Website On Github Pages](https://www.youtube.com/watch?v=OltY8JIaP-4)
  + <https://pages.github.com/>
  + [How to Deploy a Static Website to GitHub Pages - Free Hosting Tutorial](https://www.youtube.com/watch?v=AD-3nVI3-_U)
* Cloudflare Pages
  + [Deploy your website for free with Cloudflare Pages and GitHub!](https://m.youtube.com/watch?v=MTc2CTYoszY&t=0s)
  + [Git integration guide · Cloudflare Pages docs](https://developers.cloudflare.com/pages/get-started/git-integration/)

Note: There are other options aside from these that you can use!

# **General Tips/Information**

Docker Desktop

* Docker is a very *weird* system to use. The DevEnvironments we use can be somewhat unstable causing some unexpected problems to occur. The most recent development team (the team writing this entire document) had multiple situations where docker “broke.” One time Docker corrupted a member’s Windows Subsystem for Linux (WSL), which is required to help Docker update. They had to use [this tutorial](https://superuser.com/questions/1619233/completely-reinstall-wsl) to fix it.
* If Docker’s DevEnvironments are like this, then why are we using it? The DevEnvironments were originally an experimental feature that was being worked on and updated. Eventually, the developers of Docker gave up on creating the feature, leaving it in a somewhat buggy state. The original Peoria Fresh development team that decided to use the DevEnvironments chose to use them BEFORE the Docker developers dropped the feature. We didn’t want to completely start over using something else, so we continued to work with it.

Heidi

* When HeidiSQL is updated for one person, it is NOT updated for everyone. You will need to write a “migration” to update it for everyone. These migration files are stored in the backend in a folder called “migrations”
* You can connect your HeidiSQL app directly to a Google Cloud Server through the secrets tab in the Server.

# **Additional Resources**

This section features information about the other files in the Open-Source-Help Folder of the Frontend GitHub.

* Peoria Fresh - Open Sourcing Tutorial.docx: The file you’re currently reading!
* 24-25 New Art.zip: A zip folder containing all of the art we created for badges, flyers, seed packets, and some UI PNGs we made when we were designing the website before we coded it.

# **Thank You!**

Thank you so much for adapting our code! We hope that your project is able to help food banks get all the produce they need in your city!