**WSL and Windows Users Must Read Before Next Lecture**

This note will cover important fixes for volumes issues and file watchers not updating changes on Windows. Please do not continue with the next lecture until you have read this note in full. Solution #2 is a best practice, however, it will require more work, configuration and code changes. Solution #1 will allow you to move forward very quickly and requires only a small code change.

**Solution #1 (Quickest)**

Many students have indicated recently in our QA that they wish to know the quickest and least complicated solution to solve the code not updating on Windows issue regardless of whether it is a best practice.  **Please understand, this solution only resolves the reloading of the web application, it will not help with the re-running of the tests.**Solution #2 is the only thing that will help with both the reloading of the app and the re-running of tests after a code change.

This solution assumes that the project exists on the Windows file system and was not created on or copied to the WSL Linux file system.

There is currently an undocumented solution in the latest versions of Webpack:

<https://github.com/facebook/create-react-app/issues/12397>

1. You can add the variable to your frontend src/package.json start script like so:
   1. "scripts": {
   2. "start": "WATCHPACK\_POLLING=true react-scripts start",
2. Build the image as you normally would:

docker build -f Dockerfile.dev -t USERNAME:frontend .

1. From the root of your frontend project directory, run the container using **PowerShell:**

docker run -it -p 3000:3000 -v /app/node\_modules -v **${PWD}**:/app USERNAME:frontend

Do **not** use the pwd variable shown in the course videos. This will not work with Windows terminals. Please use **${PWD}** which is correct for use with PowerShell.

**Solution #2 (Best Practice)**

When using WSL to run Docker Desktop on Windows, the project should have been created on the Linux file system and **all docker commands should be run within WSL**as per best practices:

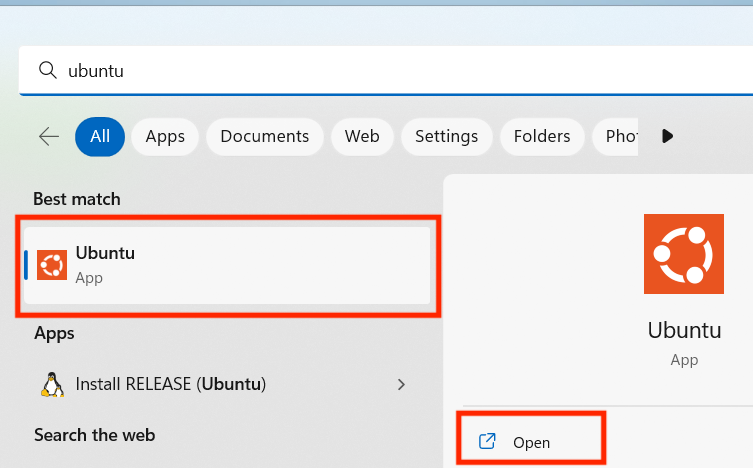
<https://docs.docker.com/desktop/windows/wsl/#best-practices>

If the project was created on the Windows file system, the volumes may not work correctly and performance may greatly suffer. To address this, you will need to copy your project to the Linux File system using the following instructions:

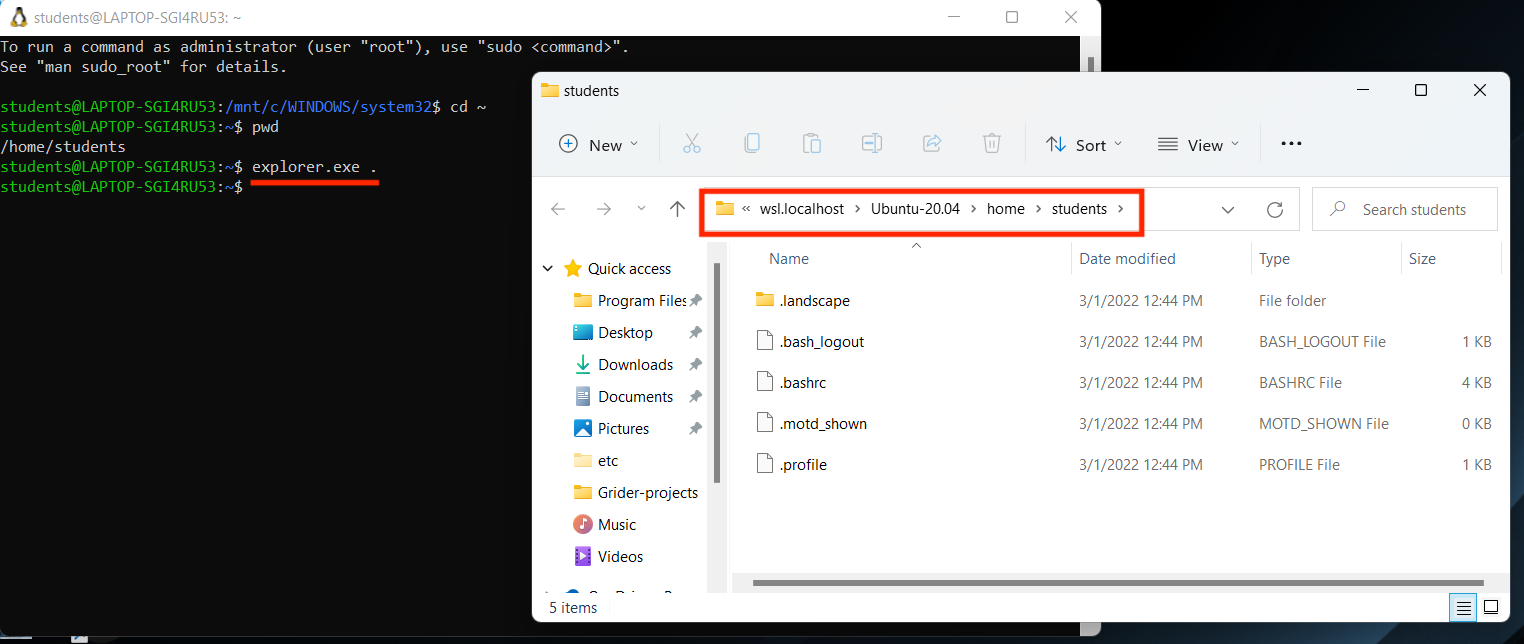
1. To open your WSL operating system use the search / magnifying glass in the bottom system tray:



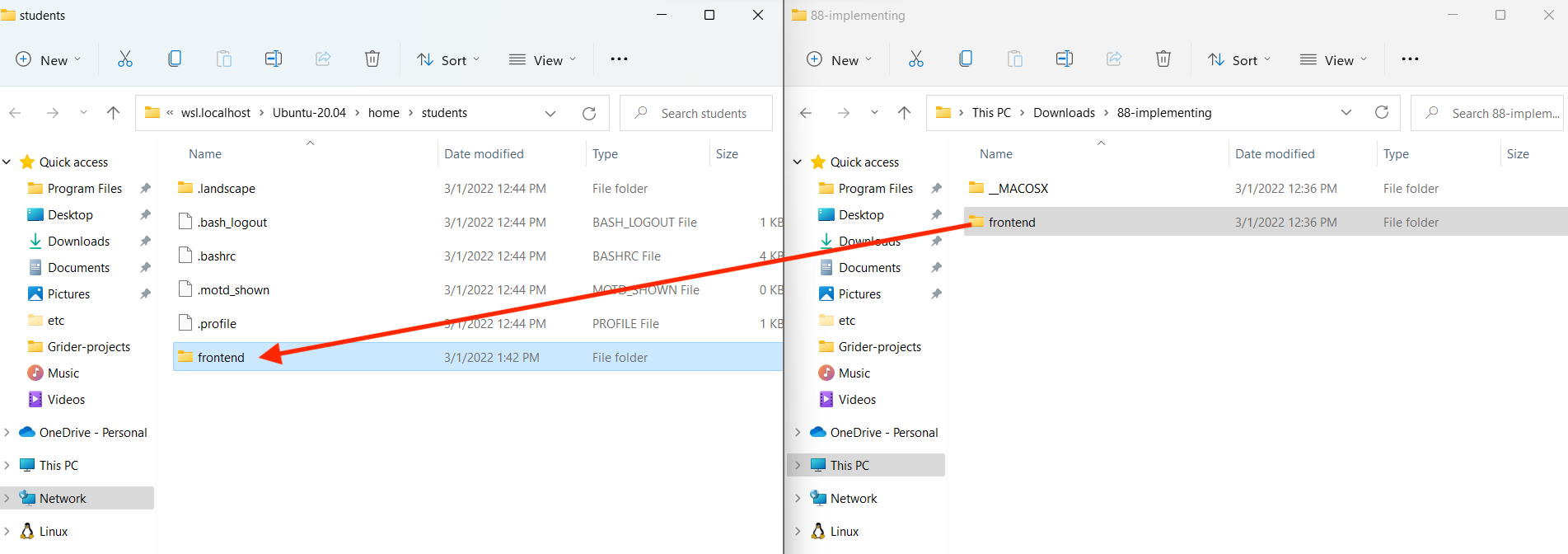
1. Type the name of your distribution (by default it is Ubuntu) and click **Open:**



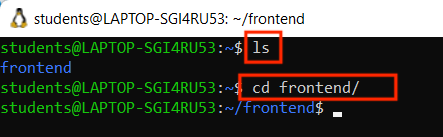
1. When the terminal launches it should automatically open to the home directory on the Linux filesystem.
2. Run explorer.exe . to open up the file explorer at /home/USERNAME directory within WSL:



1. Move the frontend project directory into the WSL file browser window:



1. Your project path should now look like **/home/USERNAME/frontend.**Run ls to confirm that you are in the correct location. Then, run cd frontend to change into the project directory.



1. Delete any **node\_modules** or **package-lock.json** files that may exist in the project directory. If these were generated on the Windows file system and were copied over, they will conflict.
2. Update your **Dockerfile.dev** to look like this:
   1. FROM node:16-alpine
   3. USER node
   5. RUN mkdir -p /home/node/app
   6. WORKDIR /home/node/app
   7. pwd
   8. COPY --chown=node:node ./package.json ./
   9. RUN npm install
   10. COPY --chown=node:node ./ ./
   12. CMD ["npm", "start"]

**Explanation of changes:**

USER command- this will set the user name to use as the default user name for the current stage

There might be some difference between who is using it

We are specifying that the **USER** which will execute **RUN**, **CMD**, or **ENTRYPOINT** instructions will be the **node** user, as opposed to **root** (default).

<https://docs.docker.com/engine/reference/builder/#user>

We are then creating a directory of **/home/node/app** prior to the **WORKDIR** instruction. This will prevent a permissions issue since **WORKDIR** by default will create a directory if it does not exist and set ownership to **root**.

WORKDIR will change the current working location to the specified location

Or if uwant to move inside to that directory execute that command

WORKDIR /a

In this case, ,sofar we are outside of a ,now in the current directory a new folder is created (if it is not there) called “a”, and we will be automoved to

That inside folder “a”

Whereas mkdir will definitely create a folder

The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile. If the WORKDIR doesn’t exist, it will be created even if it’s not used in any subsequent Dockerfile instruction.

The WORKDIR instruction can be used multiple times in a Dockerfile. If a relative path is provided, it will be relative to the path of the previous WORKDIR instruction. For example:

WORKDIR /a

WORKDIR b

WORKDIR c

RUN pwd

The output of the final pwd command in this Dockerfile would be /a/b/c.

The inline **chown** commands will set ownership of the files you are copying from your local environment to the **node** user in the container.

The end result is that some files and directories will no longer be owned by root, and no npm processes will be run by the root user. Instead, they will all be owned and run by the **node** user.

1. Using the WSL terminal build your Docker image as you typically would:  
   docker build -f Dockerfile.dev -t USERNAME:frontend .
2. Using the WSL terminal, start and run a container. It is very important that you do not use a PWD variable as shown in the lecture video. Use the ~ alias for the home directory or type out the full path:  
   Using **~** alias:  
   docker run -it -p 3000:3000 -v **/home/node/app/**node\_modules -v ~/frontend:**/home/node/app** USERNAME:frontend

Using the full path:  
docker run -it -p 3000:3000 -v **/home/node/app/**node\_modules -v /home/YOURUSERNAME/frontend:**/home/node/app** USERNAME:frontend

1. Going forward in this course, **all Docker commands and projects should be run within WSL** and not Windows.

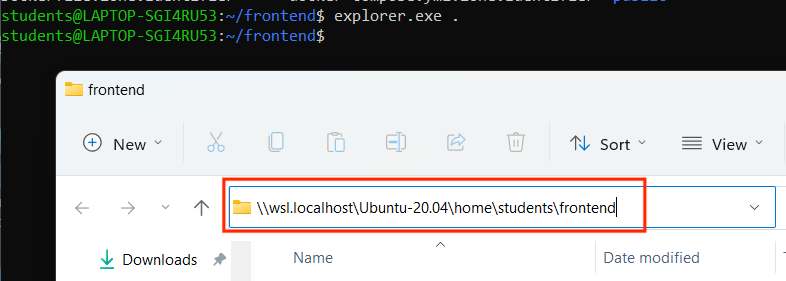
**Adding WSL project folder to VSCode**

**Option #1**

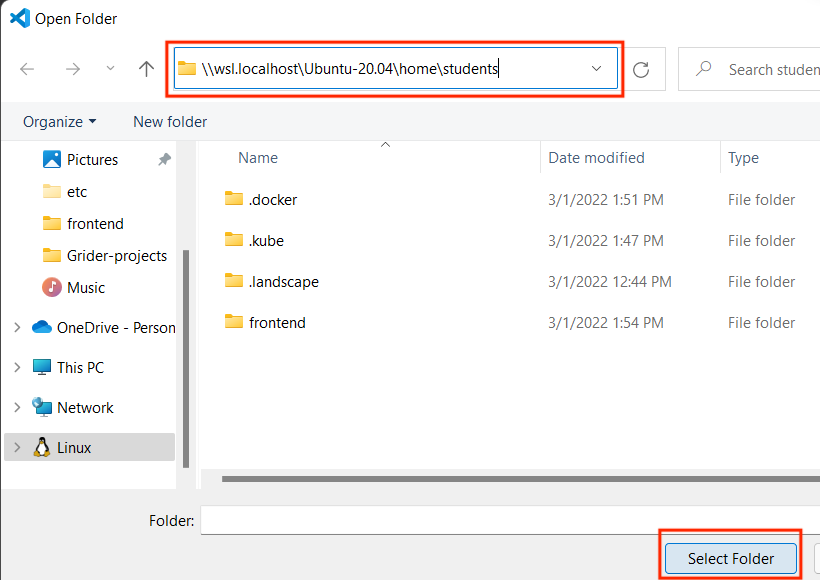
<https://code.visualstudio.com/docs/remote/wsl#_open-a-remote-folder-or-workspace>

**Option #2**

With the terminal of your distro, make sure you are inside your frontend project directory and run the explorer.exe . command. This will open a file browser window within the WSL location. Copy the location path:



Go to VSCode and select **File**, then **Open Folder**. Paste the wsl address you just copied into the file browser window and click the **Select Folder** button:



This will add the project that is located in the WSL file system into your VSCode workspace.

**Docker Desktop with HyperV**

Using HyperV as a backend is no longer common or recommended given the widespread adoption of WSL. To avoid confusion these instructions have been removed.