# Steps to set up Dev-Environment and docker image:

*(This guide uses parts of the setup guide from CS1660 and CS0300. Feel free to check out their guide for additional debugging tips during the dev-environment setup process as well. Big thanks to Professor DeMarinis for his help.)*

**Why dev-environment?**

Last year, every student was allowed to do local development on their native systems, but this ended up causing setup problems for each assignment for the entire year. As such, we decided to try using a dev-environment this year with linux. This way, bugs with setup will hopefully be avoidable or at least standardized between different operating systems. An additional benefit of this is that once you set up your dev-environment, Go, Git, and SQLite should automatically be installed in your dev-environment.

---

## Download Docker

Download and install Docker Desktop, located here. On Linux machines, follow the instructions here.

*Already have docker installed? If you already have Docker installed, we recommend updating to the latest version by reinstalling it from Docker's website. Old Docker can sometimes cause issues.*

Note: if docker asks for privileged access, say yes.

Mac users: We do NOT recommend installing Docker with homebrew. This may install a less-than-latest Docker version, which may cause problems.

---

## Windows Only: Install WSL

To run the following steps in this setup, you will need to set up Windows Subsystem for Linux (WSL). WSL should already be enabled after you install Docker, but you may still need to install a Linux distribution. This will run in an actual Linux VM, and you will run your Docker container within that VM (turtles all the way down for you!).

1. **Do I have a Linux distribution (Linux distro) installed?**
   - ➢ Run *wsl -l -v* in the Command Prompt or Powershell. If there is only “Docker Desktop” and “Docker Desktop Data”, you do not have a Linux distribution installed. Proceed to step 2.
   - ➢ Otherwise, you have a Linux distro installed. Proceed to step 3.

2. **Install a Linux Distribution.**
   - ➢ Run *wsl --set-default-version 2* to ensure Ubuntu will be installed under WSL 2.
   - ➢ Install “Ubuntu 22.04” from Microsoft Store.
   - ➢ Click “Open” after Ubuntu is downloaded. A terminal will open and guide you through the installation process.

3. **Ensure your Linux Distribution runs on WSL 2.**
   - ➢ From the output of *wsl -l -v*, find out if your Linux distro is using WSL 1 or WSL 2. If it’s WSL1:
     - ○ Run *wsl --set-version <distro name> 2* to update your distro to use WSL 2.
4. **Set your default Linux distro**
   - ➢ Run *wsl --setdefault <distro-name>* to configure your default Linux distro. *<distro-name>* should be “Ubuntu-22.04” if you installed using step 2.

Enter *wsl* in your Command Prompt or Powershell, and you’ll enter into your WSL! For the rest of the setup, run commands within your WSL Linux environment, unless otherwise specified.

You will also need to connect Docker with WSL. To do so, open your Docker Desktop’s settings (on its top right corner), click “Resources”, “WSL integration”, then enable integration with your Linux distro. Then, click “Apply and Restart”.

---

## Verify Docker Installation:

Verify Docker is installed by executing the following:

```
$ docker --version
```

Once you execute the above command, you should see the Docker version number, for example:

```
$ docker --version
Docker version 24.0.7, build afdd53b
```

After installing Docker, a Docker process (the Docker daemon) will run in the background. Run the following command to verify:

```
$ docker info
```

If you see the following error:

```
ERROR: Cannot connect to the Docker daemon at
unix:///var/run/docker.sock. Is the docker daemon running?
```

It means Docker hasn’t started running yet. On Windows or macOS, ensure your Docker Desktop is running. On Linux, try the command *sudo systemctl* docker restart in a terminal.

**Super important instructions for mac users:**

We have noticed that some MacOS users may experience issues with files not updating when you save them under some combinations of Docker settings: follow these instructions to check your Docker installation and make sure it's set correctly.

Do not skip this step or you may encounter problems later!

---

## At this point, you should have docker downloaded and ready for use!

### To set up your environment, do the following:

1. Open a terminal and enter the directory on your computer where you want to do your coursework.
    a. Clone the dev-environment from this GitHub repository: https://github.com/CSCI-1270-Staff/dev-environment

2. Use terminal to navigate into the dev-environment repo you just cloned and run the command:
    ```
    ./run-container setup
    ```
    ***Note:*** If you are on Windows, you will need to run this in WSL

3. Then, start the container by running in the **dev-environment** folder:
    ```
    ./run-container
    ```
    ➢ In the future, just repeat steps 4 and 5 in order to resume your development..

    ***Note:*** If you are on Windows, you will need to run this in WSL

4. Attach VSCode to this container and you should be able to develop freely!
    ➢ Detailed VSCode Steps from the CS300 setup guide:
        i. Download and Install VS Code on your computer (not the course container) normally

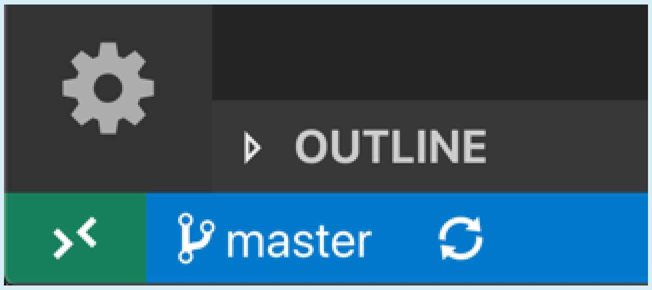
ii. Navigate to the extensions tab by clicking this icon on the left side of the screen:

iii. Search for and install the “Docker” and “Dev Containers” VS Code extensions via the extensions tab. Also install “WSL” if you are on Windows.

   1. **(Recommended) Also install the Go extension by the Go Team on Google. This will give you intellisense for your local development.**

iv. Make sure your course container is running (either by connecting to it, or checking the docker desktop app).

v. Click the green button (can be a different color depending on the theme)  in the bottom left of VS Code, then click “Attach to running container” and select your cs1270 course container:



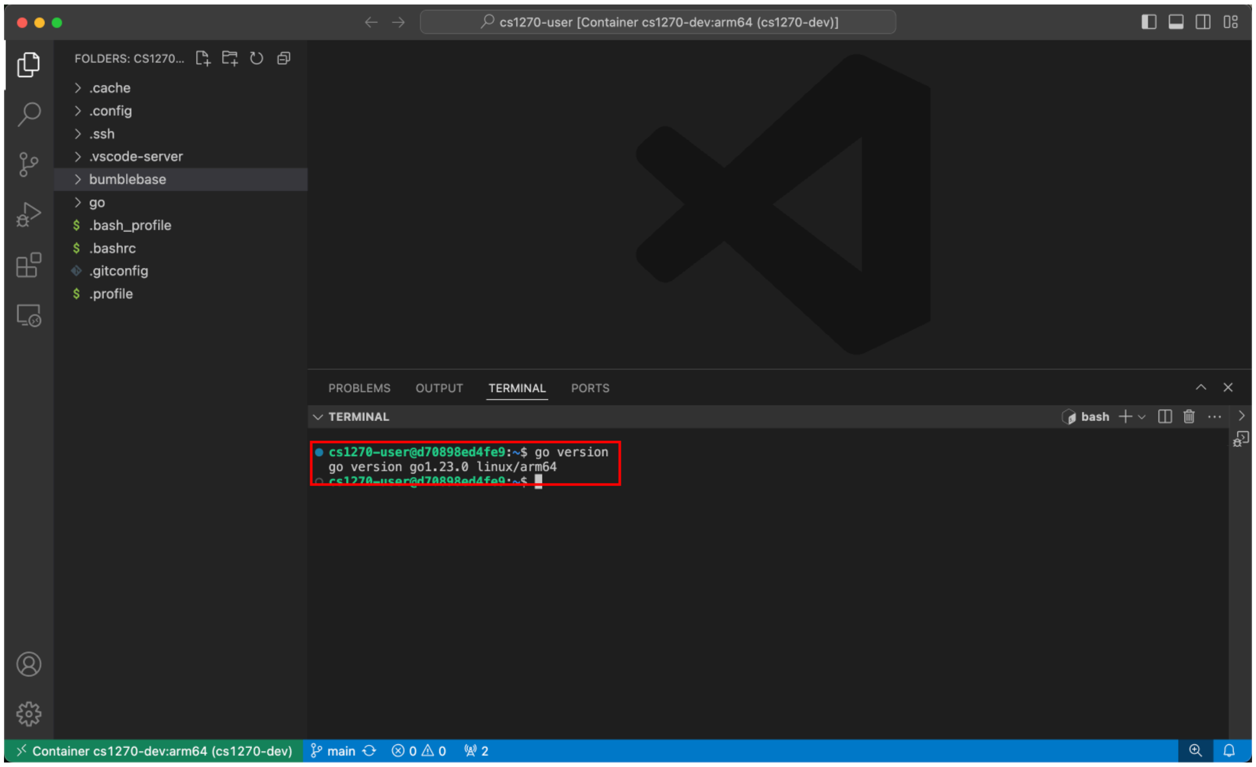
vi. You can now open any folder you want to edit, and can get a terminal from your course container by clicking View > Terminal.

**Congratulations! You should now be able to run a shell, as well as write and run your programs in the dev-environment!**

---

### Verify successful dev-environment setup:

To verify that your setup is working, check that your Go version is 1.23.0. You can check your Go version by running the command: ‘*go version*’.



```
cs1270-user@d70898ed4fe9:~$ go version
go version go1.23.0 linux/arm64
cs1270-user@d70898ed4fe9:~$
```

You can also try creating some sample files in your dev-environment **home** directory and checking that the files still persist after you exit your dev-environment.
*(Note: the home directory is the only directory that is mounted on a volume, and thus, it is the only directory that persists outside of the dev-environment.)*

To familiarize yourself with using the dev-environment, try closing and re-opening the dev-environment.

➢ When opening the dev-environment, you can run the *./run-container* command as shown in step 3 of the setup. Make sure that you first open the Docker Desktop application!
  ○ *(Note: windows users – make sure to run the command command in wsl.)*
➢ To close the dev-environment, you can keep running *exit* in the terminal you used to enter the dev-environment.

## Next Steps:

Move to Setup Github Guide - 2024 .
*(Note: This will be available after the first assignment is released!)*