Onboarding

Created	@January 29, 2023 9:20 AM
Tags	general
Files & media	
Level	Beginner
Related Resources	
Status	Done

Software Team Onboarding

Welcome to the MSU RoboCats software team! This document will serve as a guide through the onboarding process. Please do not hesitate to reach out to other software team members if you run into issues, or have any questions during the setup process.

This document will cover the following content:

> ⚠️ Note, the setup process has been segmented into separate modules and submodules. Feel free to select which modules you follow according to your current knowledge and development environment

Getting Started with Git and GitHub

GitHub is a frontend service built around the Git Version Control System (VCS). Git and GitHub are the crux of our agile development process, offering a suite of tools that allow multiple developers to work on code without ‘stepping on each others toes’ so to speak. Additionally, the software team uses GitHub for project management and planning. If you have never used Git or GitHub, it is recommended that you explore the following resources:

Git: https://git-scm.com/book/en/v2

GitHub: https://git-scm.com/book/en/v2/GitHub-Account-Setup-and-Configuration

Once again, if you have additional questions regarding Git, GitHub, or VCS’, do not hesitate to reach out your fellow team members! Version control is an extremely useful practice that is used everywhere in industry; it is a great philosophy to understand.

Creating a GitHub Account

To contribute to our code repositories, you will need to have a GitHub account. This can be easily accomplished by navigating to the GitHub signup page and following the instructions to create an account.

⚠️ Your new GitHub account will likely follow you throughout your entire development career. As such, it is recommended that you choose a professional account name and use your primary email address when creating an account (though, note that this information *can* be updated after account creation). Additionally, consider enabling two factor authentication for best security practice

Installing Git

Git is the underlying version control software that GitHub is based on. It is a separate program that is installed locally on your device, and used to interface with GitHub repositories (most commonly, pulling code to your device, committing changes, and pushing those changes to the remote repository).

To install Git, navigate to the Git download page and select your target platform. Select the latest build to to begin downloading the installer. Once the download completes, run the installer and follow the prompts to complete the download. Simply accept all of the default install options (unless you have motive for a custom Git install).

Configuring Git

Once installed, your can configure Git to use your GitHub credentials. This will allow you to access controlled/private repositories, as well as ensure that commits and changes are correctly associated with your account.

All configuration in git is performed using the `git config` command. To set your account username globally (so that it is associated with all Git actions performed across your device), you can use the following command:

⚠️ Note: Replace the text within brackets (i.e., [USERNAME]) with your information

```
git config --global user.name "[USERNAME]"
```

Similarly, to associate an email address with your Git actions you can use the following command:

```
gti config --global user.email "[EMAIL]"
```

If you would like a more comprehensive guide for configuring Git (such as how to anonymize your email address in commits), you can reference the following materials:

Configuring Git for GitHub: https://docs.github.com/en/get-started/quickstart/set-up-git#setting-up-git

Git Configuration: https://git-scm.com/book/en/v2/Customizing-Git-Git-Configuration

Joining the MSU RoboCats Organization

To access and contribute to the RoboCats repositories, your GitHub account needs to be added as a contributor to the MSU RoboCats GitHub organization. To do so, please provide your GitHub username and GitHub email address to club leadership. This can be done in person, or you can send a direct message via Discord.

A club leader will inform you once you have been added as a contributor, after which you can begin setting up your local development environment. In the meantime, continue reading to complete the Getting Started with Docker and Getting Started with Visual Studio Code sections.

Getting Started with Docker

Docker is popular containerization platform that allows for the virtualization of applications. If you are unfamiliar with containerization, it is recommended that you checkout the following resources. The y will provide you with a high level overview of what Docker is and give you a taste of the underlying technology that makes it work. This will prove very useful in industry, where large applications are moving towards distributed, containerized micro service architectures.

What are Containers: https://www.docker.com/resources/what-container/

Docker Presentation: https://docs.google.com/presentation/d/15UmvY0xIgY8aiVnutcznby3Ko7ZnbIog/edit?

usp=share_link&ouid=103475242643168753844&rtpof=true&sd=true

Installing Windows Subsystem for Linux

Docker works by relying on the standardization of the Linux kernel to expose the same underlying application environment independent of the host hardware. If you are using an operating system that does not have access to a Linux kernel by default (such as Microsoft Windows), then you will need to install some additional software. For windows, you can install Windows Subsystem for Linux 2 (WSL2) to add this support. For other operating systems, you will need to explore other options.

To install WSL2, open a windows PowerShell or Command Prompt **as an administrator**.

> ⚠️ It is advised that you install the Windows Terminal application for a better terminal experience. This will prove useful throughout your software development career
>
> Installing Windows Terminal

With the PowerShell or Command Prompt open, enter the following command to begin installation:

```
wsl --install
```

By default, this will install the latest Ubuntu distribution on your Windows machine. After it finishes installing and decompressing, it will open the WSL shell and prompt you for account credentials. These credentials will belong to the **separate** user account on Linux distribution that you installed, and **do not** need to be the same as your Windows account credentials. That said, there is no harm in using the same credentials.

For more information on installing Windows Subsystem for Linux, please reference the following documentation:

WSL Installation: https://learn.microsoft.com/en-us/windows/wsl/install

Installing Docker Desktop

With WSL installed, you can now install Docker Desktop. Simply navigate to the Docker Desktop download page and select the appropriate target distribution. Once the installer is downloaded, follow the prompts to install Docker Desktop. Simply accept all of the default install options (unless you have motive for a custom Docker install).

Getting Started with Visual Studio Code

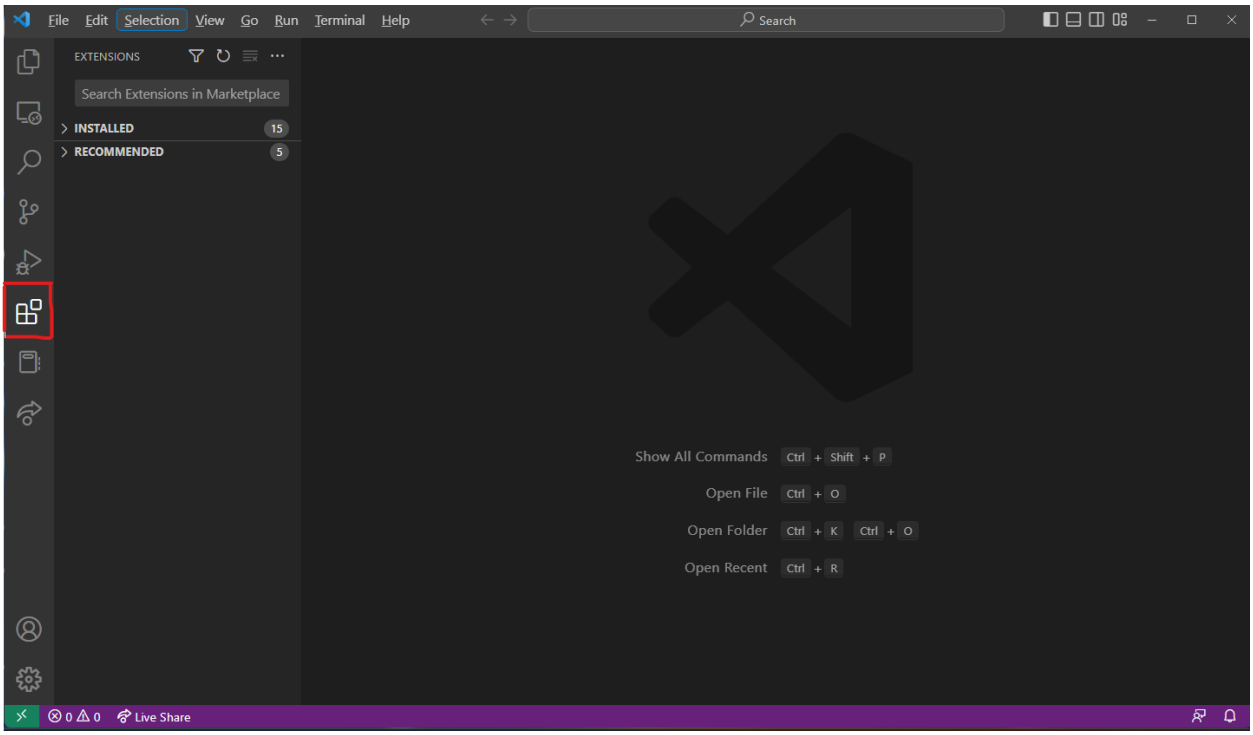
Visual Studio Code is the standard editor used by out Software Team. It offers a lightweight yet expansive development environment that integrates well with Docker. The following sub sections will cover the installation and configuration process for VS Code, however you are free to use another IDE of your choosing.

Installing VS Code

To install VS Code, simple navigate to the VS Code download page and select your target distribution. Run the installer once it finished downloading, and follow the prompts to install VS Code. Simply accept all of the default install options (unless you have motive for a custom VCS Code install).

Configuring the IDE

Once installed, there a few extensions that are recommended for a more seamless development experience with our codebase. Firstly, open the extensions tab in the VS Code editor:



Once open, search for and install the following packages:

Name: Docker
Id: ms-azuretools.vscode-docker
Description: Makes it easy to create, manage, and debug containerized applications.
Version: 1.23.3
Publisher: Microsoft
VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker

---

Name: Dev Containers
Id: ms-vscode-remote.remote-containers
Description: Open any folder or repository inside a Docker container and take advantage of Visual Studio Code's full feature set.
Version: 0.266.1
Publisher: Microsoft
VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers

---

Name: Python
Id: ms-python.python

Description: IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, code formatting, refactoring, unit tests, and more.
Version: 2022.20.2
Publisher: Microsoft
VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=ms-python.python

---

Name: Remote - SSH
Id: ms-vscode-remote.remote-ssh
Description: Open any folder on a remote machine using SSH and take advantage of VS Code's full feature set.
Version: 0.94.0
Publisher: Microsoft
VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh

---

Name: Remote Explorer
Id: ms-vscode.remote-explorer
Description: View remote machines for Remote - SSH and Remote Server.
Version: 0.0.3
Publisher: Microsoft
VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=ms-vscode.remote-explorer

---

Name: Remote Development
Id: ms-vscode-remote.vscode-remote-extensionpack
Description: An extension pack that lets you open any folder in a container, on a remote machine, or in WSL and take advantage of VS Code's full feature set.
Version: 0.23.0
Publisher: Microsoft
VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.vscode-remote-extensionpack

Once these packages have been installed, restart your development environment to ensure the changes take effect.

Setting up the Development Environment

With WSL, Docker, and Git installed and configured, you can now setup the standard development environment used by the software team. Since most of our onboard software is developed in Python, we will begin by installing Python.

Installing Python

To install python, simply navigate to the Python download page. It is recommended to install python version 3.10.6, which the version used for active development. Download the installer for your target platform and follow the prompts to install Python. When prompted to set Python in the PATH, select yes. Additionally, select `pip` in the list of additional features to install. Otherwise, simply use the default configuration options.

Cloning the Development Repository

To clone our development repository (which is the entrypoint to developing all of our onboard software), navigate to a new directory and run the following command:

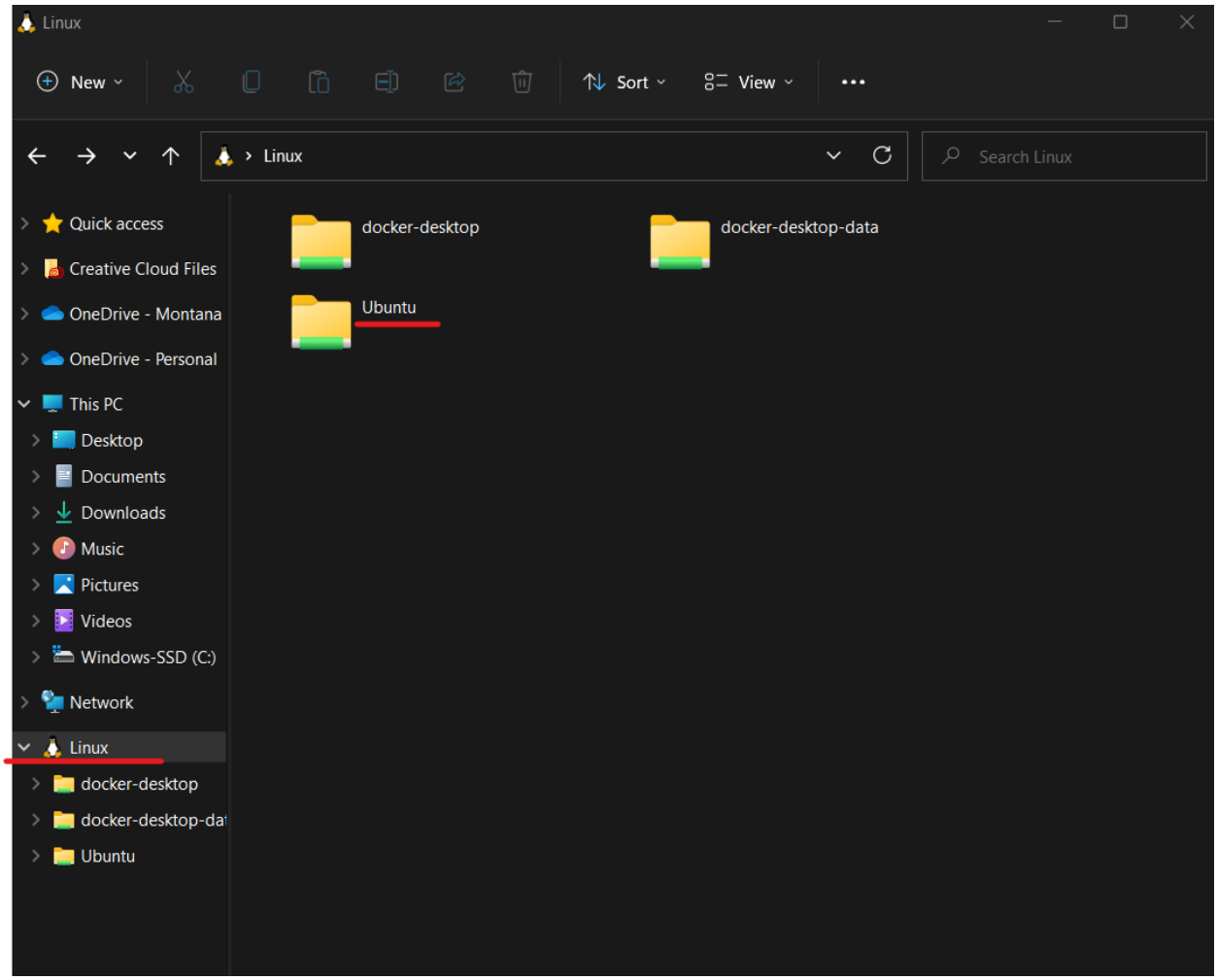
```
git clone https://github.com/MSURoboCats/developmentEnvironment.git
```

This will clone the `developmentEnvironment` repository into a new folder.

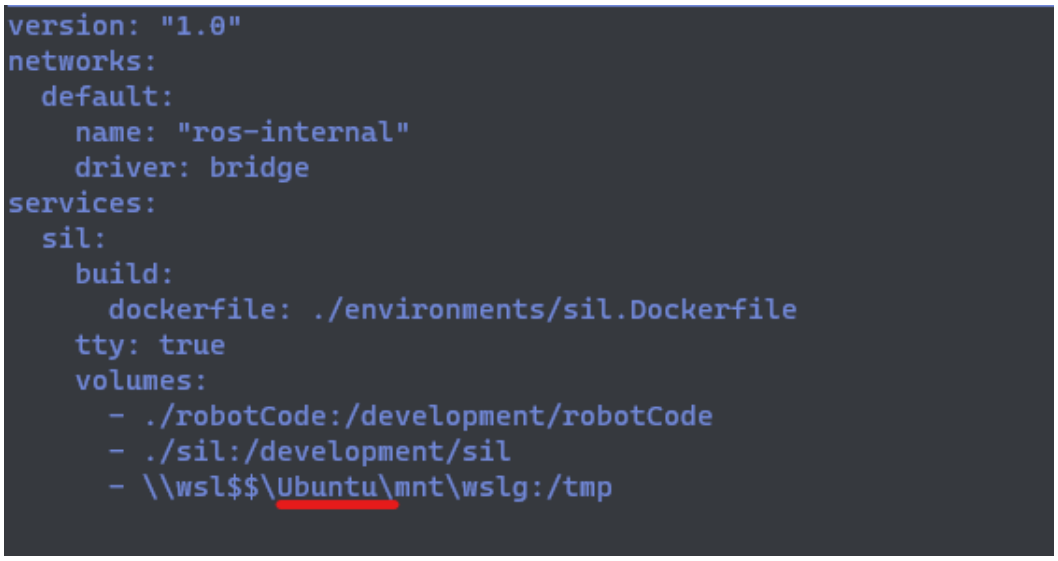
Building the Development Environment

> 💡 Make sure that docker is running when building the Development Environment

1. You will need to go to your Find the `wslg` directory in your Linux directories. For this example it is the `Ubuntu/mnt/wslg`. However, the top directory could be `Ubuntu-20.04` or even `Ubuntu-22.04` . You should be able to see the Linux directories on the bottom left of your file explorer. Note the name of the top Ubuntu directory. If the top level directory name is `Ubuntu-20.04` then you can skip step 2.



2. Now that you have found the correct directory you will need to go into the `docker-compose.yaml` file to edit the `volumes`. you will need to replace the `Ubuntu-20.04` with the name of the directory that you found. For this example the directory name that I am using is `Ubuntu`.



```
version: "1.0"
networks:
  default:
    name: "ros-internal"
    driver: bridge
services:
  sil:
    build:
      dockerfile: ./environments/sil.Dockerfile
    tty: true
    volumes:
      - ./robotCode:/development/robotCode
      - ./sil:/development/sil
      - \\wsl$$\Ubuntu\mnt\wslg:/tmp
```

With the `developmentEnvironment` cloned onto your local machine, you can then build the Docker containers and git submodules. To do this, navigate into the `developmentEnvironment` directory and run the following command:

```
python make.py build
```

This may take a few minutes as Docker resolves the container information. After building the environment, you should notice that the `sil` and `robotCode` directories are populated. These directories are individual Git repositories which you can make and commit changes in just like any other Git repository.

Developing in the Environment

A simple development workflow in this new environment is as follows:

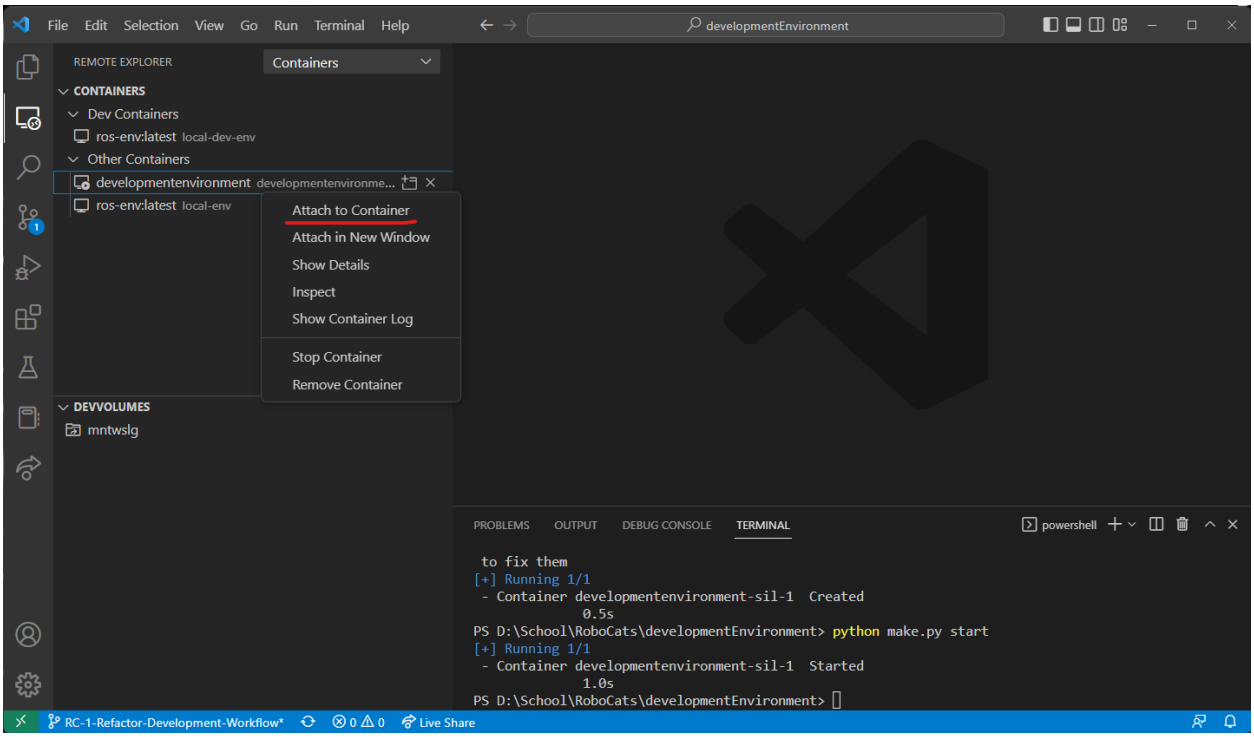
1. Start the docker containers by running the following command in the root `developmentEnvironment` directory:

   ```
   python make.py start
   ```

   This will start the development containers.

> ⚠️ Note, these containers can be stopped by running `python make.py stop` in the same directory

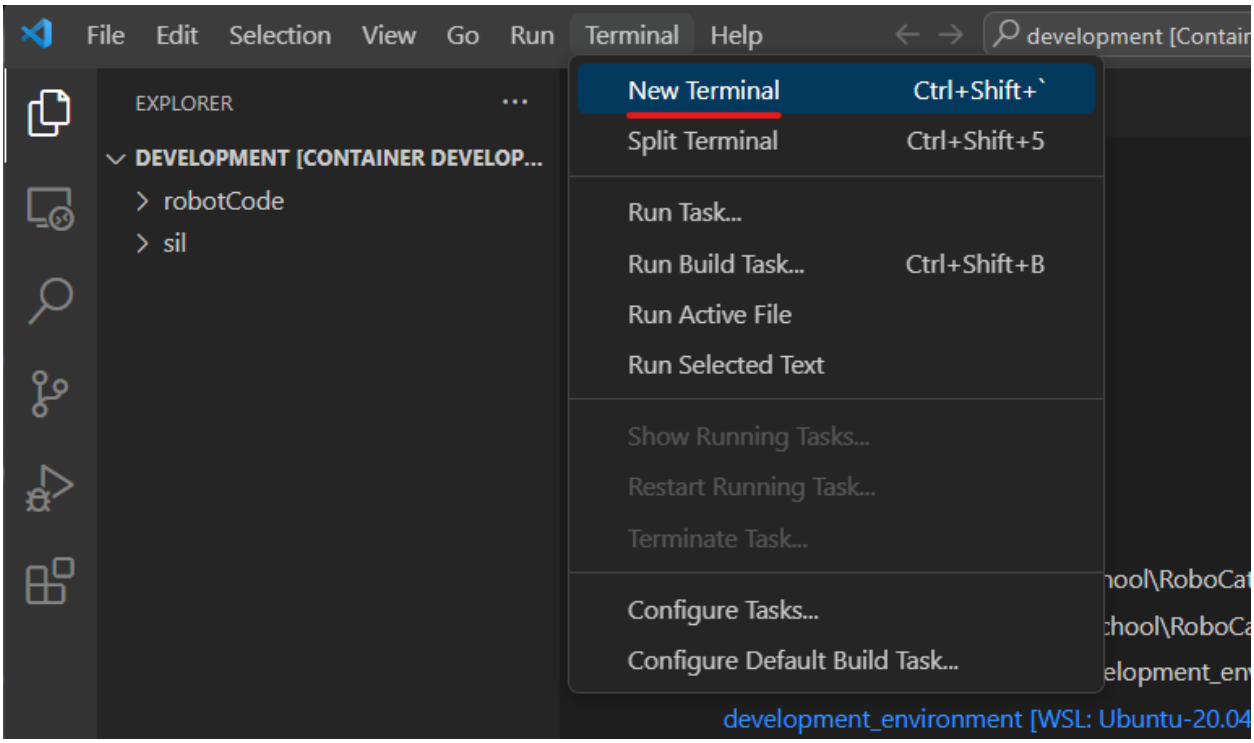
2. Attach to the containers in visual studio code by navigating to the *Remote Explorer* tab and selecting *containers* as your target. You should see your container running with the name “developmentenvironment-sil-1”.



Attaching to development container in VS Code

Right click on that container and select *attach to container*. You can then open the development repositories (namely `sil` and `robotCode` by opening the `/development` directory in the file explorer tab.

3. With the container attached in VS Code, you can also start the Software in the Loop (SIL) simulation environment. To do this, open a terminal window (Terminal > New Terminal) in VS Code.



This terminal should automatically be attached to the container. Type the following command to start the SIL environment:

```
ign gazebo
```

This will launch the Gazebo simulation software in a new window. You can then access the