Installing FEniCSx via Docker for use in VSCode on a Mac

Eric M. Stewart and Jorge Nin

June 25, 2024

In this document, we describe the process of creating a new Docker container from scratch using the v0.8.0 release of FEniCSx / dolfinx and set up Visual Studio Code (VSCode) to have full functionality running FEniCSx codes in Jupyter notebook files with inline plotting and animations.[1]

These instructions are written for installation on a Mac computer, and work equally well for machines using an Intel / AMD processor or an M-series / Apple Silicon / ARM processor. There is one small extra step exclusively for ARM machines, which is pointed out in red in Step 4 of Section 2.1.

The basic installation process is essentially identical for Mac and Windows machines, but the Screenshots in this version will be more directly helpful for Mac users.

1 Docker

In this Section, we provide some background information on Docker and walk through installing Docker Desktop. If you are familiar with Docker and have Docker Desktop installed, skip to Section 2.

1.1 What is Docker

Docker is a platform that uses containerization technology to simplify the deployment and management of software applications. In the context of software development, especially for complex engineering applications like FeniCSx, Docker offers several advantages:

- Containerization:
  - Containers are isolated environments where applications can run. They are lightweight because they share the host system's kernel (but not the OS itself) and are not burdened with the overhead of a virtual machine.
  - For FEniCSx, this means that each instance of the application runs in a consistent environment, irrespective of the underlying hardware or software configurations of the host system.
- Consistency and Reproducibility:
  - Docker ensures that FEniCSX runs identically on every machine, avoiding the common"it works on my machine" problem in software development.
- Ease of Distribution and Version Control:
  - Docker images (blueprints for containers) can be version-controlled and shared through Docker Hub or other registries, ensuring that all students and researchers access the same version of FEniCSx and its dependencies.

Because of these advantages, we will be using Docker to run FEniCSx.

[1] Some screenshots may have references to a "nightly" release of FEniCSx instead of v0.8.0. You can safely ignore this difference.

1.2 Installing Docker

It is recommended that you install Docker desktop for first time use. You can find the download here:

- `https://www.Docker.com/products/Docker-desktop`

and download the appropriate version for your operating system. An account is not necessary for basic install and usage.

After downloading open up the image and drag it into the applications folder. Docker should now be installed.

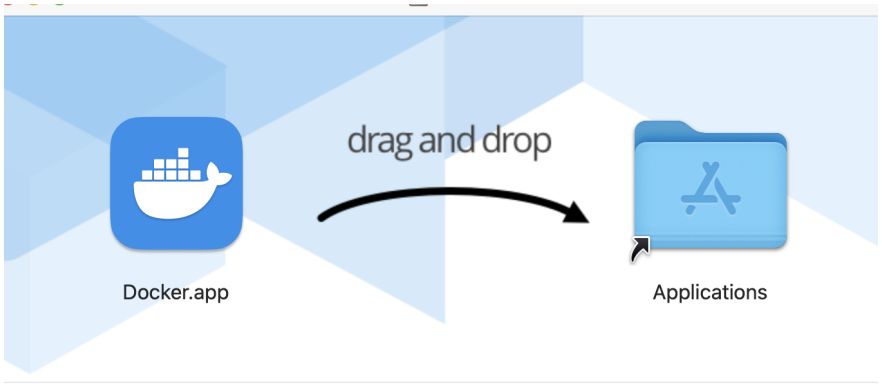
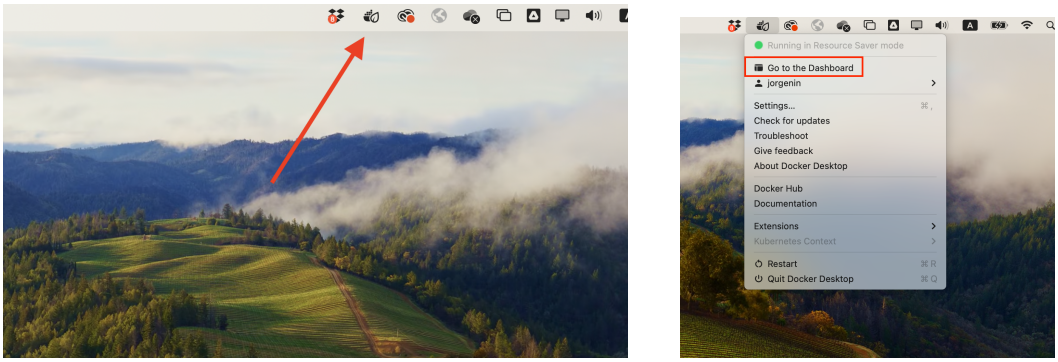


Figure 1: Docker install process

After Docker has finished installing open up the Docker desktop application. The application may not instantly open. Instead it may be in the background, look at figure 2 to see if your Docker is running in the background.



(a) Docker Open in the background

(b) Clicking on the Docker Symbol

Figure 2: How to check if Docker is open in the background. Click on "Go to dashboard" to open up Docker.

1.3 Docker Layout

Once Docker is open you should see a window similar to this. This is the dashboard.

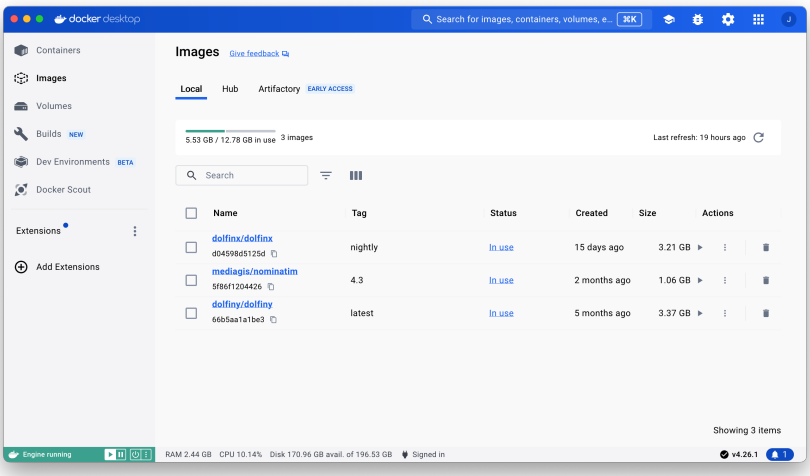


Figure 3: The Docker Dashboard

There are multiple tabs but we will only discuss two of them: the Containers tab, and the Images tab.

1.3.1 Images

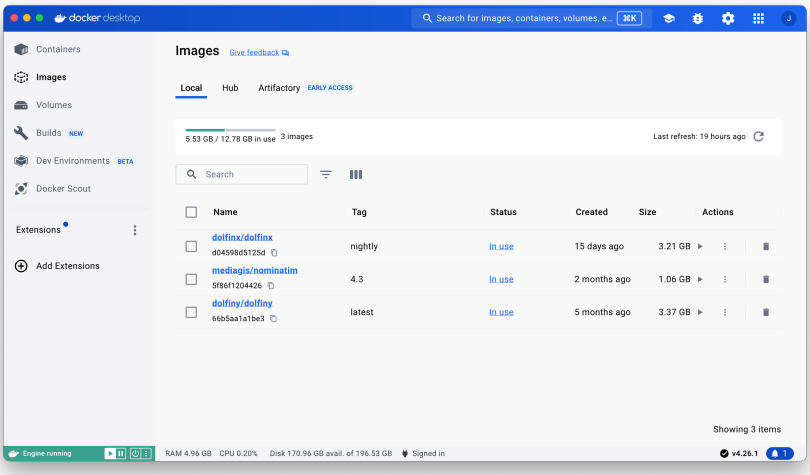


Figure 4: The Docker Images Tab

The Docker images tab contains all of the Images you currently have installed. Image are essentially different operating systems you can have. If you want to run multiple versions of FEniCSx for example you could have multiple images of it.

1.3.2 Containers

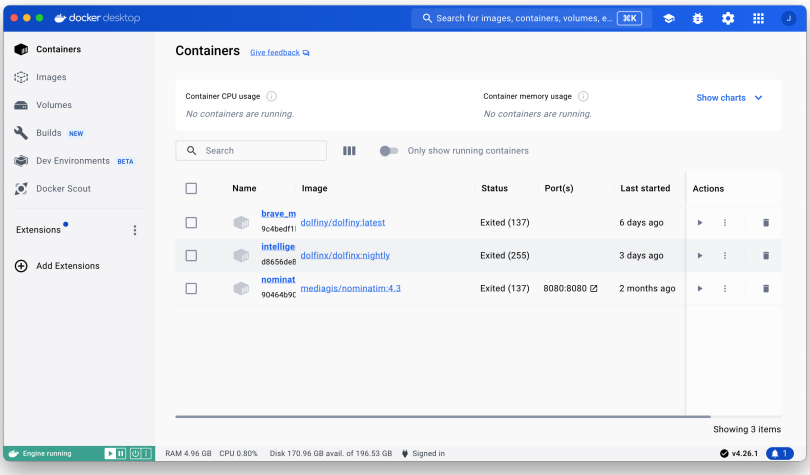


Figure 5: The Docker Containers Tab

The Docker containers tab has all of the containers that you have setup. Each container is essentially a different instance of a specific image. It is normally recommended that for every project you have a different seperate container made. Though those containers could use all the same image.

1.4 Recommended Settings

Because we will be running FEA software which benefits from being able to most of our computers resources, there are a few changes that should be made to optimize performance.

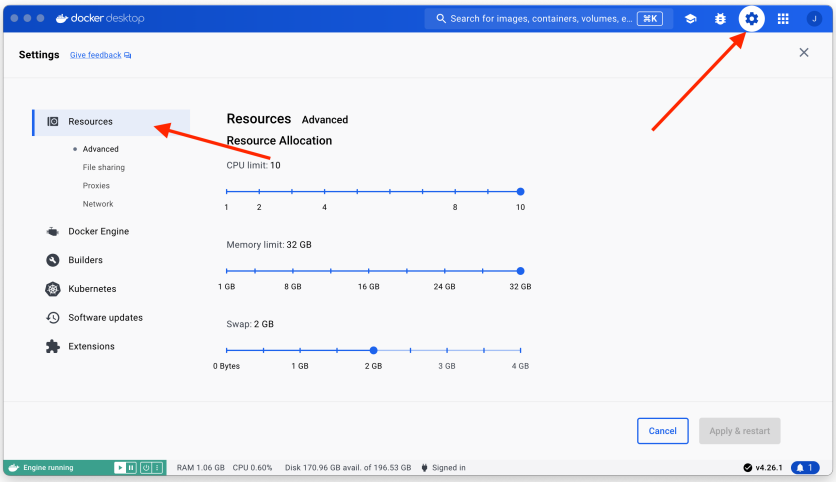


Figure 6: The Docker Settings tab

We will be going to the Docker settings tab, and then resources as seen in Figure 6. In here we will be maximizing the number of CPU's available and the memory limit. Allowing the container to use more memory can be highly impactful to improving performance of FEniCSx codes.

2 FEniCSx Container Setup

In this section we will be setting up the FEniCSx image and container so that it runs properly. We will also explain how to mount a folder in the volume so that it is shared between the host and Docker container.

2.1 Building a FEniCSx v0.8.0 installation

1. Pull the Docker image of the v0.8.0 release of dolfinx using this command in the MacOS terminal:

```
> Docker pull dolfinx/dolfinx:v0.8.0
```

2. Create a Docker container with a shared volume with the following command:

```
> Docker run -it -v /Users/eric/Desktop/FenicsWrk:/home/shared dolfinx/dolfinx:v0.8.0
```

   - Replace the example filepath `/Users/eric/Desktop/FenicsWrk` with the desired location of the shared folder on your local machine. You will need to create this folder before running this command.
   - After running this command, the container will be running in the terminal. Type in `exit` to leave MacOS terminal.

3. Open up the Docker Desktop dashboard. Make sure the new environment is running (the one with the `dolfinx/dolfinx:v0.8.0` image, as shown in Fig. 7. Open up the command line interface for this container by clicking the three-dots icon and selecting "Open in terminal" as shown in Fig. 7.

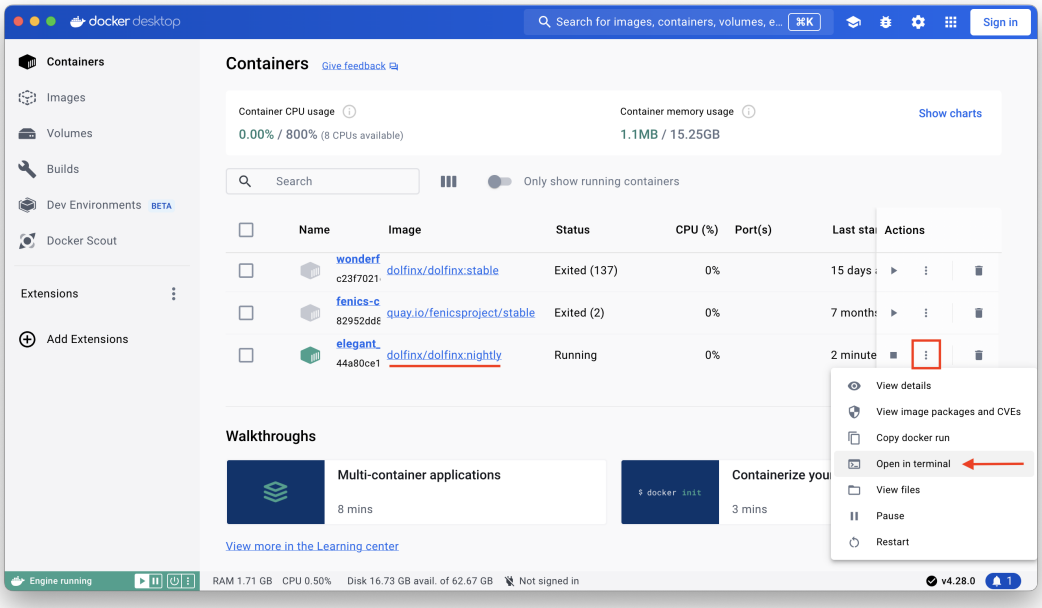


Figure 7: Opening the command line interface in Docker Desktop.

4. In the container's terminal window, run the following commands. First,

```
apt-get update && apt-get clean && apt-get install -y xvfb vim ffmpeg
```

then, and this one line is only necessary if you are using an M-series / Apple silicon Mac,

```
pip3 install "https://github.com/finsberg/vtk-aarch64/releases/download/vtk-9.2.6-cp310
    /vtk-9.2.6.dev0-cp310-cp310-linux_aarch64.whl"
```

Finally, (for both M-series and Intel Macs),

```
pip3 install "pyvista[all,trame]" jupyterlab ipython ipywidgets
```

5. Your Docker container is now properly set up for running Jupyter notebooks with code for FEniCSx (version 0.8.0), with PyVista visualizations. You can connect to this container in an instance of VSCode using the instructions below.

3 Editing and running code in Visual Studio Code

We use Visual Studio Code (VSCode) to edit and run FEniCSx codes in the form of Jupyter notebooks (.ipynb) and Python scripts (.py). If you do not have VScode, install it on your system:

- `https://code.visualstudio.com/`

Open up VScode, click the extensions tab (icon on the far left with three blocks and a fourth block clicking into place). Search for the **Docker** extension published by Microsoft shown in the Figure below, and install it. Also, search for and install the **Dev Containers** extension which allows us to use the full suite of VSCode development tools within a container.

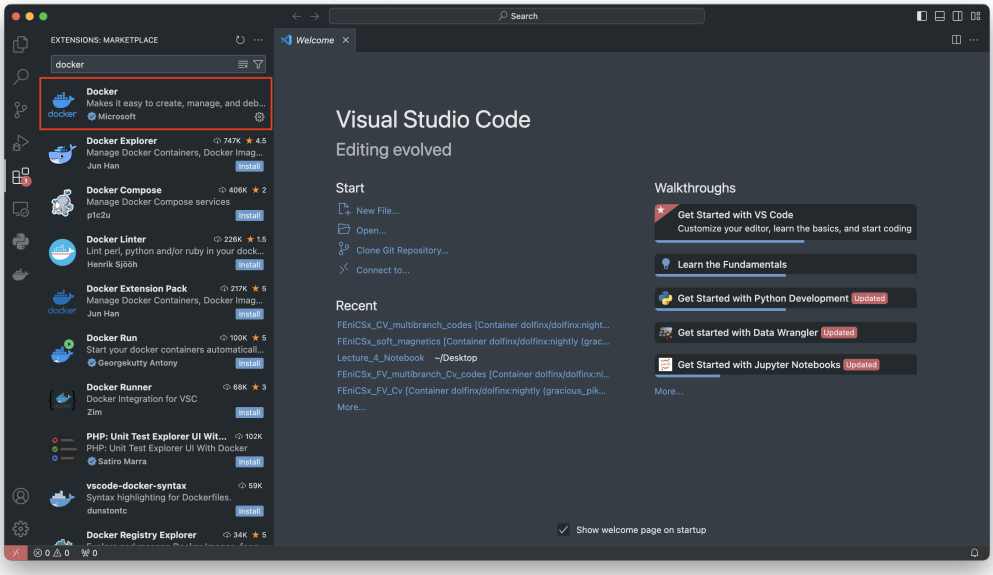


Figure 8: Installing the Docker extension.

3.1 Connecting to the Container through VSCode

We now launch VSCode and connect to the container we just created.

1. We can go to the Docker extension, and find the container we just launched. Right clicking on any of the containers will allow us to attach a VSCode instance.

2. Right click on the "v0.8.0" FEniCSx container and then click `Attach Visual Studio Code`. This will open a new window of VSCode.

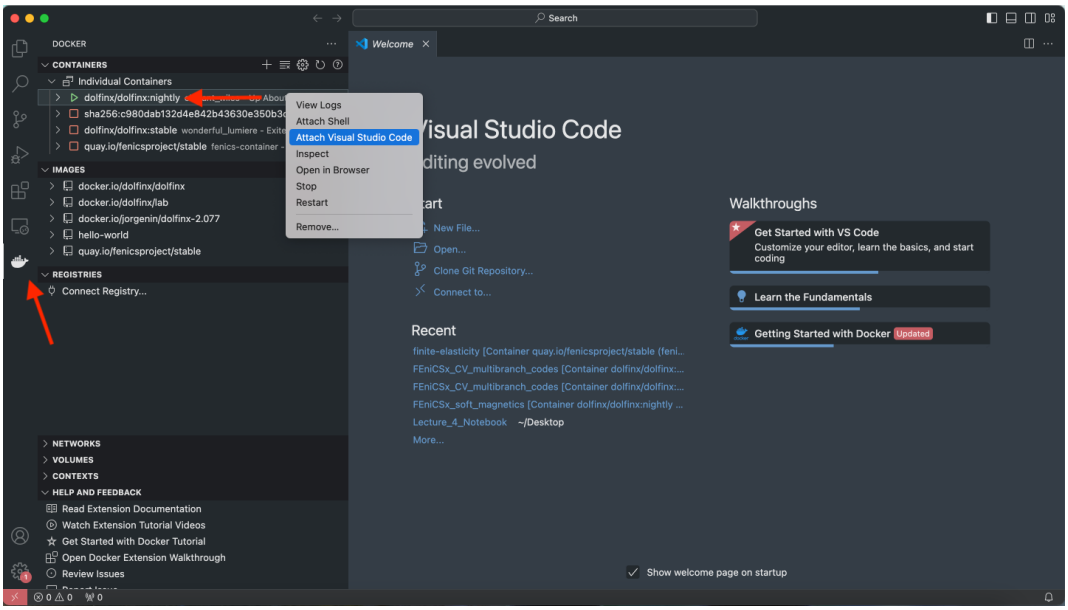


Figure 9: How to open up the Docker extension and launch into the file. Right clicking on any of the containers will allow us to attach a VSCode instance.

3. Ensure we are connected by looking at the bottom left corner of the screen.

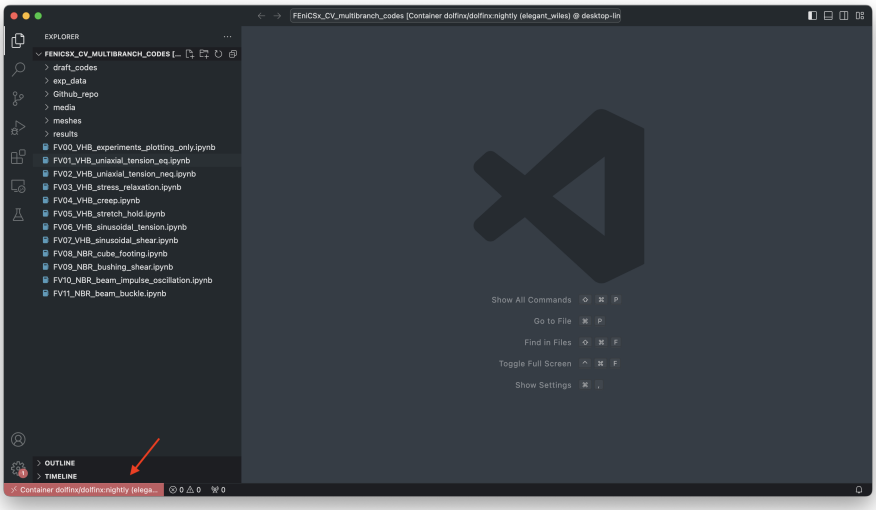


Figure 10: A VSCode instance which is "attached" to our v0.8.0 FEniCSx container. The red arrow shows where VSCode indicates the container to which you are connected.

4. You can now close the original instance of VSCode which is not connected to the Docker container.

5. To run FEniCSx codes, search for the **Python** and **Jupyter** extensions and install them. These extensions will also automatically install their dependencies, which contain the tools necessary to run Jupyter notebooks and produce interactive in-line plots.[2]

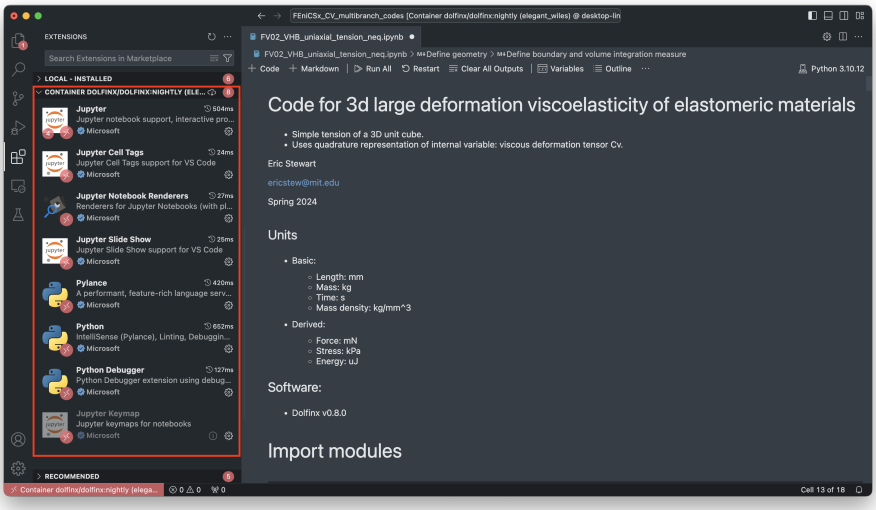


Figure 11: Extensions in the host are not automatically carried over to the container. The first time you attach to a container, you will need to install any extensions you wish to use. Installing the **Python** and **Jupyter** extensions will result in all necessary dependencies — the 8 individual extensions shown — being installed in the container.

[2]Some extensions may require you to reload the VScode window and re-attach to the Docker container to install properly.

4 Quick usage guide for the FEniCSx container

After the container has first been created and set up as described in this document, you can use the Docker container to run FEniCSx codes in the future by following these steps.

1. Start up Docker Desktop. In the Docker dashboard, Click the "play" button next to your `dolfinx/dolfinx:v0.8.0` container to start it.
2. Open VSCode. Attach an instance of VSCode to the running FEniCSx Docker container using the Docker extension (cf. Section 3.1).
3. In the VSCode window connected to your FEniCSx container, navigate to your shared directory `/home/shared` using `File -> Open Folder`, and open up and run the FEniCSx code of your choice.
    - This directory will be populated with the contents of your local machine's folder which is shared with the container, e.g. your `FEniCSWrk` folder from Section 2.1. Put your FEniCSx codes here to run them.
    - The first time you run a code, a pop-up menu may ask you to choose which Python kernel to use. Select the option for the python kernel installed in the Docker container, in the location `/usr/bin/python3`.

Note that:

- The first time you attach VSCode to this container, you will have to install the **Python** and **Jupyter** extensions (which automatically also installs their dependencies, a total of 8 packages).
- After these packages have been installed once, future instances of VSCode that attach to this container will already have the extensions installed.