ece 260c. **Software Setup Guide**

Welcome to ECE 260C SP25. This guide covers the setup process for the major tools we'll use:

- OpenROAD
- OpenROAD-flow-scripts
- Yosys
- Miscellaneous tools that the OpenROAD/Flow rely on like OpenSTA

OpenROAD is the (open-source) implementation tool you'll use in this course. It takes the place of Cadence Innovus or Synopsys IC Compiler, which you may have used in ECE 260B. To implement your design, you can use OpenROAD by leveraging tcl and Python scripts.

Yosys takes the place of synthesis tools like Cadence Genus or Synopsys Design Compiler. It can also be used with tcl.

In ECE 260B, you wrote or copied in scripts that you would go on to execute using tools like Innovus. These scripts (like this one) and your execution procedures come together to make what is called your implementation flow. OpenROAD-Flow-Scripts packages several implementation scripts together for various designs across different technologies (see the above link). **In this course, we will see how we can apply ORFS' existing scripts and understand when to deploy more advanced custom scripting to implement a design.**

Contents

System Requirements

These specs are recommended to ensure you can complete your runs in a timely manner:

- A 64-bit x86 or Arm-based CPU from Intel, AMD, or Apple.
- 8 GB of RAM minimum, 16 GB recommended
- About 20-40 GB of free storage
- **Optionally**, A PyTorch-compatible GPU – usually any recent enough GPU, integrated or discrete, from Intel, AMD, Nvidia, or Apple

Installation Methods

OpenROAD/Yosys are UNIX applications that can run on Linux or macOS but, due to their complexity, it's generally recommended to run them within containers. In this course, we recommend using **Docker**. We provide a very brief summary of Docker in the next section. Students interested in a more detailed understanding of the tool are welcome to check tutorial links such as this.

UNIX users can set up OpenROAD to run on their computers directly without Docker but we do not recommend it, especially on macOS. Direct installation will not be covered in this guide but if you are interested, please refer to the ORFS documentation.

Please note that, due to resource/permission constraints, you cannot install these applications (OpenROAD/yosys and Docker) on ieng6. If you do not have access to a machine powerful enough to run OpenROAD, **please use the Purdue ChipsHub's OpenLANE2 tool linked here.**

What is Docker? (brief summary)

Docker is a containerization system that allows you to run isolated Linux images on your computer in a lightweight manner. On Linux machines, this uses Linux kernel mechanisms to provide isolation. On Macs and Windows PCs, this uses a shared Linux VM.

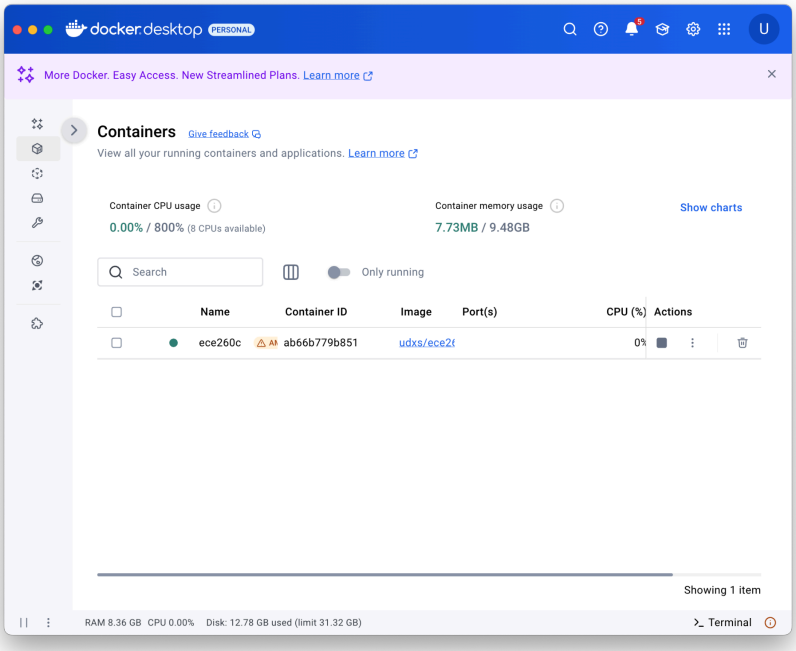
When using Docker, you write a Dockerfile that describes the commands that Docker should run to create an image, containing your choice of **dependencies** – usually an operating system and installed packages. This image can then be recorded and distributed. Images may be used to create containers that each have the same dependencies as the original image, with Docker efficiently recording only the changes you make on top of that image. An image can be 'run' to create a new container and from there a container can be suspended and restarted as you

please. For our use case, we will use interactive containers, which present you with a shell just like you are already used to.

For this course, we will be providing you with images that you can use to create persistent containers. For this guide, we will use our first image, suitable for the first labs, which we call *Image I: EDA Essentials*.

Installing Docker

Windows or macOS Users: Install Docker Desktop from this link here, which pairs the Docker engine we discussed above with a helpful GUI:

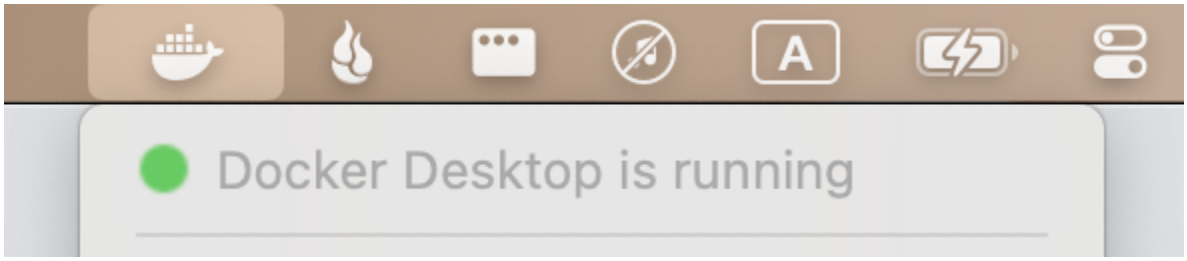


Linux Users: It's up to you whether you want to install Docker Desktop (link above) or just use the CLI-only Docker Engine, which you can install from here. For best performance, do not run both Docker Desktop and Docker CLI at the same time. Ubuntu Users: do not install Docker Engine from snap.

Please note again that you cannot install Docker on ieng6.

Preparing Docker

After Docker is installed, run Docker Desktop (or, on Linux, start the Docker service). You'll be able to see an indicator that the engine is running after a while. It will look similar to this:



For Windows/Linux Users: You **don't need** to make any changes to Settings.

For macOS Users: Go to ⚙ Settings > Resources in Docker Desktop and set the resource allocation to the following:

- CPU: As many cores/threads as you have.
- Memory Limit: 6 GB or 4 GB less than your total system RAM, whichever is higher
- Swap: 2 to 4 GB
- Disk Usage Limit: 32 GB at least.

Note for Arm Mac users (M1 chip or newer)

Go to ⚙ Settings > General in Docker Desktop, scroll down to Virtual Machine Options. Then, select Apple Virtualization Framework as the VMM. Then check "Use Rosetta for x86_64/amd64 emulation on Apple Silicon." This will greatly improve OpenROAD's performance.

You may also need to use the `--platform linux/amd64` flag in the `docker run` command (shown **below**).

Running Docker for this Course

Docker commands can be run on your system terminal. This could be:

- Terminal on macOS
- Windows Terminal

- Windows Powershell
- GNOME Terminal, Alacritty, etc. on Linux

When we want to run OpenROAD without any GUI (not sufficient for running ORFS – scroll to *Enabling GUI* to see the full run command), we can create an interactive container in your terminal using:

```
docker run -it --name my-lab-0 ghcr.io/udxs/ece260c-essential:main
```

(**M1 or newer Mac users:** add the `--platform linux/amd64` flag here)

Breakdown of the above command: Name your container (which is what will hold your files) `my-lab-0` and run it with the `-interactive` and `-tty` flags to allow you to enter its interactive shell like in a normal SSH session. This creates a new container named `my-lab-0` and opens the session for you. It'll look something like this:

```
                 ___ ___ ___
 ___ ___ ___    |_  |  _|   |___
| -_|  _| -_|   |  _| . | | |  _|_
|___|___|___|   |___|___|___|___|_|


ECE 260C Image I: EDA Essentials
        Includes OpenROAD, Yosys, KLayout, Git, and GitHub CLI. GUI Supported.

Components Copyright (c) 2025 The Regents of the University of California

A copy of OpenROAD Flow Scripts is provided in this container as a faster alternative to git clone.
You can copy it in using the orfs_copy or orfs_copy [dest] command.

(base) root@ab66b779b851:/# exit
```

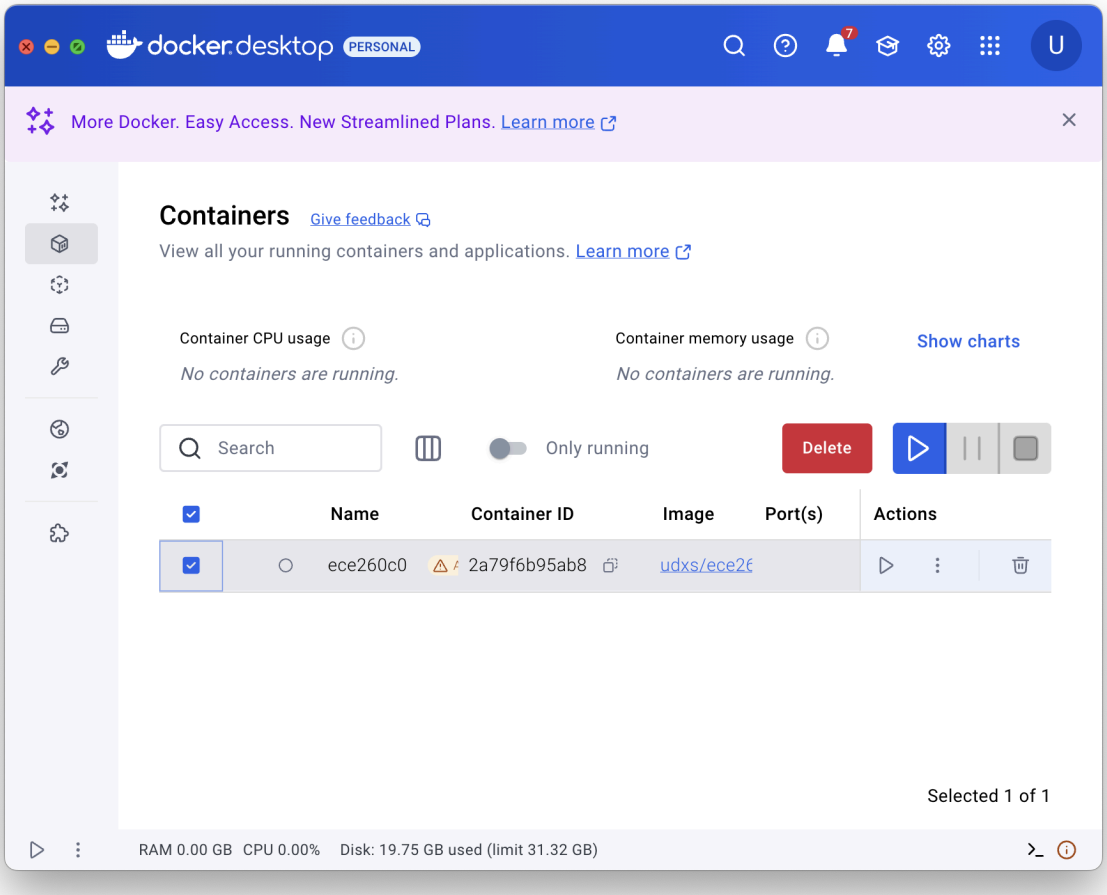
Suspending and Unsuspending the Container

When you exit this shell, either by using the `exit` command, closing the terminal window, or using Ctrl+C. The container will be suspended. We can unsuspend it by using:

```
docker start -i my-lab-0
```

Please avoid creating too many containers as this can overwhelm your resource limits (causing strange errors in runtime). **We suggest you create a single container called something like "ece260c1" and reuse it. Alternatively, you can create multiple containers, one per lab, but you may want to delete older ones when you're done later**.

Use Docker Desktop or reference the Docker CLI guide to know how to check for existing containers and to see their execution status. **You can delete containers** in the GUI by going to the container tab in the sidebar, selecting the checkbox next to a container, and hitting Delete:



.

Please note that running `docker run` with a given image for the first time will trigger a download. The `ece260c-essential` image is 7 GB and can take a while to download.

Enabling GUI

We can use the GUI version of OpenROAD through X11, a system for rendering GUIs on.

For macOS, follow the guide here: Xorg/X11 forwarding on macOS and docker
After, you can simply run

```
 docker run -it --name my-lab-0 -e DISPLAY=docker.for.mac.host.internal:0
ghcr.io/udxs/ece260c-essential:main
```

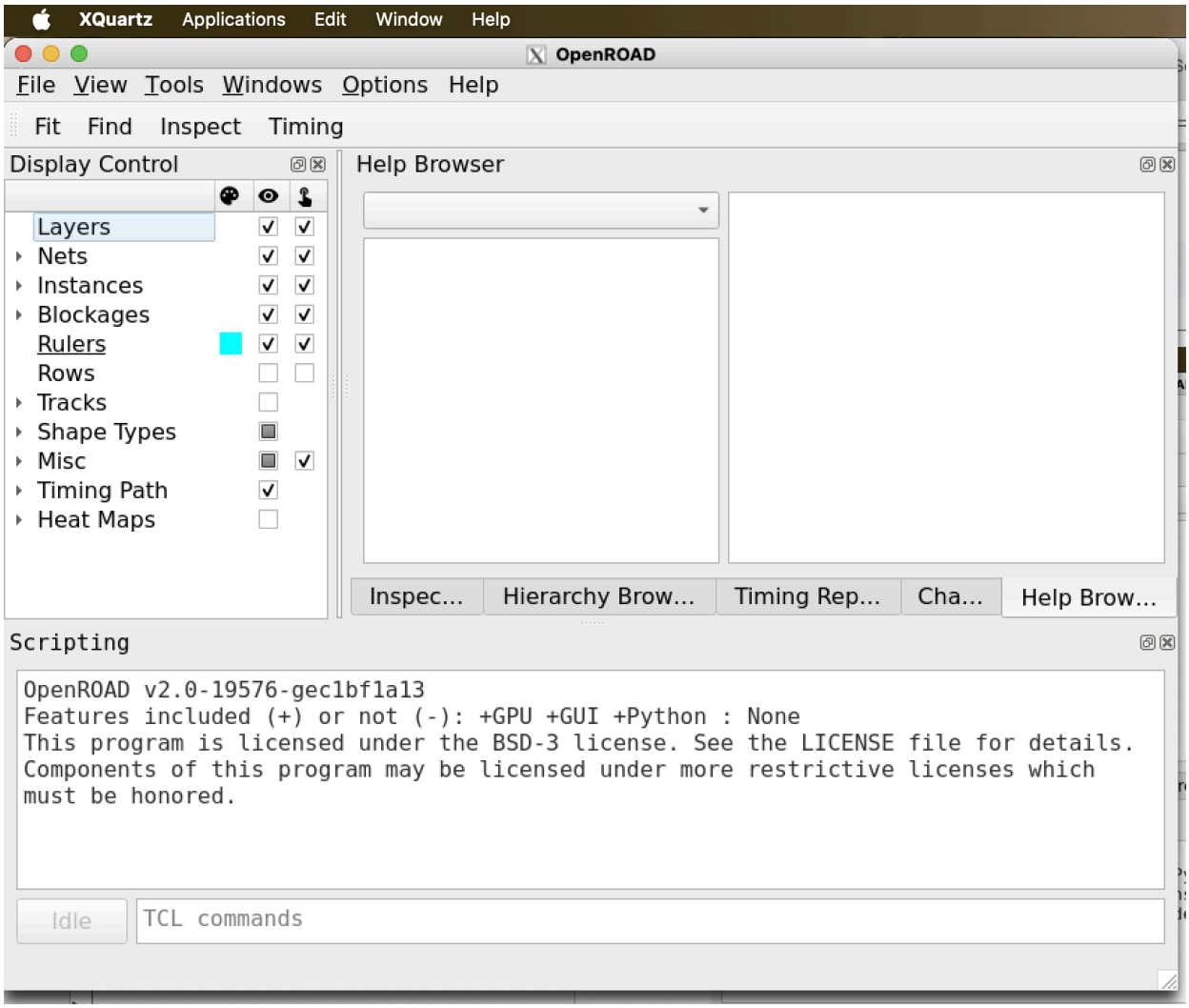
The `docker start` command remains the same as in *Suspending and Unsuspending the Container* section – this only needs to be done for the container's creation (whenever you use `docker run`).

For Windows 11 (or Windows 10 Build 19044+), you can use this version of the command to create a container with the name `ece260c0`, using Windows Powershell:

```
docker run -it -v /run/desktop/mnt/host/wslg/.X11-unix:/tmp/.X11-unix ` -v
/run/desktop/mnt/host/wslg:/mnt/wslg ` -e DISPLAY=:0 ` -e
WAYLAND_DISPLAY=wayland-0 ` -e XDG_RUNTIME_DIR=/mnt/wslg/runtime-dir `
--name ece260c0 ghcr.io/udxs/ece260c-essential:main
```

For Linux, there are many variables particularly depending on what compositor you're using. Ubuntu users should first ensure they **do not have Docker installed in Snap**. You may try x11docker but this will create a new container on every run so you may need to experiment with volume mounts to persist your data.

You can test if the GUI is working by running `openroad -gui`.



Continued Usage – Saving your Files with Git

Unless you create a mount (not covered in this guide), data will persist solely within your container and will disappear when you delete the container. To backup your results, you can use standard Git commands (commit, clone, push, pull, etc.).
In the container, we have provided `git` and the helper tool `gh` – the GitHub command line. Use `gh auth login` to setup git for your private GitHub repositories. This will start a login interface. Select the **options below**, go to https://github.com/login/device, and paste in your uniquely generated code when prompted:

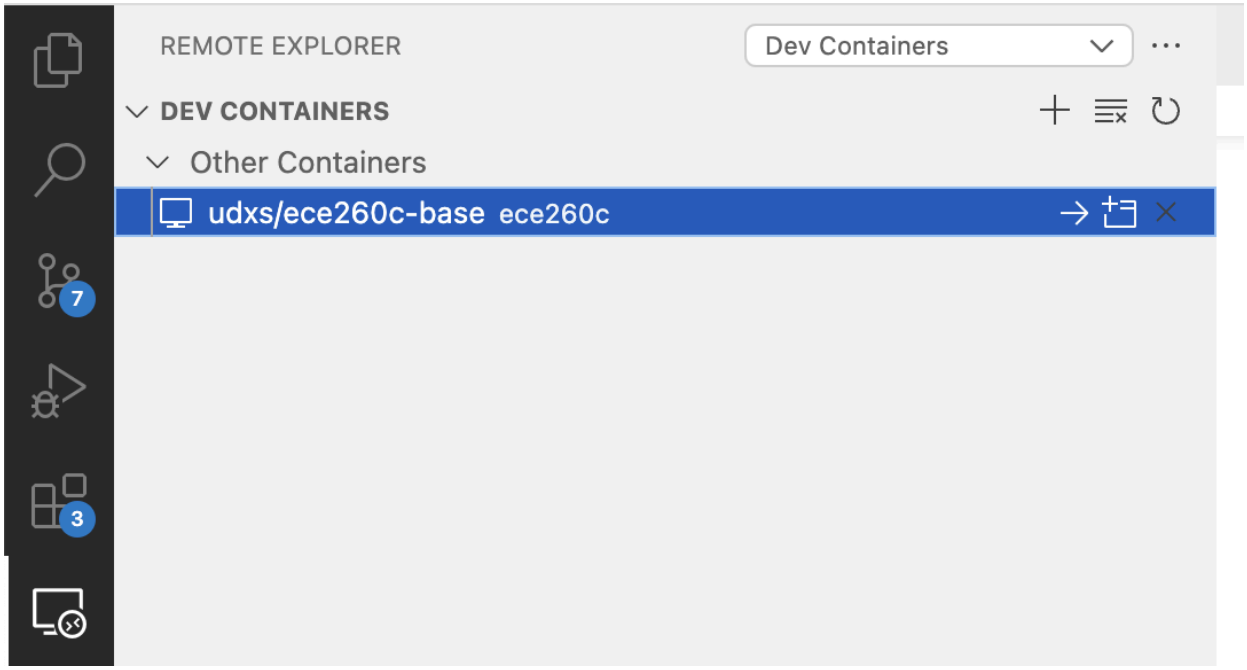
```
(base) root@2a79f6b95ab8:/# gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 3889-F17C
Press Enter to open https://github.com/login/device in your browser...
```

After a **successful login**, you will be able to use `gh repo clone your-repo-name` to clone the actual repo.

Since we'll be using GitHub Classroom, you will need to successfully set this up.

Using VS Code Dev Containers

Visual Studio Code provides the most flexible way to work with your container. With VS Code installed, follow this link to the Extensions Marketplace and install the Dev Containers addon. When this is done, you'll be able to open Remote Explorer in the Sidebar (**screenshot below**). Then, you can select the container you created above.
Right click > Attach in Current Window will allow you to open the container. **By default, the VS Code window opens into /root** but you can go to File > Open Folder... to open the directory you want to work on (e.g., for lab 0, /work/orfs_lab0).



Once connected, you can return to the Explorer tab in the sidebar to view/edit files and view images.

Note: The first time you do this for a given container, it can take a minute to install the remote server inside the container.

If you have any questions or problems, please post them on Piazza or send an email to a TA with “ECE 260C” in the subject line.