Local OCaml Environment Setup

In this tutorial, we will introduce three approaches to setup an OCaml development environment on your local machine. Following any one of them will be sufficient.

- If you use VS Code and don't want to install an OCaml toolchain on your machine, you can pick the **VS Code + Docker** approach (the most carefree one).
- If you don't feel like using VS Code but know how to use docker, you can pick the **Docker CLI** approach
- The **"Bare Metal"** approach resembles the set of instructions provided in CMSC 330

OCaml DE with Docker and VS Code

In this section, we will learn about how to use Visual Studio Code and Docker container to develop, compile and test OCaml programs on your local machine. This method uses the **Remote - Container** extension available on VS Code and a Docker image that ships a suite of OCaml tools running on top of Arch Linux.

Preparation

Before we get started, please have the following tools installed

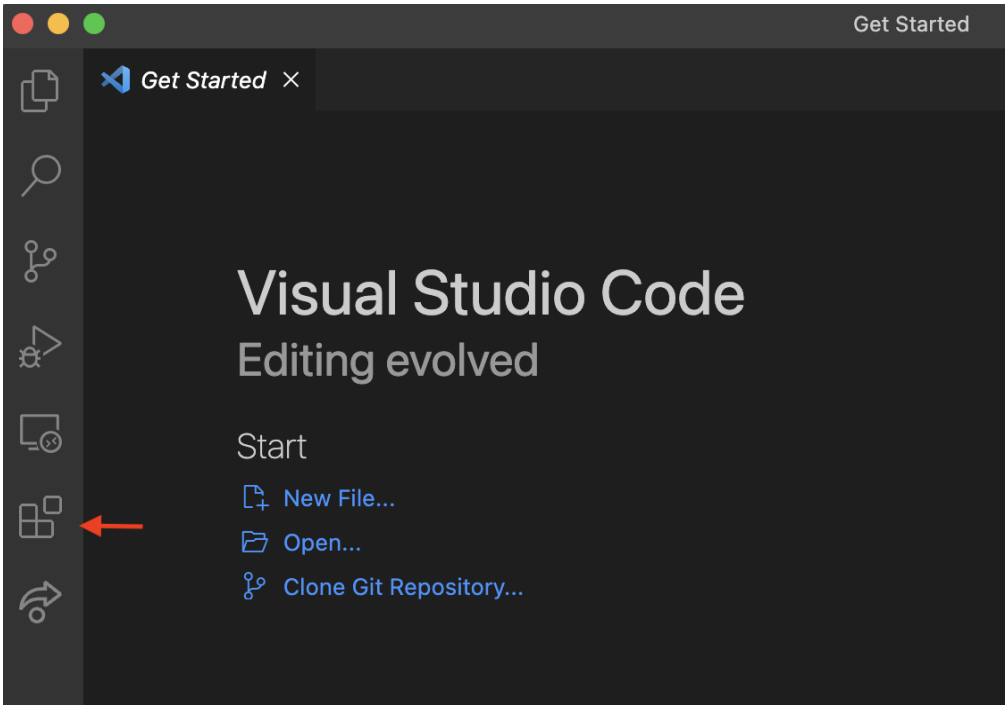
- Visual Studio Code
- Docker platform

Besides these softwares, make sure that you have downloaded and extracted `ocaml-dev.zip` (available on ELMS). We will use this directory to do the workspace setup, and we suggest putting all OCaml projects in this workspace directory.

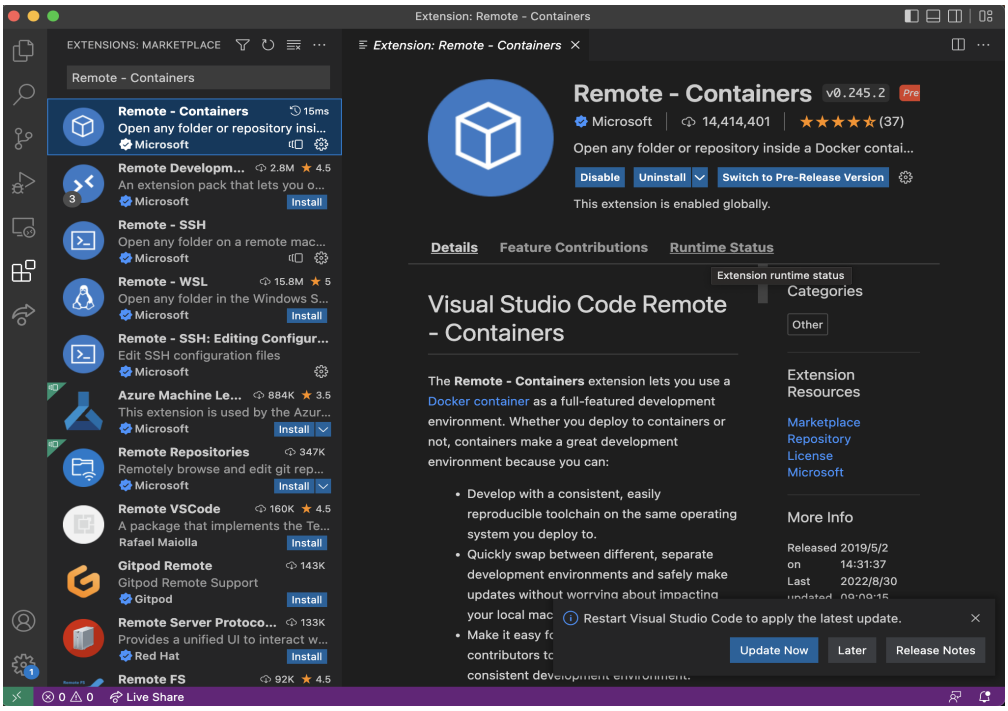
You may rename the directory but **do not** delete `.devcontainer/`.

Extension Installation

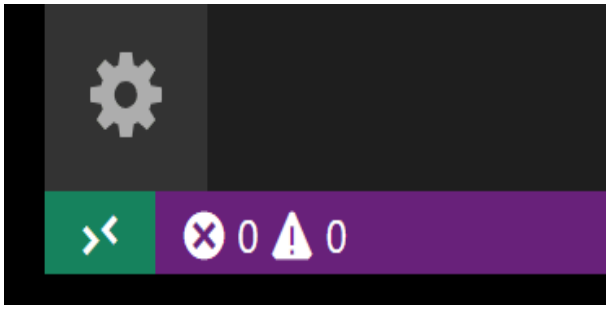
Open VS Code, go to the extension on the left column.



Search **Remote - Containers** and install this extension.

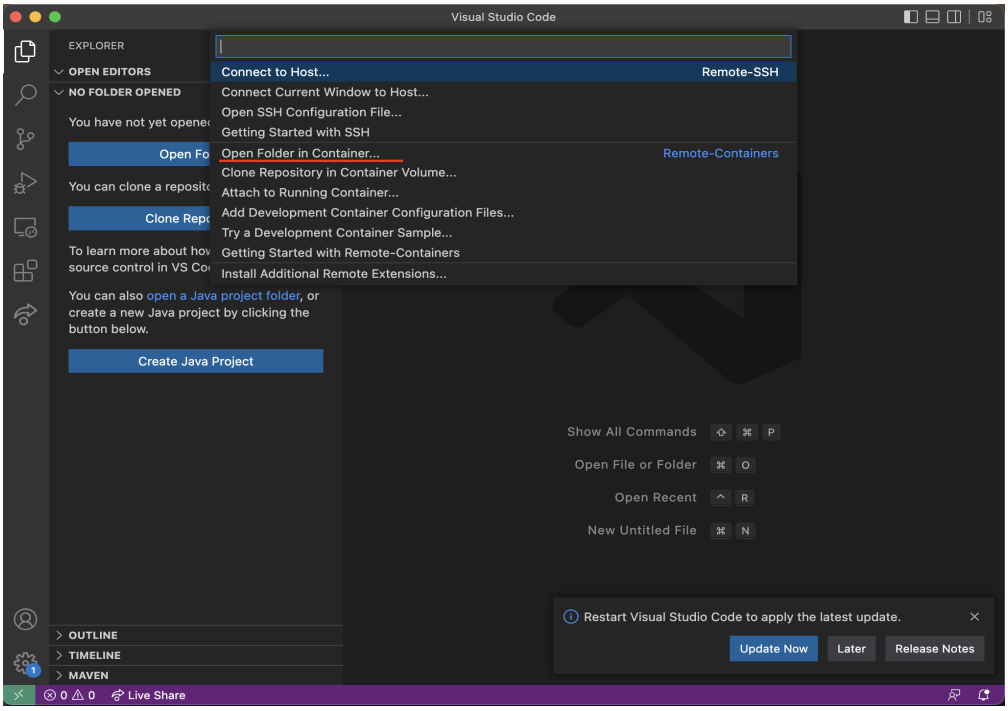


With the **Remote - Containers** extension installed, you will see a new Status bar item at the far left.



Docker Development Environment Initialization

Click on the Status bar item and select **Open Folder in Container...**.



Then select the `ocaml-dev` directory you extracted and wait the container to be loaded. This might take a while.

Once it finishes, you can access tools such as `opam`, `ocamlc` and `dune` via VS Code terminal. Those tools are provided by the OCaml DE container we've just setup.

Move Files

Copy the OCaml code file to the `ocaml-dev` directory. You will be able to see those files from the `File` tab.

OCaml DE with Docker CLI

(In this part, we assume that you know how to install docker and access terminal as well as some UNIX basics.)

A Docker image that contains a minimal OCaml development environment can be fetched by running
`docker pull mzhu65536:arch-ocaml:4.14.0-dev`.

You can use the following command to launch an interactive shell with
current directory mounted to `/opt` in the container environment.
`docker run -it --rm -v $(pwd):/opt mzhu65536/arch-ocaml:4.14.0-dev`

Please refer to the documentation about `docker run` for advanced usages and the meaning of each flag.

Bare Metal

You can find installation instructions for your system at:
https://opam.ocaml.org/doc/Install.html

- MacOSX: https://opam.ocaml.org/doc/Install.html#macOS
- Ubuntu: https://opam.ocaml.org/doc/Install.html#Ubuntu

  You might need to do `apt install gcc, binutils-dev, make and pkg-config`.
- *Windows*: https://opam.ocaml.org/doc/Install.html#Windows
  (If you are using WSL running Ubuntu, then you should follow the instructions for Ubuntu.)

Once `opam` is installed, run the following sequence of commands.

```
opam init -y -a && opam update
opam switch create 4.14.0 && opam switch 4.14.0
eval $(opam env)
opam install -y dune ounit utop
```

After finishing the installation, you can `cd` to the project directory.

Tips

`Makefile`

If the startup code contains a `Makefile`, you may run `make` to compile and run `make utop` to debug interactively.

`dune`

The following are some useful commands to interface with OCaml projects when a `dune` file is present:

- `dune build` (to compile program)
- `dune utop` (to debug in an interactive session)
- `dune clean` (to clean the building cache)
- `dune test` (to run the tests)

(For better editor supports, you may want to give `merlin` or `ocaml-lsp` a try.)