LiP lab

LiP students are expected to fork the repository and push solutions to the exercises on their fork.

Invitation to the Discord server: **https://discord.gg/JKPWbpXBf4**

[Tutorial] Getting started

Before diving into the course, let's take care of a few technicalities.

This set up process is quite involved and may seem overwhelming at first, but bear with it for now. It will pay off in the subsequent lessons. Arguably, the technologies we introduce here will very likely serve the purposes of other courses and even of your future projects.

1. Fork this repository

The first thing to do is to fork this repository. Forking will create a new repository that shares the same files and history of this repository, except it's owned by you and you can freely edit it.

You can find the "Fork" action at the top right of the page, next to "Star" button. Select it, then **keep the default options** and hit "Create fork".

By now you should be reading this guide from your fork's webpage.

Important

The teachers and tutors will update the upstream repository with new content from time to time.

To reflect these changes to your fork, you have to synchronize your fork regularly.

The sync action is easily accessible from your fork's main page via the button:

or in the GitHub CLI via the command `gh repo sync`.

2. Set up the development environment

Next, we'll configure your local OS for containerized development.

1. If you're on Windows, install the WSL 2 back-end. Follow the official instructions.
2. Install Docker Desktop.
3. Lastly, install Visual Studio Code.

[Windows users only] Configure WSL and Docker

Skip this section if you don't use Windows.

- Hit the keys `Win + S` and search for "WSL" or "Ubuntu". Clicking the first result should open a pitch-black window with white text on it.

Read it carefully, and make sure you understand it as you go through the initialization procedure. It will eventually ask you to enter a username and a password for your account. Note these down.

Everytime you start WSL, you should be logged in to your user account. This is made clear by the shell prompt ending with a `$` sign. If something still isn't quite right, refer to this Microsoft Learn guide for a proper setup.

•On Docker Desktop, make sure WSL 2 integration is enabled. To do so, perform the actions shown in this gif:

Install git

From now on we will be working solely on the command line of a Linux shell. If you're on Windows, that means you're going to be typing commands within a WSL shell running Ubuntu. Otherwise, as a Linux or macOS user, you're going to be using your OS's native shell.

Many commercial Linux distributions, including the one shipped with WSL, already come with `git` preinstalled, but it doesn't hurt to check:

```
git --version
```

If that command fails, then you must install `git` for your distro. For Ubuntu and WSL it boils down to the two commands:

```
sudo apt update
sudo apt install git
```

Install the GitHub CLI

Next, we want the GitHub CLI. The GitHub CLI is a useful tool to manage your online repositories from the comfort of the command line. We just need it to perform `git` commands as an authenticated user.

To install the GitHub CLI, follow the installation instructions for Linux. Then check it's installed with:

```
gh --version
```

Login to GitHub from the shell

First, authenticate to your GitHub account from the GitHub CLI. Run:

```
gh auth login
```

and follow the on-screen procedure carefully.

Next, we need to let `git` know about your GitHub profile. Run the following commands, being sure to use the username and the email of your GitHub account.

```
git config --global user.name <YOUR-USERNAME>
git config --global user.email <YOUR-EMAIL@EXAMPLE.COM>
```

From now on `git` will sign your commits with the given credentials and will act on the behalf of your GitHub account whenever you push to a remote repository, such as your fork.

Clone your LiP lab fork

Run the following command from your home folder, replacing `YOUR-USERNAME` with your GitHub username:

```
git clone https://github.com/YOUR-USERNAME/lip
```

This downloads a local copy of your fork in a new directory called `lip`.

3. Open the VS Code container

We will now invoke VS Code's command-line interface `code` to launch VS Code inside the `lip` folder:

```
code lip
```

Briefly after the VS Code window shows up, you should see a notification like the one in the image:

Click on the blue button and wait for a while. If everything goes well, VS Code will have opened the repo's container set up for OCaml development.

Try it out the Dev Container by opening the integrated terminal (`Ctrl + J`, then click on the "+" icon) and enter the command `utop`. Play with some OCaml expressions, then exit `utop` by pressing `Ctrl + D`.

Congrats! You are now ready to write and test OCaml code.

Note

The Dev Container you've just opened transforms your VS Code into a fully integrated OCaml IDE. It comprises the OCaml Platform extension and an installation of the OCaml compiler enriched with many useful libraries. In particular, this installation includes:

- **dune**, a build system for OCaml projects, similar to make;
- **utop**, a REPL interface for OCaml;
- **Menhir**, a parser generator;
- **ppx_inline_test**, a tool for writing in-line tests in OCaml.

First project

To check that everything is installed correctly, we set up a first project (see here for more detailed instructions).

Navigate to `lip/basics` folder. Then, create a new project called `helloworld` using dune. Below, the lines starting with `>` contain the expected output of the given shell commands:

```
dune init project helloworld
> Success: initialized project component named helloworld
```

This command creates a directory `helloworld` with the following file structure:

```
helloworld/
├── dune-project
├── bin
│   └── dune
│   └── main.ml
├── lib
│   └── dune
├── test
│   ├── dune
│   └── helloworld.ml
└── helloworld.opam
```

To check that the OCaml installation was successful, try to build the project from the `helloworld` directory:

```
cd helloworld
dune build
```

If there are no errors, the output should be empty. Run the project as follows:

```
dune exec helloworld
> Hello, World!
```

We will discuss the project structure in detail in the next exercise. For the moment, note that the `_build` directory contains the output of the `dune build` command. This includes the `main.exe` executable inside the `_build/default/bin/` subdirectory.

Note

In this very first project, all the source code is in `./bin/main.ml`. For more complex projects, we will mainly write our source code in the `lib` directory.

Push the solution to GitHub

We won't do much else in this first project, so let's see how to record our changes to the fork.

Assuming you're still under the `basics/helloworld` directory, run the following command:

```
git add .
```

Let's break it down. We're invoking `git` with two arguments. The first one, `add`, is a git subcommand that lets you add new or modified files to the set of changes that should be recorded in the next commit (i.e. the *index*). The second argument, `.`, stands for the current directory (`basics/helloworld`).

Tip

The command `git status` lets you review the changes you've staged for a commit.

If you wish to unstage a change from the index, invoke `git restore --staged` on the changed files.
Next, we commit the contents of the `helloworld` folder. A commit must be supplemented with a commit message describing the changes.

```
git commit -m "Create first dune project in basics/helloworld"
```

As before, `commit` is a subcommand of git. It accepts an optional argument, introduced by `-m`, that lets us specify the commit message as a string.

The changes are still stored on our local file system. To update the remote fork repository with your commit, issue the command:

```
git push
```

The dual action of pushing is pulling. You'll only need to pull some commits to your local fork when the teachers update the upstream repository and you have synced your fork as previously noted. The command to pull is, you guessed it:

```
git pull
```

This might not work if you have some pending changes not yet committed to your working tree. In this case you can temporarily store away the modified files with:

```
git stash
```

and restore them later on top of the newer commits using:

```
git stash apply
```

Tip

You can always append the `--help` option to any of the above git subcommands to fully explore their functionality.

We've shown just a few basic git commands here. Refer to the Git Cheat Sheet for more important commands.

Conclusion

You made it to the end of Getting Started tutorial! You and your system should now be ready to take on the more theoretical stuff.

Here's a final diagram to help you understand the lab workflow.

Now proceed to refresh your OCaml knowledge with these warm-up exercises:

1. Adder
2. Recognizer
3. Tug of war

Course outline

Lexing and parsing

1. A toy lexer

2. A toy parser
3. Game of life

Arithmetic expressions

1. A minimal language of boolean expressions
2. Boolean expressions with not, and, or
3. Typed arithmetic expressions with dynamic type checking
4. Typed arithmetic expressions with static type checking

Imperative languages

1. A simple imperative language
2. Declaration blocks
3. Functions

References

- OCaml Programming: Correct + Efficient + Beautiful
- OCaml from the very beginning
- B. Pierce. Types and Programming Languages. MIT Press, 2002

How to install Linux on Windows with WSL

- Article
- 08/28/2023
- 10 contributors

Feedback

In this article

Developers can access the power of both Windows and Linux at the same time on a Windows machine. The Windows Subsystem for Linux (WSL) lets developers install a Linux distribution (such as Ubuntu, OpenSUSE, Kali, Debian, Arch Linux, etc) and use Linux applications, utilities, and Bash command-line tools directly on Windows, unmodified, without the overhead of a traditional virtual machine or dualboot setup.

Prerequisites

You must be running Windows 10 version 2004 and higher (Build 19041 and higher) or Windows 11 to use the commands below. If you are on earlier versions please see the manual install page.

Install WSL command

You can now install everything you need to run WSL with a single command. Open PowerShell or Windows Command Prompt in **administrator** mode by right-clicking and selecting "Run as administrator", enter the wsl --install command, then restart your machine.

PowerShellCopy

```
wsl --install
```

This command will enable the features necessary to run WSL and install the Ubuntu distribution of Linux. (This default distribution can be changed).

If you're running an older build, or just prefer not to use the install command and would like step-by-step directions, see **WSL manual installation steps for older versions**.

The first time you launch a newly installed Linux distribution, a console window will open and you'll be asked to wait for files to de-compress and be stored on your machine. All future launches should take less than a second.

Note

The above command only works if WSL is not installed at all. If you run `wsl --install` and see the WSL help text, please try running `wsl --list --online` to see a list of available

distros and run `wsl --install -d <DistroName>` to install a distro. To uninstall WSL, see Uninstall legacy version of WSL or unregister or uninstall a Linux distribution.

Change the default Linux distribution installed

By default, the installed Linux distribution will be Ubuntu. This can be changed using the `-d` flag.

- To change the distribution installed, enter: `wsl --install -d <Distribution Name>`. Replace `<Distribution Name>` with the name of the distribution you would like to install.
- To see a list of available Linux distributions available for download through the online store, enter: `wsl --list --online` or `wsl -l -o`.
- To install additional Linux distributions after the initial install, you may also use the command: `wsl --install -d <Distribution Name>`.

Tip

If you want to install additional distributions from inside a Linux/Bash command line (rather than from PowerShell or Command Prompt), you must use .exe in the command: `wsl.exe --install -d <Distribution Name>` or to list available distributions: `wsl.exe -l -o`.

If you run into an issue during the install process, check the installation section of the troubleshooting guide.

To install a Linux distribution that is not listed as available, you can import any Linux distribution using a TAR file. Or in some cases, as with Arch Linux, you can install using an `.appx` file. You can also create your own custom Linux distribution to use with WSL.

Set up your Linux user info

Once you have installed WSL, you will need to create a user account and password for your newly installed Linux distribution. See the Best practices for setting up a WSL development environment guide to learn more.

Set up and best practices

We recommend following our Best practices for setting up a WSL development environment guide for a step-by-step walk-through of how to set up a user name and password for your installed Linux distribution(s), using basic WSL commands, installing and customizing Windows Terminal, set up for Git version control, code editing and debugging using the VS Code remote server, good practices for file storage, setting up a database, mounting an external drive, setting up GPU acceleration, and more.

Check which version of WSL you are running

You can list your installed Linux distributions and check the version of WSL each is set to by entering the command: `wsl -l -v` in PowerShell or Windows Command Prompt.

To set the default version to WSL 1 or WSL 2 when a new Linux distribution is installed, use the command: `wsl --set-default-version <Version#>`, replacing `<Version#>` with either 1 or 2.

To set the default Linux distribution used with the `wsl` command, enter: `wsl -s <DistributionName>` or `wsl --set-default <DistributionName>`, replacing `<DistributionName>` with the name of the Linux distribution you would like to use. For example, from PowerShell/CMD, enter: `wsl -s Debian` to set the default distribution to Debian. Now running `wsl npm init` from Powershell will run the `npm init` command in Debian.

To run a specific wsl distribution from within PowerShell or Windows Command Prompt without changing your default distribution, use the command: `wsl -d <DistributionName>`, replacing `<DistributionName>` with the name of the distribution you want to use.

Learn more in the guide to Basic commands for WSL.

Upgrade version from WSL 1 to WSL 2

New Linux installations, installed using the `wsl --install` command, will be set to WSL 2 by default.

The `wsl --set-version` command can be used to downgrade from WSL 2 to WSL 1 or to update previously installed Linux distributions from WSL 1 to WSL 2.

To see whether your Linux distribution is set to WSL 1 or WSL 2, use the command: `wsl -l -v`.

To change versions, use the command: `wsl --set-version <distro name> 2` replacing `<distro name>` with the name of the Linux distribution that you want to update. For example, `wsl --set-version Ubuntu-20.04 2` will set your Ubuntu 20.04 distribution to use WSL 2.

If you manually installed WSL prior to the `wsl --install` command being available, you may also need to enable the virtual machine optional component used by WSL 2 and install the kernel package if you haven't already done so.

To learn more, see the Command reference for WSL for a list of WSL commands, Comparing WSL 1 and WSL 2 for guidance on which to use for your work scenario, or Best practices for setting up a WSL development environment for general guidance on setting up a good development workflow with WSL.

Ways to run multiple Linux distributions with WSL

WSL supports running as many different Linux distributions as you would like to install. This can include choosing distributions from the Microsoft Store, importing a custom distribution, or building your own custom distribution.

There are several ways to run your Linux distributions once installed:

- Install Windows Terminal ***(Recommended)*** Using Windows Terminal supports as many command lines as you would like to install and enables you to open them in multiple tabs or

window panes and quickly switch between multiple Linux distributions or other command lines (PowerShell, Command Prompt, Azure CLI, etc). You can fully customize your terminal with unique color schemes, font styles, sizes, background images, and custom keyboard shortcuts. Learn more.

- You can directly open your Linux distribution by visiting the Windows Start menu and typing the name of your installed distributions. For example: "Ubuntu". This will open Ubuntu in its own console window.
- From Windows Command Prompt or PowerShell, you can enter the name of your installed distribution. For example: `ubuntu`
- From Windows Command Prompt or PowerShell, you can open your default Linux distribution inside your current command line, by entering: `wsl.exe`.
- From Windows Command Prompt or PowerShell, you can use your default Linux distribution inside your current command line, without entering a new one, by entering:`wsl [command]`. Replacing `[command]` with a WSL command, such as: `wsl -l -v` to list installed distributions or `wsl pwd` to see where the current directory path is mounted in wsl. From PowerShell, the command `get-date` will provide the date from the Windows file system and `wsl date` will provide the date from the Linux file system.

The method you select should depend on what you're doing. If you've opened a WSL command line within a Windows Prompt or PowerShell window and want to exit, enter the command: `exit`.

Want to try the latest WSL preview features?

Try the most recent features or updates to WSL by joining the Windows Insiders Program. Once you have joined Windows Insiders, you can choose the channel you would like to receive preview builds from inside the Windows settings menu to automatically receive any WSL updates or preview features associated with that build. You can choose from:

- Dev channel: Most recent updates, but low stability.
- Beta channel: Ideal for early adopters, more reliable builds than the Dev channel.
- Release Preview channel: Preview fixes and key features on the next version of Windows just before its available to the general public.

Additional resources

- Windows Command Line Blog: Install WSL with a single command now available in Windows 10 version 2004 and higher

Get Docker

Docker is an open platform for developing, shipping, and running applications.

Docker allows you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

You can download and install Docker on multiple platforms. Refer to the following section and choose the best installation path for you.

> **Docker Desktop terms**
>
> Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

Docker Desktop for Mac
A native application using the macOS sandbox security model that delivers all Docker tools to your Mac.

Docker Desktop for Windows
A native Windows application that delivers all Docker tools to your Windows computer.

Docker Desktop for Linux
A native Linux application that delivers all Docker tools to your Linux computer.

> **Note**
>
> If you're looking for information on how to install Docker Engine, see Docker Engine installation overview.

Install Docker Desktop on Windows

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

Docker Desktop for Windows - x86_64
Docker Desktop for Windows - Arm (Beta)

For checksums, see Release notes

System requirements

Tip

Should I use Hyper-V or WSL?

Docker Desktop's functionality remains consistent on both WSL and Hyper-V, without a preference for either architecture. Hyper-V and WSL have their own advantages and disadvantages, depending on your specific set up and your planned use case.

WSL 2 backend, x86_64 Hyper-V backend, x86_64 WSL 2 backend, Arm (Beta)

- WSL version 1.1.3.0 or later.
- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit:
    - We recommend Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher.
    - Minimum required is Home or Pro 21H2 (build 19044) or higher, or Enterprise or Education 21H2 (build 19044) or higher.
- Turn on the WSL 2 feature on Windows. For detailed instructions, refer to the Microsoft documentation.
- The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:
    - 64-bit processor with Second Level Address Translation (SLAT)
    - 4GB system RAM
    - Enable hardware virtualization in BIOS. For more information, see Virtualization.

For more information on setting up WSL 2 with Docker Desktop, see WSL.

Note

Docker only supports Docker Desktop on Windows for those versions of Windows that are still within Microsoft's servicing timeline. Docker Desktop is not supported on server versions of Windows, such as Windows Server 2019 or Windows Server 2022.

For more information on how to run containers on Windows Server, see Microsoft's official documentation.

Important

To run Windows containers, you need Windows 10 or Windows 11 Professional or Enterprise edition. Windows Home or Education editions only allow you to run Linux containers.

Containers and images created with Docker Desktop are shared between all user accounts on machines where it is installed. This is because all Windows accounts use the same VM to build and run containers. Note that it is not possible to share containers and images between user accounts when using the Docker Desktop WSL 2 backend.

Running Docker Desktop inside a VMware ESXi or Azure VM is supported for Docker Business customers. It requires enabling nested virtualization on the hypervisor first. For more information, see Running Docker Desktop in a VM or VDI environment.

How do I switch between Windows and Linux containers?

Install Docker Desktop on Windows

Install interactively

1. Download the installer using the download button at the top of the page, or from the release notes.
2. Double-click `Docker Desktop Installer.exe` to run the installer. By default, Docker Desktop is installed at `C:\Program Files\Docker\Docker`.
3. When prompted, ensure the **Use WSL 2 instead of Hyper-V** option on the Configuration page is selected or not depending on your choice of backend.

   If your system only supports one of the two options, you won't be able to select which backend to use.
4. Follow the instructions on the installation wizard to authorize the installer and proceed with the install.
5. When the installation is successful, select **Close** to complete the installation process.
6. Start Docker Desktop.

If your administrator account is different to your user account, you must add the user to the **docker-users** group:

1. Run **Computer Management** as an **administrator**.
2. Navigate to **Local Users and Groups** > **Groups** > **docker-users**.
3. Right-click to add the user to the group.
4. Sign out and sign back in for the changes to take effect.

Install from the command line

After downloading `Docker Desktop Installer.exe`, run the following command in a terminal to install Docker Desktop:

```
$ "Docker Desktop Installer.exe" install
```

If you're using PowerShell you should run it as:

```
Start-Process 'Docker Desktop Installer.exe' -Wait install
```

If using the Windows Command Prompt:

```
start /w "" "Docker Desktop Installer.exe" install
```

By default, Docker Desktop is installed at `C:\Program Files\Docker\Docker`.

The `install` command accepts the following flags:

- `--quiet`: Suppresses information output when running the installer
- `--accept-license`: Accepts the Docker Subscription Service Agreement now, rather than requiring it to be accepted when the application is first run
- `--no-windows-containers`: Disables the Windows containers integration
- `--allowed-org=<org name>`: Requires the user to sign in and be part of the specified Docker Hub organization when running the application
- `--backend=<backend name>`: Selects the default backend to use for Docker Desktop, `hyper-v`, `windows` or `wsl-2` (default)
- `--installation-dir=<path>`: Changes the default installation location (`C:\Program Files\Docker\Docker`)
- `--admin-settings`: Automatically creates an `admin-settings.json` file which is used by admins to control certain Docker Desktop settings on client machines within their organization. For more information, see Settings Management.
    - It must be used together with the `--allowed-org=<org name>` flag.
    - For example:

      ```
      --allowed-org=<org name> --admin-
      settings='{"configurationFileVersion": 2,
      "enhancedContainerIsolation": {"value": true, "locked": false}}'
      ```

- `--proxy-http-mode=<mode>`: Sets the HTTP Proxy mode, `system` (default) or `manual`
- `--override-proxy-http=<URL>`: Sets the URL of the HTTP proxy that must be used for outgoing HTTP requests, requires `--proxy-http-mode` to be `manual`
- `--override-proxy-https=<URL>`: Sets the URL of the HTTP proxy that must be used for outgoing HTTPS requests, requires `--proxy-http-mode` to be `manual`

- `--override-proxy-exclude=<hosts/domains>`: Bypasses proxy settings for the hosts and domains. Uses a comma-separated list.
- `--proxy-enable-kerberosntlm`: Enables Kerberos and NTLM proxy authentication. If you are enabling this, ensure your proxy server is properly configured for Kerberos/NTLM authentication. Available with Docker Desktop 4.32 and later.
- `--hyper-v-default-data-root=<path>`: Specifies the default location for the Hyper-V VM disk.
- `--windows-containers-default-data-root=<path>`: Specifies the default location for the Windows containers.
- `--wsl-default-data-root=<path>`: Specifies the default location for the WSL distribution disk.
- `--always-run-service`: After installation completes, starts `com.docker.service` and sets the service startup type to Automatic. This circumvents the need for administrator privileges, which are otherwise necessary to start `com.docker.service`. `com.docker.service` is required by Windows containers and Hyper-V backend.

  Note

  If you're using PowerShell, you need to use the `ArgumentList` parameter before any flags. For example:

  ```
  Start-Process 'Docker Desktop Installer.exe' -Wait -ArgumentList
  'install', '--accept-license'
  ```

If your admin account is different to your user account, you must add the user to the **docker-users** group:

```
$ net localgroup docker-users <user> /add
```

Start Docker Desktop

Docker Desktop does not start automatically after installation. To start Docker Desktop:

1. Search for Docker, and select **Docker Desktop** in the search results.

2. The Docker menu ( ) displays the Docker Subscription Service Agreement.

   Here’s a summary of the key points:

   - Docker Desktop is free for small businesses (fewer than 250 employees AND less than $10 million in annual revenue), personal use, education, and non-commercial open source projects.
   - Otherwise, it requires a paid subscription for professional use.
   - Paid subscriptions are also required for government entities.
   - The Docker Pro, Team, and Business subscriptions include commercial use of Docker Desktop.

3. Select **Accept** to continue. Docker Desktop starts after you accept the terms.

   Note that Docker Desktop won't run if you do not agree to the terms. You can choose to accept the terms at a later date by opening Docker Desktop.

   For more information, see Docker Desktop Subscription Service Agreement. It is recommended that you read the FAQs.

Tip

As an IT administrator, you can use endpoint management (MDM) software to identify the number of Docker Desktop instances and their versions within your environment. This can provide accurate license reporting, help ensure your machines use the latest version of Docker Desktop, and enable you to enforce sign-in.

- Intune
- Jamf
- Kandji
- Kolide
- Workspace One

Where to go next

- Explore Docker's core subscriptions to see what Docker can offer you.
- Get started with Docker.
- Explore Docker Desktop and all its features.
- Troubleshooting describes common problems, workarounds, and how to get support.
- FAQs provide answers to frequently asked questions.
- Release notes lists component updates, new features, and improvements associated with Docker Desktop releases.
- Back up and restore data provides instructions on backing up and restoring data related to Docker.