Download and install

Note

You can also build Emscripten from source if you prefer that to downloading binaries using the emsdk.

Tip

if you'd like to install emscripten using the **unofficial** packages instead of the **officially supported** emsdk, see the bottom of the page.

Installation instructions using the emsdk (recommended)

First check the Platform-specific notes below and install any prerequisites.

The core Emscripten SDK (emsdk) driver is a Python script. You can get it for the first time with

```
# Get the emsdk repo
git clone https://github.com/emscripten-core/emsdk.git

# Enter that directory
cd emsdk
```

Note

You can also get the emsdk without git, by selecting "Clone or download => Download ZIP" on the emsdk GitHub page.

Run the following emsdk commands to get the latest tools from GitHub and set them as active:

```
# Fetch the latest version of the emsdk (not needed the first time you clone)
git pull

# Download and install the latest SDK tools.
./emsdk install latest

# Make the "latest" SDK "active" for the current user. (writes .emscripten file)
./emsdk activate latest

# Activate PATH and other environment variables in the current terminal
source ./emsdk_env.sh
```

Note

On Windows, run `emsdk.bat` instead of `./emsdk`, and `emsdk_env.bat` instead of `source ./emsdk_env.sh`.

Note

On Windows, if you use the `activate` command, the step of `emsdk_env.bat` is optional. If you want to know more, see activate SDK version.

Note

`git pull` will fetch the current list of tags, but very recent ones may not yet be present there. You can run `./emsdk update-tags` to update the list of tags directly.

If you change the location of the SDK (e.g. take it to another computer on an USB), re-run the `./emsdk activate latest` and `source ./emsdk_env.sh` commands.

Emsdk install targets

In the description above we asked the emsdk to install and activate `latest`, which is the latest tagged release. That is often what you want.

You can also install a specific version by specifying it, for example,

```
./emsdk install 1.38.45
```

Note

When installing old versions from before the build infrastructure rewrite (anything before `1.38.33`), you need to write something like `./emsdk install sdk-1.38.20-64bit` (add `sdk-` and `-64bit`) as that was the naming convention at the time.

There are also "tip-of-tree builds", which are the very latest code that passes integration tests on Chromium CI. This is updated much more frequently than tagged releases, but may be less stable (we tag releases manually using a more careful procedure). Tip-of-tree builds may be useful for continuous integration that uses the emsdk (as Emscripten's GitHub CI does), and you may want to use it in your own CI as well, so that if you find a regression on your project you can report it and prevent it from reaching a tagged release. Tip-of-builds may also be useful if you want to test a feature that just landed but didn't reach a release yet. To use a tip-of-tree build, use the `tot` target, and note that you must specify the backend explicitly,

```
# Get a tip-of-tree
./emsdk install tot
```

(In the above examples we installed the various targets; remember to also `activate` them as in the full example from earlier.)

Platform-specific notes

Windows

1. Install Python 3.6 or newer (older versions may not work due to a GitHub change with SSL).

Note

Instead of running emscripten on Windows directly, you can use the Windows Subsystem for Linux to run it in a Linux environment.

macOS

Note

Emscripten requires macOS 10.14 Mojave or above.

If you use the Emscripten SDK it includes a bundled version of Python 3. Otherwise you will need to manually install and use Python 3.6 or newer.

These instructions explain how to install **all** the required tools. You can test whether some of these are already installed on the platform and skip those steps.

1. Install the *Xcode Command Line Tools*. These are a precondition for *git*.

   Install Xcode from the macOS App Store.

   In **Xcode | Preferences | Downloads**, install *Command Line Tools*.

1. Install *git*:

   Make sure the OS allows installing git.

   Install Xcode and the Xcode Command Line Tools (should already have been done). This will provide *git* to the system PATH (see this stackoverflow post).

   Download and install git directly from http://git-scm.com/.

1. Install *cmake* if you do not have it yet:

   Download and install latest CMake from Kitware CMake downloads.

Linux

Note

Emsdk does not install any tools to the system, or otherwise interact with Linux package managers. All file changes are done inside the **emsdk/** directory.

Python is not provided by *emsdk*. The user is expected to install this beforehand with the *system package manager*:

```
# Install Python
sudo apt-get install python3

# Install CMake (optional, only needed for tests and building Binaryen or LLVM)
sudo apt-get install cmake
```

Note

If you want to use your system's Node.js instead of the emsdk's, it may be `node` instead of `nodejs`, and you can adjust the `NODE_JS` attribute of your `.emscripten` file to point to it.

Git is not installed automatically. Git is only needed if you want to use tools from a development branch.

```
# Install git
sudo apt-get install git
```

Verifying the installation

The easiest way to verify the installation is to compile some code using Emscripten.

You can jump ahead to the Emscripten Tutorial, but if you have any problems building you should run through the basic tests and troubleshooting instructions in Verifying the Emscripten Development Environment.

Updating the SDK

Tip

You only need to install the SDK once! After that you can update to the latest SDK at any time using Emscripten SDK (emsdk).

Type the following in a command prompt

```
# Fetch the latest registry of available tools.
./emsdk update

# Download and install the latest SDK tools.
./emsdk install latest

# Set up the compiler configuration to point to the "latest" SDK.
./emsdk activate latest

# Activate PATH and other environment variables in the current terminal
source ./emsdk_env.sh
```

The package manager can do many other maintenance tasks ranging from fetching specific old versions of the SDK through to using the versions of the tools on GitHub (or even your own fork). Check out all the possibilities in the "How to" guides.

Uninstalling the Emscripten SDK

If you want to remove the whole SDK, just delete the directory containing the SDK.

It is also possible to remove specific tools in the SDK using emsdk.

Using the Docker image

The entire Emscripten SDK is also available in the form of a docker image. For example:

```
docker run --rm -v $(pwd):/src -u $(id -u):$(id -g) \
  emscripten/emsdk emcc helloworld.cpp -o helloworld.js
```

See the Docker Hub page for more details and examples.

Installation using unofficial packages

Note

The *emsdk* is the only officially supported way to use Emscripten that is supported by the Emscripten project, and the only one that we constantly test (emsdk CI, Emscripten GitHub CI, Chromium CI).

While we don't officially support other ways of getting Emscripten, we definitely appreciate the efforts by third parties to package Emscripten for users' convenience, and we'd like to help out, please get in touch if you are such a packager!

The following is a partial list of such unofficial emscripten packages:

Windows

package info: *emscripten* in chocolatey

maintainer: @aminya

Homebrew

package info: https://formulae.brew.sh/formula/emscripten

maintainer: @chenrui333

Arch Linux

package info: https://archlinux.org/packages/extra/x86_64/emscripten

maintainer: Sven-Hendrik Haase <svenstaro@archlinux.org>