Halide setup - Linux

1. Dependencies

```
sudo apt-get update
sudo apt install build-essential g++ libjpeg-dev libpng-dev libz-dev
```

After running the llvm installation script, write down the version installed
Proceed by following either 2.a or 2.b

2.a Building from scratch

Clone Halide repo inside Home directory

```
git clone https://github.com/halide/Halide.git Halide-repo
```

Build

```
sudo bash -c "$(wget -O - https://apt.llvm.org/llvm.sh)"
mkdir build && cd build && make distrib LLVM_CONFIG=llvm-config-18 -f ../Halide-repo/Makefile
cd ..
```

Finishing

```
mv build/distrib ~/Halide
rm -rf build Halide-repo
```

2.b Getting binaries

Download the latest binary
https://github.com/halide/Halide/releases/

Extract the Halide root to the Home directory and change its name from Halide-1x.x.x-x86-64-linux to just Halide (Make sure "Halide" has subfolders "include", "lib", etc.)

If the folder tools is not inside Halide, copy it from Halide/share/Halide/tools to Halide/tools

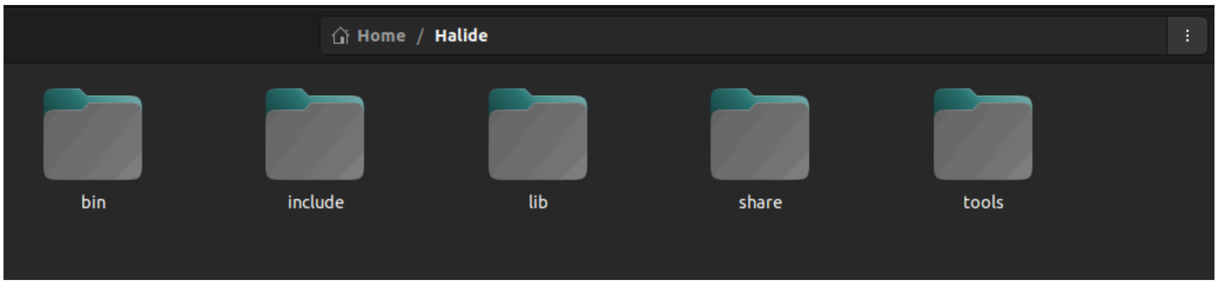
3. Final setups

Configure environment variables

```
echo 'export HALIDE_ROOT=$HOME/Halide' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HALIDE_ROOT/lib' >> ~/.bashrc
```

Update

```
source ~/.bashrc
```



Final folder structure should look like this.

Halide setup - Linux

- BUILDING

Besides C++ compilation flags, you have to add:

1. Includes directories (-I capital i, include):
    a. $HALIDE_ROOT include directory: -I $HALIDE_ROOT/include
    b. $HALIDE_ROOT/tools if you use any header file in it: -I $HALIDE_ROOT/tools
    c. The directory of the c_header generated; if "bin" is the directory ("-o" argument when running the generator binary): -I bin
    d. libpng include directory if you use it: `libpng-config --cflags`
2. Linker flags:
    a. Halide flags for Linux: -lHalide -lz -ldl -lpthread
    b. Halide flags for other OS: -lHalide -lz
    c. Halide flags for libpng and libjpeg if you use them: -ljpeg `libpng-config --ldflags`
3. Libraries directories (-L, capital l, libraries):
    a. $HALIDE_ROOT library directory when building with -lHalide: -L $HALIDE_ROOT/lib

NOTES

1. You only need to compile the C++ code when using realize, but the binary compiles Halide code at run time (JIT compilation).
2. When using generators, compile_to_file, or compile_to_static_library, there are three steps:
    Step1. Compile C++ code with g++
    Step2. Compile Halide code and generate c_header(.h), and object(.o) or static_library(.a)
    Step3. Compile C++ code
3. When using generators, you must compile the Halide code with $HALIDE_ROOT/tools/GenGen.cpp
4. See the arguments to generator binary at https://halide-lang.org/tutorials/tutorial_lesson_15_generators_usage.html
5. You may need to add -D_GLIBCXX_USE_CXX11_ABI=0 when building using older Halide versions and newer GCC versions