Compile Met.3D from source code: Linux (Ubuntu/SuSE)

Recommended way for compiling Met.3D: conda environment

This page provides old installation guidelines to compile Met.3D from source on openSuSE and Ubuntu Linux systems using a development environment installed from the system repositories. We have switched entirely to using conda environments for developing and building Met.3D, and we recommend using conda instead. This page is kept for reference.

Note

The current versions of **Met.3D 1.7 and higher** require *libproj* version 8 or higher, which may need to be installed manually. We recommend using conda.

The following installation guidelines have been tested with the older **Met.3D 1.6** under **openSUSE 15.0/15.1** and **Ubuntu 18.04/20.04 LTS**. Still, try using conda if possible.

I know what I am doing!

I know what I am doing!

If you are experienced with compiling software under Linux, have a look at the listing at the bottom of this page - it lists all commands necessary to compile Met.3D on a clean Ubuntu system. Other systems will be similar.

This page provides installation guidelines for installing Met.3D on openSUSE and Ubuntu Linux systems using the cmake build system. Met.3D requires a number of libraries to be installed and a few external data packages to be downloaded. Most of these packages can be installed via the respective system package managers (YaST or aptitude), however, a few have to be downloaded and compiled manually.

Note

If you have Docker available on your system, you can test the Met.3D compilation using a Docker image. An example using an Ubuntu image is provided at the end of this page.

System requirements: You will need an OpenGL 4.3 capable graphics card and an appropriate Linux driver to run Met.3D. The driver will most likely be a proprietary driver (Nvidia or AMD); open-source drivers for Linux currently do not provide the required capabilities. Before you continue with the installation, make sure that graphics card and driver are installed. If everything is installed correctly, the `glxinfo` command should output something similar to (the important thing is the `OpenGL core profile` version > 4.3):

```
# glxinfo | grep OpenGL

OpenGL vendor string: NVIDIA Corporation
OpenGL renderer string: GeForce GTX TITAN/PCIe/SSE2
OpenGL core profile version string: 4.4.0 NVIDIA 340.96
OpenGL core profile shading language version string: 4.40 NVIDIA via Cg compiler
```

Install available dependencies via your package manager

openSUSE

For openSUSE 15.0, the following additional repository is required to obtain the ECMWF eccodes library (similar for other openSUSE versions):

- http://download.opensuse.org/repositories/home:/SStepke/openSUSE_Leap_15.0/

The repository can be added in the “Software Repositories” windows in YaST. Afterwards, the following packages can be installed in the “Software Management” window (or on the command line).

- libqt5-qtbase-devel (or, for Met.3D versions < 1.3: libqt4 and libqt4-devel) - and further packages for Qt5 development
- liblog4cplus-### and log4cplus-devel
- gdal, libgdal20 and gdal-devel
- netcdf, netcdf-devel, libnetcdf_c++4-devel, libnetcdf_c++4-1
- hdf5, libhdf5 and hdf5-devel
- glew and glew-devel
- libfreetype6 and freetype2-devel
- eccodes and eccodes-devel (or, for Met.3D versions < 1.3: grib_api and grib_api-devel)
- libGLU1

- gsl and gsl-devel
- libproj##
- software development basics (gcc, gcc-fortran, git, wget, zip)

Ubuntu

For Ubuntu 18.04/20.04, the following packages need to be installed via aptitude:

- qt5-default
- liblog4cplus-dev
- libgdal-dev
- libnetcdf-dev
- libnetcdf-c++4-dev
- libeccodes-dev
- libfreetype6-dev
- libgsl-dev
- libglew-dev
- libproj-dev
- software development basics (packages build-essential, gfortran, git, wget, zip)

Install remaining required libraries from their sources

The dependencies `glfx` and `qcustomplot` are for both systems not available (at least not in the requried versions). They need to be complied manually.

> **Note**
>
> Install both libraries to places where cmake for Met.3D can find them (`/your/target/dir` in the commands listed below). If you are unsure, use a subdirectory of the `met-3d-base` directory introduced in the next section, e.g. `~\met.3d-base\local`.

glfx

Get the glfx sources from https://code.google.com/p/glfx/ or https://github.com/maizensh/glfx.git

```
cd glfx
cmake -DCMAKE_INSTALL_PREFIX:PATH=/your/target/dir CMakeLists.txt
make -j 12
make install
```

To make it easier for cmake for Met.3D to automatically find the libraries, choose one directory from `cmake/common_settings.cmake` as /your/target/dir.

qcustomplot

qcustomplot is required in a version >= 2.0. Get the qcustomplot sources from http://www.qcustomplot.com/

You will need the archives `QCustomPlot.tar.gz` and `QCustomPlot-sharedlib.tar.gz` for version >= 2.0.

Extract the `QCustomPlot.tar.gz` archive (the `/qcustomplot` directory) and put the contents of `QCustomPlot-sharedlib.tar.gz` **inside** the `/qcustomplot` directory. Go to

```
qcustomplot/qcustomplot-sharedlib/sharedlib-compilation
```

and run:

```
qmake # (on openSUSE, this may be qmake-qt5)
make
```

Next, copy the resulting libraries and the qcustomplot.h header to directories where cmake for Met.3D can find them automatically (look at `cmake/common_settings.cmake`). These files are required:

```
./include:
qcustomplot.h

./lib or ./lib64:
libqcustomplotd.so -> libqcustomplotd.so.2.0.0*
libqcustomplotd.so.2 -> libqcustomplotd.so.2.0.0*
libqcustomplotd.so.2.0 -> libqcustomplotd.so.2.0.0*
libqcustomplotd.so.2.0.0*
libqcustomplot.so -> libqcustomplot.so.2.0.0*
libqcustomplot.so.2 -> libqcustomplot.so.2.0.0*
libqcustomplot.so.2.0 -> libqcustomplot.so.2.0.0*
libqcustomplot.so.2.0.0*
```

Alternatively, the sources are available from the git repository at https://gitlab.com/DerManu/QCustomPlot.git If you fetch the code from the repository, you'll also need to run `run-amalgamate.sh`. See the Docker Ubuntu example at the end of this page.

Download source and data packages

We recommend to place the following packages along with the Met.3D sources into a specific directory structure.

Create a base directory `met.3d-base` and a subdirectory `third-party`:

```
met.3d-base/
    third-party/
```

Change into `third-party` to execute the following commands.

1) qtpropertybrowser

Met.3D requires the qtpropertybrowser framework from the "qt-solutions" repository. The qtpropertybrowser sources are directly compiled into the Met.3D executable and hence do not have to be build beforehand. They can be downloaded with git:

```
# [in met.3d-base/third-party]
git clone https://github.com/qtproject/qt-solutions.git
```

2) Fonts

Met.3D requires a TrueType font file. We recommend the "FreeSans" font from the GNU FreeFont package. It can be downloaded from http://ftp.gnu.org/gnu/freefont/. At the time of writing, the most recent version is 20120503:

```
# [in met.3d-base/third-party]
wget http://ftp.gnu.org/gnu/freefont/freefont-ttf-20120503.zip
unzip freefont-ttf-20120503.zip
```

3) Vector and raster map, coastline and country borderline data

Met.3D requires a base map image in GeoTIFF format, as well as coastline and country borderline vector data in shapefile format. we recommend to use the free data from http://www.naturalearthdata.com. The medium resolution files (50m) work fine (they require roughly 300 MB of disk space).

For coastline data, we use the "Coastline" dataset (http://www.naturalearthdata.com/downloads/50m-physical-vectors/):

```
# [in met.3d-base/third-party]
mkdir naturalearth
cd naturalearth
wget http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/physical/ne_50m_coastline.zip
unzip ne_50m_coastline.zip
```

For country boundaries, we use the "Admin 0 – Boundary Lines" dataset (http://www.naturalearthdata.com/downloads/50m-cultural-vectors/):

```
# [in met.3d-base/third-party/naturalearth]
wget http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/cultural
/ne_50m_admin_0_boundary_lines_land.zip
unzip ne_50m_admin_0_boundary_lines_land.zip
```

For the raster basemap, we use the “Cross Blended Hypso with Shaded Relief and Water” dataset (http://www.naturalearthdata.com/downloads/50m-raster-data/50m-cross-blend-hypso/):

```
# [in met.3d-base/third-party/naturalearth]
wget http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/raster/HYP_50M_SR_W.zip
unzip HYP_50M_SR_W.zip
```

You should now have the following directory structure (`...` denotes other files):

```
met.3d-base/
    third-party/
        qt-solutions/
            qtpropertybrowser/
                *
            ...
        freefont-20120503/
            FreeSans.ttf
            ...
        naturalearth/
            HYP_50M_SR_W/
                HYP_50M_SR_W.tif
                ...
            ne_50m_coastline.shp
            ne_50m_admin_0_boundary_lines_land.shp
            ...
```

Checkout Met.3D from the GIT repository

The latest version of Met.3D can be checked out from https://gitlab.com/wxmetvis/met.3d/

Place the repository into the `met.3d-base` base directory:

```
# [in met.3d-base]
git clone https://gitlab.com/wxmetvis/met.3d.git
```

This will checkout the latest development version. Alternatively, checkout the latest “stable” version by selecting the latest tag:

```
# [in met.3d-base]
cd met.3d
git tag -l
git checkout tags/<latest tag>
```

For example, if `git tag -l` returns a list in which *1.3.0* is the latest tag, checkout this version by entering `git checkout tags/1.3.0`.

Configure cmake for Met.3D

We provide cmake scripts for Makefile creation and compilation of Met.3D. You can either build Met.3D from the command line as described below, or use a cmake GUI (e.g., `cmake-curses-gui`, `cmake-gui`) to configure cmake. Alternatively, start the build process within C++ IDEs like QtCreator, CLion, or Visual Studio Code (these typically provide functionality to open `CMakeLists.txt` and to run the build process).

From the command line:

First, in `met.3d-base/`, create a subdirectory `build` into which the executable can be built and change into this directory:

```
# [in met.3d-base]
mkdir build
cd build
```

Create a `Makefile` by:

```
# [in met.3d-base/build]
cmake -DCMAKE_BUILD_TYPE=RELEASE ../met.3d
```

Met.3D can also be built in *debug mode*; change `-DCMAKE_BUILD_TYPE=RELEASE` to `-DCMAKE_BUILD_TYPE=DEBUG` to achieve this.

If some libraries are not located within the default header/library folders (given in common_settings.cmake), it is likely that you have to manually set the include directory and used libraries for a certain package. For example, if cmake could not find the include directory of GDAL, it will output something like `missing GDAL_INCLUDE_DIR`. In that case, add `-DGDAL_INCLUDE_DIR=/real/path/to/gdal/includes` to the command and run cmake again. Or use the GUI to set the missing directories and libraries and restart the configuring and generation process.

Compile Met.3D

After cmake has created the `Makefile`, run make (the "-j 12" option for make starts 12 parallel compilation threads, modify this number to match the number of CPU cores in your system).

```
# [in met.3d-base/build]
make -j 12
```

Compilation may take a few minutes. If no errors are reported, an executable named `Met3D` should be created in the build directory.

Start Met.3D

Before Met.3D can be started, two environment variables `MET3D_HOME` and `MET3D_BASE` need to be set. `MET3D_HOME` points to the Met.3D source directory (at least the subdirectories `/src/glsl` and `/config` need to be available as these contain code loaded at runtime):

```
export MET3D_HOME=/your/path/to/met.3d-base/met.3d
```

The additional environment variable `MET3D_BASE` is used in the default configuration files to refer to the paths with third-party data (see `/config/default_frontend.cfg.template`; feel free to change this if you like):

```
export MET3D_BASE=/your/path/to/met.3d-base
```

To start Met.3D, simply type:

```
# [e.g. in met.3d-base/build]
./Met3D
```


Note

On first start-up, you will see an empty window. Please follow the user guide to learn how to create visualizations.

Test compilation in a Docker container

In case you have Docker available on your system, you can test the compilation of Met.3D in a container. The following commands start an Ubuntu 18.04 container, install all dependencies and compile Met.3D. You won't be able to start Met.3D from the container, but the commands may be useful for tests or to install on your actual system.

To download and start the container:

```
docker info # make sure Docker is running
docker pull ubuntu:latest # download Ubuntu image
docker run -t -i ubuntu bash # start container
```

Within the container, to set up the system and compile Met.3D:

```
# update repositories and upgrade current system
apt update
apt upgrade

# install gcc compiler suite, see https://help.ubuntu.com/community/InstallingCompilers
apt install build-essential

# install required development tools
apt install cmake git wget zip gfortran

# section A), install the required dependencies (to list dependencies of a package use "apt depends <name>")
# "-dev" packages also install the corresponding libraries
apt install qt5-default liblog4cplus-dev libgdal-dev libnetcdf-dev libnetcdf-c++4-dev libeccodes-dev
libfreetype6-dev libgsl-dev libglew-dev libproj-dev

# section B), create "met.3d-base" directory and install the two remaining libraries "glfx" and "qcustomplot"
cd ~
mkdir met.3d-base && cd met.3d-base
mkdir local
mkdir third-party && cd third-party

git clone https://github.com/maizensh/glfx.git
cd glfx
cmake -DCMAKE_INSTALL_PREFIX:PATH=~/met.3d-base/local CMakeLists.txt
make -j 12
make install

cd ~/met.3d-base/third-party
git clone https://gitlab.com/DerManu/QCustomPlot.git
cd QCustomPlot
./run-amalgamate.sh
cp qcustomplot.h ~/met.3d-base/local/include/
cd sharedlib/sharedlib-compilation/
qmake
make -j 12
cp libqcustomplot* ~/met.3d-base/local/lib/

# section C), download remaining third-party dependencies
cd ~/met.3d-base/third-party
git clone https://github.com/qtproject/qt-solutions.git

wget http://ftp.gnu.org/gnu/freefont/freefont-ttf-20120503.zip
unzip freefont-ttf-20120503.zip

mkdir naturalearth
cd naturalearth wget https://naciscdn.org/naturalearth/50m/physical/ne_50m_coastline.zip
unzip ne_50m_coastline.zip
wget https://naciscdn.org/naturalearth/50m/cultural/ne_50m_admin_0_boundary_lines_land.zip
unzip ne_50m_admin_0_boundary_lines_land.zip
wget https://naciscdn.org/naturalearth/50m/raster/HYP_50M_SR_W.zip
unzip HYP_50M_SR_W.zip

# sections D)-F), checkout and compile Met.3D
cd ~/met.3d-base/
git clone https://gitlab.com/wxmetvis/met.3d.git
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=RELEASE ../met.3d
make -j 12

# now a binary "Met3D" should have been created
```