
CMake Python Distributions Documentation

Release 3.14.4.post2.dev0+gaa9a0fd

Jean-Christophe Fillion-Robin

May 28, 2019

1	Installation	3
1.1	Install package with pip	3
1.2	Install from source	3
1.3	Dependencies	3
2	Usage	5
3	Building the CMake Python wheel	7
3.1	Overview	7
3.2	Prerequisites	7
3.3	Quick start	7
3.4	Source distribution (sdist)	7
3.5	Binary distribution (build, bdist, bdist_wheel)	8
3.6	Default value for BUILD_CMAKE_FROM_SOURCE	8
3.7	Controlling verbosity	8
3.8	Optimizations	9
4	Understanding the Build-system	11
4.1	CMakeLists.txt	11
5	Contributing	15
5.1	Types of Contributions	15
5.2	Get Started	16
5.3	Pull Request Guidelines	17
5.4	Tips	17
6	Credits	19
7	History	21
8	Updating the CMake version	23
9	Making a release	25
9.1	Prerequisites	25
9.2	Documentation conventions	25
9.3	Setting up environment	25
9.4	PyPI: Step-by-step	26

10 Indices and tables

29

11 Resources

31

CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice.

The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as **ITK** and **VTK**.

The CMake python wheels provide **CMake 3.14.4**.

1.1 Install package with pip

To install with pip:

```
$ pip install cmake
```

1.2 Install from source

See *Building the CMake Python wheel*

1.3 Dependencies

1.3.1 Python Packages

The development dependencies (for testing and coverage) are:

```
codecov>=2.0.5
coverage>=4.2
flake8>=3.0.4
path.py>=11.5.0
pytest>=3.0.3
pytest-cov>=2.4.0
pytest-runner>=2.9
pytest-virtualenv>=1.2.5
scikit-build>=0.10.0
setuptools>=28.0.0
twine
virtualenv>=15.0.3
```

(continues on next page)

(continued from previous page)

```
wheel  
wheeltools
```


CHAPTER 2

Usage

After *installing* the package using *pip*, the executables `cmake`, `cpack` and `ctest` will be available in the `PATH` and can be used to configure and build any projects.

Building the CMake Python wheel

3.1 Overview

This project has been designed to work with `scikit-build`.

It provides a `setup.py` and allows to create both source and binary distributions of CMake.

This is done ensuring source files and build artifacts are copied and/or generated in expected locations.

3.2 Prerequisites

In addition of `Git`, `Python` and `CMake`, building the wheel with `BUILD_CMAKE_FROM_SOURCE` set to `ON` also requires a `C++ Compiler`.

3.3 Quick start

Build the CMake Python wheel with the following command:

```
mkvirtualenv build-cmake
pip install -r requirements-dev.txt
python setup.py bdist_wheel
```

3.4 Source distribution (sdist)

CMake sources will always be downloaded in the `<ROOT>/src` directory.

This will ensure that the rules specified in `<ROOT>/MANIFEST.in` can successfully glob the source files.

The source distribution is generated using the following command:

```
python setup.py sdist
```

3.5 Binary distribution (build, bdist, bdist_wheel)

The project has two mode of operations:

1. build CMake from source (BUILD_CMAKE_FROM_SOURCE set to ON)
2. download CMake binaries (BUILD_CMAKE_FROM_SOURCE set to OFF)

The binary distribution is generated using the following command:

```
python setup.py bdist_wheel
```

Changing the default mode is achieved by explicitly passing the option to CMake:

```
python setup.py bdist_wheel -- -DBUILD_CMAKE_FROM_SOURCE:BOOL=ON
```

3.6 Default value for BUILD_CMAKE_FROM_SOURCE

Depending on the platform, option BUILD_CMAKE_FROM_SOURCE has different default:

- Linux: ON
- MacOSX: OFF
- Windows: OFF

3.7 Controlling verbosity

3.7.1 configure and build output

By default, the output associated to the configure and build steps of the *CMakeProject-build* external project are logged into files. This can be changed by setting the BUILD_VERBOSE option:

```
python setup.py bdist_wheel -- -DBUILD_VERBOSE:BOOL=1
```

3.7.2 list of files copied into the distributions

By default, the complete list of files copied into the distributions are reported. This can be changed passing the `--hide-listing` option:

```
python setup.py --hide-listing sdist
python setup.py --hide-listing bdist_wheel
```

3.8 Optimizations

On a given platform, when building different “flavor” of CMake python wheels (one for each <python tag>-<abi> tag), the whole process can be made faster in two ways.

3.8.1 Caching downloads

To avoid the re-download of CMake sources and/or binary packages, passing the option `-DCMakePythonDistributions_ARCHIVE_DOWNLOAD_DIR:PATH=/path/to/cache` enables successive build to re-use existing archives instead of re-downloading them.

3.8.2 Re-using build tree

And finally, on a given platform, to avoid rebuilding CMake, the idea is to first create a standalone build of the CMake project and then building the wheel using it.

Step 1: Standalone build:

```
mkdir -p standalone-build && cd $_
cmake -DCMakePythonDistributions_ARCHIVE_DOWNLOAD_DIR:PATH=/path/to/cache -G Ninja ../
```

Step 2: Faster build reusing download and build directories:

```
python setup.py bdist_wheel -- \
  -DCMakePythonDistributions_ARCHIVE_DOWNLOAD_DIR:PATH=/path/to/cache \
  -DCMakeProject_BINARY_DIR:PATH=/path/to/standalone-build
```

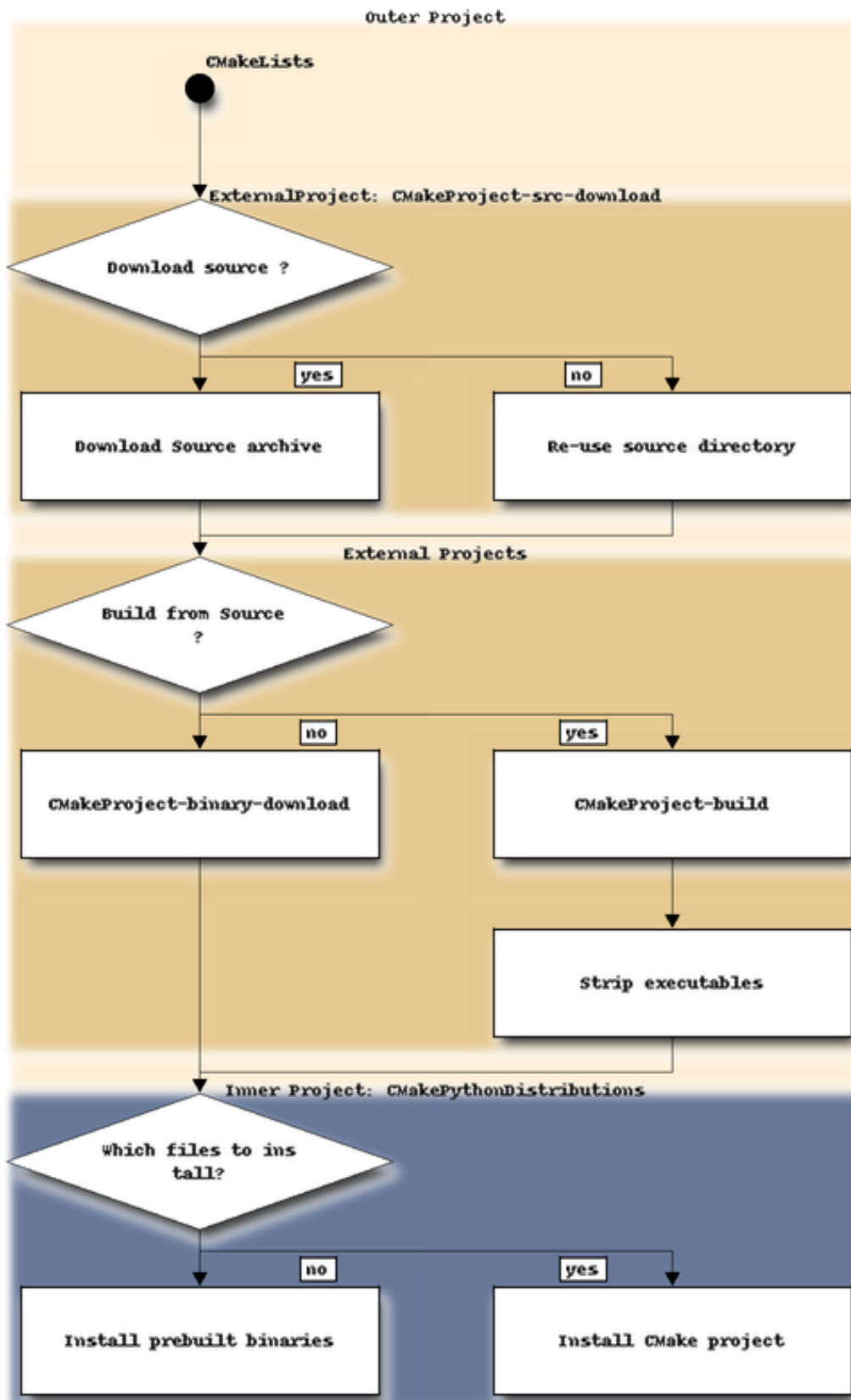
Understanding the Build-system

4.1 CMakeLists.txt

The build system is described by the `CMakeLists.txt` and is composed of few projects each responsible for a specific task. Once configured, the *Outer Project* is responsible for driving the overall build composed of multiple project called *external project*. Here is the list of *external project*:

- `CMakeProject-src-download`
- `CMakeProject-binary-download`
- `CMakeProject-build`
- `CMakePythonDistributions`: This corresponds to the *Inner Project* represented below.

The flow chart represented below illustrates which external projects are included based on the configure options and describes the role of each one:



Name	Description
CMakeLists	See CMakeLists.txt
Download source ?	If option <code>CMakeProject_SOURCE_DIR</code> is set, skip source download.
Download Source archive	External project downloading archives from https://cmake.org/files/ .
Re-use source directory	Empty external project.
Build from Source ?	Answer based on option <code>BUILD_CMAKE_FROM_SOURCE</code>
CMakeProject-binary-download	External project downloading pre-built binary archives from https://cmake.org/files/ .
CMakeProject-build	External project building CMake from source.
Strip executables	If possible, reduce wheel size stripping <code>cmake</code> , <code>cpack</code> and <code>ctest</code> executables
Which files to install?	Answer based on option <code>BUILD_CMAKE_FROM_SOURCE</code>
Install prebuilt binaries	Recursively glob all files and explicitly add install rules.
Install CMake project	Achieved by including <code>\${CMakeProject_BINARY_DIR}/cmake_install.cmake</code> .

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Types of Contributions

You can contribute in many ways:

5.1.1 Report Bugs

Report bugs at <https://github.com/scikit-build/cmake-python-distributions/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

5.1.4 Write Documentation

The `cmake-python-distributions` project could always use more documentation. We welcome help with the official `cmake-python-distributions` docs, in docstrings, or even on blog posts and articles for the web.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/scikit-build/cmake-python-distributions/issues>.

If you are proposing a new feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started

Ready to contribute? Here's how to set up `cmake-python-distributions` for local development.

1. Fork the `cmake-python-distributions` repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/cmake-python-distributions.git
```

3. Install your local copy into a virtualenv. Assuming you have `virtualenvwrapper` installed (*pip install virtualenvwrapper*), this is how you set up your cloned fork for local development:

```
$ mkvirtualenv cmake-python-distributions
$ cd cmake-python-distributions/
$ pip install -r requirements-dev.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass `flake8` and the tests:

```
$ flake8
$ python setup.py bdist_wheel
$ python setup.py test
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in *README.rst*.
3. The pull request should work for Python 2.7, and 3.3, 3.4, 3.5. Check [AppVeyor](#), [CircleCI](#) and [TravisCi](#) and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests/test_cmake.py
```


CHAPTER 6

Credits

Please see the GitHub project page at <https://github.com/scikit-build/cmake-python-distributions/graphs/contributors>

CHAPTER 7

History

`cmake-python-distributions` was initially developed in September 2016 by Jean-Christophe Fillion-Robin to facilitate the distribution of project using `scikit-build` and depending on CMake.

Updating the CMake version

A developer should use the following steps to update the version $X.Y.Z$ of CMake associated with the current CMake python distributions.

Available CMake archives can be found at <https://cmake.org/files>.

1. Install *requests*:

```
$ pip install requests
```

2. Execute *scripts/update_cmake_version.py* command line tool with the desired $X.Y.Z$ CMake version available for download. For example:

```
$ release=3.14.4
$ python scripts/update_cmake_version.py ${release}
Collecting URLs and SHA256s from 'https://api.github.com/repos/Kitware/CMake/
↳releases/tags/v3.14.4'
[...]
Collecting URLs and SHA256s from 'https://api.github.com/repos/Kitware/CMake/
↳releases/tags/v3.14.4' - done
Updating 'CMakeUrls.cmake' with CMake version 3.14.4
Updating 'CMakeUrls.cmake' with CMake version 3.14.4 - done
Updating docs/index.rst
Updating docs/index.rst - done
Updating README.rst
Updating README.rst - done
Updating tests/test_distribution.py
Updating tests/test_distribution.py - done
```

3. Create a topic named *update-to-cmake-X.Y.Z* and commit the changes. For example:

```
release=3.14.4
git checkout -b update-to-cmake-${release}
git add CMakeUrls.cmake docs/index.rst README.rst tests/test_distribution.py
git commit -m "Update to CMake ${release}"
```

4. Push the topic and create a *Pull Request*.

5. If all CI tests are passing, merge the topic and consider *making a new release*.

A core developer should use the following steps to create a release *X.Y.Z* of **cmake-python-distributions** on PyPI. This is usually done after *Updating the CMake version*.

9.1 Prerequisites

- All CI tests are passing on [AppVeyor](#), [CircleCI](#) and [Travis CI](#).
- You have a [GPG signing key](#).

9.2 Documentation conventions

The commands reported below should be evaluated in the same terminal session.

Commands to evaluate starts with a dollar sign. For example:

```
$ echo "Hello"  
Hello
```

means that `echo "Hello"` should be copied and evaluated in the terminal.

9.3 Setting up environment

1. First, [register for an account on PyPI](#).
2. If not already the case, ask to be added as a `Package Index Maintainer`.
3. Create a `~/.pypirc` file with your login credentials:

```
[distutils]
index-servers =
  pypi
  pypitest

[pypi]
username=<your-username>
password=<your-password>

[pypitest]
repository=https://test.pypi.org/legacy/
username=<your-username>
password=<your-password>
```

where `<your-username>` and `<your-password>` correspond to your PyPI account.

9.4 PyPI: Step-by-step

1. Make sure that all CI tests are passing on [AppVeyor](#), [CircleCI](#) and [Travis CI](#).
2. Download the latest sources

```
$ cd /tmp && \
  git clone git@github.com:scikit-build/cmake-python-distributions cmake-
  ↪python-distributions-release && \
  cd cmake-python-distributions-release
```

3. List all tags sorted by version

```
$ git fetch --tags && \
  git tag -l | sort -V
```

4. Choose the next release version number

```
$ release=X.Y.Z
```

Warning: To ensure the packages are uploaded on [PyPI](#), tags must match this regular expression:
`^[0-9]+(\.[0-9]+)*(\.post[0-9]+)?$`.

5. In `README.rst`, update [PyPI](#) download count after running [this big table query](#) and commit the changes.

```
$ git add README.rst && \
  git commit -m "README: Update download stats [ci skip]"
```

Note: To learn more about `pypi-stats`, see [How to get PyPI download statistics](#).

5. Tag the release

```
$ git tag --sign -m "cmake-python-distributions ${release}" ${release}_
  ↪origin/master
```

Warning: We recommend using a [GPG signing key](#) to sign the tag.

6. Publish the release tag

```
$ git push origin ${release}
```

Note: This will trigger builds on each CI services and automatically upload the wheels and source distribution on [PyPI](#).

7. Check the status of the builds on [AppVeyor](#), [CircleCI](#) and [Travis CI](#).

8. Once the builds are completed, check that the distributions are available on [PyPI](#).

9. Create a clean testing environment to test the installation

```
$ pushd $(mktemp -d) && \  
  mkvirtualenv cmake-${release}-install-test && \  
  pip install cmake && \  
  cmake --version
```

Note: If the `mkvirtualenv` command is not available, this means you do not have [virtualenvwrapper](#) installed, in that case, you could either install it or directly use [virtualenv](#) or [venv](#).

10. Cleanup

```
$ popd && \  
  deactivate && \  
  rm -rf dist/* && \  
  rmvirtualenv cmake-${release}-install-test
```


CHAPTER 10

Indices and tables

- `genindex`
- `modindex`
- `search`

CHAPTER 11

Resources

This project is maintained by Jean-Christophe Fillion-Robin from Kitware Inc. It is covered by the [Apache License, Version 2.0](#).

CMake is distributed under the OSI-approved BSD 3-clause License. For more information about CMake, visit <http://cmake.org>

- Documentation: <http://cmake-python-distributions.readthedocs.io/en/latest/>
- Source code: <https://github.com/scikit-build/cmake-python-distributions>
- Mailing list: <https://groups.google.com/forum/#!forum/scikit-build>