scikit-build Documentation

Release 0.2.0

scikit-build team

September 17, 2016

Contents

Contents:

CHAPTER 1

scikit-build

Improved build system generator for CPython C extensions.

Better support is available for additional compilers, build systems, cross compilation, and locating dependencies and determining their build requirements. The **scikit-build** package is fundamentally just glue between the *setuptools* Python module and CMake. Currently, the package is available to perform builds in a *setup.py* file. In the future, the project aims to be a build tool option in the currently developing pyproject.toml build system specification.

- Free software: MIT license
- Documentation: http://scikit-build.readthedocs.org
- Source code: https://github.com/scikit-build/scikit-build
- Mailing list: https://groups.google.com/forum/#!forum/scikit-build

CHAPTER 2

Installation

With pip:

```
$ pip install scikit-build
```

To install the latest from source, first obtain the source code:

```
$ git clone https://github.com/scikit-build/scikit-build
$ cd scikit-build
```

then install with:

```
$ pip install .
```

or:

```
$ pip install -e .
```

for development.

2.1 Dependencies

2.1.1 Python Packages

The project has a few common Python package dependencies. The runtime dependencies are:

```
wheel==0.29.0
setuptools==25.0.0
```

the build time dependencies (also required for development) are:

```
codecov==2.0.5
coverage==4.1
cython==0.24.1
flake8==3.0.0
nose==1.3.7
six==1.10.0
```

2.1.2 Compiler Toolchain

The same compiler toolchain used to build the CPython interpreter should also be available. For example, on *Ubuntu Linux*, this can be installed with:

```
$ sudo apt-get install build-essential
```

On *Mac OSX*, install XCode to build packages for the system Python.

On Windows, install the version of Visual Studio used to create the target version of CPython

2.1.3 CMake

Download standard CMake binaries for your platform or build from source with a C++ compiler if binaries are not available.

CHAPTER 3

Usage

To use scikit-build in a project:

```
# in your project's setup.py file, instead of from setuptools import setup
from skbuild import setup
```

TODO (scikit-build developer): need to provide small, self-contained setup function calls for (at least) 2 use cases: when a CMakeLists.txt file already exists, and when users want scikit-build to create a CMakeLists.txt file based on specifying some input files.

Alternatively, you can drive CMake more directly yourself with scikit-build:

```
# cmaker defines an object that provides convenient access to CMake configure & build functionality
from skbuild import cmaker
maker = cmaker.CMaker()
# based on CMakeLists.txt file in current directory, generates Makefiles, Visual Studio solutions, o
maker.configure()
# executes the build with the appropriate build tool and performs installation to local folders
maker.make()
```

CHAPTER 4

Contributing

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

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/scikit-build/scikit-build/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

The scikit-build project could always use more documentation, whether as part of the official scikit-build docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

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

If you are proposing a 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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *scikit-build* for local development.

1. Fork the *scikit-build* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/scikit-build.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv scikit-build
$ cd scikit-build/
$ python setup.py develop
```

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, including testing other Python versions with tox:

```
$ flake8 skbuild tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

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.

4.3 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, and for PyPy. Check https://travis-ci.org/scikit-build/scikit-build/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_skbuild
```

CHAPTER 5

Credits

Please see the GitHub project page at https://github.com/scikit-build/scikit-build/graphs/contributors

CHAPTER 6

History

PyCMake was created at SciPy 2014 in response to general difficulties building C++ and Fortran based Python extensions across platforms. It was renamed to “scikit-build” in 2016.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search