scikit-build-core:
A modern build-backend for CPython C/C++/Fortran/Cython extensions

Revolutionizing Python Extension Building

Speaker: Jean-Christophe Fillion-Robin @ Kitware
SciPy 2024

Who am I ?

- Distinguished Engineer @ Kitware in the North Carolina office, lead developer of 3D Slicer
- Maintainer of scikit-build, scikit-build-core, cmake and ninja python packages
- Maintainer of python-cmake-buildsystem
- Maintainer of dockcross

 jfillionr

 jcfr

 thehiddenbrain

#scikit-build

Goals of this talk

- Introduce scikit-build-core and its significance in Python packaging
- Explain how it simplifies building C/C++/Fortran/Cython extensions
- Demonstrate its key features and benefits
- Show real-world adoption and impact

Outline

- Evolution of Python packaging
- Importance of build systems
- Overview of scikit-build-core
- Example
- Review of innovative features
- Adoption & statistics
- Next steps

History of Packaging (1 / 2)

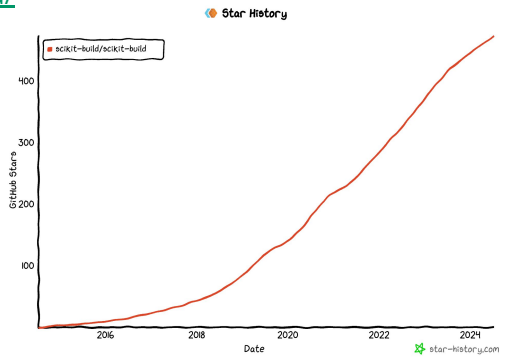
- Introduction of `distutils` (in Python 1.6 in 2000)
- Challenges: difficult distribution, large multi-module packages
- Rise of Python distributions (e.g., Enthought, Conda)

History of Packaging (2 / 2)

- Introduction of wheel format (2013)
- `setuptools` becomes de facto standard (2013)
- `pip` replaces `easy_install` (2014)

https://pypi.org/project/scikit-build/

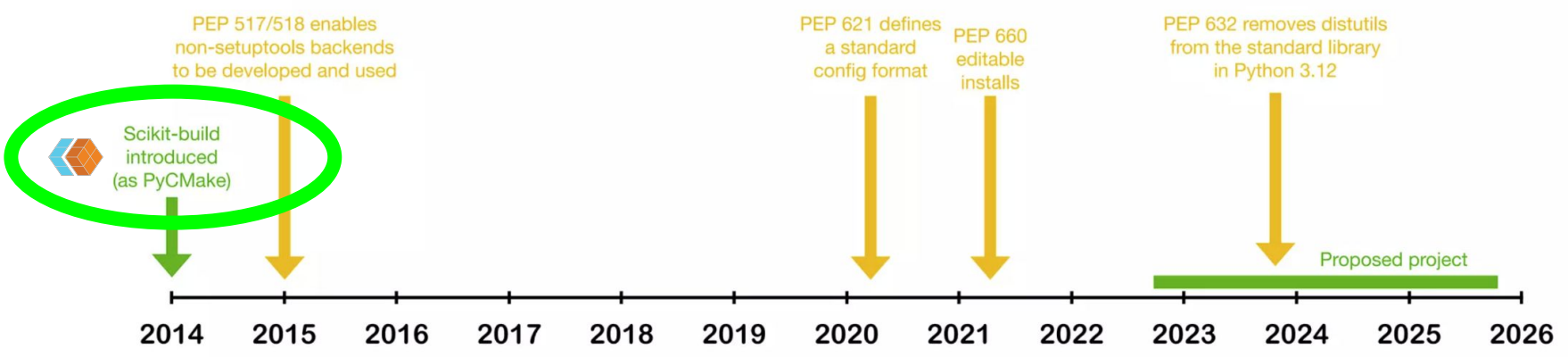
scikit-build (classic)



- Released as PyCMake at SciPy 2014
- Renamed to scikit-build at SciPy 2016
- ~800k downloads per month (as of July 11, 2024)
- Limitations: tied to setuptools, prone to breakage

History of Packaging: The PEP Revolution

- PEP 517: Build system abstraction (aka build backend)
- PEP 518: Build system requirements
- PEP 621: Metadata specification
- PEP 660: Editable installs
- See https://www.python.org/dev/peps/



Before PEP 517 (2017): setuptools/distutils

```
from setuptools import Extension, setup

setup(
    name="mylib",
    version="1.0.0",
    ext_modules=[
        Extension(
            name="mylib.foo",
            sources=["foo.c"],
        ),
    ]
)
```

Source: https://gatsiesheikar.files.wordpress.com/2013/03/headdesk.png

Simple👌, **unless** you need 💣:

- 3rd party dependencies
- Compiler specific flags (e.g C++ version)
- Parallel (file) compile
- Caching & smart recompile
- `bdist_wheel` customization (no public API)
- Fortran
- Other compilers
- IDE support
- Tooling integration (debugger, ...)
- Cross-compilation

Before PEP 517 (2017): Extending setuptools/distutils

Distutils has layered complexity

- Hard to debug
- Easy to break on update
- Hard to extend

Examples of setuptools-based builders

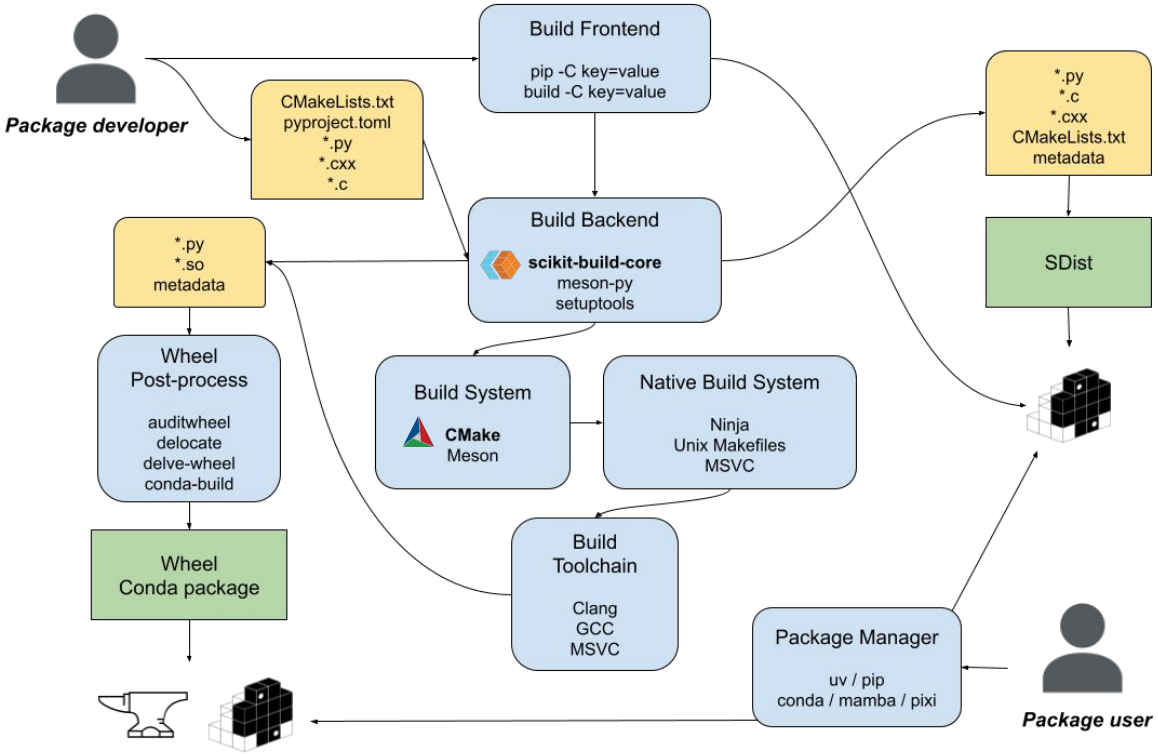
- **scikit-build (classic)**
- setuptools-rust
- pybind11's setup_helper
- cython (integrated)

Source: https://gatsiesheikar.files.wordpress.com/2013/03/headdesk.png

Introducing scikit-build-core

- Built on modern packaging standards (PEP 517/518/621/660/...)
- Free from `setuptools` legacy
- Enhanced features for cross-compilation and multi-platform support
- NSF-funded development OAC-2209877

Binary Python Package Generation Overview



The Simplest Example (1/3): `main.cpp`

```cpp
#include <pybind11/pybind11.h>

PYBIND11_MODULE(example, m) {
    m.def("square", [](double x) { return x*x; });
}
```

Source: https://github.com/scikit-build/scikit-build-sample-projects/tree/main/projects/core-pybind11-hello

The Simplest Example (2/3): `pyproject.toml`

```toml
[build-system]
requires = ["scikit-build-core", "pybind11"]
build-backend = "scikit_build_core.build"

[project]
name = "example"
version = "0.0.1"
```

The Simplest Example (3/3): CMakeLists.txt

```
cmake_minimum_required(VERSION 3.15...3.30)

project(example LANGUAGES CXX)

set(PYBIND11_NEWPYTHON ON)
find_package(pybind11 CONFIG REQUIRED)

pybind11_add_module(example example.cpp)
install(TARGETS example LIBRARY DESTINATION .)
```

Common Configuration Needs

```
[tool.scikit-build]
minimum-version = "0.10"
build.verbose = true
logging.level = "INFO"
wheel.expand-macos-universal-tags = true
```

Innovative Features: Dynamic CMake/Ninja Requirement

- Checks for system CMake/Ninja before requiring wheels
- Improves compatibility with systems like WebAssembly, BSD, etc.
- PEP 517 has an API for a build tool to declare its dependencies.

https://pypi.org/project/cmake/

https://pypi.org/project/ninja

Innovative Features: Integration with External Packages

- `site-packages` added to CMake search path
- `cmake.prefix` entry point for adding prefix dirs
- `cmake.module` entry point for adding CMake helper files

More details at
https://scikit-build-core.readthedocs.io/en/latest/cmakelists.html#finding-other-packages

Innovative Features: Dual Editable Modes

- Default mode:
  - Custom finder combining source and installed files
  - Automatic rebuilds on source changes
- In-place mode:
  - Similar to `setuptools build_ext --inplace`

More details at
https://scikit-build-core.readthedocs.io/en/latest/configuration.html#editable-installs

Innovative Features: Dynamic Metadata

- Custom metadata plugins system
- Three built-in plugins
  - Regex
  - `setuptools_scm` wrapper
  - `hatch-fancy-pypi-readme` wrapper
- Support for "in-tree" plugins written inside a specific package
- Example:

```
name = "mypackage"
dynamic = ["version"]

[tool.scikit-build.metadata.version]
provider = "scikit_build_core.metadata.regex"
input = "src/mypackage/__init__.py"
```

More details at
https://scikit-build-core.readthedocs.io/en/latest/configuration.html#dynamic-metadata

Innovative Features: File Generation

```
[[tool.scikit-build.generate]]
path = "package/_version.py"
template = '''
version = "${version}"
'''
```

More details at
https://scikit-build-core.readthedocs.io/en/latest/configuration.html#writing-metadata

Innovative Features: Overrides

```
[[tool.scikit-build.overrides]]
if.platform-system = "darwin"
cmake.version = ">=3.18"
```

More details at
https://scikit-build-core.readthedocs.io/en/latest/configuration.html#overrides

Scikit-build-core's Design: Configuration System

- Over 40 options
- Provide a JSON Schema for the TOML configuration
- Supported via `pyproject.toml`, config-settings (`-C` in `build` or `pip`) or env. variable starting with `SKBUILD`
- Streamlined maintenance with dataclass-based configuration

Scikit-build-core's Design: File API

- Full implementation of CMake's File API reader
- Uses dataclasses following official schema
- Enables plugins to read build process information

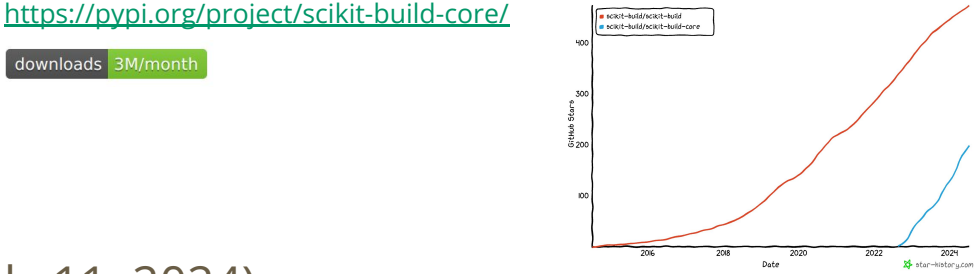
Scikit-build-core's Design: Plugins for Other Systems

- Setuptools plugin
- Hatchling plugin
- Designed to support custom plugins wrapping CMake builds

Adoption: Overview

- Simplification: **800+ lines of `setup.py` to 20 lines of CMake**
- New platform support (e.g., Windows, PyPy)
- Faster multithreaded compile times

https://pypi.org/project/scikit-build-core/



Adoption: Statistics

- **2.9M** monthly downloads (as of July 11, 2024)
- Used by a **dozen** of top 8000 PyPI packages (e.g `awkward-cpp`, `pyzmq`, `lightgbm`, `cmake`, `phik`, `clang-format`)
- **255 packages** using scikit-build-core (as of June 25, 2024, +82 from last year)
- **764 non-fork mentions** of `scikit_build_core` in `pyproject.toml` (as of July 3, 2024, +92 from last year).

Sources:

- https://pypistats.org/packages/scikit-build-core
- https://hugovk.github.io/top-pypi-packages/
- https://github.com/henryiii/pystats
- https://scikit-build-core.readthedocs.io/en/latest/projects.html
- GitHub Code Search

Adoption: Rapids.ai Case Study

- Used in package generator for all NVIDIA Rapids.ai packages (e.g `cudf`, `cugraph`, `cuml`, `rmm`)
- Handles CUDA variants
- Custom wrapper for name and dependency modification

Adoption: Ninja / CMake / clang-format

- PyPI redistribution of CLI tools
  - `ninja` and `cmake`: first-party scikit-build-core projects
  - `clang-format` : pip-installable wheels that are under 2 MB
- Convenient CMake variable `${SKBUILD_SCRIPTS_DIR}`
- Settings to remove dependency on specific python version:

```
[tool.scikit-build]
wheel.py-api = "py3"
```

Adoption: PyZMQ Case Study

```
[tool.scikit-build]
wheel.packages = ["zmq"]
wheel.license-files = ["licenses/LICENSE*"]
# 3.15 is required by scikit-build-core
cmake.version = ">=3.15"
# only build/install the pyzmq component
cmake.targets = ["pyzmq"]
install.components = ["pyzmq"]
```

Summary of Key Benefits

- Static configuration via `pyproject.toml`
- Powerful controls not possible with setuptools
- Improved cross-platform support
- Simplified build process for complex extensions

How to get started ?

Go to https://learn.scientific-python.org/development/

NEW PROJECT TEMPLATE

This guide comes with a copier/cookiecutter/cruft template for making new repos, scientific-python/cookie. Eleven build backends including compiled backends, generation tested in Nox, and kept in-sync with the guide.

```
pipx run cookiecutter gh:scientific-python/cookie
```

```
jcfr / tmp 1 pipx run cookiecutter gh:scientific-python/cookie
You've downloaded /home/jcfr/.cookiecutters/cookie before. Is it okay to delete and re-download it? [y/n] (y)
  [1/9] The name of your project (package): awesome
  [2/9] The name of your (GitHub?) org (org): awesome
  [3/9] The url to your GitHub or GitLab repository (https://github.com/awesome/awesome):
  [4/9] Your name (My Name):
  [5/9] Your email (me@email.com):
  [6/9] A short description of your project (A great package.):
  [7/9] Select a license
    1 - BSD
    2 - Apache
    3 - MIT
    Choose from [1/2/3] (1):
  [8/9] Choose a build backend
    1 - Hatchling                  - Pure Python (recommended)
    2 - Flit-core                  - Pure Python (minimal)
    3 - PDM-backend                - Pure Python
    4 - Whey                       - Pure Python
    5 - Poetry                     - Pure Python
    6 - Setuptools with pyproject.toml - Pure Python
    7 - Setuptools with setup.py   - Pure P
    8 - Setuptools and pybind11    - Compiled C++
    9 - Scikit-build-core          - Compiled C++ (recommended)
    10 - Meson-python              - Compiled C++ (also good)
    11 - Maturin                   - Compiled Rust (recommended)
    Choose from [1/2/3/4/5/6/7/8/9/10/11] (1): 9
  [9/9] Use version control for versioning [y/n] (y):
```

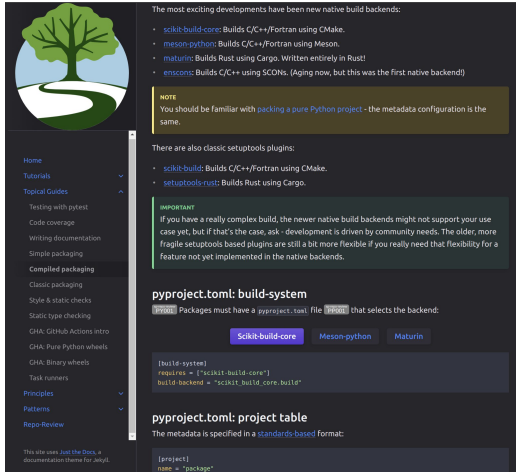
9 - Scikit-build-core - Compiled C++ (recommende

- ✅ GitHub Actions (testing & deploy)
- ✅ Cibuildwheel settings
- ✅ Pre-commit (ruff, mypy)
- ✅ ReadTheDocs-ready
- ✅ noxfile for local development

Wheel Published

Resources

- Scientific Python Library Development Guide
  https://learn.scientific-python.org/development/
- Documentation:
  https://scikit-build-core.readthedocs.io/
- GitHub:
  https://github.com/scikit-build/scikit-build-core
- CMake:
  https://cmake.org



Related Work

- Pybind11: Build system improvements
- Nanobind: Improved Stable ABI support
- cibuildwheel and build: Enhanced testing and usage
- validate-pyproject: Improved schema validation
- Pyodide: Better CMake support for WebAssembly

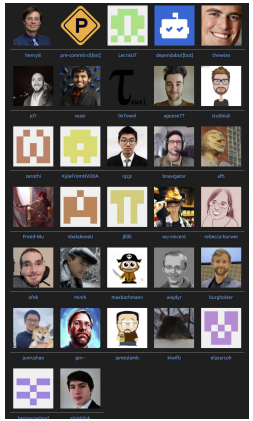
Sprints @ SciPy 2024

- Wanna experience the joy of publishing wheels 🚀, bring your project ✨
- Looking to be a contributor, join us 🤓

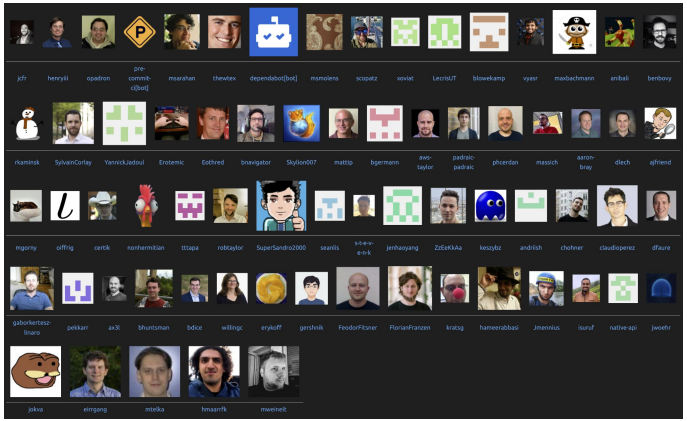
Thanks You

- My team members Henry Schreiner & Matt McCormick
- Attendees of the monthly developer and community scikit-build meetings
- Cibuildwheel & CMake teams
- PyPA, PEPs contributors, the wider community
- Scikit-build-core (left) & scikit-build (right) contributors:

~30



~70



scikit-build

kitware
Delivering Innovation

CMake

This material is based upon work supported by the National Science Foundation under Grant Number OAC-2209877.

Upcoming Releases

- 0.10: https://github.com/scikit-build/scikit-build-core/milestone/3
- 0.11: Work on plugins
  - setuptool plugin + update scikit-build classic to internally use scikit-build-core
  - hatch plugin: Work on adding editable (starting with in-place)
- 0.12: To be confirmed
  - at first, plugin pinned to X.Y of scikit-build-core
- 1.0