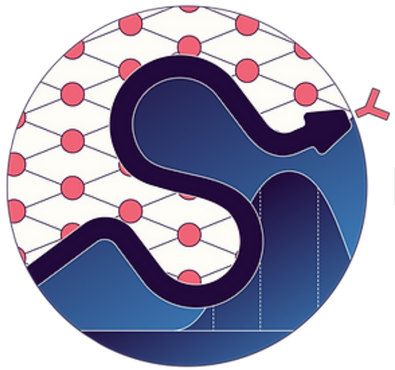
Building binary extensions

with pybind11, scikit-build, and cibuildwheel

SciPy 2022 - Austin



Maintainers track

July 15, 2022

Henry Schreiner
Princeton University

Joe Rickerby
Nord Projects

Ralf Grosse-Kunstleve
Google

Wenzel Jakob
Ecole Polytechnique Fédérale de Lausanne

Matthieu Darbois
PyPA

Aaron Gokaslan
Facebook Research

Jean-Christophe Fillion-Robin
Kitware

Matt McCormick
Kitware

Package: *pybind11*

Header-only pure C++11 CPython/PyPy interface

Trivial to add to a project
No special build requirements

No dependencies
No precompile phase
Not a new language

Designed for one purpose!

Think of it like the missing C++ API for CPython

Example of usage

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

int add(int i, int j) {
    return i + j;
}

PYBIND11_MODULE(example, m) {
    m.def("add", &add);
}
```

Standard include

Normal C++ to bind

Create a Python module

Signature statically inferred

Docs and parameter names optional

```
g++ -shared -fPIC example.cpp $(pipx run pybind11 --includes) -o example$(python3-config --extension-suffix)
```

Complete, working, no-install example (linux version)!

Many great features

Simple enough for tiny projects
714K code mentions of `pybind11` on GitHub

Powerful enough for huge projects
SciPy, PyTorch, dozens of Google projects

NumPy support without NumPy headers
No need to lock minimum NumPy at build

Buffer protocol and array classes
Includes Eigen support too

Most STL containers and features supported
Including C++17 additions, like `std::variant` (or boost)

Trampoline classes and multiple inheritance
Complex C++ is supported

Vectorize methods or functions
`py::vectorize`, even on a lambda function

Supports interop in both directions
Can even be used to embed Python in C++

Powerful object lifetime controls
`py::keep_alive`, `std::shared_ptr`, and more

Complete control of the Python representation
Special methods, inheritance, pickling, and more

Small binaries prioritized
Perfect for WebAssembly with Pyodide

Cross-extension ABI
One extension can return a type wrapped in another one

New in 2.8

2.7 was released at last SciPy!

py::raise_from
Chaining exceptions

Local exception mapping
Custom rules for each module

Array improvements
View and reshape arrays

Custom __new__ support
Fixing old bug

Python builtins as callbacks
Better consistency

Embedded interpreter improvements
argv setting, __file__ set

py::slice None support
Using std::optional

Rerun CMake with different Python
Updates cache if stale

mingw support added
Tested in CI now too

py::custom_type_setup
Advanced GC tricks and more

New in 2.9

py::args can be followed
No longer required to be last

py::const_name replaces py::_
More readable, gettext clash removed

Better support for std::string_view
Interoperable with built-in types

More built-in exception chaining
Using recent py::raise_from addition

py::multiple_inheritance detection
Only explicitly required if bases are hidden

Codebase formatted with clang-format
Using clang-format wheel & pre-commit

Lots of new checks, small fixes, better support for PyPy, CPython 3.11, etc.

New in 2.10

Removed Python 2.7 & 3.5 support
Removed 1K+ LoC, simpler, cleaner

Removed MSVC 2015 (& limited 2017)
Hard to find & not very useful with 3.5 gone

Python 3.11 beta support
First version working with all changes

Better exception handling
Easier without Python 2.7

py::any set and py::frozenset
Copy to set (like py::set)

New docs theme (Furo)
Cleaner, ToC for pages, dark-mode

Support std::monostate
std::variant as optional or non-constructible

py::capsule::set_name
Can manipulate later (DLPack)

NumPy type enhancements
More accessors and type number constructor

CMake fixes
Needed for Pyodide (web assembly) & VCPKG

2.10.0 releasing today

Upcoming plans

https://github.com/pybind/pybind11/wiki/Roadmap

Refactors
Decoupling unit tests
Breaking into smaller, IWUY headers

Merge the smart holder branch
Full interoperability with `std::unique_ptr` and `std::shared_ptr`
Opt-in with `<pybind11/smart_holder.h>`
Safe passing to C++, including trampolines

Optional Precompilation
Would dramatically speed up compile
Huge change, so being held back by other work

Further *ideas* (not promised yet)
Better MyPy integration (some minor work done)
Better Scikit-build integration
Upstreaming some improvements from nanobind (next slide)

Maintenance: Release notes

Somewhat interesting release note system

Pull requests get a template to fill out

```
<!--
Title (above): please place [branch_name] at the beginning if you are targeting a branch other than master.
*Do not target stable*. It is recommended to use conventional commit format, see conventionalcommits.org, but
not required.
-->
## Description

<!-- Include relevant issues or PRs here, describe what changed and why -->


## Suggested changelog entry:

<!-- Fill in the below block with the expected RestructuredText entry. Delete if no entry needed;
     but do not delete header or rst block if an entry is needed! Will be collected via a script. -->

```rst

```

<!-- If the upgrade guide needs updating, note that here too -->
```

Maintenance: Release notes (2)

GHA bot tags PR's after they get merged

```yaml
name: Labeler

on:
  pull_request_target:
    types: [closed]

jobs:
  label:
    name: Labeler
    runs-on: ubuntu-latest
    steps:
    - uses: actions/labeler@main
      if: github.event.pull_request.merged == true
      with:
        repo-token: ${{ secrets.GITHUB_TOKEN }}
        configuration-path: .github/labeler_merged.yml
```

You might need permissions: contents: …

+ some labeler config files

Maintenance: Release notes (3)

Script runner in nox

```python
@nox.session(reuse_venv=True)
def make_changelog(session: nox.Session) -> None:
    """
    Inspect the closed issues and make entries for a changelog.
    """

    session.install("ghapi", "rich")
    session.run("python", "tools/make_changelog.py")
```

Controls script dependencies

```
~/g/s/pybind11  skylion007/311-testing $…  nox -s make_changelog                    182ms  Thu Jul 14 22:42:52 2022
nox > Running session make_changelog
nox > Re-using existing virtual environment at .nox/make_changelog.
nox > python -m pip install ghapi rich
nox > python tools/make_changelog.py

* optimizes Eigen sparse matrix casting by removing unnecessary temporary.
  `#4064 <https://github.com/pybind/pybind11/pull/4064>`_

* .
  `#4060 <https://github.com/pybind/pybind11/pull/4060>`_

* Ensure proper behavior when garbage collecting classes with dynamic attributes in Python >=3.9.
  `#4051 <https://github.com/pybind/pybind11/pull/4051>`_

* A couple long-standing `PYBIND11_NAMESPACE` `__attribute__((visibility("hidden")))` inconsistencies are now fixed (affects only unusual environments).
  `#4043 <https://github.com/pybind/pybind11/pull/4043>`_

* Implicit conversion of the literal `0` to `pybind11::handle` is now disabled.
  `#4008 <https://github.com/pybind/pybind11/pull/4008>`_

* New theme for the documentation.
  `#3109 <https://github.com/pybind/pybind11/pull/3109>`_


------------------------------

Missing: chore: bump clang-tidy to 13
  https://github.com/pybind/pybind11/pull/3997

Template:

## Suggested changelog entry:

```rst

```

nox > Session make_changelog was successful.
~/g/s/pybind11 skylion007/311-testing $…   2609ms Thu Jul 14 22:42:58 2022
```

PRs can be updated, final polish on copy/paste
Labels manually removed (in bulk)

Bonus package: nanobind

C++17+ & Python 3.8+ only

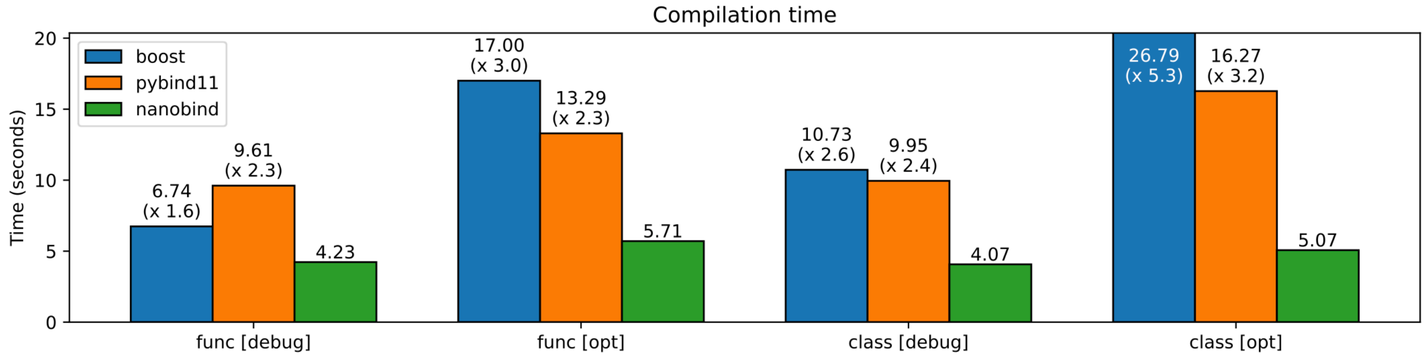
Similar API to pybind11

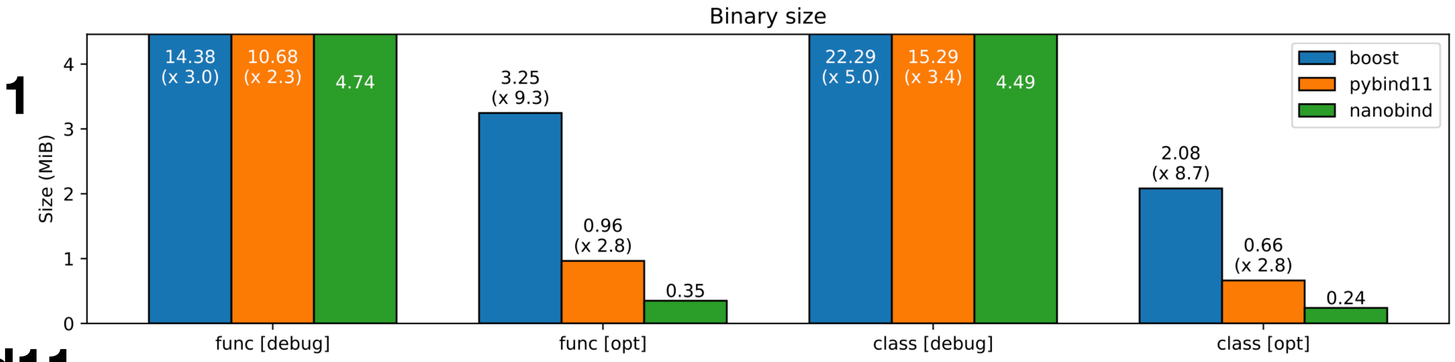
Intentionally more limited than pybind11

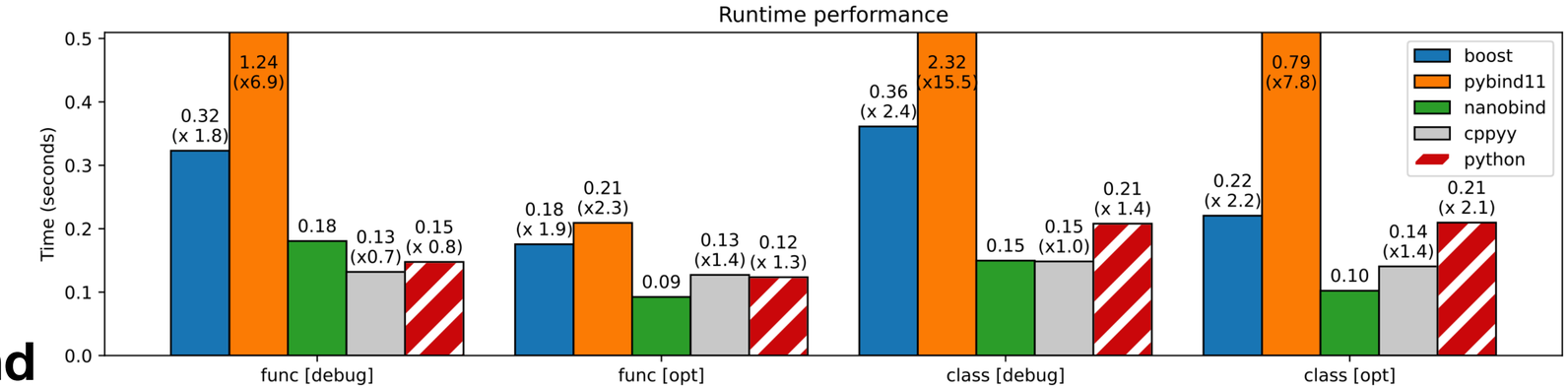
Focus on small, efficient bindings

Some ideas can be backported to pybind11

https://github.com/wjakob/nanobind







Bonus package: build

```
pipx run build
```

Builds SDist from source, wheel from SDist

```
pipx run build --sdist --wheel
```

Builds SDist and wheel from source

SDist and wheel builder

New features this year

Prettier printout
Better error messages
Python 3.11 support

Planned

Use Flit (better bootstrapping)
Redesign environment support
Threadsafe API

.github/workflows/wheels.yml

```yaml
on: [push, pull_request]

jobs:
  build_wheels:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4

    - name: Build wheels
      run: pipx run build

    - uses: actions/upload-artifact@v3
      with:
        path: ./wheelhouse/*.whl
```

Package: 🎡 cibuildwheel

Multiplatform redistributable wheel builder

Supports all major CI providers
GitHub Action provided too!
Can run locally, on servers, etc.

Affiliated (shared maintainer) with manylinux
Close collaboration with PyPy devs
Joined the PyPA in 2021
Used by matplotlib, mypy, scikit-learn, and more

Supports:
Targeting macOS 10.9+
Apple Silicon cross-compiling 3.8+
All variants of manylinux (including emulation)
musllinux
PyPy 3.7-3.9
Repairing and testing wheels
Reproducible pinned defaults (can unpin)

2.0 was released at last SciPy!

.github/workflows/wheels.yml

```yaml
on: [push, pull_request]

jobs:
  build_wheels:
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os:
        - ubuntu-22.04
        - windows-2022
        - macos-11

    steps:
    - uses: actions/checkout@v4

    - name: Build wheels
      uses: pypa/cibuildwheel@v2.8.0

    - uses: actions/upload-artifact@v3
      with:
        path: ./wheelhouse/*.whl
```

Manylinux

manylinux1	**RHEL 5 (CentOS)**	Support ended Jan 1, 2022 Updated till CI breaks
manylinux2010	**RHEL 6 (CentOS)**	Support ending Aug 1, 2022
manylinux2014	**RHEL 7 (CentOS)**	**Default in cibuildwheel**
manylinux_2_24	**Debian 9**	Support likely to end
manylinux_2_28	**RHEL 8 (Almalinux)**	
musllinux_1_1	Alpine 3.12	MUSL distributions

Cibuildwheel: what's new

New in cibuildwheel 2.1-2.4

Local Windows & MacOS runs
TOML overrides array
manylinux2014 default
musllinux
Environment variable passthrough
Experimental Windows ARM

New in cibuildwheel 2.5-2.8

ABI3 wheel support
Build from SDist
tomllib on Python 3.11 (host)
CPython 3.11 beta support
manylinux_2_28 support (RHEL 8)
Linux builds from Windows
Podman support
Support for Pythonless wheels

In Development

New setup-python environment
Python 3.6 removed as host
Python 3.11b4 in progress as of today

Cibuildwheel: what might be

Cross compiling Windows ARM
Supporting setuptools, hopefully more

Cross compiling Linux
Stuck with missing RH dev toolkit trick

Local run improvements
Skip an environment variable

cibuildwheel tips

Build wheels locally

```
pipx run cibuildwheel --platform linux
```

Use pyproject.toml config

Keep the GitHub Action up-to-date!
.github/dependabot.yml

```yaml
version: 2
updates:
  - package-ecosystem: "github-actions"
    directory: "/"
    schedule:
      interval: "daily"
    ignore:
      - dependency-name: "actions/*"
        update-types:
          - version-update:semver-minor
          - version-update:semver-patch
```

cibuildwheel tips

Build wheels locally

```
pipx run cibuildwheel --platform linux
```

Use pyproject.toml config

Keep the GitHub Action up-to-date!
`.github/dependabot.yml`

```yaml
version: 2
updates:
  - package-ecosystem: "github-actions"
    directory: "/"
    schedule:
      interval: "daily"
```

No longer needed,
Dependabot respects version level!
(v1 -> v2, not v2.0.0)

GHA: building a composite action

GitHub Action's new setup-python@v4.1.0 feature is perfect for composite pipx actions!

```yaml
name: mypkg
description: 'Runs a package'

runs:
  using: composite
  steps:
    - uses: actions/setup-python@v4
      id: python
      with:
        python-version: "3.7 - 3.10"
        update-environment: false

    - run: >
        pipx run
        --python '${{ steps.python.outputs.python-path }}'
        --spec '${{ github.action_path }}'
        mypkg
      shell: bash
```

No side effects!

Used in nox & cibuildwheel

Package you want to run

Package: scikit-build

CMake based build backend for Python

Scikit-build is a CMake-setuptools adaptor from KitWare, the makers of CMake
First introduced as PyCMake at SciPy 2014 and renamed in 2016
Includes CMake for Python and ninja for Python

CMake and Ninja for Python
All manylinux archs • Apple Silicon • cibuildwheel • nox

Currently designed as a setuptools wrapper
But that is changing…

Pure Python Packaging

```toml
# pyproject.toml

[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"

[project]
name = "example"
version = "0.1.0"
```

Not tied to a single solution!
Adding distutils to the stdlib was a disaster
Setuptools extending distutils was also a disaster
But modern standards free us from this!

Many pure-Python backends today!
Hatch, PDM, Flit, and more!
Even modern setuptools supports this

But what if you want to add compiled extensions?
Fall back 8+ years into distutils/setuptools hacks

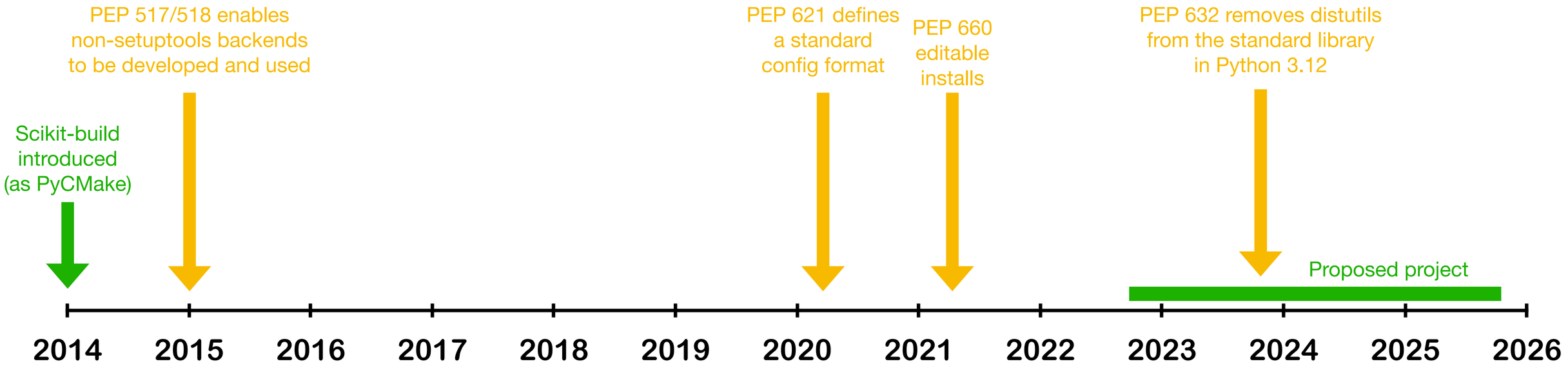
See https://scikit-hep.org/developer
Or https://packaging.python.org

Packaging timeline

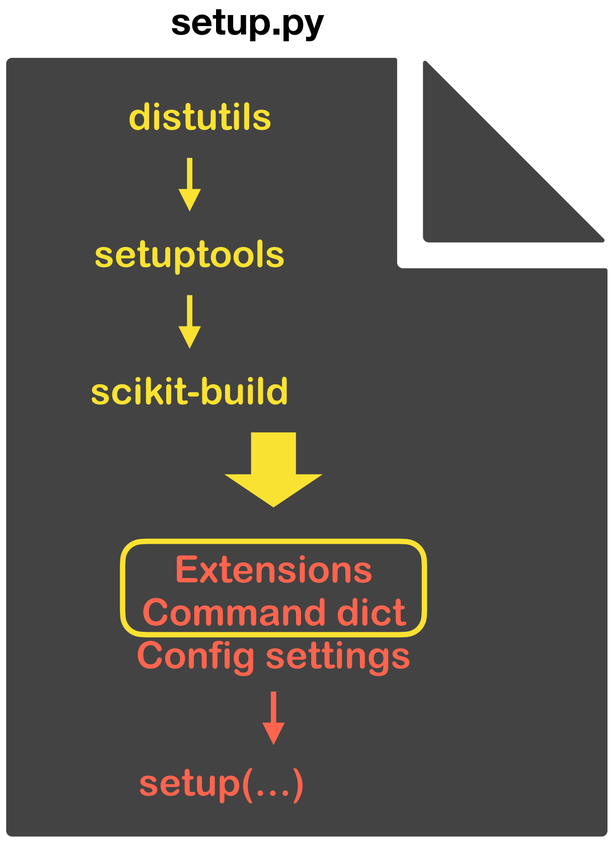
Before 2015, the only method to distribute packages was distutils/setuptools

Python Enhancement Proposals (PEPs) drive Python development - and provide standards for packaging now too!

Good pure Python backends exist now (hatchling) - scikit-build could be a great compiled / CMake one!



Scikit-build (current design)



MANIFEST.in
setup.cfg
pyproject.toml
CMakeLists.txt

Raw setuptools for C++?

No C++ standard support
No parallel compiles
No common libraries
No custom libraries
Hacks for cross compiling

NumPy alone needs 13K lines to support this!

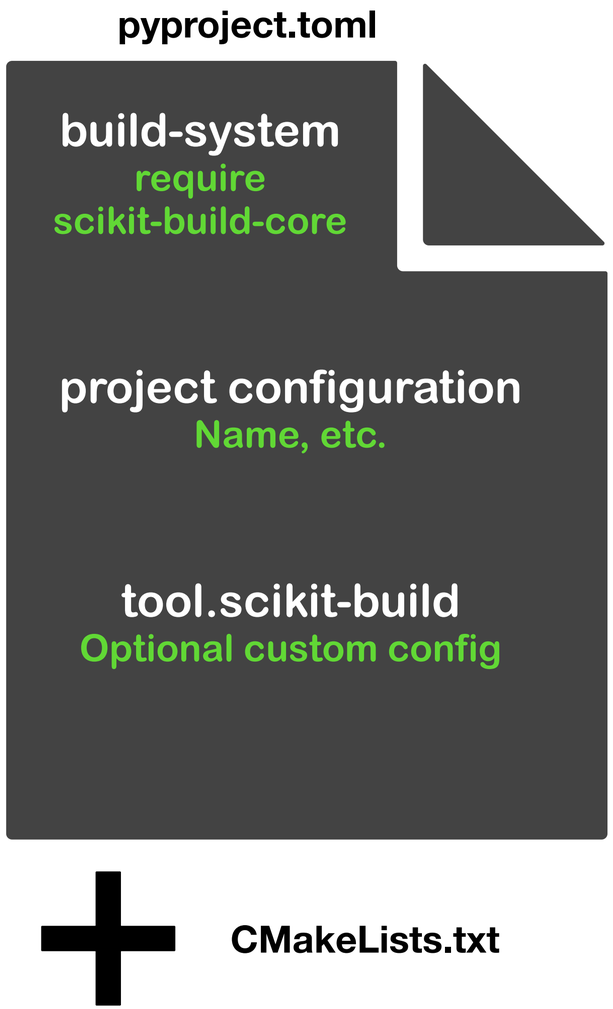
Scikit-build wraps CMake!

Best in class feature support
~60% of all C++ projects
Very popular in the sciences (incl. HEP)
Common and custom library support

Problems with setuptools bridge

Inherits problems from setuptools
Deep dependencies on internals
Poor interactions with new standards
Deep nesting is hard to work with / debug
Internals poorly documented
Not composable with other plugins

Scikit-build (new design)



Design considerations:

Good testing suite
Excellent error messages
Public API

Planned other interfaces:

Extension-based setuptools wrapper
Can be used to support classic “scikit-build”
Plugin mode (see next page)

You could use pybind11 from pip in CMake!

CMake code:

Redesign with modern “FindPython” support
Provide extension discovery system

(Scikit-build-core may be a placeholder name)

Extensionlib (w/ PEP?)

(Rough example of idea)

```toml
[build-system]
requires = ["hatchling", “scikit-build-core”, "mypyc"]
build-backend = "hatchling.build"

[project]
name = "example"
version = "0.1.0"

[[build-system.extensions]]
build-backend = "scikit_build_core.extension"
src = "src1/CMakeLists.txt"

[[build-system.extensions]]
build-backend = "scikit_build_core.extension"
src = "src2/CMakeLists.txt"

[[build-system.extensions]]
build-backend = "mypyc.extension"
src = "src/*.py"
```

Since the proposal, “extensionlib” and/or a PEP has been discussed at PyCon 2022. A PoC lib by @ofek exists. Stay tuned!

Provides a common API for build-backends to extension creation tools

We could be a plugin for all builders via a single interface!

NSF Proposal

Year 1 plans:

Introduce scikit-build-core: modern PEP 517 backend
Support PEP 621 configuration
Support use as plugin (possibly via extesionlib)
Tighter CMake integration (config from PyPI packages)
Distutils-free code ready for Python 3.12

Year 2 plans:

Convert projects to Scikit-build

Year 3 plans:

Website, tutorials, outreach

https://iscinumpy.dev/post/scikit-build-proposal/

On Wednesday, NSF 2209877 was awarded!

Updates can touch many repos!

Scikit-build family:
scikit-build, cmake, ninja, moderncmakedomain, scikit-build-examples

PyPA family:
cibuildwheel, build

pybind family:
pybind11, python_example, cmake_example, scikit_build_example

scikit-hep family:
Developer pages, cookie

Scikit-build, since last time

CMake
Current release (3.23) will likely drop manylinux1 wheels

Scikit-build 0.13
ARM on Windows support
MSVC 2022 support
Target link libraries support via keyword

Scikit-build 0.14
Install target

Scikit-build 0.15
Cygwin support
(Limited) FindPython support
Final Python 2.7 & 3.5 release
Final MSVC 2015 release

+ lots of fixes
& minor improvements

Questions?

To get in touch, currently we are using GitHub Discussions on all three packages.

Special thank you to Google for sponsoring our expanded CI, and for Ralf Grosse-Kunstleve's time.

Henry Schreiner supported by IRIS-HEP and the NSF under Cooperative Agreement OAC-1836650

https://iscinumpy.dev

henryiii

henryschreiner3

https://scikit-hep.org
https://iris-hep.org

C++ & Python

pybind11 (python_example, cmake_example, scikit_build_example) • Conda-Forge ROOT

Building Python Packages

cibuildwheel • build • scikit-build (cmake, ninja, sample-projects) • Scikit-HEP/cookie

Scikit-HEP: Histograms

boost-histogram • Hist • UHI • uproot-browser

Scikit-HEP: Other

Vector • Particle • DecayLanguage • repo-review

Other C++

CLI11 • GooFit

Other Ruby

Jekyll-Indico

Other Python

Plumbum • POVM • PyTest GHA annotate-failures

My books and workshops

Modern CMake • CMake Workshop
Computational Physics Class
Python CPU, GPU, Compiled minicourses
Level Up Your Python

Scikit-HEP Developer Pages

Bonus slides (from Scikit-HEP talk yesterday)

Scikit-HEP developer pages

Scikit-HEP was growing quickly
How to keep quality high across all packages?
scikit-hep.org/developer!

Topics
Packaging (classic & simple)
Style guide (pre-commit)
Development setup
pytest
Static typing (MyPy)
GitHub Actions (3 pages)
Nox

Utilities
Cookie-cutter
Repo-review

Maintained by nox
& GitHub Actions

Guided our other packages, like Awkward

Universally useful advice for maintaining packages!

Search Scikit-HEP

Scikit-HEP

Scikit-HEP project - welcome!

The Scikit-HEP project is a community-driven and community-oriented project with the aim o
providing Particle Physics at large with an ecosystem for data analysis in Python. Read more

New users can start with our user pages. See our developer pages for information on develop
Python packages.

NEWS · TUTORIAL · RESOURCES · CITE US · GET IN TOUCH

Basics:

Awkward Array — Manipulate JSON-like data with NumPy-like idioms.

hepunits — Units and constants in the HEP system of units.

VECTOR — Manipulate Lorentz, 3D, and 2D vectors in NumPy, Numba, or Aw

Style

Pre-commit hooks
Black
Check-Manifest (setuptools only)
Type checking
PyCln
Flake8
YesQA
isort
PyUpgrade
Setup.cfg format (setuptools only)
Spelling
PyGrep hooks
Prettier
Clang-format (C++ only)
Shellcheck (shell scripts only)
PyLint (noisy)

Modern (simple) packaging

pyproject.toml

```toml
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"

[project]
name = "package"
version = "0.1.2"
```

No setup.py, setup.cfg, or MANIFEST.in required!

Cookiecutter

```
pipx run cookiecutter gh:scikit-hep/cookie
```

11 backends to pick from
Generation tested by nox
In sync with the developer pages

Setuptools
Setuptools PEP 621
Flit Hatch PDM
Poetry

Scikit-build
Setuptools C++
Maturin (Rust)

+ more!

Scikit-HEP Repo Review (beta)

You can check the style of a GitHub repository below. Enter any repository, such as `scikit-hep/hist`, and the branch you want to check, such as `main` (it must exist). This will produce a list of results - green checkmarks mean this rule is followed, red errors mean the rule is not. A yellow warning sign means that the check was skipped because a previous required check failed. Some checks will fail, that's okay - the goal is bring all possible issues to your attention, not to force compliance with arbitrary checks.

You can also run this tool locally:

```
pipx run 'scikit-hep-repo-review[cli]' <path to repo>
```

Org/Repo: pypa/build
e.g. scikit-hep/hist

Branch: main
e.g. main

Results for pypa/build@main

general

- PY001: Has a pyproject.toml
- PY002: Has a README.(md|rst) file
- PY003: Has a LICENSE* file
- PY004: Has docs folder

**Runs in WebAssembly locally!
(via Pyodide)**