ROOT on Homebrew, pip plans

Henry Schreiner

May 10, 2022

Topics

Homebrew

Introduction to brew
ROOT package use
Developing and maintaining

pip plans

Introduction to packaging
Advanced packaging
Plans with Scikit-build

Homebrew

https://brew.sh

Package manager for macOS & Linux

Simple Ruby eDSL for package recipes

Binaries for 3 Intel macOS, 2 AS, and Linux — Can build from source or install from `--head`

Single prefix (linux user installable)

Latest only philosophy — Special cases can be manually added with `<pkg>@<version>`

Cask support (pre-built binaries)

Using brew

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Only requires git ~~and Ruby~~ (actually, bootstraps it's own Ruby these days)

`brew install <package>`	Install a package
`brew update`	Update all packages
`brew cleanup`	Remove cached files
`brew search <expr>`	Look for a package
`brew info <package>`	See dependencies, etc.

```
brew bump-formula-pr --url https://root.cern.ch/download/root_v6.26.02.source.tar.gz --version 6.26.02 root
```

Brew bundles

Single file for setting up a new Mac!

`brew bundle`

```
# Brewfile

brew "python"        # The (almost) latest Python
brew "root"          # High Energy Physics toolkit
```

My actual Brewfile ->

https://iscinumpy.dev/post/setup-a-new-mac/

```
tap "homebrew/bundle"       # First line of a bundle
tap "homebrew/cask"         # Not needed on command line
tap "homebrew/cask-fonts"   # Just needed for font casks below
tap "homebrew/core"         # Not needed on command line

# Building tools
brew "boost"         # C++ library
brew "ccache"        # Faster builds by caching
brew "cmake"         # Build software projects
brew "ninja"         # Replacement for make
brew "doxygen"       # Doxygen generates C++ documentation
brew "pre-commit"    # Allows pre-commit hooks to be installed and managed
brew "tbb"           # Threaded building blocks from Intel
brew "swig"          # Software wrapper interface generator
brew "qt"            # The Qt Toolkit

# General utilities
brew "colordiff"     # More colorful diffs outside of git
brew "coreutils"     # Basic stuff with a g prefix
brew "gnu-sed"       # Adds the gsed command, more powerful than BSD sed
brew "gnu-time"      # Nicer timing
brew "openssl"       # Security stuff
brew "git"           # The latest version of git instead of Apple's older one
brew "git-gui"       # A quicker way to apply partial changes
brew "htop"          # htop is better than top for checking processes
brew "tree"          # tree is nice for looking at directories
brew "wget"          # Mac's have curl by default, but not wget
brew "bash"          # Bash 5 instead of 3, in case you need it
brew "rename"        # Rename files utility
brew "clang-format"  # Format C++ files
brew "tmux"          # Split windows and saving terminal sessions (screen replacement)
brew "gh"            # GitHub's command line interface, from gh's tap
brew "bat"           # Nicely colorized replacement for cat

# Personal customization options
brew "fish"          # My favorite shell. Might move to zsh when macOS does, though
brew "lmod"          # See my posts on lmod
brew "macvim"        # VI for macOS, with mvim graphical interface too
brew "interactive-rebase-tool" # Run git config --global sequence.editor interactive-rebase-tool
brew "bash-completion" # Nicer completion for bash if you use it

# Programming languages
brew "python"        # Python 3.8
brew "numpy"         # Now is Python3 only (numpy@1.16 is for python@2)
brew "go"            # Used by hugo, can be useful to have
brew "node"          # Javascript (for gitbooks, etc)
brew "yarn"          # Package manager for node.js
brew "ruby"          # Just to be extra sure the system Ruby never gets modified
brew "rbenv"         # Use this for Ruby (pyenv also exists)
brew "rust"          # Was trying out mdbook
brew "lua"           # Lightweight language like Python
brew "java"          # Meh. What can I say?

# Python programs
brew "pipx"          # Better way to add PyPI applications
brew "pipenv"        # All-in-one environment tool
brew "nox"           # Tool for standard development environments
brew "tox"           # Old tool for standard development environments
brew "poetry"        # Nice all-in-one packaging tool
brew "jupyterlab"    # Programming environment
brew "black"         # Python formatting
brew "mypy"          # Python type checking
brew "cookiecutter" # Quickly start new projects

# Packages
brew "hugo"          # Fast website generator
brew "pandoc"        # Convert between document formats
brew "pdftk-java"    # PDF Tool Kit (Java port)
brew "qt"            # The #1 graphics library for C++ and Python
brew "root"          # High Energy Physics toolkit

brew "libsodium"     # Have no idea why I needed this


# Fonts
cask "font-hack-nerd-font"
cask "font-sauce-code-pro-nerd-font"

# Core
cask "iterm2"        # A great terminal
cask "mactex"        # LaTeX. Huge.
cask "miniconda"     # Nice way to get a system Conda install
cask "java"          # The programming language vm

# Programs
cask "google-chrome"# Since once and a while a site doesn't work with Safari
cask "gimp"          # Photo editor
cask "blender"       # The 3D application
cask "inkscape"      # 2D vector drawings

# Editors
cask "macdown"       # Nice Markdown
cask "texstudio"     # Nice IDE for LaTeX
cask "meld"          # Compare files graphically.
cask "tikzit"        # Fast drawings
cask "visual-studio-code"

# Daemons
cask "docker"        # Allows running and building docker images
cask "dropbox"       # The cloud
cask "synergy"       # Share a mouse and keyboard between computers. Free option is okay.
cask "amethyst"      # Simulate a non-overlapping window manager with keyboard shortcuts
cask "xquartz"       # Legacy Linux apps may need this

# Chat
cask "mattermost"
cask "skype"
cask "slack"
cask "element"
```

Brew link

Installed to /usr/local/opt/root (Intel macOS):
`brew --prefix root`

Link: make symlinks to /usr/opt (Intel)
`brew link root`

Unlink: remove symlinks
`brew unlink root`

No need for thisroot scripts!

Formula

```ruby
class Wget < Formula
  homepage "https://www.gnu.org/software/wget/"
  url "https://ftp.gnu.org/gnu/wget/wget-1.15.tar.gz"
  sha256 "52126be8cf1bddd7536886e74c053ad7d0ed2aa89b4b630f76785bac21695fcd"

  depends_on "pkg-config" => :build
  depends_on "libidn2"
  depends_on "openssl@1.1"

  on_linux do
    depends_on "util-linux"
  end

  def install
    system "./configure", "--prefix=#{prefix}"
    system "make", "install"
  end

  test do
    system bin/"wget", "-O", "/dev/null", "https://google.com"
  end
end
```

Instant formula edits:

```
brew edit wget
brew install --build-from-source
```

Very elegant, powerful Ruby eDSL

Lots of options and access to important details

Binaries injected by CI

Everything in git

brew info root

```
root: stable 6.26.02 (bottled), HEAD
Object oriented framework for large scale data analysis
https://root.cern.ch/
/usr/local/Cellar/root/6.26.02_1 (6,416 files, 537.2MB) *
  Poured from bottle on 2022-04-27 at 23:23:42
From: https://github.com/Homebrew/homebrew-core/blob/HEAD/Formula/root.rb
License: LGPL-2.1-or-later
==> Dependencies
Build: cmake ✔, ninja ✔
Required: cfitsio ✔, davix ✔, fftw ✔, gcc ✔, gl2ps ✔, glew ✔, graphviz ✔, gsl ✔, lz4 ✔, mysql-client ✔, numpy ✔, openblas ✔, openssl@1.1 ✔, pcre ✔,
python@3.9 ✔, sqlite ✔, tbb ✔, xrootd ✔, xz ✔, zstd ✔
==> Requirements
Required: Xcode ✔
==> Options
--HEAD
      Install HEAD version
==> Caveats
As of ROOT 6.22, you should not need the thisroot scripts; but if you
depend on the custom variables set by them, you can still run them:

For bash users:
  . /usr/local/bin/thisroot.sh
For zsh users:
  pushd /usr/local >/dev/null; . bin/thisroot.sh; popd >/dev/null
For csh/tcsh users:
  source /usr/local/bin/thisroot.csh
For fish users:
  . /usr/local/bin/thisroot.fish

Emacs Lisp files have been installed to:
  /usr/local/share/emacs/site-lisp/root
==> Analytics
install: 1,609 (30 days), 4,524 (90 days), 15,670 (365 days)
install-on-request: 1,594 (30 days), 4,485 (90 days), 15,535 (365 days)
build-error: 6 (30 days)
```

Specifics

Currently using Python 3.9
(Still the “python” formula)

C++17 mode
(Used to depend on macOS version)

Currently using built-in LLVM
(Build requires long-timeout label)

Can’t use the default GCC 5 on Linux
(Internal brew GCC fine)

```
inreplace "cmake/modules/SearchInstalledSoftware.cmake" do |s|
  # Enforce secure downloads of vendored dependencies. These are
  # checksummed in the cmake file with sha256.
  s.gsub! "http://lcgpackages", "https://lcgpackages"
  # Patch out check that skips using brewed glew.
  s.gsub! "CMAKE_VERSION VERSION_GREATER 3.15", "CMAKE_VERSION VERSION_GREATER 99.99"
end
```

This is all the patching done! Vanilla ROOT otherwise

Bottles for all supported platforms

```
bottle do
  sha256 arm64_monterey: "4f5223ee441865d869a1b935d31a22c85f8f9aac869d69eba7d9109aaebbee3b"
  sha256 arm64_big_sur:  "980a8bdec3fd26a6912066935634bb5826dbfeaee72cdfb8f3d921531aeba61e"
  sha256 monterey:       "aab0d84528e3ecd8441ad4bacd54f60d7183a9153f3273f5fa46877924369a15"
  sha256 big_sur:        "3d79db03d061064ba67b06dc2b79d8c823d517d656e1c94793c17bcfecc9c97e"
  sha256 catalina:       "bad4b634d1adb2287765b5bd1c097888277b84415a4efcab20d1e2385ffeb8f5"
  sha256 x86_64_linux:   "054d674fcbd968b84a21c88c229e666553f319882ea20ea6864a80ceeb431326"
end
```

Python: how to use in venv

Like most distributions of ROOT, this does not play well with virtualenvs!

Solution: use `--system-site-packages`

Demo (from scratch in Docker)

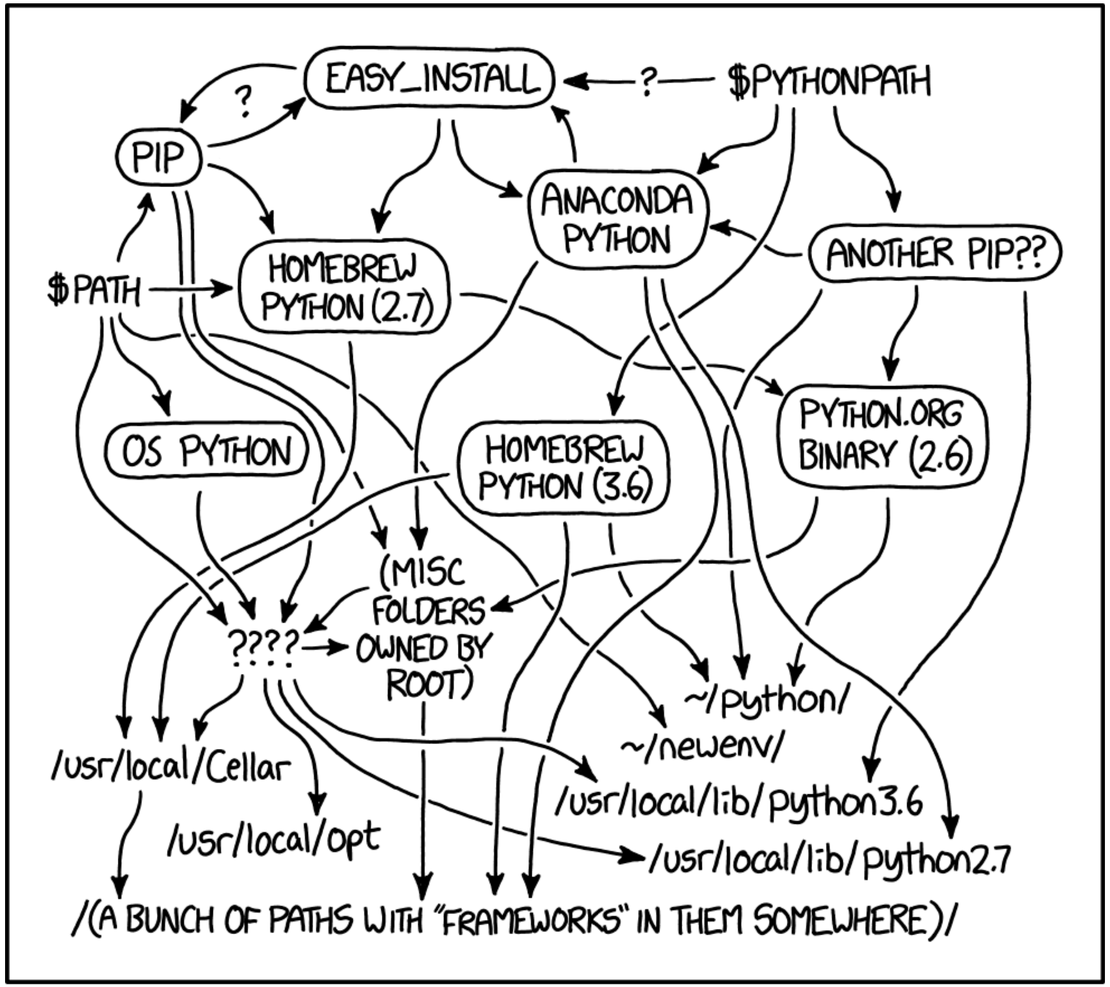
```
docker run --rm -it ubuntu
apt update && apt install -y curl git build-essential
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"' >> /root/.profile
eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
brew install root
root
   ------------------------------------------------------------
  | Welcome to ROOT 6.26/02                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Apr 12 2022, 16:28:03                 |
  | From tags/v6-26-02@v6-26-02                                      |
  | With g++-11 (Homebrew GCC 11.3.0) 11.3.0                         |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------

root [0]
```

Future ideas

Dual Python build (3.9 & 3.10)? (easy transition)

Use llvm@13 directly next release? (current version requires patches)

Intro to Python Packaging



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987

How do I install a package?

sudo pip install <package> — Terrible - no explanation needed

pip install <package> — Bad - installs to your global system or reverts to --user

pip install --user <package> — Bad - installs to a “global” user directory

tensorflow -> typing_extensions >=3.7,<4
black -> typing_extensions >=4

These will never really be used together! Black is an app!

Unsolvable environment!

Updating existing packages is much harder than a fresh solve

A new user can’t be sure of a solve if things change

Solution: virtual environments

Or use virtualenv, faster!

```
python -m venv .venv
```

```
. .venv/bin/activate
# work here
deactivate
```

Conda version: avoid the base environment

Aside: “app” solution: pipx

Creates an internal venv, exports just the entrypoint
`pipx install mypy`
Only executables available!

Creates a temporary venv, refresh after 1 week
`pipx run mypy`
No worries about what is installed or updating!

Try `pipx run uproot-browser browse <rootfile>`

Beyond virtualenv's

pip-tools: pip-compile
Full environment locking with hashes

Poetry, PDM:
Single environment solution

Hatch:
Multi environment solution

`pdm install` **Restores exact locked environment**

`pdm update` **Updates using the original requirements**

Perfect for deploying a website…
Or a reproducible analysis!

How to make ROOT work?

We need pip install root!

How far are we from that? What would it take?

ROOT uses CMake…

Wouldn’t it be nice if we could use CMake in Python?

Scikit-build

Started in 2014 as PyCMake, developed by KitWare

Two new maintainers recently joined!

Packages:
scikit-build • cmake • ninja • moderncmakedomain

Current design:
Wrapper around setuptools for CMake

Plans waiting on funding:
https://iscinumpy.dev/post/scikit-build-proposal
ROOT was one of the partner projects!

Proposal outline

Stage 1: rework scikit-build

Develop scikit-build-core
PEP 517 builder, setuptools/distutils free

Compatibility layer for scikit-build
Limited public API helps

PEP 621 direct build
Best for many cases?

Proper setuptools extension
And Hatch, Poetry, etc.
Generalize, perhaps?

Add extension discovery mechanism
Easy integration with pybind11, other Python packages!
Possible support in CMake itself

```
# pyproject.toml
requires = ["pybind11", …]

# CMakeLists.txt
find_package(pybind11 CONFIG REQUIRED)
```

Stage 2: help partner projects adapt/update scikit-build

ROOT was one of 10+ partner projects!
New scikit-build example on numpy.org

Stage 3: extensive docs work and tutorial workshops

Active space!

Setuptools is no longer the only way to package!
(PEP 517, 518, 621, 660)

Interest growing in binary builds & plugins!
(See Packaging Summit at PyConUS 2022)

---

clang-format-wheel

Scikit-Build
Runs LLVM's CMake build

cibuildwheel
Builds python-independent binary wheels

1-2 MB binaries on PyPI
No "binding", only entrypoint!

```
pipx run clang-format
```

Also see give-me-python!

Use with pre-commit, even on pre-commit.ci!

```yaml
- repo: https://github.com/pre-commit/mirrors-clang-format
  rev: "v14.0.1"
  hooks:
  - id: clang-format
    types_or: [c++, c, cuda]
```

Experimental ROOT test

Added scikit-build setup.py, set a few options
Basic pyproject.toml
Hacky empty package folder to make setuptools happy

```
python -m pip install . -v
pipx run build --sdist --wheel
```

✔

Correctly runs CMake build!
“root” command works!
Library in wrong place, but manually importable!

✘

166MB SDist
222MB wheel

Tons of random files included
Structure incorrect for Python package
Lots of “global” (data) files

Further reading

https://iscinumpy.dev

My PyCon US 2022 talk & packaging summit

This work was partially supported by the National Science Foundation under Cooperative Agreement OAC-1836650.

https://iscinumpy.dev

My Projects

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

henryiii
henryschreiner3