scikit-ci-addons Documentation

Release 0.25.0.post0.dev0+geb8de6c

The scikit-build team

Feb 13, 2022

User guide

scikit-ci-addons is a command line tool and a set of scripts useful to help drive the CI of projects leveraging services like AppVeyor, CircleCI, or Travis CI.

Originally developed to help install prerequisites for building Python extension, it is now useful to support other type of projects.

CHAPTER 1

Installation

1.1 Install package with pip

To install with pip:

```
$ pip install scikit-ci-addons
```

1.2 Install from source

To install scikit-ci-addons from the latest source, first obtain the source code:

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

then install with:

```
$ pip install .
```

or:

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

for development.

1.3 Dependencies

1.3.1 Python Packages

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

```
githubrelease>=1.5.7
lxml;python_version<'3.8'
```

The development dependencies (for testing and coverage) are:

```
codecov==2.0.5
coverage==4.2
flake8==3.7.7
pytest==3.0.3
pytest-cov==2.4.0
pytest-mock>=1.4.0
pytest-runner==2.9
wheel==0.29.0
```

CHAPTER 2

Usage

The scikit-ci-addons command line executable allows to discover, execute and get the path of any of the distributed *add-ons*.

2.1 Executing an add-on

```
ci_addons ADDON_NAME
```

where `ADDON_NAME` can be any of the names displayed using `ci_addons --list`.

For example:

```
$ ci_addons appveyor/patch_vs2008
```

2.2 Listing available add-ons

```
ci_addons --list
```

For example:

```
$ ci_addons --list

anyci/ctest_junit_formatter.py
anyci/publish_github_release.py
anyci/run.sh
anyci/ctest_junit_formatter.xsl
anyci/noop.py
anyci/docker.py

appveyor/enable-worker-remote-access.ps1
```

(continues on next page)

(continued from previous page)

```
appveyor/install_cmake.py
appveyor/apply_mingw_path_fix.py
appveyor/run.cmd
appveyor/patch_vs2008.py
appveyor/run-with-mingw.cmd
appveyor/cancel-queued-build.ps1
appveyor/rolling-build.ps1
appveyor/tweak_environment.py
appveyor/run-with-visual-studio.cmd

circle/install_cmake.py

travis/install_cmake.py
travis/enable-worker-remote-access.sh
travis/run-with-pyenv.sh
travis/install_pyenv.py

windows/install-miniconda3.ps1
windows/install-utils.ps1
windows/install-cmake.ps1
windows/install-python-27-x64.ps1
windows/install-nsis.ps1
windows/install-svn.ps1
windows/install-ninja.ps1
windows/install-python.ps1
windows/install-python-36-x64.ps1
windows/install-git.ps1
windows/install-flang.ps1
```

Note: To learn more about each add-on, consider reading the *add-ons* section.

2.3 Getting directory containing all add-ons

```
ci_addons --home
```

For example:

```
$ ci_addons --home
/home/jcfr/.virtualenvs/test/local/lib/python2.7/site-packages
```

2.4 Installing add-ons into selected directory

```
ci_addons --install DIR
```

where `DIR` is a valid path to an existing directory.

For example:

```
$ ci_addons --install /tmp
/tmp/anyci/ctest_junit_formatter.py
/tmp/anyci/publish_github_release.py
/tmp/anyci/run.sh
/tmp/anyci/ctest_junit_formatter.xsl
/tmp/anyci/noop.py
/tmp/anyci/docker.py
/tmp/appveyor/enable-worker-remote-access.ps1
/tmp/appveyor/install_cmake.py
/tmp/appveyor/apply_mingw_path_fix.py
/tmp/appveyor/run.cmd
/tmp/appveyor/patch_vs2008.py
/tmp/appveyor/run-with-mingw.cmd
/tmp/appveyor/cancel-queued-build.ps1
/tmp/appveyor/rolling-build.ps1
/tmp/appveyor/tweak_environment.py
/tmp/appveyor/run-with-visual-studio.cmd
/tmp/circle/install_cmake.py
/tmp/travis/install_cmake.py
/tmp/travis/enable-worker-remote-access.sh
/tmp/travis/run-with-pyenv.sh
/tmp/travis/install_pyenv.py
/tmp/windows/install-miniconda3.ps1
/tmp/windows/install-utils.ps1
/tmp/windows/install-cmake.ps1
/tmp/windows/install-python-27-x64.ps1
/tmp/windows/install-nsis.ps1
/tmp/windows/install-svn.ps1
/tmp/windows/install-ninja.ps1
/tmp/windows/install-python.ps1
/tmp/windows/install-python-36-x64.ps1
/tmp/windows/install-git.ps1
/tmp/windows/install-flang.ps1
```

2.5 Getting full path of an add-on

```
ci_addons --path PATH
```

where `PATH` can be any of these:

- relative path with or without extension (e.g `appveyor/patch_vs2008.py` or `appveyor/patch_vs2008.py`)
- full path (e.g `/path/to/appveyor/patch_vs2008.py`)
- script name with or without extension (e.g `patch_vs2008.py` or `patch_vs2008`). If there are multiple add-ons with the same bame, `ci_addons` reports an error message listing the add-ons to choose from.

For example:

```
$ ci_addons --path appveyor/patch_vs2008.py
/home/jcfr/.virtualenvs/test/local/lib/python2.7/site-packages/appveyor/patch_vs2008.
↪py
```

Note: This function is particularly useful when the selected add-on is not a python script and is expected to be used

as an input to an other tool.

2.6 Calling scikit-ci-addons through `python -m ci_addons`

You can invoke scikit-ci-addons through the Python interpreter from the command line:

```
python -m ci_addons [...]
```

This is equivalent to invoking the command line script `ci_addons [...]` directly.

2.7 Getting help on version, option names

```
ci_addons --version   # shows where ci_addons was imported from
ci_addons -h | --help # show help on command line
```

CHAPTER 3

Add-ons

Each category is named after a CI worker (e.g AppVeyor) or operating system (e.g Windows) and references add-ons designed to be used on the associated continuous integration service or system.

An add-on is a file that could either directly be executed or used as a parameter for an other tool.

3.1 Anyci

This a special category containing scripts that could be executed on a broad range of CI services.

3.1.1 ctest_junit_formatter

Add-on converting test results from CTest to JUnit format.

The add-on get the name of the latest build tag by reading the first line of <BUILD_DIR>/Testing/TAG, and then convert the file <BUILD_DIR>/Testing/<LATEST_TAG>/Test.xml. The conversion results is outputted on stdout.

This add-on supports both Python 2 and Python 3 and is based on stackoverlow answer contributed by Calvin1602 and MOnsDaR.

Usage:

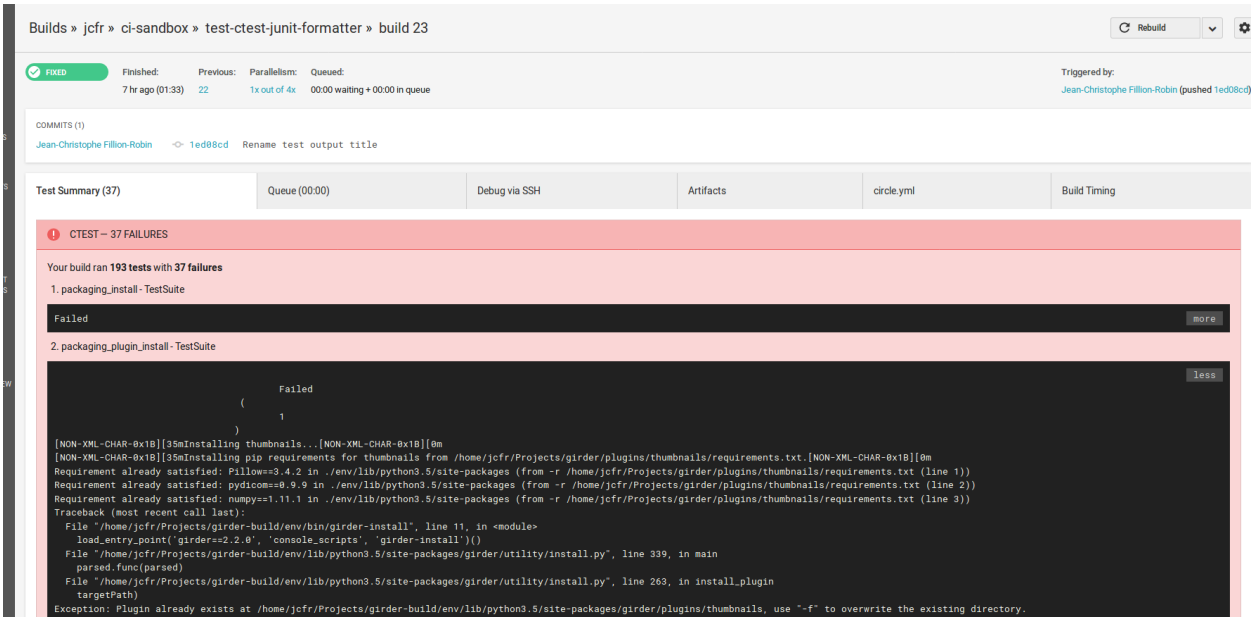
```
ci_addons ctest_junit_formatter BUILD_DIR > JUnit.xml
```

Example of use from CircleCI:

```
$ mkdir ${CIRCLE_TEST_REPORTS}/CTest
$ ci_addons ctest_junit_formatter BUILD_DIR > ${CIRCLE_TEST_REPORTS}/CTest/JUnit-$
↪{CIRCLE_NODE_INDEX}.xml
```

Note: CircleCI will automatically aggregate test results generated by different node.

Example of CircleCI test summary with failing tests:



3.1.2 `docker.py`

Add-on facilitating docker use on CI services.

It allows to load an image from local cache, pull and save back using a convenience one-liner.

Usage:

```
ci_addons docker load-pull-save [-h] [--cache-dir CACHE_DIR] [--verbose]
                                NAME[:TAG|@DIGEST]
```

Example:

```
$ ci_addons docker load-pull-save hello-world:latest
[anyci:docker.py] Loading cached image from /home/jcfr/docker/hello-world-latest.tar
[anyci:docker.py]   -> cached image not found
[anyci:docker.py] Pulling image: hello-world:latest
[anyci:docker.py]   -> done
[anyci:docker.py] Reading image ID from current image
[anyci:docker.py]   -> image ID:␣
↪sha256:c54a2cc56cbb2f04003c1cd4507e118af7c0d340fe7e2720f70976c4b75237dc
[anyci:docker.py] Caching image
[anyci:docker.py]   -> image cached: /home/jcfr/docker/hello-world-latest.tar
[anyci:docker.py] Saving image ID into /home/jcfr/docker/hello-world-latest.image_id
[anyci:docker.py]   -> done
```

Note:

- Image is saved into the cache only if needed.

  In addition to the image archive (e.g *image-name.tar*), a file containing the image ID is also saved into the cache directory (e.g *image-name.image_id*).

This allows to quickly read back the image ID of the cached image and determine if the current image should be saved into the cache.

3.1.3 noop.py

Display name of script and associated argument (basically the value of `sys.argv`).

3.1.4 publish_github_release.py

Add-on automating the creation of GitHub releases.

Based on the git branch found in the current working directory, it allows to automatically create a GitHub `prerelease` and/or `release` and upload associated packages.

Getting Started

To create a pre-release named `latest`:

```
ci_addons publish_github_release --prerelease-packages "dist/*"
```

To create a release named after the current tag:

```
ci_addons publish_github_release --release-packages "dist/*"
```

In both case, packages found in *dist* directory are uploaded.

Note: Pre-releases are created only if the current commit is *NOT* a tag (`latest` tag is automatically ignored). Similarly, releases are created *ONLY* if current commit is a tag (different from `latest`).

Terminology

Prerelease: By default, this corresponds to a GitHub prerelease associated with a tag named `latest` and named `Latest (updated on YYYY-MM-DD HH:MM UTC)`. The prerelease is automatically updated each time the `publish_github_release` script is executed. Updating the `latest` prerelease means that (1) the latest tag is updated to point to the current HEAD, (2) the name is updated and (3) latest packages are uploaded to replace the previous ones. GitHub prerelease are basically release with *draft* option set to False and *prerelease* option set to True.

Release: This corresponds to a GitHub release automatically created by `publish_github_release` script only if it found that HEAD was associated with a tag different from `latest`. It has both *draft* and *prerelease* options set to False. Once packages have been associated with such a release, they are not expected to be removed.

Usage

```
ci_addons publish_github_release [-h]
                                 [--release-packages [PATTERN [PATTERN ...]]]
                                 [--prerelease-packages [PATTERN [PATTERN ...]]]
                                 [--prerelease-packages-clear-pattern PATTERN]
                                 [--prerelease-packages-keep-pattern PATTERN]
                                 [--prerelease-tag PRERELEASE_TAG]
```

(continues on next page)

(continued from previous page)

```
                                    [--prerelease-name PRERELEASE_NAME]
                                    [--prerelease-sha PRERELEASE_SHA]
                                    [--token GITHUB_TOKEN]
                                    [--exit-success-if-missing-token]
                                    [--re-upload]
                                    [--display-python-wheel-platform]
                                    [--dry-run]
                                    ORG/PROJECT
```

Note:

- Packages to upload can be a list of paths or a list of globbing patterns.

Mini-language for packages selection

To facilitate selection of specific packages, if any of the strings described below are found in arguments associated with with either `--prerelease-packages` or `--release-packages`, they will be replaced.

<PYTHON_WHEEL_PLATFORM>: This string is replaced by the current platform as found in python wheel package names (e.g manylinux1, macosx, or win). Executing `ci_addons publish_github_release --display-python-wheel-platform` returns the same string.

<COMMIT_DATE>: This string is replaced by the YYYYMMDD date as returned by `git show -s --format="%ci"`.

<COMMIT_SHORT_SHA>: This string is replaced by the sha as returned by `git rev-parse --short=7 HEAD`.

<COMMIT_DISTANCE>: This string is replaced by the distance to the tag specified using `--prerelease-tag`. If the tag does not exist, it corresponds to the number of commits. This is particularly useful when selecting prerelease packages generated using pep440-pre style implemented in *python-versioneer*.

Use case: Automatic upload of release packages associated with a tag

In this example, the script automatically detects that the current branch HEAD is associated with the tag **1.0.0** and automatically uploads all packages found in the `dist` directory.

```
$ cd PROJECT

$ git describe
1.0.0

$ ci_addons publish_github_release ORG/PROJECT \
  --release-packages "dist/*"
Checking if HEAD is a release tag
Checking if HEAD is a release tag - yes (found 1.0.0: creating release)

created '1.0.0' release
  Tag name      : 1.0.0
  ID            : 5436107
  Created       : 2017-02-13T06:36:29Z
  URL           : https://github.com/ORG/PROJECT/releases/tag/1.0.0
  Author        : USERNAME
```

(continues on next page)

(continued from previous page)

```
  Is published  : True
  Is prerelease : False

uploading '1.0.0' release asset(s) (found 2):
  uploading dist/sandbox-1.0.0-cp27-cp27m-manylinux1.whl
  download_url: https://github.com/ORG/PROJECT/releases/download/1.0.0/sandbox-1.0.0-
↪cp27-cp27m-manylinux1.whl

  uploading dist/sandbox-1.0.0-cp35-cp35m-manylinux1.whl
  download_url: https://github.com/ORG/PROJECT/releases/download/1.0.0/sandbox-1.0.0-
↪cp35-cp35m-manylinux1.whl
```

Use case: Automatic creation of "nightly" prerelease from different build machines

When building projects using continuous integration services (e.g Appveyor, TravicCI, or CircleCI), the *publish_github_release* script has the following responsibilities:

- update the nightly tag reference
- update the release name
- keep only the most recent packages. This means that after successfully uploading package generating on a given platform, the older ones will be removed.

To fulfill its requirements, *publish_github_release* provides two convenient options `--prerelease-packages-clear-pattern` and `--prerelease-packages-keep-pattern`.

prerelease-packages-clear-pattern: This option allows to select all packages that should be removed from the prerelease. For example, on a machine responsible to generate windows python wheels, the following pattern can be used `:"*win*.whl"`.

prerelease-packages-keep-pattern: This option allows to keep packages that have been selected by the previous globbing pattern. For example, assuming development package names contain the date of the commit they are built from, specifying a globbing pattern with the date allows to delete older packages while keeping only the new ones built from that commit.

In the following example, we assume a prerelease done on 2017-02-12 with 16 packages (4 linux, 4 macosx, and 8 windows) already exists. The command reported below corresponds to the execution of the script on a linux machine, after one additional commit has been done the next day.

```
$ cd PROJECT

$ git describe
1.0.0-2-g9d40177

$ commit_date=$(git log -1 --format="%ad" --date=local | date +%Y%m%d)
$ echo $commit_date
20170213

$ ci_addons publish_github_release ORG/PROJECT \
  --prerelease-packages dist/*.dev${commit_date}*manylinux1*.whl \
  --prerelease-packages-clear-pattern "*manylinux1*.whl" \
  --prerelease-packages-keep-pattern "*.dev${commit_date}*.whl"
Checking if HEAD is a release tag
Checking if HEAD is a release tag - no (creating prerelease)

release nightly: already exists
```

(continues on next page)

(continued from previous page)

```
uploading 'nightly' release asset(s) (found 4):
  uploading dist/sandbox-1.0.0.dev20170213-cp27-cp27m-manylinux1_x86_64.whl
  download_url: https://github.com/ORG/PROJECT/releases/download/nightly/sandbox-1.0.
↪0.dev20170213-cp27-cp27m-manylinux1_x86_64.whl

  uploading dist/sandbox-1.0.0.dev20170213-cp34-cp34m-manylinux1_x86_64.whl
  download_url: https://github.com/ORG/PROJECT/releases/download/nightly/sandbox-1.0.
↪0.dev20170213-cp34-cp34m-manylinux1_x86_64.whl

  uploading dist/sandbox-1.0.0.dev20170213-cp35-cp35m-manylinux1_x86_64.whl
  download_url: https://github.com/ORG/PROJECT/releases/download/nightly/sandbox-1.0.
↪0.dev20170213-cp35-cp35m-manylinux1_x86_64.whl

  uploading dist/sandbox-1.0.0.dev20170213-cp36-cp36m-manylinux1_x86_64.whl
  download_url: https://github.com/ORG/PROJECT/releases/download/nightly/sandbox-1.0.
↪0.dev20170213-cp36-cp36m-manylinux1_x86_64.whl

deleting 'nightly' release asset(s) (matched: 8, matched-but-keep: 4, not-matched:␣
↪12):
  deleting sandbox-1.0.0.dev20170212-cp27-cp27m-manylinux1_x86_64.whl
  deleting sandbox-1.0.0.dev20170212-cp34-cp34m-manylinux1_x86_64.whl
  deleting sandbox-1.0.0.dev20170212-cp35-cp35m-manylinux1_x86_64.whl
  deleting sandbox-1.0.0.dev20170212-cp36-cp36m-manylinux1_x86_64.whl
  nothing to delete

resolved 'master' to '9d40177e6d3a69890de8ea359de2d02a943d2e10'
updating 'nightly' release:
  target_commitish: '62fe605938ff252e4ddee05b5209299a1aa9a39e' ->
↪'9d40177e6d3a69890de8ea359de2d02a943d2e10'
  tag_name: 'nightly' -> 'nightly-tmp'

deleting reference refs/tags/nightly
updating 'nightly-tmp' release:
  tag_name: 'nightly-tmp' -> 'nightly'

deleting reference refs/tags/nightly-tmp
updating 'nightly' release:
  target_commitish: '62fe605938ff252e4ddee05b5209299a1aa9a39e' ->
↪'9d40177e6d3a69890de8ea359de2d02a943d2e10'
```

Use case: Automatic creation of GitHub releases and prereleases

This can be done by combining the options `--release-packages` and `--prerelease-packages`.

Note also the use of `--display-python-wheel-platform` to automatically get the current python platform.

For example:

```
$ commit_date=$(git log -1 --format="%ad" --date=local | date +%Y%m%d)

$ platform=$(ci_addons publish_github_release ORG/PROJECT --display-python-wheel-
↪platform)
$ echo $platform
manylinux1
```

(continues on next page)

(continued from previous page)

```
$ ci_addons publish_github_release ORG/PROJECT \
    --release-packages "dist/*" \
    --prerelease-packages dist/*.dev${commit_date}*${platform}*.whl \
    --prerelease-packages-clear-pattern "*${platform}*.whl" \
    --prerelease-packages-keep-pattern "*.dev${commit_date}*.whl"
```

The same can also be achieved across platform using the convenient mini-language for package selection:

```
$ ci_addons publish_github_release ORG/PROJECT \
    --release-packages "dist/*" \
    --prerelease-packages "dist/*.dev<COMMIT_DATE>*<PYTHON_WHEEL_PLATFORM>*.whl" \
    --prerelease-packages-clear-pattern "*<PYTHON_WHEEL_PLATFORM>*.whl" \
    --prerelease-packages-keep-pattern "*.dev<COMMIT_DATE>*.whl"
```

Testing

Since the add-on tests interact with GitHub API, there are not included in the regular scikit-ci-addons collection of tests executed using pytest. Instead, they needs to be manually executed following these steps:

(1) Generate a personal access token with at least `public_repo` scope enabled.

(2) Create a *test* project on GitHub with at least one commit.

(3) Check out sources of your *test* project.

(4) Create a virtual environment, download scikit-ci-addons source code, and install its requirements.

(5) Execute the test script.

For example:

```
export GITHUB_TOKEN=...   # Change this with the token generated above in step (1)
TEST_PROJECT=jcfr/sandbox # Change this with the project name created above in step␣
↪(2)

cd /tmp
git clone https://github.com/scikit-build/scikit-ci-addons
cd scikit-ci-addons/
mkvirtualenv scikit-ci-addons-test
pip install -r requirements.txt
SRC_DIR=$(pwd)

cd /tmp
git clone https://github.com/$TEST_PROJECT test-project
cd test-project

python $SRC_DIR/anyci/tests/test_publish_github_release.py $TEST_PROJECT --no-
↪interactive
```

3.1.5 `run.sh`

Wrapper script executing command and arguments passed as parameters.

3.2 Appveyor

These scripts are designed to work on worker from http://appveyor.com/

3.2.1 enable-worker-remote-access.ps1

Enable access to the build worker via Remote Desktop.

Usage:

```
- ci_addons --install ../
- ps: ../appveyor/enable-worker-remote-access.ps1 [-block|-check_for_block]
```

Example:

```
- ci_addons --install ../
- ps: ../appveyor/enable-worker-remote-access.ps1 -block
```

Note:

- Calling this script will enable and display the Remote Desktop connection details. By default, the connection will be available for the length of the build.
- Specifying `-block` option will ensure the connection remains open for at least 60 mins.
- Specifying `-check_for_block` option will keep the connection open only if the environment variable `BLOCK` has been set to `1`.

3.2.2 install_cmake.py

Download and install in the PATH the specified version of CMake binaries.

Usage:

```
ci_addons appveyor/install_cmake.py X.Y.Z
```

Example:

```
$ ci_addons appveyor/install_cmake.py 3.6.2
```

Note:

- CMake archive is downloaded and extracted into `C:\\cmake-X.Y.Z`. That same directory can then be added to the cache. See Build Cache documentation for more details.
- `C:\\cmake-X.Y.Z` is prepended to the `PATH`. TODO: Is the env global on AppVeyor ? Or does this work only with scikit-ci ?

3.2.3 run-with-visual-studio.cmd

This is a wrapper script setting the Visual Studio environment matching the selected version of Python. This is particularly important when building Python C Extensions.

Usage:

```
ci_addons --install ../
../appveyor/run-with-visual-studio.cmd \\path\\to\\command [arg1 [...]]
```

Example:

```
SET PYTHON_DIR="C:\\Python35"
SET PYTHON_VERSION="3.5.x"
SET PYTHON_ARCH="64"
SET PATH=%PYTHON_DIR%;%PYTHON_DIR%\\Scripts;%PATH%
ci_addons --install ../
../appveyor/run-with-visual-studio.cmd python setup.by bdist_wheel
```

Author:

- Olivier Grisel

License:

- CC0 1.0 Universal

Note:

- Python version selection is done by setting the `PYTHON_VERSION` and `PYTHON_ARCH` environment variables.
- Possible values for `PYTHON_VERSION` are:
  - `"2.7.x"`
  - `"3.4.x"`
  - `"3.5.x"`
- Possible values for `PYTHON_ARCH` are:
  - `"32"`
  - `"64"`

3.2.4 patch_vs2008.py

This script patches the installation of Visual C++ 2008 Express so that it can be used to build 64-bit projects.

Usage:

```
ci_addons appveyor/patch_vs2008.py
```

Credits:

- Xia Wei, sunmast#gmail.com

Links:

- http://www.cppblog.com/xcpp/archive/2009/09/09/vc2008express_64bit_win7sdk.html

---

Note: The add-on download vs2008_patch.zip and execute `setup_x64.bat`.

3.2.5 rolling-build.ps1

Cancel on-going build if there is a newer build queued for the same PR

Usage:

```
- ps: rolling-build.ps1
```

Note:

- If there is a newer build queued for the same PR, cancel this one. The AppVeyor 'rollout builds' option is supposed to serve the same purpose but it is problematic because it tends to cancel builds pushed directly to master instead of just PR builds (or the converse). credits: JuliaLang developers.

3.2.6 tweak_environment.py

Usage:

```
ci_addons tweak_environment.py
```

Note:

- Update `notepad++` settings:
  - `TabSetting.replaceBySpace` set to `yes`

3.3 Circle

These scripts are designed to work on worker from http://circleci.com/

3.3.1 install_cmake.py

Download and install in the PATH the specified version of CMake binaries.

Usage:

```
ci_addons circle/install_cmake.py X.Y.Z
```

Example:

```
$ ci_addons circle/install_cmake.py 3.6.2
```

Note:

- The script will skip the download in two cases:
  - if current version matches the selected one.
  - if archive already exist in `$HOME/downloads` directory.

- Adding directory `$HOME/downloads` to the CircleCI cache can speed up the build. For more details, see Caching Dependencies.

3.4 Travis

These scripts are designed to work on worker from http://travis-ci.org/

3.4.1 install_cmake.py

Download and install in the PATH the specified version of CMake binaries.

Usage:

```
ci_addons appveyor/install_cmake.py X.Y.Z
```

Example:

```
$ ci_addons appveyor/install_cmake.py 3.6.2
```

Note:

- The script automatically detects the operating system (`Linux` or `macOS`) and install CMake in a valid location.
- The archives are downloaded in `$HOME/downloads` to allow caching. See Caching Dependencies and Directories The script on only preforms the download if the correct CMake archive is found in `$HOME/downloads`.
- Linux:
  - Download directory is `/home/travis/downloads`.
  - To support worker with and without `sudo` enabled, CMake is installed in `HOME` (i.e /home/travis). Since `~/bin` is already in the `PATH`, CMake executables will be available in the PATH after running this script.
- macOS:
  - Download directory is `/Users/travis/downloads`.
  - Consider using this script only if the available version does **NOT** work for you. See the Compilers-and-Build-toolchain in Travis documentation.
  - What does this script do ? First, it removes the older version of CMake executable installed in `/usr/local/bin`. Then, it installs the selected version of CMake using `sudo cmake-gui --install`.

3.4.2 install_pyenv.py

Usage:

```
export PYTHON_VERSION=X.Y.Z
ci_addons travis/install_pyenv.py
```

Note:

- Update the version of `pyenv` using `brew`.

- Install the version of python selected setting `PYTHON_VERSION` environment variable.

3.4.3 run-with-pyenv.sh

This is a wrapper script setting the environment corresponding to the version selected setting `PYTHON_VERSION` environment variable.

Usage:

```
export PYTHON_VERSION=X.Y.Z
ci_addons --install ../
../travis/run-with-pyenv.sh python --version
```

3.4.4 enable-worker-remote-access.sh

Enable access to the Travis build worker via netcat.

Prerequisites:

- To make use of this add-on, you first need to:
  1. create an account on https://dashboard.ngrok.com
  2. get the associated token (e.g `xxxxxxxxxxxxxxxxxxxx`)

Usage:

- encrypt the environment variable and associated value using the travis client:

```
travis-cli encrypt NGROK_TOKEN=xxxxxxxxxxxxxxxxxxxx -r org/repo
```

- update `travis.yml`:

```
[...]
env:
 global:
  - secure: "xyz...abc...dev="
  [...]

install:
 - [...]
 - wget https://raw.githubusercontent.com/scikit-build/scikit-ci-addons/master/
↪travis/enable-worker-remote-access.sh -O ../enable-worker-remote-access.sh
 - chmod u+x ../enable-worker-remote-access.sh

script:
 - [...]

after_success:
 - ../enable-worker-remote-access.sh

after_failure:
 - ../enable-worker-remote-access.sh
```

- next time travis build the project it will download ngrok and setup the tunnel. Output should be similar to this one:

```
Executing ngrok
Executing nc
Authtoken saved to configuration file: /Users/travis/.ngrok2/ngrok.yml
INFO[06-05|07:11:10] no configuration paths supplied
INFO[06-05|07:11:10] using configuration at default config path path=/Users/
↪travis/.ngrok2/ngrok.yml
INFO[06-05|07:11:10] open config file                         path=/Users/travis/.
↪ngrok2/ngrok.yml err=nil
DBUG[06-05|07:11:10] init storage                             obj=controller mem␣
↪size=52428800 err=nil
DBUG[06-05|07:11:10] Dialing direct, no proxy                 obj=tunSess
[...]
DBUG[06-05|07:11:10] decoded response                         obj=csess␣
↪id=7d08567ce4a5 clientid=169864eb02eb6fba5f585bb6d27445cf sid=7
resp="&{ClientId:... URL:tcp://0.tcp.ngrok.io:18499 Proto:tcp Opts:map[Addr:0.tcp.
↪ngrok.io:18499] Error: Extra:map[Token:xxxxxxxxxxxxxx]}" err=nil
```

where the url and port allowing to remotely connect are `0.tcp.ngrok.io` and `18499`.

- connection with the worker can be established using netcat. In the example below the command `pwd` and then `ls` are executed:

```
$ nc 0.tcp.ngrok.io 18499
pwd
/Users/travis/build/jcfr/ci-sandbox
ls
LICENSE
README.md
appveyor.yml
circle.yml
images
ngrok
pipe
scripts
```

Note: To easily install the travis client, you could the dockerized version from jcfr/docker-travis-cli. It can easily be installed using:

```
curl https://raw.githubusercontent.com/jcfr/docker-travis-cli/master/travis-cli.sh \
    -o ~/bin/travis-cli && \
chmod +x ~/bin/travis-cli
```

Credits:

- Initial implementation copied from fniephaus/travis-remote-shell
- Support for working with recent version of `netcat` adapted from colesbury/travis-remote-shell and emulating-netcat-e@stackoverflow.

3.5 Windows

These scripts are designed to work on any windows workstation running Windows 7 and above and can be directly used from a powershell terminal (or command line terminal) using a simple one-liner.

Content of the scripts can easily be inspected in the associated source repository.

For example, on a new system without python or git installed, they can be installed from a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-python-36-x64.ps1'))
iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-git.ps1'))
```

Read here to learn about the powershell execution policy.

Details for each `install-*.ps1` scripts are reported below.

3.5.1 install-cmake.ps1

Install selected CMake version in `C:\cmake-X.Y.Z`.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

$cmakeVersion="3.8.1"
iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-cmake.ps1'))
```

Note:

- CMake is **NOT** added to the `PATH`
- setting `$cmakeVersion` to "X.Y.Z" before executing the script allows to select a specific CMake version.
- on AppVeyor, the download and install can be skipped by adding directory `C:\cmake-X.Y.Z` to the `cache`. For more details, see https://www.appveyor.com/docs/build-cache/#configuring-cache-items

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.2 install-flang.ps1

Install latest `flang` in a new conda environment named *flang-env*.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-flang.ps1'))
```

Flang is a Fortran compiler targeting LLVM, it was announced in 2015.

Source code is hosted on GitHub at https://github.com/flang-compiler/flang, the windows fork is hosted as https://github.com/isuruf/flang

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.3 install-git.ps1

Install Git 2.11.0 (including Git Bash) on the system.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-git.ps1'))
```

Note:

- Git executables are added to the `PATH`

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.4 install-miniconda3.ps1

Install latest miniconda3 environment into `C:\Miniconda3`.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-miniconda3.ps1'))
```

Note:

- miniconda environment is **NOT** added to the `PATH` and registry.

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.5 install-ninja.ps1

Install ninja executable v1.7.2 into `C:\ninja-1.7.2`.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-ninja.ps1'))
```

Note:

- ninja executable is **NOT** added to the `PATH`

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.6 install-nsis.ps1

Install NSIS 3.01 on the system.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-nsis.ps1'))
```

Note:

- nsis executable is added to the `PATH`

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.7 install-python.ps1

Install Python 2.7.15, 3.4.4, 3.5.4, 3.6.8, 3.7.2 and 3.8.0a2 (32 and 64-bit) along with pip and virtualenv in the following directories:

```
C:\Python27-x64
C:\Python27-x86

C:\Python34-x64
```

(continues on next page)

(continued from previous page)

```
C:\Python34-x86

C:\Python35-x64
C:\Python35-x86

C:\Python36-x64
C:\Python36-x86

C:\Python37-x64
C:\Python37-x86

C:\Python38-x64
C:\Python38-x86
```

Note:

- python interpreter is **NOT** added to the `PATH`
- setting `$pythonVersion` to either "2.7", "3.4", "3.5", "3.6", "3.7" or "3.8" before executing the script allows to install a specific version. By default, all are installed.
- setting `$pythonArch` to either "86", "32" or "64" before executing the script allows to install python for specific architecture. By default, both are installed. Values "86" and "32" correspond to the same architecture.
- setting `$pythonPrependPath` to 1 will add install and Scripts directories the PATH and .PY to PATHEXT. This variable should be set only if `$pythonVersion` and `$pythonArch` are set. By default, the value is 0.

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

Warning:

- The downloaded versions of python may **NOT** be the latest version including security patches. If running in a production environment (e.g webserver), these versions should be built from source.

3.5.8 install-python-27-x64.ps1

Install Python 2.7 64-bit and update the PATH.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-python-27-x64.ps1'))
```

This is equivalent to:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

$pythonVersion = "2.7"
$pythonArch = "64"
$pythonPrependPath = "1"
iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-python.ps1'))
```

Note:

- `C:\Python27-x64` and `C:\Python27-x64\Scripts` are prepended to the `PATH`

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.9 install-python-36-x64.ps1

Install Python 3.6 64-bit and update the PATH.

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-python-36-x64.ps1'))
```

This is equivalent to:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

$pythonVersion = "3.6"
$pythonArch = "64"
$pythonPrependPath = "1"
iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-python.ps1'))
```

Note:

- `C:\Python36-x64` and `C:\Python36-x64\Scripts` are prepended to the `PATH`

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.10 install-svn.ps1

Install Slik SVN 1.9.5 in the following directory:

```
C:\SlikSvn
```

From a powershell terminal open as administrator:

```
Set-ExecutionPolicy Unrestricted -Force
[System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor 48

iex ((new-object net.webclient).DownloadString('https://raw.githubusercontent.com/
↪scikit-build/scikit-ci-addons/master/windows/install-svn.ps1'))
```

Note:

- svn executable is added to the `PATH`

Note:

- to understand why `SecurityProtocol` is set, see *Addressing "The underlying connection was closed" error*

3.5.11 install-utils.ps1

This script is automatically included (and downloaded if needed) by the other addons, it provides convenience functions useful to download and install programs:

`Always-Download-File($url, $file)`:

Systematically download *$url* into *$file*.

`Download-File($url, $file)`:

If file is not found, download *$url* into *$file*.

`Download-URL($url, $downloadDir)`:

Download *$url* into *$downloadDir*. The filename is extracted from *$url*.

`Install-MSI($fileName, $downloadDir, $targetDir)`:

Programatically install MSI installers *$downloadDir$fileName* into *$targetDir*. The package is installed for all users.

`Which($progName)`

Search for *$progName* in the `PATH` and return its full path.

`Download-7zip($downloadDir)`:

If not found, download 7zip executable `7za.exe` into *$downloadDir*. The function returns the full path to the executable.

`Always-Extract-Zip($filePath, $destDir)`:

Systematically extract zip file *$filePath* into *$destDir* using 7zip. If 7zip executable `7za.exe` is not found in *$downloadDir*, it is downloaded using function `Download-7zip`.

`Extract-Zip($filePath, $destDir)`:

Extract zip file into *$destDir* only if *$destDir* does not exist.

3.5.12 Frequently Asked Questions

Installing add-on from a Windows command line terminal

This can be using the following syntax:

```
@powershell -ExecutionPolicy Unrestricted "iex ((new-object net.webclient).
↪DownloadString('https://raw.githubusercontent.com/scikit-build/scikit-ci-addons/
↪master/windows/install-ninja.ps1'))"
```

Addressing "The underlying connection was closed" error

```
PS C:\Users\dashboard> iex ((new-object net.webclient).DownloadString('https://raw.
↪githubusercontent.com/scikit-build/scikit-ci-addons/master/windows/install-python.
↪ps1'))

Error: 0
Description: The underlying connection was closed: An unexpected error occurred on a␣
↪receive.
```

As explained the chololatey documentation, this most likely happens because the build script is attempting to download from a server that needs to use TLS 1.1 or TLS 1.2 and has restricted the use of TLS 1.0 and SSL v3.

The first things to try is to use the following snippet replacing `https://file/to/download` with the appropriate value:

```
$securityProtocolSettingsOriginal = [System.Net.ServicePointManager]::SecurityProtocol

try {
    # Set TLS 1.2 (3072), then TLS 1.1 (768), then TLS 1.0 (192), finally SSL 3.0 (48)
    # Use integers because the enumeration values for TLS 1.2 and TLS 1.1 won't
    # exist in .NET 4.0, even though they are addressable if .NET 4.5+ is
    # installed (.NET 4.5 is an in-place upgrade).
    [System.Net.ServicePointManager]::SecurityProtocol = 3072 -bor 768 -bor 192 -bor␣
↪48
} catch {
    Write-Warning 'Unable to set PowerShell to use TLS 1.2 and TLS 1.1 due to old .
↪NET Framework installed. If you see underlying connection closed or trust errors,␣
↪you may need to upgrade to .NET Framework 4.5 and PowerShell v3'
}

iex ((new-object net.webclient).DownloadString('https://file/to/download'))

[System.Net.ServicePointManager]::SecurityProtocol = $securityProtocolSettingsOriginal
```

If that does not address the problem, you should update the version of *.NET* installed and install a newer version of PowerShell:

- https://en.wikipedia.org/wiki/.NET_Framework_version_history#Overview
- https://social.technet.microsoft.com/wiki/contents/articles/21016.how-to-install-windows-powershell-4-0.aspx

CHAPTER 4

Contributing

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

4.1 Types of Contributions

You can contribute in many ways:

4.1.1 Report Bugs

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

If you are reporting a bug, please include:

- Any details about your CI 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-ci-addons project could always use more documentation. We welcome help with the official scikit-ci-addons docs, in docstrings, or even on blog posts and articles for the web.

4.1.5 Submit Feedback

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

4.2 Get Started

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

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

```
$ git clone git@github.com:your_name_here/scikit-ci-addons.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 scikit-ci-addons
$ cd scikit-ci-addons/
$ 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
$ python setup.py test
$ tox
```

   If needed, you can get flake8 and tox by using *pip install* to 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 3.4, 3.5, 3.6 and 3.7. Check https://travis-ci.org/scikit-build/scikit-ci-addons/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

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

CHAPTER 6

History

scikit-ci-addons was initially developed in May 2016 by Omar Padron to facilitate the continuous integration of the scikit-build project.

At that time, it consisted of code directly embedded in the CI script used in scikit-build project.

Then, in early September 2016, with the desire to setup cross-platform continuous integration for other project and avoid duplication or maintenance hell, the code was factored out by Jean-Christophe Fillion-Robin into a set of reusable scripts available in the scikit-ci project. By simply cloning the repository, it was possible to more easily enable CI for other projects.

While this was an improvement, this prevented the distribution of standalone and simple scikit-ci package. To better separate concerns and facilitate testing and maintenance, in late September 2016, the scripts were moved into their own project and scikit-ci-addons was born.

Finally, in late October 2016, Jean-Christophe came up with the concept of scikit-ci-addons command line tool allowing to execute the scripts (or add-ons) distributed within the scikit-ci-addons package.

CHAPTER 7

Making a release

A core developer should use the following steps to create a release *X.Y.Z* of **scikit-ci-addons** on PyPI.

7.1 Prerequisites

- All CI tests are passing on AppVeyor, CircleCI and Travis CI.
- You have a GPG signing key.

7.2 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.

7.3 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.

7.4 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/scikit-ci-addons && \
  cd scikit-ci-addons
```

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.

6. Tag the release

```
$ git tag --sign -m "scikit-ci-addons ${release}" ${release} master
```

> **Warning:** We recommend using a GPG signing key to sign the tag.

7. Create the source distribution and wheel

```
$ python setup.py sdist bdist_wheel
```

8. Publish the both release tag and the master branch

```
$ git push origin ${release} && \
  git push origin master
```

9. Upload the distributions on PyPI

```
twine upload dist/*
```

Note: To first upload on TestPyPI , do the following:

```
$ twine upload -r pypitest dist/*
```

10. Create a clean testing environment to test the installation

```
$ mkvirtualenv scikit-ci-addons-${release}-install-test && \
  pip install scikit-ci-addons && \
  ci_addons --list && \
  ci_addons --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.

To install from TestPyPI, do the following:

```
$ pip install -i https://test.pypi.org/simple scikit-ci-addons
```

11. Cleanup

```
$ deactivate  && \
  rm -rf dist/* && \
  rmvirtualenv scikit-ci-addons-${release}-install-test
```

12. Add a `Next Release` section back in *CHANGES.rst*, commit and push local changes.

```
$ git add CHANGES.rst && \
  git commit -m "CHANGES.rst: Add \"Next Release\" section [ci skip]" && \
  git push origin master
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

CHAPTER 9

Resources

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