bandersnatch Documentation

Release 4.4.0

PyPA

Jan 08, 2021

CONTENTS

bandersnatch is a PyPI mirror client according to *PEP 381* https://www.python.org/dev/peps/pep-0381/.

Bandersnatch hits the XMLRPC API of pypi.org to get all packages with serial or packages since the last run's serial. bandersnatch then uses the JSON API of PyPI to get shasums and release file paths to download and workout where to layout the package files on a POSIX file system.

As of 4.0 bandersnatch: - Is fully asyncio based (mainly via aiohttp) - Only stores PEP503 nomalized packages names for the /simple API - Only stores JSON in normailzed package name path too

Contents:

CHAPTER

ONE

INSTALLATION

The following instructions will place the bandersnatch executable in a virtualenv under `bandersnatch/bin/bandersnatch`.

- bandersnatch **requires** `>= Python 3.6`

1.1 pip

This installs the latest stable, released version.

(>= 3.6.1 required)

```
$ python3.6 -m venv bandersnatch
$ bandersnatch/bin/pip install bandersnatch
$ bandersnatch/bin/bandersnatch --help
```

CHAPTER

TWO

MIRROR CONFIGURATION

The mirror configuration settings are in a configuration section of the configuration file named **[mirror]**.

This section contains settings to specify how the mirroring software should operate.

2.1 directory

The mirror directory setting is a string that specifies the directory to store the mirror files.

The directory used must meet the following requirements:

- The filesystem must be case-sensitive filesystem.
- The filesystem must support large numbers of sub-directories.
- The filesystem must support large numbers of files (inodes)

Example:

```
[mirror]
directory = /srv/pypi
```

2.2 json

The mirror json seting is a boolean (true/false) setting that indicates that the json packaging metadata should be mirrored in additon to the packages.

Example:

```
[mirror]
json = false
```

2.3 release-files

The mirror release-files setting is a boolean (true/false) setting that indicates that the package release files should be mirrored. Defaults to `true`. When this option is disabled (via setting to false), you should also specify the `root_uri` configuration. If the uri is empty, it will be set to https://files.pythonhosted.org/.

Example:

```
[mirror]
release-files = true
```

2.4 master

The master setting is a string containing a url of the server which will be mirrored.

The master url string must use https: protocol.

The default value is: https://pypi.org

Example:

```
[mirror]
master = https://pypi.org
```

2.5 timeout

The timeout value is an integer that indicates the maximum number of seconds for web requests.

The default value for this setting is 10 seconds.

Example:

```
[mirror]
timeout = 10
```

2.6 global-timeout

The global-timeout value is an integer that indicates the maximum runtime of individual aiohttp coroutines.

The default value for this setting is 18000 seconds, or 5 hours.

Example:

```
[mirror]
global-timeout = 18000
```

2.7 workers

The workers value is an integer from from 1-10 that indicates the number of concurrent downloads.

The default value is 3.

Recommendations for the workers setting:

- leave the default of 3 to avoid overloading the pypi master
- official servers located in data centers could run 10 workers
- anything beyond 10 is probably unreasonable and is not allowed.

2.8 hash-index

The hash-index is a boolean (true/false) to determine if package hashing should be used.

The Recommended setting: the default of false for full pip/pypi compatibility.

> **Warning:** Package index directory hashing is incompatible with pip, and so this should only be used in an environment where it is behind an application that can translate URIs to filesystem locations.

2.8.1 Apache rewrite rules when using hash-index

When using this setting with an apache server. The apache server will need the following rewrite rules:

```
RewriteRule ^([^/])([^/]*)/$ /mirror/pypi/web/simple/$1/$1$2/
RewriteRule ^([^/])([^/]*)/([^/]+)$/ /mirror/pypi/web/simple/$1/$1$2/$3
```

2.8.2 NGINX rewrite rules when using hash-index

When using this setting with an nginx server. The nginx server will need the following rewrite rules:

```
rewrite ^/simple/([^/])([^/]*)/$ /simple/$1/$1$2/ last;
rewrite ^/simple/([^/])([^/]*)/([^/]+)$/ /simple/$1/$1$2/$3 last;
```

2.9 stop-on-error

The stop-on-error setting is a boolean (true/false) setting that indicates if bandersnatch should stop immediately if it encounters an error.

If this setting is false it will not stop when an error is encountered but it will not mark the sync as successful when the sync is complete.

```
[mirror]
stop-on-error = false
```

2.10 log-config

The log-config setting is a string containing the filename of a python logging configuration file.

Example:

```
[mirror]
log-config = /etc/bandersnatch-log.conf
```

2.11 root_uri

The root_uri is a string containing a uri which is the root added to relative links.

Note: This is generally not necessary, but was added for the official internal PyPI mirror, which requires serving packages from https://files.pythonhosted.org

Example:

```
[mirror]
root_uri = https://example.com
```

2.12 diff-file

The diff file is a string containing the filename to log the files that were downloaded during the mirror. This file can then be used to synchronize external disks or send the files through some other mechanism to offline systems. You can then sync the list of files to an attached drive or ssh destination such as a diode:

```
rsync -av --files-from=/srv/pypi/mirrored-files / /mnt/usb/
```

You can also use this file list as an input to 7zip to create split archives for transfers, allowing you to size the files as you needed:

```
7za a -i@"/srv/pypi/mirrored-files" -spf -v100m path_to_new_zip.7z
```

Example:

```
[mirror]
diff-file = /srv/pypi/mirrored-files
```

2.13 diff-append-epoch

The diff append epoch is a boolean (true/false) setting that indicates if the diff-file should be appended with the current epoch time. This can be used to track diffs over time so the diff file doesn't get cobbered each run. It is only used when diff-file is used.

Example:

```
[mirror]
diff-append-epoch = true
```

CHAPTER

THREE

MIRROR FILTERING

NOTE: All references to whitelist/blacklist are deprecated, and will be replaced with allowlist/blocklist in 5.0

The mirror filter configuration settings are in the same configuration file as the mirror settings. There are different configuration sections for the different plugin types.

Filtering Plugin package lists need to use the **Raw PyPI Name** (non PEP503 normalized) in order to get filtered.

E.g. to Blocklist ACMPlus you'd need to use that *exact* casing in `bandersnatch.conf`

- A PR would be welcome fixing the normalization but it's an invasive PR

3.1 Plugins Enabling

The plugins setting is a list of plugins to enable.

Example (enable all installed filter plugins):

- *Explicitly* enabling plugins is now **mandatory** for *activating plugins*
- They will *do nothing* without activation

Also, enabling will get plugin's defaults if not configured in their respective sections.

```
[plugins]
enabled = all
```

Example (only enable specific plugins):

```
[plugins]
enabled =
    allowlist_project
    blocklist_project
    ...
```

3.2 allowlist / blocklist filtering settings

The blocklist / allowlist settings are in configuration sections named **[blocklist]** and **[allowlist]** these section provides settings to indicate packages, projects and releases that should / should not be mirrored from PyPI.

This is useful to avoid syncing broken or malicious packages.

3.3 packages

The packages setting is a list of python pep440 version specifier of packages to not be mirrored. Enable version specifier filtering for blocklist and allowlist packages through enabling the 'blocklist_release' and 'allowlist_release' plugins, respectively.

Any packages matching the version specifier for blocklist packages will not be downloaded. Any packages not matching the version specifier for allowlist packages will not be downloaded.

Example:

```
[plugins]
enabled =
    blocklist_project
    blocklist_release
    allowlist_project
    allowlist_release

[blocklist]
packages =
    example1
    example2>=1.4.2,<1.9,!=1.5.*,!=1.6.*

[allowlist]
packages =
    black==18.5
    ptr
```

3.4 Metadata Filtering

Packages and release files may be selected by filtering on specific metadata value.

General form of configuration entries is:

```
[filter_some_metadata]
tag:tag:path.to.object =
    matcha
    matchb
```

3.5 requirements files Filtering

Packages and releases might be given as requirements.txt files

if requirements_path is missing it is assumed to be system root folder ('/')

```
[plugins]
enabled =
    project_requirements
    project_requirements_pinned
[allowlist]
requirements_path = /my_folder
requirements =
    requirements.txt
```

3.5.1 Project Regex Matching

Filter projects to be synced based on regex matches against their raw metadata entries straight from parsed downloaded json.

Example:

```
[regex_project_metadata]
not-null:info.classifiers =
        .*Programming Language :: Python :: 2.*
```

Valid tags are `all`,`any`,`none`,`match-null`,`not-null`, with default of `any:match-null`

All metadata provided by json is available, including `info`, `last_serial`, `releases`, etc. headings.

3.5.2 Release File Regex Matching

Filter release files to be downloaded for projects based on regex matches against the stored metadata entries for each release file.

Example:

```
[regex_release_file_metadata]
any:release_file.packagetype =
    sdist
    bdist_wheel
```

Valid tags are the same as for projects.

Metadata available to match consists of `info`, `release`, and `release_file` top level structures, with `info` containing the package-wide inthe fo, `release` containing the version of the release and `release_file` the metadata for an individual file for that release.

3.6 Prerelease filtering

Bandersnatch includes a plugin to filter our pre-releases of packages. To enable this plugin simply add `prerelease_release` to the enabled plugins list.

```
[plugins]
enabled =
    prerelease_release
```

3.7 Regex filtering

Advanced users who would like finer control over which packages and releases to filter can use the regex Bandersnatch plugin.

This plugin allows arbitrary regular expressions to be defined in the configuration, any package name or release version that matches will *not* be downloaded.

The plugin can be activated for packages and releases separately. For example to activate the project regex filter simply add it to the configuration as before:

```
[plugins]
enabled =
    regex_project
```

If you'd like to filter releases using the regex filter use `regex_release` instead.

The regex plugin requires an extra section in the config to define the actual patterns to used for filtering:

```
[filter_regex]
packages =
    .+-evil$
releases =
    .+alpha\d$
```

Note the same `filter_regex` section may include a `packages` and a `releases` entry with any number of regular expressions.

3.8 Platform-specific binaries filtering

This filter allows advanced users not interesting in Windows/macOS/Linux specific binaries to not mirror the corresponding files.

```
[plugins]
enabled =
    exclude_platform
[blocklist]
platforms =
    windows
```

Available platforms are: `windows` `macos` `freebsd` `linux`.

3.9 Keep only latest releases

You can also keep only the latest releases based on greatest Version numbers.

```
[plugins]
enabled =
    latest_release

[latest_release]
keep = 3
```

By default, the plugin does not filter out any release. You have to add the `keep` setting.

You should be aware that it can break requirements. Prereleases are also kept.

CHAPTER

FOUR

CONTRIBUTING

So you want to help out? **Awesome**. Go you!

4.1 Code of Conduct

Everyone interacting in the bandersnatch project's codebases, issue trackers, chat rooms, and mailing lists is expected to follow the PSF Code of Conduct.

4.2 Getting Started

Bandersnatch is developed using the GitHub Flow

4.2.1 Pre Install

Please make sure you system has the following:

- Python 3.6.1 or greater

4.2.2 Development venv

One way to develop and install all the dependencies of bandersnatch is to use a venv.

- First create one and upgrade `pip`

```
python3.6 -m venv /path/to/venv
/path/to/venv/bin/pip install --upgrade pip
```

For example:

```
$ python3.6 -m venv bandersnatchvenv
$ bandersnatchvenv/bin/pip install --upgrade pip
Collecting pip
  Using cached https://files.pythonhosted.org/packages/0f/74/
↪ecd13431bcc456ed390b44c8a6e917c1820365cbebcb6a8974d1cd045ab4/pip-10.0.1-py2.py3-
↪none-any.whl
Installing collected packages: pip
  Found existing installation: pip 9.0.3
    Uninstalling pip-9.0.3:
```

(continues on next page)

(continued from previous page)

```
      Successfully uninstalled pip-9.0.3
Successfully installed pip-10.0.1
```

- Then install the dependencies to the venv:

```
/path/to/venv/bin/pip install -r requirements.txt -r test-requirements.txt
```

For example:

```
$ bandersnatchvenv/bin/pip install -r requirements.txt -r test-requirements.txt
Collecting six==1.10.0 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/c8/0a/
↪b6723e1bc4c516cb687841499455a8505b44607ab535be01091c0f24f079/six-1.10.0-py2.py3-
↪none-any.whl
Collecting pyparsing==2.1.10 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/2b/f7/
↪e5a178fc3ea4118a0edce2a8d51fc14e680c745cf4162e4285b437c43c94/pyparsing-2.1.10-py2.
↪py3-none-any.whl (56kB)
    100% || 61kB 2.3MB/s
Collecting python-dateutil==2.6.0 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/40/8b/
↪275015d7a9ec293cf1bbf55433258fbc9d0711890a7f6dc538bac7b86bce/python_dateutil-2.6.0-
↪py2.py3-none-any.whl (194kB)
    100% || 194kB 1.3MB/s
Collecting packaging==16.8 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/87/1b/
↪c39b7c65b5612812b83d6cab7ef2885eac9f6beb0b7b8a7071a186aea3b1/packaging-16.8-py2.py3-
↪none-any.whl
Collecting requests==2.12.4 (from -r requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/ed/9e/
↪60cc074968c095f728f0d8d28370e8d396fa60afb7582735563cccf223dd/requests-2.12.4-py2.
↪py3-none-any.whl (576kB)
    100% || 583kB 3.2MB/s
Collecting xmlrpc2==0.3.1 (from -r requirements.txt (line 7))
Collecting bandersnatch==2.1.3 (from -r requirements.txt (line 8))
  Downloading https://files.pythonhosted.org/packages/25/41/
↪9082fcbf20ff536f990e538957eed7474d78b9dcecd018530684ae058995/bandersnatch-2.1.3-py3-
↪none-any.whl
Collecting flake8 (from -r test-requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/b9/dc/
↪14e9d94c770b8c4ef584e906c7583e74864786a58d47de101f2767d50c0b/flake8-3.5.0-py2.py3-
↪none-any.whl (69kB)
    100% || 71kB 4.8MB/s
Collecting pep8 (from -r test-requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/42/3f/
↪669429ce58de2c22d8d2c542752e137ec4b9885fff398d3eceb1a7f5acb4/pep8-1.7.1-py2.py3-
↪none-any.whl (41kB)
    100% || 51kB 9.6MB/s
Collecting pytest (from -r test-requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/76/52/
↪fc48d02492d9e6070cb672d9133382e83084f567f88eff1c27bd2c6c27a8/pytest-3.5.1-py2.py3-
↪none-any.whl (192kB)
    100% || 194kB 2.8MB/s
Collecting pytest-codecheckers (from -r test-requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/53/09/
↪263669db13955496e77017f389693c1e1dd77d98fd4afd51b133162e858f/pytest-codecheckers-0.
↪2.tar.gz
```

(continues on next page)

(continued from previous page)

```
Collecting pytest-cov (from -r test-requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/30/7d/
↪7f6a78ae44a1248ee28cc777586c18b28a1df903470e5d34a6e25712b8aa/pytest_cov-2.5.1-py2.
↪py3-none-any.whl
Collecting pytest-timeout (from -r test-requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/69/7f/
↪33a67c2494c6c337daca935192b7de09d30b54e568c981ed0681380393c4/pytest_timeout-1.2.1-
↪py2.py3-none-any.whl
Collecting pytest-cache (from -r test-requirements.txt (line 7))
  Downloading https://files.pythonhosted.org/packages/d1/15/
↪082fd0428aab33d2bafa014f3beb241830427ba803a8912a5aaeaf3a5663/pytest-cache-1.0.tar.gz
Requirement already satisfied: setuptools in /private/tmp/bandersnatchvenv/lib/
↪python3.6/site-packages (from -r test-requirements.txt (line 8)) (39.0.1)
Collecting tox (from -r test-requirements.txt (line 9))
  Downloading https://files.pythonhosted.org/packages/e6/41/
↪4dcfd713282bf3213b0384320fa8841e4db032ddcb80bc08a540159d42a8/tox-3.0.0-py2.py3-none-
↪any.whl (60kB)
    100% || 61kB 2.2MB/s
Collecting pycodestyle<2.4.0,>=2.0.0 (from flake8->-r test-requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/e4/81/
↪78fe51eb4038d1388b7217dd63770b0f428370207125047312886c923b26/pycodestyle-2.3.1-py2.
↪py3-none-any.whl (45kB)
    100% || 51kB 4.4MB/s
Collecting mccabe<0.7.0,>=0.6.0 (from flake8->-r test-requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/87/89/
↪479dc97e18549e21354893e4ee4ef36db1d237534982482c3681ee6e7b57/mccabe-0.6.1-py2.py3-
↪none-any.whl
Collecting pyflakes<1.7.0,>=1.5.0 (from flake8->-r test-requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/d7/40/
↪733bcc64da3161ae4122c11e88269f276358ca29335468005cb0ee538665/pyflakes-1.6.0-py2.py3-
↪none-any.whl (227kB)
    100% || 235kB 2.6MB/s
Collecting py>=1.5.0 (from pytest->-r test-requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/67/a5/
↪f77982214dd4c8fd104b066f249adea2c49e25e8703d284382eb5e9ab35a/py-1.5.3-py2.py3-none-
↪any.whl (84kB)
    100% || 92kB 3.8MB/s
Collecting pluggy<0.7,>=0.5 (from pytest->-r test-requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/ba/65/
↪ded3bc40bbf8d887f262f150fbe1ae6637765b5c9534bd55690ed2c0b0f7/pluggy-0.6.0-py3-none-
↪any.whl
Collecting more-itertools>=4.0.0 (from pytest->-r test-requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/7a/46/
↪886917c6a4ce49dd3fff250c01c5abac5390d57992751384fe61befc4877/more_itertools-4.1.0-
↪py3-none-any.whl (47kB)
    100% || 51kB 3.9MB/s
Collecting attrs>=17.4.0 (from pytest->-r test-requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/41/59/
↪cedf87e91ed541be7957c501a92102f9cc6363c623a7666d69d51c78ac5b/attrs-18.1.0-py2.py3-
↪none-any.whl
Collecting coverage>=3.7.1 (from pytest-cov->-r test-requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/a3/7e/
↪c94c21d643bfe7017615994df7b52292a33c8dcf36a6f694af110594edba/coverage-4.5.1-cp36-
↪cp36m-macosx_10_12_x86_64.whl (178kB)
    100% || 184kB 3.3MB/s
Collecting execnet>=1.1.dev1 (from pytest-cache->-r test-requirements.txt (line 7))
  Downloading https://files.pythonhosted.org/packages/f9/76/
↪3343e69a2a1602052f587898934e5fea395d22310d39c07955596597227c/execnet-1.5.0-py2.py3-
↪none-any.whl
```

(continues on next page)

(continued from previous page)

```
Collecting virtualenv>=1.11.2 (from tox->-r test-requirements.txt (line 9))
  Downloading https://files.pythonhosted.org/packages/ed/ea/
↪e20b5cbebf45d3096e8138ab74eda139595d827677f38e9dd543e6015bdf/virtualenv-15.2.0-py2.
↪py3-none-any.whl (2.6MB)
    100% || 2.6MB 3.3MB/s
Collecting apipkg>=1.4 (from execnet>=1.1.dev1->pytest-cache->-r test-requirements.
↪txt (line 7))
  Downloading https://files.pythonhosted.org/packages/94/72/
↪fd4f2e46ce7b0d388191c819ef691c8195fab09602bbf1a2f92aa5351444/apipkg-1.4-py2.py3-
↪none-any.whl
Installing collected packages: six, pyparsing, python-dateutil, packaging, requests,␣
↪xmlrpc2, bandersnatch, pycodestyle, mccabe, pyflakes, flake8, pep8, py, pluggy,␣
↪more-itertools, attrs, pytest, pytest-codecheckers, coverage, pytest-cov, pytest-
↪timeout, apipkg, execnet, pytest-cache, virtualenv, tox
  Running setup.py install for pytest-codecheckers ... done
  Running setup.py install for pytest-cache ... done
Successfully installed apipkg-1.4 attrs-18.1.0 bandersnatch-2.1.3 coverage-4.5.1␣
↪execnet-1.5.0 flake8-3.5.0 mccabe-0.6.1 more-itertools-4.1.0 packaging-16.8 pep8-1.
↪7.1 pluggy-0.6.0 py-1.5.3 pycodestyle-2.3.1 pyflakes-1.6.0 pyparsing-2.1.10 pytest-
↪3.5.1 pytest-cache-1.0 pytest-codecheckers-0.2 pytest-cov-2.5.1 pytest-timeout-1.2.
↪1 python-dateutil-2.6.0 requests-2.12.4 six-1.10.0 tox-3.0.0 virtualenv-15.2.0␣
↪xmlrpc2-0.3.1
```

4.3 Running Bandersnatch

You will need to customize `src/bandersnatch/default.conf` and run via the following:

WARNING: Bandersnatch will go off and sync from pypi.org and use large amounts of disk space!

```
cd bandersnatch
/path/to/venv/bin/pip install --upgrade .
/path/to/venv/bin/bandersnatch -c src/bandersnatch/default.conf mirror
```

4.4 Running Unit Tests

We use tox to run tests. `tox.ini` has the options needed, so running tests is very easy.

```
cd bandersnatch
/path/to/venv/bin/tox [-vv]
```

For example:

```
$ tox
GLOB sdist-make: /Users/dhubbard/PycharmProjects/bandersnatch/setup.py
py36 create: /Users/dhubbard/PycharmProjects/bandersnatch/.tox/py36
py36 installdeps: -rtest-requirements.txt
py36 inst: /Users/dhubbard/PycharmProjects/bandersnatch/.tox/dist/bandersnatch-2.2.1.
↪zip
py36 installed: apipkg==1.4,attrs==18.1.0,bandersnatch==2.2.1,certifi==2018.4.16,
↪chardet==3.0.4,coverage==4.5.1,execnet==1.5.0,flake8==3.5.0,idna==2.6,mccabe==0.6.1,
↪more-itertools==4.1.0,packaging==17.1,pep8==1.7.1,pluggy==0.6.0,py==1.5.3,
↪pycodestyle==2.3.1,pyflakes==1.6.0,pyparsing==2.2.0,pytest==3.5.1,pytest-cache==1.0,
↪pytest-codecheckers==0.2,pytest-cov==2.5.1,pytest-timeout==1.2.1,python-dateutil==2.
↪7.3,requests==2.18.4,six==1.11.0,tox==3.0.0,urllib3==1.22,virtualenv==15.2.0,
↪xmlrpc2==0.3.1
```

(continues on next page)

(continued from previous page)

```
py36 runtests: PYTHONHASHSEED='42669967'
py36 runtests: commands[0] | pytest
============================================================================================================
↪test session starts␣
↪============================================================================================================
platform darwin -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /Users/dhubbard/PycharmProjects/bandersnatch, inifile: pytest.ini
plugins: timeout-1.2.1, cov-2.5.1, codecheckers-0.2
timeout: 10.0s method: signal
collected 94 items

src/bandersnatch/__init__.py ..                                                                          ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [  2%]
src/bandersnatch/buildout.py ..                                                                          ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [  4%]
src/bandersnatch/log.py ..                                                                               ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [  6%]
src/bandersnatch/main.py ..                                                                              ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [  8%]
src/bandersnatch/master.py ..                                                                            ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [ 10%]
src/bandersnatch/mirror.py ..                                                                            ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [ 12%]
src/bandersnatch/package.py ..                                                                           ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [ 14%]
src/bandersnatch/release.py ..                                                                           ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [ 17%]
src/bandersnatch/utils.py ..                                                                             ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [ 19%]
src/bandersnatch/tests/conftest.py ..                                                                    ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [ 21%]
src/bandersnatch/tests/test_main.py .......                                                              ␣
↪                                                                                                        ␣
↪                                                                                                        ␣
↪  [ 28%]
src/bandersnatch/tests/test_master.py ..........                                                         ␣
↪                                                                                                        ␣
↪  [ 40%]
```

(continues on next page)

(continued from previous page)

```
src/bandersnatch/tests/test_mirror.py ..................
↪
↪
↪  [ 61%]
src/bandersnatch/tests/test_package.py ..........................
↪
↪
↪  [ 93%]
src/bandersnatch/tests/test_utils.py ......
↪
↪
↪  [100%]

---------- coverage: platform darwin, python 3.6.5-final-0 -----------
Coverage HTML written to dir htmlcov


================================================================================================
↪94 passed in 3.40 seconds
↪================================================================================================
_____________________________________________
↪_____________________________________ summary _____________________________________
↪_____________________________________________
↪__________
  py36: commands succeeded
  congratulations :)
```

You want to see:

```
py36: commands succeeded
congratulations :)
```

4.5 Making a release

To be completed - @cooper has never used zc.buildout

CHAPTER

FIVE

BANDERSNATCH

5.1 bandersnatch package

5.1.1 Package contents

5.1.2 Submodules

5.1.3 bandersnatch.configuration module

Module containing classes to access the bandersnatch configuration file

class bandersnatch.configuration.**BandersnatchConfig**(**args: Any*, ***kwargs: Any*)

Bases: object

SHOWN_DEPRECATIONS = False

check_for_deprecations() → None

load_configuration() → None

Read the configuration from a configuration file

class bandersnatch.configuration.**SetConfigValues**(*json_save*, *root_uri*, *diff_file_path*, *diff_append_epoch*, *digest_name*, *storage_backend_name*, *cleanup*, *release_files_save*)

Bases: tuple

property cleanup

Alias for field number 6

property diff_append_epoch

Alias for field number 3

property diff_file_path

Alias for field number 2

property digest_name

Alias for field number 4

property json_save

Alias for field number 0

property release_files_save

Alias for field number 7

property root_uri

Alias for field number 1

property storage_backend_name
Alias for field number 5

class bandersnatch.configuration.**Singleton**
Bases: type

bandersnatch.configuration.**validate_config_values**(*config: configparser.ConfigParser*) → *bandersnatch.configuration.SetConfigValues*

5.1.4 bandersnatch.delete module

async bandersnatch.delete.**delete_packages**(*config: configparser.ConfigParser*, *args: argparse.Namespace*, *master:* bandersnatch.master.Master) → int

bandersnatch.delete.**delete_path**(*blob_path: pathlib.Path*, *dry_run: bool* = *False*) → int

5.1.5 bandersnatch.filter module

Blocklist management

class bandersnatch.filter.**Filter**(**args: Any*, ***kwargs: Any*)
Bases: object

Base Filter class

property allowlist

property blocklist

check_match(***kwargs: Any*) → bool
Check if the plugin matches based on the arguments provides.

Returns True if the values match a filter rule, False otherwise

Return type bool

deprecated_name: str = ''

filter(*metadata: dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initialize_plugin() → None
Code to initialize the plugin

name = 'filter'

class bandersnatch.filter.**FilterMetadataPlugin**(**args: Any*, ***kwargs: Any*)
Bases: *bandersnatch.filter.Filter*

Plugin that blocks sync operations for an entire project based on info fields.

name = 'metadata_plugin'

class bandersnatch.filter.**FilterProjectPlugin**(**args: Any*, ***kwargs: Any*)
Bases: *bandersnatch.filter.Filter*

Plugin that blocks sync operations for an entire project

name = 'project_plugin'

class bandersnatch.filter.**FilterReleaseFilePlugin**(**args: Any*, ***kwargs: Any*)
Bases: *bandersnatch.filter.Filter*

Plugin that modify the download of specific release or dist files

name = 'release_file_plugin'

class bandersnatch.filter.**FilterReleasePlugin**(**args: Any*, ***kwargs: Any*)
Bases: *bandersnatch.filter.Filter*

Plugin that modifies the download of specific releases or dist files

name = 'release_plugin'

class bandersnatch.filter.**LoadedFilters**(*load_all: bool = False*)
Bases: object

A class to load all of the filters enabled

ENTRYPOINT_GROUPS = ['bandersnatch_filter_plugins.v2.project', 'bandersnatch_filter_pl

filter_metadata_plugins() → List[*bandersnatch.filter.Filter*]
Load and return the metadata filtering plugin objects

Returns List of objects derived from the bandersnatch.filter.Filter class

Return type list of bandersnatch.filter.Filter

filter_project_plugins() → List[*bandersnatch.filter.Filter*]
Load and return the project filtering plugin objects

Returns List of objects derived from the bandersnatch.filter.Filter class

Return type list of bandersnatch.filter.Filter

filter_release_file_plugins() → List[*bandersnatch.filter.Filter*]
Load and return the release file filtering plugin objects

Returns List of objects derived from the bandersnatch.filter.Filter class

Return type list of bandersnatch.filter.Filter

filter_release_plugins() → List[*bandersnatch.filter.Filter*]
Load and return the release filtering plugin objects

Returns List of objects derived from the bandersnatch.filter.Filter class

Return type list of bandersnatch.filter.Filter

5.1.6 bandersnatch.log module

bandersnatch.log.**setup_logging**(*args: Any*) → logging.StreamHandler

5.1.7 bandersnatch.main module

async bandersnatch.main.**async_main**(*args: argparse.Namespace*, *config: configparser.ConfigParser*) → int

bandersnatch.main.**main**(*loop: Optional[asyncio.events.AbstractEventLoop] = None*) → int

5.1.8 bandersnatch.master module

class bandersnatch.master.**Master**(*url: str*, *timeout: float = 10.0*, *global_timeout: Optional[float] = 18000.0*)

Bases: object

async all_packages() → Dict[str, int]

async changed_packages(*last_serial: int*) → Dict[str, int]

async check_for_stale_cache(*path: str*, *required_serial: Optional[int]*, *got_serial: Optional[int]*) → None

get(*path: str*, *required_serial: Optional[int]*, ***kw: Any*) → AsyncGenerator[aiohttp.client_reqrep.ClientResponse, None]

async get_package_metadata(*package_name: str*, *serial: int = 0*) → Any

async rpc(*method_name: str*, *serial: int = 0*) → Any

async url_fetch(*url: str*, *file_path: pathlib.Path*, *executor: Optional[Union[concurrent.futures.process.ProcessPoolExecutor, concurrent.futures.thread.ThreadPoolExecutor]] = None*, *chunk_size: int = 65536*) → None

property xmlrpc_url

exception bandersnatch.master.**StalePage**

Bases: Exception

We got a page back from PyPI that doesn't meet our expected serial.

exception bandersnatch.master.**XmlRpcError**

Bases: aiohttp.client_exceptions.ClientError

Issue getting package listing from PyPI Repository

5.1.9 bandersnatch.mirror module

class bandersnatch.mirror.**BandersnatchMirror**(*homedir: pathlib.Path*, *master: bandersnatch.master.Master*, *storage_backend: Optional[str] = None*, *stop_on_error: bool = False*, *workers: int = 3*, *hash_index: bool = False*, *json_save: bool = False*, *digest_name: Optional[str] = None*, *root_uri: Optional[str] = None*, *keep_index_versions: int = 0*, *diff_file: Optional[Union[pathlib.Path, str]] = None*, *diff_append_epoch: bool = False*, *diff_full_path: Optional[Union[pathlib.Path, str]] = None*, *flock_timeout: int = 1*, *diff_file_list: Optional[List] = None*, ***, *cleanup: bool = False*, *release_files_save: bool = True*)

Bases: *bandersnatch.mirror.Mirror*

async cleanup_non_pep_503_paths(*package:* bandersnatch.package.Package) → None
Before 4.0 we use to store backwards compatible named dirs for older pip This function checks for them and cleans them up

async determine_packages_to_sync() → None
Update the self.packages_to_sync to contain packages that need to be synced.

async download_file(*url: str*, *sha256sum: str*, *chunk_size: int = 65536*) → Optional[pathlib.Path]

errors = False

finalize_sync() → None

find_package_indexes_in_dir(*simple_dir: pathlib.Path*) → List[str]
Given a directory that contains simple packages indexes, return a sorted list of normalized package names. This presumes every directory within is a simple package index directory.

find_target_serial() → int

gen_data_requires_python(*release: Dict*) → str

generate_simple_page(*package:* bandersnatch.package.Package) → str

property generationfile

get_simple_dirs(*simple_dir: pathlib.Path*) → List[pathlib.Path]
Return a list of simple index directories that should be searched for package indexes when compiling the main index page.

json_file(*package_name: str*) → pathlib.Path

json_pypi_symlink(*package_name: str*) → pathlib.Path

need_index_sync = True

need_wrapup = False

on_error(*exception: BaseException*, ***kwargs: Dict*) → None

async process_package(*package:* bandersnatch.package.Package) → None

record_finished_package(*name: str*) → None

save_json_metadata(*package_info: Dict*, *name: str*) → bool
Take the JSON metadata we just fetched and save to disk

simple_directory(*package:* bandersnatch.package.Package) → pathlib.Path

property statusfile

sync_index_page() → None

async sync_release_files(*package:* bandersnatch.package.Package) → None
Purge + download files returning files removed + added

sync_simple_page(*package:* bandersnatch.package.Package) → None

property todolist

property webdir

wrapup_successful_sync() → None

class bandersnatch.mirror.**Mirror**(*master:* bandersnatch.master.Master, *workers: int = 3*)
Bases: object

async determine_packages_to_sync() → None
Update the self.packages_to_sync to contain packages that need to be synced.

finalize_sync() → None

now = None

on_error(*exception: BaseException*, ***kwargs: Dict*) → None

async package_syncer(*idx: int*) → None

packages_to_sync: Dict[str, Union[int, str]] = {}

async process_package(*package:* bandersnatch.package.Package) → None

async sync_packages() → None

synced_serial: Optional[int] = 0

async synchronize(*specific_packages: Optional[List[str]] = None*) → Dict[str, Set[str]]

target_serial: Optional[int] = None

async bandersnatch.mirror.**mirror**(*config: configparser.ConfigParser*, *specific_packages: Optional[List[str]] = None*) → int

5.1.10 bandersnatch.package module

class bandersnatch.package.**Package**(*name: str*, *serial: int = 0*)
Bases: object

filter_all_releases(*release_filters: List[Filter]*) → bool
Filter releases and removes releases that fail the filters

filter_all_releases_files(*release_file_filters: List[Filter]*) → bool
Filter release files and remove empty releases after doing so.

filter_metadata(*metadata_filters: List[Filter]*) → bool
Run the metadata filtering plugins

property info

property last_serial

property metadata

property release_files

property releases

async update_metadata(*master: Master*, *attempts: int = 3*) → None

5.1.11 bandersnatch.storage module

Storage management

class bandersnatch.storage.**Storage**(**args: Any*, *config: Optional[configparser.ConfigParser] = None*, ***kwargs: Any*)

Bases: object

Base Storage class

PATH_BACKEND
alias of pathlib.Path

static canonicalize_package(*name: str*) → str

compare_files(*file1: Union[pathlib.Path, str]*, *file2: Union[pathlib.Path, str]*) → bool
Compare two files and determine whether they contain the same data. Return True if they match

copy_file(*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*) → None
Copy a file from **source** to **dest**

delete(*path: Union[pathlib.Path, str]*, *dry_run: bool = False*) → int
Delete the provided path.

delete_file(*path: Union[pathlib.Path, str]*, *dry_run: bool = False*) → int
Delete the provided path, recursively if necessary.

property directory

exists(*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path exists

find(*root: Union[pathlib.Path, str]*, *dirs: bool = True*) → str
A test helper simulating 'find'.

Iterates over directories and filenames, given as relative paths to the root.

get_flock_path() → Union[pathlib.Path, str]

get_hash(*path: Union[pathlib.Path, str]*, *function: str = 'sha256'*) → str
Get the sha256sum of a given **path**

get_json_paths(*name: str*) → Sequence[Union[pathlib.Path, str]]

get_lock(*path: str*) → filelock.BaseFileLock
Retrieve the appropriate *FileLock* backend for this storage plugin

Parameters **path** (*str*) – The path to use for locking

Returns A *FileLock* backend for obtaining locks

Return type filelock.BaseFileLock

hash_file(*path: Union[pathlib.Path, str]*, *function: str = 'sha256'*) → str

initialize_plugin() → None
Code to initialize the plugin

is_dir (*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path is a directory.

is_file (*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path is a file.

iter_dir (*path: Union[pathlib.Path, str]*) → Generator[Union[pathlib.Path, str], None, None]
Iterate over the path, returning the sub-paths

mkdir (*path: Union[pathlib.Path, str]*, *exist_ok: bool = False*, *parents: bool = False*) → None
Create the provided directory

name = 'storage'

open_file (*path: Union[pathlib.Path, str]*, *text: bool = True*) → Generator[IO, None, None]
Yield a file context to iterate over. If text is true, open the file with 'rb' mode specified.

read_file (*path: Union[pathlib.Path, str]*, *text: bool = True*, *encoding: str = 'utf-8'*, *errors: Optional[str] = None*) → Union[str, bytes]
Yield a file context to iterate over. If text is true, open the file with 'rb' mode specified.

rewrite (*filepath: Union[pathlib.Path, str]*, *mode: str = 'w'*, ***kw: Any*) → Generator[IO, None, None]
Rewrite an existing file atomically to avoid programs running in parallel to have race conditions while reading.

rmdir (*path: Union[pathlib.Path, str]*, *recurse: bool = False*, *force: bool = False*, *ignore_errors: bool = False*, *dry_run: bool = False*) → int
Remove the directory. If recurse is True, allow removing empty children. If force is true, remove contents destructively.

symlink (*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*) → None
Create a symlink at **dest** that points back at **source**

update_safe (*filename: Union[pathlib.Path, str]*, ***kw: Any*) → Generator[IO, None, None]
Rewrite a file atomically.

Clients are allowed to delete the tmpfile to signal that they don't want to have it updated.

write_file (*path: Union[pathlib.Path, str]*, *contents: Union[str, bytes]*) → None
Write data to the provided path. If **contents** is a string, the file will be opened and written in "r" + "utf-8" mode, if bytes are supplied it will be accessed using "rb" mode (i.e. binary write).

class bandersnatch.storage.**StoragePlugin** (**args: Any*, *config: Optional[configparser.ConfigParser] = None*, ***kwargs: Any*)

Bases: *bandersnatch.storage.Storage*

Plugin that provides a storage backend for bandersnatch

flock_path: Union[pathlib.Path, str]

name = 'storage_plugin'

bandersnatch.storage.**load_storage_plugins** (*entrypoint_group: str*, *enabled_plugin: Optional[str] = None*, *config: Optional[configparser.ConfigParser] = None*, *clear_cache: bool = False*) → Set[*bandersnatch.storage.Storage*]

Load all storage plugins that are registered with pkg_resources

Parameters

- **entrypoint_group** (*str*) – The entrypoint group name to load plugins from
- **enabled_plugin** (*str*) – The optional enabled storage plugin to search for

- **config** (*configparser.ConfigParser*) – The optional configparser instance to pass in
- **clear_cache** (*bool*) – Whether to clear the plugin cache

Returns A list of objects derived from the Storage class

Return type List of Storage

bandersnatch.storage.**storage_backend_plugins**(*backend: Optional[str] = 'filesystem'*, *config: Optional[configparser.ConfigParser] = None*, *clear_cache: bool = False*) → Iterable[*bandersnatch.storage.Storage*]

Load and return the release filtering plugin objects

Parameters

- **backend** (*str*) – The optional enabled storage plugin to search for
- **config** (*configparser.ConfigParser*) – The optional configparser instance to pass in
- **clear_cache** (*bool*) – Whether to clear the plugin cache

Returns List of objects derived from the bandersnatch.storage.Storage class

Return type list of bandersnatch.storage.Storage

5.1.12 bandersnatch.utils module

bandersnatch.utils.**bandersnatch_safe_name**(*name: str*) → str

Convert an arbitrary string to a standard distribution name Any runs of non-alphanumeric/. characters are replaced with a single '-'.

- This was copied from *pkg_resources* (part of *setuptools*)

bandersnatch also lower cases the returned name

bandersnatch.utils.**convert_url_to_path**(*url: str*) → str

bandersnatch.utils.**find**(*root: Union[pathlib.Path, str]*, *dirs: bool = True*) → str

A test helper simulating 'find'.

Iterates over directories and filenames, given as relative paths to the root.

bandersnatch.utils.**hash**(*path: pathlib.Path*, *function: str = 'sha256'*) → str

bandersnatch.utils.**make_time_stamp**() → str

Helper function that returns a timestamp suitable for use in a filename on any OS

bandersnatch.utils.**recursive_find_files**(*files: Set[pathlib.Path]*, *base_dir: pathlib.Path*) → None

bandersnatch.utils.**rewrite**(*filepath: Union[str, pathlib.Path]*, *mode: str = 'w'*, ***kw: Any*) → Generator[IO, None, None]

Rewrite an existing file atomically to avoid programs running in parallel to have race conditions while reading.

bandersnatch.utils.**unlink_parent_dir**(*path: pathlib.Path*) → None

Remove a file and if the dir is empty remove it

bandersnatch.utils.**user_agent**() → str

5.1.13 bandersnatch.verify module

async bandersnatch.verify.**delete_unowned_files**(*mirror_base: pathlib.Path*, *executor: concurrent.futures.thread.ThreadPoolExecutor*, *all_package_files: List[pathlib.Path]*, *dry_run: bool*) → int

async bandersnatch.verify.**get_latest_json**(*master: bandersnatch.master.Master*, *json_path: pathlib.Path*, *config: configparser.ConfigParser*, *executor: Optional[concurrent.futures.thread.ThreadPoolExecutor] = None*, *delete_removed_packages: bool = False*) → None

async bandersnatch.verify.**metadata_verify**(*config: configparser.ConfigParser*, *args: argparse.Namespace*) → int

Crawl all saved JSON metadata or online to check we have all packages if delete - generate a diff of unowned files

bandersnatch.verify.**on_error**(*stop_on_error: bool*, *exception: BaseException*, *package: str*) → None

async bandersnatch.verify.**verify**(*master: bandersnatch.master.Master*, *config: configparser.ConfigParser*, *json_file: str*, *mirror_base_path: pathlib.Path*, *all_package_files: List[pathlib.Path]*, *args: argparse.Namespace*, *executor: Optional[concurrent.futures.thread.ThreadPoolExecutor] = None*, *releases_key: str = 'releases'*) → None

async bandersnatch.verify.**verify_producer**(*master: bandersnatch.master.Master*, *config: configparser.ConfigParser*, *all_package_files: List[pathlib.Path]*, *mirror_base_path: pathlib.Path*, *json_files: List[str]*, *args: argparse.Namespace*, *executor: Optional[concurrent.futures.thread.ThreadPoolExecutor] = None*) → None

5.2 bandersnatch_filter_plugins package

5.2.1 Package contents

5.2.2 Submodules

5.2.3 bandersnatch_filter_plugins.blocklist_name module

class bandersnatch_filter_plugins.blocklist_name.**BlockListProject**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterProjectPlugin*

blocklist_package_names: List[str] = []

check_match(***kwargs: Any*) → bool

Check if the package name matches against a project that is blocklisted in the configuration.

Parameters **name** (*str*) – The normalized package name of the package/project to check against the blocklist.

Returns True if it matches, False otherwise.

Return type bool

deprecated_name: str = 'blacklist_project'

filter(*metadata: Dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initialize_plugin() → None
Initialize the plugin

name = 'blocklist_project'

class bandersnatch_filter_plugins.blocklist_name.**BlockListRelease**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterReleasePlugin*

blocklist_package_names: List[packaging.requirements.Requirement] = []

deprecated_name: str = 'blacklist_release'

filter(*metadata: Dict*) → bool
Returns False if version fails the filter, i.e. matches a blocklist version specifier

initialize_plugin() → None
Initialize the plugin

name = 'blocklist_release'

5.2.4 bandersnatch_filter_plugins.filename_name module

class bandersnatch_filter_plugins.filename_name.**ExcludePlatformFilter**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterReleaseFilePlugin*

Filters releases based on regex patterns defined by the user.

filter(*metadata: Dict*) → bool
Returns False if file matches any of the filename patterns

initialize_plugin() → None
Initialize the plugin reading patterns from the config.

name = 'exclude_platform'

5.2.5 bandersnatch_filter_plugins.latest_name module

class bandersnatch_filter_plugins.latest_name.**LatestReleaseFilter**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterReleasePlugin*

Plugin to download only latest releases

filter(*metadata: Dict*) → bool
Returns False if version fails the filter, i.e. is not a latest/current release

initialize_plugin() → None
Initialize the plugin reading patterns from the config.

keep = 0

name = 'latest_release'

5.2.6 bandersnatch_filter_plugins.metadata_filter module

class bandersnatch_filter_plugins.metadata_filter.**RegexFilter**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.Filter*

Plugin to download only packages having metadata matching at least one of the specified patterns.

filter(*metadata: Dict*) → bool
Filter out all projects that don't match the specified metadata patterns.

initialize_plugin() → None
Initialize the plugin reading patterns from the config.

initilized = False

match_patterns = 'any'

name = 'regex_filter'

nulls_match = True

patterns: Dict = {}

class bandersnatch_filter_plugins.metadata_filter.**RegexProjectMetadataFilter**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterMetadataPlugin*, *bandersnatch_filter_plugins.metadata_filter.RegexFilter*

Plugin to download only packages having metadata matching at least one of the specified patterns.

filter(*metadata: Dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initilize_plugin() → None

initilized = False

match_patterns = 'any'

name = 'regex_project_metadata'

nulls_match = True

patterns: Dict = {}

class bandersnatch_filter_plugins.metadata_filter.**RegexReleaseFileMetadataFilter**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterReleaseFilePlugin*, *bandersnatch_filter_plugins.metadata_filter.RegexFilter*

Plugin to download only release files having metadata matching at least one of the specified patterns.

filter(*metadata: Dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initilize_plugin() → None

initilized = False

match_patterns = 'any'

name = 'regex_release_file_metadata'

nulls_match = True

patterns: Dict = {}

class bandersnatch_filter_plugins.metadata_filter.**VersionRangeFilter**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.Filter*

Plugin to download only items having metadata version ranges matching specified versions.

filter(*metadata: Dict*) → bool
Return False for input not having metadata entries matching the specified version specifier.

initialize_plugin() → None
Initialize the plugin reading version ranges from the config.

initilized = False

name = 'version_range_filter'

nulls_match = True

specifiers: Dict = {}

class bandersnatch_filter_plugins.metadata_filter.**VersionRangeProjectMetadataFilter**(**args: Any*, ***kwargs Any*)

Bases: *bandersnatch.filter.FilterMetadataPlugin*, *bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter*

Plugin to download only projects having metadata entries matching specified version ranges.

filter(*metadata: dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initialize_plugin() → None
Code to initialize the plugin

initilized = False

name = 'version_range_project_metadata'

nulls_match = True

specifiers: Dict = {}

class bandersnatch_filter_plugins.metadata_filter.**VersionRangeReleaseFileMetadataFilter**(**ar*
Any
***k*
Any

Bases: *bandersnatch.filter.FilterReleaseFilePlugin*, *bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter*

Plugin to download only release files having metadata entries matching specified version ranges.

filter(*metadata: dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initialize_plugin() → None
Code to initialize the plugin

initilized = False

name = 'version_range_release_file_metadata'

nulls_match = True

specifiers: Dict = {}

5.2.7 bandersnatch_filter_plugins.prerelease_name module

class bandersnatch_filter_plugins.prerelease_name.**PreReleaseFilter**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterReleasePlugin*

Filters releases considered pre-releases.

PRERELEASE_PATTERNS = ('.+rc\\d+$', '.+a(lpha)?\\d+$', '.+b(eta)?\\d+$', '.+dev\\d+$')

filter(*metadata: Dict*) → bool
Returns False if version fails the filter, i.e. follows a prerelease pattern

initialize_plugin() → None
Initialize the plugin reading patterns from the config.

name = 'prerelease_release'

patterns: List[Pattern] = []

5.2.8 bandersnatch_filter_plugins.regex_name module

class bandersnatch_filter_plugins.regex_name.**RegexProjectFilter**(**args: Any, **kwargs: Any*)

Bases: *bandersnatch.filter.FilterProjectPlugin*

Filters projects based on regex patters defined by the user.

check_match(*name: str*) → bool
Check if a release version matches any of the specified patterns.

Parameters **name** (*str*) – Release name

Returns True if it matches, False otherwise.

Return type bool

filter(*metadata: Dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initialize_plugin() → None
Initialize the plugin reading patterns from the config.

name = 'regex_project'

patterns: List[Pattern] = []

class bandersnatch_filter_plugins.regex_name.**RegexReleaseFilter**(**args: Any, **kwargs: Any*)

Bases: *bandersnatch.filter.FilterReleasePlugin*

Filters releases based on regex patters defined by the user.

filter(*metadata: Dict*) → bool
Returns False if version fails the filter, i.e. follows a regex pattern

initialize_plugin() → None
Initialize the plugin reading patterns from the config.

name = 'regex_release'

patterns: List[Pattern] = []

5.2.9 bandersnatch_filter_plugins.allowlist_name module

class bandersnatch_filter_plugins.allowlist_name.**AllowListProject**(**args: Any, **kwargs: Any*)

Bases: *bandersnatch.filter.FilterProjectPlugin*

allowlist_package_names: List[str] = []

check_match(***kwargs: Any*) → bool
Check if the package name matches against a project that is blocklisted in the configuration.

Parameters **name** (*str*) – The normalized package name of the package/project to check against the blocklist.

Returns True if it matches, False otherwise.

Return type bool

deprecated_name: str = 'whitelist_project'

filter(*metadata: Dict*) → bool
Check if the plugin matches based on the package's metadata.

Returns True if the values match a filter rule, False otherwise

Return type bool

initialize_plugin() → None
Initialize the plugin

name = 'allowlist_project'

class bandersnatch_filter_plugins.allowlist_name.**AllowListRelease**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.filter.FilterReleasePlugin*

allowlist_package_names: List[packaging.requirements.Requirement] = []

deprecated_name: str = 'whitelist_release'

filter(*metadata: Dict*) → bool
Returns False if version fails the filter, i.e. doesn't matches an allowlist version specifier

initialize_plugin() → None
Initialize the plugin

name = 'allowlist_release'

class bandersnatch_filter_plugins.allowlist_name.**AllowListRequirements**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch_filter_plugins.allowlist_name.AllowListProject*

name = 'project_requirements'

class bandersnatch_filter_plugins.allowlist_name.**AllowListRequirementsPinned**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch_filter_plugins.allowlist_name.AllowListRelease*

name = 'project_requirements_pinned'

bandersnatch_filter_plugins.allowlist_name.**get_requirement_files**(*allowlist: SectionProxy*) → Iterator[pathlib.Path]

5.3 bandersnatch_storage_plugins package

5.3.1 Package contents

5.3.2 Submodules

5.3.3 bandersnatch_storage_plugins.filesystem module

class bandersnatch_storage_plugins.filesystem.**FilesystemStorage**(**args: Any*, ***kwargs: Any*)

Bases: *bandersnatch.storage.StoragePlugin*

PATH_BACKEND
alias of pathlib.Path

compare_files(*file1: Union[pathlib.Path, str]*, *file2: Union[pathlib.Path, str]*) → bool
Compare two files, returning true if they are the same and False if not.

copy_file(*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*) → None
Copy a file from **source** to **dest**

delete_file(*path: Union[pathlib.Path, str]*, *dry_run: bool = False*) → int
Delete the provided path, recursively if necessary.

exists(*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path exists

find(*root: Union[pathlib.Path, str]*, *dirs: bool = True*) → str
A test helper simulating 'find'.

Iterates over directories and filenames, given as relative paths to the root.

get_hash(*path: Union[pathlib.Path, str]*, *function: str = 'sha256'*) → str
Get the sha256sum of a given **path**

get_lock(*path: Optional[str] = None*) → filelock.UnixFileLock
Retrieve the appropriate *FileLock* backend for this storage plugin

Parameters **path** (*str*) – The path to use for locking

Returns A *FileLock* backend for obtaining locks

Return type *SwiftFileLock*

is_dir(*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path is a directory.

is_file(*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path is a file.

mkdir(*path: Union[pathlib.Path, str]*, *exist_ok: bool = False*, *parents: bool = False*) → None
Create the provided directory

name = 'filesystem'

open_file(*path: Union[pathlib.Path, str]*, *text: bool = True*, *encoding: str = 'utf-8'*) → Generator[IO, None, None]
Yield a file context to iterate over. If text is true, open the file with 'rb' mode specified.

read_file(*path: Union[pathlib.Path, str]*, *text: bool = True*, *encoding: str = 'utf-8'*, *errors: Optional[str] = None*) → Union[str, bytes]
Return the contents of the requested file, either a bytestring or a unicode string depending on whether **text** is True

rewrite(*filepath: Union[pathlib.Path, str]*, *mode: str = 'w'*, ***kw: Any*) → Generator[IO, None, None]
Rewrite an existing file atomically to avoid programs running in parallel to have race conditions while reading.

rmdir(*path: Union[pathlib.Path, str]*, *recurse: bool = False*, *force: bool = False*, *ignore_errors: bool = False*, *dry_run: bool = False*) → int
Remove the directory. If recurse is True, allow removing empty children. If force is true, remove contents destructively.

update_safe(*filename: Union[pathlib.Path, str]*, ***kw: Any*) → Generator[IO, None, None]
Rewrite a file atomically.

Clients are allowed to delete the tmpfile to signal that they don't want to have it updated.

walk(*root: Union[pathlib.Path, str]*, *dirs: bool = True*) → List[pathlib.Path]

write_file(*path: Union[pathlib.Path, str]*, *contents: Union[str, bytes]*) → None
Write data to the provided path. If **contents** is a string, the file will be opened and written in "r" + "utf-8" mode, if bytes are supplied it will be accessed using "rb" mode (i.e. binary write).

5.3.4 bandersnatch_storage_plugins.swift module

class bandersnatch_storage_plugins.swift.**SwiftFileLock**(*lock_file: str*, *timeout: int = - 1*, *backend: Optional[*bandersnatch_storage_plugins.swift.SwiftStor *= None*)
Bases: filelock.BaseFileLock

Simply watches the existence of the lock file.

property is_locked
True, if the object holds the file lock.

Changed in version 2.0.0: This was previously a method and is now a property.

property path_backend

class bandersnatch_storage_plugins.swift.**SwiftPath**(**args: Any*)
Bases: pathlib.Path

BACKEND: ***bandersnatch_storage_plugins.swift.SwiftStorage***

absolute() → *bandersnatch_storage_plugins.swift.SwiftPath*
Return an absolute version of this path. This function works even if the path doesn't point to anything.

No normalization is done, i.e. all '.' and '..' will be kept along. Use resolve() to get the canonical path to a file.

property backend

exists() → bool
Whether this path exists.

is_dir() → bool
Whether this path is a directory.

is_file() → bool
Whether this path is a regular file (also True for symlinks pointing to regular files).

is_symlink() → bool
Whether this path is a symbolic link.

iterdir(*conn: Optional[swiftclient.client.Connection] = None*, *recurse: bool = False*, *include_swiftkeep: bool = False*) → Generator[*bandersnatch_storage_plugins.swift.SwiftPath*, None, None]
Iterate over the files in this directory. Does not yield any result for the special paths '.' and '..'.

mkdir(*mode: int = 511*, *parents: bool = False*, *exist_ok: bool = False*) → None
Create a new directory at this given path.

read_bytes() → bytes
Open the file in bytes mode, read it, and close the file.

read_text(*encoding: Optional[str] = None*, *errors: Optional[str] = None*) → str
Open the file in text mode, read it, and close the file.

classmethod register_backend(*backend:* bandersnatch_storage_plugins.swift.SwiftStorage) → None

symlink_to(*src: Union[pathlib.Path, str]*, *target_is_directory: bool = False*, *src_container: Optional[str] = None*, *src_account: Optional[str] = None*) → None
Make this path a symlink pointing to the given path. Note the order of arguments (self, target) is the reverse of os.symlink's.

touch() → None
Create this file with the given access mode, if it doesn't exist.

unlink(*missing_ok: bool = False*) → None
Remove this file or link. If the path is a directory, use rmdir() instead.

write_bytes(*contents: bytes*, *encoding: Optional[str] = 'utf-8'*, *errors: Optional[str] = None*) → int
Open the file in bytes mode, write to it, and close the file.

write_text(*contents: Optional[str]*, *encoding: Optional[str] = 'utf-8'*, *errors: Optional[str] = None*) → int
Open the file in text mode, write to it, and close the file.

class bandersnatch_storage_plugins.swift.**SwiftStorage**(**args: Any*, *config: Optional[configparser.ConfigParser] = None*, ***kwargs: Any*)
Bases: *bandersnatch.storage.StoragePlugin*

PATH_BACKEND
alias of *bandersnatch_storage_plugins.swift.SwiftPath*

compare_files(*file1: Union[pathlib.Path, str]*, *file2: Union[pathlib.Path, str]*) → bool
Compare two files, returning true if they are the same and False if not.

connection() → Generator[swiftclient.client.Connection, None, None]

copy_file(*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*, *dest_container: Optional[str] = None*) → None
Copy a file from **source** to **dest**

copy_local_file(*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*) → None
Copy the contents of a local file to a destination in swift

property default_container

delete_file(*path: Union[pathlib.Path, str]*, *dry_run: bool = False*) → int
Delete the provided path, recursively if necessary.

property directory

exists(*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path exists

find(*root: Union[pathlib.Path, str]*, *dirs: bool = True*) → str
A test helper simulating 'find'.

Iterates over directories and filenames, given as relative paths to the root.

flock_path: Union[pathlib.Path, str]

get_config_value(*config_key: str*, **env_keys: Any*, *default: Optional[str] = None*) → Optional[str]

get_container(*container: Optional[str] = None*) → List[Dict[str, str]]
Given the name of a container, return its contents.

Parameters **container** (*str*) – The name of the desired container, defaults to *default_container*

Returns A list of objects in the container if it exists

Return type List[Dict[str, str]]

Example:

```
>>> plugin.get_container("bandersnatch")
[{
    'bytes': 1101, 'last_modified': '2020-02-27T19:10:17.922970',
    'hash': 'a76b4c69bfcf82313bbdc0393b04438a',
    'name': 'packages/pyyaml/PyYAML-5.3/LICENSE',
    'content_type': 'application/octet-stream'
 }, {
    'bytes': 1779, 'last_modified': '2020-02-27T19:10:17.845520',
    'hash': 'c60081e1ad65830b098a7f21a8a8c90e',
    'name': 'packages/pyyaml/PyYAML-5.3/PKG-INFO',
    'content_type': 'application/octet-stream'
 }, {
    'bytes': 1548, 'last_modified': '2020-02-27T19:10:17.730490',
    'hash': '9a8bdf19e93d4b007598b5eb97b461eb',
    'name': 'packages/pyyaml/PyYAML-5.3/README',
    'content_type': 'application/octet-stream'
 }, ...
]
```

get_hash(*path: Union[pathlib.Path, str]*, *function: str = 'sha256'*) → str
Get the sha256sum of a given **path**

get_lock(*path: Optional[str] = None*) → *bandersnatch_storage_plugins.swift.SwiftFileLock*
Retrieve the appropriate *FileLock* backend for this storage plugin

Parameters **path** (*str*) – The path to use for locking

Returns A *FileLock* backend for obtaining locks

Return type *SwiftFileLock*

get_object(*container_name: str*, *file_path: str*) → bytes
Retrieve an object from swift, base64 decoding the contents.

initialize_plugin() → None
Code to initialize the plugin

is_dir(*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path is a directory.

is_file (*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path is a file.

is_symlink (*path: Union[pathlib.Path, str]*) → bool
Check whether the provided path is a symlink

mkdir (*path: Union[pathlib.Path, str]*, *exist_ok: bool = False*, *parents: bool = False*) → None
Create the provided directory

This operation is a no-op on swift.

name = 'swift'

open_file (*path: Union[pathlib.Path, str]*, *text: bool = True*) → Generator[IO, None, None]
Yield a file context to iterate over. If text is false, open the file with 'rb' mode specified.

read_file (*path: Union[pathlib.Path, str]*, *text: bool = True*, *encoding: str = 'utf-8'*, *errors: Optional[str] = None*) → Union[str, bytes]
Return the contents of the requested file, either a a bytestring or a unicode string depending on whether **text** is True

rewrite (*filepath: Union[pathlib.Path, str]*, *mode: str = 'w'*, ***kw: Any*) → Generator[IO, None, None]
Rewrite an existing file atomically to avoid programs running in parallel to have race conditions while reading.

rmdir (*path: Union[pathlib.Path, str]*, *recurse: bool = False*, *force: bool = False*, *ignore_errors: bool = False*, *dry_run: bool = False*) → int
Remove the directory. If recurse is True, allow removing empty children.

If force is true, remove contents destructively.

symlink (*src: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*, *src_container: Optional[str] = None*, *src_account: Optional[str] = None*) → None
Create a symlink at **dest** that points back at **source**

update_safe (*filename: Union[pathlib.Path, str]*, ***kw: Any*) → Generator[IO, None, None]
Rewrite a file atomically.

Clients are allowed to delete the tmpfile to signal that they don't want to have it updated.

update_timestamp (*path: Union[pathlib.Path, str]*) → None

walk (*root: Union[pathlib.Path, str]*, *dirs: bool = True*, *conn: Optional[swiftclient.client.Connection] = None*) → List[*bandersnatch_storage_plugins.swift.SwiftPath*]

write_file (*path: Union[pathlib.Path, str]*, *contents: Union[str, bytes, IO]*, *encoding: Optional[str] = None*, *errors: Optional[str] = None*) → None
Write data to the provided path. If **contents** is a string, the file will be opened and written in "r" + "utf-8" mode, if bytes are supplied it will be accessed using "rb" mode (i.e. binary write).

PYTHON MODULE INDEX

b

INDEX

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X