bandersnatch Documentation

Release 5.3.0

PyPA

Aug 01, 2022

CONTENTS

1 Command line usage **3**
- 1.1 bandersnatch optional arguments . . . 3
- 1.2 bandersnatch delete . . . 3
  - 1.2.1 bandersnatch delete positional arguments . . . 3
  - 1.2.2 bandersnatch delete optional arguments . . . 3
- 1.3 bandersnatch mirror . . . 4
  - 1.3.1 bandersnatch mirror optional arguments . . . 4
- 1.4 bandersnatch verify . . . 4
  - 1.4.1 bandersnatch verify optional arguments . . . 4
- 1.5 bandersnatch sync . . . 4
  - 1.5.1 bandersnatch sync positional arguments . . . 4
  - 1.5.2 bandersnatch sync optional arguments . . . 5

2 Contents **7**
- 2.1 Installation . . . 7
  - 2.1.1 pip . . . 7
- 2.2 Storage options for bandersnatch . . . 7
  - 2.2.1 Filesystem Support . . . 7
  - 2.2.2 Amazon S3 . . . 8
  - 2.2.3 OpenStack Swift . . . 9
- 2.3 Mirror configuration . . . 10
  - 2.3.1 directory . . . 10
  - 2.3.2 json . . . 10
  - 2.3.3 release-files . . . 11
  - 2.3.4 master . . . 11
  - 2.3.5 timeout . . . 11
  - 2.3.6 global-timeout . . . 11
  - 2.3.7 workers . . . 12
  - 2.3.8 hash-index . . . 12
  - 2.3.9 stop-on-error . . . 12
  - 2.3.10 log-config . . . 13
  - 2.3.11 root_uri . . . 13
  - 2.3.12 diff-file . . . 13
  - 2.3.13 diff-append-epoch . . . 13
  - 2.3.14 compare-method . . . 14
  - 2.3.15 proxy . . . 14
- 2.4 Mirror filtering . . . 14
  - 2.4.1 Plugins Enabling . . . 14
  - 2.4.2 allowlist / blocklist filtering settings . . . 15
  - 2.4.3 packages . . . 15

2.4.4 Metadata Filtering . . . 15
2.4.5 requirements files Filtering . . . 16
2.4.6 Prerelease filtering . . . 17
2.4.7 Regex filtering . . . 17
2.4.8 Platform/Python-specific binaries filtering . . . 17
2.4.9 Keep only latest releases . . . 18
2.4.10 Block projects above a specified size threshold . . . 18
2.5 Serving your Mirror . . . 19
2.5.1 BanderX . . . 19
2.6 Contributing . . . 20
2.6.1 Code of Conduct . . . 20
2.6.2 Getting Started . . . 20
2.6.3 Creating a Pull Request . . . 22
2.6.4 Linting . . . 23
2.6.5 Running Bandersnatch . . . 23
2.6.6 Running Unit Tests . . . 23
2.6.7 Making a bandersnatch release to GitHub + PyPI . . . 25
2.7 bandersnatch . . . 25
2.7.1 bandersnatch package . . . 25
2.7.2 bandersnatch_filter_plugins package . . . 37
2.7.3 bandersnatch_storage_plugins package . . . 43

Python Module Index **49**

Index **51**

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 normalized package name path too

CHAPTER

ONE

COMMAND LINE USAGE

PyPI PEP 381 mirroring client.

```
bandersnatch [-h] [--version] [-c CONFIG] [--debug] {delete,mirror,verify,sync} ...
```

1.1 bandersnatch optional arguments

- ***-h***, ***--help*** - show this help message and exit
- ***--version*** - show program's version number and exit
- ***-c*** `CONFIG`, ***--config*** `CONFIG` - use configuration file (default: %(default)s) (default: `/etc/bandersnatch.conf`)
- ***--debug*** - Turn on extra logging (DEBUG level)

1.2 bandersnatch delete

Consulte metadata (locally or remotely) and delete entire package artifacts.

```
bandersnatch delete [-h] [--dry-run] [--workers WORKERS] [pypi_packages [pypi_packages ..
↪.]]
```

1.2.1 bandersnatch delete positional arguments

- ***pypi_packages*** (default: `None`)

1.2.2 bandersnatch delete optional arguments

- ***-h***, ***--help*** - show this help message and exit
- ***--dry-run*** - Do not download or delete files
- ***--workers*** `WORKERS` - # of parallel iops [Defaults to bandersnatch.conf] (default: `0`)

1.3 bandersnatch mirror

Performs a one-time synchronization with the PyPI master server.

```
bandersnatch mirror [-h] [--force-check]
```

1.3.1 bandersnatch mirror optional arguments

- ***-h***, ***--help*** - show this help message and exit
- ***--force-check*** - Force bandersnatch to reset the PyPI serial (move serial file to /tmp) to perform a full sync

1.4 bandersnatch verify

Read in Metadata and check package file validity

```
bandersnatch verify [-h] [--delete] [--dry-run] [--json-update] [--workers WORKERS]
```

1.4.1 bandersnatch verify optional arguments

- ***-h***, ***--help*** - show this help message and exit
- ***--delete*** - Enable deletion of packages not active
- ***--dry-run*** - Do not download or delete files
- ***--json-update*** - Enable updating JSON from PyPI
- ***--workers*** WORKERS - # of parallel iops [Defaults to bandersnatch.conf] (default: `0`)

1.5 bandersnatch sync

Synchronize specific packages with the PyPI master server.

```
bandersnatch sync [-h] [--skip-simple-root] package [package ...]
```

1.5.1 bandersnatch sync positional arguments

- ***package*** - The name of package to sync (default: `None`)

1.5.2 bandersnatch sync optional arguments

- ***-h***, ***--help*** - show this help message and exit
- ***--skip-simple-root*** - Skip updating simple index root page

CHAPTER

TWO

CONTENTS

2.1 Installation

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

- bandersnatch **requires** >= `Python 3.8.0`

2.1.1 pip

This installs the latest stable, released version.

```
$ python3.8 -m venv bandersnatch
$ bandersnatch/bin/pip install bandersnatch
$ bandersnatch/bin/bandersnatch --help
```

2.2 Storage options for bandersnatch

Bandersnatch was originally developed for POSIX file system. Bandersnatch now supports:

- POSIX / Windows filesystem (transparently via pathlib)
- Amazon S3
- OpenStack Swift

2.2.1 Filesystem Support

This is the default mode for bandersnatch.

Config Example

```
[mirror]
directory = /data/pypi/mirror
storage-backend = filesystem
# Optional index hashing to store simple HTML in directories
# Recommended as PyPI has a lot of packages these days
hash-index = true
```

Serving your Mirror

Simple html is stored within the file system structure. Please use your favorite http server such as Apache or NGINX. Refer to *Serving* documentation about a NGINX Docker container option.

2.2.2 Amazon S3

To enable S3 support the optional `s3` install must be done:

- `pip install bandersnatch[s3]`
- Add a `[s3]` section in the bandersnatch config file

You will need an AWS account and an S3 bucket

Config Example

```
[mirror]
# Place your s3 path here - e.g. /{bucket name}/{prefix}
directory = /my-s3-bucket/prefix
# Set storage-backend to s3
storage-backend = s3
# Provide s3 style path - e.g. /{bucket name}/{prefix}/{key}
diff-file = /your-s3-bucket/bucket-key

[s3]
# Optional Region name - can be empty if IAM are set
region_name = us-east-1
aws_access_key_id = your s3 access key
aws_secret_access_key = your s3 secret access key
# Use endpoint_url to indicate custom s3 endpoint e.g. like minio etc.
endpoint_url = endpoint url
# Optional manual signature version for compatibility
signature_version = s3v4
```

Serving your Mirror

S3 Bandersnatch mirrors are designed to be served with s3 static sites and can also be used with the Amazon CDN service or another CDN service.

I assume you have already set up an AWS account and S3 bucket, and the Bandersnatch sync job has successfully ran.

Enabling website hosting for the bucket

When you enable the website hosting for a bucket, this bucket can be viewed as static website. Using the s3 domain or your customized domain.

Please read Amazon documents to get detailed instructions

Most cloud provider who provide a s3-compatible service will provide this service as well. Please consult to your service assistant to get detailed instructions.

Use CloudFront or other cdn service to speed up the static mirror(optional)

If your mirror is targeted to global clients, you can use CloudFront or other CDN service to speed up the mirror.

Please read Amazon documents to get detailed instructions

Set redirect or url rewrite in CloudFront or other cdn(optional)

In most cases, packages and index pages are all inside `/my-s3-bucket/prefix/web`, if you set up a steps above, you should be able to use the mirror like this:

```
pip install -i my-s3-bucket.cloudfront.net/prefix/web/simple install django
```

But there are two main disadvantages:

1. The url is quite long and exposing the structure of bucket.
2. Users will be able to view all content in the bucket, including bandersnatch todo file and status file.

It is strongly recommended to set redirect or url rewrite for CDN. Please contact your service assistant for detailed instructions.

2.2.3 OpenStack Swift

To enable Swift support the optional `swift` install must be done:

- `pip install bandersnatch[swift]`
- Add a `[swift]` section in the bandersnatch config file

Config Example

```
[mirror]
directory = /prefix
storage-backend = swift

[swift]
default_container = bandersnatch
```

Serving your Mirror

Requires that the cluster has staticweb enabled.

```
# Check that staticweb is enabled
swift capabilities | grep staticweb
# Make the container world-readable and enable pseudo-directory translation
swift post bandersnatch -r '.r:*' -m 'web-index: index.html'
```

2.3 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.

2.3.1 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
```

2.3.2 json

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

Example:

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

2.3.3 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
```

2.3.4 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
```

2.3.5 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
```

2.3.6 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
```

2.3.7 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.

2.3.8 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.

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
```

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;
```

2.3.9 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
```

2.3.10 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
```

2.3.11 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
```

2.3.12 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
```

2.3.13 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
```

2.3.14 compare-method

The compare method is used to set how to compare an existing file with upstream file to determine whether a download is required:

- hash: this is the default which reads local file content and computes hashes (currently sha256sum), it is reliable but sometimes slower;
- stat: use file size and change time to compare, which is named after the stat() syscall, this avoids retrieving the full file content thus reducing some io workloads.

Example:

```
[mirror]
compare-method = hash
```

2.3.15 proxy

The proxy is used only when requesting master server, eg. downloading index or package file from pypi.org. The proxy value will be passed to aiohttp as proxy parameter, like `aiohttp.get(link, proxy=yourproxy)`, check the aioproxy manual for more details: https://docs.aiohttp.org/en/stable/client_advanced.html#proxy-support

Example:

```
[mirror]
proxy=http://myproxy.com
```

2.4 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 can use the PEP503 normalized names. Any non-normalized names in `bandersnatch.conf` will be automatically converted.

E.g. to Blocklist discord.py the string 'discord-py' is correct, but 'discord.PY' will also work.

2.4.1 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
    ...
```

2.4.2 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.

2.4.3 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
```

2.4.4 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
```

2.4.5 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
```

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.

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 info, `release` containing the version of the release and `release_file` the metadata for an individual file for that release.

2.4.6 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
```

2.4.7 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.

2.4.8 Platform/Python-specific binaries filtering

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

You can also exclude Python versions by their minor version (ex. Python 2.6, 2.7) if you're sure your mirror does not need to serve these binaries.

```
[plugins]
enabled =
    exclude_platform
[blocklist]
platforms =
    windows
    py2.6
    py2.7
```

Available platforms are:

- `windows`
- `macos`
- `freebsd`
- `linux`

Available python versions are:

- `py2.4` ~ `py2.7`
- `py3.1` ~ `py3.10`

2.4.9 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.

2.4.10 Block projects above a specified size threshold

There is an increasing number of projects that consume a large amount of space. At the time of writing (Jan 2021) the stats shows some of the top projects consume over 100GB each, and the top 100 projects all consume more than 8GB each.

If your usecase for a PyPI mirror is to have the diversity of packages but you have storage constraints, it may be preferable to block large packages. This can be done with the `size_project_metadata` plugin.

```
[plugins]
enabled =
    size_project_metadata

[size_project_metadata]
max_package_size = 1G
```

This will block the download of any project whose total size exceeds 1GB. (The value of `max_package_size` can be either an integer number of bytes or a human- readable value as shown.)

It can be combined with an allowlist to overrule the size limit for large projects you are actually interested in and want make exceptions for. The following has the logic of including all projects where the size is <1GB *or* the name is numpy.

```
[plugins]
enabled =
    size_project_metadata

[allowlist]
```

(continues on next page)

(continued from previous page)

```
packages =
    numpy

[size_project_metadata]
max_package_size = 1G
```

If the allowlist_project is also enabled, then the filter becomes a logical and, e.g. the following will include all projects where the size is <1GB *and* the name appears in the allowlist:

```
[plugins]
enabled =
    size_project_metadata
    allowlist_project

[allowlist]
packages =
    numpy
    scapy
    flask

[size_project_metadata]
max_package_size = 1G
```

Note that because projects naturally grow in size, one that was once within the size can grow beyond the limit, and will stop being updated. It is then a choice for the maintainer to make whether to add the package to the exception list (and possibly run a `bandersnatch mirror --force-check`) or to prune the project from the mirror (with `bandersnatch delete <package_name>`).

2.5 Serving your Mirror

So if you've had a successful `bandersnatch mirror` run, you're now ready to serve your mirror. Any webserver can do this, as long as it can serve the simple HTML and packages directory that the HTML links to.

2.5.1 BanderX

`banderx` is a very simple NGINX docker image with a sample config included. The example only does HTTP and expects you to do your own HTTPS/TLS elsewhere.

- Default config is not setup for `hash_index = true` synced bandersnatch mirror
  - The *hash_index* serving config is in the example config and needs to be uncommented
  - It also sets the correct JSON MIME type for `/json` + `/pypi`

Docker Build

- `cd src/banderx`
- `docker build -t banderx .`

Docker Run

- `docker run --name bandersnatch_nginx --mount type=bind,source=/data/pypi/web, target=/data/pypi/web banderx`
- For custom config add:
  - `--mount type=bind,source=$PWD/nginx.conf,target=/config/nginx.conf`

Bind Mount Nginx Config

If you want a different nginx config bind mount to:

- `/config/nginx.conf`

The config defaults for the mirror to be bind mounted to:

- `/data/pypi/web`

2.6 Contributing

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

2.6.1 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.

2.6.2 Getting Started

Bandersnatch is developed using the GitHub Flow

Pre Install

Please make sure you system has the following:

- Python 3.8.0 or greater
- git client
- docker
  - *Optional* but needed to run S3 Tests

Checkout `bandersnatch`

Lets now cd to where we want the code and clone the repo:

- `cd somewhere`
- `git clone git@github.com:pypa/bandersnatch.git`

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 -m venv /path/to/venv
/path/to/venv/bin/pip install --upgrade pip
```

For example:

```
$ python3 -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:
      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 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
...
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
```

(continues on next page)

(continued from previous page)

- Then install bandersnatch in editable mode:

```
/path/to/venv/bin/pip install -e .
```

- (Optional) finally setup pre-commit to run automatically before committing:

```
/path/to/venv/bin/pre-commit run -a
```

Congrats, now you have a bandersnatch development environment ready to go! Just a few details to cover left.

S3 Unit Tests

S3 unittests are more integration tests. They depend on minio to work.

- You will either need to skip them or install mino
- Install options: https://docs.min.io/docs/

Docker Install

Docker is an easy way to get minio to run for tests to pass.

```
docker run \
  -p 9000:9000 \
  -p 9001:9001 \
  --name minio1 \
  -v /Users/cooper/tmp/minio:/data \
  minio/minio server /data --console-address ":9001"
```

2.6.3 Creating a Pull Request

Changelog entry

PRs must have an entry in CHANGES.md that references the PR number in the format of "PR #{number}". You can get the number your PR will be assigned beforehand using Next PR Number. **Some trivial changes (eg. typo fixes) won't need an entry, but most of the time, your change will. If unsure, take a look at what's been logged before or just add one to be safe.**

This is enforced by a GitHub Actions workflow.

2.6.4 Linting

We use pre-commit to run linters and formatters. If you never configured pre-commit to run automatically or just want to do a full check of the codebase, please run pre-commit directly.

```
cd bandersnatch
/path/to/venv/bin/pre-commit -a
```

2.6.5 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
```

2.6.6 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] [-e py3|docs]
```

Example output:

```
$ 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
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%]
```

(continues on next page)

(continued from previous page)

```
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%]
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 ______________________________________
↪_____________________________________________________________________________________
```

(continues on next page)

(continued from previous page)

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

You want to see:

```
py3: commands succeeded
congratulations :)
```

2.6.7 Making a bandersnatch release to GitHub + PyPI

Please rely on our pypi_upload GitHub actions to build and upload our releases.

- To cut a release first make a PR updating:
  - the version in `setup.cfg` + `src/badnersnatch/__init__.py`
  - Update `CHANGES.md`. Here check for typos + missing commits that should be mentioned
    - Example PR: https://github.com/pypa/bandersnatch/pull/1069
- THen, once merged and CI is passing
  - Cut a GitHub Release and GitHub Actions will package and upload to PyPI.
  - https://github.com/pypa/bandersnatch/releases
    - Select "Draft a new release"

Conventions

- Use the version as the branch and tag names
- Copy the Change Log for the version from CHANGES.md
  - The web form supports markdown so it can be directly copied

2.7 bandersnatch

2.7.1 bandersnatch package

Package contents

Submodules

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*, *compare_method*, *download_mirror*, *download_mirror_no_fallback*)

Bases: tuple

cleanup: bool

Alias for field number 6

compare_method: str

Alias for field number 8

diff_append_epoch: bool

Alias for field number 3

diff_file_path: str

Alias for field number 2

digest_name: str

Alias for field number 4

download_mirror: str

Alias for field number 9

download_mirror_no_fallback: bool

Alias for field number 10

json_save: bool

Alias for field number 0

release_files_save: bool

Alias for field number 7

root_uri: str

Alias for field number 1

storage_backend_name: str

Alias for field number 5

class bandersnatch.configuration.**Singleton**

Bases: type

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

bandersnatch.delete module

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

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

async bandersnatch.delete.**delete_simple_page**(*simple_base_path: pathlib.Path*, *package: str*, *hash_index: bool = False*, *dry_run: bool = True*) → int

bandersnatch.filter module

Blocklist management

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

Bases: object

Base Filter class

property allowlist: SectionProxy

property blocklist: SectionProxy

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_plugins.v2.metadata', 'bandersnatch_filter_plugins.v2.release', 'bandersnatch_filter_plugins.v2.release_file']

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

bandersnatch.log module

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

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

bandersnatch.master module

class bandersnatch.master.**Master**(*url: str*, *timeout: float = 10.0*, *global_timeout: Optional[float] = 18000.0*, *proxy: Optional[str] = None*)

Bases: object

async **all_packages**() → Any

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

async **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**: str

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

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*, *compare_method: Optional[str] = None*, *download_mirror: Optional[str] = None*, *download_mirror_no_fallback: Optional[bool] = False*)

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*, *file_size: str*, *upload_time: datetime.datetime*, *sha256sum: str*, *chunk_size: int = 65536*, *urlpath: str = ''*) → Optional[pathlib.Path]

errors = False

finalize_sync(*sync_index_page: bool = True*) → 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_html_file_tags(*release: Dict*) → str

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

property generationfile: pathlib.Path

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

populate_download_urls(*release_file: Dict[str, str]*) → Tuple[str, List[str]]

Populate download URLs for a certain file, possible combinations are:

- download_mirror is not set: return "url" attribute from release_file
- download_mirror is set, no_fallback is false: prepend "download_mirror + path" before "url"
- download_mirror is set, no_fallback is true: return only "download_mirror + path"

Theoritically we are able to support multiple download mirrors by prepending more urls in the list.

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: pathlib.Path

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: pathlib.Path

property webdir: pathlib.Path

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(*sync_index_page: bool = True*) → 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

pypi_repository_version = '1.0'

async sync_packages() → None

synced_serial: Optional[int] = 0

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

target_serial: Optional[int] = None

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

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: Any

property last_serial: int

property metadata: Dict[str, Any]

property release_files: List

property releases: Any

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

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: str

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_file_size(*path: Union[pathlib.Path, str]*) → int

Get the size of a given **path** in bytes

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._api.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

get_upload_time(*path: Union[pathlib.Path, str]*) → datetime.datetime

Get the upload time of a given **path**

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

move_file(*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*) → None

Move a file from **source** to **dest**

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.

set_upload_time(*path: Union[pathlib.Path, str]*, *time: datetime.datetime*) → None

Set the upload time of a given **path**

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

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.**find_all_files**(*files: Set[pathlib.Path]*, *base_dir: pathlib.Path*) → None

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.**parse_version**(*version: str*) → List[str]

Converts a version string to a list of strings to check the 1st part of build tags. See PEP 425 (https://peps.python.org/pep-0425/#python-tag) for details.

Parameters **version** (*str*) – string in the form of '{major}.{minor}' e.g. '3.6'

Returns

list of 1st element strings from build tag tuples See https://peps.python.org/pep-0425/#python-tag for details. Some Windows binaries have only the 1st part before the file extension. e.g. ['-cp36-', '-pp36-', '-ip36-', '-jy36-', '-py3.6-', '-py3.6.']

Return type List[str]

bandersnatch.utils.**removeprefix**(*original: str*, *prefix: str*) → str

Return a string with the given prefix string removed if present. If the string starts with the prefix string, return string[len(prefix):]. Otherwise, return the original string.

Parameters

- **original** (*str*) – string to remove the prefix (e.g. 'py3.6')
- **prefix** (*str*) – the prefix to remove (e.g. 'py')

Returns either the modified or the original string (e.g. '3.6')

Return type str

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

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*, *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

2.7.2 bandersnatch_filter_plugins package

Package contents

Submodules

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

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] = []

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'

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'

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'

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.

initialized = 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

initialized = False

initilize_plugin() → None

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

initialized = False

initilize_plugin() → None

match_patterns = 'any'

name = 'regex_release_file_metadata'

nulls_match = True

patterns: Dict = {}

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

Bases: *bandersnatch.filter.FilterMetadataPlugin*, *bandersnatch_filter_plugins.allowlist_name.AllowListProject*

Plugin to download only packages having total file sizes less than a configurable threshold.

allowlist_package_names: List[str] = []

filter(*metadata: Dict*) → bool

Return False for projects with metadata indicating total file sizes greater than threshold.

initialize_plugin() → None

Initialize the plugin reading settings from the config.

initialized = False

max_package_size: int = 0

name = 'size_project_metadata'

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.

initialized = 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

initialized = False

name = 'version_range_project_metadata'

nulls_match = True

specifiers: Dict = {}

class bandersnatch_filter_plugins.metadata_filter.**VersionRangeReleaseFileMetadataFilter**(**args: Any*, ***kwargs: 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

initialized = False

name = 'version_range_release_file_metadata'

nulls_match = True

specifiers: Dict = {}

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] = []

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(***kwargs: Any*) → 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] = []

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 allowlisted 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

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] = []

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]

2.7.3 bandersnatch_storage_plugins package

Package contents

Submodules

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_file_size(*path: Union[pathlib.Path, str]*) → int

Return the file size of provided path.

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._unix.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*

get_upload_time(*path: Union[pathlib.Path, str]*) → datetime.datetime

Get the upload time of a given **path**

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

move_file(*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*) → None

Move a file from **source** to **dest**

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.

set_upload_time(*path: Union[pathlib.Path, str]*, *time: datetime.datetime*) → None

Set the upload time of a given **path**

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).

bandersnatch_storage_plugins.swift module

class bandersnatch_storage_plugins.swift.**SwiftFileLock**(*lock_file: str*, *timeout: int = - 1*, *backend: Optional[*bandersnatch_storage_plugins.swift.SwiftStorage*] = None*)

Bases: filelock._api.BaseFileLock

Simply watches the existence of the lock file.

property is_locked: bool

A boolean indicating if the lock file is holding the lock currently.

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

Type return

property path_backend: Type[*bandersnatch_storage_plugins.swift.SwiftPath*]

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: *bandersnatch_storage_plugins.swift.SwiftStorage*

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: str

delete_file(*path: Union[pathlib.Path, str]*, *dry_run: bool = False*) → int

Delete the provided path, recursively if necessary.

property directory: str

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_file_size(*path: Union[pathlib.Path, str]*) → int

Get the size of a given **path** in bytes

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.

get_upload_time(*path: Union[pathlib.Path, str]*) → datetime.datetime

Get the upload time of a given **path**

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.

move_file(*source: Union[pathlib.Path, str]*, *dest: Union[pathlib.Path, str]*, *dest_container: Optional[str] = None*) → None

Move a file from **source** to **dest**

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.

set_upload_time(*path: Union[pathlib.Path, str]*, *time: datetime.datetime*) → None

Set the upload time of a given **path**

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

bandersnatch, 25
bandersnatch.configuration, 25
bandersnatch.delete, 27
bandersnatch.filter, 27
bandersnatch.log, 28
bandersnatch.main, 29
bandersnatch.master, 29
bandersnatch.mirror, 29
bandersnatch.package, 32
bandersnatch.storage, 32
bandersnatch.utils, 35
bandersnatch.verify, 36
bandersnatch_filter_plugins, 37
bandersnatch_filter_plugins.allowlist_name, 42
bandersnatch_filter_plugins.blocklist_name, 37
bandersnatch_filter_plugins.filename_name, 37
bandersnatch_filter_plugins.latest_name, 38
bandersnatch_filter_plugins.metadata_filter, 38
bandersnatch_filter_plugins.prerelease_name, 41
bandersnatch_filter_plugins.regex_name, 41
bandersnatch_storage_plugins, 43
bandersnatch_storage_plugins.filesystem, 43
bandersnatch_storage_plugins.swift, 45

INDEX

A

absolute() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
all_packages() (*bandersnatch.master.Master method*), 29
allowlist (*bandersnatch.filter.Filter property*), 27
allowlist_package_names (*bandersnatch_filter_plugins.allowlist_name.AllowListProject attribute*), 42
allowlist_package_names (*bandersnatch_filter_plugins.allowlist_name.AllowListRelease attribute*), 42
allowlist_package_names (*bandersnatch_filter_plugins.metadata_filter.SizeProjectMetadataFilter attribute*), 39
AllowListProject (*class in bandersnatch_filter_plugins.allowlist_name*), 42
AllowListRelease (*class in bandersnatch_filter_plugins.allowlist_name*), 42
AllowListRequirements (*class in bandersnatch_filter_plugins.allowlist_name*), 43
AllowListRequirementsPinned (*class in bandersnatch_filter_plugins.allowlist_name*), 43
async_main() (*in module bandersnatch.main*), 29

B

BACKEND (*bandersnatch_storage_plugins.swift.SwiftPath attribute*), 45
backend (*bandersnatch_storage_plugins.swift.SwiftPath property*), 45
bandersnatch
    module, 25
bandersnatch.configuration
    module, 25
bandersnatch.delete
    module, 27
bandersnatch.filter
    module, 27
bandersnatch.log
    module, 28
bandersnatch.main
    module, 29
bandersnatch.master
    module, 29
bandersnatch.mirror
    module, 29
bandersnatch.package
    module, 32
bandersnatch.storage
    module, 32
bandersnatch.utils
    module, 35
bandersnatch.verify
    module, 36
bandersnatch_filter_plugins
    module, 37
bandersnatch_filter_plugins.allowlist_name
    module, 42
bandersnatch_filter_plugins.blocklist_name
    module, 37
bandersnatch_filter_plugins.filename_name
    module, 37
bandersnatch_filter_plugins.latest_name
    module, 38
bandersnatch_filter_plugins.metadata_filter
    module, 38
bandersnatch_filter_plugins.prerelease_name
    module, 41
bandersnatch_filter_plugins.regex_name
    module, 41
bandersnatch_safe_name() (*in module bandersnatch.utils*), 35
bandersnatch_storage_plugins
    module, 43
bandersnatch_storage_plugins.filesystem
    module, 43
bandersnatch_storage_plugins.swift
    module, 45
BandersnatchConfig (*class in bandersnatch.configuration*), 25
BandersnatchMirror (*class in bandersnatch.mirror*), 29
blocklist (*bandersnatch.filter.Filter property*), 27
blocklist_package_names (*bander-*

snatch_filter_plugins.blocklist_name.BlockListProject attribute), 37
blocklist_package_names (*bandersnatch_filter_plugins.blocklist_name.BlockListRelease attribute*), 37
BlockListProject (*class in bandersnatch_filter_plugins.blocklist_name*), 37
BlockListRelease (*class in bandersnatch_filter_plugins.blocklist_name*), 37

C

canonicalize_package() (*bandersnatch.storage.Storage static method*), 32
changed_packages() (*bandersnatch.master.Master method*), 29
check_for_deprecations() (*bandersnatch.configuration.BandersnatchConfig method*), 25
check_for_stale_cache() (*bandersnatch.master.Master method*), 29
check_match() (*bandersnatch.filter.Filter method*), 27
check_match() (*bandersnatch_filter_plugins.allowlist_name.AllowListProject method*), 42
check_match() (*bandersnatch_filter_plugins.blocklist_name.BlockListProject method*), 37
check_match() (*bandersnatch_filter_plugins.regex_name.RegexProjectFilter method*), 41
cleanup (*bandersnatch.configuration.SetConfigValues attribute*), 26
cleanup_non_pep_503_paths() (*bandersnatch.mirror.BandersnatchMirror method*), 30
compare_files() (*bandersnatch.storage.Storage method*), 32
compare_files() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 43
compare_files() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46
compare_method (*bandersnatch.configuration.SetConfigValues attribute*), 26
connection() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46
convert_url_to_path() (*in module bandersnatch.utils*), 35
copy_file() (*bandersnatch.storage.Storage method*), 32
copy_file() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 43
copy_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46
copy_local_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46

D

default_container (*bandersnatch_storage_plugins.swift.SwiftStorage property*), 46
delete() (*bandersnatch.storage.Storage method*), 32
delete_file() (*bandersnatch.storage.Storage method*), 32
delete_file() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 43
delete_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46
delete_packages() (*in module bandersnatch.delete*), 27
delete_path() (*in module bandersnatch.delete*), 27
delete_simple_page() (*in module bandersnatch.delete*), 27
delete_unowned_files() (*in module bandersnatch.verify*), 36
deprecated_name (*bandersnatch.filter.Filter attribute*), 27
determine_packages_to_sync() (*bandersnatch.mirror.BandersnatchMirror method*), 30
determine_packages_to_sync() (*bandersnatch.mirror.Mirror method*), 31
diff_append_epoch (*bandersnatch.configuration.SetConfigValues attribute*), 26
diff_file_path (*bandersnatch.configuration.SetConfigValues attribute*), 26
digest_name (*bandersnatch.configuration.SetConfigValues attribute*), 26
directory (*bandersnatch.storage.Storage property*), 33
directory (*bandersnatch_storage_plugins.swift.SwiftStorage property*), 46
download_file() (*bandersnatch.mirror.BandersnatchMirror method*), 30
download_mirror (*bandersnatch.configuration.SetConfigValues at-*

tribute), 26
download_mirror_no_fallback (*bandersnatch.configuration.SetConfigValues attribute*), 26

E

ENTRYPOINT_GROUPS (*bandersnatch.filter.LoadedFilters attribute*), 28
errors (*bandersnatch.mirror.BandersnatchMirror attribute*), 30
ExcludePlatformFilter (*class in bandersnatch_filter_plugins.filename_name*), 37
exists() (*bandersnatch.storage.Storage method*), 33
exists() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 43
exists() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
exists() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46

F

FilesystemStorage (*class in bandersnatch_storage_plugins.filesystem*), 43
Filter (*class in bandersnatch.filter*), 27
filter() (*bandersnatch.filter.Filter method*), 27
filter() (*bandersnatch_filter_plugins.allowlist_name.AllowListProject method*), 42
filter() (*bandersnatch_filter_plugins.allowlist_name.AllowListRelease method*), 42
filter() (*bandersnatch_filter_plugins.blocklist_name.BlockListProject method*), 37
filter() (*bandersnatch_filter_plugins.blocklist_name.BlockListRelease method*), 37
filter() (*bandersnatch_filter_plugins.filename_name.ExcludePlatformFilter method*), 37
filter() (*bandersnatch_filter_plugins.latest_name.LatestReleaseFilter method*), 38
filter() (*bandersnatch_filter_plugins.metadata_filter.RegexFilter method*), 38
filter() (*bandersnatch_filter_plugins.metadata_filter.RegexProjectMetadataFilter method*), 38
filter() (*bandersnatch_filter_plugins.metadata_filter.RegexReleaseFileMetadataFilter method*), 39
filter() (*bandersnatch_filter_plugins.metadata_filter.SizeProjectMetadataFilter method*), 39
filter() (*bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter method*), 40
filter() (*bandersnatch_filter_plugins.metadata_filter.VersionRangeProjectMetadataFilter method*), 40
filter() (*bandersnatch_filter_plugins.metadata_filter.VersionRangeReleaseFileMetadataFilter method*), 40
filter() (*bandersnatch_filter_plugins.prerelease_name.PreReleaseFilter method*), 41
filter() (*bandersnatch_filter_plugins.regex_name.RegexProjectFilter method*), 41
filter() (*bandersnatch_filter_plugins.regex_name.RegexReleaseFilter method*), 42
filter_all_releases() (*bandersnatch.package.Package method*), 32
filter_all_releases_files() (*bandersnatch.package.Package method*), 32
filter_metadata() (*bandersnatch.package.Package method*), 32
filter_metadata_plugins() (*bandersnatch.filter.LoadedFilters method*), 28
filter_project_plugins() (*bandersnatch.filter.LoadedFilters method*), 28
filter_release_file_plugins() (*bandersnatch.filter.LoadedFilters method*), 28
filter_release_plugins() (*bandersnatch.filter.LoadedFilters method*), 28
FilterMetadataPlugin (*class in bandersnatch.filter*), 27
FilterProjectPlugin (*class in bandersnatch.filter*), 27
FilterReleaseFilePlugin (*class in bandersnatch.filter*), 27
FilterReleasePlugin (*class in bandersnatch.filter*), 28
finalize_sync() (*bandersnatch.mirror.BandersnatchMirror method*), 30
finalize_sync() (*bandersnatch.mirror.Mirror method*), 31
find() (*bandersnatch.storage.Storage method*), 33
find() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 43
find() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46
find() (*in module bandersnatch.utils*), 35
find_all_files() (*in module bandersnatch.utils*), 35
find_package_indexes_in_dir() (*bandersnatch.mirror.BandersnatchMirror method*), 30
find_target_serial() (*bandersnatch.mirror.BandersnatchMirror method*), 30
flock_path (*bandersnatch.storage.StoragePlugin attribute*), 34
flock_path (*bandersnatch_storage_plugins.swift.SwiftStorage attribute*), 46

G

gen_html_file_tags() (*bandersnatch.mirror.BandersnatchMirror method*), 30
generate_simple_page() (*bandersnatch.mirror.BandersnatchMirror method*), 30

generationfile (*bandersnatch.mirror.BandersnatchMirror property*), 30
get() (*bandersnatch.master.Master method*), 29
get_config_value() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46
get_container() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 46
get_file_size() (*bandersnatch.storage.Storage method*), 33
get_file_size() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 43
get_file_size() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47
get_flock_path() (*bandersnatch.storage.Storage method*), 33
get_hash() (*bandersnatch.storage.Storage method*), 33
get_hash() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 43
get_hash() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47
get_json_paths() (*bandersnatch.storage.Storage method*), 33
get_latest_json() (*in module bandersnatch.verify*), 36
get_lock() (*bandersnatch.storage.Storage method*), 33
get_lock() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
get_lock() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47
get_object() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47
get_package_metadata() (*bandersnatch.master.Master method*), 29
get_requirement_files() (*in module bandersnatch_filter_plugins.allowlist_name*), 43
get_simple_dirs() (*bandersnatch.mirror.BandersnatchMirror method*), 30
get_upload_time() (*bandersnatch.storage.Storage method*), 33
get_upload_time() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
get_upload_time() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47

H

hash() (*in module bandersnatch.utils*), 35
hash_file() (*bandersnatch.storage.Storage method*), 33

I

info (*bandersnatch.package.Package property*), 32
initialize_plugin() (*bandersnatch.filter.Filter method*), 27
initialize_plugin() (*bandersnatch.storage.Storage method*), 33
initialize_plugin() (*bandersnatch_filter_plugins.allowlist_name.AllowListProject method*), 42
initialize_plugin() (*bandersnatch_filter_plugins.allowlist_name.AllowListRelease method*), 42
initialize_plugin() (*bandersnatch_filter_plugins.blocklist_name.BlockListProject method*), 37
initialize_plugin() (*bandersnatch_filter_plugins.blocklist_name.BlockListRelease method*), 37
initialize_plugin() (*bandersnatch_filter_plugins.filename_name.ExcludePlatformFilter method*), 37
initialize_plugin() (*bandersnatch_filter_plugins.latest_name.LatestReleaseFilter method*), 38
initialize_plugin() (*bandersnatch_filter_plugins.metadata_filter.RegexFilter method*), 38
initialize_plugin() (*bandersnatch_filter_plugins.metadata_filter.SizeProjectMetadataFilter method*), 39
initialize_plugin() (*bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter method*), 40
initialize_plugin() (*bandersnatch_filter_plugins.metadata_filter.VersionRangeProjectMetada method*), 40
initialize_plugin() (*bandersnatch_filter_plugins.metadata_filter.VersionRangeReleaseFileMe method*), 41
initialize_plugin() (*bandersnatch_filter_plugins.prerelease_name.PreReleaseFilter method*), 41
initialize_plugin() (*bandersnatch_filter_plugins.regex_name.RegexProjectFilter method*), 41
initialize_plugin() (*bandersnatch_filter_plugins.regex_name.RegexReleaseFilter method*), 42

initialize_plugin() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47
initialized (*bandersnatch_filter_plugins.metadata_filter.RegexFilter attribute*), 38
initialized (*bandersnatch_filter_plugins.metadata_filter.RegexProjectMetadataFilter attribute*), 39
initialized (*bandersnatch_filter_plugins.metadata_filter.RegexReleaseFileMetadataFilter attribute*), 39
initialized (*bandersnatch_filter_plugins.metadata_filter.SizeProjectMetadataFilter attribute*), 39
initialized (*bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter attribute*), 40
initialized (*bandersnatch_filter_plugins.metadata_filter.VersionRangeProjectMetadataFilter attribute*), 40
initialized (*bandersnatch_filter_plugins.metadata_filter.VersionRangeReleaseFileMetadataFilter attribute*), 41
initilize_plugin() (*bandersnatch_filter_plugins.metadata_filter.RegexProjectMetadataFilter method*), 39
initilize_plugin() (*bandersnatch_filter_plugins.metadata_filter.RegexReleaseFileMetadataFilter method*), 39
is_dir() (*bandersnatch.storage.Storage method*), 33
is_dir() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
is_dir() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
is_dir() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47
is_file() (*bandersnatch.storage.Storage method*), 33
is_file() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
is_file() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
is_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 47
is_locked (*bandersnatch_storage_plugins.swift.SwiftFileLock property*), 45
is_symlink() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
is_symlink() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
iter_dir() (*bandersnatch.storage.Storage method*), 33
iterdir() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45

J

json_file() (*bandersnatch.mirror.BandersnatchMirror method*), 30
json_pypi_symlink() (*bandersnatch.mirror.BandersnatchMirror method*), 30
json_save (*bandersnatch.configuration.SetConfigValues attribute*), 26

K

keep (*bandersnatch_filter_plugins.latest_name.LatestReleaseFilter attribute*), 38

L

last_serial (*bandersnatch.package.Package property*), 32
LatestReleaseFilter (*class in bandersnatch_filter_plugins.latest_name*), 38
load_configuration() (*bandersnatch.configuration.BandersnatchConfig method*), 25
load_storage_plugins() (*in module bandersnatch.storage*), 34
LoadedFilters (*class in bandersnatch.filter*), 28

M

main() (*in module bandersnatch.main*), 29
make_time_stamp() (*in module bandersnatch.utils*), 35
Master (*class in bandersnatch.master*), 29
match_patterns (*bandersnatch_filter_plugins.metadata_filter.RegexFilter attribute*), 38
match_patterns (*bandersnatch_filter_plugins.metadata_filter.RegexProjectMetadataFilter attribute*), 39
match_patterns (*bandersnatch_filter_plugins.metadata_filter.RegexReleaseFileMetadataF attribute*), 39
max_package_size (*bandersnatch_filter_plugins.metadata_filter.SizeProjectMetadataFilter attribute*), 40
metadata (*bandersnatch.package.Package property*), 32
metadata_verify() (*in module bandersnatch.verify*), 36
Mirror (*class in bandersnatch.mirror*), 31
mirror() (*in module bandersnatch.mirror*), 32
mkdir() (*bandersnatch.storage.Storage method*), 33
mkdir() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
mkdir() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45

mkdir() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
module
    bandersnatch, 25
    bandersnatch.configuration, 25
    bandersnatch.delete, 27
    bandersnatch.filter, 27
    bandersnatch.log, 28
    bandersnatch.main, 29
    bandersnatch.master, 29
    bandersnatch.mirror, 29
    bandersnatch.package, 32
    bandersnatch.storage, 32
    bandersnatch.utils, 35
    bandersnatch.verify, 36
    bandersnatch_filter_plugins, 37
    bandersnatch_filter_plugins.allowlist_name, 42
    bandersnatch_filter_plugins.blocklist_name, 37
    bandersnatch_filter_plugins.filename_name, 37
    bandersnatch_filter_plugins.latest_name, 38
    bandersnatch_filter_plugins.metadata_filter, 38
    bandersnatch_filter_plugins.prerelease_name, 41
    bandersnatch_filter_plugins.regex_name, 41
    bandersnatch_storage_plugins, 43
    bandersnatch_storage_plugins.filesystem, 43
    bandersnatch_storage_plugins.swift, 45
move_file() (*bandersnatch.storage.Storage method*), 33
move_file() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
move_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48

N

name (*bandersnatch.filter.Filter attribute*), 27
name (*bandersnatch.filter.FilterMetadataPlugin attribute*), 27
name (*bandersnatch.filter.FilterProjectPlugin attribute*), 27
name (*bandersnatch.filter.FilterReleaseFilePlugin attribute*), 28
name (*bandersnatch.filter.FilterReleasePlugin attribute*), 28
name (*bandersnatch.storage.Storage attribute*), 33
name (*bandersnatch.storage.StoragePlugin attribute*), 34
name (*bandersnatch_filter_plugins.allowlist_name.AllowListProject attribute*), 42
name (*bandersnatch_filter_plugins.allowlist_name.AllowListRelease attribute*), 43
name (*bandersnatch_filter_plugins.allowlist_name.AllowListRequirements attribute*), 43
name (*bandersnatch_filter_plugins.allowlist_name.AllowListRequirementsP attribute*), 43
name (*bandersnatch_filter_plugins.blocklist_name.BlockListProject attribute*), 37
name (*bandersnatch_filter_plugins.blocklist_name.BlockListRelease attribute*), 37
name (*bandersnatch_filter_plugins.filename_name.ExcludePlatformFilter attribute*), 38
name (*bandersnatch_filter_plugins.latest_name.LatestReleaseFilter attribute*), 38
name (*bandersnatch_filter_plugins.metadata_filter.RegexFilter attribute*), 38
name (*bandersnatch_filter_plugins.metadata_filter.RegexProjectMetadataFi attribute*), 39
name (*bandersnatch_filter_plugins.metadata_filter.RegexReleaseFileMetada attribute*), 39
name (*bandersnatch_filter_plugins.metadata_filter.SizeProjectMetadataFilt attribute*), 40
name (*bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter attribute*), 40
name (*bandersnatch_filter_plugins.metadata_filter.VersionRangeProjectMet attribute*), 40
name (*bandersnatch_filter_plugins.metadata_filter.VersionRangeReleaseFil attribute*), 41
name (*bandersnatch_filter_plugins.prerelease_name.PreReleaseFilter attribute*), 41
name (*bandersnatch_filter_plugins.regex_name.RegexProjectFilter attribute*), 42
name (*bandersnatch_filter_plugins.regex_name.RegexReleaseFilter attribute*), 42
name (*bandersnatch_storage_plugins.filesystem.FilesystemStorage attribute*), 44
name (*bandersnatch_storage_plugins.swift.SwiftStorage attribute*), 48
need_index_sync (*bandersnatch.mirror.BandersnatchMirror attribute*), 30
need_wrapup (*bandersnatch.mirror.BandersnatchMirror attribute*), 30
now (*bandersnatch.mirror.Mirror attribute*), 31
nulls_match (*bandersnatch_filter_plugins.metadata_filter.RegexFilter attribute*), 38
nulls_match (*bandersnatch_filter_plugins.metadata_filter.RegexProjectMetadataFilter attribute*), 39

nulls_match (*bandersnatch_filter_plugins.metadata_filter.RegexReleaseFileMetadataFilter attribute*), 39
nulls_match (*bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter attribute*), 40
nulls_match (*bandersnatch_filter_plugins.metadata_filter.VersionRangeProjectMetadataFilter attribute*), 40
nulls_match (*bandersnatch_filter_plugins.metadata_filter.VersionRangeReleaseFileMetadataFilter attribute*), 41

O

on_error() (*bandersnatch.mirror.BandersnatchMirror method*), 30
on_error() (*bandersnatch.mirror.Mirror method*), 31
on_error() (*in module bandersnatch.verify*), 36
open_file() (*bandersnatch.storage.Storage method*), 33
open_file() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
open_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48

P

Package (*class in bandersnatch.package*), 32
package_syncer() (*bandersnatch.mirror.Mirror method*), 31
packages_to_sync (*bandersnatch.mirror.Mirror attribute*), 31
parse_version() (*in module bandersnatch.utils*), 35
PATH_BACKEND (*bandersnatch.storage.Storage attribute*), 32
PATH_BACKEND (*bandersnatch_storage_plugins.filesystem.FilesystemStorage attribute*), 43
path_backend (*bandersnatch_storage_plugins.swift.SwiftFileLock property*), 45
PATH_BACKEND (*bandersnatch_storage_plugins.swift.SwiftStorage attribute*), 46
patterns (*bandersnatch_filter_plugins.metadata_filter.RegexFilter attribute*), 38
patterns (*bandersnatch_filter_plugins.metadata_filter.RegexProjectMetadataFilter attribute*), 39
patterns (*bandersnatch_filter_plugins.metadata_filter.RegexReleaseFileMetadataFilter attribute*), 39
patterns (*bandersnatch_filter_plugins.prerelease_name.PreReleaseFilter attribute*), 41
patterns (*bandersnatch_filter_plugins.regex_name.RegexProjectFilter attribute*), 42
patterns (*bandersnatch_filter_plugins.regex_name.RegexReleaseFilter attribute*), 42
pbf_download_urls() (*bandersnatch.mirror.BandersnatchMirror method*), 30
PRERELEASE_PATTERNS (*bandersnatch_filter_plugins.prerelease_name.PreReleaseFilter attribute*), 41
PreReleaseFilter (*class in bandersnatch_filter_plugins.prerelease_name*), 41
process_package() (*bandersnatch.mirror.BandersnatchMirror method*), 31
process_package() (*bandersnatch.mirror.Mirror method*), 31
pypi_repository_version (*bandersnatch.mirror.Mirror attribute*), 31

R

read_bytes() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
read_file() (*bandersnatch.storage.Storage method*), 33
read_file() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
read_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
read_text() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
record_finished_package() (*bandersnatch.mirror.BandersnatchMirror method*), 31
RegexFilter (*class in bandersnatch_filter_plugins.metadata_filter*), 38
RegexProjectFilter (*class in bandersnatch_filter_plugins.regex_name*), 41
RegexProjectMetadataFilter (*class in bandersnatch_filter_plugins.metadata_filter*), 38
RegexReleaseFileMetadataFilter (*class in bandersnatch_filter_plugins.metadata_filter*), 39
RegexReleaseFilter (*class in bandersnatch_filter_plugins.regex_name*), 42
register_backend() (*bandersnatch_storage_plugins.swift.SwiftPath class method*), 45
release_files (*bandersnatch.package.Package property*), 32

release_files_save (*bandersnatch.configuration.SetConfigValues attribute*), 26
releases (*bandersnatch.package.Package property*), 32
removeprefix() (*in module bandersnatch.utils*), 35
rewrite() (*bandersnatch.storage.Storage method*), 34
rewrite() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
rewrite() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
rewrite() (*in module bandersnatch.utils*), 36
rmdir() (*bandersnatch.storage.Storage method*), 34
rmdir() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
rmdir() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
root_uri (*bandersnatch.configuration.SetConfigValues attribute*), 26
rpc() (*bandersnatch.master.Master method*), 29

S

save_json_metadata() (*bandersnatch.mirror.BandersnatchMirror method*), 31
set_upload_time() (*bandersnatch.storage.Storage method*), 34
set_upload_time() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
set_upload_time() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
SetConfigValues (*class in bandersnatch.configuration*), 26
setup_logging() (*in module bandersnatch.log*), 28
SHOWN_DEPRECATIONS (*bandersnatch.configuration.BandersnatchConfig attribute*), 25
simple_directory() (*bandersnatch.mirror.BandersnatchMirror method*), 31
Singleton (*class in bandersnatch.configuration*), 26
SizeProjectMetadataFilter (*class in bandersnatch_filter_plugins.metadata_filter*), 39
specifiers (*bandersnatch_filter_plugins.metadata_filter.VersionRangeFilter attribute*), 40
specifiers (*bandersnatch_filter_plugins.metadata_filter.VersionRangeProjectMetadataFilter attribute*), 40
specifiers (*bandersnatch_filter_plugins.metadata_filter.VersionRangeReleaseFileMetadataFilter attribute*), 41
StalePage, 29
statusfile (*bandersnatch.mirror.BandersnatchMirror property*), 31
Storage (*class in bandersnatch.storage*), 32
storage_backend_name (*bandersnatch.configuration.SetConfigValues attribute*), 26
storage_backend_plugins() (*in module bandersnatch.storage*), 34
StoragePlugin (*class in bandersnatch.storage*), 34
SwiftFileLock (*class in bandersnatch_storage_plugins.swift*), 45
SwiftPath (*class in bandersnatch_storage_plugins.swift*), 45
SwiftStorage (*class in bandersnatch_storage_plugins.swift*), 46
symlink() (*bandersnatch.storage.Storage method*), 34
symlink() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
symlink_to() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 45
sync_index_page() (*bandersnatch.mirror.BandersnatchMirror method*), 31
sync_packages() (*bandersnatch.mirror.Mirror method*), 31
sync_release_files() (*bandersnatch.mirror.BandersnatchMirror method*), 31
sync_simple_page() (*bandersnatch.mirror.BandersnatchMirror method*), 31
synced_serial (*bandersnatch.mirror.Mirror attribute*), 31
synchronize() (*bandersnatch.mirror.Mirror method*), 31

T

target_serial (*bandersnatch.mirror.Mirror attribute*), 31
todolist (*bandersnatch.mirror.BandersnatchMirror property*), 31
touch() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 46

U

unlink() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 46
unlink_parent_dir() (*in module bandersnatch.utils*), 36
update_metadata() (*bandersnatch.package.Package method*), 32
update_safe() (*bandersnatch.storage.Storage method*), 34
update_safe() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44

update_safe() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
update_timestamp() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
url_fetch() (*bandersnatch.master.Master method*), 29
user_agent() (*in module bandersnatch.utils*), 36

V

validate_config_values() (*in module bandersnatch.configuration*), 26
verify() (*in module bandersnatch.verify*), 36
verify_producer() (*in module bandersnatch.verify*), 36
VersionRangeFilter (*class in bandersnatch_filter_plugins.metadata_filter*), 40
VersionRangeProjectMetadataFilter (*class in bandersnatch_filter_plugins.metadata_filter*), 40
VersionRangeReleaseFileMetadataFilter (*class in bandersnatch_filter_plugins.metadata_filter*), 40

W

walk() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
walk() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
webdir (*bandersnatch.mirror.BandersnatchMirror property*), 31
wrapup_successful_sync() (*bandersnatch.mirror.BandersnatchMirror method*), 31
write_bytes() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 46
write_file() (*bandersnatch.storage.Storage method*), 34
write_file() (*bandersnatch_storage_plugins.filesystem.FilesystemStorage method*), 44
write_file() (*bandersnatch_storage_plugins.swift.SwiftStorage method*), 48
write_text() (*bandersnatch_storage_plugins.swift.SwiftPath method*), 46

X

xmlrpc_url (*bandersnatch.master.Master property*), 29
XmlRpcError, 29