bandersnatch Documentation

Release 6.5.0

PyPA

Apr 27, 2024

CONTENTS

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

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

As of 6.0:

- Supports PEP691 - HTML + JSON Simple Index

As of 4.0:

- 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 options

- ***-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 ...]
```

1.2.1 bandersnatch delete positional arguments

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

1.2.2 bandersnatch delete options

- ***-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 options

- ***-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 options

- ***-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 options

- ***-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.

2.2.1.1 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
```

2.2.1.2 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
- Prefix keys with `config_param_` to add the key and its value as parameters to the underlying Boto3 S3 calls

You will need an AWS account and an S3 bucket

2.2.2.1 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
# Optional example for overriding parameters in Boto3 S3 calls
config_param_ServerSideEncryption = aws:kms
config_param_SSEKMSKeyId = your KMS key ID
```

2.2.2.2 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.

2.2.2.3 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.

2.2.2.4 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

2.2.2.5 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

2.2.3.1 Config Example

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

[swift]
default_container = bandersnatch
```

2.2.3.2 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]** section of the configuration file contains general options for how Bandersnatch should operate. This includes settings like the source repository to mirror, how to store mirrored files, and the kinds of files to include in the mirror.

The following options are currently *required*:

- *directory*
- *master*
- *workers*
- *timeout*
- *global-timeout*
- *stop-on-error*
- *hash-index*

2.3.1 Examples

These examples only show `[mirror]` options; a complete configuration may include *mirror filtering plugins* and/or options for a *storage backend*.

2.3.1.1 Minmal

A basic configuration with reasonable defaults for the required options:

```
[mirror]
; base destination path for mirrored files
directory = /srv/pypi

; upstream package repository to mirror
master = https://pypi.org

; parallel downloads - keep low to avoid overwhelming upstream
workers = 3

; per-request time limit
timeout = 15

; global time limit - applied to aiohttp coroutines
global-timeout = 18000

; continue syncing when an error occurs
stop-on-error = false

; use PyPI-compatible folder structure for index files
hash-index = false
```

This will mirror index files and package release files from PyPI and store the mirror in `/srv/pypi`. Add configuration for *mirror filtering plugins* to optionally filter what packages are mirrored in a variety of ways.

2.3.1.2 Alternative Download Source

It is possible to download metadata from one repository, but package release files from another:

```
[mirror]
directory = /srv/pypi
; Project and package metadata received from this repository
master = https://pypi.org
; Package distribution artifacts downloaded from here if possible
download-mirror = https://pypi-mirror.example.com/

; required options from basic config
workers = 3
timeout = 15
global-timeout = 18000
stop-on-error = false
hash-index = false
```

This will download release files from `https://pypi-mirror.example.com` if possible and fall back to PyPI if a download fails. See *download-mirror*. Add *download-mirror-no-fallback* to download release files exclusively from `download-mirror`.

2.3.1.3 Index Files Only

It is possible to mirror just index files without downloading any package release files:

```
[mirror]
directory = /srv/pypi-filtered
master = https://pypi.org
simple-format = ALL
release-files = false
root_uri = https://files.pythonhosted.org/

; required options from basic config
workers = 3
timeout = 15
global-timeout = 18000
stop-on-error = false
hash-index = false
```

This will mirror index files for projects and versions allowed by your *mirror filters*, but will not download any package release files. File URLs in index files will use the configured `root_uri`. See *release-files* and *root_uri*.

2.3.2 Option Reference

2.3.2.1 `directory`

The directory where mirrored files are stored. *This option is always required.*

Type
folder path

Required
yes

The exact interpretation of this value depends on the configured *storage backend*. For the default *filesystem* backend, the directory used should meet the following requirements:

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

2.3.2.2 `storage-backend`

The *storage backend* used to save data and metadata when mirroring packages.

Type
string

Required
no

Default
`filesystem`

See also:

Available storage backends are documented at *Storage options for bandersnatch*.

2.3.2.3 simple-format

The Simple Repository API index file formats to generate.

Type
one of `HTML`, `JSON`, or `ALL`

Required
no

Default
`ALL`

PEP 691 – JSON-based Simple API for Python Package Indexes extended the Simple Repository API to support both HTML and JSON. Bandersnatch generates project index files in both formats by default. Set this option to restrict index files to a single data format.

simple-format index files describes the generated folder structure and file names.

2.3.2.4 release-files

Mirror package release files. Release files are the uploaded sdist and wheel files for mirrored projects.

Type
boolean

Required
no

Default
true

Disabling this will mirror repository *index files* and/or *project metadata* without downloading any associated package files. *release-files folder structure* describes the folder structure for mirrored package release files.

Note: If `release-files = false`, you should also specify the *root_uri* option.

2.3.2.5 json

Save copies of JSON project metadata downloaded from PyPI.

Type
boolean

Required
no

Default
false

When enabled, this saves copies of all JSON project metadata downloaded from PyPI's JSON API. These files are used by the *bandersnatch verify* subcommand.

json API metadata files describes the folder structure generated by this option. The format of the saved JSON is not standardized and is specific to Warehouse.

Note: This option does *not* effect the generation of simple repository API index files in JSON format (*simple-format*).

2.3.2.6 root_uri

A base URL to generate absolute URLs for package release files.

Type
URL

Required
no

Default
`https://files.pythonhosted.org/`

Bandersnatch creates index files containing relative URLs by default. Setting this option generates index files with absolute URLs instead.

If *release-files* is disabled *and* this option is unset, Bandersnatch uses a default value of `https://files.pythonhosted.org/`.

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

2.3.2.7 diff-file

File location to write a list of all new or changed files during a mirror operation.

Type
file or folder path

Required
no

Default
`<mirror directory>/mirrored-files`

Bandersnatch creates a plain-text file at the specified location containing a list of all files created or updated during the last mirror/sync operation. The files are listed as absolute paths separated by blank lines.

This is useful when mirroring to an offline network where it is required to only transfer new files to the downstream mirror. The diff file can be used to copy new files to an external drive, sync the list of files to an SSH destination such as a diode, or send the files through some other mechanism to an offline system.

If the specified path is a directory, Bandersnatch will use the file name "`mirrored-files`" within that directory.

The file will be overwritten on each mirror operation unless *diff-append-epoch* is enabled.

Example Usage

The diff file can be used with rsync for copying only new files:

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

It can also be used with 7zip to create split archives for transfers:

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

2.3.2.8 diff-append-epoch

Append the current epoch time to the file name for *diff-file*.

Type
boolean

Required
no

Default
false

For example, the configuration:

```
[mirror]
; ...
diff-file = /srv/pypi/new-files
diff-append-epoch = true
```

Will generate diff files with names like `/srv/pypi/new-files-1568129735`. This can be used to track diffs over time by creating a new diff file each time Bandersnatch runs.

2.3.2.9 hash-index

Group generated project index folders by the first letter of their normalized project name.

Type
boolean

Required
yes

Enabling this changes the way generated index files are organized. Project folders are grouped into subfolders alphabetically as shown here: *hash-index index files*. This has the effect of splitting up a large `/web/simple` directory into smaller subfolders, each containing a subset of the index files. This can improve file system efficiency when mirroring a very large number of projects, but requires a web server capable of translating Simple Repository API URLs into file paths.

> **Warning:** It is recommended to set this to `false` for full pip/pypi compatibility.
>
> The path structure created by this option is *incompatible* with the Simple Repository API. Serving the generated `web/simple/` folder directly will not work with pip. `hash-index` should only be used with a web server that can translate request URIs into alternative filesystem locations.
>
> Requests for subfolders of `/web/simple` must be re-written using the first letter of the requested project name:
>
> - Requested path: `/simple/someproject/index.html`
> - Translated path: `/simple/s/someproject/index.html`

Example Apache `RewriteRule` Configuration

Configuration like the following is required to use the `hash-index` option with an Apache web server:

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

Example NGINX `rewrite` Configuration

Configuration like the following is required to use `hash-index` with an NGINX web server:

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

2.3.2.10 `master`

The URL of the Python package repository server to mirror.

Type
URL

Required
yes

Bandersnatch requests metadata for projects and packages from this repository server, and downloads package release files from the URLs specified in the received metadata.

To mirror packages from PyPI, set this to `https://pypi.org`.

The URL *must* use the `https:` protocol.

See also:

Bandersnatch can download package release files from an alternative source by configuring a *download-mirror*.

2.3.2.11 `proxy`

Use an HTTP proxy server.

Type
URL

Required
no

Default
none

The proxy server is used when sending requests to a repository server set by the *master* or *download-mirror* option.

See also:

HTTP proxies are supported through the `aiohttp` library. See the aiohttp manual for details on what connection types are supported: https://docs.aiohttp.org/en/stable/client_advanced.html#proxy-support

Note: Alternatively, you can specify a proxy URL by setting one of the environment variables `HTTPS_PROXY`, `HTTP_PROXY`, or `ALL_PROXY`. *This method supports both HTTP and SOCKS proxies.* Support for `socks4`/`socks5` uses the aiohttp-socks library.

SOCKS proxies are not currently supported via the `mirror.proxy` config option.

2.3.2.12 `timeout`

The network request timeout to use for all connections, in seconds. This is the maximum allowed time for individual web requests.

Type
number, in seconds

Required
yes

Note: It is recommended to set this to a relatively low value, e.g. 10 - 30 seconds. This is so temporary problems will fail quickly and allow retrying, instead of having a process hang infinitely and leave TCP unable to catch up for a long time.

2.3.2.13 `global-timeout`

The maximum runtime of individual aiohttp coroutines, in seconds.

Type
number, in seconds

Required
yes

Note: It is recommended to set this to a relatively high value, e.g. 3,600 - 18,000 (1 - 5 hours). This supports coroutines mirroring large package files on slow connections.

2.3.2.14 `download-mirror`

Download package release files from an alternative repository server.

Type
URL

Required
no

Default
none

By default, Bandersnatch downloads packages from the URL supplied in the master server's JSON response. Setting this option to a repository URL will try to download release files from that repository first, and fallback to the URL supplied by the master server if that is unsuccessful (unable to get content or checksum mismatch).

This is useful to sync most of the files from an existing, nearby mirror - for example, when creating a new mirror identical to an existing one for the purpose of load sharing.

2.3.2.15 download-mirror-no-fallback

Disable the fallback behavior for *download-mirror*.

Type
boolean

Required
no

Default
false

When set to `true`, Bandersnatch only downloads package distribution artifacts from the repository set in *download-mirror* and ignores file URLs received from the *master* server.

Warning: This could lead to more failures than expected and is not recommended for most scenarios.

2.3.2.16 cleanup

Enable cleanup of legacy simple directories with non-normalized names.

Type
boolean

Required
no

Default
false

Bandersnatch versions prior to 4.0 used directories with non-normalized package names for compatability with older versions of pip. Enabling this option checks for and removes these directories.

See also:

Python Packaging User Guide - Names and Normalization

2.3.2.17 workers

The number of worker threads used for parallel downloads.

Type
number, 1 N 10

Required
yes

Use **1 - 3** workers to avoid overloading the PyPI master (and maybe your own internet connection). If you see timeouts and have a slow connection, try lowering this setting.

Official servers located in data centers could feasibly run up to 10 workers. Anything beyond 10 is considered unreasonable.

2.3.2.18 verifiers

The number of concurrent consumers used for verifying metadata.

Type
number

Required
no

Default
3

See also:

This option is used by the *bandersnatch verify* subcommand.

2.3.2.19 stop-on-error

Stop mirror/sync operations immediately when an error occurs.

Type
boolean

Required
yes

When disabled (`stop-on-error = false`), Bandersnatch continues syncing after an error occurs, but will mark the sync as unsuccessful. When enabled, Bandersnatch will stop all syncing as soon as possible if an error occurs. This can be helpful when debugging the cause of an unsuccessful sync.

2.3.2.20 compare-method

The method used to compare existing files with upstream files.

Type
one of `hash`, `stat`

Required
no

Default
`hash`

- `hash`: compare by creating a checksums of a local file content. This is slower than `stat`, but more reliable. The hash algorithm is specified by *digest_name*.
- `stat`: compare by using file size and change time. This can reduce IO workload when frequently verifying a large number of files.

2.3.2.21 digest_name

The algorithm used to compute file hashes when *compare-method* is set to `hash`.

Type
one of `sha256`, `md5`

Default
`sha256`

2.3.2.22 keep_index_versions

Store previous versions of generated index files.

Type
number

Required
no

Default
0 (do not keep previous index versions)

This can be used as a safeguard against upstream changes generating blank index.html files.

By default or when set to 0, no prior versions are stored and `index.html` is the latest version.

When enabled by setting a value > 0, Bandersnatch stores the most recently generated versions of each index file, up to the configured number of versions. Prior versions are stored under `versions/index_<serial>_<timestamp>.html` and the current `index.html` is a symlink to the latest version.

2.3.2.23 log-config

Provide a custom logging configuration file.

type
file path

Required
no

Default
none

The file must be a Python `logging.config` module configuration file in INI format, as used with `logging.config.fileConfig`. The specified configuration replaces Bandersnatch's default logging configuration.

See also:

Refer to Configuration file format for the logging configuration file format.

Sample Alternative Logging Configuration

```
[loggers]
keys=root,file
[handlers]
keys=root,file
[formatters]
keys=common
[logger_root]
level=NOTSET
handlers=root
[logger_file]
level=INFO
handlers=file
propagate=1
qualname=bandersnatch
[formatter_common]
format=%(asctime)s %(name)-12s: %(levelname)s %(message)s
[handler_root]
class=StreamHandler
level=DEBUG
formatter=common
args=(sys.stdout,)
[handler_file]
class=handlers.TimedRotatingFileHandler
level=DEBUG
formatter=common
delay=False
args=('/repo/bandersnatch/banderlogfile.log', 'D', 1, 0)
```

2.3.3 Folder Structures

2.3.3.1 `simple-format` index files

Folder structure of generated index files for *simple-format*:

```
<mirror directory>/
└── web/
    ├── packages/...
    └── simple/
        ├── index.html
        ├── index.v1_html
        ├── index.v1_json
        ├── someproject/
        │   ├── index.html
        │   ├── index.v1_html
        │   └── index.v1_json
        ├── anotherproject/
        │   ├── index.html
        │   ├── index.v1_html
        │   └── index.v1_json
        └── ...
```

This path structure is compatible with the Simple Repository API.

If `simple-format` is set to `HTML`, Bandersnatch will only create `index.html` and `index.v1_html`. If `simple-format` is set to `JSON`, it will only create `index.v1_json`.

2.3.3.2 `release-files` folder structure

Package release files are distributed into subdirectories based on their checksum:

```
<mirror directory>/
└── web/
    ├── packages/
    │   ├── 1a/
    │   │   └── 70/
    │   │       └── e63223f8116931d365993d4a6b7ef653a4d920b41d03de7c59499962821f/
    │   │           └── click-8.1.6-py3-none-any.whl
    │   ├── 8b/
    │   │   ├── 3a/
    │   │   │   └── b569b932cf737b525eb4c7a2b615ec07b102dff64f1d8a0fe52a48b911fc/
    │   │   │       └── diff-2023.12.5.tar.gz
    │   │   └── e2/
    │   │       └── 4823d9f02d2743a02e2c236f98b96b52f7a16b2bedc0e3148322dffbd06f/
    │   │           └── black-24.1.0-cp39-cp39-win_amd64.whl
    │   ├── 31/
    │   │   ├── 5f/
    │   │   │   └── ...
    │   │   └── 7a/
    │   │       └── ...
    │   └── ...
    └── simple/
        ├── click/
        ├── diff/
        ├── black/
        ├── ...
        └── index.html
```

By default, generated index files contain releative links into the `web/packages/` directory.

2.3.3.3 `json` API metadata files

Folder structure of saved PyPI project metadata when *json* is enabled:

```
<mirror directory>/
├── web/
│   └── json/
│       ├── someproject
│       ├── anotherproject
│       └── ...
├── pypi/
│   ├── someproject/
│   │   └── json
│   ├── anotherproject/
│   │   └── json
```

(continues on next page)

(continued from previous page)

```
│   └── ...
├── packages/
│   └── ...
└── simple/
    └── ...
```

The files `web/json/someproject` and `web/pypi/someproject/json` both contain the JSON metadata for a PyPI project with the normalized name "someproject".

2.3.3.4 `hash-index` index files

When *hash-index* is enabled, project index folders are grouped by the first letter of their name - for example:

```
<mirror directory>/
└── web/
    └── simple/
        ├── b/
        │   ├── boto3/
        │   │   └── index.html
        │   └── botocore/
        │       └── index.html
        ├── c/
        │   ├── charset-normalizer/
        │   │   └── index.html
        │   ├── certifi/
        │   │   └── index.html
        │   └── cryptography/
        │       └── index.html
        ├── t/
        │   └── typing-extensions/
        │       └── index.html
        ├── ...
        └── index.html
```

The content of the index files themselves is unchanged.

2.3.4 Default Configuration File

Bandersnatch loads default values from a configuration file inside the package. You can use this file as a reference or as the basis for your own configuration.

Listing 1: Default configuration file from `src/bandersnatch/default.conf`

```
[mirror]
; The directory where the mirror data will be stored.
directory = /srv/pypi

; Save JSON metadata into the web tree:
; URL/pypi/PKG_NAME/json (Symlink) -> URL/json/PKG_NAME
json = false
```

(continues on next page)

(continued from previous page)

```
; Save package release files
release-files = true

; Cleanup legacy non PEP 503 normalized named simple directories
cleanup = false

; The PyPI server which will be mirrored.
; master = https://test.python.org
; scheme for PyPI server MUST be https
master = https://pypi.org

; The network socket timeout to use for all connections. This is set to a
; somewhat aggressively low value: rather fail quickly temporarily and re-run
; the client soon instead of having a process hang infinitely and have TCP not
; catching up for ages.
timeout = 10

; The global-timeout sets aiohttp total timeout for it's coroutines
; This is set incredibly high by default as aiohttp coroutines need to be
; equipped to handle mirroring large PyPI packages on slow connections.
global-timeout = 1800

; Number of worker threads to use for parallel downloads.
; Recommendations for worker thread 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 avoided by bandersnatch
workers = 3

; Whether to hash package indexes
; Note that 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.  For example, with the
; following Apache RewriteRule:
;     RewriteRule ^([^/])([^/]*)/$ /mirror/pypi/web/simple/$1/$1$2/
;     RewriteRule ^([^/])([^/]*)/([^/]+)$/ /mirror/pypi/web/simple/$1/$1$2/$3
; OR
; following nginx rewrite rules:
;     rewrite ^/simple/([^/])([^/]*)/$ /simple/$1/$1$2/ last;
;     rewrite ^/simple/([^/])([^/]*)/([^/]+)$/ /simple/$1/$1$2/$3 last;
; Setting this to true would put the package 'abc' index in simple/a/abc.
; Recommended setting: the default of false for full pip/pypi compatibility.
hash-index = false

; Format for simple API to be stored in
; Since PEP691 we have HTML and JSON
simple-format = ALL

; Whether to stop a sync quickly after an error is found or whether to continue
; syncing but not marking the sync as successful. Value should be "true" or
; "false".
```

(continues on next page)

(continued from previous page)

```
stop-on-error = false

; The storage backend that will be used to save data and metadata while
; mirroring packages. By default, use the filesystem backend. Other options
; currently include: 'swift'
storage-backend = filesystem

; Advanced logging configuration. Uncomment and set to the location of a
; python logging format logging config file.
; log-config = /etc/bandersnatch-log.conf

; Generate index pages with absolute urls rather than relative links. This is
; generally not necessary, but was added for the official internal PyPI mirror,
; which requires serving packages from https://files.pythonhosted.org
; root_uri = https://example.com

; Number of consumers which verify metadata
verifiers = 3

; Number of prior simple index.html to store. Used as a safeguard against
; upstream changes generating blank index.html files. Prior versions are
; stored under as "versions/index_<serial>_<timestamp>.html" and the current
; index.html will be a symlink to the latest version.
; If set to 0 no prior versions are stored and index.html is the latest version.
; If unset defaults to 0.
; keep_index_versions = 0

; Configure an option to compare whether a file is identical. By default the
; "hash" method is used which reads local file content and computes hashes,
; which is slow but more reliable; when "stat" method is used, file size and
; change time are used to compare, which is useful to reduce IO workload when
; verifying a lot of files frequently.
; Possible values are: hash (default), stat
compare-method = hash

; Configure to download packages from an alternative mirror.
; By default bandersnatch downloads packages from the server in the "url"
; value of json response from master server. This option asks bandersnatch
; to try to download from the configured PyPI mirror first, and fallback to
; "url" value if it was not successful (unable to get content or checksum
; mismatch). It is useful to sync most of the files from an existing, nearby
; mirror, for example when setting up a new server sitting next to an existing
; one for the purpose of load sharing.
; Downloading only from the mirror site without fallback is also possible,
; but be aware this could lead to more failures than expected and is not
; recommended for most scenarios.
; download-mirror = https://pypi-mirror.example.com/
; download-mirror-no-fallback = False

; vim: set ft=cfg:

; Configure a file to write out the list of files downloaded during the mirror.
```

(continues on next page)

(continued from previous page)

```
; This is useful for situations when mirroring to offline systems where a process
; is required to only sync new files to the upstream mirror.
; The file be be named as set in the diff-file, and overwritten unless the
; diff-append-epoch setting is set to true.  If this is true, the epoch date will
; be appended to the filename (i.e. /path/to/diff-1568129735)
; diff-file = /srv/pypi/mirrored-files
; diff-append-epoch = true
```

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.

Plugins for release version filtering usually act in a way, that releases are only downloaded if all filter plugin rules are satisfied. An exception to this rule is the `project_requirements_pinned` filter: if there is a version number/range specified no other filter are applied. This allows smaller mirrors with newest versions and specifically needed ones.

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
```

Requirements file can be also expressed as a glob file name. In the following example all the requirements files matching the `requirements-*.txt` pattern will be considered and loaded.

```
[plugins]
enabled =
    project_requirements
[allowlist]
requirements_path = /requirements
requirements =
    requirements-*.txt
```

2.4.5.1 Project Regex Matching

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

Example:

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

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

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

2.4.5.2 Release File Regex Matching

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

Example:

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

Valid tags are the same as for projects.

Metadata available to match consists of `info`, `release`, and `release_file` top level structures, with `info` containing the package-wide 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
```

If you only want to filter out the pre-releases for some specific projects (e.g. with nightly updates), list them in the configuration like:

```
[filter_prerelease]
packages =
    duckdb
```

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
sort_by = [version|time]
```

By default, the plugin does not filter out any release. You have to add the `keep` setting. The default is to sort by `version` number (parsed according to semantic versioning). As an alternative, `time` can be used to sort releases chronologically by upload time and select the last `keep` ones.

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]
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`

2.5.1.1 Docker Build

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

2.5.1.2 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`

2.5.1.3 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

2.6.2.1 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

2.6.2.2 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`

2.6.2.3 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
```

- 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.

2.6.2.4 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 minio \
  -v /Users/cooper/tmp/minio:/data \
  minio/minio server /data --console-address ":9001"
```

2.6.3 Creating a Pull Request

2.6.3.1 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'
```

(continues on next page)

(continued from previous page)

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

src/bandersnatch/__init__.py ..                                                                                        ␣
↪                                                                                                                      ␣
↪                                                                                                                [  2%]
src/bandersnatch/buildout.py ..                                                                                        ␣
↪                                                                                                                      ␣
↪                                                                                                                [  4%]
src/bandersnatch/log.py ..                                                                                             ␣
↪                                                                                                                      ␣
↪                                                                                                                [  6%]
src/bandersnatch/main.py ..                                                                                            ␣
↪                                                                                                                      ␣
↪                                                                                                                [  8%]
src/bandersnatch/master.py ..                                                                                          ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 10%]
src/bandersnatch/mirror.py ..                                                                                          ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 12%]
src/bandersnatch/package.py ..                                                                                         ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 14%]
src/bandersnatch/release.py ..                                                                                         ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 17%]
src/bandersnatch/utils.py ..                                                                                           ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 19%]
src/bandersnatch/tests/conftest.py ..                                                                                  ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 21%]
src/bandersnatch/tests/test_main.py .......                                                                            ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 28%]
src/bandersnatch/tests/test_master.py ...........                                                                      ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 40%]
src/bandersnatch/tests/test_mirror.py ..................                                                               ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 61%]
src/bandersnatch/tests/test_package.py ............................                                                    ␣
↪                                                                                                                      ␣
↪                                                                                                                [ 93%]
```

(continues on next page)

(continued from previous page)

```
src/bandersnatch/tests/test_utils.py ......                                                                 ␣
↪                                                                                                           ␣
↪                                                                                                         [100%]

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


=================================================================================================================
↪94 passed in 3.40 seconds␣
↪=================================================================================================================
_____________________________________________________________________________________________________
↪_____________________________________ summary ______________________________________
↪_____________________________________________________________________________________________________
  py36: commands succeeded
  congratulations :)
```

You want to see:

```
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”

2.6.7.1 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

2.7.1.1 Package contents

2.7.1.2 Submodules

2.7.1.3 bandersnatch.configuration module

Module containing classes to access the bandersnatch configuration file

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

Bases: object

SHOWN_DEPRECATIONS = False

check_for_deprecations() → None

load_configuration() → None

Read the configuration from a configuration file

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

Bases: NamedTuple

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

simple_format: SimpleFormat

Alias for field number 11

storage_backend_name: str

Alias for field number 5

class bandersnatch.configuration.**Singleton**

Bases: type

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

2.7.1.4 bandersnatch.delete module

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

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

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

2.7.1.5 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'

pinned_version_exists(*metadata: dict*) → bool

Check if version specifier exist.

Returns
True if version specifier exist, False otherwise

Return type
bool

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

Bases: *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: *Filter*

Plugin that blocks sync operations for an entire project

name = 'project_plugin'

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

Bases: *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: *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[*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[*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[*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[*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*

2.7.1.6 bandersnatch.log module

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

2.7.1.7 bandersnatch.main module

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

bandersnatch.main.**main**(*loop: AbstractEventLoop | None = None*) → int

2.7.1.8 bandersnatch.master module

class bandersnatch.master.**Master**(*url: str*, *timeout: float = 10.0*, *global_timeout: float | None = 18000.0*, *proxy: str | None = 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: int | None*, *got_serial: int | None*) → None

async get(*path: str*, *required_serial: int | None*, ***kw: Any*) → AsyncGenerator[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: Path*, *executor: ProcessPoolExecutor | ThreadPoolExecutor | None = 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: ClientError

Issue getting package listing from PyPI Repository

2.7.1.9 bandersnatch.mirror module

class bandersnatch.mirror.**BandersnatchMirror**(*homedir: Path*, *master: Master*, *storage_backend: str | None = None*, *stop_on_error: bool = False*, *workers: int = 3*, *hash_index: bool = False*, *json_save: bool = False*, *digest_name: str | None = None*, *root_uri: str | None = None*, *keep_index_versions: int = 0*, *diff_file: Path | str | None = None*, *diff_append_epoch: bool = False*, *diff_full_path: Path | str | None = None*, *flock_timeout: int = 1*, *diff_file_list: list[Path] | None = None*, ***, *cleanup: bool = False*, *release_files_save: bool = True*, *compare_method: str | None = None*, *download_mirror: str | None = None*, *download_mirror_no_fallback: bool | None = False*, *simple_format: SimpleFormat | str = 'ALL'*)

Bases: *Mirror*

async **cleanup_non_pep_503_paths**(*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*, *sha256sum: str*, *chunk_size: int = 65536*, *urlpath: str = ''*) → Path | None

errors = False

finalize_sync(*sync_index_page: bool = True*) → None

find_target_serial() → int

property **generationfile: Path**

json_file(*package_name: str*) → Path

json_pypi_symlink(*package_name: str*) → 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: 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: Package*) → Path

property statusfile: Path

async sync_release_files(*package: Package*) → None

Purge + download files returning files removed + added

sync_simple_pages(*package: Package*) → None

property todolist: Path

property webdir: Path

wrapup_successful_sync() → None

write_simple_pages(*package: Package*, *content: SimpleFormats*) → None

class bandersnatch.mirror.**Mirror**(*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, int | str] = {}

async process_package(*package: Package*) → None

async sync_packages() → None

synced_serial: int | None = 0

async synchronize(*specific_packages: list[str] | None = None*, *sync_simple_index: bool = True*) → dict[str, set[str]]

target_serial: int | None = None

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

2.7.1.10 bandersnatch.package module

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

Bases: object

filter_all_releases(*release_filters: list[Filter]*) → bool

Filter releases and removes releases that fail the filters

filter_all_releases_files(*release_file_filters: list[Filter]*) → bool

Filter release files and remove empty releases after doing so.

filter_metadata(*metadata_filters: list[Filter]*) → bool

Run the metadata filtering plugins

property info: 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

2.7.1.11 bandersnatch.storage module

Storage management

class bandersnatch.storage.**Storage**(**args: Any*, *config: ConfigParser | None = None*, ***kwargs: Any*)

Bases: object

Base Storage class

PATH_BACKEND

alias of Path

static canonicalize_package(*name: str*) → str

compare_files(*file1: Path | str*, *file2: Path | str*) → bool

Compare two files and determine whether they contain the same data. Return True if they match

copy_file(*source: Path | str*, *dest: Path | str*) → None

Copy a file from **source** to **dest**

delete(*path: Path | str*, *dry_run: bool = False*) → int

Delete the provided path.

delete_file(*path: Path | str*, *dry_run: bool = False*) → int

Delete the provided path, recursively if necessary.

property directory: str

exists(*path: Path | str*) → bool
Check whether the provided path exists

find(*root: 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: Path | str*) → int
Get the size of a given **path** in bytes

get_hash(*path: Path | str*, *function: str = 'sha256'*) → str
Get the sha256sum of a given **path**

get_json_paths(*name: str*) → Sequence[Path | str]

get_lock(*path: str*) → 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: Path | str*) → datetime
Get the upload time of a given **path**

hash_file(*path: Path | str*, *function: str = 'sha256'*) → str

initialize_plugin() → None
Code to initialize the plugin

is_dir(*path: Path | str*) → bool
Check whether the provided path is a directory.

is_file(*path: Path | str*) → bool
Check whether the provided path is a file.

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

mkdir(*path: Path | str*, *exist_ok: bool = False*, *parents: bool = False*) → None
Create the provided directory

move_file(*source: Path | str*, *dest: Path | str*) → None
Move a file from **source** to **dest**

name = 'storage'

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

rewrite(*filepath: 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: 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.

scandir(*path: Path | str*) → Generator[*StorageDirEntry*, None, None]

Read entries from the provided directory

set_upload_time(*path: Path | str*, *time: datetime*) → None

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

symlink(*source: Path | str*, *dest: Path | str*) → None

Create a symlink at **dest** that points back at **source**

update_safe(*filename: 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: Path | str*, *contents: 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.**StorageDirEntry**(**args*, ***kwargs*)

Bases: Protocol

is_dir() → bool

is_file() → bool

is_symlink() → bool

property name: str | bytes

property path: str | bytes

class bandersnatch.storage.**StoragePlugin**(**args: Any*, *config: ConfigParser | None = None*, ***kwargs: Any*)

Bases: *Storage*

Plugin that provides a storage backend for bandersnatch

name = 'storage_plugin'

bandersnatch.storage.**load_storage_plugins**(*entrypoint_group: str*, *enabled_plugin: str | None = None*, *config: ConfigParser | None = None*, *clear_cache: bool = False*) → set[*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: str | None = 'filesystem'*, *config: ConfigParser | None = None*, *clear_cache: bool = False*) → Iterable[*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*

2.7.1.12 bandersnatch.utils module

class bandersnatch.utils.**StrEnum**(*value*)

Bases: str, Enum

Enumeration class where members can be treated as strings.

value: str

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: 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[Path]*, *base_dir: Path*) → None

bandersnatch.utils.**hash**(*path: 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: str | 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: Path*) → None

Remove a file and if the dir is empty remove it

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

2.7.1.13 bandersnatch.verify module

async bandersnatch.verify.**delete_unowned_files**(*mirror_base: Path*, *executor: ThreadPoolExecutor*, *all_package_files: list[Path]*, *dry_run: bool*) → int

async bandersnatch.verify.**get_latest_json**(*master: Master*, *json_path: Path*, *executor: ThreadPoolExecutor | None = None*, *delete_removed_packages: bool = False*) → None

async bandersnatch.verify.**metadata_verify**(*config: ConfigParser*, *args: 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: Master*, *config: ConfigParser*, *json_file: str*, *mirror_base_path: Path*, *all_package_files: list[Path]*, *args: Namespace*, *executor: ThreadPoolExecutor | None = None*, *releases_key: str = 'releases'*) → None

async bandersnatch.verify.**verify_producer**(*master: Master*, *config: ConfigParser*, *all_package_files: list[Path]*, *mirror_base_path: Path*, *json_files: list[str]*, *args: Namespace*, *executor: ThreadPoolExecutor | None = None*) → None

2.7.2 bandersnatch_filter_plugins package

2.7.2.1 Package contents

2.7.2.2 Submodules

2.7.2.3 bandersnatch_filter_plugins.blocklist_name module

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

Bases: *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: *FilterReleasePlugin*

blocklist_package_names: list[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'

2.7.2.4 bandersnatch_filter_plugins.filename_name module

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

Bases: *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'

2.7.2.5 bandersnatch_filter_plugins.latest_name module

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

Bases: *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'

sort_by = 'version'

2.7.2.6 bandersnatch_filter_plugins.metadata_filter module

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

Bases: *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: *FilterMetadataPlugin*, *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: *FilterReleaseFilePlugin*, *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: *FilterMetadataPlugin*, *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: *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: *FilterMetadataPlugin*, *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: *FilterReleaseFilePlugin*, *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 = {}

2.7.2.7 bandersnatch_filter_plugins.prerelease_name module

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

Bases: *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'

package_names: list[str] = []

patterns: list[Pattern] = []

2.7.2.8 bandersnatch_filter_plugins.regex_name module

class bandersnatch_filter_plugins.regex_name.**RegexProjectFilter**(**args: Any*, ***kwargs: Any*)
Bases: *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: *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] = []

2.7.2.9 bandersnatch_filter_plugins.allowlist_name module

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

Bases: *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: *FilterReleasePlugin*

allowlist_package_names: list[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'

pinned_version_exists(*metadata: dict*) → bool

Check if version specifier exist.

Returns
True if version specifier exist, False otherwise

Return type
bool

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

Bases: *AllowListProject*

name = 'project_requirements'

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

Bases: *AllowListRelease*

name = 'project_requirements_pinned'

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

2.7.3 bandersnatch_storage_plugins package

2.7.3.1 Package contents

2.7.3.2 Submodules

2.7.3.3 bandersnatch_storage_plugins.filesystem module

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

Bases: *StoragePlugin*

PATH_BACKEND

alias of Path

compare_files(*file1: Path | str*, *file2: Path | str*) → bool

Compare two files, returning true if they are the same and False if not.

copy_file(*source: Path | str*, *dest: Path | str*) → None

Copy a file from **source** to **dest**

delete_file(*path: Path | str*, *dry_run: bool = False*) → int

Delete the provided path, recursively if necessary.

exists(*path: Path | str*) → bool

Check whether the provided path exists

find(*root: 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: Path | str*) → int

Return the file size of provided path.

get_hash(*path: Path | str*, *function: str = 'sha256'*) → str

Get the sha256sum of a given **path**

get_lock(*path: str | None = None*) → 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: Path | str*) → datetime

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

is_dir(*path: Path | str*) → bool

Check whether the provided path is a directory.

is_file(*path: Path | str*) → bool

Check whether the provided path is a file.

mkdir(*path: Path | str*, *exist_ok: bool = False*, *parents: bool = False*) → None

Create the provided directory

move_file(*source: Path | str*, *dest: Path | str*) → None

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

name = 'filesystem'

open_file(*path: 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: Path | str*, *text: bool = True*, *encoding: str = 'utf-8'*, *errors: str | None = None*) → str | bytes

Return the contents of the requested file, either a bytestring or a unicode string depending on whether **text** is True

rewrite(*filepath: 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: 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.

scandir(*path: Path | str*) → Generator[DirEntry, None, None]

Read entries from the provided directory

set_upload_time(*path: Path | str*, *time: datetime*) → None

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

update_safe(*filename: 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: Path | str*, *dirs: bool = True*) → list[Path]

write_file(*path: Path | str*, *contents: 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).

2.7.3.4 bandersnatch_storage_plugins.swift module

class bandersnatch_storage_plugins.swift.**SwiftDirEntry**(*entry: dict*)

Bases: object

is_dir() → bool

is_file() → bool

is_symlink() → bool

class bandersnatch_storage_plugins.swift.**SwiftFileLock**(*lock_file: str | os.PathLike[str]*, *timeout: float = -1*, *mode: int = 420*, *thread_local: bool = True*, ***, *is_singleton: bool = False*, ***kwargs: dict[str, Any]*)

Bases: 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[*SwiftPath*]

class bandersnatch_storage_plugins.swift.**SwiftPath**(**args: Any*)

Bases: Path

BACKEND: *SwiftStorage*

absolute() → *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: *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: Connection | None = None*, *recurse: bool = False*, *include_swiftkeep: bool = False*) → Generator[*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: str | None = None*, *errors: str | None = None*) → str

Open the file in text mode, read it, and close the file.

classmethod register_backend(*backend:* SwiftStorage) → None

symlink_to(*target: Path | str*, *target_is_directory: bool = False*, *src_container: str | None = None*, *src_account: str | None = 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: str | None = 'utf-8'*, *errors: str | None = None*) → int

Open the file in bytes mode, write to it, and close the file.

write_text(*data: str*, *encoding: str | None = 'utf-8'*, *errors: str | None = None*, *newline: str | None = None*) → int

Open the file in text mode, write to it, and close the file.

class bandersnatch_storage_plugins.swift.**SwiftStorage**(**args: Any*, *config: ConfigParser | None = None*, ***kwargs: Any*)

Bases: *StoragePlugin*

PATH_BACKEND

alias of *SwiftPath*

compare_files(*file1: Path | str*, *file2: Path | str*) → bool

Compare two files, returning true if they are the same and False if not.

connection() → Generator[Connection, None, None]

copy_file(*source: Path | str*, *dest: Path | str*, *dest_container: str | None = None*) → None

Copy a file from **source** to **dest**

copy_local_file(*source: Path | str*, *dest: Path | str*) → None

Copy the contents of a local file to a destination in swift

property default_container: str

delete_file(*path: Path | str*, *dry_run: bool = False*) → int

Delete the provided path, recursively if necessary.

property directory: str

exists(*path: Path | str*) → bool

Check whether the provided path exists

find(*root: Path | str*, *dirs: bool = True*) → str

A test helper simulating 'find'.

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

get_config_value(*config_key: str*, **env_keys: Any*, *default: str | None = None*) → str | None

get_container(*container: str | None = 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: Path | str*) → int

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

get_hash(*path: Path | str*, *function: str = 'sha256'*) → str

Get the sha256sum of a given **path**

get_lock(*path: str | None = None*) → *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: Path | str*) → datetime

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

initialize_plugin() → None

Code to initialize the plugin

is_dir(*path: Path | str*) → bool

Check whether the provided path is a directory.

is_file(*path: Path | str*) → bool

Check whether the provided path is a file.

is_symlink(*path: Path | str*) → bool

Check whether the provided path is a symlink

mkdir(*path: 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: Path | str*, *dest: Path | str*, *dest_container: str | None = None*) → None

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

name = 'swift'

open_file(*path: 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: Path | str*, *text: bool = True*, *encoding: str = 'utf-8'*, *errors: str | None = None*) → 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: 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: 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.

scandir(*path: Path | str*) → Generator[*SwiftDirEntry*, None, None]

Read entries from the provided directory

set_upload_time(*path: Path | str*, *time: datetime*) → None

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

symlink(*src: Path | str*, *dest: Path | str*, *src_container: str | None = None*, *src_account: str | None = None*) → None

Create a symlink at **dest** that points back at **source**

update_safe(*filename: 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: Path | str*) → None

walk(*root: Path | str*, *dirs: bool = True*, *conn: Connection | None = None*) → list[*SwiftPath*]

write_file(*path: Path | str*, *contents: str | bytes | IO*, *encoding: str | None = None*, *errors: str | None = None*) → None

Write data to the provided path. If **contents** is a string, the file will be opened and written in "r" + "utf-8" mode, if bytes are supplied it will be accessed using "rb" mode (i.e. binary write).

PYTHON MODULE INDEX

b

INDEX

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X