hush

unknown

Dec 20, 2021

TABLE OF CONTENTS

A Python library that helps manage secrets using tools specified by plugin hooks.

CHAPTER

ONE

HUSH

1.1 hush package

A Python library that helps manage secrets.

Uses the secret management tools (e.g. pass) specified by (internal and external) plugin hooks.

class Hush(*namespace=()*, ***, *user=None*)

Bases: `object`

Hush class to constrain context of get_secret() function.

Can be used as an alternative to calling this module's global `get_secret()` function directly.

Parameters

- **namespace** (`Iterable[str]`) –
- **user** (`Optional[str]`) –

get_secret(*key*, *namespace=()*, ***, *user=None*)

Given a `key`, retrive a secret.

Note:

- The `namespace` argument, if provided, will be used to extend the namespace specified when initializing this class.
- The `user` argument, if provided, will override the user specified when initializing this class.

Refer to `help(hush.get_secret)` for more information.

Parameters

- **key** (`str`) –
- **namespace** (`Iterable[str]`) –
- **user** (`Optional[str]`) –

Return type `Optional[str]`

get_secret(*key*, *namespace=()*, ***, *user=None*)

Given a `key`, retrieve a secret.

This function attempts to use every secret-retrieving method registered by plugins (internal and external) to obtain the desired secret.

Parameters

- **key** (str) – The key that corresponds to the secret we are hoping to retrieve.
- **namespace** (Iterable[str]) – The namespace that the secret belongs to (e.g. *["db", "foo-bar"]*). How this argument is used is specific to the tool being used to store and retrieve secrets (i.e. is specific to each hook implementation).
- **user** (Optional[str]) – If this argument is provided, secret retrieving commands are run as `user` when possible. This option defaults to the value of the HUSH_USER envvar, if defined.

Return type Optional[str]

Returns

The secret value returned by the first plugin to successfully retrieve the desired secret.

OR

None, if none of the registered plugins were able to retrieve the desired secret.

1.1.1 Subpackages

hush.plugin package

Plugin package.

This package sets up a plugin architecture using pluggy.

See pluggy's documentation[1] for more information.

[1]: https://pluggy.readthedocs.io/en/stable

manager()

Returns the PluginManager[1] responsible for configuring plugins.

[1]: https://pluggy.readthedocs.io/en/stable/api_reference.html#pluggy.PluginManager

Return type PluginManager

1.1.2 Submodules

hush.cli module

Contains the hush package's main entry point.

parse_cli_args(*argv*)

Parses command-line arguments.

Parameters **argv** (Sequence[str]) –

Return type _Pydantic_Arguments_94782378374096

run(*args*)

This function acts as this tool's main entry point.

Parameters **args** (_Pydantic_Arguments_94782378374096) –

Return type int

CHAPTER

TWO

CHANGELOG FOR `HUSH`

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

2.1 Unreleased

No notable changes have been made.

2.2 0.3.1 - 2021-10-14

2.3 Miscellaneous

- Refactored codebase to use new `get_plugin_modules()` function to collect builtin plugins.
- Prefix private module/package names with an underscore.
- Improve module design so nested imports are no longer necessary.

2.4 0.3.0 - 2021-10-06

2.4.1 Added

- Added `hush.Hush` class.

2.4.2 Miscellaneous

- Factor `pass_store.py` and `envvars.py` out of `builtin.py`.

2.5 0.2.0 - 2021-10-04

2.5.1 Added

- Add `hush.get_secret()` method.
- Add `hush` executable script.

2.6 0.1.0 - 2021-10-03

2.6.1 Miscellaneous

- First release.

CHAPTER

THREE

CONTRIBUTING

3.1 How to submit feedback?

The best way to submit feedback is to file an issue.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :).

3.2 Developer's Guide

3.2.1 Badges

Every badge shown below corresponds with a development tool used to maintain this project. Every badge is clickable and links back to that project's github / documentation site:

tools / frameworks used by test suite (i.e. used by ``make test``):

linters used to maintain code quality (i.e. used by ``make lint``):

tools / frameworks used to render documentation (i.e used by ``make build-docs``):

miscellaneous tools used to maintain this project:

3.2.2 Basic Usage

Before making a PR please run the following

- Optional one time setup: run `make use-docker` if you need to build/test this with docker
- `make lint` to check for any format or convention issues
- `make test` to run all tests

3.2.3 How do I . . . ?

3.3 New Releases

This section serves as a reminder to the maintainers of this project on how to release a new version of this package to PyPI.

Make sure all your changes are committed, that you have added a new section to the CHANGELOG.md file, and that you have bumpversion installed. Then run:

```
bumpversion patch  # possible values: major / minor / patch
git push
git push --tags
```

A new version of `hush` will then deploy to PyPI if all CI checks pass.

CHAPTER

FOUR

HUSH

A Python library that helps manage secrets using tools specified by plugin hooks.

project status badges:

version badges:

4.1 Installation

To install `python-hush` using pip, run the following commands in your terminal:

```
python3 -m pip install --user python-hush  # install hush
```

If you don't have pip installed, this Python installation guide can guide you through the process.

4.2 CLI Usage

4.3 Useful Links

- API Reference: A developer's reference of the API exposed by this project.
- cc-python: The cookiecutter that was used to generate this project. Changes made to this cookiecutter are periodically synced with this project using cruft.
- CHANGELOG.md: We use this file to document all notable changes made to this project.
- CONTRIBUTING.md: This document contains guidelines for developers interested in contributing to this project.

- Create a New Issue: Create a new GitHub issue for this project.
- Documentation: This project's full documentation.
- Usage: How do I use Hush?
- Writing Plugins: How do I write my own plugins for Hush?

CHAPTER

FIVE

USAGE

Hush can be thought of as a secrets manager, but it is more appropriate to think of it as a *manager* of other secret managers.

Keep in mind the following notes while reading the rest of the documentation:

- This package provides both a library and an executable, named `hush`, that can be used to test this package's functionality.
- Hush has multiple builtin plugins which are enabled by default (i.e. we will attempt to use them, by default, when a user requests secret retrieval).
- In the examples in this documentation, we will make use of a builtin plugin that reads secrets from environment variables.

5.1 Using the Library

The `hush` library provides two public methods to access its functionality: the `hush.get_secret()` function and the `hush.Hush()` class. We demonstrate how each can be used in the examples below.

5.1.1 Examples

```
import os

from hush import Hush, get_secret


# To retrieve a secret we must provide Hush with a key to associate with that
# secret. Below, that key is 'foobar'.
os.environ["FOOBAR"] = "Kung Fooooo!"
secret = get_secret("foobar")
print(secret)  # output: Kung Fooooo!

# A secret can optionally belong to a particular namespace. A namespace is a
# listing of names that are generally combined with the key somehow, but
# ultimately it is up to each plugin to decide how it wants to handle namespaces
# (if it chooses to handle them at all).
os.environ["DB_DEV_FOOBAR"] = "Database in Development!"
secret = get_secret("foobar", ["db", "dev"])
print(secret)  # output: Database in Development!
```

(continues on next page)

(continued from previous page)

```
# The Hush class can be used to constrain the context (i.e. paramaters) for the
# `get_secret()` function (which the `Hush.get_secret()` method wraps).
hush = Hush(namespace=["db", "dev"])
secret = hush.get_secret("foobar")
print(secret)  # output: Database in Development!
```

5.2 Using the hush Script

This package also comes with an executable script, hush, that can be used to invoke Hush from the command-line.

5.2.1 Examples

Add secrets using environment variables:

```
$ export FOOBAR="Kung Fooooo!"
$ export DB_DEV_FOOBAR="Database in Development!"
```

Use hush to retrieve those secrets:

```
$ hush foobar
Kung Fooooo!

$ hush --namespace=db,dev foobar
Database in Development!
```

CHAPTER

SIX

HOW TO WRITE YOUR OWN PLUGIN FOR HUSH?

Hush uses pluggy to manage its plugin system. As such, the instructions for creating a new external plugin for Hush are mostly the same as they are for any other application with a plugin-architecture that uses pluggy (e.g. pytest).

6.1 Examples

A good, general-purpose (i.e. not specific to Hush) example of writing external plugins using pluggy can be found here. The example below is very similar but is specific to Hush. Namely, the `hush-tmp-secrets` plugin defined below allows users to start storing secrets in plain-text files under the `/tmp/secrets` directory.

6.1.1 hush-tmp-secrets/hush_tmp_secrets.py

```
"""This module contains the hush-tmp-secrets Hush plugin's implementation."""

from pathlib import Path
from typing import Optional

from hush.plugin import hookimpl


@hookimpl
def get_secret(key: str) -> Optional[str]:
    """Get secrets by searching through the /tmp/secrets directory."""
    key_filename = f"{key}.txt"

    key_full_path = Path("/tmp/secrets") / key_filename
    if key_full_path.exists():
        return key_full_path.read_text().strip()

    return None
```

WARNING: This is just a toy example. A better, more realistic implementation of this plugin would probably (at the very least) verify the `/tmp/secrets` directory's permissions and use some form of encryption instead of storing secrets in unencrypted text files.

6.1.2 hush-tmp-secrets/setup.py

```
from setuptools import setup


setup(
    name="python-hush-tmp-secrets",
    install_requires="python-hush",
    entry_points={"hush": ["tmp_secrets = hush_tmp_secrets"]},
    py_modules=["hush_tmp_secrets"],
)
```

6.1.3 Using the hush-tmp-secrets Plugin

```
$ python3 -m pip install --editable /path/to/hush-tmp-secrets
$ echo "MOOOOO!" > /tmp/secrets/cow.txt
$ hush cow
MOOOOO!
```

> **NOTE**: See the Usage section of this documentation for more information on the `hush` script, which is used in the code-block above.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

h

INDEX