# hashivaultlib Documentation

***Release 1.3.3***

**Costas Tyfoxylos**

**Jun 08, 2021**

# Contents

Contents:

CHAPTER 1

# hashivaultlib

An extension to hvac, implementing recursive removal and retrieval of secrets and models for tokens and policies.

- Documentation: https://hashivaultlib.readthedocs.org/en/latest

## 1.1 Development Workflow

The workflow supports the following steps

- lint
- test
- build
- document
- upload
- graph

These actions are supported out of the box by the corresponding scripts under _CI/scripts directory with sane defaults based on best practices. Sourcing setup_aliases.ps1 for windows powershell or setup_aliases.sh in bash on Mac or Linux will provide with handy aliases for the shell of all those commands prepended with an underscore.

The bootstrap script creates a .venv directory inside the project directory hosting the virtual environment. It uses pipenv for that. It is called by all other scripts before they do anything. So one could simple start by calling _lint and that would set up everything before it tried to actually lint the project

Once the code is ready to be delivered the _tag script should be called accepting one of three arguments, patch, minor, major following the semantic versioning scheme. So for the initial delivery one would call

```
$ _tag –minor
```

which would bump the version of the project to 0.1.0 tag it in git and do a push and also ask for the change and automagically update HISTORY.rst with the version and the change provided.

So the full workflow after git is initialized is:

- repeat as necessary (of course it could be test - code - lint :) ) * code * lint * test
- commit and push
- develop more through the code-lint-test cycle
- tag (with the appropriate argument)
- build
- upload (if you want to host your package in pypi)
- document (of course this could be run at any point)

## 1.2 Important Information

This template is based on pipenv. In order to be compatible with requirements.txt so the actual created package can be used by any part of the existing python ecosystem some hacks were needed. So when building a package out of this **do not** simple call

```
$ python setup.py sdist bdist_egg
```

**as this will produce an unusable artifact with files missing.** Instead use the provided build and upload scripts that create all the necessary files in the artifact.

## 1.3 Project Features

- TODO

CHAPTER 2

# Installation

At the command line:

```
$ pip install hashivaultlib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv hashivaultlib
$ pip install hashivaultlib
```

Or, if you are using pipenv:

```
$ pipenv install hashivaultlib
```

# CHAPTER 3

# Usage

To develop on hashivaultlib:

```
# The following commands require pipenv as a dependency

# To lint the project
_CI/scripts/lint.py

# To execute the testing
_CI/scripts/test.py

# To create a graph of the package and dependency tree
_CI/scripts/graph.py

# To build a package of the project under the directory "dist/"
_CI/scripts/build.py

# To see the package version
_CI/scipts/tag.py

# To bump semantic versioning [--major|--minor|--patch]
_CI/scipts/tag.py --major|--minor|--patch

# To upload the project to a pypi repo if user and password are properly provided
_CI/scripts/upload.py

# To build the documentation of the project
_CI/scripts/document.py
```

To use hashivaultlib in a project:

```
from hashivaultlib import Vault
vault = Vault(url, token)

# Recursivelly retrieve all secrets under a path
```

(continues on next page)

(continued from previous page)

```
secrets = vault.retrieve_secrets_from_path('secrets/passwords')

# After editing the secrets they can be put back
vault.restore_secrets(secrets)

# Paths can also be moved to a new location.
# Each secret has an "original_path" attribute that can be manipulated
secrets = vault.retrieve_secrets_from_path('secrets/passwords')
for secret in secrets:
    secret.original_location = secret.original_location.replace('old_path', 'new_path
↪')
vault.restore_secrets(secrets)

# Recursivelly delete everything under a path
vault.delete_path('secrets/path_to_delete')

# Work with tokens
for token in vault.tokens:
    print(token.display_name)

# Delete all non root tokens
for token in vault.tokens:
    if 'root' not in token.policies:
        token.delete()
```

CHAPTER 4

# Contributing

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

## 4.1 Submit Feedback

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.

### 4.1.1 Get Started!

Ready to contribute? Here's how to set up *hashivaultlib* for local development.

1. Clone your fork locally:

```
$ git clone git@github.com:schubergphilis/hashivaultlib.git
```

2. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your clone for local development:

```
$ mkvirtualenv hashivaultlib
$ cd hashivaultlib/
$ python setup.py develop
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. Commit your changes and push your branch to the server:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

5. Submit a merge request

CHAPTER 5

# hashivaultlib

## 5.1 hashivaultlib package

### 5.1.1 Submodules

### 5.1.2 hashivaultlib.hashivaultlib module

Main code for hashivaultlib.

**class** hashivaultlib.hashivaultlib.**BrokenToken**(*vault_instance*, *data*)
Bases: *hashivaultlib.hashivaultlib.Token*

Models a broken token with only an accessor ID and errors messages.

**errors**
The errors of the token.

**class** hashivaultlib.hashivaultlib.**Token**(*vault_instance*, *data*)
Bases: object

Models a vault token and provides delete capabilities.

**accessor**
The accessor token of the token.

**Returns** The accessor token of the token

**Return type** string

**auth**
Auth data for the token.

**Returns** The auth data

**creation_time**
The creation time of the token in seconds.

**Returns** The creation time of the token in seconds

**Return type** string

**creation_time_day_format**
The creation time of the token in a day duration format.

**Returns** The creation time of the token in a day duration format

**Return type** string

**creation_ttl**
The creation ttl of the token in seconds.

**Returns** The creation ttl of the token in seconds

**Return type** string

**creation_ttl_day_format**
The creation ttl of the token in a day duration format.

**Returns** The creation ttl of the token in a day duration format

**Return type** string

**delete**()
Deletes the token by removing the accessor from the vault instance.

**display_name**
The display name of the token.

**Returns** The display name of the token

**Return type** string

**expire_time**
The expire time of the token.

**Returns** The expire time of the token if any, None otherwise

**Return type** datetime

**explicit_max_ttl**
The explicit max ttl.

**Returns** The explicit max ttl

**Return type** string

**explicit_max_ttl_day_format**
The explicit max ttl in a day duration format.

**Returns** The explicit max ttl in a day duration format

**Return type** string

**id**
The id of the token.

**Returns** The id of the token

**Return type** string

**issue_time**
The issue time of the token.

**Returns** The issue time of the token

**Return type** datetime

**lease_duration**
The duration of the lease of the token.

**Returns** The duration of the lease of the token

**Return type** string

**lease_id**
The lease ID.

**Returns** The lease ID

**Return type** string

**meta**
The meta of the token.

**Returns** The meta of the token

**Return type** string

**num_uses**
The number of uses of the token.

**Returns** The number of uses of the token

**Return type** string

**orphan**
Flag on whether the token is orphan.

**Returns** True if the token is orphan, False otherwise

**Return type** bool

**path**
The path to create the token.

**Returns** The path to create the token

**Return type** string

**policies**
The policies this token has enforced upon.

**Returns** The policies of the token

**Return type** list

**raw_data**
The raw data of the token.

**Returns** The raw data of the token

**Return type** dict

**renewable**
A flag on whether the token is renewable.

**Returns** True if token is renewable, False otherwise

**Return type** bool

**request_id**
The id of the request for the token.

**Returns** The id of the request for the token

**Return type** string

**ttl**
The ttl is seconds.

**Returns** The ttl is seconds

**Return type** string

**ttl_day_format**
The ttl in a day duration format.

**Returns** The ttl in a day duration format

**Return type** string

**warnings**
The warnings of the token.

**Returns** The warnings of the token

**wrap_info**
The wrap info of the token.

**Returns** The wrap info of the token

**class** hashivaultlib.hashivaultlib.**TokenFactory**
Bases: object

Factory to create the appropriate Token type.

**class** hashivaultlib.hashivaultlib.**Vault**(**args*, ***kwargs*)
Bases: hvac.v1.Client

Extends the hvac client for vault with some extra handy usability.

**delete_path**(*path*)
Deletes recursively a path from vault.

**Parameters** **path** – The path to remove

**restore_secrets**(*secrets*)
Restores secrets to vault in their original path.

**Parameters** **secrets** – List of secret dictionaries with "original_path" attribute set

**Returns** True on success, False otherwise

**retrieve_secrets_from_path**(*path*)
Retrieves recursively all the secrets from a path in vault.

**Parameters** **path** – The path to retrieve all the secrets for

**tokens**
Models the tokens of a vault installation.

**Returns** All tokens of a vault in a Token object format

**Return type** list

### 5.1.3 hashivaultlib.hashivaultlibexceptions module

Custom exception code for hashivaultlib.

### 5.1.4 Module contents

hashivaultlib package.

Import all parts from hashivaultlib here

CHAPTER 6

# Credits

## 6.1 Development Lead

- Costas Tyfoxylos <ctyfoxylos@schubergphilis.com>

## 6.2 Contributors

- Dario Tišlar <dtislar@schubergphilis.com>

CHAPTER 7

# History

# CHAPTER 8

## 0.1.0 (25-05-2018)

- First release

# CHAPTER 9

## 0.2.0 (30-07-2018)

- Added concurrency in retrieving the tokens from the server.

# CHAPTER 10

## 1.0.0 (16-10-2018)

- Ported the template part to python 3.7
- Officially dropped support for python 2.7

# CHAPTER 11

## 1.0.2 (25-10-2018)

- Updated template and dependencies

CHAPTER 12

## 1.1.0 (29-11-2018)

- Updated to 0.7.0 of hvac

# CHAPTER 13

## 1.1.1 (12-02-2019)

- Updated hvac to 0.7.2

# CHAPTER 14

## 1.1.2 (12-02-2019)

- Added pyhcl as a top level dependency

# CHAPTER 15

## 1.1.3 (22-05-2019)

- Fixed retrieval and deletion of paths under windows clients

# CHAPTER 16

## 1.1.4 (22-05-2019)

- fixed packaging by removing unneeded files from root

# CHAPTER 17

## 1.1.5 (18-10-2019)

- Updated template and bumped dependencies

# CHAPTER 18

## 1.1.6 (18-10-2019)

- Updated template and bumped dependencies

# CHAPTER 19

## 1.1.7 (25-05-2020)

- Updated dependencies, providing terraform 12 compatibility.

CHAPTER 20

## 1.1.8 (25-05-2020)

- Bumped dependencies

# CHAPTER 21

## 1.2.0 (16-10-2020)

- Implemented retrieve, restore and delete for v2 kv paths.

# CHAPTER 22

## 1.2.1 (16-10-2020)

- Refactored loggers to not use the root one.

# CHAPTER 23

## 1.3.0 (19-10-2020)

- Wrapped Invalid path exception.

CHAPTER 24

## 1.3.1 (28-12-2020)

- Updated underlying hvac dependency.

# CHAPTER 25

## 1.3.2 (26-04-2021)

- Bumped dependencies.

# CHAPTER 26

## 1.3.3 (26-04-2021)

- Bumped dependencies.

CHAPTER 27

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## h

# Index