enpassreadercli Documentation

Release 0.3.1

Costas Tyfoxylos

Mar 02, 2023

CONTENTS

Contents:

CHAPTER

ONE

ENPASSREADERCLI

A cli to access enpass 6 encrypted databases and read, list and search values.

- Documentation: https://enpassreadercli.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

TWO

INSTALLATION

At the command line:

```
$ pip install enpassreadercli
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv enpassreadercli
$ pip install enpassreadercli
```

Or, if you are using pipenv:

```
$ pipenv install enpassreadercli
```

Or, if you are using pipx:

```
$ pipx install enpassreadercli
```

Important note for pysqlcipher3:

pysqlcipher3 needs to compile on your workstation and it might not succeed if header files are missing. On my Mac I had to follow something like the below process:

```
brew install sqlcipher
# Assuming the version installed is 4.4.3 adjust accordingly
export C_INCLUDE_PATH=$BREWPATH/Cellar/sqlcipher/4.4.3/include
export LIBRARY_PATH=$BREWPATH/Cellar/sqlcipher/4.4.3/lib
# Activate the virtual environment that the project is installed to fix the installation␣
↪if broken
. .venv/bin/activate
pip install pysqlcipher3
```

On linux based systems probably sqlcipher-dev will need to be installed for the package to successfully compile.

CHAPTER

THREE

USAGE

To develop on enpassreadercli:

```
# 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/scripts/tag.py

# To bump semantic versioning [--major|--minor|--patch]
_CI/scripts/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 enpassreadercli from the console:

```
# Environment variables supported for required default arguments are:
#
# ENPASS_DB_PATH for the database path
# ENPASS_DB_PASSWORD for the database password
# ENPASS_DB_KEY_FILE for the key file if used.
#
# if any of the above are set the arguments can be omitted from the cli.

enpass-reader --help
usage: enpass-reader [-h] [--log-config LOGGER_CONFIG]
                         [--log-level {debug,info,warning,error,critical}] -d
```

(continues on next page)

(continued from previous page)

```
                        PATH -p PASSWORD [-k KEY_FILE] (-g ENTRY | -e | -s)

A cli to access enpass 6 encrypted databases and read, list and search values.

optional arguments:
  -h, --help            show this help message and exit
  --log-config LOGGER_CONFIG, -l LOGGER_CONFIG
                        The location of the logging config json file
  --log-level {debug,info,warning,error,critical}, -L {debug,info,warning,error,critical}
                        Provide the log level. Defaults to info.
  -d PATH, --database-path PATH
                        Specify the path to the enpass database. (Can also be
                        specified using "ENPASS_DB_PATH" environment variable)
  -p PASSWORD, --database-password PASSWORD
                        Specify the password to the enpass database. (Can also
                        be specified using "ENPASS_DB_PASSWORD" environment
                        variable)
  -k KEY_FILE, --database-key-file KEY_FILE
                        Specify the path to the enpass database key file if
                        used. (Can also be specified using
                        "ENPASS_DB_KEY_FILE" environment variable)
  -g ENTRY, --get ENTRY
                        The name of the entry to get the password of.
  -e, --enumerate       List all the passwords in the database.
  -s, --search          Interactively search for an entry in the database and
                        return that password.
  -f, --fuzzy-search    Interactively fuzzy search for an entry in the
                        database and return that password.


# Getting one password
enpass-reader -d PATH_TO_DATABASE -p PASSWORD -g some-password-name
> password-value

# Enumerate all passwords
enpass-reader -d PATH_TO_DATABASE -p PASSWORD -e
> password1-name: password1-value
> password2-name: password2-value
> password3-name: password3-value
> password4-name: password4-value

# Search interactively for a password
enpass-reader -d PATH_TO_DATABASE -p PASSWORD -s
> Title : (interactive prompt with wildcard searching and autocompletion
# after choosing a password from the autocompleted list
> password-value-for-search-entry

# Search interactively with fuzzy search for a password
enpass-reader -d PATH_TO_DATABASE -p PASSWORD -f
> Title : (interactive prompt with fuzzy searching and autocompletion
# after choosing a password from the autocompleted list
> password-value-for-search-entry
```

CHAPTER

FOUR

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 *enpassreadercli* for local development. Using of pipenv is highly recommended.

1. Clone your fork locally:

```
$ git clone https://github.com/costastf/enpassreadercli
```

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

```
$ cd enpassreadercli/
$ pipenv install --ignore-pipfile
```

3. Create a branch for local development:

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

   Now you can make your changes locally. Do your development while using the CI capabilities and making sure the code passes lint, test, build and document stages.

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

FIVE

ENPASSREADERCLI

5.1 enpassreadercli package

5.1.1 Submodules

5.1.2 enpassreadercli.enpassreadercli module

5.1.3 enpassreadercli.enpassreadercliexceptions module

Custom exception code for enpassreadercli.

5.1.4 Module contents

enpassreadercli package.

Import all parts from enpassreadercli here

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Costas Tyfoxylos <costas.tyf@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

CHAPTER

EIGHT

0.0.1 (27-03-2021)

- First code creation

CHAPTER

NINE

0.1.0 (27-03-2021)

- Initial functionality.

CHAPTER

TEN

0.1.1 (27-03-2021)

- Reverted to default styling for search function.

CHAPTER

ELEVEN

0.2.0 (27-03-2021)

- Added fuzzy searching capabilities.

CHAPTER

TWELVE

0.2.1 (07-07-2021)

- Added pipeline and bumped dependencies.

CHAPTER

THIRTEEN

0.3.0 (02-03-2023)

- Implement totp retrieval.

CHAPTER

FOURTEEN

0.3.1 (02-03-2023)

- Calculate actual totp from provided seed and return that.

CHAPTER

FIFTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

E

M