# config-sesame Documentation

***Release 0.1.0***

**1**
**1 Group**

2016-06-30

Contents

*A tool to look up secrets from a vault based on existing shareable configuration.*

CHAPTER 1

# Introduction

The `config-sesame` command line tool can be used as part of a continuous deployment pipeline to provide applications with runtime secrets. For this purpose, it scans already assembled application configuration (`application.yml`) for references to secrets stored in a "vault", and writes resolved secrets to an additional file (`secrets.yml`). See Using Config Sesame for more.

Read Overview of Config Sesame to get to know the ideas behind the design of the tool.

CHAPTER 2

# Important Links

- GitHub Project
- Issue Tracker
- PyPI
- Latest Documentation

CHAPTER 3

# Installing

*Config Sesame* can be installed from PyPI via `pip install config-sesame` as usual, see releases on GitHub for an overview of available versions – the project uses semantic versioning and follows PEP 440 conventions.

To get a bleeding-edge version from source, use these commands:

```
repo="1and1/config-sesame"
pip install -r "https://raw.githubusercontent.com/$repo/master/requirements.txt"
pip install -U -e "git+https://github.com/$repo.git#egg=${repo#*/}"
```

See Installing Config Sesame for detailed setup and configuration instructions.

To add bash completion, read the Click docs about it, or just follow these instructions:

```
cmdname=config-sesame
mkdir -p ~/.bash_completion.d
( export _$(tr a-z- A-Z_ <<<"$cmdname")_COMPLETE=source ; \
  $cmdname >~/.bash_completion.d/$cmdname.sh )
grep /.bash_completion.d/$cmdname.sh ~/.bash_completion >/dev/null \
    || echo >>~/.bash_completion ". ~/.bash_completion.d/$cmdname.sh"
. "/etc/bash_completion"
```

CHAPTER 4

# Contributing

To create a working directory for this project, call these commands:

```
git clone "https://github.com/1and1/config-sesame.git"
cd "config-sesame"
. .env --yes --develop
invoke build --docs test check
```

Contributing to this project is easy, and reporting an issue or adding to the documentation also improves things for every user. You don't need to be a developer to contribute. See Contribution Guidelines for more.

CHAPTER 5

# Documentation Contents

## 5.1 Overview of Config Sesame

### 5.1.1 Motivation

Collaborative workflows in a devops environment profit from shared code and configuration repositories, due to increased transparency and lowered maintenance efforts. Having fewer redundantly maintained copies of shareable information also reduces hand-over friction and thus error rates.

As a consequence, secrets need their own place outside of code and configuration SCM systems, with an enforced access policy. That leaves the rest of the configuration in a state where it can be freely shared amongst technical staff (i.e. put into the application's source code repository). The secrets that are left out are then replaced by *references* to those secrets, so they can be looked up later on and provided to the consuming applications as part of the delivery process. Typical secrets are passwords, SSH private keys + certificates, and API keys. Secrets can be used both for purposes of controlling the pipeline and protecting target assets, as well as merely transported to target systems by inserting them into configuration sets.

The purpose of this tool is protecting sensitive information when it's used in the delivery pipeline and its workflows. How secrets are stored and used on the target systems is out of scope, since that is strongly coupled to external (technical) restrictions of the target platforms and applications.

Protecting the secrets is done by delaying their injection into the pipeline as long as possible, and creating a *separate* configuration file on either a deployment agent (e.g. a machine running Ansible playbooks), or the target system. Using an agent machine is preferable, since then the vault access credentials are used at fewer places, and you have fewer machines to consider when maintaining your tools.

### 5.1.2 Requirements

The following tenets and requirements were considered in the design:

- Secrets need to be **managed and stored securely**, ideally *apart* from other not so sensitive configuration information.
- Secrets must be **identifiable**, so they can be **referenced** from openly available configuration.
- Secrets must be **distinguishable**, so they can be **filtered or hidden** in reports, logs, web interfaces, or for anonymous access.
- During deployment, secrets need to be **looked up and added** to already collected configuration sets, by augmenting references to them, as late as technically possible.
- **Access and use secrets as late as possible** in a pipeline, and keep tight control where they end up.

- Secrets must be **handled in a transient fashion** (don't add them to persistent storage if avoidable).
- Use authorization credentials of the *initiator* of a pipeline run to access secrets, but **without revealing them**.

### 5.1.3 Why Use a Vault?

While SCM-based encryption tools (like ansible-vault) might fit your needs, using a vault backend has additional advantages.

**Hashicorp Vault specifically offers these features:**

- dynamic credentials.
- more versatile authentication options.
- key management over time is more stringent (leases, revoke, ...).
- better auditing (non-repudiation is a primary concern).

### 5.1.4 Implementation

The first release will support Hashicorp Vault as the secrets store. If you need support for other backends, please open an issue, or even better, file a PR – see Contribution Guidelines for more.

We also restrict the configuration file support to YAML for the initial implementation, to keep things simple and because we think that amongst the standard formats it's the one most easily handled by humans, while still being very versatile and powerful.

The format used for referencing secrets is `vault:[<vault-name>:]<secret-name>`. Edge cases where a backend uses colons for their own purposes, or an application uses `vault:` as a prefix in its own configuration values, can be handled by escaping via duplication (`::`).

### 5.1.5 A Typical CD Pipeline

Figure 5.1 shows the environment in which `config-sesame` does its job. Given the existing application configuration, the application itself, and the secrets stored safely in *Vault*, it creates the `secrets.yml` file as the missing part of the inputs for the configuration management tool of your choice, to deploy the configured application to the target runtime environment.

## 5.2 Installing Config Sesame

Below you find several different options to install the tool and *Hashicorp Vault*, and how to connect them.

### 5.2.1 Installing Into a Python Virtualenv

These commands install `config-sesame` into its own virtualenv in your home. It can be easily removed again, since most everything is contained within the virtualenv's directory. Also see Installing Python Software in case your machine lacks *Python* or tools like `virtualenv`.

```
#release="config-sesame"
release="https://github.com/1and1/config-sesame/archive/master.zip#egg=config-sesame"

# install "config-sesame" to its own virtualenv
```

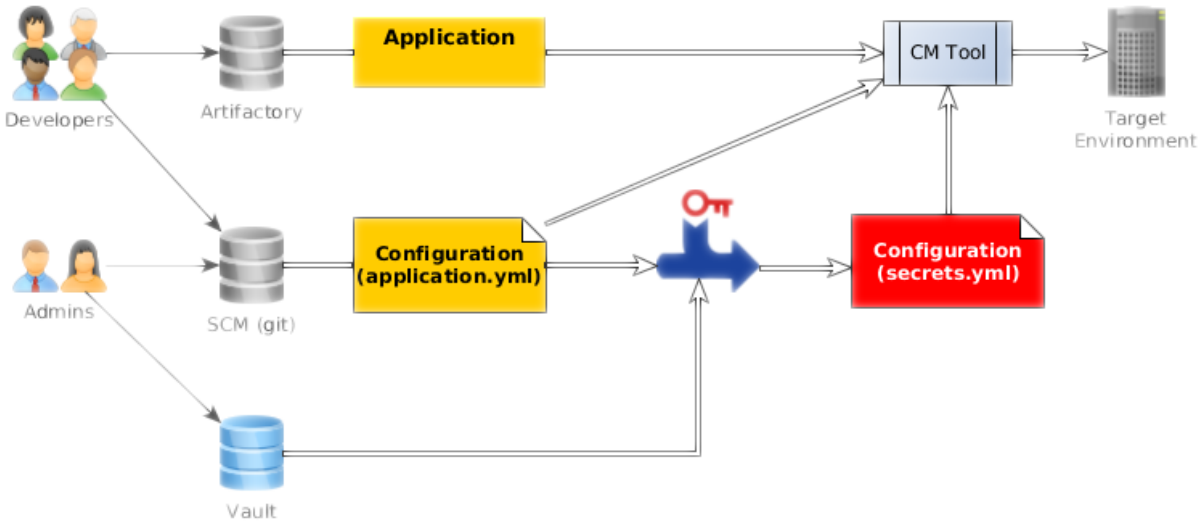


Fig. 5.1: How `config-sesame` fits into the data-flow of a CD pipeline.

```
mkdir -p ~/.local/virtualenvs
test -d ~/.local/virtualenvs/config-sesame \
    || virtualenv ~/.local/virtualenvs/config-sesame
~/.local/virtualenvs/config-sesame/bin/pip install -U pip setuptools wheel
~/.local/virtualenvs/config-sesame/bin/pip install -U "$release"
mkdir ~/bin 2>/dev/null && exec $SHELL -l
ln -s ~/.local/virtualenvs/config-sesame/bin/config-sesame ~/bin
config-sesame --version
```

On *Linux*, if you want to safely store the credentials to access *Vault* in your account's keyring, execute these additional commands:

```
sudo apt-get install libdbus-glib-1-dev python-dev libffi-dev build-essential
~/.local/virtualenvs/config-sesame/bin/pip install secretstorage dbus-python keyring
```

See keyring installation for more details on that.

### 5.2.2 Installing Hashicorp Vault

See Vault's documentation for detailed instructions. For a simple development / test installation on *Ubuntu*, this works when called in a `root` shell:

```
version=0.5.3
curl -sLS "https://releases.hashicorp.com/vault/${version}/vault_${version}_linux_amd64.zip" \
    | funzip >/usr/local/bin/vault
chmod a+x /usr/local/bin/vault
apt-get install supervisor
adduser vault --ingroup daemon --home /var/lib/vault --system --disabled-password
cat >/etc/supervisor/conf.d/vault.conf <<'EOF'
[program:vault]
command         = /usr/local/bin/vault server -dev
user            = vault
redirect_stderr = True
autostart       = True
```

```
EOF
supervisorctl update
supervisorctl tail -2200 vault
```

**Warning:** As mentioned above, this is intended for experimenting with Vault on your workstation. Do **NOT** run it this way on anything that is intended for production use.

### 5.2.3 Providing Credentials for Vault

Make sure your *Vault* server is up and reachable by using these commands:

```
export VAULT_ADDR="http://127.0.0.1:8200"
vault status
```

If the server runs on a remote machine, adapt the `VAULT_ADDR` accordingly.

Let's try accessing the server via `config-sesame` next. First, add the *vault root token* to your keyring, by calling the `config-sesame login` command which will prompt you for the token and remember it in a safe place. For the test setup as outlined above, you'll find that token in the file `/var/lib/vault/.vault-token`. You can also set the `VAULT_TOKEN` environment variable or create the `~/.vault-token` file (which is not as secure as using the keyring), otherwise you'll be prompted for the token on the console each time it is needed. Finally call the `config-sesame help` command, and it should show some information about your running *Vault* service.

Note that in a production setup, you will have a personal access token, e.g. obtained via LDAP or similar credentials.

### 5.2.4 Production Deployment

The project contains a `debian` directory that supports building a self-contained Python virtualenv wrapped into a Debian package (an "omnibus" package, all passengers on board). The packaged virtualenv is kept in sync with the host's interpreter automatically. See dh-virtualenv for more details. On platforms that are not some *Debian* flavour, consider using rpmvenv, platter or fpm. The motivation to strongly prefer native packages for deployment can be found in Python Application Deployment with Native Packages.

Note that you need to install the usual Debian development tools and `dh-virtualenv` (at least version 0.10), before you can actually build the DEB package. These incantations will perform that for you (on *Xenial*):

```
sudo apt-get install build-essential debhelper devscripts equivs
sudo mk-build-deps --install debian/control
```

*Jessie* only comes with version `0.7` – that might work, otherwise you have to build a newer version from source, or use `0.10` from backports.

Then, if you have all pre-requisites satisfied, try this:

```
dpkg-buildpackage -uc -us -b
```

or, if you followed the instructions to create a developer working directory, this instead:

```
invoke deb
```

The resulting package, if all went well, can be found in the parent of your project directory for the direct `dpkg-buildpackage` call, and in `dist` when you used `invoke`. You can upload it to a Debian package repository via e.g. *dput*, see dput-webdav for a hassle-free solution that works with *Artifactory* and *Bintray*.

## 5.3 Using Config Sesame

*See* Installing Config Sesame *for instructions on how to install and configure the tool.*

### 5.3.1 Performing Password Lookups

The following examples ilustrate the process of augmenting secrets references with their resolution from Vault; see *Details of Configuration Parsing* for more.

- `my: database: password_secret: vault:db/password`

  gets resolved to

  `my: database: password: the_actual_password`

- The configuration...

```
db:
  auth_secret: "vault:db/credentials"
```

  becomes...

```
db:
  auth:
    user: jane
    password: test123
```

### 5.3.2 A Practical Example

To see how everything works in reality, the project repository comes with test data that can be used in combination with the Vault setup described in the previous chapter.

First, let's populate the test server with some secrets:

```
$ invoke populate
vault write "secret/sesame/db/credentials" pwd="SECRET" user="kermit"
Success! Data written to: secret/sesame/db/credentials
vault write "secret/sesame/db2/password" value="ALSO_SECRET"
Success! Data written to: secret/sesame/db2/password
vault write "secret/sesame/resource/password" value="MORE_SECRETS"
Success! Data written to: secret/sesame/resource/password
```

Since this delegates the work to the `vault` command, you have to set both the `VAULT_ADDR` and `VAULT_TOKEN` environment variables beforehand.

Now we can use the sample data in `src/tests/data` to perform a lookup on these keys:

```
$ config-sesame open src/tests/data/*yml -o- -b secret/apps -b secret/sesame
db:
  auth:
    pwd: SECRET
    user: kermit
  auth_secret_url: http://127.0.0.1:8200/v1/secret/sesame/db/credentials
my:
  database:
    password: ALSO_SECRET
    password_secret_url: http://127.0.0.1:8200/v1/secret/sesame/db2/password
  resource:
```

```
    password: MORE_SECRETS
    password_secret_url: http://127.0.0.1:8200/v1/secret/sesame/resource/password
```

Note that the source of each resolved secret is also added to the result, for diagnostic and auditing purposes. In case one of the secret references cannot be resolved, we get an error:

```
$ config-sesame open src/tests/data/*yml -b foo -b bar
Usage: config-sesame open [OPTIONS] CFGFILE [...]

Error: Cannot find key "db/credentials" in any of these bases: foo, bar.
```

### 5.3.3 Details of Configuration Parsing

To support reading multiple input files, a simple merging strategy is used:

- Objects (dicts, hashes) are merged recursively.
- Scalar values and lists are simply replaced (i.e. the last file has priority).

For the purpose of finding references to secrets and writing their resolution to a new file, this always fits the bill.

The rules for handling secrets references:

- Secrets references are stored like any other configuration key, and take the form `vault:«vault-path»`.
- The `«vault-path»` part is resolved relative to a base path, e.g. “apps/«app name»/«brand»/«environment»”.
- The Vault base path is part of the tool’s configuration.
- Resolved secrets are added to `secrets.yml` as `«key»` for a reference named `«key»_secret`.
- If `«vault-path»` references a single scalar value, it is added as such.
- If `«vault-path»` references a collection of values, they are added as an object (a/k/a dict or hash).
- The URL where the secret was retrieved from is added as `«key»_secret_url`.

## 5.4 Complete API Reference

The following is a complete API reference generated from source.

### 5.4.1 config_sesame package

Config Sesame – A tool to look up secrets from a vault based on existing shareable configuration.

Copyright © 2016 1&1 Group <jh@web.de>

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

## Subpackages

### config_sesame.commands package

CLI commands.

#### Submodules

**config_sesame.commands.help module** 'help' command.

**config_sesame.commands.login module** 'login' command.

**config_sesame.commands.open module** 'open' command.

`config_sesame.commands.open.`**`lookup_key`**(*key*, *bases*, *conn*)
Look for a key in the given bases.

`config_sesame.commands.open.`**`lookup_secrets`**(*obj*, *bases*, *conn*)
Scan `obj` for secrets, and look them up.

`config_sesame.commands.open.`**`vault_key`**(*reference*)
Validate a vault key reference.

### config_sesame.util package

Helpers.

#### Submodules

**config_sesame.util.cfgdata module** Helpers for handling configuration data).

`config_sesame.util.cfgdata.`**`is_mapping`**(*obj*)
Check if `obj` offers the mapping interface.

`config_sesame.util.cfgdata.`**`load_all`**(*filename*)
Generate objects contained in `filename`.

`config_sesame.util.cfgdata.`**`merge_objects`**(*namespace*, *obj*)
Update `namespace` with data in `obj`.

`config_sesame.util.cfgdata.`**`read_merged_files`**(*cfgfiles*)
Read a list of hierachical config files, and merge their keys.

**config_sesame.util.vault module** Hashicorp Vault API (based on `hvac`).

**class** `config_sesame.util.vault.`**`APIWrapper`**(*url=u'http://localhost:8200'*, *token=None*, *cert=None*, *verify=True*, *timeout=30*, *proxies=None*, *allow_redirects=True*, *session=None*)
Bases: `hvac.v1.Client`

Wrapper for client API.

**`last_url`** = None

**class** config_sesame.util.vault.**Connection**(*url=None*, *token=None*)
Bases: object

Hashicorp Vault connection.

**__str__**()
Return human readable description of this connection.

config_sesame.util.vault.**default_credentials**(*url=None*, *token=None*)
Return default credentials from env / configuration in a tuple (url, user, token, auth_by).

config_sesame.util.vault.**get_credentials**(*url=None*, *token=None*)
Return active credentials in a tuple (url, user, token, auth_by).

#### Submodules

#### config_sesame.config module

Configuration utilities.

config_sesame.config.**envvar**(*name*, *default=None*)
Return an environment variable specific for this application (using a prefix).

config_sesame.config.**version_info**(*ctx=None*)
Return version information just like –version does.

## 5.5 Contribution Guidelines

### 5.5.1 Overview

Contributing to this project is easy, and reporting an issue or adding to the documentation also improves things for every user. You don't need to be a developer to contribute.

#### Reporting issues

Please use the *GitHub issue tracker*, and describe your problem so that it can be easily reproduced. Providing relevant version information on the project itself and your environment helps with that.

#### Improving documentation

The easiest way to provide examples or related documentation that helps other users is the *GitHub wiki*.

If you are comfortable with the Sphinx documentation tool, you can also prepare a pull request with changes to the core documentation. GitHub's built-in text editor makes this especially easy, when you choose the *"Create a new branch for this commit and start a pull request"* option on saving. Small fixes for typos and the like are a matter of minutes when using that tool.

#### Code contributions

Here's a quick guide to improve the code:

1. Fork the repo, and clone the fork to your machine.
2. Add your improvements, the technical details are further below.

3. Run the tests and make sure they're passing (`invoke test`).
4. Check for violations of code conventions (`invoke check`).
5. Make sure the documentation builds without errors (`invoke build --docs`).
6. Push to your fork and submit a pull request.

Please be patient while waiting for a review. Life & work tend to interfere.

### 5.5.2 Details on contributing code

This project is written in Python, and the documentation is generated using Sphinx. setuptools and Invoke are used to build and manage the project. Tests are written and executed using pytest and tox.

#### Set up a working development environment

To set up a working directory from your own fork, follow these steps, but replace the repository `https` URLs with SSH ones that point to your fork.

For that to work on Debian type systems, you need the `git`, `python`, and `python-virtualenv` packages installed. Other distributions are similar.

#### Add your changes to a feature branch

For any cohesive set of changes, create a *new* branch based on the current upstream `master`, with a name reflecting the essence of your improvement.

```
git branch "name-for-my-fixes" origin/master
git checkout "name-for-my-fixes"
... make changes...
invoke ci # check output for broken tests, or PEP8 violations and the like
... commit changes...
git push origin "name-for-my-fixes"
```

Please don't create large lumps of unrelated changes in a single pull request. Also take extra care to avoid spurious changes, like mass whitespace diffs. All Python sources use spaces to indent, not TABs.

#### Make sure your changes work

Some things that will increase the chance that your pull request is accepted:

- Follow style conventions you see used in the source already (and read PEP8).
- Include tests that fail *without* your code, and pass *with* it. Only minor refactoring and documentation changes require no new tests. If you are adding functionality or fixing a bug, please also add a test for it!
- Update any documentation or examples impacted by your change.
- Styling conventions and code quality are checked with `invoke check`, tests are run using `invoke test`, and the docs can be built locally using `invoke build --docs`.

Following these hints also expedites the whole procedure, since it avoids unnecessary feedback cycles.

## 5.6 Software License

Copyright © 2016 1&1 Group <jh@web.de>

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### 5.6.1 Full License Text


```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
```

```
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
```

```
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
```

```
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "{}"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

CHAPTER 6

# References

## 6.1 Tools

- Cookiecutter
- PyInvoke
- pytest
- tox
- Pylint
- twine
- bpython
- yolk3k

## 6.2 Packages

- Rituals
- Click

CHAPTER 7

# Indices and Tables

- genindex
- modindex
- search

## C

## Symbols



## A



## C



## D



## E



## G



## I



## L



## M



## R



## V