bootils Documentation

Release 0.1.0

1
1 Group

2016-06-03

Contents

Bootils offers process boot-strapping utilities that support writing robust application/service launcher and process life-cycle management scripts. It is comprised of a `bootils` Python package with building blocks for process and resource management, and a CLI tool named `nanny` that watches your child process after starting, until it grows up into a stable running state.

Continue with reading Introduction & Concepts.

CHAPTER 1

Important Links

- GitHub Project
- Issue Tracker
- PyPI
- Latest Documentation
- Google Group

CHAPTER 2

Documentation Contents

2.1 Introduction & Concepts

2.1.1 Motivation

Startup scripts that come with services are commonly of the launch-and-abandon variety, i.e. after a demon fork was successful, the service is left alone and not watched from the outside. But quite often problems arise only late in an initialization procedure, and if no monitoring system or human watches the logs and other indicators of failure, those problems go unnoticed or at least are only recognized far later than they could be.

The mission of *Bootils* is to fix that, by checking operational parameters *before launching* a service, *during its initialization*, and *while it is running*. It also assists with describing robust startup procedures, and can thus help to replace fragile default `init.d` scripts without spending lots of effort. This also helps to reduce variation in the way different service processes are managed.

2.1.2 Design Principles

Bootils follows the Unix Design Philosophy of providing small, simple, clear, modular, and extensible building blocks, to give its user the maximum amount of flexibility and reusability.

2.1.3 Feature Overview

- **Plugin Architecture** – *Bootils* has a very small core that manages a set of configured plugins, both built-in and custom ones.
- **Pre-Condition Checks** – If a service depends on the availability of resources like mount points or disk space, you can assert they're OK, instead of noticing problems only after you have one more incident to handle.
- **Facility Re-Use** – Established technologies like process supervisors, the Jolokia JMX bridge and so on can be integrated via plugins.
- **Runtime Environments** – In particular for launching Java / JVM applications, a standard runtime is provided based on the Tanuki Java Service Wrapper, which already establishes a basic level of startup and runtime monitoring.

2.2 Installation Guide

2.2.1 Overview

The following sections describe different installation options – choose the right one for you. If you use `bash`, consider *Setting Up bash Completion*.

You might also need to follow some setup procedures to make the necessary basic commands available on *Linux*, *Mac OS X*, and *Windows*.

Note: *Bootils* is tested on *Debian Wheezy* and *Ubuntu Trusty*. It will generally work on other platforms or other versions of these distributions, too. The most important pre-requisite is availability of Python 2.7 or 3.4+.

2.2.2 Installation as a Debian Package

Debian packages of release versions will eventually be available on Bintray. Follow the instructions there to extend your *APT* configuration, update your package database with `apt-get update`, and finally install the package using `apt-get install bootils`.

If you want to build your own package file directly from source, you need to follow these steps:

- Install dh-virtualenv 0.8+.
- Create a working directory.
- In the development environment, call `invoke deb` – the built package files are placed in the `dist` directory.
- Optionally, upload the package to a local Debian repository, e.g. to Artifactory using dput.

After installation, continue with the Quick-Start Guide.

2.2.3 Installation With pip

Bootils can be installed via `pip install bootils` as usual, see releases on GitHub for an overview of available versions. To get a bleeding-edge version from source, use these commands:

```
repo="Build-The-Web/bootils"
pip install -r "https://raw.githubusercontent.com/$repo/master/requirements.txt"
pip install -U -e "git+https://github.com/$repo.git#egg=${repo#*/}"
```

It is recommended to **not** do this via `sudo`, but to create a virtualenv first, or use pipsi for installation. See Contribution Guidelines on how to create a full development environment.

Continue with *Setting Up bash Completion* or the Quick-Start Guide.

2.2.4 Setting Up bash Completion

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

```
cmdname=one
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"
```

Note: The Debian package already comes equipped with an appropriate snippet, just make sure you have the `bash-completion` package installed.

The Quick-Start Guide describes the next steps.

2.3 Quick-Start Guide

The following will help you setting up and starting a very basic service. The Configuration Reference provides details on the configuration options presented here.

Refer to the examples directory on GitHub for more setups of typical services.

2.3.1 A Very Simple Example

The `netcat` example demonstrates the basic mechanics of setting up a runtime environment using Bootils, without any complexity whatsoever introduced by the service that is launched.

The detailed installation steps to run it are found in the examples/netcat README, here we'll explain the moving parts in that demo and what's their purpose.

TODO

2.4 Using the CLI Tool

The `nanny` command acts on the configuration of a service, by first checking any pre-launch requirements, starting it when those are OK, and then watching logs and other status indicators until it reaches a stable running state.

Use `nanny --help` to get a list of global options and sub-commands, `nanny ‹command› --help` for detailed help on a specific command and its options, and `nanny help` to get information like the paths to configuration files and plugin directories.

2.4.1 Performing Checks

While checking requirements is always done when launching a service, it can also be triggered explicitely by using the `check` sub-command. If any requirement isn't satisfied, the return code will reflect that – use this together with the `-q` option to test them in scripts.

The `--pre` and `--post` options can be used to select which checks to perform, if neither is given, *all* checks are active.

Each check produces a result with the following attributes: `ok` (either `true` or `false`), `name` (the qualfied name of the check), `comment` (details on the requirement, e.g. a file system path), and `diagnostics` (error messages, output of a command that actually performed the check, ...).

Unless the global option `-q` is used, check results are printed to the console; if you use `-v`, diagnostic information is included, which can help to hunt down the reason for a check failure. The available output formats are `text` (tabular output), `tap` (Perl's Test Anything Protocol), and serialization into `json`, `yaml`, or `csv`. Use the `--format` option to select them, `text` is the default.

Example:

```
$ nanny -v check -f tap
not ok 1 FileSystem:exists /etc/cassandra/jolokia-config.properties
ok 2 FileSystem:exists /etc/hosts
not ok 3 FileSystem:mounted /mnt/data
# [Errno 2] No such file or directory: '/mnt/data'
not ok 4 FileSystem:mounted /mnt/commitlog
# [Errno 2] No such file or directory: '/mnt/commitlog'
not ok 5 FileSystem:mounted /opt
# path resides in root file system
ok 6 FileSystem:mounted /home
ok 7 FileSystem:mounted /home/jhe
not ok 8 FileSystem:diskfree /home 70% 44GiB [46.9% 43.8GiB/104.6GiB free]
# violated 70% condition (46.9% 43.8GiB free)
# violated 44GiB condition (46.9% 43.8GiB free)
ok 9 Host:packages oracle-java8-jre | oracle-java8-installer
# oracle-java8-jre 8.45-1~ui1404+1 install ok installed
ok 10 Host:packages javaservicewrapper
# javaservicewrapper 3.5.22-1~ui1404+1 install ok installed
not ok 11 Host:packages jolokia-jvm-agent
# Command '[u'dpkg-query', u'-W', u'-f=${Package} ${Version} ${Status}', u'jolokia-jvm-agent']' retu
1..11
```

2.4.2 Launching Services

TODO

2.5 Configuration Reference

TODO

2.5.1 Configuration File Structure

Configuration files consist of sections that start with a `[section-name]` line, with an (unnamed) *global* section before the named ones. Nesting sub-sections into outer sections is achieved by adding more square brackets. See ConfigObj Files for more details.

Each section holds a list of key/value pairs.

Note: Later versions will offer alternative formats like YAML, as long as they're able to represent this nested structure of sections and key/value pairs.

2.5.2 Main Configuration Files

Configuration files are expected at the locations as shown by the `nanny help` command, on a Linux system that is:

```
/etc/bootils/nanny.conf
/etc/bootils/nanny.d/*.conf
~/.config/bootils/nanny.conf
```

If you define the `NANNY_CONFIG` environment variable with additional files, those will be appended to the default list – try this command to see for yourself:

```
NANNY_CONFIG=/tmp/foo.conf:/tmp/bar.conf nanny help
```

Configuration files are merged in the given order, i.e. keys that appear in files further down the list shadow those in files read earlier. This allows you to provide general settings in the default files, and then modify and extend them for a specific service.

Common usage patterns are to have everything in `/etc/bootils/nanny.conf` if you only ever run a single service (say, in a Docker container), and use the `conf.d` directory for snippets of global configuration added by packages. If you run several services on one machine, keep `nanny.conf` clear of settings specific to any service, and use `nanny.d/‹service›.conf` files for those. In the service-specific files, be sure to qualify your top-level sections with the service name, e.g. `[‹service›:pre-check]`.

2.5.3 Built-in Plugins

FileSystem

This plugin allows to check that a certain path `exists`, is `executable`, or is `mounted` (i.e. not part of the root file-system). You can also check for free space of the volume of a given path using `diskfree`.

All these attributes take a multi-line list of paths to check, `diskfree` also expects a percentage or size threshold of minimal free space. If both a percentage and a size is given, each must be satisfied for the check to be OK.

Example:

```
[pre-check]

[[FileSystem]]

exists = """
    /etc/cassandra/jolokia-config.properties
    /etc/hosts
"""

executable = """
    /bin/bash
"""

mounted = """
    /mnt/data
    /mnt/commitlog
    /opt
    /home
"""

diskfree = """
    /home 5% 42GiB
"""
```

Host

With the `Host` plugin, you can ensure that essential `packages` were indeed installed by your configuration management tool. This provides explicit diagnostics (unlike e.g. a `command not found` for some missing tool), and avoids errors that might only appear when a service tries to access an optional component that was not installed.

Example:

```
[pre-check]

[[Host]]
packages = """
    oracle-java8-jre | oracle-java8-installer
    service-wrapper
    jolokia-jvm-agent
"""
```

Network

The `network` plugin is able to check if all ports and addresses, that your server is going to use, are not already bound.

Example (short-hand notation):

```
[pre-check]

[[Network]]
ports = 80, 8081
```

Example (verbose):

```
[pre-check]

[[Network]]

[[[http]]]
port = 80
family = tcp
address = 0.0.0.0

[[[jmx_port]]]
port = 6379
family = tcp
address = 127.0.0.1
```

2.6 Custom Plugins

2.6.1 Installing Additional Plugins

TODO

The default paths for custom plugins on a POSIX system are `/etc/bootils/plugin.d` and `~/.config/bootils/plugin.d`.

2.6.2 Writing Your Own Plugins

TODO

Plugins are implemented in classes that inherit from *bootils.plugins.loader.PluginBase* and then provide appropriate method implementations like `pre_check`. `PluginBase` also provides a few helper methods, most importantly *bootils.plugins.loader.PluginBase.result()* to create check result data that can then be yielded to the core.

To get an idea how this all works when put together, look at the code of the built-in plugins in the *bootils.plugins.core* package.

2.7 Complete API Reference

The following is a complete API reference generated from source.

2.7.1 bootils package

Bootils offers process boot-strapping utilities that support writing robust application/service launcher and process life-cycle management scripts.

It is comprised of a `bootils` Python package with building blocks for process and resource management, and a CLI tool named `nanny` that watches your child process after starting, until it grows up into a stable running state.

Copyright © 2015 1&1 Group <btw-users@googlegroups.com>

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

bootils.commands package

CLI commands.

Submodules

bootils.commands.check module 'help' command.

bootils.commands.help module 'help' command.

bootils.commands.service module Service launching and process control commands.

bootils.plugins package

Package and namespace for plugin implementations.

Built-in plugins live in *bootils.plugins.core*, and custome plugins are loaded from the file system into the `bootils.plugins.custom` namespace.

Subpackages

bootils.plugins.core package Package holding Bootils *core* plugins.

Submodules

bootils.plugins.core.demon module Unix demon runtime environment.

class bootils.plugins.core.demon.**Demon**(*context*)
Bases: *bootils.plugins.loader.PluginBase*

Unix demon runtime environment.

control_start(**args*, ***options*)
Start a Unix demon.

control_stop(**args*, ***options*)
Stop a Unix demon.

bootils.plugins.core.filesystem module File system plugin.

class bootils.plugins.core.filesystem.**FileSystem**(*context*)
Bases: *bootils.plugins.loader.PluginBase*

File system checks.

post_check()
Perform checks.

pre_check()
Perform checks.

bootils.plugins.core.filesystem.**diskfree_result**(*spec*)
Return result of a single disk usage check.

bootils.plugins.core.filesystem.**on_same_fs**(*path1*, *path2*)
Check if two paths reside in the same file system.

bootils.plugins.core.host module Host + operating system plugin.

class bootils.plugins.core.host.**Host**(*context*)
Bases: *bootils.plugins.loader.PluginBase*

Host and OS checks.

post_check()
Perform checks.

pre_check()
Perform checks.

bootils.plugins.core.jsw module Tanuki Java Service Wrapper runtime environment.

Debian JSW paths (Wheezy 3.5.3; Jessie 3.5.22):

```
/usr/sbin/wrapper - ELF executable
/usr/share/wrapper/daemon.sh
/usr/share/wrapper/make-wrapper-init.sh
/usr/share/wrapper/wrapper.conf
```

class bootils.plugins.core.jsw.**JavaServiceWrapper**(*context*)
Bases: *bootils.plugins.loader.PluginBase*

Tanuki Java Service Wrapper runtime environment.

control_start(**args*, ***options*)
Start a Java service.

control_stop(**args*, ***options*)
Stop a Java service.

bootils.plugins.custom package Namespace for loading *custom* plugins.

Submodules

bootils.plugins.loader module Plugin management.

class bootils.plugins.loader.**PluginBase**(*context*)
Bases: object

Base class for plugins.

This class defines the plugin interface (callbacks), and provides sensible default implementations so that a plugin only has to define those callbacks it *needs* to override.

cfg_list(*key*, *section=None*)
Get a config value as a list.

configure(*config*)
Store plugin-specific configuration.

control(*command*, **args*, ***options*)
Control a service / process.

This delegates to a control_‹ command › method of a subclass, if one is found.

Returns True if the command was handled successfully.

Return type bool

name
Name of the plugin (e.g. for reporting).

post_check()
Perform post-launch checks and generate results.

pre_check()
Perform pre-launch checks and generate results.

result(*ok*, *name*, *comment*, *diagnostics=None*)
Create checks.CheckResult with a qualified name.

class bootils.plugins.loader.**PluginContext**
Bases: object

State held by plugins.

class bootils.plugins.loader.**PluginExecutor**(*loader*)
Bases: object

Call plugin hooks in different life-cycle phases.

configure()
Assemble configuration for each plugin and pass it on.

control(*command*, **args*, ***options*)
Delegates execution of the given command to all plugins, until one of them indicates it handled the task.

post_checks()
Perform post-launch checks.

pre_checks()
Perform pre-launch checks.

class bootils.plugins.loader.**PluginLoader**(*cfg*, *appname*)
Bases: object

Load and manage plugins, both core and custom ones.

See also Package Discovery and Resource Access using pkg_resources.

DEFAULT_PLUGIN_PATH = [u'/etc/{appname}/plugin.d', u'{appdir}/plugin.d']

discover()
Inspect the given search path and import any plugins found.

Returns the list of plugin classes.

classmethod load_into_context(*ctx*, *project=None*)
Discovers plugins and places a PluginLoader instance in ctx.obj.plugins.

bootils.util package

Helpers.

Submodules

bootils.checks module

Check helpers + results.

class bootils.checks.**CheckFormatter**(*formatting=u'text'*, *stream=None*, *verbose=False*)
Bases: object

Emit a sequence of check results.

GLUE = {'yaml': (u'', u'', u''), 'json': (u'[\n', u',\n', u'\n]\n'), 'csv': (u'', u'', u''), 'tap': (u'', u'\n', u'\n'), 'text': (u'', u'\n

close()
Print any trailing output and clean up resources.

dump(*result*)
Print a single check result.

write(*text*)
Unbuffered write of given text to output stream.

class bootils.checks.**CheckResult**(*ok*, *name*, *comment*, *diagnostics*)
Bases: tuple

__getnewargs__()
Return self as a plain tuple. Used by copy and pickle.

__getstate__()
Exclude the OrderedDict from pickling

__repr__()
Return a nicely formatted representation string

comment
Alias for field number 2

diagnostics
Alias for field number 3

name
Alias for field number 1

ok
Alias for field number 0

bootils.config module

Configuration utilities.

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

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

bootils.launcher module

Service launcher and processc control.

class bootils.launcher.**LauncherBase**(*config*)
Bases: object

Process launch & management.

init_environ()
Initialize process environment and return its old state.

restore_environ(*oldstate*)
Restore process environment to previous state as returned by init_environ.

bootils.launcher.**check_gid**(*gid*)
Get numerical GID of a group.

Raises KeyError – Unknown group name.

bootils.launcher.**check_uid**(*uid*)
Get numerical UID of a user.

Raises KeyError – Unknown user name.

bootils.launcher.**signal2int**(*sig_spec*)
Convert given signal specification to its integer value.

Parameters **sig_spec** (*int or str*) – Either already an int, a number as a string, or a case-insensitive signal name.

Returns Signal number.

Return type int

Raises `ValueError` – Bad / unknown signal name, or bad input type.

2.8 Contribution Guidelines

2.8.1 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.

2.8.2 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.

2.9 Software License

Copyright © 2015 1&1 Group <btw-users@googlegroups.com>

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.

2.9.1 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 3

References

Tools

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

Packages

- Rituals
- Click
- pluginbase
- pluggy
- psutil

Indices & Tables

- genindex
- modindex
- search

CHAPTER 5

Acknowledgements

Project sponsored by 1&1.

b

Symbols

B

C

D

E

F

G

H

I

J

L

N

O

P

R

S

V

W