logdispatchr Documentation

Release 0.1.0

Paul Ollivier

August 31, 2016

Contents

Contents:

CHAPTER 1

logdispatchr

An attempt at a simple syslog-compatible(?) log dispatcher and filter. For now, mainly a POC.

- Free software: MIT license
- Documentation: https://logdispatchr.readthedocs.io.

1.1 Features

- tail a file ?
- send the log line on the network, using msgpack => DONE
- write somewhere else => DONE

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install logdispatchr, run this command in your terminal:

```
$ pip install logdispatchr
```

This is the preferred method to install logdispatchr, as it will always install the most recent stable release.

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

2.2 From sources

The sources for logdispatchr can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/paulollivier/logdispatchr
```

Or download the tarball:

```
$ curl  -OL https://github.com/paulollivier/logdispatchr/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

The main usage of logdispatchr is through the logdispatchrd CLI utility.

This daemon listens to messages from various sources, and forwards them to outputs, with optionnal filtering.

logdispatchr is configured from a configuration file.

CHAPTER 4

Configuration

The configuration file is in toml. A sample configuration file `config.toml.sample` is provided.

There are multiple sections of interest:

4.1 Inputs and Outputs

Here is a sample configuration for a local syslog listener (an input, then):

```
[inputs.localsyslog]
key = "local.syslog" # mandatory
class = "UDPSyslogInput" # mandatory
# these have "sensible" defaults :)
host = "localhost"
port = 5141
```

As we can see, there is a section title, called "inputs.localsyslog", which is just a name we give to this input. As you can see, we also specify the class to use.

The `key` parameter is used to "tag" the Messages recieved via this input, for further routing and processing.

The other parameters are optional, and should provide reasonnable defaults.

For further information on writing such modules and the full argument list for each input/output, please look at `inputs` and `outputs`.

CHAPTER 5

Contributing

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

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/paulollivier/logdispatchr/issues.

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.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

logdispatchr could always use more documentation, whether as part of the official logdispatchr docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/paulollivier/logdispatchr/issues.

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 :)

5.2 Get Started!

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

1. Fork the *logdispatchr* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/logdispatchr.git
```

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

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

4. Create a branch for local development:

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

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 logdispatchr tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

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

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/paulollivier/logdispatchr/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_logdispatchr
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search