tailchaser

Release 0.1.0

March 08, 2016

Contents

CHAPTER 1

Overview

docs	
tests	
package	

the ultimate tailer

- Free software: BSD license

1.1 Installation

```
pip install tailchaser
```

CHAPTER 2

Usage

:: $ tailchaser /wher/my/logs/*.log

$ tailchaser -h

usage: tailer [options] source_pattern

the ultimate tail chaser

positional arguments:

source_pattern source pattern is the glob path to a file to be tailed plus its rotated versions

optional arguments:

-h, --help	show this help message and exit
--verbose	prints a lot crap, default is: False
--dryrun	prints a lot crap and no hand-off, default is: False
--dont_backfill	don't backfill with rolled logs, default is: False
--dont_follow	don't follow when you reach the end of the file exit, default is: False
--clear_checkpoint	clears the checkpoint and lets you start from the begining, default:False
--read_period READ_PERIOD	time given to read, default: 1.0
--read_pause READ_PAUSE	time to pause between reads, default: 0

To use tailchaser in a project:

```
#
# Example 1 - Tail to Elastic
#

import requests

import tailchaser

class TailToElastic(tailchaser.Tailer):
    def handoff(self, file_tailed, checkpoint, record):
        """ Expect a record like:

        20160204 10:28:15,525 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        """
```

```
        date, time, level, source, _, message = record.split(5)
        requests.json("http://someelacticserver.com:9200/myindex/log", json={
                        'timestamp': '{}T{}'.format(date, time)
                        'level': level,
                        'source': source,
                        'message': message
                        })




#
# Example 2 - Tail to Kafka - shows how to add your own arguments and then send messahes to kafka.
#


import msgpack
import tailchaser
from kafka import KafkaProducer

class TailToKafka(tailchaser.Tailer):
    def add_arguments(cls, parser=None):
        parser = super(TailToKafka, cls).add_arguments(parser)

    HOSTS = 'localhost:1234'
    TOPIC = 'log'
    def startup(self):
        self.kafka_producer = KafkaProducer(bootstrap_servers=self.HOSTS,value_serializer=msgpack.dur


    def handoff(self, file_tailed, checkpoint, record):
        """ Expect a record like:

        20160204 10:28:15,525 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        """
        self.kafka_producer.send(self.TOPIC, message)
```

2.1 Documentation

https://tailchaser.readthedocs.org/

2.2 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER 3

Installation

At the command line:

```
pip install tailchaser
```

CHAPTER 4

Usage

To use tailchaser in a project:

```
#
# Example 1 - Tail to Elastic
#

import requests

import tailchaser

class TailToElastic(tailchaser.Tailer):
    def handoff(self, file_tailed, checkpoint, record):
        """ Expect a record like:

        20160204 10:28:15,525 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        """

        date, time, level, source, _, message = record.split(5)
        requests.json("http://someelacticserver.com:9200/myindex/log", json={
                        'timestamp': '{}T{}'.format(date, time)
                        'level': level,
                        'source': source,
                        'message': message
                        })




#
# Example 2 - Tail to Kafka - shows how to add your own arguments and then send messahes to kafka.
#


import msgpack
import tailchaser
from kafka import KafkaProducer

class TailToKafka(tailchaser.Tailer):
    def add_arguments(cls, parser=None):
        parser = super(TailToKafka, cls).add_arguments(parser)
```

```
    HOSTS = 'localhost:1234'
    TOPIC = 'log'
    def startup(self):
        self.kafka_producer = KafkaProducer(bootstrap_servers=self.HOSTS,value_serializer=msgpack.dum


    def handoff(self, file_tailed, checkpoint, record):
        """ Expect a record like:

        20160204 10:28:15,525 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        20160204 10:28:15,541 INFO PropertiesLoaderSupport - Loading properties file from URL [file:C
        """
        self.kafka_producer.send(self.TOPIC, message)
```

CHAPTER 5

Reference

5.1 tailchaser

CHAPTER 6

Contributing

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

6.1 Bug reports

When 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.

6.2 Documentation improvements

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

6.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/thanos/tailchaser/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 code contributions are welcome :)

6.4 Development

To set up *tailchaser* for local development:

1. Fork tailchaser (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/tailchaser.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

6.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`) [1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

6.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 7

Authors

- thanos vassilakis - https://github.com/thanos/thanos

CHAPTER 8

Changelog

8.1 0.1.0 (2016-02-21)

- First release on PyPI.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

t

T