MongoElector Documentation

Release 0.1.2

Zeb Palmer

May 15, 2016

Contents

1 MongoElector **3**
1.1 Features . . . 3

2 Installation **5**
2.1 Stable release . . . 5
2.2 From sources . . . 5

3 Usage **7**

4 Code Documentation **9**

5 Contributing **11**
5.1 Types of Contributions . . . 11
5.2 Get Started! . . . 12
5.3 Pull Request Guidelines . . . 12
5.4 Tips . . . 13

6 Authors **15**
6.1 Creator . . . 15
6.2 Contributors . . . 15

7 History **17**
7.1 0.0.1 (2016-05-13) . . . 17

8 Indices and tables **19**

Python Module Index **21**

Contents:

CHAPTER 1

MongoElector

Note: This project is just getting started and should be considered pre-alpha with most of the intended functionality missing or incomplete. Expect the API for anything that does work, to change.

Distributed master election and locking in mongodb

- Free software: GPLv3
- Documentation: https://mongoelector.readthedocs.io.

1.1 Features

- TODO

CHAPTER 2

Installation

2.1 Stable release

To install MongoElector, run this command in your terminal:

```
$ pip install mongoelector
```

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

2.2 From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/zebpalmer/mongoelector
```

Or download the tarball:

```
$ curl  -OL https://github.com/zebpalmer/mongoelector/tarball/master
```

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

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

CHAPTER 3

Usage

To use MongoElector in a project:

```
import mongoelector
```

CHAPTER 4

Code Documentation

class mongoelector.**MongoLocker** (*key=None*, *dbconn=None*, *ttl=600*)
Distributed Lock in MongoDB.

Used by MongoElector, but can be used as a standalone distributed lock.

Parameters

- **key** (*str*) – Name of distributed lock
- **dbconn** (*PyMongo db connection*) – Client connection to mongodb database
- **ttl** (*int*) – Lock will expire (ttl seconds) after aquired unless renewed or released

aquire()
attempts to aquire the lock

locked()
returns status of lock

Returns Lock status

Return type bool

release()
releases lock

class mongoelector.**MongoElector**
This object will do lots of awesome distributed master election coolness

Does nothing. yet.

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/zebpalmer/MongoElector/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Your python 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" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

MongoElector could always use more documentation, whether as part of the official MongoElector 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/zebpalmer/MongoElector/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 *mongoelector* for local development.

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

```
$ git clone git@github.com:your_name_here/mongoelector.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 mongoelector
$ cd mongoelector/
$ 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:

```
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv. You will also need an instance of MongoDB running, tests default connecting to localhost.

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 2.7, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/zebpalmer/MongoElector/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_mongoelector
```

CHAPTER 6

Authors

6.1 Creator

- Zeb Palmer <zeb@zebpalmer.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.0.1 (2016-05-13)

- Hello World

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

m

mongoelector, 9

A

aquire() (mongoelector.MongoLocker method), 9

L

locked() (mongoelector.MongoLocker method), 9

M

MongoElector (class in mongoelector), 9
mongoelector (module), 9
MongoLocker (class in mongoelector), 9

R

release() (mongoelector.MongoLocker method), 9