pyetcd Documentation

Release 1.7.2

TwinDB Development Team

Jul 19, 2017

Contents

Contents:

CHAPTER 1

pyetcd

Python library to work with Etcd.

- Free software: Apache Software License 2.0
- Documentation: https://pyetcd.readthedocs.io.

Features

- Basic key operations

Credits

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

CHAPTER 2

Installation

Stable release

To install pyetcd, run this command in your terminal:

```
$ pip install pyetcd
```

This is the preferred method to install pyetcd, 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.

From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/twindb/pyetcd
```

Or download the tarball:

```
$ curl  -OL https://github.com/twindb/pyetcd/tarball/master
```

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

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

CHAPTER 3

Usage

Connect to local etcd instance. Write a key and read it after:

```
from pyetcd.client import Client

client = Client()
client.write('/message', 'Hello world')
response = client.read('/message')

print(response.node['value'])
```

Write and read from a remote etcd cluster:

```
from pyetcd.client import Client

client = Client([
    '10.0.1.10',
    '10.0.1.11',
    '10.0.1.12'
])
client.write('/message', 'Hello world')
response = client.read('/message')

print(response.node['value'])
```

Watch a key:

```
from pyetcd.client import Client

client = Client([
    '10.0.1.10',
    '10.0.1.11',
    '10.0.1.12'
])
client.write('/message', 'Hello world')
response = client.read('/message', wait=True)
```

```
print(response.node['value'])
```

CHAPTER 4

pyetcd package

Submodules

pyetcd.client module

class pyetcd.client.**Client**(*host='127.0.0.1'*, *port=2379*, *srv_domain=None*, *version_prefix='v2'*, *protocol='http'*, *allow_reconnect=True*)

Bases: object

Etcd Client class

Parameters

- **host** – etcd node hostname or list of hostnames or list of tuples (hostname, port) (default='127.0.0.1')
- **port** – TCP port to connect to (default=2379)
- **srv_domain** – Domain name if DNS discovery is used
- **version_prefix** – API version prefix (default='v2')
- **allow_reconnect** – If client fails to connect to a cluster node connect to the next node in the cluster
- **protocol** – Protocol to connect to the cluster (default='http')

Raises ***ClientException*** – if any errors

compare_and_delete(*key*, *prev_value=None*, *prev_index=None*)

This command will delete a key only if the client-provided conditions are equal to the current conditions.

Parameters

- **key** – the key
- **prev_value** – checks the previous value of the key.
- **prev_index** – checks the previous modifiedIndex of the key.

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error.

EtcdNodeExist if any condition fails.

compare_and_swap(*key*, *value*, *prev_value=None*, *prev_index=None*, *prev_exist=None*, *ttl=None*)
This command will set the value of a key only if the client-provided conditions are equal to the current conditions.

Parameters

- **key** – key string
- **value** – key value
- **prev_value** – checks the previous value of the key.
- **prev_index** – checks the previous modifiedIndex of the key.
- **prev_exist** – checks existence of the key: if prevExist is True, it is an update request; if prevExist is False, it is a create request.
- **ttl** – set ttl on the key in seconds

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error.

EtcdNodeExist if condition prev_exist=False fails.

EtcdTestFailed if condition prev_value='bar' fails.

delete(*key*)
Delete a key

Parameters **key** – Key

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error

mkdir(*directory*)
Create directory

Parameters **directory** – string with directory name

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error

read(*key*, ***kwargs*)
Read key value

Parameters **key** – Key

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error

rmdir(*directory*, *recursive=False*)
Delete directory

Parameters

- **directory** – string with directory name
- **recursive** – recursively delete directory if not empty

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error

update_ttl(*key*, *ttl*)
Update key's ttl

Parameters

- **key** – the key
- **ttl** – new ttl

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error.

EtcdKeyNotFound if the key doesn't exist

version()
Return Etcd server version

Returns string with Etcd server version. E.g. '2.3.7'

version_cluster()
Return Etcd cluster version

Returns string with Etcd cluster version. E.g. '2.3.0'

version_server()
Same as .version()

Returns string with Etcd server version. E.g. '2.3.7'

write(*key*, *value*, *ttl=None*)
Write value to a key

Parameters

- **key** – Key
- **value** – Value
- **ttl** – Keys in etcd can be set to expire after a specified number of seconds. You can do this by setting a TTL (time to live) on the key.

Returns EtcdResult

Raises ***EtcdException*** – if etcd responds with error or HTTP error

exception pyetcd.client.**ClientException**
Bases: exceptions.Exception

Exception for errors in Client class

Module contents

exception pyetcd.**EtcdClientInternal**
Bases: *pyetcd.EtcdException*

Error ecodeClientInternal (http code 500)

exception pyetcd.**EtcdDirNotEmpty**
Bases: *pyetcd.EtcdException*

Error EcodeDirNotEmpty (http code 108)

exception pyetcd.**EtcdEventIndexCleared**
Bases: *pyetcd.EtcdException*

Error EcodeEventIndexCleared (http code 401)

exception pyetcd.**EtcdException**
Bases: exceptions.Exception

Generic Etcd error.

exception pyetcd.**EtcdExistingPeerAddr**
Bases: *pyetcd.EtcdException*

Error ecodeExistingPeerAddr (http code 109)

exception pyetcd.**EtcdIndexNaN**
Bases: *pyetcd.EtcdException*

Error EcodeIndexNaN (http code 203)

exception pyetcd.**EtcdIndexOrValueRequired**
Bases: *pyetcd.EtcdException*

Error ecodeIndexOrValueRequired (http code 207)

exception pyetcd.**EtcdIndexValueMutex**
Bases: *pyetcd.EtcdException*

Error ecodeIndexValueMutex (http code 208)

exception pyetcd.**EtcdInvalidActiveSize**
Bases: *pyetcd.EtcdException*

Error ecodeInvalidActiveSize (http code 403)

exception pyetcd.**EtcdInvalidField**
Bases: *pyetcd.EtcdException*

Error EcodeInvalidField (http code 209)

exception pyetcd.**EtcdInvalidForm**
Bases: *pyetcd.EtcdException*

Error EcodeInvalidForm (http code 210)

exception pyetcd.**EtcdInvalidRemoveDelay**
Bases: *pyetcd.EtcdException*

Error ecodeInvalidRemoveDelay (http code 404)

exception pyetcd.**EtcdKeyIsPreserved**
Bases: *pyetcd.EtcdException*

Error ecodeKeyIsPreserved (http code 106)

exception pyetcd.**EtcdKeyNotFound**
Bases: *pyetcd.EtcdException*

Error EcodeKeyNotFound (http code 100)

exception pyetcd.**EtcdLeaderElect**
Bases: *pyetcd.EtcdException*

Error EcodeLeaderElect (http code 301)

exception pyetcd.**EtcdNameRequired**
Bases: *pyetcd.EtcdException*

Error ecodeNameRequired (http code 206)

exception pyetcd.**EtcdNoMorePeer**
Bases: *pyetcd.EtcdException*

Error ecodeNoMorePeer (http code 103)

exception pyetcd.**EtcdNodeExist**
Bases: *pyetcd.EtcdException*

Error EcodeNodeExist (http code 105)

exception pyetcd.**EtcdNotDir**
Bases: *pyetcd.EtcdException*

Error EcodeNotDir (http code 104)

exception pyetcd.**EtcdNotFile**
Bases: *pyetcd.EtcdException*

Error EcodeNotFile (http code 102)

exception pyetcd.**EtcdPrevValueRequired**
Bases: *pyetcd.EtcdException*

Error EcodePrevValueRequired (http code 201)

exception pyetcd.**EtcdRaftInternal**
Bases: *pyetcd.EtcdException*

Error EcodeRaftInternal (http code 300)

exception pyetcd.**EtcdRefreshTTLRequired**
Bases: *pyetcd.EtcdException*

Error EcodeRefreshTTLRequired (http code 212)

exception pyetcd.**EtcdRefreshValue**
Bases: *pyetcd.EtcdException*

Error EcodeRefreshValue (http code 211)

class pyetcd.**EtcdResult**(*response*)
Bases: object

Response from Etcd API

Parameters **response** – Response from server as `requests.(get|post|put)` returns.

Raises ***EtcdException*** – if payload is invalid or contains errorCode.

action
Action type

followers
Followers of leader

id

leader
Leader of cluster

leaderInfo

name

node
Node class instance. It holds the current key value.

prevNode
Node class instance. It holds the previous key value.

recvAppendRequestCnt

sendAppendRequestCnt

startTime

state

version_etcdcluster
Version of Etcd cluster

version_etcdserver
Version of Etcd server

x_etcd_index

exception pyetcd.**EtcdRootROnly**
Bases: *pyetcd.EtcdException*

Error EcodeRootROnly (http code 107)

exception pyetcd.**EtcdStandbyInternal**
Bases: *pyetcd.EtcdException*

Error ecodeStandbyInternal (http code 402)

exception pyetcd.**EtcdTTLNaN**
Bases: *pyetcd.EtcdException*

Error EcodeTTLNaN (http code 202)

exception pyetcd.**EtcdTestFailed**
Bases: *pyetcd.EtcdException*

Error EcodeTestFailed (http code 101)

exception pyetcd.**EtcdTimeoutNaN**
Bases: *pyetcd.EtcdException*

Error ecodeTimeoutNaN (http code 205)

exception pyetcd.**EtcdUnauthorized**
Bases: *pyetcd.EtcdException*

Error EcodeUnauthorized (http code 110)

exception pyetcd.**EtcdValueOrTTLRequired**
Bases: *pyetcd.EtcdException*

Error ecodeValueOrTTLRequired (http code 204)

exception pyetcd.**EtcdValueRequired**
Bases: *pyetcd.EtcdException*

Error ecodeValueRequired (http code 200)

exception pyetcd.**EtcdWatcherCleared**
Bases: *pyetcd.EtcdException*

Error EcodeWatcherCleared (http code 400)

class pyetcd.**ResponseNode**(*key=None*, *value=None*, *created_index=None*, *modified_index=None*)
Bases: object

Etcd response includes information about nodes.

Parameters

- **key** – the HTTP path to which the request was made. We set /message to Hello world, so the key field is /message. etcd uses a file-system-like structure to represent the key-value pairs, therefore all keys start with /.
- **value** – the value of the key after resolving the request.
- **created_index** – an index is a unique, monotonically-incrementing integer created for each change to etcd. This specific index reflects the point in the etcd state member at which a given key was created
- **modified_index** – like node.createdIndex, this attribute is also an etcd index. Actions that cause the value to change include set, delete, update, create, compareAndSwap and compareAndDelete. Since the get and watch commands do not change state in the store, they do not change the value of node.modifiedIndex.

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/twindb/pyetcd/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/twindb/pyetcd/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 :)

Get Started!

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

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

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

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, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/twindb/pyetcd/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests/test_client.py
$ py.test tests/test_etcdresult.py
```

To run individual tests:

```
$ py.test tests/test_client.py::test_write
```

CHAPTER 6

Credits

Development Lead

- TwinDB Development Team <dev@twindb.com>

Contributors

None yet. Why not be the first?

CHAPTER 7

History

0.1.0 (2016-09-17)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

U

V

W

X