python-cueclient Documentation

Release 0.0.1

Openstack Cue Team

August 26, 2015

Contents

1 Getting Started **3**
- 1.1 Installing Cue Client from source . . . 3

2 Cue Command Line Tool **5**
- 2.1 Credentials . . . 5
- 2.2 Using the command line tool . . . 5
- 2.3 Cluster Create . . . 5
- 2.4 Cluster Show . . . 6
- 2.5 Cluster Delete . . . 6
- 2.6 Cluster List . . . 6
- 2.7 Subcommands . . . 7

3 Python Bindings **9**
- 3.1 Introduction . . . 9
- 3.2 Authentication . . . 10
- 3.3 Cue Functions . . . 10

4 Contributing **13**
- 4.1 Coding Standards . . . 13

5 Indices and tables **15**

Contents:

CHAPTER 1

Getting Started

The python-cueclient can be used either as a command line tool or as a binding to access Cue.

1.1 Installing Cue Client from source

1. Clone the CueClient repo from GitHub

```
$ git clone https://github.com/openstack/python-cueclient.git
$ cd python-cueclient
```

2. Setup virtualenv

Note: This is an optional step, but will allow CueClient's dependencies to be installed in a contained environment that can be easily deleted if you choose to start over or uninstall Cue.

```
$ virtualenv --no-site-packages .venv
$ . .venv/bin/activate
```

3. Install CueClient and its dependencies

```
$ pip install -r requirements.txt -r test-requirements.txt
$ python setup.py develop
```

1.1.1 Installation to use as Command Line Tool

4. To access the shell for cue client 'python-openstackclient' has to be installed.

```
$ pip install python-openstackclient
```

Note: This step can be skipped if you choose to use python-cueclient as only a binding to Cue API.

CHAPTER 2

Cue Command Line Tool

The python-cueclient can be used as a command line tool for accessing Cue API.

2.1 Credentials

As with any OpenStack utility, **python-cueclient** requires certain information to talk to the REST API, username, password, auth url (from where the other required endpoints are retrieved once you are authenticated).

To provide your access credentials (username, password, tenant name or project_name) you can pass them on the command line with the `--os-username`, `--os-password`, `--os-tenant-name` or `--os-project-name` params, but it's easier to just set them as environment variables:

```
export OS_USERNAME=<your_username>
export OS_PASSWORD=<your_password>
export OS_PROJECT_NAME=<project_name>
```

You will also need to define the authentication url with `--os-auth-url` or set is as an environment variable as well:

```
export OS_AUTH_URL=<url_to_openstack_identity>
```

Since Keystone can return multiple regions in the Service Catalog, you can specify the one you want with `--os-region-name` (or `export OS_REGION_NAME`). It defaults to the first in the list returned.

2.2 Using the command line tool

With enough details now in environment, you can use the cue client to create,list,show or delete cluster(s).

The Openstack Client can be called interactively by simply typing:

```
openstack
```

2.3 Cluster Create

Required fields for 'create' : name, network id , flavor and size.

```
(openstack) message-broker cluster create --name cluster_04 --nic 3dd26c0b-03f2-4d2e-ae87-c02d7f33c78
+-------------+--------------------------------------+
| Field       | Value                                |
```

```
+-------------+--------------------------------------+
| created_at  | 2015-02-17T18:25:28+00:00            |
| endpoints   | []                                   |
| flavor      | 2                                    |
| id          | 06d3c0e4-4972-4ca9-91c1-373b1c74e8e1 |
| name        | cluster_04                           |
| network_id  | 3dd26c0b-03f2-4d2e-ae87-c02d7f33c788 |
| size        | 3                                    |
| status      | BUILDING                             |
| updated_at  | 2015-02-17T18:25:28+00:00            |
| volume_size | None                                 |
+-------------+--------------------------------------+
```

2.4 Cluster Show

Required field for 'show' : cluster-id

```
(openstack) message-broker cluster show 06d3c0e4-4972-4ca9-91c1-373b1c74e8e1
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2015-02-17T18:25:28+00:00            |
| endpoints   | []                                   |
| flavor      | 2                                    |
| id          | 06d3c0e4-4972-4ca9-91c1-373b1c74e8e1 |
| name        | cluster_04                           |
| network_id  | 3dd26c0b-03f2-4d2e-ae87-c02d7f33c788 |
| size        | 3                                    |
| status      | BUILDING                             |
| updated_at  | 2015-02-17T18:25:28+00:00            |
| volume_size | None                                 |
+-------------+--------------------------------------+
```

2.5 Cluster Delete

Required field for 'delete' : cluster-id

```
(openstack) message-broker cluster delete 06d3c0e4-4972-4ca9-91c1-373b1c74e8e1
```

2.6 Cluster List

```
(openstack) message-broker cluster list
+--------------------------------------+-------------+----------+--------+------+
| id                                   | name        | status   | flavor | size |
+--------------------------------------+-------------+----------+--------+------+
| 06d3c0e4-4972-4ca9-91c1-373b1c74e8e1 | cluster_04  | DELETING | 2      |    3 |
| 09fa2dc2-7ebb-423f-9726-f45b53f0df99 | cluster_02  | DELETING | 1      |    3 |
| 2d6a5359-2c45-44bb-baa9-3ccd2a48c217 | cluster_03  | BUILDING | 2      |    2 |
+--------------------------------------+-------------+----------+--------+------+
```

2.7 Subcommands

Here are the full list of subcommands:

subcommand	Notes
message-broker cluster create	Create Cluster
message-broker cluster delete	Delete Cluster
message-broker cluster show	Show Cluster
message-broker cluster list	List Clusters

CHAPTER 3

Python Bindings

The python-cueclient can be used to interact with the Cue API from any other python program.

3.1 Introduction

Below is a simple example of how to instantiate and perform basic tasks using the bindings.

```
#!/usr/bin/env python

from keystoneclient.auth.identity import v3 as keystone_v3_auth
from keystoneclient import session as keystone_session
from cueclient.v1 import client

auth = keystone_v3_auth.Password(
        auth_url="http://example.com:5000/v3",
        username="admin",
        password="password",
        project_name="admin",
        project_domain_name="default",
        user_domain_name="default"
)

session = keystone_session.Session(auth=auth)

# Create an instance of the client
cue_client = client.Client(session=session)

# Cluster List - returns list of cluster objects
list_response = cue_client.clusters.list()

# Iterate the list, printing some useful information
for cluster in list_response:

    print "Cluster ID: %s \t Name: %s \t NetworkId: %s \t Flavor: %s \t Size: %s" % \
            (cluster.id, cluster.name, cluster.network_id, cluster.flavor, cluster.size)
```

And the output this program might produce:

```
$ python /example.py
Cluster ID: 213cdd06-c361-4cca-93b5-7ed651d46936       Name: test_binding2     NetworkId: 33333
Cluster ID: 24f299fd-0509-4218-bf80-6c0481452480       Name: test_binding4     NetworkId: 44444
```

3.2 Authentication

Cue supports Keystone authentication.

3.2.1 Keystone Authentication

Below is a sample of standard authentication with keystone v3:

```
#!/usr/bin/env python

from keystoneclient.auth.identity import v3 as keystone_v3_auth
from keystoneclient import session as keystone_session

auth = keystone_v3_auth.Password(
        auth_url="http://example.com:5000/v3",
        username="admin",
        password="password",
        project_name="admin",
        project_domain_name="default",
        user_domain_name="default"
)
```

3.3 Cue Functions

3.3.1 Cluster List

```
#!/usr/bin/env python

from keystoneclient.auth.identity import v3 as keystone_v3_auth
from keystoneclient import session as keystone_session
from cueclient.v1 import client

auth = keystone_v3_auth.Password(
        auth_url="http://example.com:5000/v3",
        username="admin",
        password="password",
        project_name="admin",
        project_domain_name="default",
        user_domain_name="default"
)

session = keystone_session.Session(auth=auth)
cue_client = client.Client(session=session)

# Cluster List
list_response = cue_client.clusters.list()
```

3.3.2 Cluster Show

```
#!/usr/bin/env python

from keystoneclient.auth.identity import v3 as keystone_v3_auth
```

```
from keystoneclient import session as keystone_session
from cueclient.v1 import client

auth = keystone_v3_auth.Password(
        auth_url="http://example.com:5000/v3",
        username="admin",
        password="password",
        project_name="admin",
        project_domain_name="default",
        user_domain_name="default"
)

session = keystone_session.Session(auth=auth)
cue_client = client.Client(session=session)

cluster_id = "0a352f9a-8aa8-411e-9d6d-4e6217d70afd"

# Cluster Show
show_response = cue_client.clusters.get(cluster_id)
```

3.3.3 Cluster Create

```
#!/usr/bin/env python

from keystoneclient.auth.identity import v3 as keystone_v3_auth
from keystoneclient import session as keystone_session
from cueclient.v1 import client

auth = keystone_v3_auth.Password(
        auth_url="http://example.com:5000/v3",
        username="admin",
        password="password",
        project_name="admin",
        project_domain_name="default",
        user_domain_name="default"
)

session = keystone_session.Session(auth=auth)
cue_client = client.Client(session=session)

# Cluster create
create_response = cue_client.clusters.create(name="test_binding5",
                    nic="55555", flavor="1",size="2",volume_size="0")
```

3.3.4 Cluster Delete

```
#!/usr/bin/env python

from keystoneclient.auth.identity import v3 as keystone_v3_auth
from keystoneclient import session as keystone_session
from cueclient.v1 import client

auth = keystone_v3_auth.Password(
        auth_url="http://example.com:5000/v3",
        username="admin",
```

```
        password="password",
        project_name="admin",
        project_domain_name="default",
        user_domain_name="default"
)

session = keystone_session.Session(auth=auth)
cue_client = client.Client(session=session)

delete_id = "dc86d96f-6b37-4e2d-9805-4542450f427d"

# Cluster Delete
delete_response = cue_client.clusters.delete(delete_id)
```

CHAPTER 4

Contributing

Code is hosted on GitHub. Submit bugs to the Cue Client project on Launchpad. Submit code to the openstack/python-cueclient project using Gerrit.

Here's a quick summary:

Install the git-review package to make life easier

```
pip install git-review
```

Branch, work, & submit:

```
# cut a new branch, tracking master
git checkout --track -b bug/id origin/master
# work work work
git add stuff
git commit
# rebase/squash to a single commit before submitting
git rebase -i
# submit
git-review
```

4.1 Coding Standards

Cue Client uses the OpenStack flake8 coding standards guidelines. These are stricter than pep8, and are run by gerrit on every commit.

You can use tox to check your code locally by running

```
# For just flake8 tests
tox -e flake8
# For tests + flake8
tox
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

I

install
    python-cueclient, 3

P

python-cueclient
    install, 3