Libra Client Documentation

Release 2015-10-17-beta

Andrew Hutchings

October 17, 2015

Contents

CHAPTER 1

Introduction

Libra Client is a Python command line client which is used to manipulate Atlas API compatible OpenStack Load Balancer as a Service installations.

It is designed to be similar to the Python Nova client and in fact uses this as a common base code.

CHAPTER 2

Installation

2.1 From Ubuntu Package via PPA

1. Install utility

```
sudo apt-get install python-software-properties
```

2. Add the PPA

```
sudo apt-add-repository ppa:libra-core/ppa
```

3. Update the package indexes

```
sudo apt-get update
```

4. Install packages

```
sudo apt-get install python-libraclient
```

2.2 From PyPI

The **python-libraclient** package is published on PyPI and so can be installed using the pip tool, which will manage installing all python dependencies.

Note: The pip tool isn't bundled by default with some versions of the different distributions, please install it typically using a package manager for the platform you use.

Note: Needs to be done in a Virtual Environment or as root.

```
pip install python-libraclient
```

Warning: The packages on PyPI may lag behind the git repo in functionality.

2.3 Setup the client from source

If you want the latest version, straight from github:

```
git clone git@github.com:stackforge/python-libraclient.git
cd python-libraclient
virtualenv .venv
source .venv/bin/activate
pip install -r requirements.txt -r test-requirements.txt
python setup.py install
```

2.4 Setup the client in development mode

Installing in development mode allows your to make changes to the source code & test directly without having to re-run the "python setup.py install" step. You can find out more about this in the Development Mode online docs.

```
git clone git@github.com:stackforge/python-libraclient.git
cd python-libraclient
virtualenv .venv
source .venv/bin/activate
pip install -r requirements.txt -r test-requirements.txt
python setup.py develop
```

CHAPTER 3

Usage

3.1 Synopsis

libra [*GENERAL OPTIONS*] [*COMMAND*] [*COMMAND_OPTIONS*]

3.2 Description

libra is a utility designed to communicate with Atlas API based Load Balancer as a Service systems.

3.3 Global Options

--help, -h
Show help message and exit

--debug
Turn on HTTP debugging for requests

--insecure
Don't validate SSL certs

--os_bypass_url <os_bypass-url>
URL to use as an endpoint instead of the one specified by the Service Catalog

--service_type <service-type>
Alternative service type to use for your cloud provider (default is 'hpext:lbaas')

--os_auth_url <auth-url>
The OpenStack authentication URL. Default is OS_AUTH_URL or LIBRA_URL environment variables

--os_username <auth-user-name>
The user name to use for authentication. Default is OS_USERNAME or LIBRA_USERNAME environment variables

--os_password <auth-password>
The password to use for authentication. Default is OS_PASSWORD or LIBRA_PASSWORD environment variables

--os_tenant_name <auth-tenant-name>
The tenant to authenticate to. Default is OS_TENANT_NAME or LIBRA_PROJECT_ID environment variables

--os_region_name <region-name>
The region the load balancer is located. Default is OS_REGION_NAME or LIBRA_REGION_NAME environment variables

3.4 Client Commands

3.4.1 algorithms

Gets a list of supported algorithms

3.4.2 create

Create a load balancer

--name <name>
The name of the node to be created

--port <port>
The port the load balancer will listen on

--protocol <protocol>
The protocol type for the load balancer (HTTP, TCP or GALERA). The Galera option adds support for deadlock avoidance in Galera clusters, see Serveral Nine's Blog on this.

--node <ip:port:option=value:...>
The IP and port for a load balancer node (can be used multiple times to add multiple nodes). Additional node options may be specified after the ip:port portion in a option=value format.

--vip <vip>
The virtual IP ID of an existing load balancer to attach to

3.4.3 delete

Delete a load balancer

<id>
The ID of the load balancer

3.4.4 limits

Show the API limits for the user

3.4.5 list

List all load balancers

--deleted
Show deleted load balancers

3.4.6 logs

Send a snapshot of logs to an object store

<id>
The ID of the load balancer

--storage <store>
Storage type

--endpoint <endpoint>
Object store endpoint to use

--basepath <basepath>
Object store based directory

--token <token>
Object store authentication token

3.4.7 update

Update a load balancer's configuration

<id>
The ID of the load balancer

--name <name>
A new name for the load balancer

--algorithm <algorithm>
A new algorithm for the load balancer

3.4.8 monitor-list

List the health monitor for a load balancer

<id>
The ID of the load balancer

3.4.9 monitor-delete

Delete the health monitor for a load balancer

<id>
The ID of the load balancer

3.4.10 monitor-update

Update the health monitor for a load balancer

<id>
The ID of the load balancer

3.4.11 node-add

Add a node to a load balancer

<id>
The ID of the load balancer

--node <ip:port:option=value:...>
The node address in ip:port format (can be used multiple times to add multiple nodes). Additional node options may be specified after the ip:port portion in a option=value format.

3.4.12 node-delete

Delete a node from the load balancer

<id>
The ID of the load balancer

<nodeid>
The ID of the node to be removed

3.4.13 node-list

List the nodes in a load balancer

<id>
The ID of the load balancer

3.4.14 node-update

Update a node's state in a load balancer

<id>
The ID of the load balancer

<nodeid>
The ID of the node to be updated

--condition <condition>
The new state of the node (either ENABLED or DISABLED)

3.4.15 node-show

Get the status of a node in a load balancer

<id>
The ID of the load balancer

<nodeid>
The ID of the node in the load balancer

3.4.16 protocols

Gets a list of supported protocols

3.4.17 show

Get the status of a single load balancer

<id>
The ID of the load balancer

3.4.18 virtualips

Get a list of virtual IPs

<id>
The ID of the load balancer

CHAPTER 4

Examples

4.1 Create Load Balancer

```
libra --os_auth_url=https://company.com/openstack/auth/url \
--os_username=username --os_password=pasword --os_tenant_name=tenant \
--os_region_name=region create --name=my_load_balancer \
--node 192.168.1.1:80 --node 192.168.1.2:80
```

This example will create a basic load balancer which will listen on port 80 and direct traffic in a round-robin fashion to two nodes, 192.168.1.1 and 192.168.1.2. Both these nodes are web servers listening on port 80. The Libra Client will then return a table similar to the below:

Property	Value
status updated protocol name algorithm created virtualIps port nodes id	BUILD 2013-10-31T11:59:24 HTTP test ROUND_ROBIN 2013-10-31T11:59:24 <VIP: 359 - PUBLIC IPV4 15.125.20.157> 80 <Node: 15.126.201.193:80> <Node: 15.126.201.70:80> 80303

4.2 Create a Load Balancer with Node Options

```
libra --os_auth_url=https://company.com/openstack/auth/url \
--os_username=username --os_password=pasword --os_tenant_name=tenant \
--os_region_name=region create --name=my_load_balancer \
--node 192.168.1.1:80:weight=1 --node 192.168.1.2:80:weight=2
```

Nearly identical to the above example, this creates a new load balancer with two nodes, but one is more heavily weighted than the other, causing it to accept more traffic.

4.3 Create a Shared Load Balancer

It is possible for a single logical load balancer to balancer traffic for both HTTP and HTTPS for a site. For this example we will add an HTTPS load balancer to the load balancer we created previously:

```
libra --os_auth_url=https://company.com/openstack/auth/url \
--os_username=username --os_password=pasword --os_tenant_name=tenant \
--os_region_name=region create --name=my_load_balancer \
--node 192.168.1.1:443 --node 192.168.1.2:443 --protocol=TCP --port=443 \
--vip=52
```

We have taken the IP ID which was provided in the original create and given this as a VIP number in the command. We are also setting to TCP mode so the SSL termination happens at the web server and set the load balancer to listen on port 443. The result is as follows:

Property	Value
status updated protocol name algorithm created virtualIps port nodes id	BUILD 2013-10-31T11:59:24 HTTP test ROUND_ROBIN 2013-10-31T11:59:24 <VIP: 359 - PUBLIC IPV4 15.125.20.157> 80 <Node: 15.126.201.193:80> <Node: 15.126.201.70:80> 80303

4.4 Add a Node

```
libra --os_auth_url=https://company.com/openstack/auth/url \
--os_username=username --os_password=pasword --os_tenant_name=tenant \
--os_region_name=region node-add 158 --node=192.168.1.3:443
```

In this example we have take the ID of the load balancer of the previos example to add a web server to. The result should look something like this:

ID	Address	Port	Condition	Status
	192.168.1.3	443		

Symbols

L