An lc-tools Tutirial

Roman Bogorodskiy
novel@FreeBSD.org

May 30, 2011

Contents

0.1 Introduction

An `lc-tools` is a set of command line tools to manage various Cloud (aka IaaS) Providers. It's written in Python and uses libcloud to interact with provider's API.xx

0.2 Getting Started

0.2.1 Installing

Dependencies

The only external dependency is `libcloud`. Please visit libcloud download page to get information how to download and install it.

Installing from PyPI

Latest stable version of `lc-tools` could be installed from PyPI:

```
easy_install lctools
```

Installing from source

To get latest development version you can checkout sources from project's github page:

```
git clone https://github.com/novel/lc-tools.git
```

Now you should have all the sources and should be ready to proceed to installation. As `lc-tools` use `setuptools` installation process is fairly simple:

```
$ cd lc-tools
$ sudo python setup.py install
```

This will install all the tools and you will be able to use them after configuration (you will know how to configure `lc-tools` in the next section: **??**).

However, there are some additional tools available in `lc-tools` that's not installed by default – it's various provider specific tools.

What are provider specific tools, you might ask. You see, `libcloud` is designed to provide an unified API to the cloud, so its model is almost an intersection of APIs of various cloud providers. However, various provider can have its specific API calls, for example, provider Foo might have a call to return information how many servers could be created in your current account or, say, what's the maximum allowed rate of requests to the API per minute. Usually such features are implemented in `libcloud` as extra calls and are not available in common API. So in order to be able to use this features, provider specific tools has to be written.

Currently, `lc-tools` supports such scripts for Rackspace and GoGrid. In order to enable them both, please use the command:

```
$ sudo python setup.py install --providertools="gg rs"
```

This means that we're installing additional tools for GoGrid (gg) and Rackspace (rs). Please refer to **??** to details on usage of this tools.

Note!: latest development version requires latest `libcloud` sources!

0.2.2 Configuring

All configuration data is stored in text file `/.lcrc`. It doesn't exist by default so you have to create it:

```
touch ~/.lcrc
chmod 600 ~/.lcrc
```

We need `chmod` to be sure others can't access you secret keys. If file mode will be too permissive, `lc-tools` will not run.

Structure of the file is following:

```
[default]
driver = foogrid
access_id = your_key_id
secret_key = your_password

[barcloud]
driver = barcloud
access_id = somekey
secret_key = some_key
```

You probably noted that config file is separated by sections (**default**, **barcloud** in the example above). Each section corresponds to a single cloud account, it's possible to switch them and **default** section is used by default and should always be present.

All fields are necessary and cannot be omitted.

driver
: name of the specific cloud provider for the account. To get a list of all available drivers please execute

```
lc-drivers-list|cut -d "." -f3|sort|uniq
```

access_id
: access_id for the account. It could be either login or some generated key

secret_key
: secretkey for the account, in other words it's a password for given access_id

Please refer to your cloud prover for detailed documentation on getting account's credentials.

0.3 Basic Usage

0.3.1 General Info

Lc-tools package consists of several tools each of them does its own thing: listing images, creating nodes, etc. An easy way to learn what tools exist type `lc-` or `lb-` in your shell's prompt and press Tab key to auto complete.

Every tools allows to switch configuration profile (please refer to **??** if you don't know what it is). This is done using `-p` switch like this:

```
lc-node-list -p myacc2
```

Where `myacc2` is a name of profile you want to use. Again, this feature works for every tool, not only `lc-node-list`.

0.3.2 Compute

Glossary

Various cloud providers use different terms like images, flavours, nodes, servers etc. We will just use terms used in libcloud internally.

Node
: Single instance of *virtual server* on the cloud.

Image
: Understand it as a *server template*, the base from your nodes will deliver from.

Size
: Size of the *node*, i.e. hardware features of *node*. It could imply RAM, disk space and so on.

Listing Images

You can list available images using `lc-image-list` tool. Here is a sample of its output:

```
$ lc-image-list|grep -i centos
image CentOS 5.2 (32-bit) w/ RightScale (id = 62)
image CentOS 5.2 (64-bit) w/ RightScale (id = 63)
image CentOS 5.3 (32-bit) w/ None (id = 1531)
image CentOS 5.3 (64-bit) w/ None (id = 1532)
$
```

The first line is a shell command we've issued. I've piped it to `grep` to output only images containing 'centos' it their names, otherwise the list would be too long.

For every image you see its name and id, for example for *CentOS 5.3 (64-bit) w/ None* id will be `1532`. Please remember it because you will need it if you're going to use this image.

Listing Sizes

Sizes can be viewed using `lc-sizes-list` tool. An example of its output for GoGrid:

```
$ lc-sizes-list
     size 512MB (id=512MB, ram=512, disk=30 bandwidth=None)
     size 4GB (id=4GB, ram=4096, disk=240 bandwidth=None)
     size 2GB (id=2GB, ram=2048, disk=120 bandwidth=None)
     size 8GB (id=8GB, ram=8192, disk=480 bandwidth=None)
     size 1GB (id=1GB, ram=1024, disk=60 bandwidth=None)
     $
```

Please note that ids are not always numeric as could be seen by this example. So, if we want to create, say, 2GB server, we should use id `2GB`.

Node Creation

Now as we've learned how to observe available images and sized we can proceed with node creation using `lc-node-add` tool.

It's as simple as:

```
$ lc-node-add -i 62 -s 1GB -n mynewnode
$
```

Here we created a new node with image `id = 62`, size `id = 1GB` and name `mynewnode`.

In the next section we will discuss how to list nodes to make sure your new node created and obtain details about it.

Listing Nodes

Getting list of nodes is not harder than getting list of images or sizes:

```
$ lc-node-list
100xxx  mynode1     173.204.xx.yy  Running
100xxx  mynode2    173.204.xx.zz  Running
$
```

Format of this output is following:

image_id	name	public_ips	state

Where:

image_id
Id of the node automatically generated by cloud provider

name
Name of the node you gave at creation.

public_ip
Comma-separated list of public ips assigned to the node.

status
Status of the node, helps to understand if node is usable or not.

Operations on Individual Nodes

Tool called `lc-node-do` allows to operate on individual nodes. Currently, it allows rebooting and destroying (deleting) nodes.

It can be done this way:

```
$ lc-node-do -i 123 destroy # for deleting
$ lc-node-do -i 124 reboot  # for rebooting
```

Here an argument for `-i` switch is an id of the node we're working with and the next argument is an action, i.e. what we want to do with the node.

It's possible to specify more than one node id at time, for example:

```
$ lc-node-do -i 10,34,98 destroy
$
```

This command will destroy nodes with ids: 10, 34 and 98. Also, it's possible to specify ranges of ids like that:

```
$ lc-node-do -i 100-119 destroy
$
```

This will destroy nodes with ids starting from 100 and ending with 119.

0.3.3 Load Balancers

Glossary

Load Balancer
Load Balancer instance.

Member
Member node of the `Load Balancer`. Load Balancer balances requests it receives across all the Member nodes registered to it.

Load Balancer Creation

To create a new load balancer `lb-add` tool should be used. Here's an example:

```
$ lb-add -n hellobalancer -P 80 192.168.0.1:80 192.168.0.2:8080
$
```

We just created a balancer with name `hellobalancer` listening on port `80` and balancing load across two nodes: `192.168.0.1:80` and `192.168.0.2:8080`.

Listing Load Balancers

List of currently active Load Balancers could be obtained using `lb-list`:

```
$ lb-list
12345 testlb979ec2c6 10.0.5.52:80
12346 testlba6f9c38b 10.0.53.71:80
12347 hellobalancer 10.0.5.128:80
$
```

Format of the output is following:

balancer_id	name	endpoint_ip

Operations on Members

You can view list of Load Balancer Member nodes using `lb-member-list` tool like that:

```
$ lb-member-list -i 12347
<Member: id=33385, address=192.168.0.1:80>
<Member: id=33388, address=192.168.0.2:8080>
$
```

You just specify id of the Balancer you're working with (in our case it's `12347`) and get a list of members attached to it.

To add a new node to the Load Balancer you need to know its IP and port and pass it to `lb-member-add` tool:

```
$ lb-member-add -i 12347 192.168.0.3:8080
$
```

And, finally, to remove a member from Load Balancer, use `lb-member-remove`:

```
$ lb-member-remove -i 12347 33388
```

In this case we have to specify id on the Load Balancer and id of the member we're removing (id of the member could be obtained using `lb-member-list`.

Destroying Load Balancer

Load Balancer can be removed using `lb-destroy` command like that:

```
$ lb-destroy -i 12347
```

You just need to pass id of the Balancer you want to destroy.

0.4 Provider Specific

As it was discussed earlier, `lc-tools` comes with support for additional non-standard features of some cloud providers. Some of these features are available out of the box, and others needs to be configured. Every section has an *Availability* keyword at the very beginning. It could either be: *Availability: default* which would mean that feature should be available without extra configuration, or *Availability: gg*, which would mean that you should configure `lc-tools` to install GoGrid specific scripts passing

```
--providerspecific=gg
```

to `setup.py install`. Please refer to **??** for more details on that.

0.4.1 GoGrid

Controlling IP Addresses

Availability: gg

By default, the first available IP address is used for a new node. However, you can explicitly specify IP address you want to use for a node using extra argument `ip` for `lc-node-add`:

```
$ lc-node-add -s 1GB -i 123 -n foo ip=1.2.3.4
```

You can find out what IP addresses are available for you account using `lc-gg-ip-list` tool:

```
$ lc-gg-ip-list
ip 10.0.175.66 (id=12345, public=False, state=Unassigned, subnet=10.0.175.0/255.255.255.0)
ip 10.0.175.67 (id=12346, public=False, state=Unassigned, subnet=10.0.175.0/255.255.255.0)
...
$
```

There are several filtering options available in `lc-gg-ip-list`:

-a use `-a true` to list only assigned addresses and `-a false` to list unassigned ones

-u use `-u true` to show only public addresses and `-a false` to list private addresses

Creating a Sandbox Server

Availability: default

In terms of GoGrid sandbox server is a server which will be used as a base for some server image. It can be created by passing extra argument `ex_issandbox` like that:

```
$ lc-node-add -s 1GB -i 123 -n mysandbox ex_issandbox=true
```

The created server will be a sandbox server.

Please refer to GoGrid wiki to get more details on sandboxes and server images.

Creating an Image from Sandbox

Availability: gg

Once you have prepared your sandbox server for image creation you can do it with `lc-gg-image-save` tool:

```
$ lc-gg-image-save -i 123 -n mynewimage
```

Where `123` is an id of the sandbox server to be used as a base for new image and `mynewimage` is a name you want to give to your new image.

Note: at the time of writing preparing an image is not quite trivial and involves logging in to the box over ssh and running some scripts. Please refer to GoGrid wiki for details.

Resizing a Node

Availability: gg

GoGrid has introduced an ability to change a size of running node in API version `1.6`. Suppose, we have a node with id `100` and 1Gb amount of RAM and we want to increase it to 2Gb. In this case `lc-gg-node-edit` tool may prove helpful:

```
$ lc-gg-node-edit -i 100 -s 2GB
```

Here, `-i` switch specifies id of a node we want to modify and `-s` specifies id of a size we want to use for the node.

0.4.2 Rackspace

Checking Limits

Availability: rs

Rackspace binds various limits to accounts. These limits might be absolute, like a maximum total value of RAM consumed by all servers or these might be daily limits (i.e. they are reseted periodically), for example number of POST requests per minute.

`lc-rs-limits` tool displays current state of limits for your Rackspace account:

```
$ lc-rs-limits
Rates:

    verb             URI value  remaining     unit resetTime
     PUT               *   10         10   MINUTE 2011-01-06 16:12:04
     GET *changes-since*    3          3   MINUTE 2011-01-06 16:12:04
    POST               *   10         10   MINUTE 2011-01-06 16:12:04
    POST       /servers*  500        500      DAY 2011-01-06 16:12:04
  DELETE               *  600        600   MINUTE 2011-01-06 16:12:04

Absolute limits:

maxPrivateIPs: 10
maxTotalRAMSize: 921600
maxIPGroupMembers: 25
maxIPGroups: 25
```

Please refer to Rackspace documentation for better information of the limits information presented by `lc-rs-limits`.