Working in the multi-cloud with libcloud

Grig Gheorghiu
Evite
grig.gheorghiu@evite.com

Multi-cloud – what is that?

- The Cloud is becoming ubiquitous
  - Amazon EC2, Rackspace Cloud, Joyent, GoGrid, Terremark, vCloud, Linode, Dreamhost, etc.
- Any given cloud is FLAKY
- Multi-Cloud advantages
  - High-availability/fault tolerance
  - Disaster recovery
  - Performance (geolocation closer to users)

Multi-cloud projects

- Commercial vendors
  - RightScale (mostly EC2, Rackspace in Beta)
  - Cloudkick (now owned by Rackspace)
  - enStratus
- Open Source tools
  - OpenStack (Python)
  - libcloud (Python)
  - jclouds (Java)
  - deltacloud (Ruby)
  - Overmind (Django + libcloud)

libcloud: an introduction

- Python library for multi-cloud management
- Started by CloudKick, now in Apache Incubator
- Deals with instances, i.e. compute, not storage
  - Storage in the works in 2011
- Goes for breadth instead of depth
  - Supports 18 cloud providers
  - Main operations supported: list, create, destroy

libcloud: main concepts

- Provider: a cloud vendor
- Driver: encapsulates common provider operations, as well as specific provider functionality
  - Example: EC2 key operations
- Image: OS flavor available for deployment
- Size: amount of compute, storage and network capacity that a given instance will use
- Location: data center hosting the instance (e.g. EC2 availability zone)

libcloud and SSL

- Current libcloud version: 0.4.2
- New feature: SSL cert validation
- Install ssl module for Python 2.5
- Download CA cert bundle if you don't have it
- Enable SSL cert validation
  - libcloud.security.VERIFY_SSL_CERT = True
  - libcloud.security.CA_CERTS_PATH.append("/path/to/cacert.pem")

Connecting to a provider

EC2 example

```
from libcloud.types import Provider

from libcloud.providers import get_driver

EC2_ACCESS_ID  = ”MY_ACCESS_ID”

EC2_SECRET_KEY  = ”MY_SECRET_KEY”

EC2Driver = get_driver(Provider.EC2)

conn = EC2Driver(EC2_ACCESS_ID, EC2_SECRET_KEY)
```

Connecting to a provider (cont.)

Rackspace example

```
from libcloud.types import Provider

from libcloud.providers import get_driver

USER  = ”MY_USER”

API_KEY  = ”MY_API_KEY”

RackspaceDriver = get_driver(Provider.Rackspace)

conn = RackspaceDriver(USER, API_KEY)
```

Working with images

- list_images returns list of NodeImage objects (no less than 7,271 images for EC2 East!)

```
images = conn.list_images()

images[0].__dict__

{'extra': {}, 'driver': <libcloud.drivers.ec2.EC2NodeDriver
    object at 0xb7f95d8c>, 'id': 'aki-00806369', 'name': 'karmic-
    kernel-zul/ubuntu-kernel-2.6.31-300-ec2-i386-20091001-
    test-04.manifest.xml'}
```

- Creating a NodeImage object in EC2 based on AMI ID:

```
image = NodeImage(id="ami-014da868", name="", driver="")
```

Working with sizes

- list_sizes returns list of NodeSize objects

```
sizes = conn.list_sizes()

sizes[0].__dict__

{'name': 'Large Instance', 'price': '.38', 'ram': 7680, 'driver':
    <libcloud.drivers.ec2.EC2NodeDriver object at
    0xb7f10d8c>, 'bandwidth': None, 'disk': 850, 'id': 'm1.large'}
```

- Creating a NodeSize object in EC2 based on id:

```
size = NodeSize(id="m1.small", name="", ram=None,
    disk=None, bandwidth=None, price=None, driver="")
```

Working with locations

- Locations are not very well supported
- list_locations returns list of NodeLocation objects

```
locations = conn.list_locations()

locations[0].__dict__

{'country': 'US', 'availability_zone': <ExEC2AvailabilityZone:
    name=us-east-1a, zone_state=available, region_name=us-
    east-1>, 'driver': <libcloud.drivers.ec2.EC2NodeDriver object
    at 0xb7f31d6c>, 'id': '0', 'name': 'Amazon US N. Virginia'}
```

- To get a NodeSize object in EC2, iterate through list until location.availability_zone.name == 'us-east-1N' (N=a,b,c,d)

Working with EC2 keys

- A key is not required to create an EC2 node, but you can't ssh without a key...
- Method specific to EC2 driver: ex_create_keypair

```
keyname = sys.argv[1]

resp = conn.ex_create_keypair(name=keyname)

key_material = resp.get('keyMaterial')

private_key = '/root/.ssh/%s.pem' % keyname

f = open(private_key, 'w'); f.write(key_material + '\n'); f.close()

os.chmod(private_key, 0600)
```

Launching a node/instance

- Common create_node API
  - Name, image and size are required
  - Location and key are useful
  - Instantiate image, size, location either from scratch or by choosing from list of objects based on criteria such as object name or id
- EC2 example

```
node = conn.create_node(name='testEC2', image=image,
    size=size, location=location, ex_keyname=keyname)
```

- If location is not specified, random availability zone is chosen
- Can also specify security group via ex_security_group param

Launching a node/instance (cont.)

- node.__dict__ can be inspected for info on newly created instance (esp the 'extra' key):

{'name': 'i-f692ae9b', 'extra': {'status': 'running', 'productcode': [], 'groups': ['default'], 'instanceId': 'i-f692ae9b', 'dns_name': 'ec2-184-72-92-114.compute-1.amazonaws.com', 'launchdatetime': '2010-12-14T20:25:22.000Z', 'imageId': 'ami-014da868', 'kernelid': None, 'keyname': 'k1', 'availability': 'us-east-1d', 'launchindex': '0', 'ramdiskid': None, 'private_dns': 'domU-12-31-39-04-65-11.compute-1.internal', 'instancetype': 'm1.small'}, 'driver': <libcloud.drivers.ec2.ec2nodedriver 0x93f42cc="" at="" object="">, 'public_ip': ['ec2-184-72-92-114.compute-1.amazonaws.com'], 'state': 0, 'private_ip': ['domU-12-31-39-04-65-11.compute-1.internal'], 'id': 'i-f692ae9b', 'uuid': '76fcd974aab6f50092e5a637d6edbac140d7542c'}

Launching a node/instance (cont.)

- Rackspace example
- No key needed, ssh is via password initially (make sure you get password immediately after node creation)

```
node = conn.create_node(name='testrackspace', image=image,
    size=size)

print node.__dict__
```


```
{'name': 'testrackspace', 'extra': {'metadata': {}, 'password':
    'testrackspaceO1jk6O5jV', 'flavorId': '1', 'hostId':
    '9bff080afbd3bec3ca1400483110 49f9', 'imageId': '71'}, 'driver':
    <libcloud.drivers.rackspace.rackspacenodedriver 0x877c3ec="" at=""
    object="">, 'public_ip': ['184.106.187.226'], 'state': 3, 'private_ip':
    ['10.180.67.242'], 'id': '497741', 'uuid':
    '1fbf7c3fde339af9fa901af6bf0b73d4d10472bb'}
```

Listing nodes

nodes = conn.list_nodes()

EC2 example:

{'name': 'i-118c297d', 'extra': {'status': 'running', 'productcode': [], 'groups': ['default'], 'instanceId': 'i-118c297d', 'dns_name': 'ec2-50-16-89-82.compute-1.amazonaws.com', 'launchdatetime': '2011-01-19T18:11:16.000Z', 'imageId': 'ami-014da868', 'kernelid': None, 'keyname': 'lc1', 'availability': 'us-east-1a', 'clienttoken': '', 'launchindex': '0', 'ramdiskid': None, 'private_dns': 'ip-10-112-37-252.ec2.internal', 'instancetype': 'm1.small'}, 'driver': <libcloud.drivers.ec2.EC2NodeDriver object at 0xb7b3dcec>, 'public_ip': ['ec2-50-16-89-82.compute-1.amazonaws.com'], 'state': 0, 'private_ip': ['ip-10-112-37-252.ec2.internal'], 'id': 'i-118c297d', 'uuid': 'be685979ef0284653b9435c3c5a064d5b2b4d2fc'}

{'name': 'i-878c29eb', 'extra': {'status': 'pending', 'productcode': [], 'groups': ['default'], 'instanceId': 'i-878c29eb', 'dns_name': '', 'launchdatetime': '2011-01-19T18:12:20.000Z', 'imageId': 'ami-014da868', 'kernelid': None, 'keyname': 'lc2', 'availability': 'us-east-1a', 'clienttoken': '', 'launchindex': '0', 'ramdiskid': None, 'private_dns': '', 'instancetype': 'm1.small'}, 'driver': <libcloud.drivers.ec2.EC2NodeDriver object at 0xb7b3dcec>, 'public_ip': [''], 'state': 3, 'private_ip': [''], 'id': 'i-878c29eb', 'uuid': '944cb9d05f37f7a73022e3ebadbb238951448cec'}

Listing nodes (cont.)

Rackspace example:

{'name': 'testrackspace2', 'extra': {'metadata': {}, 'password': None, 'flavorId': '1', 'hostId': '8f113614123c0f70a0d6c89e48f6a297', 'imageId': '71'}, 'driver': <libcloud.drivers.rackspace.RackspaceNodeDriver object at 0xb7b0deac>, 'public_ip': ['184.106.65.161'], 'state': 3, 'private_ip': ['10.180.188.178'], 'id': '558564', 'uuid': '043af166941ea03cef160614264585d4e5685e75'}

- Password is not shown anymore

Destroying nodes

EC2 example:

```
id_to_destroy = 'i-878c29eb'

nodes = conn.list_nodes()

for node in nodes:

    if node.name == id_to_destroy:

        conn.destroy_node(node)
```

- For EC2 list_nodes will show the node in 'terminated' state for a while

Destroying nodes (cont.)

Rackspace example:

```
name_to_destroy = 'testrackspace2'
nodes = conn.list_nodes()
for node in nodes:
    if node.name == name_to_destroy:
        conn.destroy_node(node)
```

- For Rackspace list_nodes will not show the node anymore

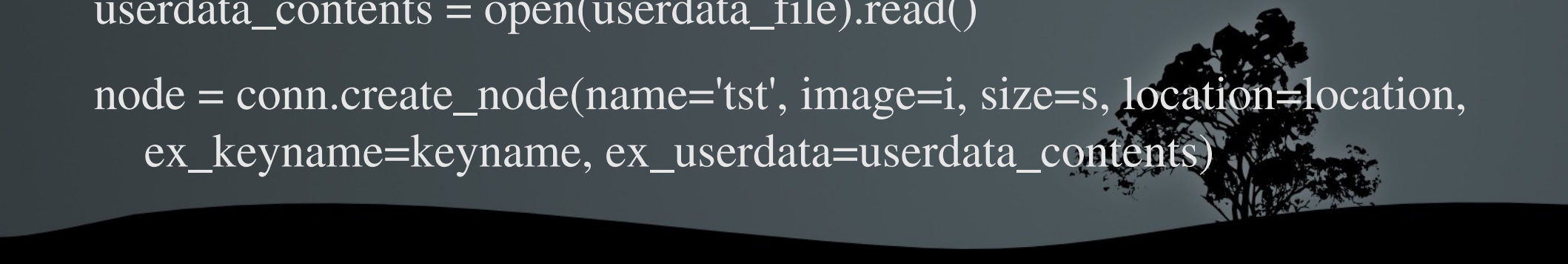
Bootstrapping EC2 nodes with user data

- Ubuntu EC2 instances automatically run scripts passed via user data
- EC2 libcloud driver has extra ex_userdata parameter for create_node
- BUT! Need to pass contents of user data file and not path to the file!

```
userdata_file = "/root/proj/test_libcloud/userdata.sh"

userdata_contents = open(userdata_file).read()

node = conn.create_node(name='tst', image=i, size=s, location=location,
    ex_keyname=keyname, ex_userdata=userdata_contents)
```



Bootstrapping Rackspace nodes with deployment scripts

- Rackspace libcloud driver can use
  - SSHKeyDeployment
  - ScriptDeployment
  - MultiStepDeployment

```
sd = SSHKeyDeployment(open("/root/.ssh/id_rsa.pub").read())

script = ScriptDeployment("apt-get -y install munin-node")

msd = MultiStepDeployment([sd, script])

node = conn.create_node(name='testrackspace2', image=image,
    size=size, deploy=msd)
```

The Overmind Project

- Django Web app based on libcloud
- Create providers (EC2, Rackspace, Hetzner)
- Create, list, reboot, destroy nodes
- REST API available
- Next version: queuing (RabbitMQ + celery)
- Future versions: configuration management with chef-solo, monitoring
- https://github.com/tobami/overmind/