Controlling the cloud with Python

Pycon.it
9 maggio 2009

Me...

Luca Mearelli

http://luca.im

@lmea

kiaraservice.com

Cloud?

*-as-a-service
elastic & fluid
scalable
programmable

Amazon Web Services

EC2

S3

CloudFront

SimpleDB

SQS

Elastic MapReduce

MechanicalTurk

...

Boto

http://code.google.com/p/boto

```
python setup.py install
export AWS_ACCESS_KEY_ID=<your key>
export AWS_SECRET_ACCESS_KEY=<your secret>
# ~/.boto
# /etc/boto.cfg
```

S3 - Simple Storage Service

storage service

scalable

replicated & distributed

web aware (HTTP / bittorrent)

flexible security

eventually consistent

S3 - Connecting

```
import boto
conn = boto.connect_s3()
conn = S3Connection()

conn.is_secure
conn = boto.connect_s3(is_secure=False)
```

S3 - Buckets

```
bucket = conn.create_bucket('pycon3')
rs = conn.get_all_buckets()
for bucket in rs :
    print bucket.name

bucket.delete()
conn.delete_bucket('pycon3')
```

S3 - Keys & Objects

```
from boto.s3 import Key
key = Key(bucket)
key = bucket.new_key('aaaa')

key.name = 'aaaa'
key.exists()
```

S3 - Keys & Objects

```
key.set_contents_from_string(
        'My Test',
        headers={'Content-Type':'text/plain'})

key.set_contents_from_filename('test.txt')
```

S3 - Keys & Objects

```
rs = bucket.get_all_keys()
rs = bucket.get_all_keys(prefix='test/level1/',
                         delimiter='/',
                         maxkeys=5)
key = bucket.get_key('demo/foo')
```

S3 - Keys & Objects

```
key = bucket.get_key('foo')

fp = open('foo.txt', 'wb')
key.get_contents_to_file(fp)

s = key.get_contents_as_string()
```

S3 - Keys & Objects

```
key.set_metadata('format', 'avi')
key.get_metadata('format')
```

S3 - Permissions

```
bucket.make_public(recursive=True)
key.make_public()
key.generate_url(10)
key.set_acl('public-read-write')
CannedACLStrings = ['private',
                    'public-read',
                    'public-read-write',
                    'authenticated-read']
```

S3 - Logging

```
lb = conn.create_bucket('pycon3.logging')
pb = conn.create_bucket('pycon3.public')
lb.set_as_logging_target()
pb.enable_logging(lb)
```

EC2 - Elastic Compute Cloud

Computing on demand

Any application on any OS

Persistent/ephemeral storage

Server customization

Flexible security

EC2 - Connecting

```
conn = boto.connect_ec2()
conn = EC2Connection()

regions = boto.regions()
conn = regions[0].connect()
```

EC2 - Images

```
rs = conn.get_all_images(owners=['012345678901'])

image = conn.get_image(image_id='ami-5647a33f')
```

EC2 - Keypairs

```
key_pair = conn.create_key_pair('my_key')
key_pair.save('/Users/luca/.ssh')
```

EC2 - Instances

```
res = conn.run_instances('ami-5647a33f',
                         instance_type='m1.small',
                         key_name='my_key',
                         min_count=1, max_count=1,
                         security_groups=['web'],
                         user_data=None)

inst = res.instances[0]
```

EC2 - Instance

```
while not instance.update() == 'running':
    print instance.state
    time.sleep(5)

print inst.id
print inst.public_dns_name
print inst.private_dns_name
print inst.state
print inst.key_name
print inst.launch_time
```

EC2 - Instance

```
inst.stop()
res.stop_all()

inst.reboot()

console = inst.get_console_output()
print console.instance_id
print console.timestamp
print console.output
```

EC2 - Firewall

```
group = conn.create_security_group(
                    group_name='web',
                    group_desc='Web Servers')

rs = conn.get_all_security_groups()
```

EC2 - Firewall

```
conn.authorize_security_group(
     'web',
     ip_protocol='tcp',
     from_port='80',
     to_port='80',
     cidr_ip='0.0.0.0/0')

group.revoke('tcp', 80, 80, '0.0.0.0/0')
```

EC2 - Elastic block storage

```
ebs = conn.create_volume(size,
                         zone,
                         snapshot=None)

conn.attach_volume(volume_id=ebs.volume_id,
                   instance_id=inst.id,
                   '/sdb')

conn.create_snapshot(volume_id)
```

SQS - Simple Queue Service

Distributed queue

Web scale

Redundant infrastructure

SQS - Connecting

```
conn = boto.connect_sqs()
conn = SQSConnection()
```

SQS - Queue

```
queue = conn.create_queue('myqueue')
queue.url
#'https://queue.amazonaws.com/591131556747/myqueue'
queue.get_timeout()
queue.set_timeout(120)
rs = conn.get_all_queues()
queue = conn.get_queue('myqueue')
```

SQS - Messages

```
from boto.sqs import Message

msg = Message()
msg.set_body('A test message')
msg.attributes['timestamp'] = 1202131

queue.write(msg)
```

SQS - Messages

```
Message()
MHMessage()
RawMessage()
JSONMessage()

queue.set_message_class(MHMessage)
```

SQS - Messages

```
queue.count()

rs = queue.get_messages(num_messages=1,
                        visibility_timeout=None)

msg = queue.read(visibility_timeout=None)

msg.get_body()
```

SQS - Messages

```
msg.delete()

queue.clear()
```

SQS - Dumping & loading

```
queue.save_to_file(fp, sep='\n')
queue.load_from_file(fp)

queue.save_to_s3(bucket, prefix=None)
queue.load_from_s3(bucket, prefix=None)
```

SDB - Simple DB

Structured data storage

Schema-free

Queryable

SDB - Connecting

```
conn = boto.connect_sdb()
conn = SDBConnection()
```

SDB - Domains

```
dom = conn.create_domain('pycon3')
dom = conn.lookup('pycon3')

rs  = conn.get_all_domains()
```

SDB - Domains

```
md = conn.get_metadata()
md.item_count
```

SDB - Items

```
item = dom.new_item('item1')
item['k1'] = 'value'
item['k2'] = 10
item['k3'] = ['a','b','c']
item.add_value('k3', 'd')
item.save()
```

SDB - Read & query

```
dom.get_item('item1')

rs = dom.query("['k1' = 'value']")

rs = dom.select(
        "select * from pycon3 where k1 = 'value'")

for item in rs :
      print item.name
```

SDB - Query examples

```
"['city' = 'Seattle' or 'city' = 'Portland']"

"['author' starts-with 'Robert']"

"['author' does-not-start-with'Robert']"

"['first name' = 'John'] intersection ['last name'
= 'Smith']"

"['tag' starts-with 'Amazon'] union ['description'
= 'SimpleDB']"

"not ['country' = 'USA' or 'country' = 'UK']"
```

SDB - Select examples

```
"select * from mydomain where city = 'Seattle' or
city = 'Portland'"

"select * from mydomain where author like 'Rob%'"

"select * from mydomain where year is not null"

"select * from mydomain where every(keyword) in
('Book', 'Paperback') "

"select itemName() from mydomain"

"select count(*) from mydomain"
```

SDB - Dumping to xml

```
d = dom.to_xml()
```

???

Grazie!

Luca :-)

@lmea #pycontre #aws

http://lmea-docs.s3.amazonaws.com/controlling_the_cloud_with_python