Intro to Boto

http://boto.cloudhackers.com/

(and gevent, http://www.gevent.org/)

Presented 2011-07-01
tommi.virtanen@dreamhost.com

What is Boto

The Python client library for interacting with AWS APIs. Plural.

Tutorial at http://boto.cloudhackers.com/s3_tut.html

API reference at http://boto.cloudhackers.com/ref/s3.html

This presentation is no substitute for those.

Basic usage

```
>>> import boto.s3
>>> from boto.s3.connection import S3Connection
>>> conn = S3Connection('ACCESSKEY', 'SECRET')
>>> bucket = conn.create_bucket('mybucket')
>>> key = conn.new_key('foo')
>>> key.set_contents_from_string('bar')
```

Buckets and keys

```
bucket = connection.get_bucket(name)
bucket = connection.get_bucket(name, validate=False)
bucket = connection.create_bucket(name)

key = bucket.get_key(name)
key = bucket.new_key(name)
```

~/.boto can make interactive use nicer

[Credentials]
aws_access_key_id = ...
aws_secret_access_key = ...

chmod go= ~/.boto
environment is always public,
get rid of bad habits

```
>>> from boto.s3.connection import S3Connection
>>> conn = S3Connection()
```

RGW needs special stuff

```
>>> conn = boto.s3.connection.S3Connection(
...   host='localhost',
...   port=7280,
...   is_secure=False,
...   calling_format=
...     boto.s3.connection.OrdinaryCallingFormat(),
...   )
```

For an example of how to store this in ~/.boto:
https://github.com/tv42/s3-url-sign
(should submit upstream...)

640kB should be enough for everyone

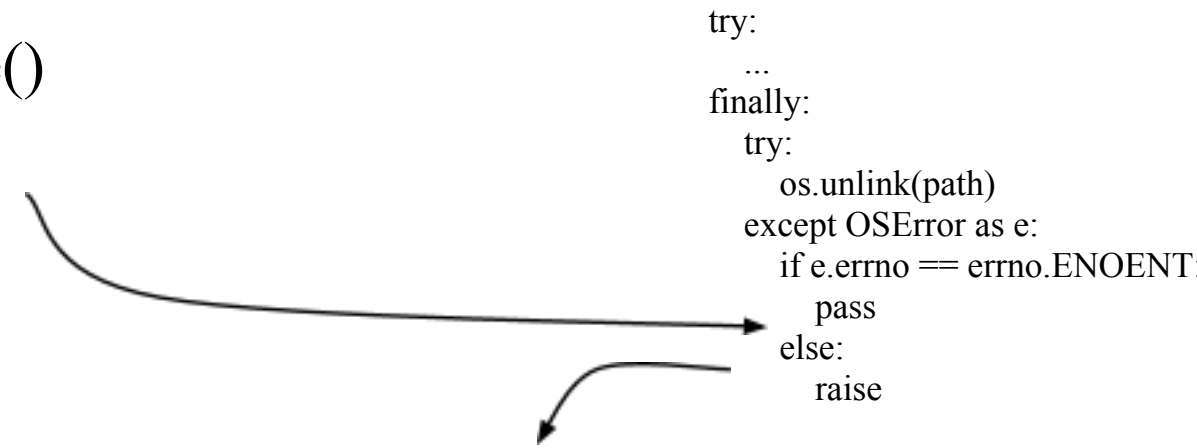
```
# store in RAM
key.set_contents_from_string(s)
s = key.get_contents_as_string()

# store in a file
key.set_contents_from_filename('/path/to/file.ext')

path = os.path.join(tmpdir, 'foo')
key.get_contents_to_filename(path)
```

Temp files don't need names

```
path = os.path.join(tmpdir, 'foo')
key.get_contents_to_filename(path)
with file(path) as f:
    firstline = f.readline()
os.unlink(path)
```



```
try:
    ...
finally:
    try:
        os.unlink(path)
    except OSError as e:
        if e.errno == errno.ENOENT:
            pass
        else:
            raise
```

Better:

```
with tempfile.SpooledTemporaryFile() as f:
    key.get_contents_to_file(f)
    f.seek(0)
    firstline = f.readline()
```

Generate content dynamically (quack like a duck)

```
class CastleOfAaaaaargh(object):
    def __init__(self, size):
        self.size = size
        self.offset = 0
    def seek(self, offset):
        self.offset = offset
    def tell(self):
        return self.offset
    def read(self, size=-1):
        if size < 0:
            size = self.size - self.offset
        size = min(size, self.size - self.offset)
        self.offset += size
        return size*'A'
```

Many hands make light work

http://www.gevent.org/

greenlets, not threads; lightweight, single core
schedule at IO or sleep

```
>>> from gevent import monkey
>>> monkey.patch_all()

>>> import gevent
>>> g = gevent.spawn(sum, [1, 2, 3])
>>> g.get()
6
>>> sum([1,2,3])
6
```

Group/Pool for lots of similar tasks

```
from gevent import monkey; monkey.patch_all()

import gevent.pool

def plus((a, b)):
    return a+b

group = gevent.pool.Group()
print group.map(
    plus,
    ((42, x) for x in xrange(100)),
    )
```

```
[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105,
106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141]
```

Group/Pool for lots of similar tasks

```
from gevent import monkey; monkey.patch_all()

import gevent.pool
import time

def sorter(n):
    time.sleep(n/10.0)
    return n

group = gevent.pool.Group()
gen = group.imap_unordered(sorter, xrange(9, 0, -1))
for result in gen:
    print result
```

Write objects in parallel

... and if that doesn't make it go fast enough,
do multipart uploads!

Questions?

Thank you