OpenStack API's and WSGI

Mike Pittaro
La Honda Research Center
mikeyp@LaHondaResearch.org
@pmikeyp

Introduction

- Taking a peek under the hood of OpenStack
  - From the API inwards
- Prerequisites
  - A basic understanding of Web, REST, and HTTP
  - Some knowledge of Python
- Why learn this ?
  - Understand OpenStack's API implementation
  - Easier to understand and troubleshoot
  - First step to modifying OpenStack
  - First step to extending API's

How Deep is a 'peek' ?

- *Everything should be made as simple as possible, but not simpler.*
    - Albert Enistein

- *Simple is Better than Complex*
- *Complex is better than Complicated*
    - PEP 20

What is OpenStack ?

- OpenStack is a global collaboration of developers and cloud computing technologists producing the ubiquitous open source cloud computing platform for public and private clouds.
- The project aims to deliver solutions for all types of clouds by being simple to implement, massively scalable, and feature rich.
- The technology consists of a series of interrelated projects delivering various components for a cloud infrastructure solution.

How is OpenStack Implemented?

- OpenStack is a collection of services
  - Compute (Nova)
  - Object Storage (Swift)
  - Image Service (Glance)
  - Identity (Keystone)
  - Dashboard (Horizon)
- Each service is a 'WebApp'
  - REST API server ('frontend')
  - One or more backend servers
  - Messaging interface between them

OpenStack API's

- All Interaction with OpenStack is via API's
  - http://docs.openstack.org/api/
  - http://api.openstack.org/
- API QuickStart
  - http://docs.openstack.org/api/quick-start/content/
- The API's use HTTP + json (or xml)
  - Use curl or wget or browser plugins
  - Use any programming language via HTTP libraries
  - Use the Python novaclient library

OpenStack In Action

- OpenStack includes a nova command
  - It's built using the novaclient library

```
mikeyp@blade1:devstack$ nova --username admin --password devstack image-list
+--------------------------------------+---------------------------------------------+--------+--------+
|                  ID                  |                     Name                    | Status | Server |
+--------------------------------------+---------------------------------------------+--------+--------+
| 43bafe10-700c-45af-90a8-b5d794812e62 | cirros-0.3.0-x86_64-blank-ramdisk           | ACTIVE |        |
| 45ad4046-9780-4968-83c6-460f168321c7 | cirros-0.3.0-x86_64-blank-kernel            | ACTIVE |        |
| 6216fc7c-7f87-45e0-be0f-eefef2d5be33 | ttylinux-uec-amd64-11.2_2.6.35-15_1         | ACTIVE |        |
| 92a1e0bd-c4a5-4f3f-a66f-1f8b990f2b0e | ttylinux-uec-amd64-11.2_2.6.35-15_1-kernel  | ACTIVE |        |
| 95d8db11-b175-43d2-b3de-d7b806e54dde | cirros-0.3.0-x86_64-blank                   | ACTIVE |        |
| e543bb77-5a7d-4ef0-9a7a-92ca6c8a0b35 | cirros-0.3.0-x86_64-rootfs                  | ACTIVE |        |
+--------------------------------------+---------------------------------------------+--------+--------+

mikeyp@blade1:devstack$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+
| ID |    Name   | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor |
+----+-----------+-----------+------+-----------+------+-------+-------------+
| 1  | m1.tiny   | 512       | 0    | 0         |      | 1     | 1.0         |
| 2  | m1.small  | 2048      | 10   | 20        |      | 1     | 1.0         |
| 3  | m1.medium | 4096      | 10   | 40        |      | 2     | 1.0         |
| 4  | m1.large  | 8192      | 10   | 80        |      | 4     | 1.0         |
| 5  | m1.xlarge | 16384     | 10   | 160       |      | 8     | 1.0         |
+----+-----------+-----------+------+-----------+------+-------+-------------+
```

Using novaclient

```
#!/usr/bin/env python

import logging

import novaclient
from novaclient.v1_1 import client

# enable debug logging
logger = logging.getLogger('novaclient.client')
logger.setLevel(logging.INFO)
debug_stream = logging.StreamHandler()
logger.addHandler(debug_stream)

auth_url = 'http://10.100.20.22:5000/v2.0'
user = 'admin'
password = 'devstack'
project = 'demo'
region = 'RegionOne'
service = 'compute'

nova = client.Client(user, password, project, auth_url,
                     region_name=region, service_type=service)

results = nova.images.list(detailed=True)
for image in results:
    print image.id, image.name, image.status
```

```
mikeyp@blade1:api_examples$ python image_list.py
e543bb77-5a7d-4ef0-9a7a-92ca6c8a0b35 cirros-0.3.0-x86_64-rootfs ACTIVE
95d8db11-b175-43d2-b3de-d7b806e54dde cirros-0.3.0-x86_64-blank ACTIVE
45ad4046-9780-4968-83c6-460f168321c7 cirros-0.3.0-x86_64-blank-kernel ACTIVE
43bafe10-700c-45af-90a8-b5d794812e62 cirros-0.3.0-x86_64-blank-ramdisk ACTIVE
92a1e0bd-c4a5-4f3f-a66f-1f8b990f2b0e ttylinux-uec-amd64-11.2_2.6.35-15_1-kernel ACTIVE
6216fc7c-7f87-45e0-be0f-eefef2d5be33 ttylinux-uec-amd64-11.2_2.6.35-15_1 ACTIVE
```

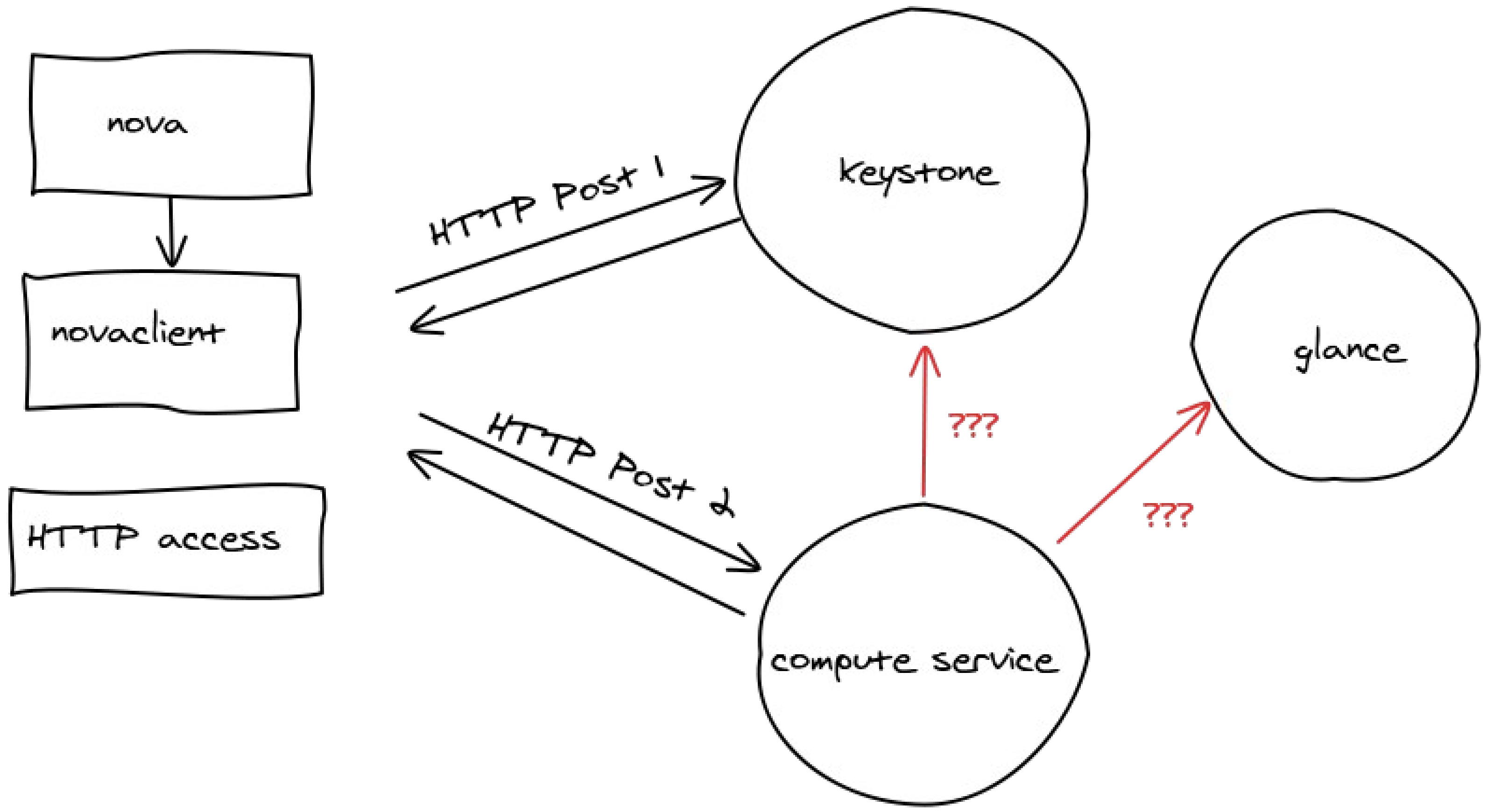
Keystone API using urllib2

```
def get_keystone_token():
    """authenticate against keystone identity service
    returns an auth token, and the service url

    """
    user = 'admin'
    password = 'devstack'
    project = 'demo'
    auth_url = 'http://10.100.20.22:5000/v2.0/tokens'

    auth_request = urllib2.Request(auth_url)
    auth_request.add_header('Content-Type', 'application/json;charset=utf8')
    auth_request.add_header('Accept', 'application/json')
    auth_request.add_header('User-Agent', 'python-mikeyp')

    auth_data = {"auth":
        {"tenantName": project,
         "passwordCredentials": {
            "username": user,
            "password": password}
        }
    }
    auth_request.add_data(json.dumps(auth_data))
    auth_response = urllib2.urlopen(auth_request)
    response_data = json.loads(auth_response.read())

    token = response_data['access']['token']['id']

    service_list = response_data['access']['serviceCatalog']
    for s in service_list:
        if s['type'] == 'compute' and s['name'] == "'Compute Service'":
            break
    nova_url = s['endpoints'][0]['publicURL']
    return (token, nova_url)
```

Images API using urllib2

```
#!/usr/bin/env python

import urllib2
import json

# def get_keystone_token():
   # see previous page

token, service_url  = get_keystone_token()

image_api = service_url + '/images/detail'

images_request = urllib2.Request(image_api)
images_request.add_header('Content-Type', 'application/json;charset=utf8')
images_request.add_header('Accept', 'application/json')
images_request.add_header('User-Agent', 'python-mikeyp')
images_request.add_header('X-Auth-Token', token)
images_request.add_header('X-Auth-Project-Id', 'demo')

image_response = urllib2.urlopen(images_request)
image_data = json.loads(image_response.read())
print json.dumps(image_data, indent=4)
```

What's been happening ?



OpenStack 'Web Stack'

- Paste HTTP Server
  - HTTP protocol + networking
- WebOb requests and responses
  - Wrappers for HTTP Requests and Responses
- OpenStack code
  - Nova, glance, keystone, etc
- Web Service Gateway Interface (WSGI)
  - The specification for web servers and applications
  - WSGI is not code – no import

WSGI In a Nutshell

- WSGI Application
  - A Python callable passed two arguments:
    - WSGI Environment
    - A start_response function
  - Application calls start_response, and returns response
- WSGI Server
  - The Server calls the application
- Middleware
  - Both a server and application
  - Use to 'wrap' or 'pipeline' requests

Simple WSGI Application

```
"""Hello World using Paste + WSGI """

from paste import httpserver

def application(environ, start_response):
    start_response('200 OK', [('Content-type', 'text/html')])
    return ['Hello World']

httpserver.serve(application, host='127.0.0.1', port=8080)
```

WSGI With WebOb + Paste

wsgi_webob.py

```
"""Hello World using WebOb, Paste + WSGI """

from webob import Response
from webob.dec import wsgify

from paste import httpserver
from paste.deploy import loadapp

INI_PATH = '/home/mikeyp/Documents/Projects/OpenStack/presentations/api_examples/wsgi_webob.ini'

@wsgify
def application(request):

    return Response('Hello, World of WebOb !')


def app_factory(global_config, **local_config):
    return application

wsgi_app = loadapp('config:' + INI_PATH)

httpserver.serve(wsgi_app, host='127.0.0.1', port=8080)
```

wsgi_webob_ini.py

```
[app:main]
paste.app_factory = wsgi_webob:app_factory
```

WSGI middleware

```python
"""Hello World (authorized version) using WebOb, Paste + WSGI """

from webob import Response
from webob.dec import wsgify
from webob import exc

from paste import httpserver
from paste.deploy import loadapp

INI_PATH = '/home/mikeyp/Documents/Projects/OpenStack/presentations/api_examples/wsgi_webob_mid.ini'

@wsgify
def application(request):

    return Response('Hello, Secret World of WebOb !')

@wsgify.middleware
def auth_filter(request, app):

    if request.headers.get('X-Auth-Token') != 'open-sesame':
        return exc.HTTPForbidden()
    return app(request)

def app_factory(global_config, **local_config):
    return application

def filter_factory(global_config, **local_config):
    return auth_filter

wsgi_app = loadapp('config:' + INI_PATH)

httpserver.serve(wsgi_app, host='127.0.0.1', port=8080)
```

WSGI Middleware Config

```
[pipeline:main]
pipeline = auth hello

[app:hello]
paste.app_factory = wsgi_webob_mid:app_factory

[filter:auth]
paste.filter_factory = wsgi_webob_mid:filter_factory
```

Glance API Server – the code

- Paste Config file
  - etc/glance-api-config.py
- Glance API server startup
  - glance/common/wsgi.py
  - glance/api/v1/router.py
- Main glance api files
  - glance/api/v1/images.py

Keystone middleware

- Authentication Token Verification
  - keystone/middleware/auth_token.py
  - WSGI middleware
  - Contains filter factory

Details and Complexity

- Lots more to learn – but not tonight
- Pluggable OpenStack API Extensions
- Front ends and load balancing
- Threading and concurrency
- URL mapping and dispatch

References

- OpenStack
  - http://www.openstack.org
- WSGI
  - http://www.wsgi.org
- Paste Web Server
  - http://pythonpaste.org/
- WebOb
  - http://www.webob.org/
- P3333 (WSGI)
  - http://www.python.org/dev/peps/pep-3333/
- HTTP RFC 2616
  - http://www.ietf.org/rfc/rfc2616.txt
- RESTful Web Services (Book)
  - http://shop.oreilly.com/product/9780596529260.do