Distributed Systems

06r. Introduction of OpenStack

Paul Krzyzanowski
TA: Yuanzhen Gu
Rutgers University
Fall 2014
OpenStack Overview

• OpenStack is an Infrastructure as a service (IaaS) which is know as a Cloud Operating System, that takes resources such as compute, storage, network, virtualization technologies and controls those resources at a data center level.

• OpenStack’s basic requirement: “cloud must be simple to implement and massively scalable.”
Community with Broad Commercial Support
Why OpenStack

Datacenters are being virtualized, Servers are first
Hypervisors provide abstraction between SW and HW (Servers)

- Hardware abstraction for each server
- Better resource utilization for each server

Next: Storage, Network...the building blocks

- Compute Pool Virtualized Servers
- Network Pool Virtualized Networks
- Storage Pool Virtualized Storage

- Flexibility, Efficiency are key drivers
- Resource pools for apps starting to form...

1. Virtualization
2. Cloud Data Center
3. Cloud Federation

Automation & Efficiency
Why OpenStack

But questions arise as the environment grows...
“VM sprawl” can make things unmanageable very quickly

- How do you make your apps cloud aware?
- Where should you provision new VMs?
- How do you empower employees to self-service?
- How do you keep track of it all?

A Cloud Management Layer Is Missing

1. Virtualization
2. Cloud Data Center
3. Cloud Federation

Automation & Efficiency
Here’s solution: OpenStack

Solution: OpenStack, The Cloud Operating System
A new management layer that adds automation and control

- Connects to apps via APIs
- Self-service Portals for users
  - USERS
  - ADMINS

CLOUD OPERATING SYSTEM

- Creates Pools of Resources
- Automates The Network

1. Server Virtualization
2. Cloud Data Center
3. Cloud Federation

Automation & Efficiency
OpenStack Project

- **OpenStack Compute (Nova):** Provision OpenStack Compute: provision and manage large networks of virtual machines
- **OpenStack Object Store (Swift):** Create petabytes of reliable storage using standard servers
- **OpenStack Image Service (Glance):** Catalog and manage large libraries of server images
- **OpenStack Quantum Service:** provide Network as a service to compute.
- **Other components:** Dashboard, Authentication(Keystone), CLI...
Identity ("Keystone")

- Keystone provides a single point of integration for OpenStack policy, catalog, token and authentication.
- **keystone** handles API requests as well as providing configurable catalog, policy, token and identity services.
- Standard backends include LDAP or SQL, as well as Key Value Stores (KVS).
- Most people will use this as a point of customization for their current authentication services.
Keystone Main Functions

• **Provides 4 primary services:**
  – Identity: User information authentication
  – Token: After logged in, replace account-password
  – Service catalog: Service units registered
  – Policies: Enforces different user levels

• **Can be backed by different databases.**
  – LDAP
  – SQL
  – Key Value Stores (KVS)
Keystone -- Process

Client → Keystoe (Request)
  ⏯ Reject unauthorized request

Keystone → OpenStack Service (Forward authorized request)
  ⏯ Use token to obtain tenant, roles, etc.
root@mtinjopenstack01:/home/yg250y# keystone service-list
Expecting authentication method via
  either a service token, --os-token or env[OS_SERVICE_TOKEN],
  or credentials, --os-username or env[OS_USERNAME].
root@mtinjopenstack01:/home/yg250y# source keystonerc
root@mtinjopenstack01:/home/yg250y# keystone service-list

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0a0457e93a9b4d5986fe2613cb147a41</td>
<td>cinder</td>
<td>volume</td>
<td>Volume Service</td>
</tr>
<tr>
<td>9075766af3c4d77857543cc5b633006</td>
<td>ec2</td>
<td>ec2</td>
<td>EC2 Compatibility Layer</td>
</tr>
<tr>
<td>9cf0f4e4d30a4b5a969f0c69fa57a6fc</td>
<td>glance</td>
<td>image</td>
<td>Image Service</td>
</tr>
<tr>
<td>ea20cd08b20b44ca92597132dd4a08d0</td>
<td>keystone</td>
<td>identity</td>
<td>Identity Service</td>
</tr>
<tr>
<td>eb9a07f19824401a456c854b58e9dbe</td>
<td>nova</td>
<td>compute</td>
<td>Compute Service</td>
</tr>
<tr>
<td>c2af4a5c1a5648ac9e27928954bf23e5</td>
<td>quantum</td>
<td>network</td>
<td>OpenStack Networking Service</td>
</tr>
<tr>
<td>6273c40470be4250bd515f0b69fab9c0</td>
<td>swift</td>
<td>object-store</td>
<td>Object Storage Service</td>
</tr>
</tbody>
</table>
Object Storage (“Swift”)

- Stores and serves objects (files)
- Employs object level replication to safeguard data
- Accepts client requests via Objectstore API or HTTP from clients through **swift-proxy**
- Maintains distributed account and container databases
- Stores objects according the ring layout on filesystem with extended attributes (XFS, EXT4, etc.)
Glance

- Image storage and indexing.
- Keeps a database of metadata associated with an image, discover, register, and retrieve.
- Built on top of Swift, images store in Swift
- Two servers:
  - Glance-api: public interface for uploading and managing images.
  - Glance-registry: private interface to metadata database
- Support multiple image formats
**Image Service ("Glance")**

- **glance-api** accepts Image API calls for image discovery, image retrieval and image storage.
- **glance-registry** stores, processes and retrieves metadata about images (size, type, etc.).
- Database to store the image metadata.
- A storage repository for the actual image files. In many deployments, this is OpenStack Swift.
root@mtinjopenstack01:/home/yg250y# glance image-list
+---------------------------------+-----------+---------------------+-------------------+----------------+------------------+
| ID                             | Name      | Disk Format         | Container Format  | Size           | Status           |
+---------------------------------+-----------+---------------------+-------------------+----------------+------------------+
| f3780eef-f881-410d-92d5-3309ad0eca6d | CirrOS 0.3.1 | qcow2              | bare              | 13147648       | active           |
+---------------------------------+-----------+---------------------+-------------------+----------------+------------------+

root@mtinjopenstack01:/home/yg250y# nova image-list
+---------------------------------+-----------+---------+-----------+
| ID                             | Name      | Status  | Server    |
+---------------------------------+-----------+---------+-----------+
| f3780eef-f881-410d-92d5-3309ad0eca6d | CirrOS 0.3.1 | ACTIVE |           |
+---------------------------------+-----------+---------+-----------+

root@mtinjopenstack01:/home/yg250y#
Block Storage ("Cinder")

- **cinder-api** accepts API requests and routes them to cinder-volume for action.
- **cinder-volume** acts upon the requests by reading or writing to the Cinder database to maintain state, interacting with other processes (like cinder-scheduler) through a message queue and directly upon block storage providing hardware or software. It can interact with a variety of storage providers through a driver architecture. Currently, there are drivers for IBM, SolidFire, NetApp, Nexenta, Zadara, Linux iSCSI and other storage providers.
- Much like nova-scheduler, the **cinder-scheduler** daemon picks the optimal block storage provider node to create the volume on.

```
root@mtinjopenstack01:/home/yg250y# cinder list

+-------------------------------+--------+-------------+--------+-------------+--------+-------------+-----------+---------------------+
| ID       | Status  | Display Name | Size   | Volume Type | Bootable | Attached to |
|----------+---------+--------------+--------+-------------+----------+------------|
| 0d98766d-b0b2-422a-a615-ded50fff473c | available | test        | 1      | None        | false    |

root@mtinjopenstack01:/home/yg250y#  
```
Nova

- **Major components:**
  - **API**: public facing interface
  - **Message Queue**: Broker to handle interactions between services, currently based on RabbitMQ
  - **Scheduler**: coordinates all services, determines placement of new resources requested
  - **Compute Worker**: hosts VMs, controls hypervisor and VMs when receives cmds on Msg Queue
  - **Volume**: manages permanent storage
Compute (“Nova”)

- **nova-api** accepts and responds to end user compute API calls.
- Supports OpenStack Compute API, Amazon’s EC2 API and a special Admin API (for privileged users to perform administrative actions).
- Initiates most of the orchestration activities (such as running an instance).
- Enforces some policy (mostly quota checks).
- Authentication is handled through middleware before getting to this daemon.
Nova Compute

- The **nova-compute** process is primarily a worker daemon that creates and terminates virtual machine instances via hypervisor's APIs (XenAPI for XenServer/XCP, libvirt for KVM or QEMU, VMwareAPI for VMware, etc.).

- The process by which it does so is fairly complex but the basics are simple: accept actions from the queue and then perform a series of system commands (like launching a KVM instance) to carry them out while updating state in the database.
### nova service-list

<table>
<thead>
<tr>
<th>Binary</th>
<th>Host</th>
<th>Zone</th>
<th>Status</th>
<th>State</th>
<th>Updated_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-cert</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-08-04T14:05:12.000000</td>
</tr>
<tr>
<td>nova-compute</td>
<td>mtinjopenstack01</td>
<td>nova</td>
<td>enabled</td>
<td>up</td>
<td>2013-08-04T14:05:19.000000</td>
</tr>
<tr>
<td>nova-conductor</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-08-04T14:05:17.000000</td>
</tr>
<tr>
<td>nova-consoleauth</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-08-04T14:05:21.000000</td>
</tr>
<tr>
<td>nova-network</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-08-04T14:05:17.000000</td>
</tr>
<tr>
<td>nova-scheduler</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-08-04T14:05:22.000000</td>
</tr>
</tbody>
</table>

### nova-manage service list

<table>
<thead>
<tr>
<th>Binary</th>
<th>Host</th>
<th>Zone</th>
<th>Status</th>
<th>State</th>
<th>Updated_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-conductor</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2013-08-04 14:06:48</td>
</tr>
<tr>
<td>nova-network</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2013-08-04 14:06:47</td>
</tr>
<tr>
<td>nova-scheduler</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2013-08-04 14:06:42</td>
</tr>
<tr>
<td>nova-compute</td>
<td>mtinjopenstack01</td>
<td>nova</td>
<td>enabled</td>
<td>:-)</td>
<td>2013-08-04 14:06:39</td>
</tr>
<tr>
<td>nova-cert</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2013-08-04 14:06:42</td>
</tr>
<tr>
<td>nova-consoleauth</td>
<td>mtinjopenstack01</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2013-08-04 14:06:41</td>
</tr>
</tbody>
</table>

### nova image-list

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>f3780eef-f881-410d-92d5-3309ad0eca6d</td>
<td>CirrOS 0.3.1</td>
<td>ACTIVE</td>
<td></td>
</tr>
</tbody>
</table>

### nova flavor-list

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Memory_MB</th>
<th>Disk</th>
<th>Ephemeral</th>
<th>Swap</th>
<th>VCPUs</th>
<th>RXTX_Factor</th>
<th>Is_Public</th>
<th>extra_specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m1.tiny</td>
<td>512</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1.0</td>
<td>False</td>
<td>True</td>
<td>{}</td>
</tr>
<tr>
<td>2</td>
<td>m1.small</td>
<td>2048</td>
<td>20</td>
<td>0</td>
<td>1</td>
<td>1.0</td>
<td>False</td>
<td>True</td>
<td>{}</td>
</tr>
<tr>
<td>3</td>
<td>m1.medium</td>
<td>4096</td>
<td>40</td>
<td>0</td>
<td>2</td>
<td>1.0</td>
<td>False</td>
<td>True</td>
<td>{}</td>
</tr>
<tr>
<td>4</td>
<td>m1.large</td>
<td>8192</td>
<td>80</td>
<td>0</td>
<td>4</td>
<td>1.0</td>
<td>False</td>
<td>True</td>
<td>{}</td>
</tr>
<tr>
<td>5</td>
<td>m1.xlarge</td>
<td>16384</td>
<td>160</td>
<td>0</td>
<td>8</td>
<td>1.0</td>
<td>False</td>
<td>True</td>
<td>{}</td>
</tr>
</tbody>
</table>

### nova network-list

<table>
<thead>
<tr>
<th>ID</th>
<th>Label</th>
<th>Cidr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1f992992-a67a-4d9a-88c4-e624e2847af</td>
<td>ext_net</td>
<td>None</td>
</tr>
</tbody>
</table>
Networking ("Quantum")

- **quantum-server** accepts API requests and then routes them to the appropriate quantum plugin for action.
- Quantum ships with plugins and agents for:
  - Cisco virtual and physical switches
  - Nicira NVP product
  - NEC OpenFlow products
  - Open vSwitch
  - Linux bridging
  - Ryu Network Operating System
  - Midokua
- The common agents are L3 (layer 3), DHCP (dynamic host IP addressing) and the specific plug-in agent.
```
root@mtinjopenstack01:/etc/init.d# quantum

(quantum) net-list
+----------------------------------|----------|-------------------+
| id                              | name     | subnets           |
+----------------------------------|----------|-------------------+
| 1f992992-a67a-4d9a-88c5-e624e2847a9 | ext_net  | a833d69a-71c5-43ff-b40c-3fdd33495ad9 135.25.138.0/24 |

(quantum) subnet-list
+----------------------------------|----------|-------------------|-------------------+
| id                              | name     | cidr              | allocation pools  |
+----------------------------------|----------|-------------------|-------------------+
| a833d69a-71c5-43ff-b40c-3fdd33495ad9 | 135.25.138.0/24 | {"start": "135.25.138.181", "end": "135.25.138.184"} |

(quantum) router-list
+----------------------------------|----------|-------------------|
| id                              | name     | external_gateway_info |
+----------------------------------|----------|-------------------|
| 6448e01-8cd4-4eb2-a764-66af2552afe | testrouter1 | {"network_id": "1f992992-a67a-4d9a-88c5-e624e2847a9"} |

(quantum) port-list
+----------------------------------|----------|-------------------|-------------------|
| id                              | name     | mac_address       | fixed_ips         |
+----------------------------------|----------|-------------------|-------------------|
| a833d69a-71c5-43ff-b40c-3fdd33495ad9 | fa:16:e:47:c9 | {"subnet_id": "a833d69a-71c5-43ff-b40c-3fdd33495ad9", "ip_address": "135.25.138.181"} |
```

Dashboard ("Horizon")

- Django application that users can access in their web browser
- Communicates with each OpenStack service through their API (and sometimes their admin API)
• Server/horizon, login dashboard

• Administrate resources via Graphic Interface
root@mtjinopenstack01:~# nova boot --flavor 1 --image f3780eef-f881-410d-92d5-3309ad0eca6d --key_name MyKey --security_group web --nic net-id=92aab103-2fb8-419d-bee3-6be9e81b52e0 Webserver_2

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>BUILD</td>
</tr>
<tr>
<td>updated</td>
<td>2013-08-06T10:02:29Z</td>
</tr>
<tr>
<td>OS-EXT-STS:task_state</td>
<td>scheduling</td>
</tr>
<tr>
<td>OS-EXT-SRV-ATTR:host</td>
<td>None</td>
</tr>
<tr>
<td>key_name</td>
<td>MyKey</td>
</tr>
<tr>
<td>image</td>
<td>CirrOS 0.3.1</td>
</tr>
<tr>
<td>hostId</td>
<td></td>
</tr>
<tr>
<td>OS-EXT-STS:vm_state</td>
<td>building</td>
</tr>
<tr>
<td>OS-EXT-SRV-ATTR:instance_name</td>
<td>instance-00000038</td>
</tr>
<tr>
<td>OS-EXT-SRV-ATTR:hypervisor_hostname</td>
<td>None</td>
</tr>
<tr>
<td>flavor</td>
<td>ml1.tiny</td>
</tr>
<tr>
<td>id</td>
<td>9e1f72da-5790-44b1-9ec1-a4b40adfb2ea</td>
</tr>
<tr>
<td>security_groups</td>
<td>[{'u'name': u'web'}]</td>
</tr>
<tr>
<td>user_id</td>
<td>bcc575ed4cc4d207bf657e323c7668a5</td>
</tr>
<tr>
<td>name</td>
<td>Webserver_2</td>
</tr>
<tr>
<td>adminPass</td>
<td>Ee73KSiZJC22</td>
</tr>
<tr>
<td>tenant_id</td>
<td>dd4da34de91d46d4b2ca379b9f0d1d02</td>
</tr>
<tr>
<td>created</td>
<td>2013-08-06T10:02:29Z</td>
</tr>
<tr>
<td>OS-DCF:diskConfig</td>
<td>MANUAL</td>
</tr>
<tr>
<td>metadata</td>
<td>{}</td>
</tr>
<tr>
<td>accessIPv4</td>
<td></td>
</tr>
<tr>
<td>accessIPv6</td>
<td></td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
</tr>
<tr>
<td>OS-EXT-STS:power_state</td>
<td>0</td>
</tr>
<tr>
<td>OS-EXT-AZ:availability_zone</td>
<td>nova</td>
</tr>
<tr>
<td>config_drive</td>
<td></td>
</tr>
</tbody>
</table>

root@mtjinopenstack01:~# nova list

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>f4ecdc37-3b4a-4546-94e8-b15146487747</td>
<td>Client</td>
<td>ACTIVE</td>
<td>net_1=135.25.138.184</td>
</tr>
<tr>
<td>66e008ea-4d26-4551-be3e-45ee6ebc5b38</td>
<td>Webserver_1</td>
<td>ACTIVE</td>
<td>net_1=135.25.138.182</td>
</tr>
<tr>
<td>9e1f72da-5790-44b1-9ec1-a4b40adfb2ea</td>
<td>Webserver_2</td>
<td>ACTIVE</td>
<td>net_1=135.25.138.183</td>
</tr>
</tbody>
</table>
LBaaS feature in Neutron

• Load-balancer-as-a-Service (LBaaS) is a feature added to Quantum during grizzly, and an advanced service in Neutron.

• LBaaS allows the ability to provision on demand loadbalancers pragmatically

• Allows one to create several instances all running the same application and then distribute the load across them in order to scale out an application and provide high availability.
Example: Loadbalancing with Neutron

• Build a multi-tier application with OpenStack, HTTP port 80 to two web servers, database server over port 3306 accessed by web servers;

• use Open vSwitch Neutron plugin;

• Clients send requests, which to be loadbalanced between the two web servers;
$ while true; do echo -e 'HTTP/1.0 200 OK\r\n\r\n\nweb_server1' | sudo nc -l -p 80; done
GET / HTTP/1.1
Host: 135.25.138.182
User-Agent: Wget
Connection: close

$ while true; do echo -e 'HTTP/1.0 200 OK\r\n\r\n\nweb_server2' | sudo nc -l -p 80; done
GET / HTTP/1.1
Host: 135.25.138.183
User-Agent: Wget
Connection: close

Connecting to 135.25.138.182 (135.25.138.182:80)
web_server1
- 100% |****************************************************************| 12 0:00:00 ETA
Connecting to 135.25.138.183 (135.25.138.183:80)
web_server2
- 100% |****************************************************************| 12 0:00:00 ETA
Connecting to 135.25.138.182 (135.25.138.182:80)
web_server1
- 100% |****************************************************************| 12 0:00:00 ETA
Connecting to 135.25.138.183 (135.25.138.183:80)
web_server2
- 100% |****************************************************************| 12 0:00:00 ETA
Connecting to 135.25.138.182 (135.25.138.182:80)
web_server1
- 100% |****************************************************************| 12 0:00:00 ETA
Connecting to 135.25.138.183 (135.25.138.183:80)
web_server2
- 100% |****************************************************************| 12 0:00:00 ETA
Loadbalancing with OVS Neutron plugin

```bash
$ wget -O - http://135.25.138.185
Connecting to 135.25.138.185 (135.25.138.185:80)
web_server1
- 100% |****************| 12 0:00:00 ETA
$ wget -O - http://135.25.138.185
Connecting to 135.25.138.185 (135.25.138.185:80)
web_server2
- 100% |****************| 12 0:00:00 ETA
$ wget -O - http://135.25.138.185
Connecting to 135.25.138.185 (135.25.138.185:80)
web_server1
- 100% |****************| 12 0:00:00 ETA
$ wget -O - http://135.25.138.185
Connecting to 135.25.138.185 (135.25.138.185:80)
web_server2
- 100% |****************| 12 0:00:00 ETA
$ wget -O - http://135.25.138.185
Connecting to 135.25.138.185 (135.25.138.185:80)
web_server1
- 100% |****************| 12 0:00:00 ETA
$ wget -O - http://135.25.138.185
Connecting to 135.25.138.185 (135.25.138.185:80)
web_server2
- 100% |****************| 12 0:00:00 ETA
```
The end