Cloud middleware

Part2: Let’s pick one cloud IaaS middleware: OpenStack

Sergio Maffioletti
S3IT: Service and Support for Science IT,
University of Zurich
http://www.s3it.uzh.ch/
Part2: content

1. Understand the OpenStack ecosystem
2. Understand OpenStack architecture
What is OpenStack?

OpenStack Foundation: “Open source software for building private and public clouds”

- Open source project (Apache 2.0).
- Up to 1’128 contributors, including commercial companies.
- Biggest contributor is Rackspace.
- Releases every 6 months. (check releases).
- Currently the only real alternative to proprietary clouds.
OpenStack capabilities

VMs on demand
Volumes
Network
Object storage
Multi-tenancy
OpenStack capabilities

VMs on demand
  - provisioning.
  - snapshotting.

Volumes

Network

Object storage

Multi-tenancy
OpenStack capabilities

VMs on demand

Volumes
- block storage devices.
- allow persistent storage.
- R/W to single instance.
- provisioned via API.

Network

Object storage

Multi-tenancy
OpenStack capabilities

VMs on demand

Volumes

Networks
- define network connectivity and IP addressing.
- L3 forwarding and NAT to load balancing
- virtual network, subnet, and port abstractions to describe networking resources.

Object storage

Multi-tenancy
**OpenStack capabilities**

VMs on demand

Volumes

Network

**Object storage**
- redundant, scalable.
- global API access.
- no POSIX interface (only objects).

Multi-tenancy
OpenStack capabilities

VMs on demand

Volumes

Network

Object storage

Multi-tenancy
  – quotas for different tenants.
  – user can be associated with multiple tenants.
OpenStack Architecture

- Everything is in Python (plus auxiliary shell scripts)
- Build around independent components
- Highly distributed architecture
- Intrinsic HA for OpenStack services (MySQL and RabbitMQ have to be properly configured)
- SQL database used to store persistent data
- RabbitMQ used for inter-service communication and notification
- Web API services (mostly Django)
**keystone** provides the authentication service
OpenStack logical view

nova provides computational services
OpenStack logical view

neutron provides network services
**glance** provides image store
OpenStack logical view

**cinder** provides block persistent store
OpenStack logical view

**swift** provides object persistent store
OpenStack logical view

horizon provides web user interface
OpenStack software overview
Typical deployment scenario

Control Cluster
- Horizon
- Keystone
- Quantum
- Glance
- Cloud Ctrl. (nova)
- Cinder

Compute Cluster
- compute node
- compute node
- compute node
- compute node
- compute node

Storage Cluster (Swift)
- storage node
- storage node
- storage node
- storage node
- storage node
- storage node

Heavy CPU and RAM

Heavy Disk and I/O
keystone - authentication service

- Stores authentication information (users, passwords, tokens, projects, roles).
- Holds a catalog of available services and their endpoints.
- Can use different backends (SQL database, LDAP).
- It’s the entry point for OpenStack API.
keystone Data Model

- **User**: has account credentials, is associated with one or more tenants.
- **Tenant**: unit of ownership in OpenStack, contains one or more users.
- **Role**: a first-class piece of metadata associated with many user-tenant pairs.
- **Token**: identifying credential associated with a user or user and tenant.
**nova service**

Service responsible of managing virtual instances.

**nova-api** Web API frontend, accepts requests, validates them and contact other services if needed. Supports OpenStack Compute API, Amazon’s EC2 API and a special Admin API.

**nova-scheduler** it takes a virtual machine instance request from the **message-queue** an determines where it should run.
Message Queue is a unified way for collaboration between components.

Use multiple queues within single MQ instance.

Usually RabbitMQ.
**nova-scheduler**
	nova-scheduler determines which compute host the request should run.

**nova-scheduler** : provision VM to particular host.

- provision VMs of the particular tenant to isolated hosts.
- provision all VMs on different hosts.
- provision VMs to "higher density" hosts.
nova-scheduler filters

- Filters statically configured.
- Multiple filters can be specified (Affinity, anti-affinity, ...).
nova - compute service

Running on each compute node, interacts with the hypervisor and actually controls the VM.
Image courtesy of Mirantis
neutron - network service

Service responsible of creating and managing networks. It is supposed to replace \texttt{nova-network}.

Still not widely used, but very feature rich.

- L2 and L3 networks.
- Allow creation of multiple networks and subnets.
- Plugin architecture.
- Supports Load Balancer As a Service.
- Integrates with network devices (Cisco, NEC).
Network configuration flow

1. Allocate MAC addresses.
2. Allocate IPs (for each network).
3. Associate IP and MAC with VM (DB).
   - update DHCP config
   - initialize gateway
cinder - block storage

- Creates and export Volumes via iSCSI to the compute node.
- Volumes are mounted transparently from the virtual machines.
- Supports multiple storage backends (NFS, LVM, Ceph, GlusterFS but also SAN/NAS devices from IBM, NetApp etc. .).

composed of multiple services:
cinder-api Web API frontend.
cinder-volume Manages block storage devices. You can have many of these.
cinder-scheduler Decides which cinder-volume has to provide the Volume for an instance.
glance - image service

Service responsible of storing image information and, optionally, image files.

- Holds information about available images.
- Optionally allow to download and upload images.
- Images can be stored on different backends (RDB, S3, swift, filesystem).
- Multiple image formats supported (raw, vhd, vdi, qcow2, ami, ...).
swift - object storage

Object storage distributed service.

- Redundant, scalable object storage on commodity hardware.
- Not a POSIX filesystem.
- Scales horizontally simply by adding new servers.
- Supports AWS S3 APIs.
Life of a virtual machine

1. Authentication is performed either by the web interface horizon or nova command line tool.
2. nova-api is contacted and a new request is created.
3. nova-scheduler find an appropriate host.
4. nova-compute reads the request and start an instance.
5. neutron/nova-network configure the network.
6. nova-compute contacts cinder to provision the Volume.
7. nova-compute fetches VM image from glance.
8. nova-compute starts the virtual machine.
9. horizon/nova poll nova-api until the VM is ready.
Initial state

UI: Horizon or CLI

Nova
- Nova API
- Scheduler
- Nova DB

Compute
- Hypervisor
- VM
- Network
- nova-compute

Cinder
- Cinder API
- Scheduler
- Cinder Volume
- Cinder DB

Block Storage
- Storage

Glance
- Glance API
- Glance registry
- Keystone DB

Swift
- Proxy server
- Object Store

Keystone
- Keystone server
- Keystone DB

Neutron
- Neutron server
- Neutron DB
Step 1: Authentication and Authorization

UI: Horizon or CLI

client initiates HTTP POST request to Keystone

Keystone
- Keystone server
- Keystone DB

Nova
- Nova API
- Scheduler
- Nova DB

Compute
- Nova-compute
- Hypervisor
- VM
- Network

Neutron
- Neutron server
- Neutron DB

Cinder
- Cinder API
- Scheduler
- Cinder DB
- Cinder Volume

Block Storage
- Storage

Glance
- Glance API
- Glance registry
- Keystone DB

Swift
- Proxy server
- Object Store
1. Client initiates HTTP POST request to keystone
2. keystone parses HTTP requests and verifies
   – Authentication
   – Access Control
   – Authorization
3. a **token** is saved in the **keystone-db** and returned to the client to be used with later interactions with OpenStack services for this request.
Step 1: Authentication and Authorization

UI: Horizon or CLI

auth token is saved in the database and returned to the client

Keystone
- Keystone server
- Keystone DB

Compute
- Hypervisor
- VM
- Network

Nova
- Nova API
- Scheduler
- Nova DB

Cinder
- Cinder API
- Scheduler
- Cinder Volume
- Cinder DB

Block Storage
- Storage

Glance
- Glance API
- Glance registry
- Keystone DB

Swift
- Proxy server
- Object Store
Step 2: Send API request to nova-API
nova-API validate request process

1. checks via **keystone** the validity of the token
2. validates parameters and create a new request in the **nova-db**
3. calls the **nova-scheduler** via **message-queue**
Step 2.1: checks via keystone the validity of the token
Step 2.2: Validates parameters and create a new request in the nova-db

**nova-db** stores current state of all objects in the compute cluster.
Step 2.3: calls the nova-scheduler via message-queue

Request has been validated but not actions have been taken yet.
Step 3: nova-scheduler request process

1. reads the request from **message-queue**
2. fetches information about the whole cluster from **nova-db**
3. finds an appropriate host via filtering and weighting
4. calls the chosen **nova-compute** host via **message-queue**
Step 3.1: nova-scheduler reads message from message-queue
Step 3.1: nova-scheduler fetched information from nova-db

UI: Horizon or CLI

scheduler fetches information about the whole cluster from the Database

Nova
- Nova API
- Scheduler
- Nova DB

Compute
- nova-compute
- Hypervisor
- VM
- Network

Cinder
- Cinder API
- Scheduler
- Cinder Volume
- Cinder DB

Block Storage
- Storage

Glance
- Glance API
- Glance registry
- Keystone DB

Swift
- Proxy server
- Object Store

Keystone
- Keystone server
- Keystone DB

Neutron
- Neutron server
- Neutron DB

S3IT lectures Cloud middleware Varenna, 25.07.2014
Step 3.2: nova-scheduler calls the chosen nova-compute host via message-queue.
Step 4: nova-compute reads the request and starts the instance

1. reads the request from message-queue
2. reads VM information from nova-db
Step 4.1: nova-compute reads message from message-queue
Step 4.2: nova-compute reads VM information from nova-db

From *Grizzly* release *nova-conductor* has been introduced to address remote-DB access.
Step 5: neutron configures Network

1. **nova-compute** queries **neutron** for Network service
2. **neutron** Associate IP and MAC with VM (DB)
   - setup network - L2
   - setup network - L3
Step 5: nova-compute queries neutron for Network service
Step 6: nova-compute contacts cinder to provision Volumes

1. **nova-compute** gets Volume data from **cinder**
2. **nova-compute** initiate iSCSI connector
3. **nova-compute** instructs Hypervisor to mount the iSCSI Volume as a new block device.
**Step 6.1: nova-compute contacts cinder to provision the Volume**

**cinder** provides Volume information (optional step for persistent data).
Step 6.2: nova-compute requests Volume

Nova-compute set-up iSCSI initiator and instruct hypervisor to mount iSCSI volume as new device.
Step 7: nova-compute fetches VM image

1. **nova-compute** requests image from **glance** via Image ID
2. **glance** returns an URI if image ID is valid
3. **nova-compute** downloads image using URI.
7.1: nova-compute requests image from glance

Nova

- Nova API
- Scheduler
- Nova DB

Cinder

- Cinder API
- Scheduler
- Cinder Volume
- Cinder DB

Compute

- nova-compute
- Hypervisor
- VM
- Network

Block Storage

- Storage

Glance

- Glance API
- Glance registry
- Keystone DB

UI: Horizon or CLI

Keystone

- Keystone server
- Keystone DB

Neutron

- Neutron server
- Neutron DB

Swift

- Proxy server
- Object Store

nova-compute retrieves base OS VM image from Glance.
Step 7.2: nova-compute downloads image from swift

nova-compute downloads image using URI provided by Glance. From Swift or from other Glance backends.
Step 8: nova-compute starts VM

1. **nova-compute** fetches information about VM from **nova-db**

2. creates a command to Hypervisor
   - in case of KVM/libvirt this is a single VM XML config file.

3. delegates to Hypervisor the activation of VM

4. Periodically polls VM status from Hypervisor and updates **nova-db**
Step 8.1: VM can be started

Nova computes passes information about VM to Hypervisor. Hypervisor creates an instance.
Step 8.2: nova-compute polls VM status and updates nova-db

Nova-DB is periodically updated with new VM information. iSCSI communication directly with Volume.

UI: Horizon or CLI

Nova
- Nova API
- Scheduler
- Nova DB

Compute
- nova-compute
- Hypervisor
- VM
- Network

Cinder
- Cinder API
- Scheduler
- Cinder Volume
- Cinder DB

Block Storage
- Storage

Glance
- Glance API
- Glance registry
- Keystone DB

Swift
- Proxy server
- Object Store

Keystone
- Keystone server
- Keystone DB

Neutron
- Neutron server
- Neutron DB
Step 9: horizon/nova CLI poll nova-api for updated VM status

Horizon queries nova-api for VM status and power state. Information are taken from Nova-DB.

UI: Horizon or CLI

- Nova
  - Nova API
  - Scheduler
  - Nova DB
  - queue

- Compute
  - nova-compute
  - Hypervisor
  - VM
  - Network

- Cinder
  - Cinder API
  - Scheduler
  - Cinder Volume
  - Cinder DB
  - queue

- Block Storage
  - Storage

- Glance
  - Glance API
  - Glance registry
  - Keystone DB

- Swift
  - Proxy server
  - Object Store

- Keystone
  - Keystone server
  - Keystone DB

- Neutron
  - Neutron server
  - Neutron DB
Recap

- User logs into **horizon** and initiates a VM create request,
- **keystone** authorizes,
- **nova** initiates provisioning and saves state to **nova-db**,
- **nova-scheduler** finds appropriate host,
- **neutron** configures networking,
- **cinder** provides block device,
- image URI is looked up through **glance**,
- image is retrieved via **swift**,
- VM is rendered.