OpenStack Fundamentals

OpenStack – open source software for building private and public clouds. Owned by the community.

What is cloud? Distrubuted and redundant @ a hardware and software level.

Rackspace is a distro of openstack – class was plain openstack

Example:

Linux – source code is free but hard to setup

distributions are setups that most people use

cherished server -> cherished VM -> disposable VMs

Regions / Cells <- disaster recovery

Object Storage (Swift) – Not a file system, not like a file share.

resources:

trystack.org

stackalytics.com

restructured text – text editing provided by Python project

cloud-keyring – verify repo for cloud-archive – makes sure you get current version of openstack

Identity – Keystone

Authentication – who you are? The older version of Keystone gave you a token (UUID)

Authorization – Not handled by Keystone, this is handled by other projects / services

IceHouse – Public Key (PKI token) – embed and signed by Keystone

problem – really large deployments breaks APIs – needs to be compressed to PKIz

UUID method moved back in Juno

token temporary – set length in config

Tenant level – all things openstack creates are owned

Keystone service – It represents another openstack service

Service Catelog – service and endpoint

Endpoint – URL where the service is listening and responding to API calls

Role – keystone doesn’t define it but knows about them, set roles in keystone

Keystone v3 – in Icehouse but not in CLI, \* only accessible through direct API calls

New Features – 1. Domains, Groups, Federation

tenant /project – getting changed to ‘project’

Keystone region – allows you to group endpoints together based on location for example

* could allow you to manage 2 different sets of services

Ubuntu – Conconicle

LTS – version of ubuntu allows for newer version of openstack to be applied through backend repo

Icehouse version = 1:2014:1.1

Juno = 1:2014.2

admin token -> /etc/keystone/keystone.conf

service token can be changed

qemu-img – create disk image

virt-install – build VM, install OS

Edit images

virt-sparsify – create larger images when you don’t have enough space

Can control image access by tenant if not marked as public

Glance services – 2 services for each, parent and child

Floating IP – to allow a VM to be replaced by another behind the scenes

Day 2

Setup Overview

Install Message broker (RabbitMQ)

Install MySQL(Database)

Install proper version of Openstack

Install packages

Stop service

edit config files

create and Init DB

Keystone config for each component

* service user in services tenant
* admin role
* service entry & endpoints

restart service

Logs – located in /var/log

Security groups – All VMs (Iptables)

Firewall as a service – (Tenant can modify setup) (Iptables) – 1 per tenant

Neutron – manages network, subnet, port

security standpoint – compute node is @ higher risk -> DB access through conductor

conductor can be load balanced – direct access to DB

Nova config file – api-paste.ini (python?)

Nova copies glance image into its own repo when building VM the 1st time. That is why 1st build takes longer.

flavors stored in Nova DB

TCP port 22 is SSH

ICMP = ping

QCOW – Copy on Write

QEMU – KVM worked with QEMU to adopt their tools for KVM.

Cinder – Block storage used for C drive of images initially

Can’t create volume and then attach from nova command. Need to create volume 1st – probably orchestrate with Heat.

Chef –

1. Create volume
2. Create Chef instance
3. Attach volume
4. Install Chef on that voume

Day 3 – Neutron part 2

Troubleshoot network issues from VM

1. Can I ping the router / verify IP address is correct
2. IP A – check network
3. IP route
4. check security group

How is Cinder setup in our environment?

Swift –

Don’t need to setup in Raid and also not recommended

Ring – namespace represented by a ring

Storing by name won’t work – get’s uneven – more start with ‘t’ than ‘x’ for example

Stored by Hash – more random and better distribution

part of hash is index (partition)

ring contains has values to point to location

ring exists in proxy and object server

we define zones

Try to make each zone similar in space

assign weight based on drive size, numbers are relative to each other (for example 1, 2, 4 same as 10, 20 , 40 or 100, 200, 400)

Best practice for # of zones – replicas +2 (default is 3 copies so 5 zones)

Zone’s check with each other

if they can’t contact a zone file is copied to handoff server

Delete doesn’t actually delete but creates a tombstone file, 2nd process runs to deletee the files

Note – can be a delay in container DB update, this would cause file to still show up even though deleted. This will resolve itself.

ring is a file that gets replicated like any other file

3 rings (object, container, account)

Container – stored in SQLight DBs keeps track of objects in container

2 DBs account and container

success for a swift object had to be greater than 50 %. If replica =2, both have to work, if 3, 2 have to work

storage locations = disks

3 partitions should have 5 zones

3 primary, 2 handoff

Handoff location is assigned by partition. The zone changes so all capacity is used:

example:

Zone 1 Zone2 Zone3 Zone4

A1 A2 A3 AH

B2 BH B1 B3

CH C1 C2 C3

Dispersion report – once you create the objects you don’t need to do it again unless it’s way under 1%.

drive audit –

log file pattern – change the log, regex controlled and configurable

Swift Use Cases - Disaster Recovery, Archiving

Horizon – uses service catelog to determine what it shows in the browser

Neutron

shared network – other tenants can add ports to network

image – protected – no one can delete it

**Controller Setup**

rabbitmq

mysql

keystone

glance

nova (all pieces except compute)

neutron-server

neutron-plugin-ml2

cinder-api

-scheduler

-dashboard

**Neutron Setup**

openvswitch

neutron plugin-open vs

neutron – dhcp agent

l3 – agent

metadata-agent

**Compute setup**

openvswitch – switch

neutron-plugin

nova-compute

cinder-volume

The queue is how nodes talk to each other

Controller – containers all of the api pieces and the queue

other nodes communicate via the controller

Controller Add on’s for swift

install HAProxy works as a load balancer to send to swift-proxy on other nodes

install swift-client