IaaS Cloud (OpenStack) overview

Antonio Messina `<antonio.messina@s3it.uzh.ch>`

S^{3}IT - Services and Support for Science IT, University of Zurich

13 January 2015

A few words about me

Provide support to scientists or their computational needs

A few words about me

Provide support to scientists or their computational needs

- deploy and operate the computational infrastructure(s)
- create services on top of it
- port applications
- fix scientific code
- consult scientists (possibly *before* they write the pipeline)

A few words about me

Provide support to scientists or their computational needs

- deploy and operate the computational infrastructure(s)
- create services on top of it
- port applications
- fix scientific code
- consult scientists (possibly *before* they write the pipeline)

so I have to be a:

- System Administrator/Engineer/Whatever
- Software Developer/Architect/foo_bar()
- PR Expert

What is OpenStack?

OpenStack is a cloud operating systemcloud operating system that controls large pools oflarge pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

What is OpenStack?

> *OpenStack is a* ***cloud operating system*** *that controls large pools oflarge pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.*

It's a complex piece of software plus a buzzword

What is OpenStack?

> *OpenStack is a cloud operating systemcloud operating system that controls **large pools of** compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.*

It is scalable (thousands of nodes)

What is OpenStack?

> *OpenStack is a cloud operating systemcloud operating system that controls large pools oflarge pools of **compute, storage, and networking resources** throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.*

It is not just about CPU

What is OpenStack?

> *OpenStack is a cloud operating systemcloud operating system that controls large pools oflarge pools of compute, storage, and networking resources throughout a datacenter, all managed through a **dashboard** that gives administrators control while **empowering their users** to provision resources through a web interface.*

User-centric

Above all

- creation is done via Web GUI, CLI or **network APIs**

  ⇒ it can be *scripted*

- Actual provisioning of the VMs can be delegated **to the user**.
- No need to fill up a form or open an issue to create a VM.
- Time to provision a VM can be very short (depending on the type of setup!!)
  - ~ 10 seconds to create
  - $\sim 20 - 50$ seconds to login (depending on the image)

OpenStack vs. traditional HPC/HTC batch system

- quota management vs. fairsharing
- software installation (user vs. admin)
- scalability vs. performance
- shared vs. exclusive resource
- continuous growth vs. fixed size

OpenStack vs. rest of the world

What's different from KVM/VMWare/Virtualbox?

- specs of the VMs are chosen form a list of predefined **flavors** that define:
  - Nr. of CPUs
  - amount of RAM
  - size disk
- complex network setup are possible
  - although not always needed
- OS already installed (but adapted automatically to the current instance)
- multiple options for storage (volumes and object storage)
- VMs are spawned on possibly thousands of nodes

Cattle vs. pets

- pets are given names like `kenny.example.org`
- you care about them
- they are unique, you check on them every day
- when they get ill, you nurse them back to health

Cattle vs. pets

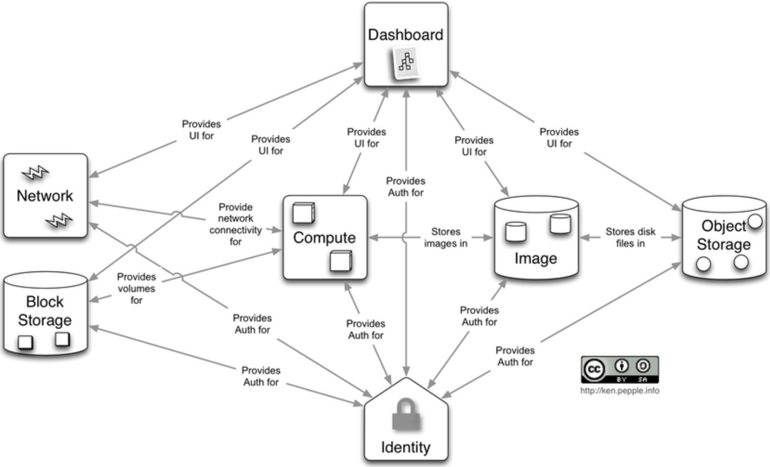
- pets are given names like `kenny.example.org`
- you care about them
- they are unique, you check on them every day
- when they get ill, you nurse them back to health

- cattle are given names like `vm-001.example.org`
- they are all the same
- when they get ill, you shoot them and get another one

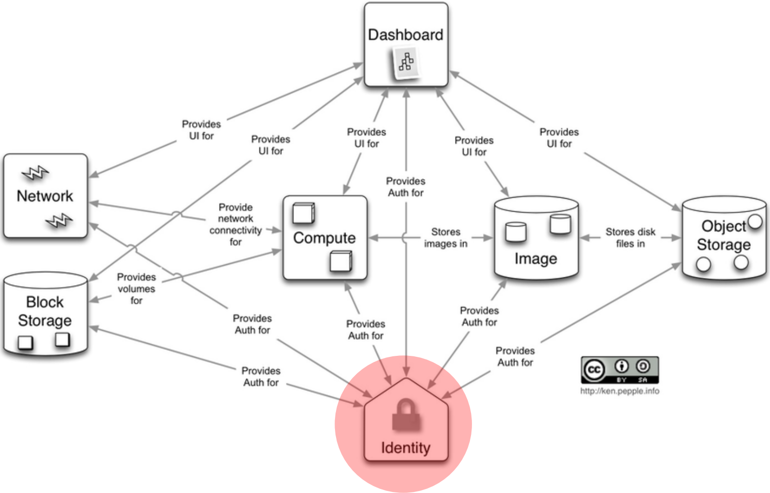
OpenStack Architecture

- written in Python (plus auxiliary shell scripts)
- built around **independent components**
- **highly distributed** architecture
  - designed for very big installations
- **intrinsic HA** of *most* OpenStack services (MySQL and RabbitMQ have to be properly configured)
- ***SQL** database used to store persistent data
- **RabbitMQ** used for RPC and notification
- **RESTful APIs** for all the services

OpenStack logical view

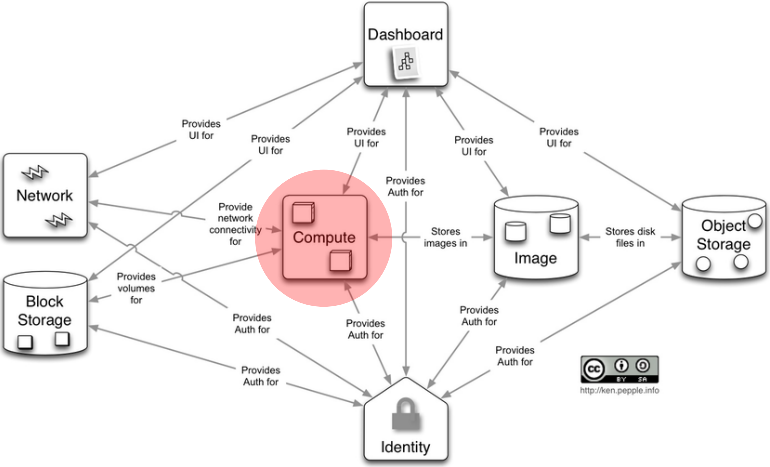


OpenStack logical view



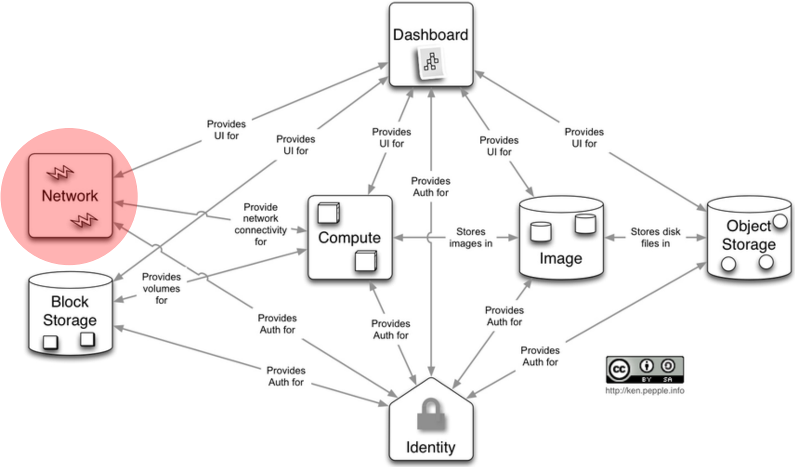
Keystone provides the authentication service

OpenStack logical view



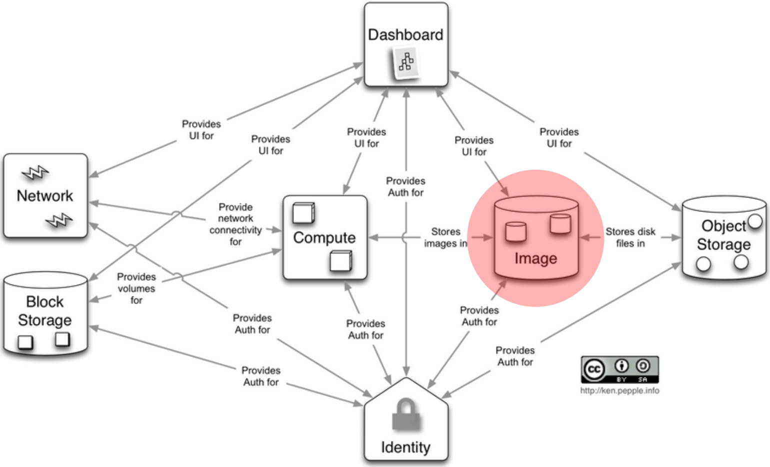
Nova provides computational services

OpenStack logical view



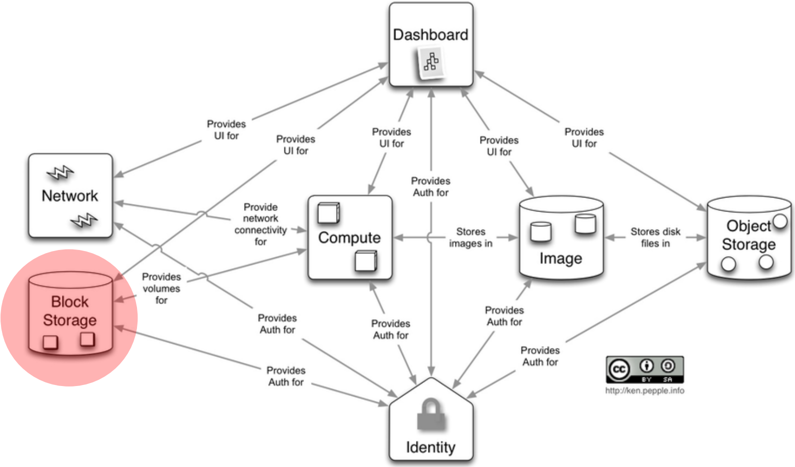
Neutron (`nova-network`) provides network services

OpenStack logical view



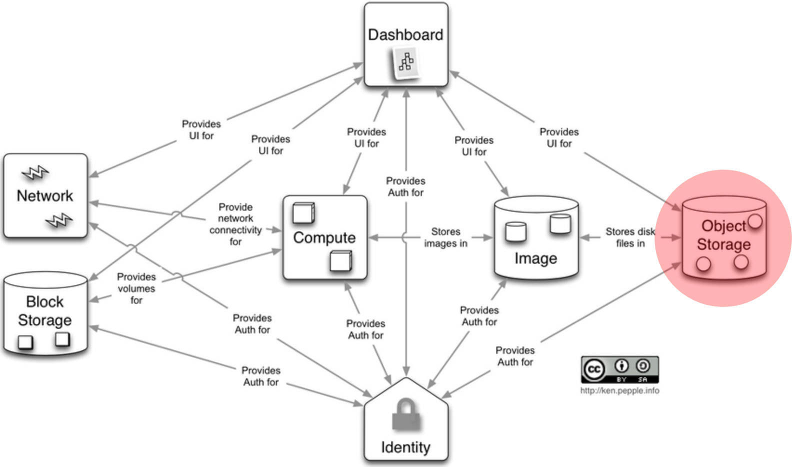
Glance provides image store

OpenStack logical view



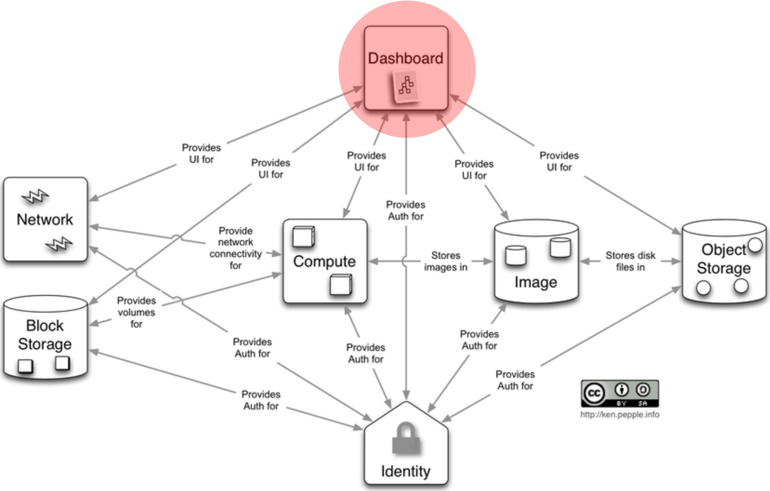
Cinder provides block persistent store

OpenStack logical view



Swift provides object persistent store

OpenStack logical view



Horizon provides web user interface

keystone - authentication service

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

nova - compute service

Service responsible of managing virtual instances.

nova-api Web API frontend, accepts requests, validates them and contact other services if needed.

nova-scheduler decides where to start an instance

nova-compute running on each compute node, interacts with the hypervisor and actually starts the vm.

nova-network old, simple, (working) implementation of network service. Does not support Software Defined Networks.

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)

neutron - network service

Service responsible of creating and managing networks. It is supposed to replace **nova-network**.

Still not widely used, but very feature rich.

- L2 and L3 networks.
- Allow creation of multiple networks and subnets.
- Plugin architecture.
- Supports advanced network services (Load Balancer, Firwall, DNS as a service)
- Integrates with network devices (Cisco, Brocade. . .)

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.

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.

It's not the only choice: **Ceph**, **GlusterFS** and others can be used instead.

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. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

Life of a virtual machine

1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
   1.1 keystone is contacted and authentication is performed
   1.2 a **token** is saved in the database and returned to the client to be used with later interactions with OpenStack services for this request.
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. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

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:
   2.1 checks via **keystone** the validity of the token
   2.2 checks the authorization of the user
   2.3 validates parameters and create a new request in the database
   2.4 calls the scheduler via queue
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

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
   - 3.1 reads the request
   - 3.2 find an appropriate host via filtering and weighting
   - 3.3 calls the chosen **nova-compute** host via queue
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

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:
   4.1 generates a proper configuration for the hypervisor
   4.2 get image URI via image id
   4.3 download the image
   4.4 request to allocate network via queue
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

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. **nova-compute** contacts **cinder** to provision the volumeif requested) **nova-compute** contacts **cinder** to provision the volume
    - 5.1 gets connection parameters from cinder
    - 5.2 uses iscsi to make the volume available on the local machine
    - 5.3 asks the hypervisor to provision the local volume as virtual volume of the specified virtual machine
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

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. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
   - 6.1 allocates a valid private ip
   - 6.2 if requested, it allocates a floating ip
   - 6.3 configures the host as needed (dnsmasq, iptables, Open VSwitch. . .)
   - 6.4 updates the request status
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

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. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

Life of a virtual machine

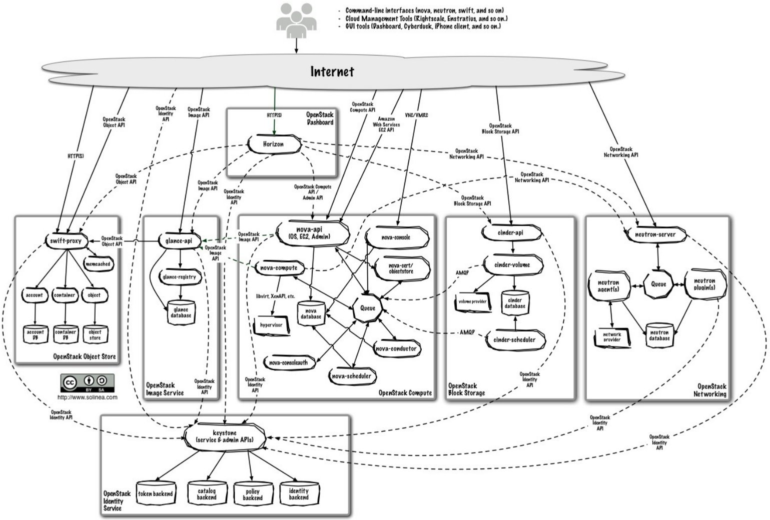
1. Authentication is performed either by the web interface **horizon** or **nova** command line tool:
2. **nova-api** is contacted and a new request is created:
3. **nova-scheduler** find an appropriate host
4. **nova-compute** reads the request and start an instance:
5. (if requested) **nova-compute** contacts **cinder** to provision the volume
6. **neutron/nova-network** configure the network
7. **nova-compute** starts the virtual machine
8. **horizon/nova** poll **nova-api** until the VM is ready.

Notes on installation

- Please, please, please, use a **deployment and configuration manager**. There are many: Puppet, Chef, CFEngine, Ansible, SaltStack... Just pick the one you like most.
- Do not underestimate the **complexity** of the system.
- Plan in advance, and **plan for failures**.
- RTFM: the OpenStack website is now plenty of documentation[1]
  - Install Guide (for Ubuntu 12.04/14.04)
  - Architecture Design Guide
  - Cloud Administrator Guide
  - Training guide
  - Operations Guide
  - High Availability Guide
  - Security Guide

[1] it wasn't like this 2 years ago...

OpenStack software overview



OpenStack software overview

Other OpenStack services

Projects **integrated** in Juno:

- Ceilometer (Metering)
- Heat (Orchestration)
- Trove (Database as a service)
- Sahara (Data Processing - Hadoop)

Projects in **incubation**:

- Ironic (Bare metal provisioning)
- Zaqar (aka *Marconi*) (Messaging service)
- Barbican (Secure storage of secrets)
- Designate (DNSaaS)
- TripleO (OpenStack-on-OpenStack)

Hands-on session!

Let’s have fun!

Storage solutions

volume storage Permanent or volatile storage, block level access, exclusive, only accessible from OpenStack.

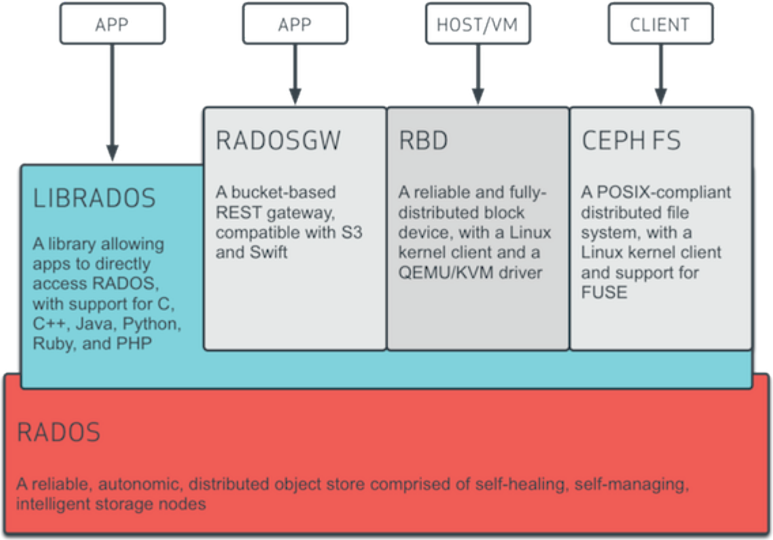
object storage RESTful web API, accessible from anywere, permanent storage, supports complex authorization (including anonymous access), usually supports geo-replication.

filesystem storage usually provided on top of OpenStack, using VMs providing NFS, Lustre, GlusterFS or CephFS. Manila is an OpenStack project to provide a filesystem-as-a-service.

Ceph overview

- Distributed object, block and filesystem storage without SPoF
- Highly reliable and highly scalable
- RadosGW (object storage) supports geo-replication
- Integrated with OpenStack (Glance, Cinder, Nova, Swift)
- Data can be replicated or stored using an erasure code (similar to RAID5/6)

Ceph Software Stack



LIBRADOS

A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP

RADOSGW

A bucket-based REST gateway, compatible with S3 and Swift

RBD

A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver

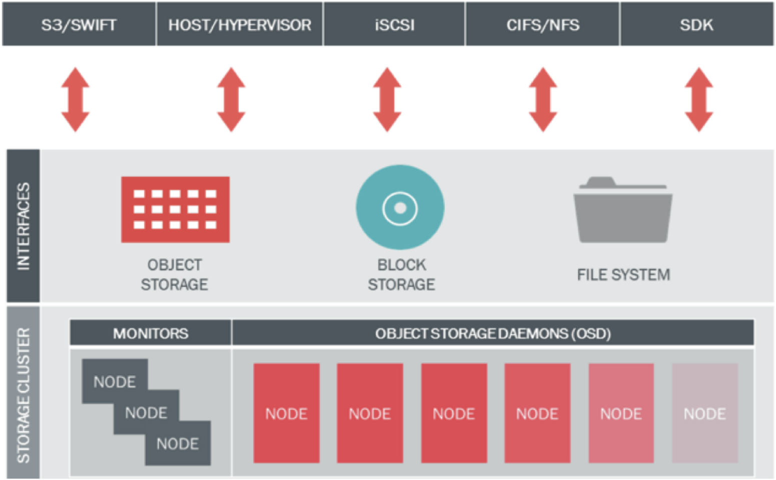
CEPH FS

A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE

RADOS

A reliable, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

Ceph Architecture



Ceph minimal glossary

osd nodes store *objects* of fixed size. They are responsible for replicating the data and rebalance the cluster if needed.

mon nodes store information on the topology of the cluster and the rules to store data.

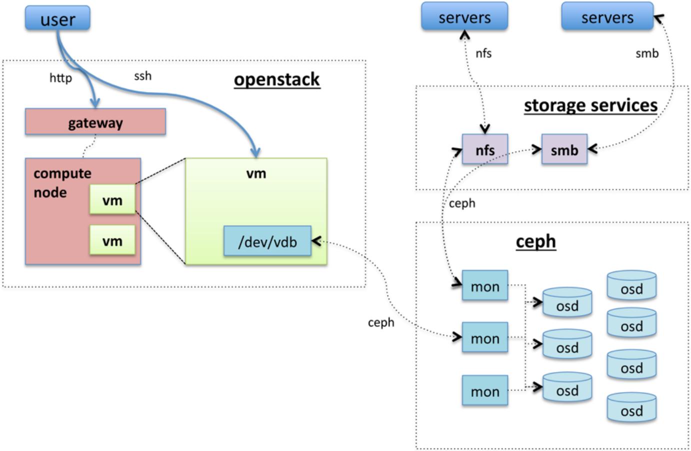
mds node stores information about the metadata of a CephFS filesystem.

cluster map is stored on the **mon** nodes, and it's used by the clients to know on which OSDs the data should be written (using the CRUSH algorithm).

radosgw a ceph client that provides S3/Swift APIs to ceph objects.

rbd client library to provide block-level access to ceph objects.

Ceph for ephemeral and cinder


user
http
ssh
openstack
gateway
compute node
vm
vm
vm
/dev/vdb
servers
servers
nfs
smb
storage services
nfs
smb
ceph
ceph
ceph
mon
mon
mon
osd
osd
osd
osd
osd
osd
osd