Distributed file systems

LINAGORA

Distributed file systems

Introduction

Clustering with Ceph

Types of Ceph client

Preparation of Ceph cluster

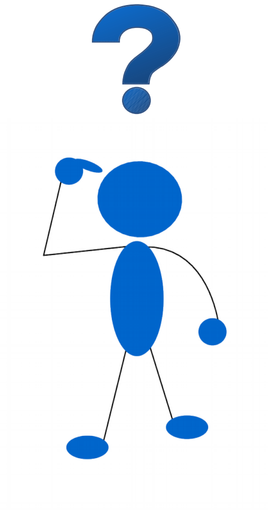
Set Ceph nodes

Set a RDB client

Introduction

What are filesystems ?

- Organization of mass storage
  - To store data
  - To retrieve data
  - To share data between multiple programs
- Two points of view
  - User : a filename including pathname and its content
  - Operating system : metadata, pointers, blocks, chunks,...



Types of filesystems

- Local filesystems
  - On one computer
  - Direct access to a block device
  - Managed by one OS
  - XFS, BRTFS, EXT*, ...

- Shared filesystems
  - Multiple OS access to one exported filesystem
  - The filesystem is connected to one computer
  - This computer manages its local block device
  - NFS, CIFS

- But problems
  - I/O bottle necks
  - Network latency
  - Host is a SPOF

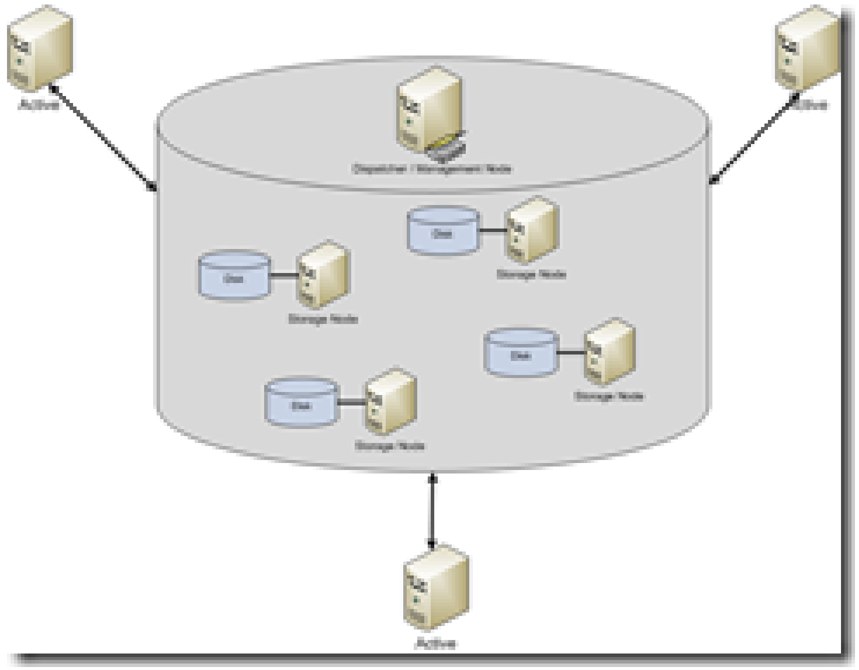
Solutions ?

- HA cluster
  - One system active
  - one system standby
  - One cache

- Cluster filesystems
  - One block device
  - Block level access
  - I/O cache consistency management : cache synchronization
  - Multiple kernel handle I/O at any time
  - SAN with OCFS, RedHat GFS, ...

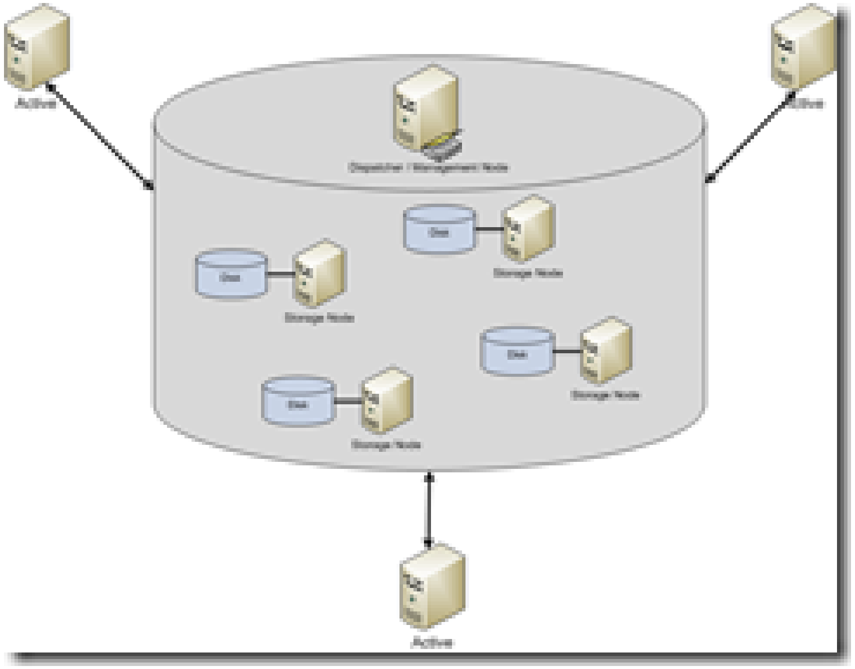
A solution : Distributed Filesystems

- A file system spread over several nodes
- Access by multiple clients
  - Large and scalable capacities
  - High I/O throughput
  - Redundancy
- Transparency
  - Access
  - Location
  - Concurrency
  - Failure
- Examples
  - GlusterFS, Gfarm, Ceph,...



How it works

- Multiple nodes
  - They run their own OS
  - They access to one block device
- One or more master nodes
  - Dispatch requests
  - Manage filesystem status
  - Manage synchronization
- Clients
  - Mount locally logical filesystem
  - Served by different nodes



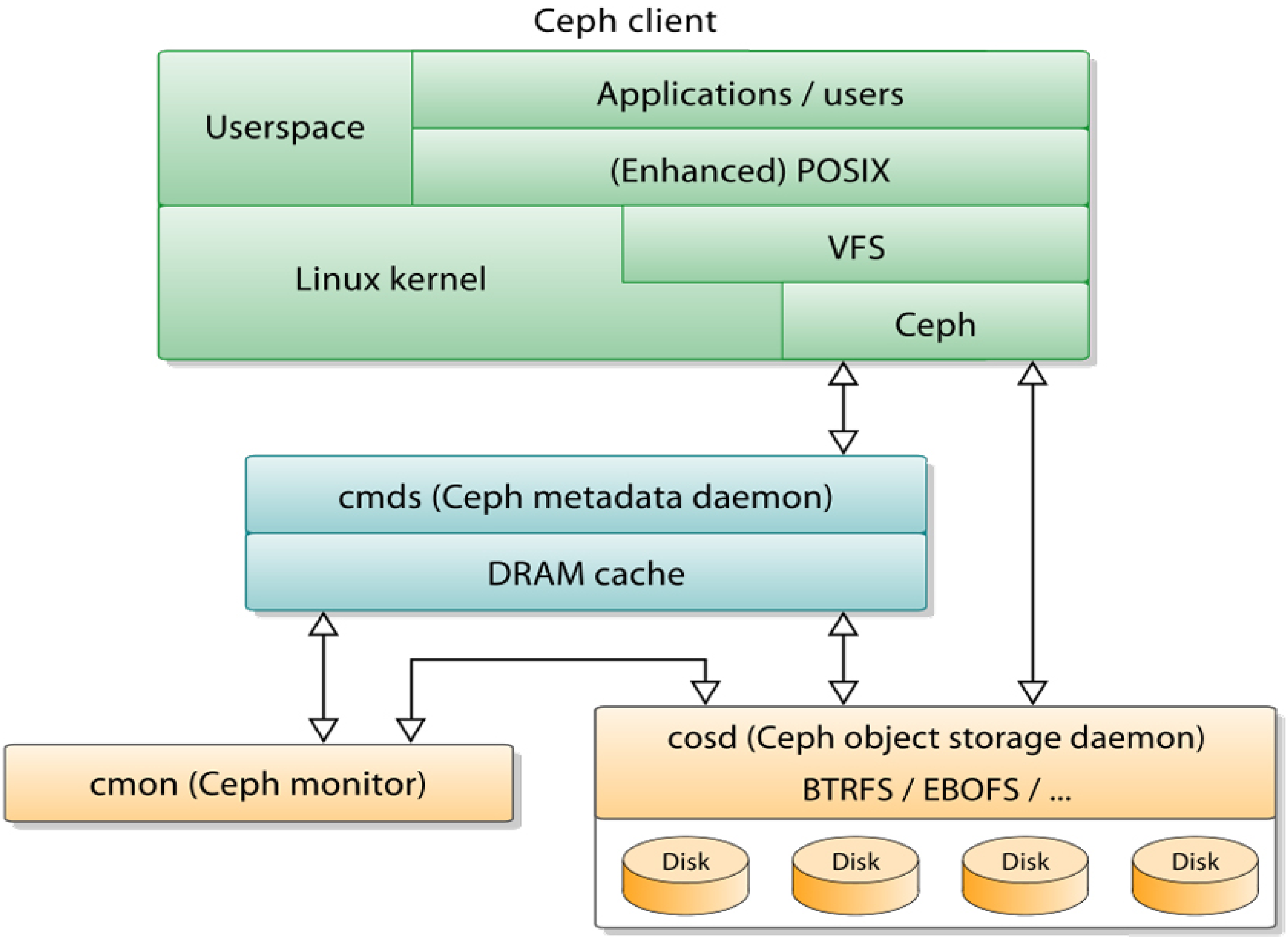
Ceph

What is CEPH ?

- Distributed storage system
  - Software-designed
  - Stores **objects, blocks, files**
  - Self-healing
  - Scalable
  - Designed for cloud infrastructure
  - All components are redunded
    - No SPOF
    - multiactive
- Open Source
  - Sage Weil, doctoral dissertation in 2006 (UCSC)
  - 2012 : Inktank Storage company
  - 2014 : Redhat aquires Inktank Storage
- april 2016 : version 10.2.0

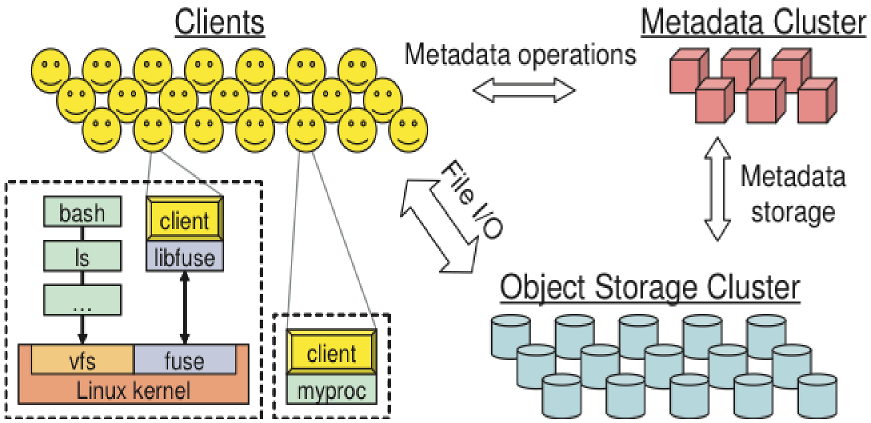
Organisation

- Several daemons to deliver service



Components : nodes

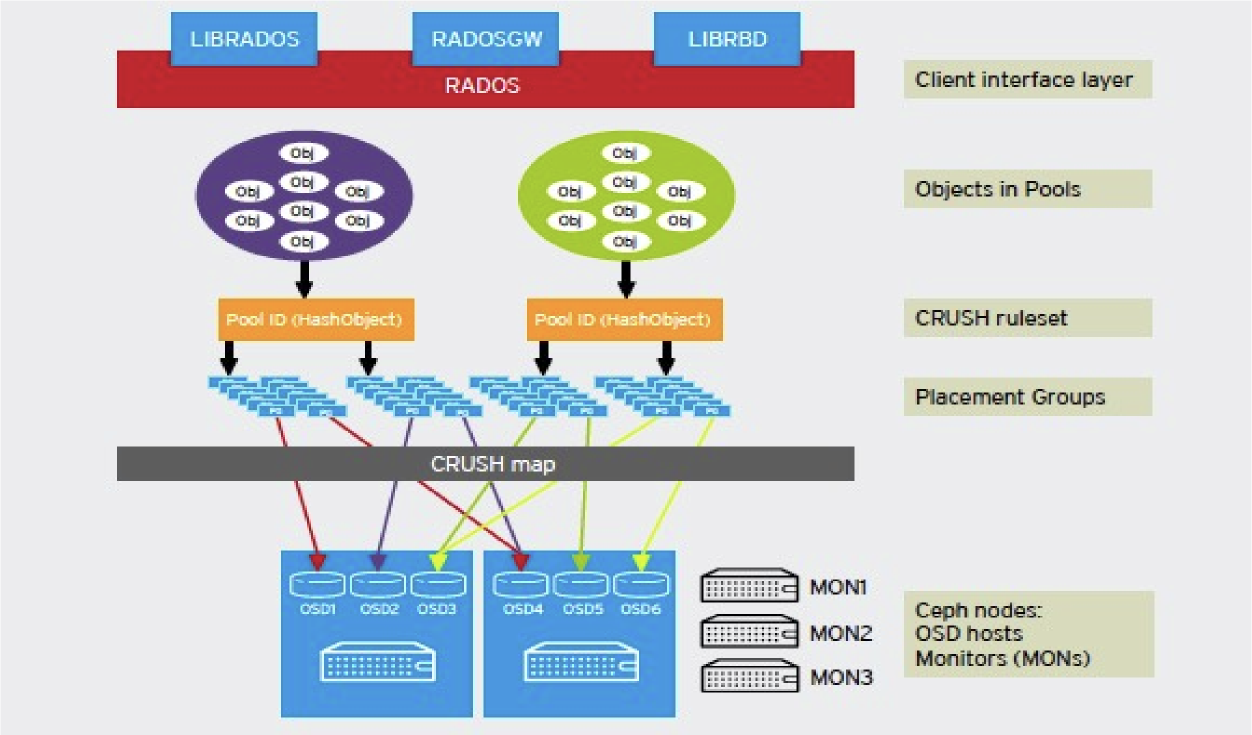
- An administration server : **admin**
  - Recommended for an easier deployment
  - For administrative task on the cluster
- A node running the metadata server : **mds**
  - Stores description of the user data
  - Load management
- At least one node running the monitor : **mon**
  - Clients get a copy of the 5 cluster maps
    - Monitor map (id, address, port,...)
    - OSD map (id, data containers, status, ...)
    - PG – Placement group - map (status, ...)
    - CRUSH map (list of storage devices, rules, ...)
    - MDS map (list of mds, ...)
  - Clusterized monitors



- At least one node running the Object Storage Device : **osd**
  - Store objects on disks
  - Reports to the **mon**

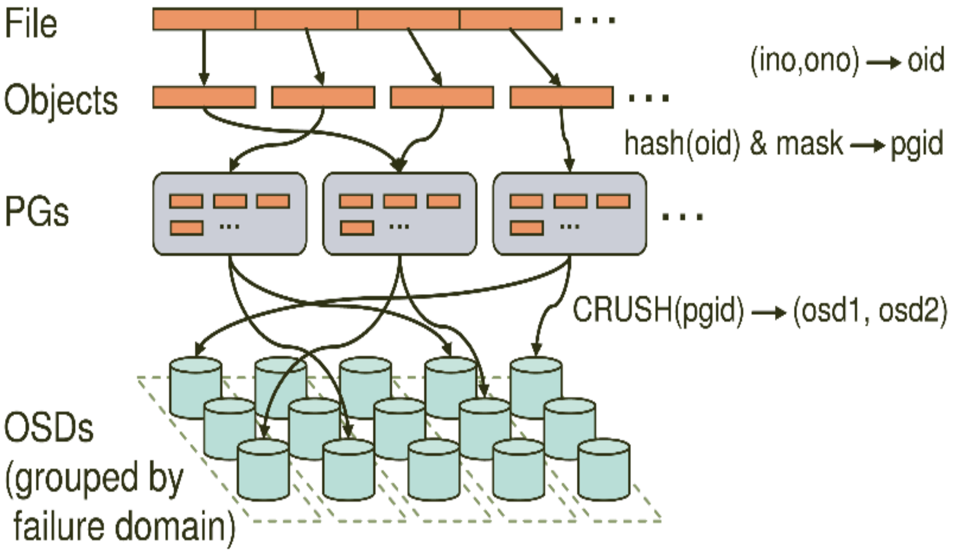
Architecture : another view

- Distribution and Modularity
  - Pools : ownership/access to objets,replication rules, number of PG
  - PG : logical container used to spread objects on OSDs (balancing)
  - CRUSH ruleset : clients compute data location and acces directly to data



Protocol

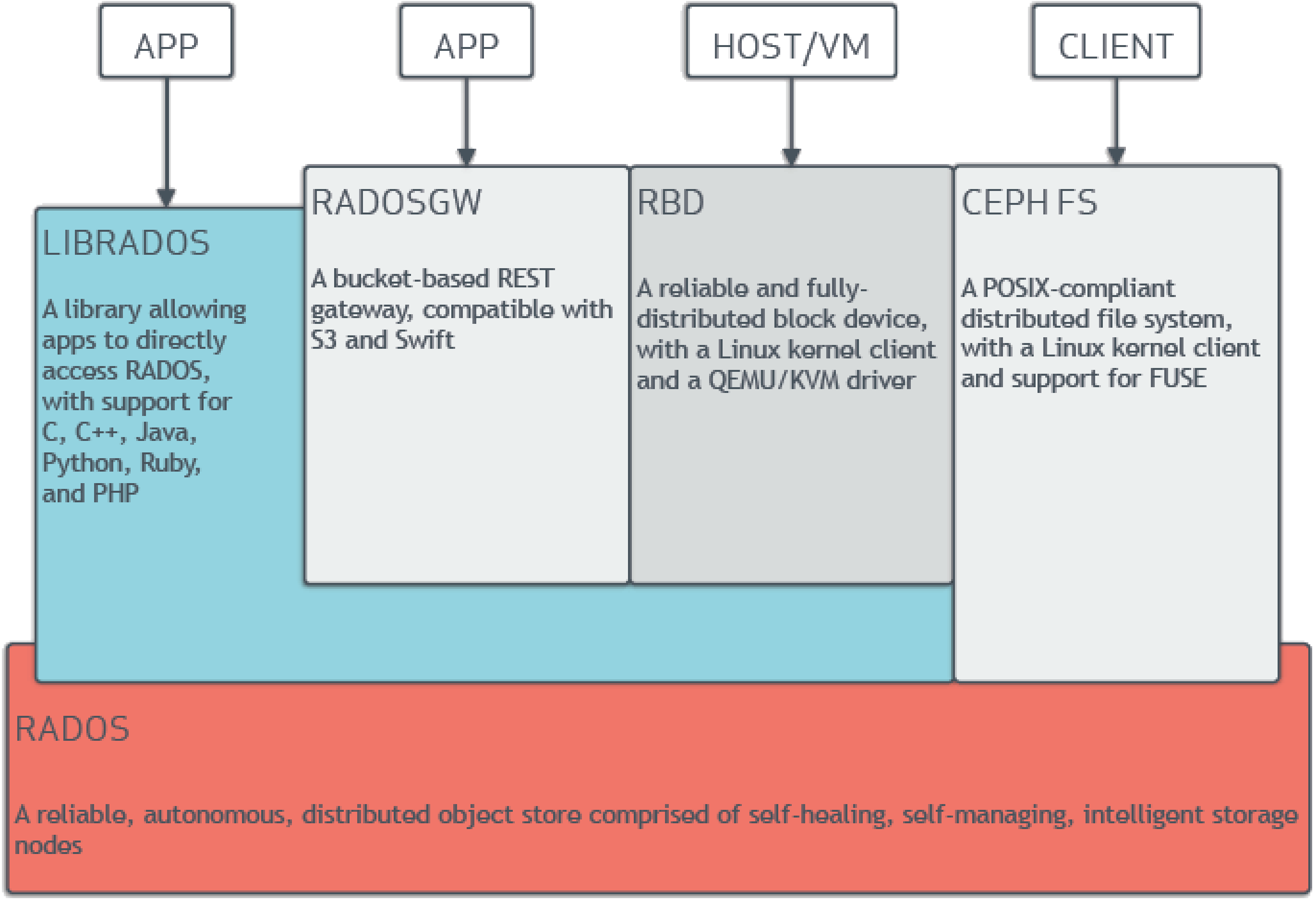
- Example with a write access
  - 1. client gets the updated cluster map from a MON server
  - 2. data are converted in objects (with object ID and pool ID)
  - 3. CRUSH determines PG and primary OSD where to store objects
  - 4. client contacts primary OSD node to store objects
  - 5. with CRUSH, primary OSD node searches secondary Pgs and secondary OSD nodes
  - 6. primary OSD replicates objects on secondary OSD nodes



Types of client

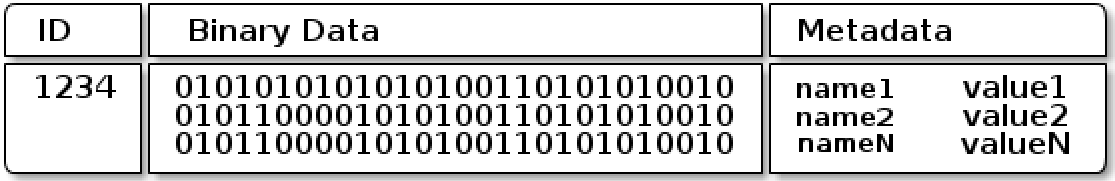
Clients

- With libraries, OS or client software, client nodes access to
  - Objects
  - Blocks
  - Files


APP
APP
HOST/VM
CLIENT
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, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

Data storage

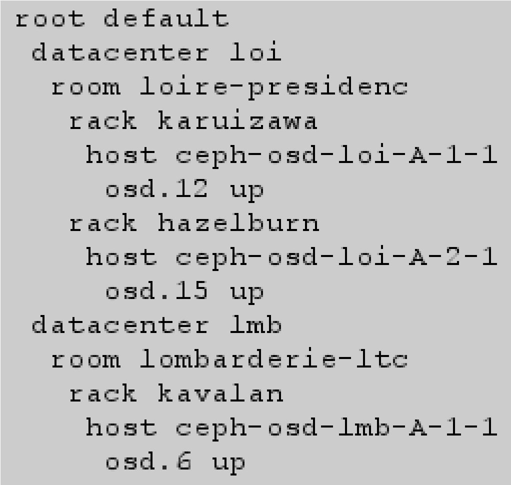
- OSD nodes
  - store objects
  - On a flat namespace
  - No hierarchy
- An object has
  - An ID
  - Binary data
  - Metadata (name/value pairs)
  - The client puts the semantic it wants
    - Blocks
    - Objects
    - Files



ID	Binary Data	Metadata
1234	0101010101010100110101010010 0101100001010100110101010010 0101100001010100110101010010	name1 value1 name2 value2 nameN valueN

Data location

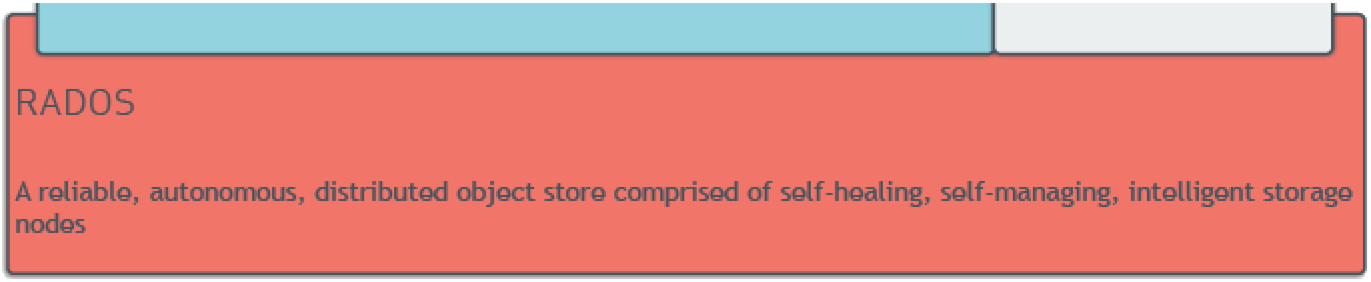
- Clients and OSD nodes compute data location
  - Hash algorithm
  - CRUSH : Controlled Replication Under Scalable Hashing
  - CRUSH used by clients to store objects
  - CRUSH used by OSD nodes to store objects and their replicas
  - Maps are updated



```
root default
 datacenter loi
  room loire-presidenc
   rack karuizawa
    host ceph-osd-loi-A-1-1
     osd.12 up
   rack hazelburn
    host ceph-osd-loi-A-2-1
     osd.15 up
 datacenter lmb
  room lombarderie-ltc
   rack kavalan
    host ceph-osd-lmb-A-1-1
     osd.6 up
```

RADOS

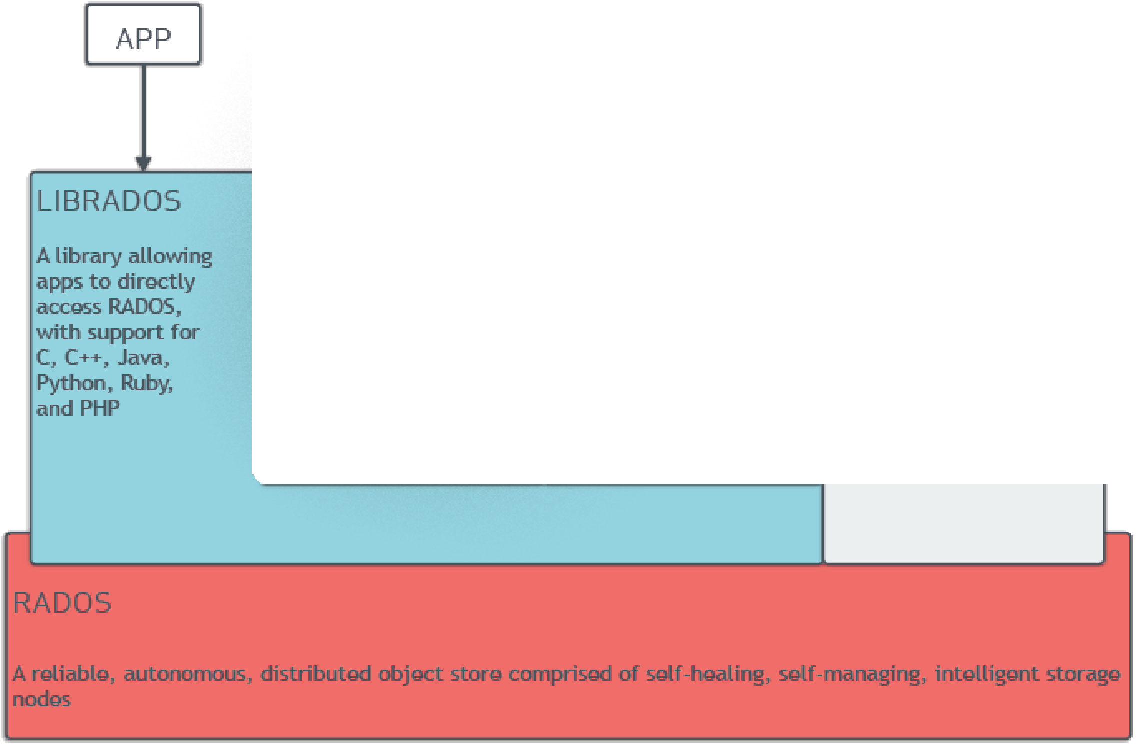
- RADOS
  - Lowest layer of Ceph storage clusters
  - Reliable Autonomic Distributed Object Store



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

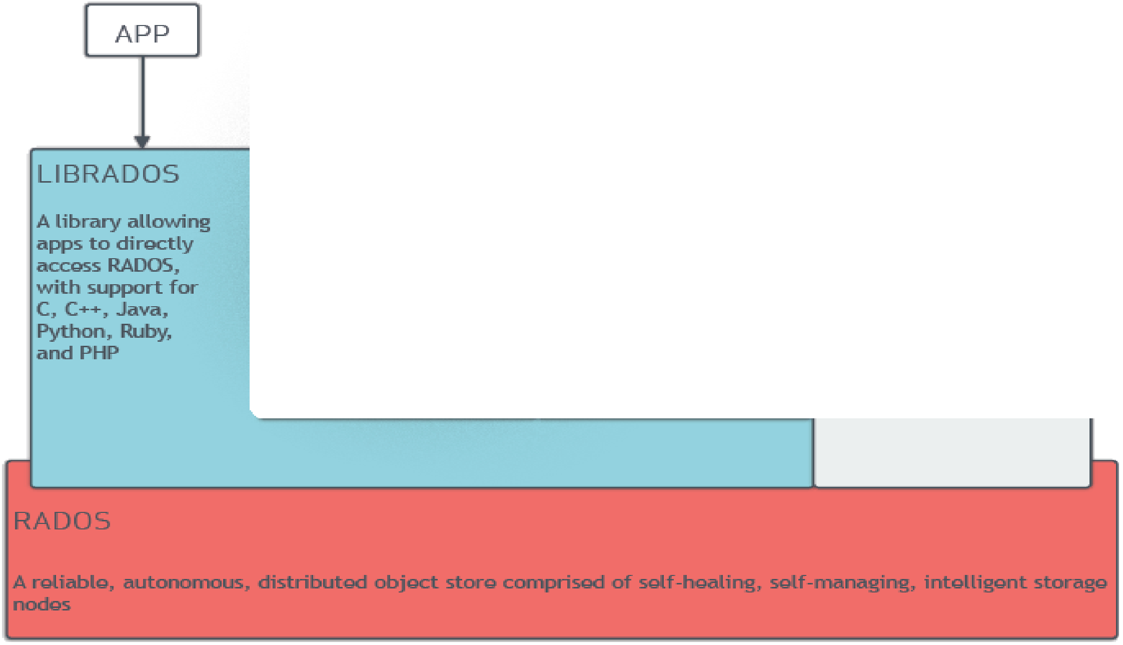
librados

- Used to develop Ceph Clients
  - For a parallel access object storage
  - Asynchronous communication protocol
  - Actions
    - R/W
    - Create/remove
    - Append/truncate
    - Manage XATTR's
    - Manage Name/Value p
    - Data stripping
    - ...
  - Library **librados**


APP
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

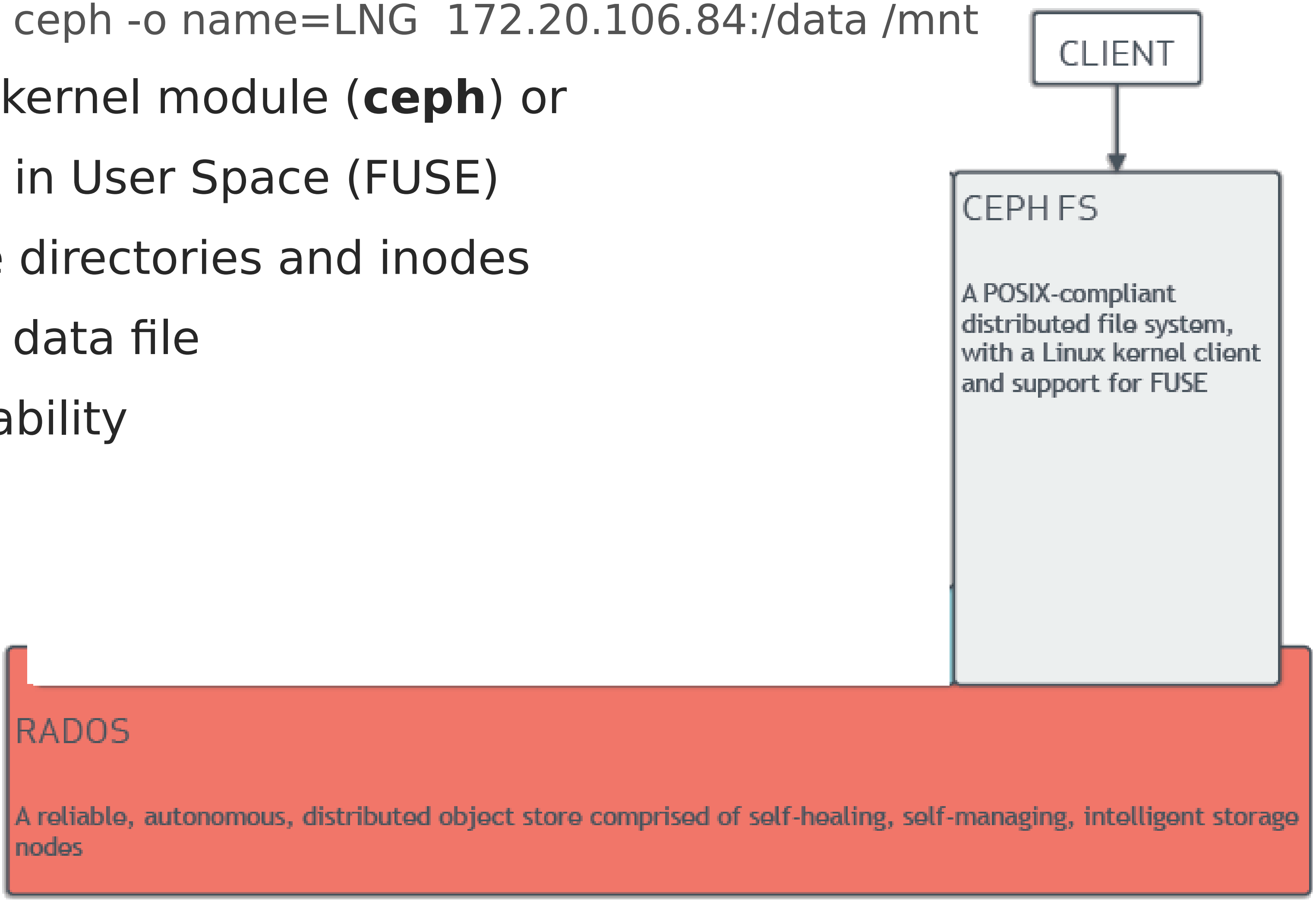
RBD : RADOS Block Device

- Block device service
  - Clients access
    - with Xen, QEMU or libvirt (**qemu-img create -f rbd rbd:image 2G**)
    - Kernel Ceph block device (**modprobe rbd**)
    - Command **rbd**
  - Stores blocks over multiple objects on Ceph
  - Objects are mapped on PG (placement group)
  - Choice of FS format (BTRFS is recommended)


APP
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

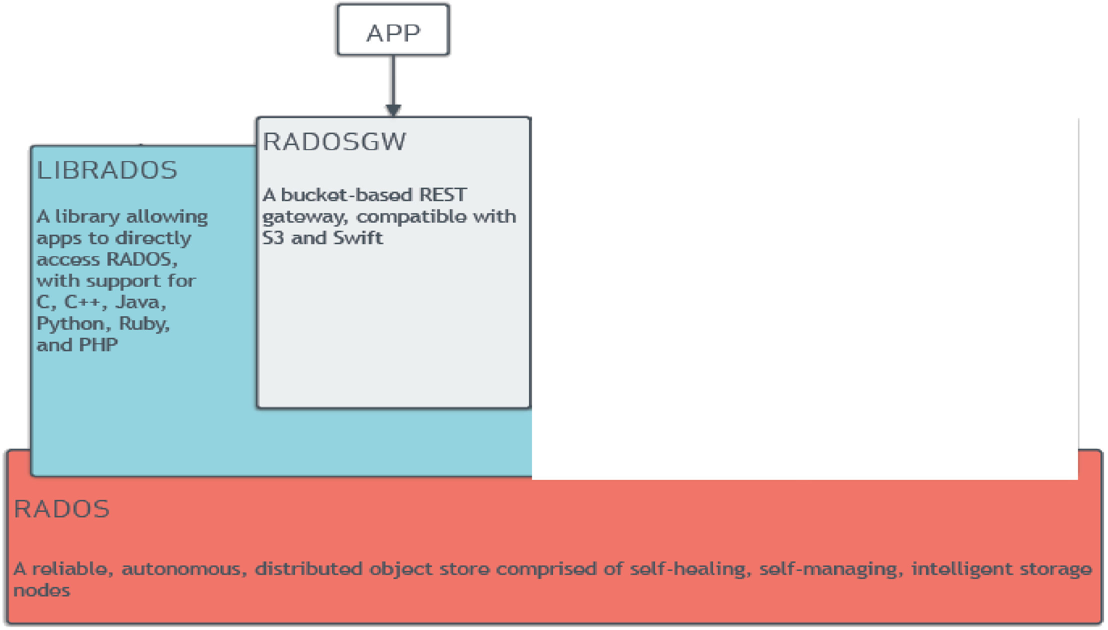
CephFS : Distributed File System

- CephFS
  - Clients access with **mount** command
    - mount -t ceph -o name=LNG 172.20.106.84:/data /mnt
  - Client use kernel module (**ceph**) or
  - Filesystem in User Space (FUSE)
  - **MDS** store directories and inodes
  - **OSD** store data file
  - High-availability
  - Scalability


CLIENT
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

RADOS Gateway

- RESTful interface
  - Access to objects only
  - Two standard APIs used by apps based on clusters
    - Amazon-S3
    - OpenStack-Swift
  - Based on Ceph Object Gateway daemon : **radosgw**


APP
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
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

Preparation

Needs

- For a test cluster : 5 nodes
  - cluster-admin
  - cluster-mon
  - cluster-mds
  - cluster-osd1
  - cluster-osd2
- For a production cluster : 10 nodes
  - cluster-admin
  - cluster-mon1, cluster-mon1, cluster-mon2
  - cluster-mds1, cluster-mds2
  - cluster-osd1, cluster-osd2, cluster-osd3, cluster-osd4,
- Minimun hardware
  - 1 CPU , 1 GB RAM per node
- User
  - A linux user « **ceph** » on all nodes

Helps and packages

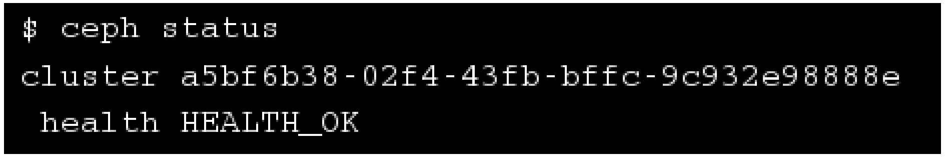
- SSH: distribute the **ceph** user SSH-key on all nodes
- Sudo : give root access to ceph user
- Name resolution
  - Give a significant hostname to nodes
  - Set DNS records for all
- Prepare control node : on ceph-admin
  - Install ceph-deploy package
- Set role to monitor node(s)
  - Ceph-deploy new cluster-mon
- Install Ceph on nodes
  - ceph-deploy install cluster-admin cluster-mon cluster-mds cluster-osd1 cluster-osd2

Roles definition

- Define the monitor node
  - ceph-deploy mon create cluster-mon
- Get authentication keys from the monitor
  - ceph-deploy gatherkeys cluster-mon
- Select disks on data nodes (ie : /dev/sdb)
  - ceph-deploy disk zap cluster-osd1:sdb (same for osd2)
- Prepare data nodes
  - ceph-deploy osd prepare cluster-osd1:sdb (same for osd2)
- Enable data nodes
  - ceph-deploy osd activate cluster-osd1:sdb (same for osd2)
- Define MDS nodes
  - ceph-deploy mds create cluster-mds

It works ?

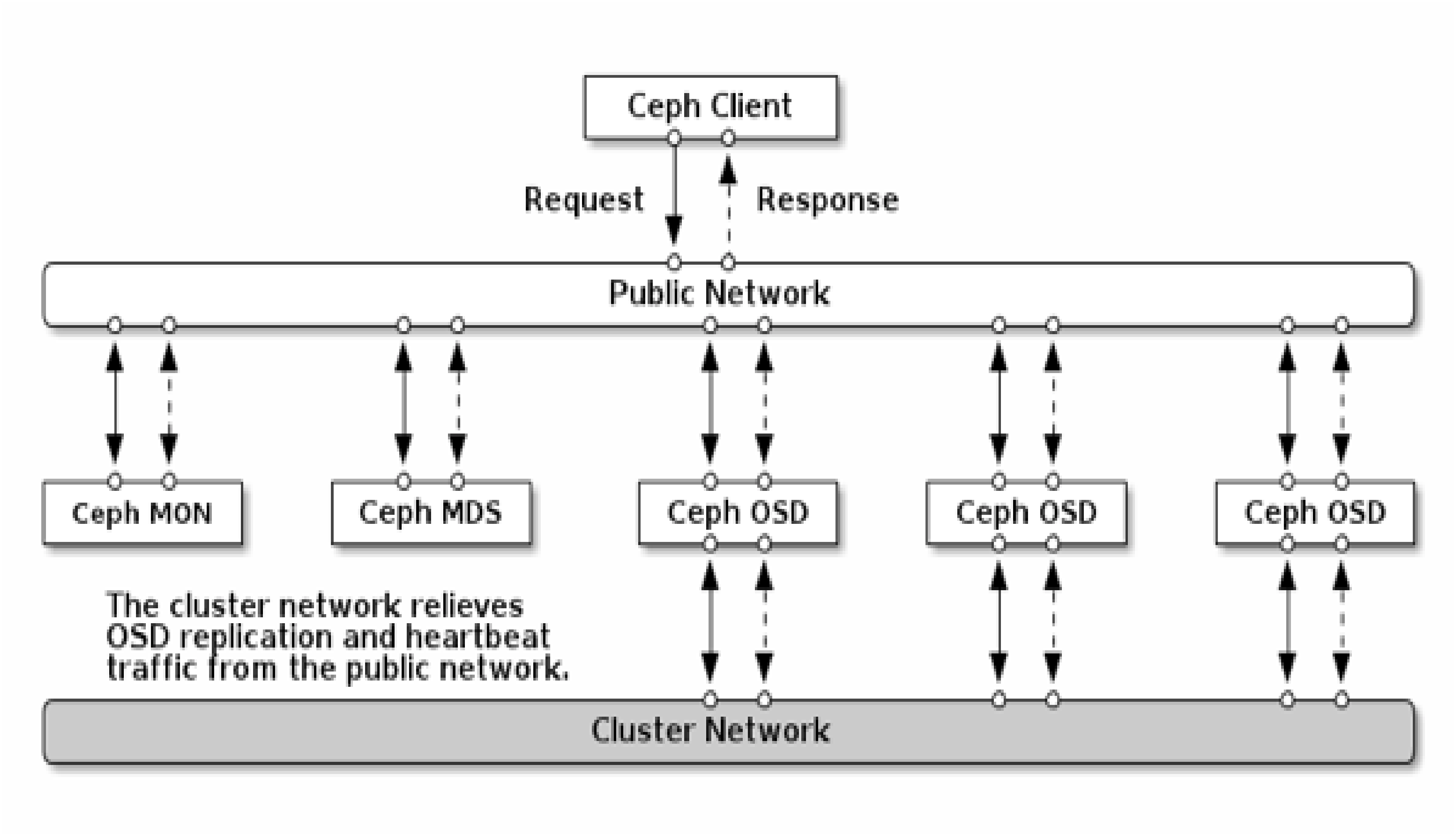
- Check it works
  - ceph status



```
$ ceph status
cluster a5bf6b38-02f4-43fb-bffc-9c932e98888e
 health HEALTH_OK
```

- Check it works
  - ceph health

Example with 3 OSD nodes



RDB client access

Create a client

- On client node called « cluster-client1 »
  - Create Linux « ceph » user
  - Install SSH keys
  - Set Sudo access
- Set DNS records for the new node
- On cluster-admin node
  - ceph-deploy install cluster-client1

Create block devices

- Create a pool called « volumes » containing objects
  - rados mkpool volumes
- Create two objects (1GB each)
  - rdb -p volumes - -size 1024 create vol1
  - rdb -p volumes - -size 1024 create vol2
- Map new objects (vol1, vol2) with block devices
  - rbd map volumes/vol1
  - rbd map volumes/vol2
- Check new blocks devices exist
  - /dev/rdb/volumes/rdb0
  - /dev/rdb/volumes/rdb1
- Format and use them !

Thanks for your attention

LINAGORA – headquarters

80, rue Roque de Fillol
92800 PUTEAUX
FRANCE

Phone : +33 (0)1 46 96 63 63
Info : info@linagora.com
Web : www.linagora.com

@linagora

facebook.com/Linagora/