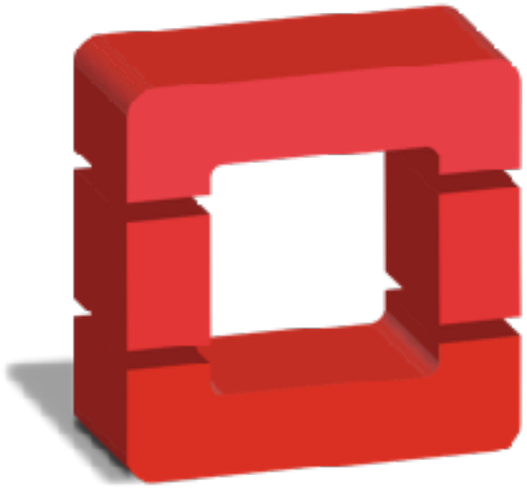


Ceph Ansible

Mr. Charnsilp Chinprasert

Outline

- Architecture
- Ceph & OpenStack
- Workshop
- Performance Tuning
- Hardware Design

Software Defined Storage

Proprietary hardware → Common, off-the-shelf hardware
Low cost, standardized supply chain

Scale-Up architecture → Scale-Out architecture
Increase operational flexibility

Hardware-based intelligence → Software-based intelligence
More programmability, agility, and control

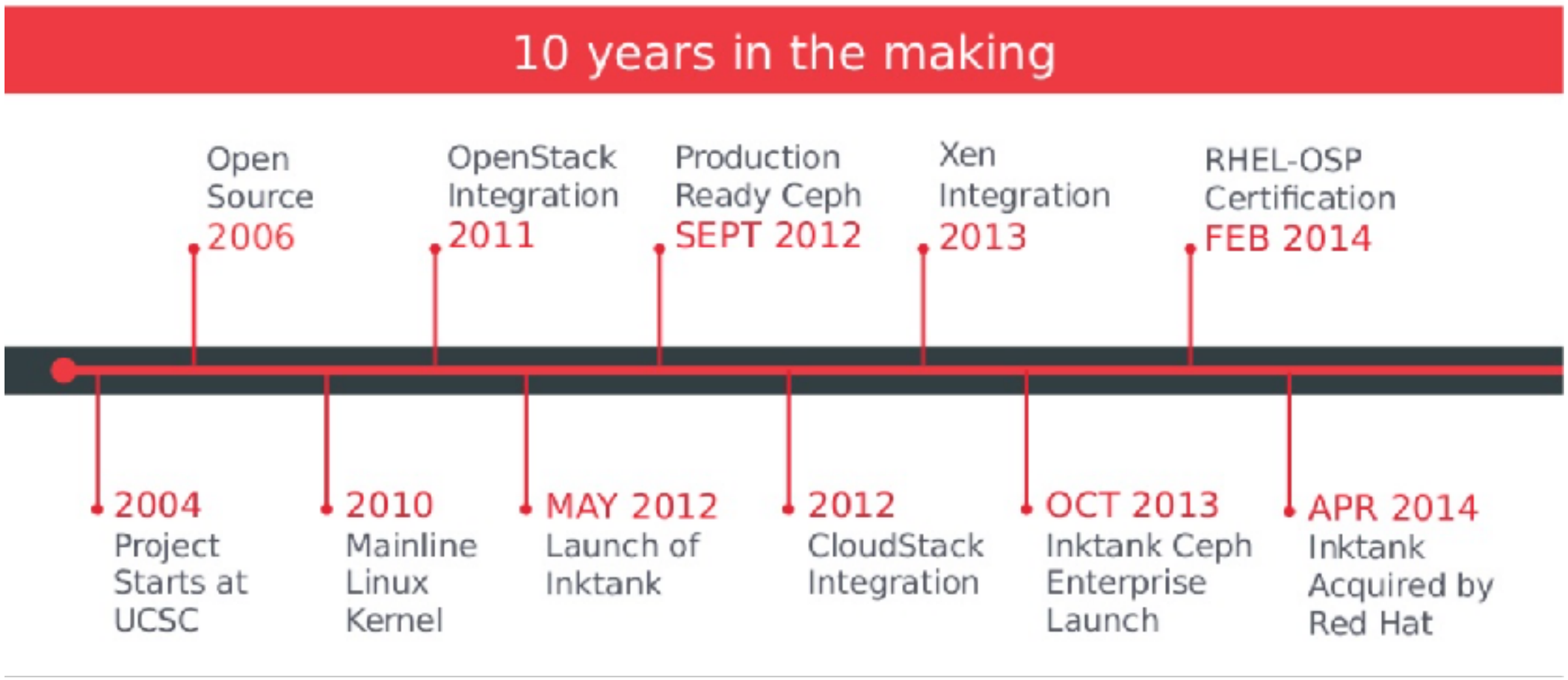
Closed development process → Open development process
More flexible, well integrated technology

What is Ceph?

- Distributed Storage System
- Self healing & Self manage
- Scale horizontally
- No single point of failure
- Runs on commodity hardware
  - no vendor lock-in
  - commodity != consumer grade
- Open source
  - GPL v2 license
  - Community =driven

ceph

History



Paper: http://www.ssrc.ucsc.edu/Papers/weil-osdi06.pdf

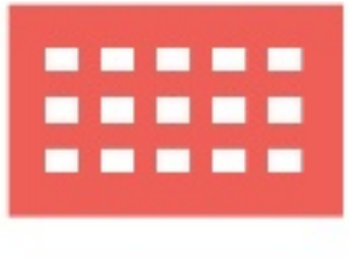
Ceph versions

TIMELINE

	Dumpling LTS	Emperor Stable	Firefly LTS	Giant Stable	Hammer LTS	Infernalis Stable	Jewel LTS
First release	August 2013	November 2013	May 2014	October 2014	April 2015	November 2015	April 2016
Estimated retirement	March 2015		January 2016		November 2016	June 2016	November 2017
Actual retirement	May 2015	May 2014	December 2015	April 2015			

	Development Testing	Dumpling LTS	Emperor Stable	Firefly LTS	Giant Stable	Hammer LTS	Infernalis Stable	Jewel LTS
August 2016						0.94.9	0.94.3	
June 2016	11.0.0							10.2.2
May 2016						0.94.7		10.2.1
April 2016	10.1.2							10.2.0
	10.1.1							
March 2016	10.1.0							
	10.0.5							
February 2016	10.0.3					0.94.6	9.2.1	
January 2016	10.0.2							
December 2015	10.0.1							
November 2015	10.0.0			0.80.11			9.2.0	
October 2015	9.1.0					0.94.5		
						0.94.4		
August 2015	9.0.3					0.94.3		
July 2015	9.0.2			0.80.10				
June 2015	9.0.1					0.94.2		
May 2015	9.0.0							
April 2015					0.87.2	0.94.1		
						0.94		
March 2015				0.80.9				

Ceph unified storage



OBJECT STORAGE

- S3 & Swift
- Multi-tenant
- CloudStack
- Geo-Replication
- Native API

BLOCK STORAGE

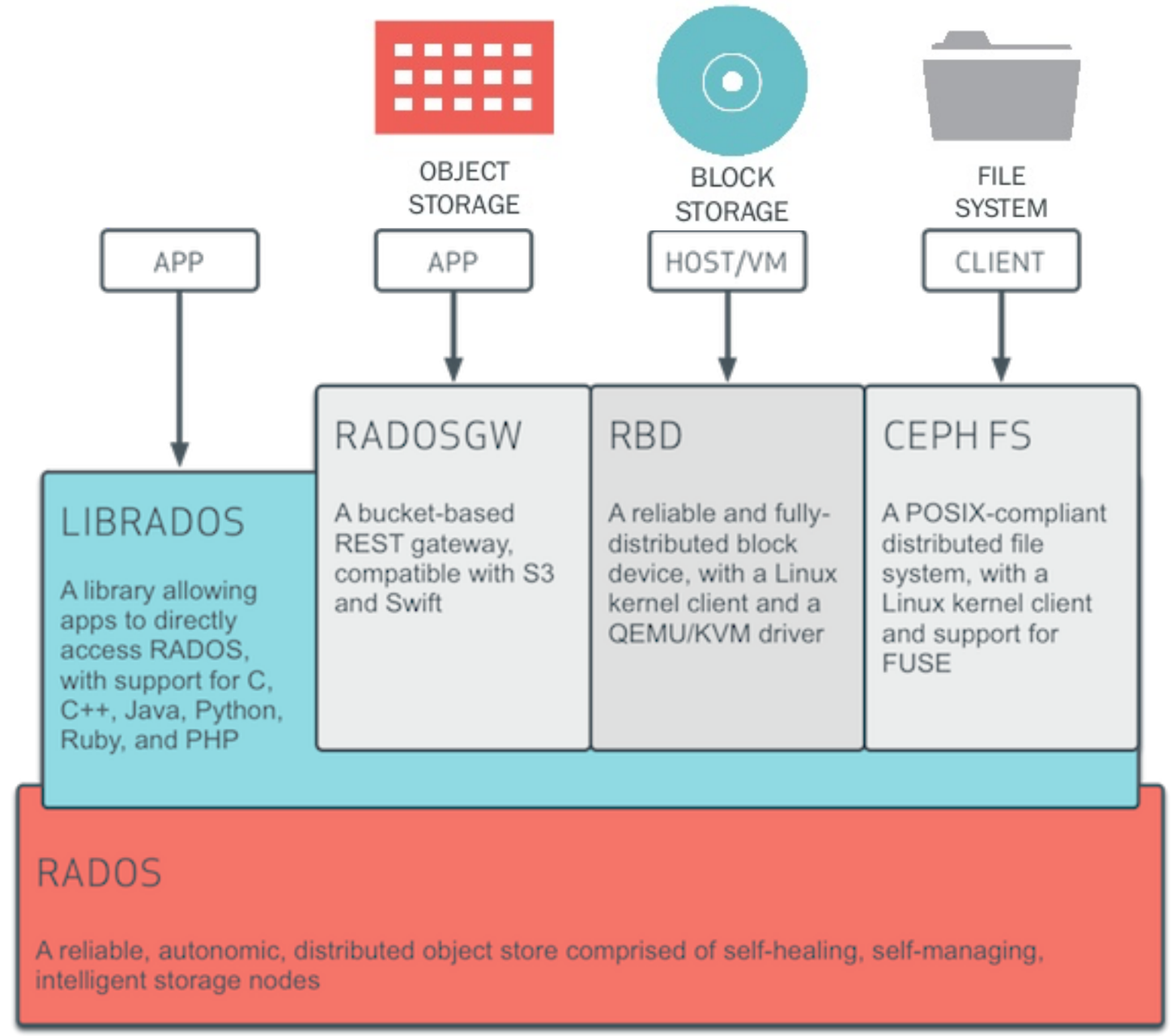
- Snapshots
- Clones
- CloudStack
- Linux Kernel
- iSCSI

FILE SYSTEM

- POSIX
- Linux Kernel
- CIFS/NFS
- HDFS
- Distributed Metadata

Ceph components

- **Rados**
  - Manage object store
- **LibRados**
  - Library for access Object store
- **RDB**
  - Linux kernel client
- **Ceph FS**
  - File system
  - Fuse
- **RadosGW**
  - REST gateway
  - S3 compatible



Ceph core components

Mon

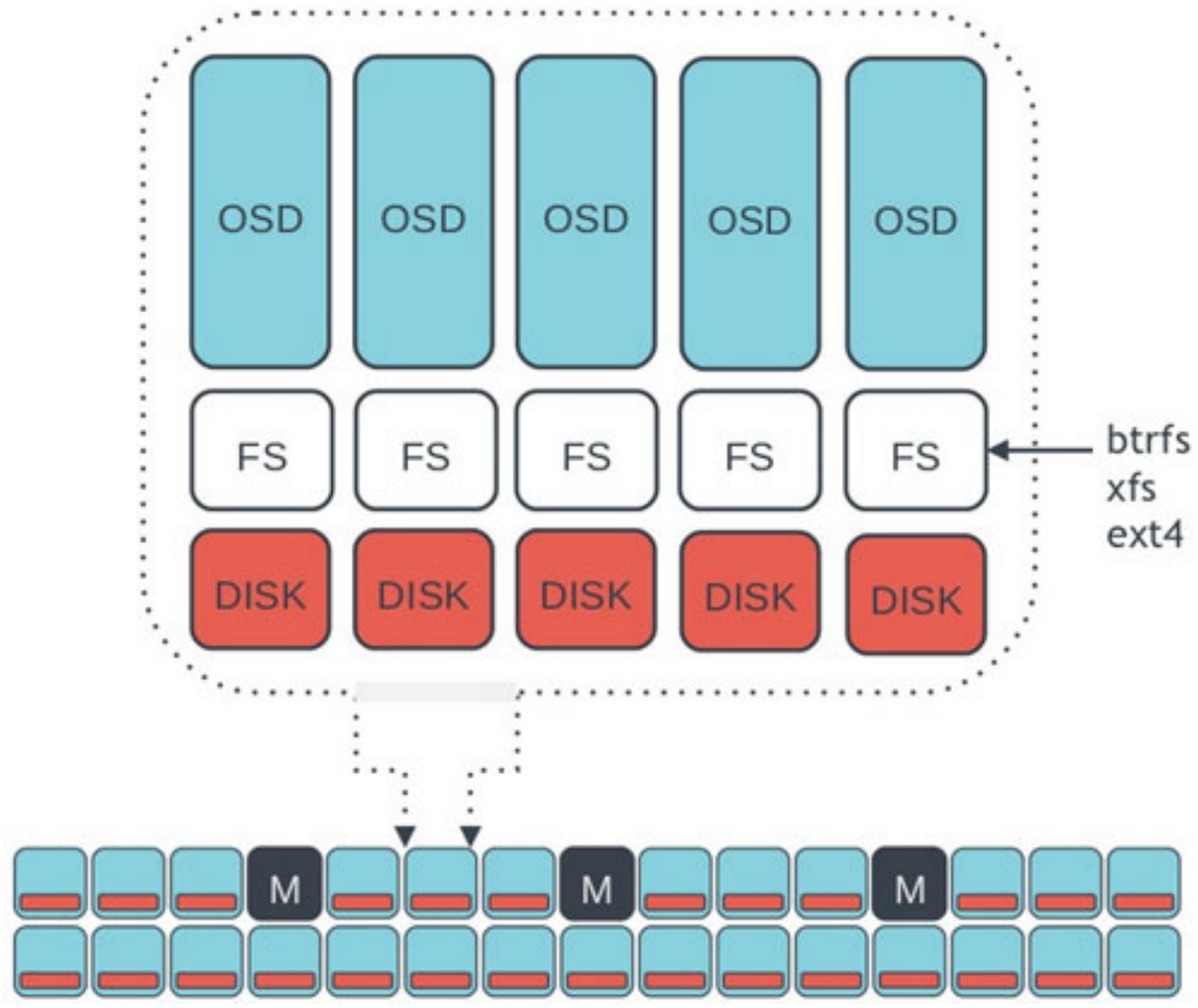
- Tracks & Monitor the health of entire cluster
- Maintains the map of cluster state
- Provide consensus for distributed decision-making
- DO NOT serve stored data to clients
- use Paxos for solving consensus in a network of unreliable processors

OSD

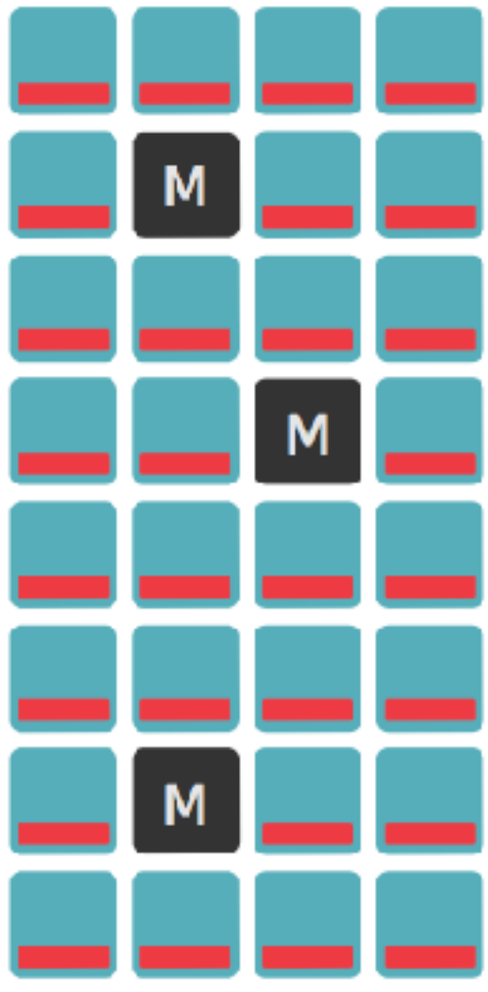
- Store the actual data as objects on physical disks
- At least 3 nodes in cluster (Data replication)
- Serve stored data to clients
- Intelligently peer to perform replication tasks

Paxos: https://en.wikipedia.org/wiki/Paxos_(computer_science)

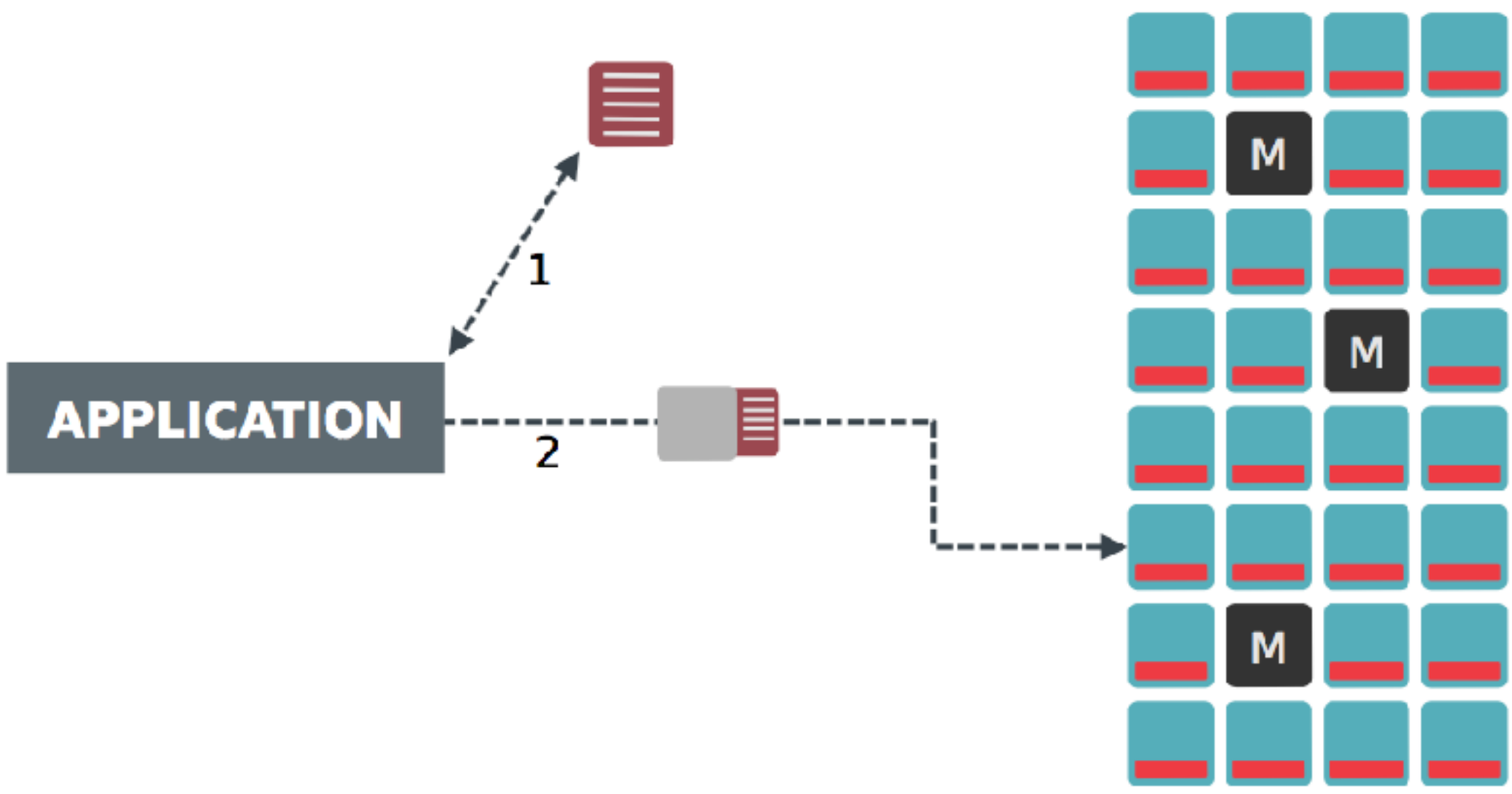
Ceph core components



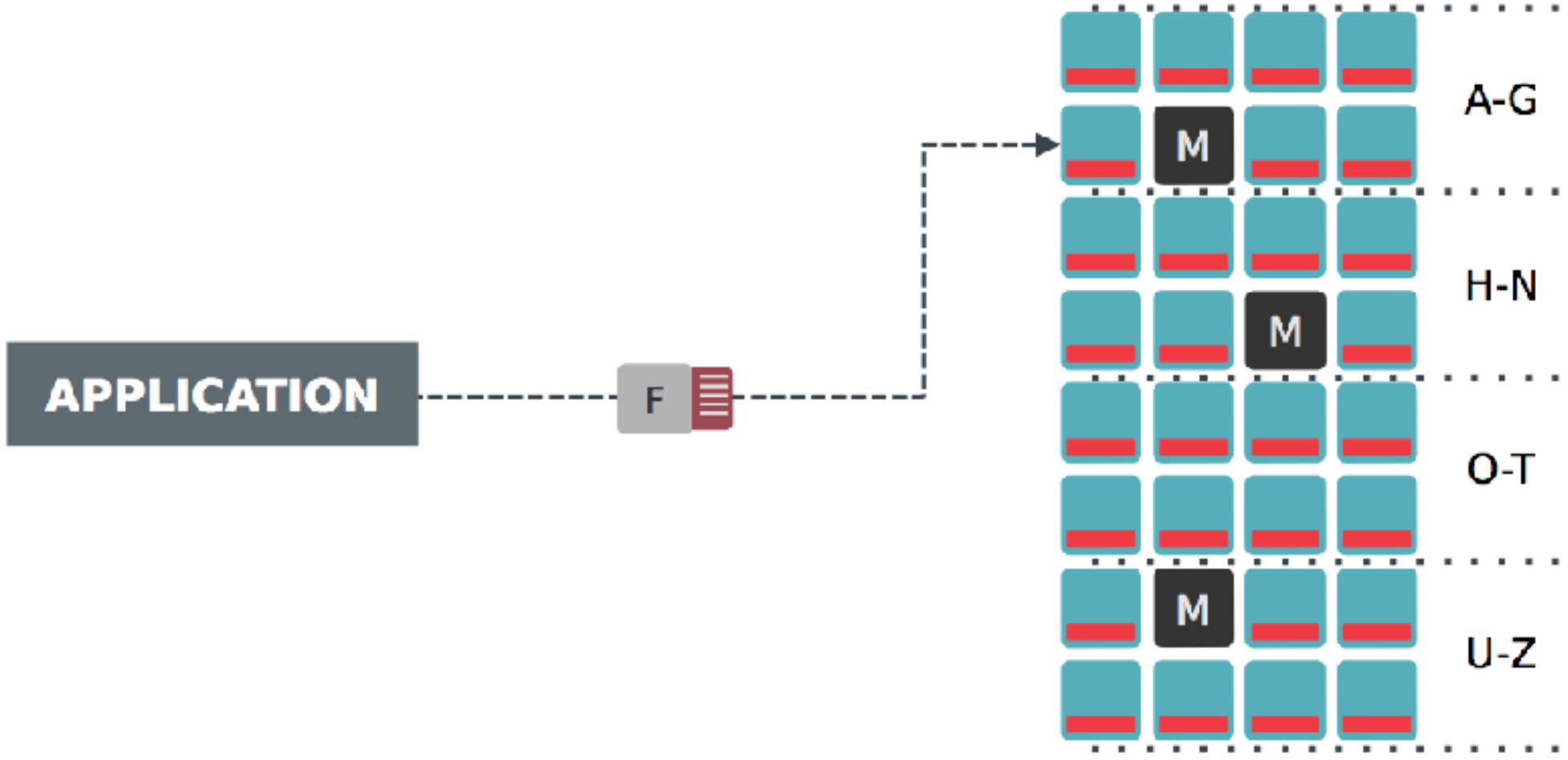
Where do object live?



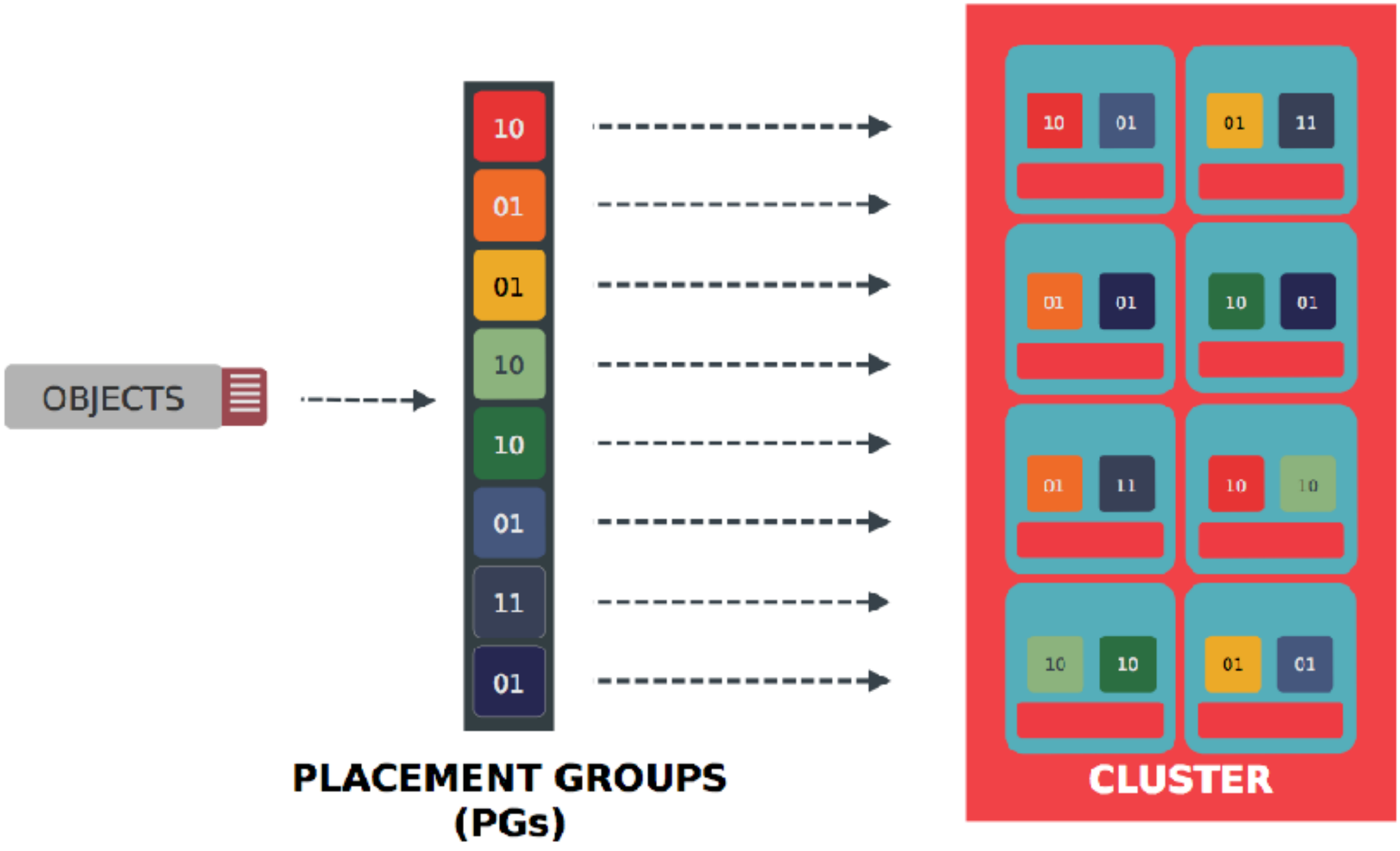
A metadata server?



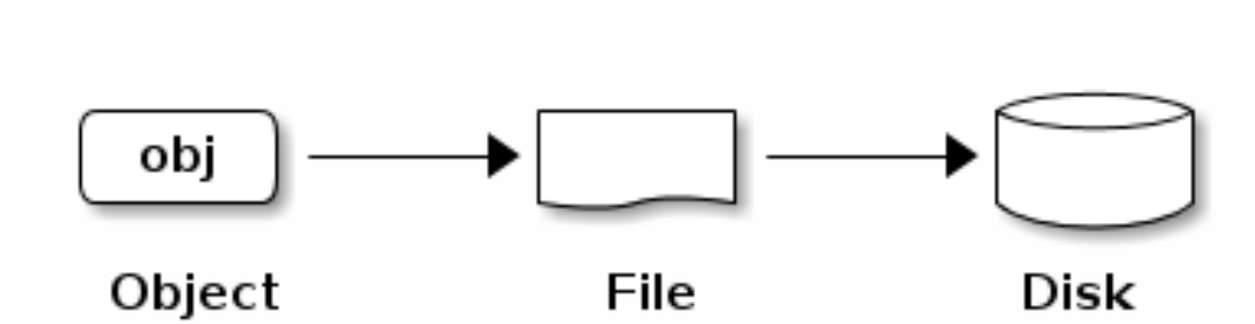
Calculated Placement



Introducing CRUSH

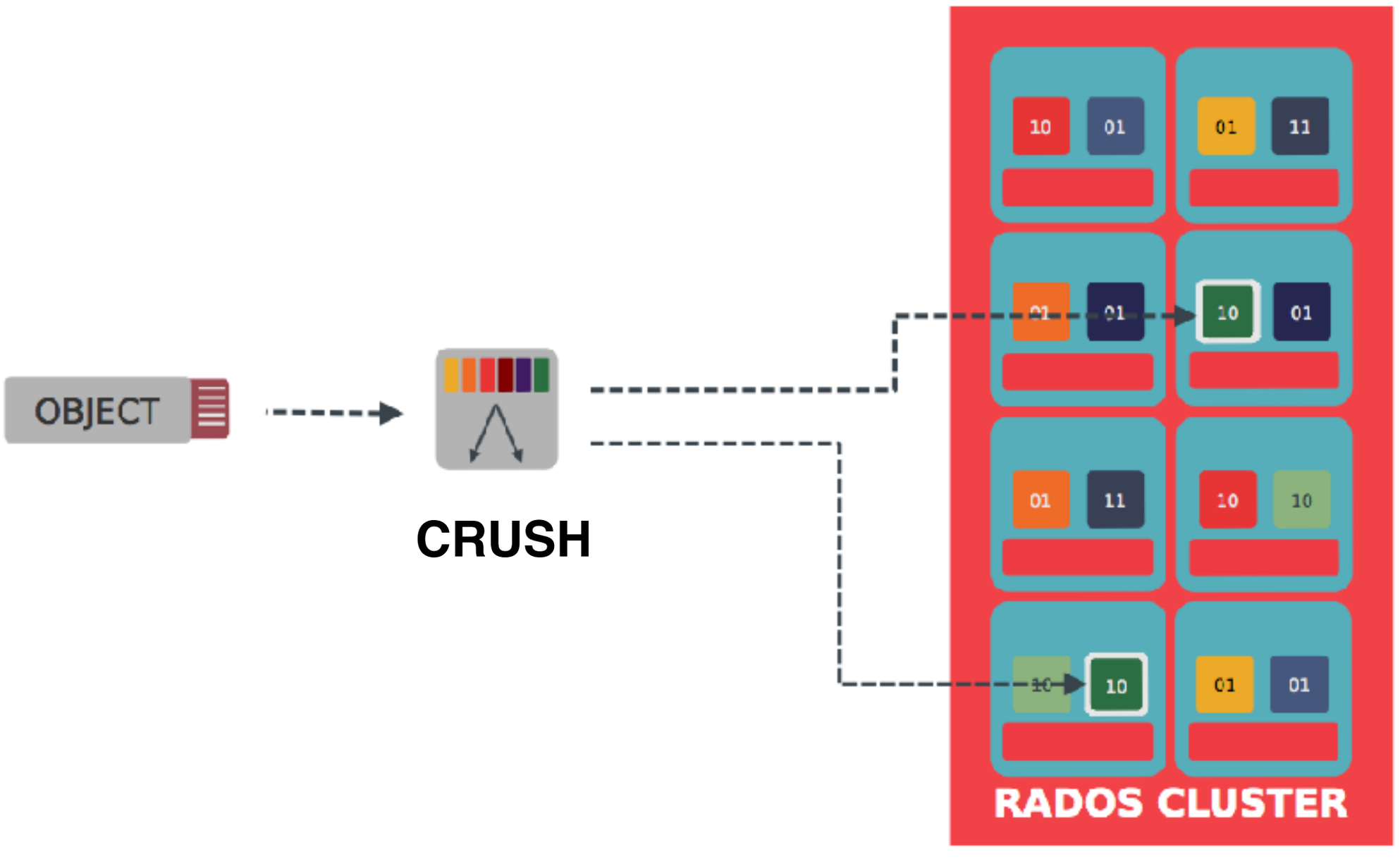


Ceph save data to Disk



A flat namespace
no hierarchy of directories!

Data Replication

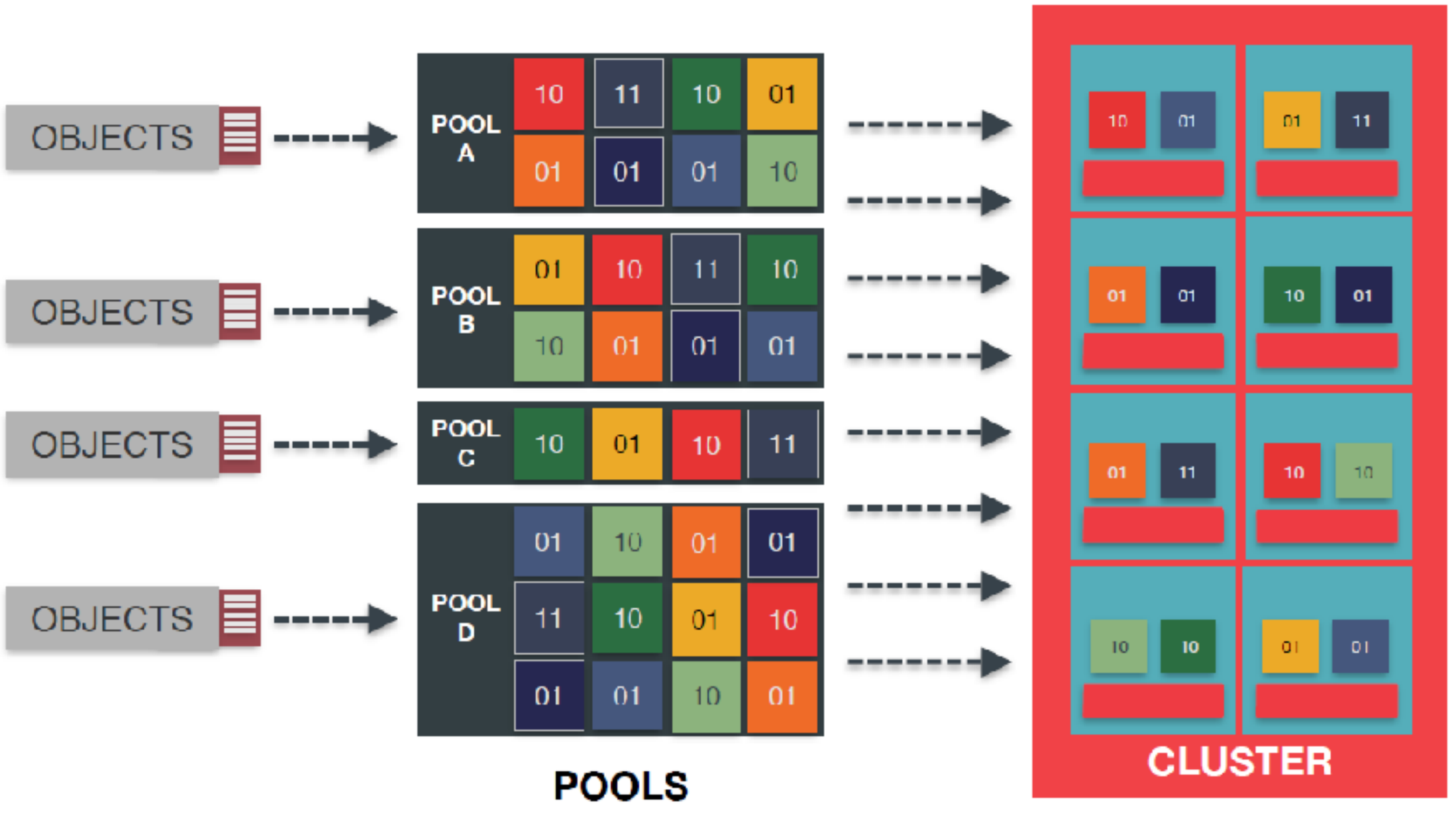


CRUSH

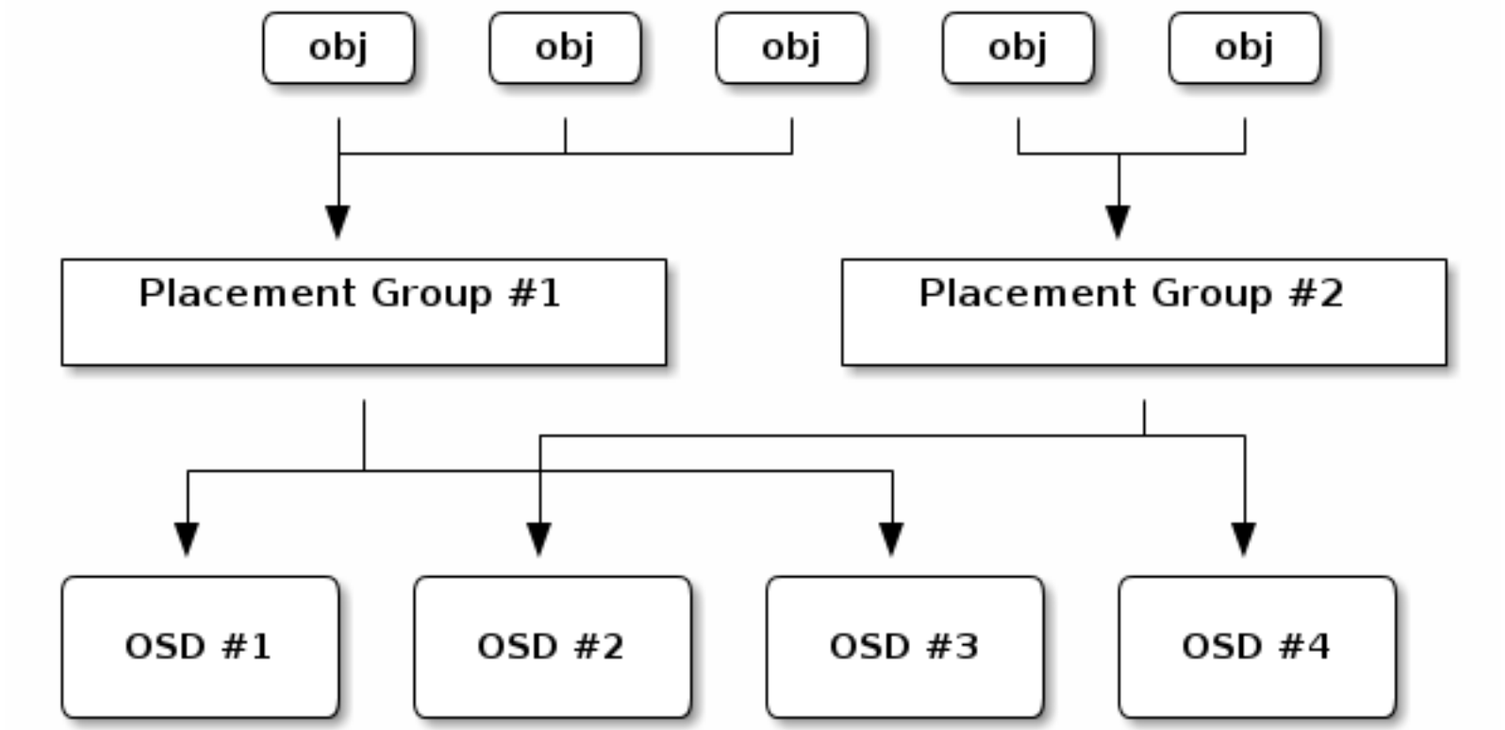
Controlled **R**eplication **U**nder **S**calable **H**ashing

- Pseudo-random placement algorithm
  - Fast calculation, no lookup
  - Repeatable, deterministic
- Statistically uniform distribution
- Rule-based configuration
  - Infrastructure topology aware
  - Adjustable replication
  - Weighting

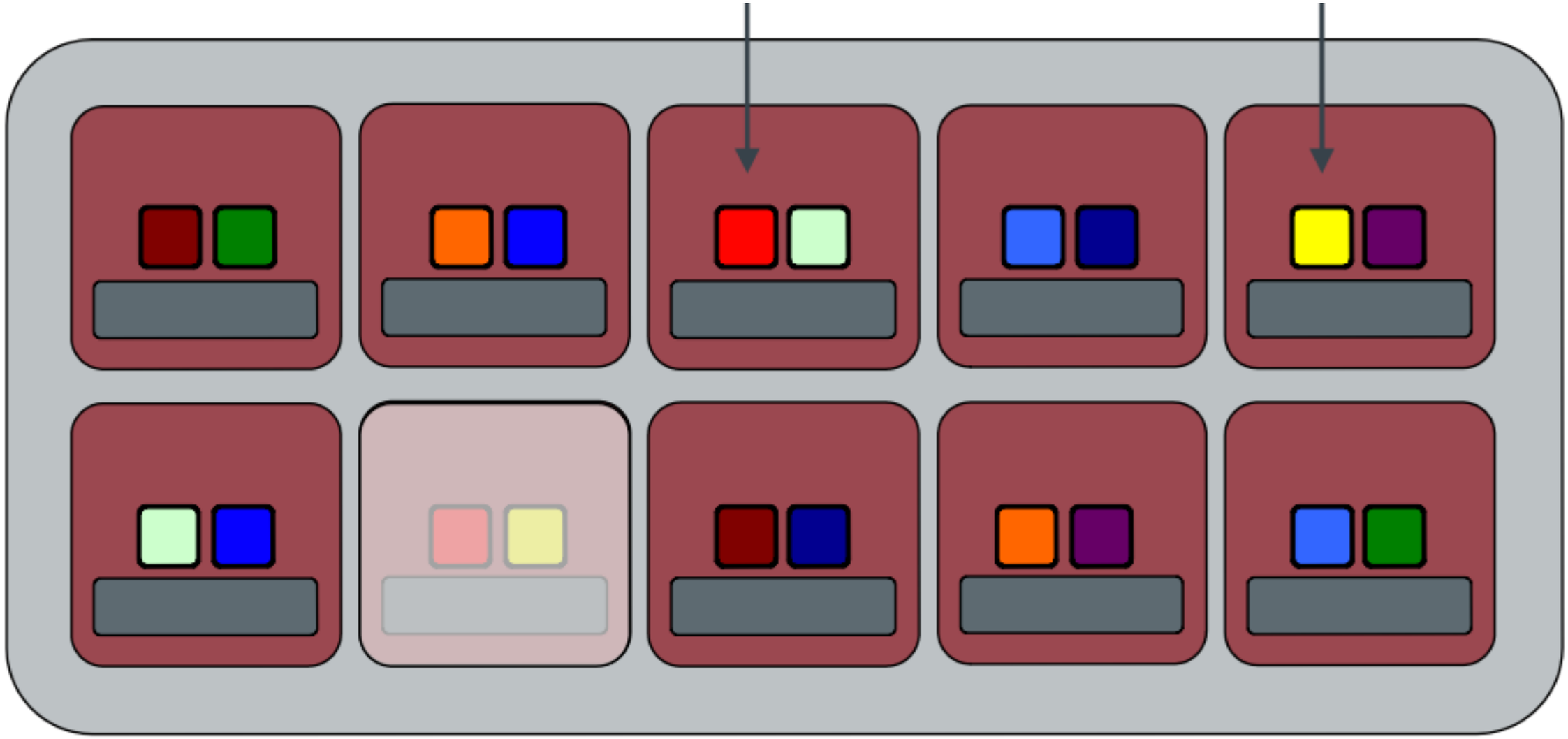
Pool


OBJECTS
OBJECTS
OBJECTS
POOL A
10 11 10 01
01 01 01 10
POOL B
01 10 11 10
10 01 01 01
POOL C
10 01 10 11
POOL D
01 10 01 01
11 10 01 10
01 01 10 01
POOLS
10 01
01 11
01 01
10 01
01 11
10 10
10 10
01 01
CLUSTER

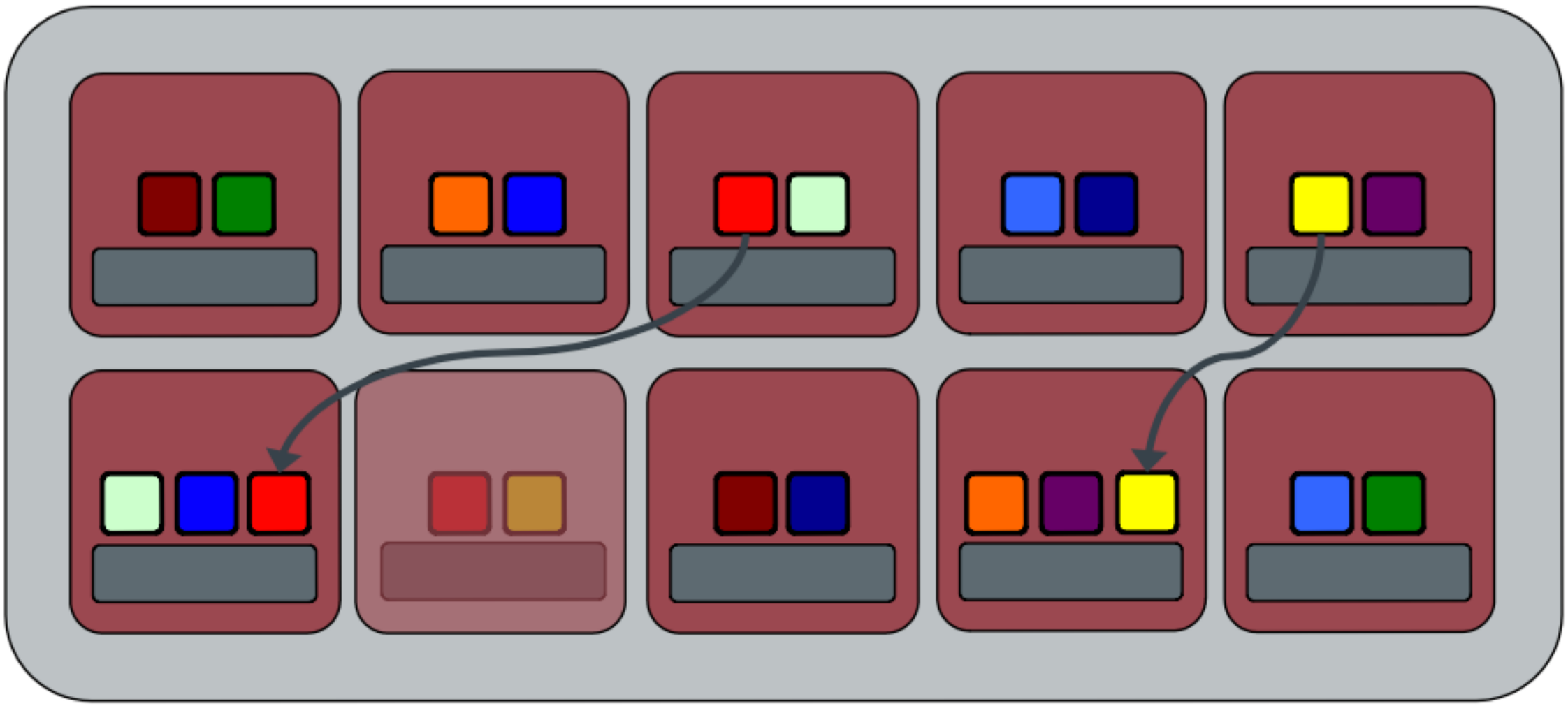
Recap


obj
obj
obj
obj
obj
Placement Group #1
Placement Group #2
OSD #1
OSD #2
OSD #3
OSD #4

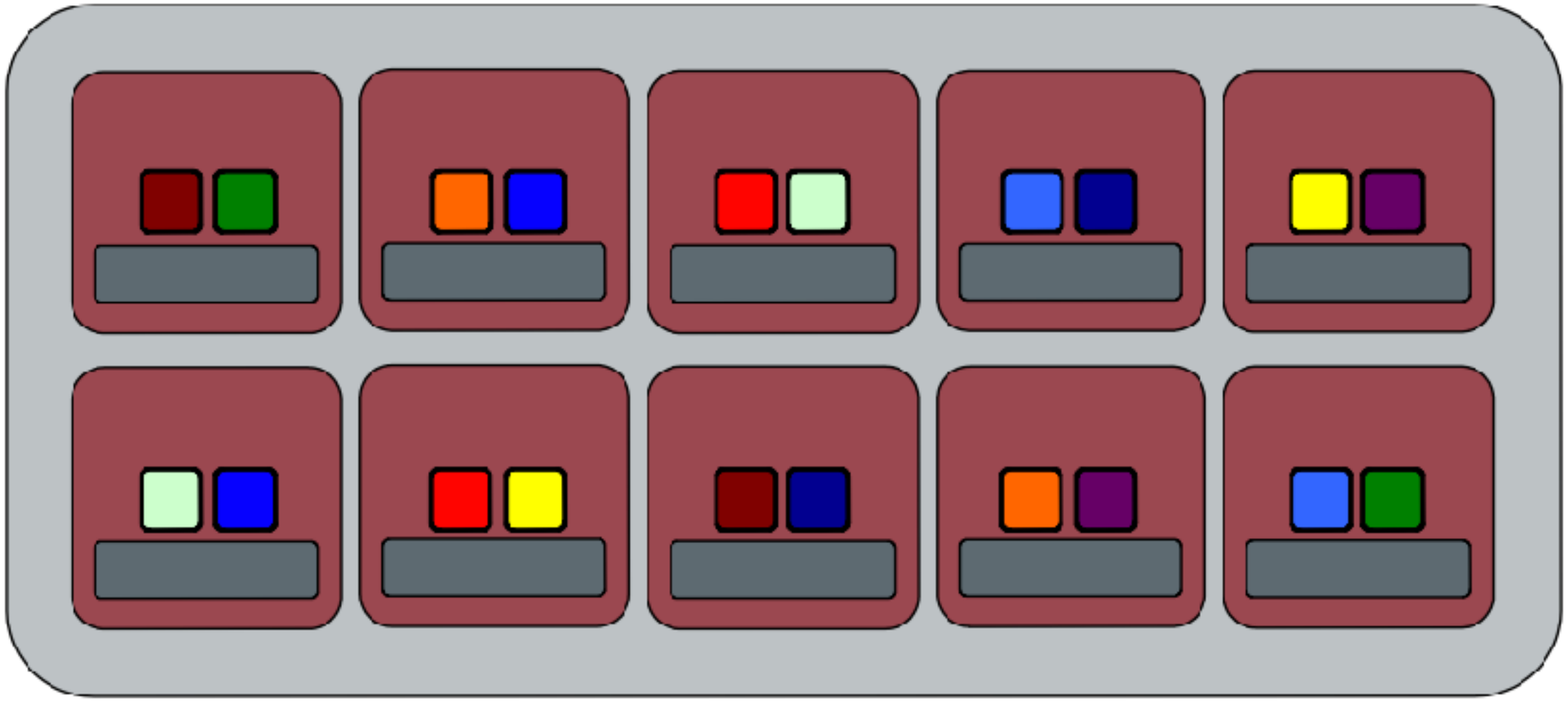
Self healing



Self healing

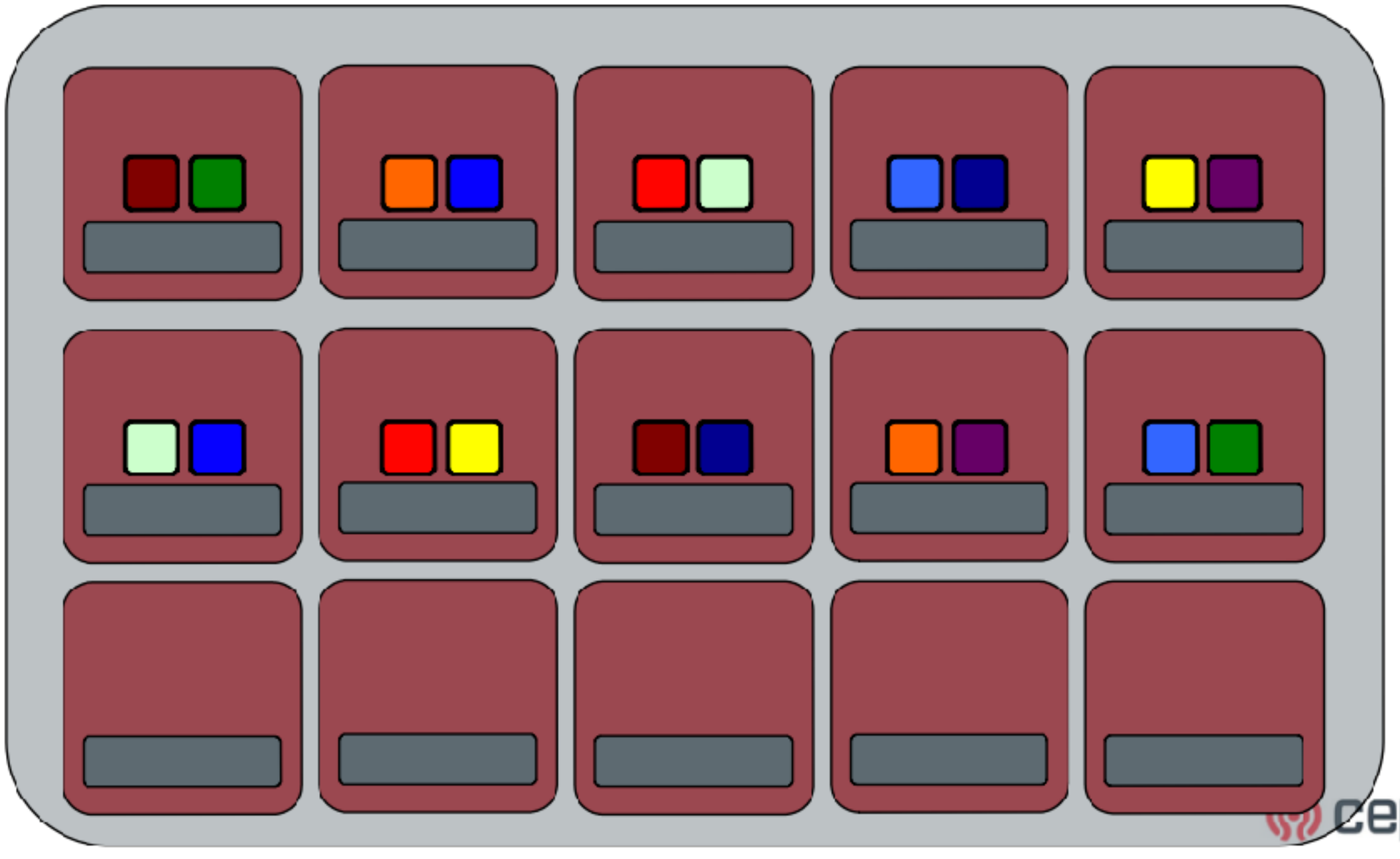


Auto Re-balancing

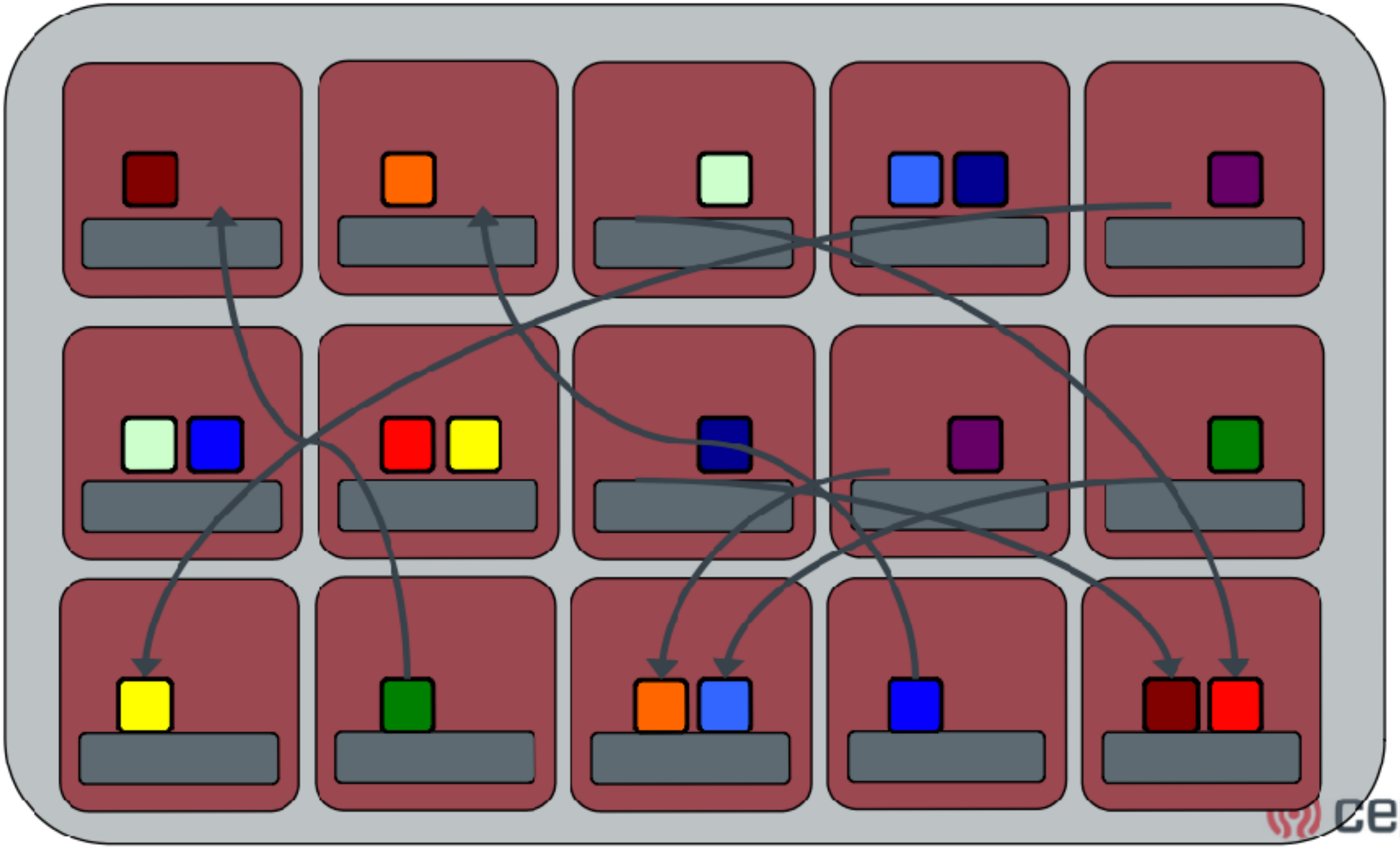


OSD node

Auto Re-balancing

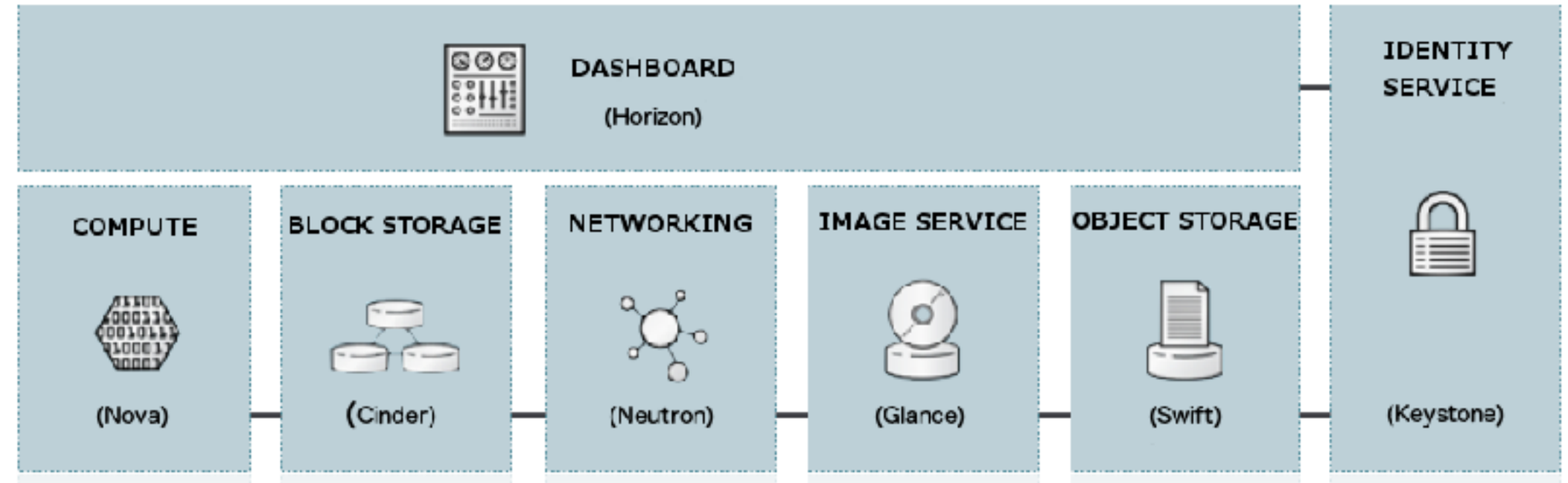


Auto Re-balancing



Ceph & OpenStack

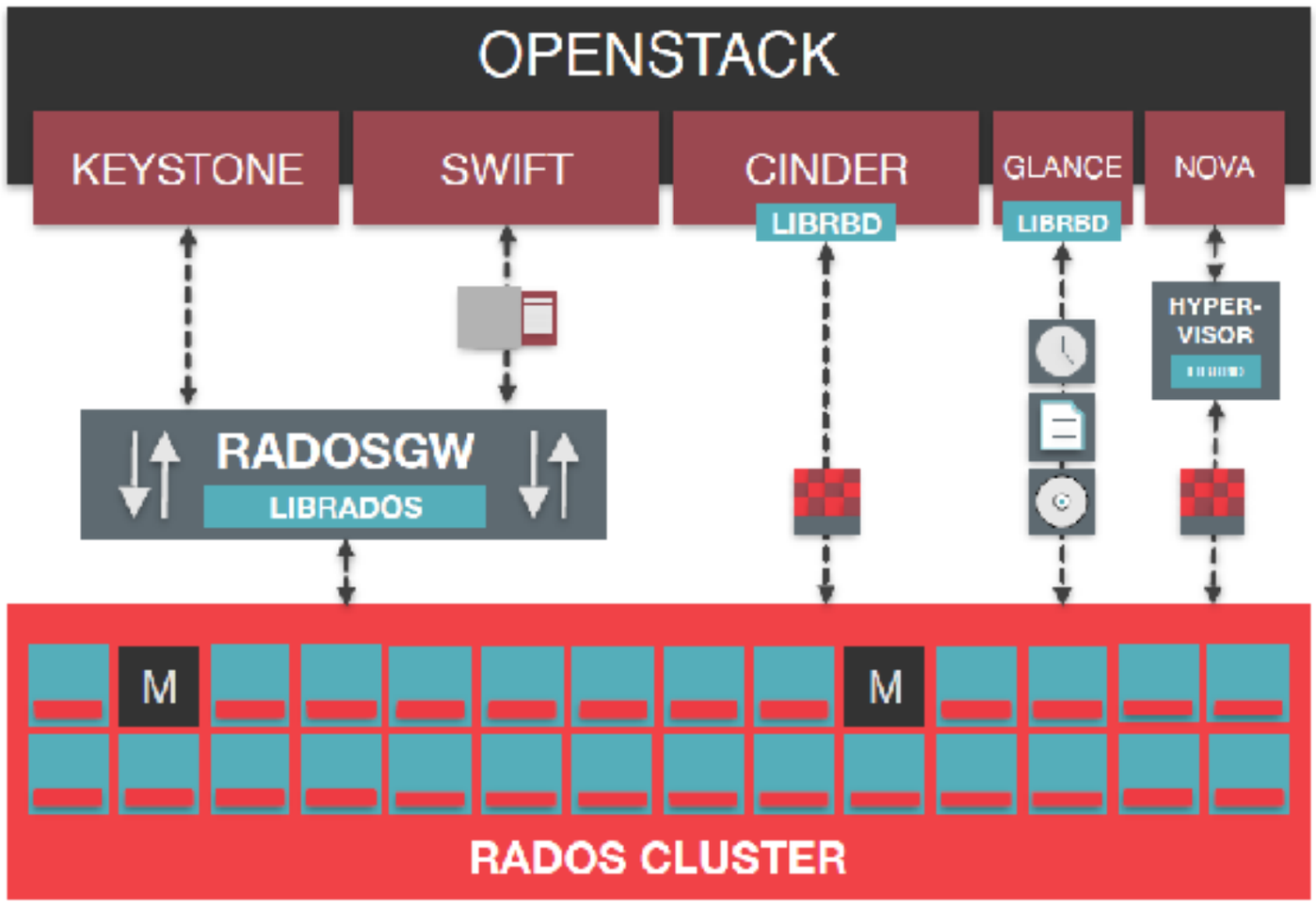
OpenStack components



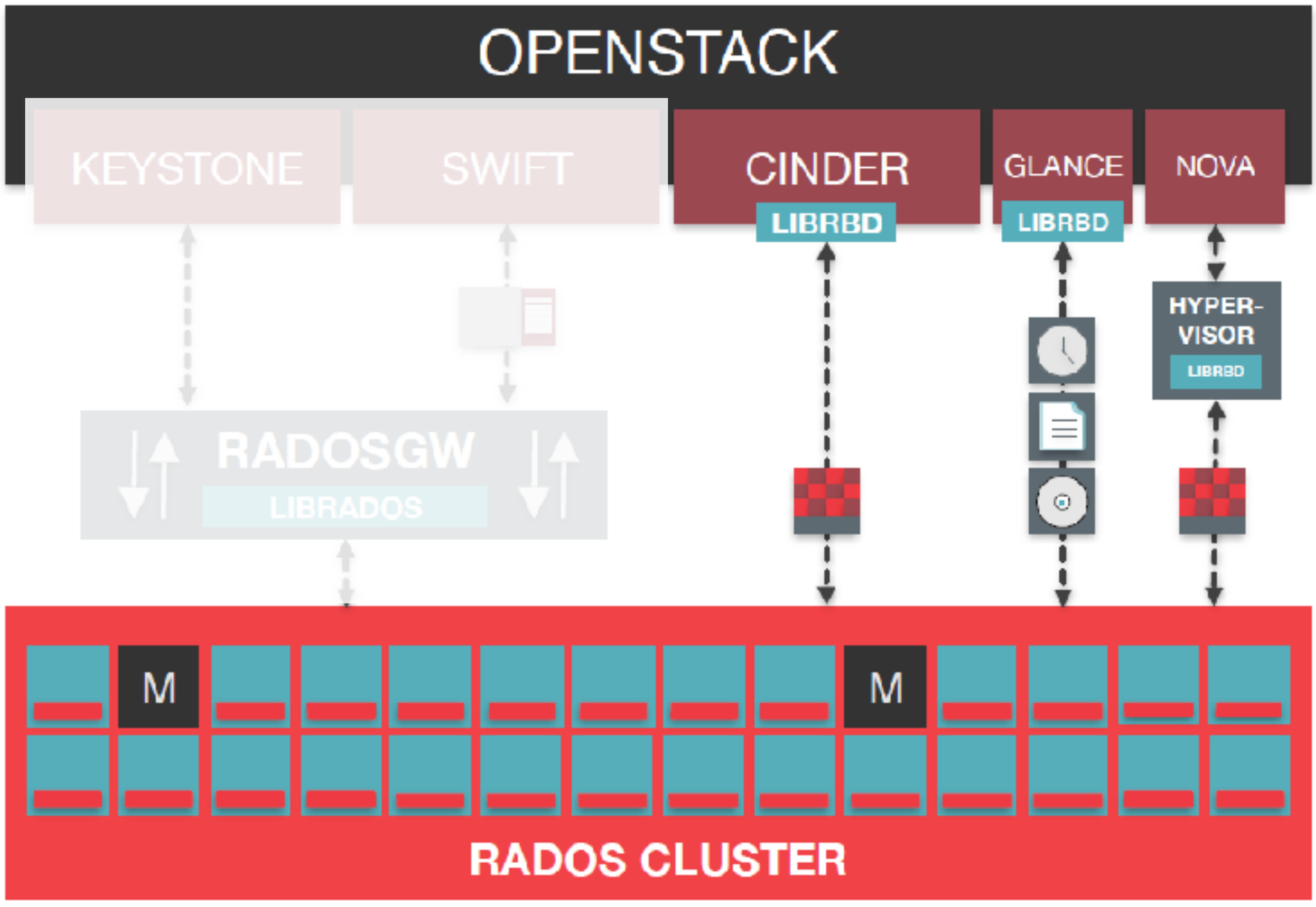
- Storage
  - Image Service (Glance)
  - Compute Storage (Nova)
  - Object Storage (Swift)

Ceph & OpenStack

- **RadosGW**
  - Keystone (Authentication)
  - Swift (Data Storage)
- **LibRBD**
  - Cinder (Volume)
  - Glance (Image)
  - Nova (Ephemeral)



Ceph & OpenStack

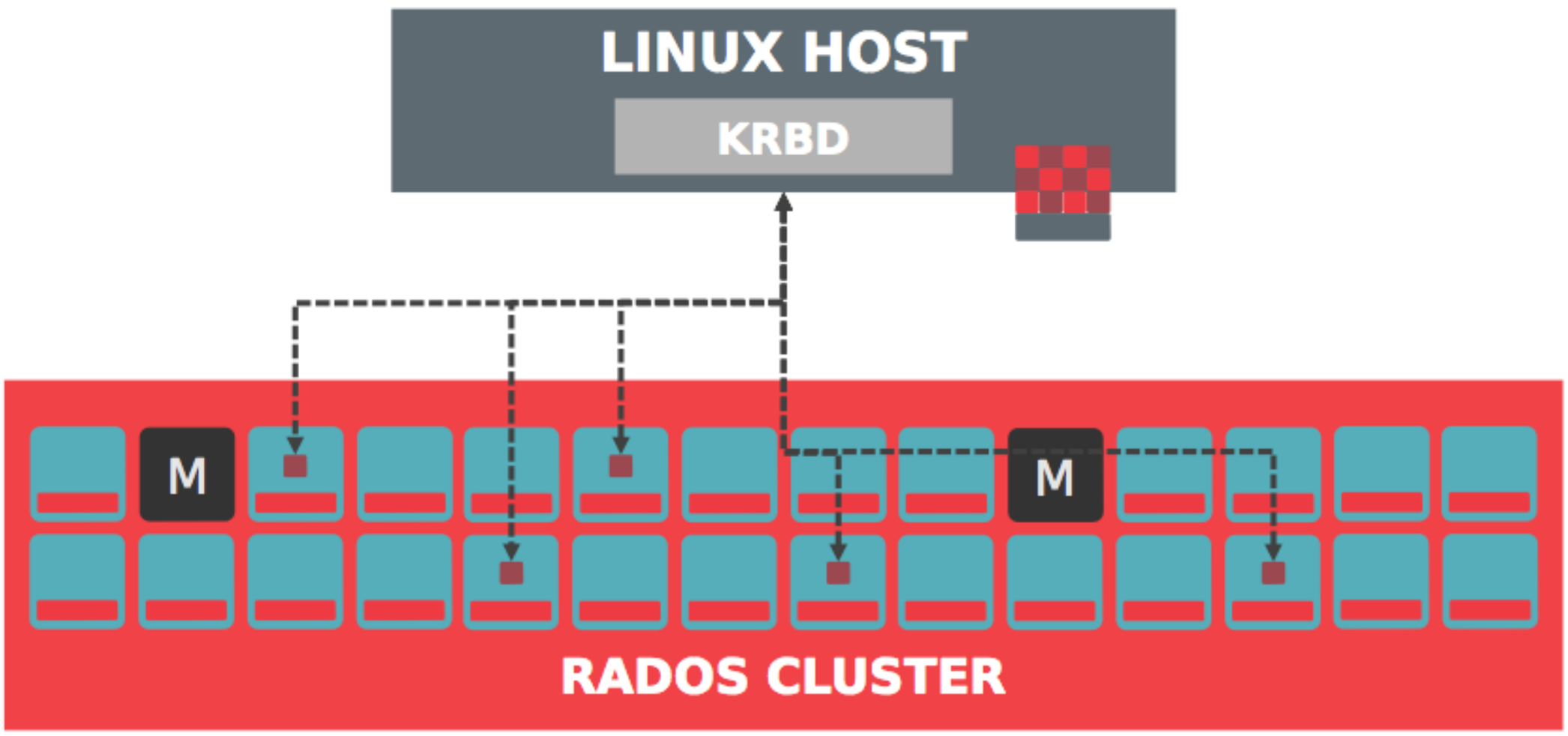


RBD Stores Virtual Disks

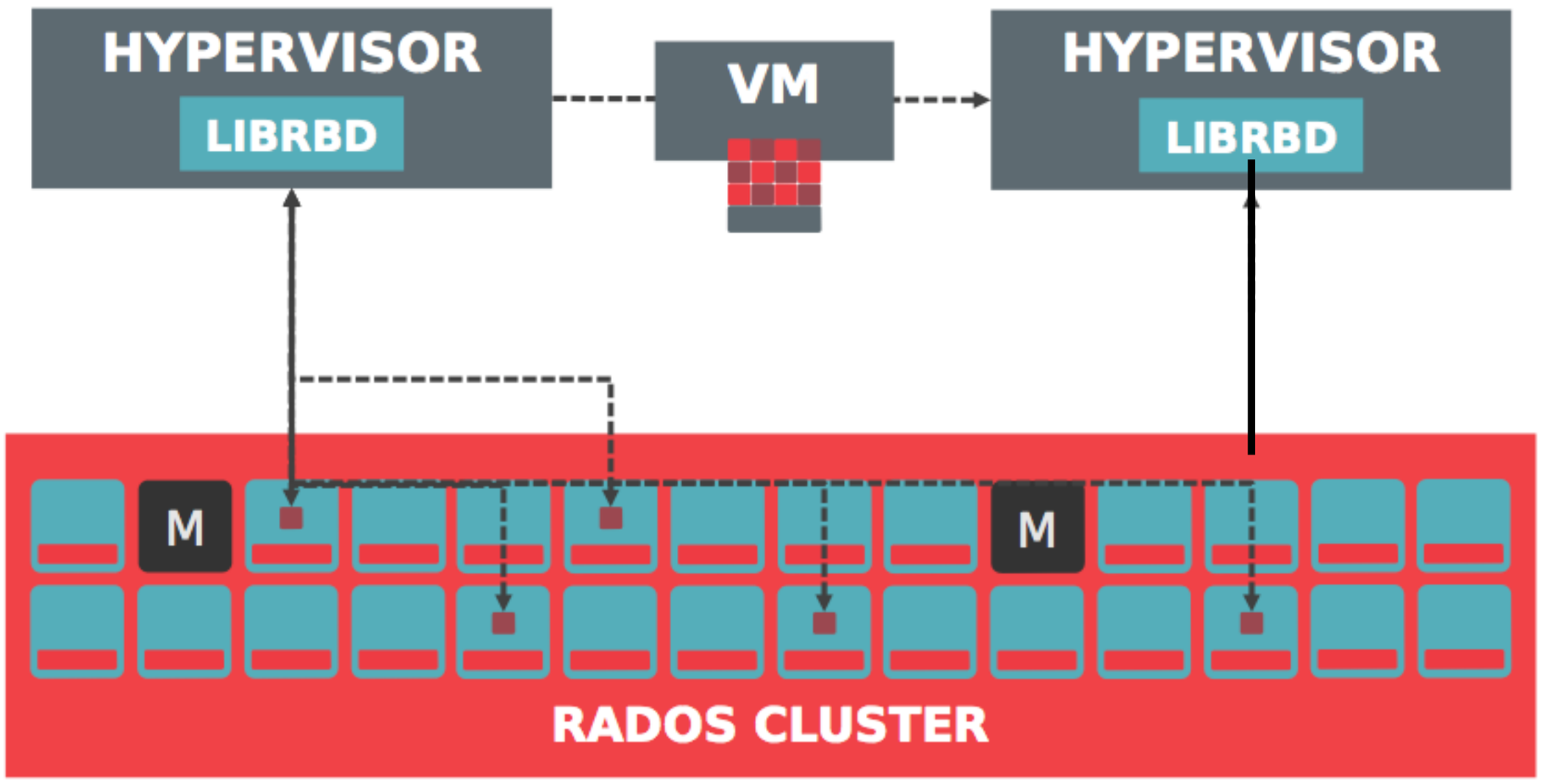
RADOS **B**LOCK **D**EVICE:

- Storage of disk images in RADOS
- Decouples VMs from host
- Images are striped across the cluster (pool)
- Snapshots
- Copy-on-write clones
- Support in:
  - Mainline Linux Kernel (2.6.39+) and RHEL 7
  - OpenStack, CloudStack, Nebula, Proxmox

Linux & RBD



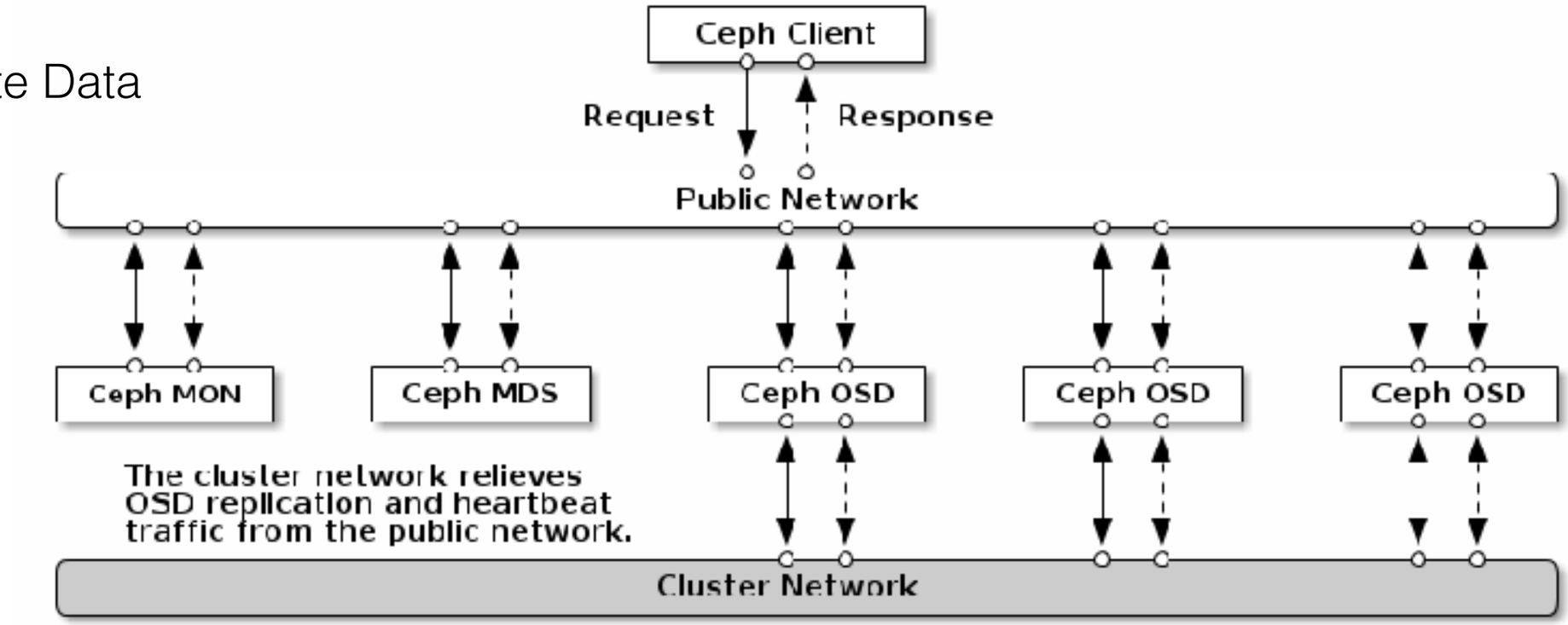
Hypervisor & Ceph RBD



Network Architecture

Network Architecture

- **Public network**
  - Recommend MTU 9000
  - Uses for Read/Write Data
- **Cluster Network**
  - MTU <= 5000
  - Replication data
  - Cluster monitoring



Preparation for OpenStack Ansible

Preparation for OpenStack Ansible

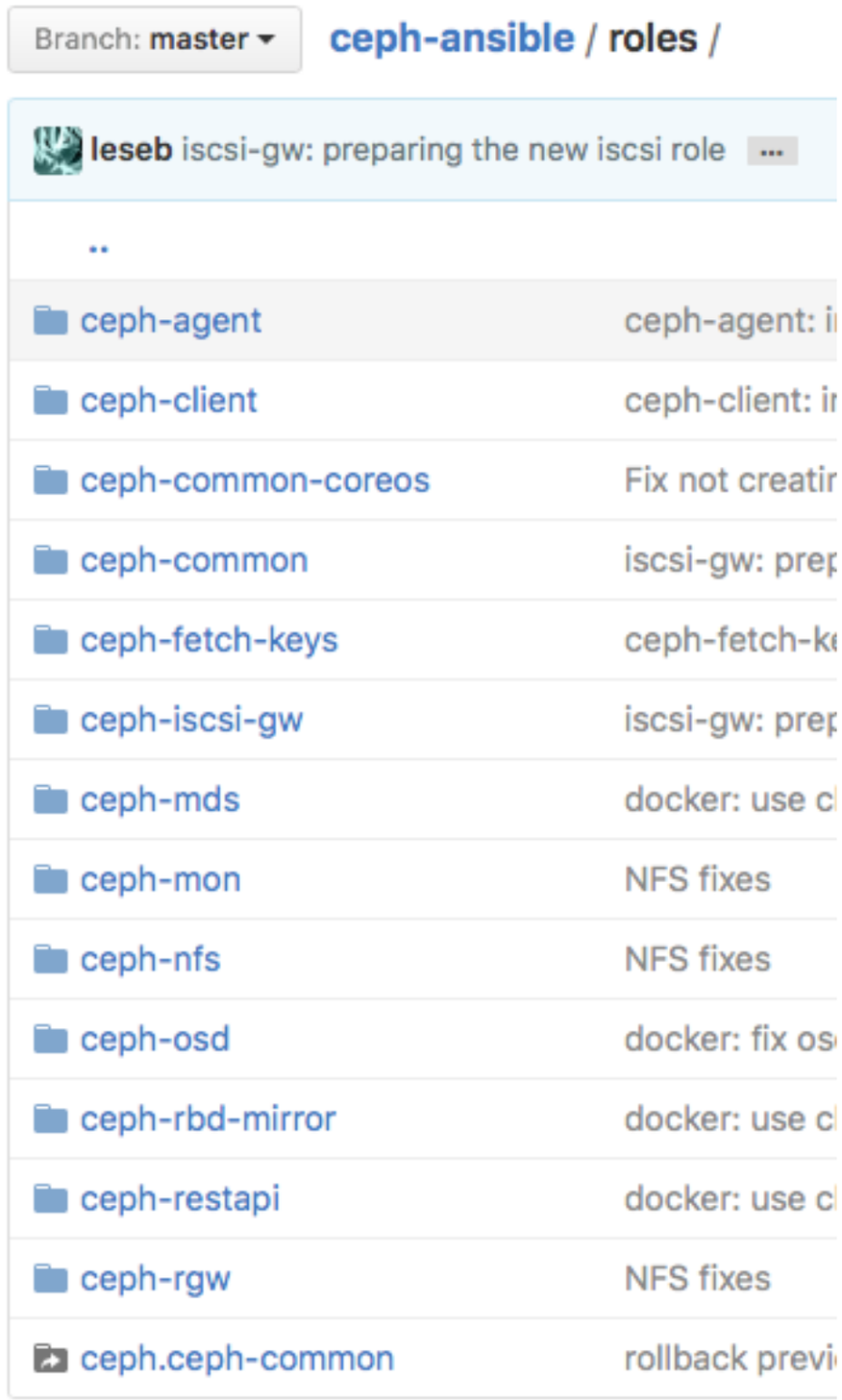
- **1. Create Users**
- **2. Create Pools**
- **3. Set PG for each pools.**

```
---
dummy:

fetch_directory: fetch/
cephx: true

openstack_config: true
openstack_glance_pool:
  name: glance-images
  pg_num: "128"
openstack_cinder_pool:
  name: cinder-volumes
  pg_num: "512"
openstack_nova_pool:
  name: nova-vms
  pg_num: "128"
openstack_cinder_backup_pool:
  name: cinder-backups
  pg_num: "128"

openstack_keys:
  - { name: client.glance, value: "mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool={{ ope
  - { name: client.cinder, value: "mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool={{ ope
  - { name: client.cinder-backup, value: "mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool
```

Why is Ceph Ansible ?

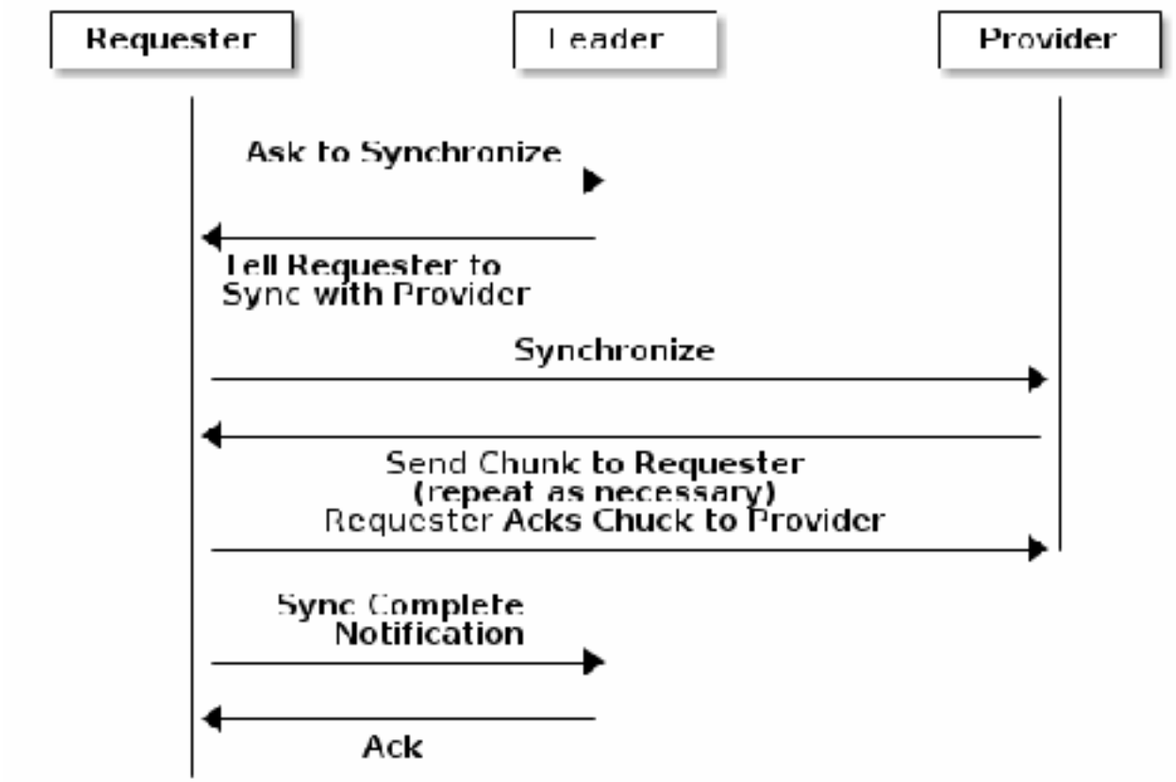
Why is Ceph Ansible ?

- Ready for production
- Basic Ceph & Linux tuning
- Automate Installation
- Configuration management
- Repeatable
- Flexible



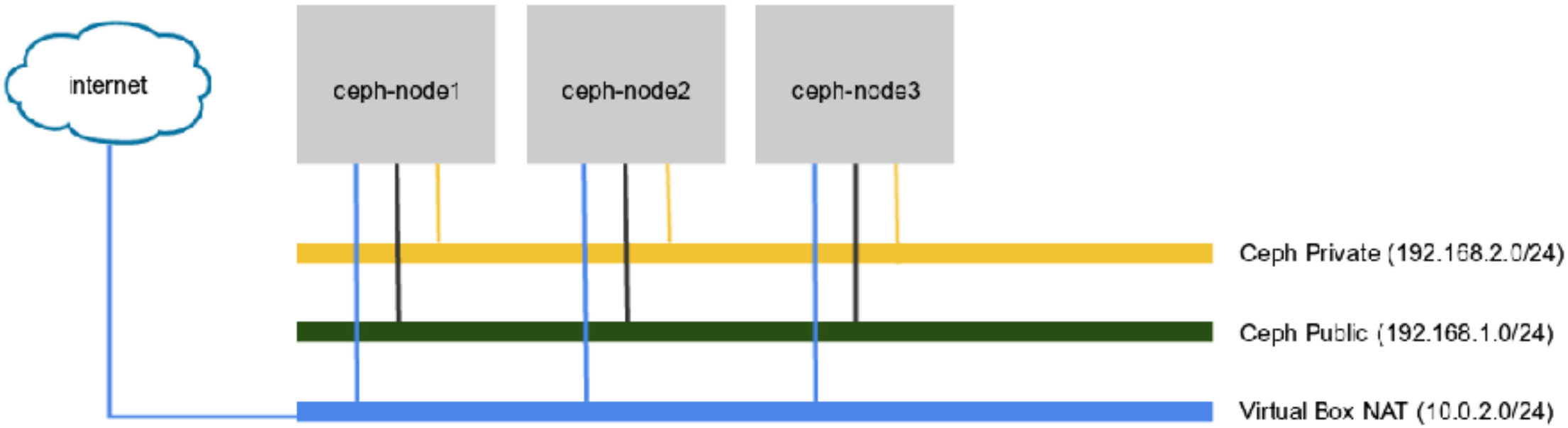
NTP

- NTP synchronization is critical for the Ceph cluster
  - Monitors can be affected by significant clock skew across the MON nodes
  - By default the maximum tolerated clock skew is 50 ms
  - “mon clock drift allowed”



Workshop

Network Design Workshop



Private network: Data replication

Public Network: Client Access

Virtual box NAT : Download package (optional)

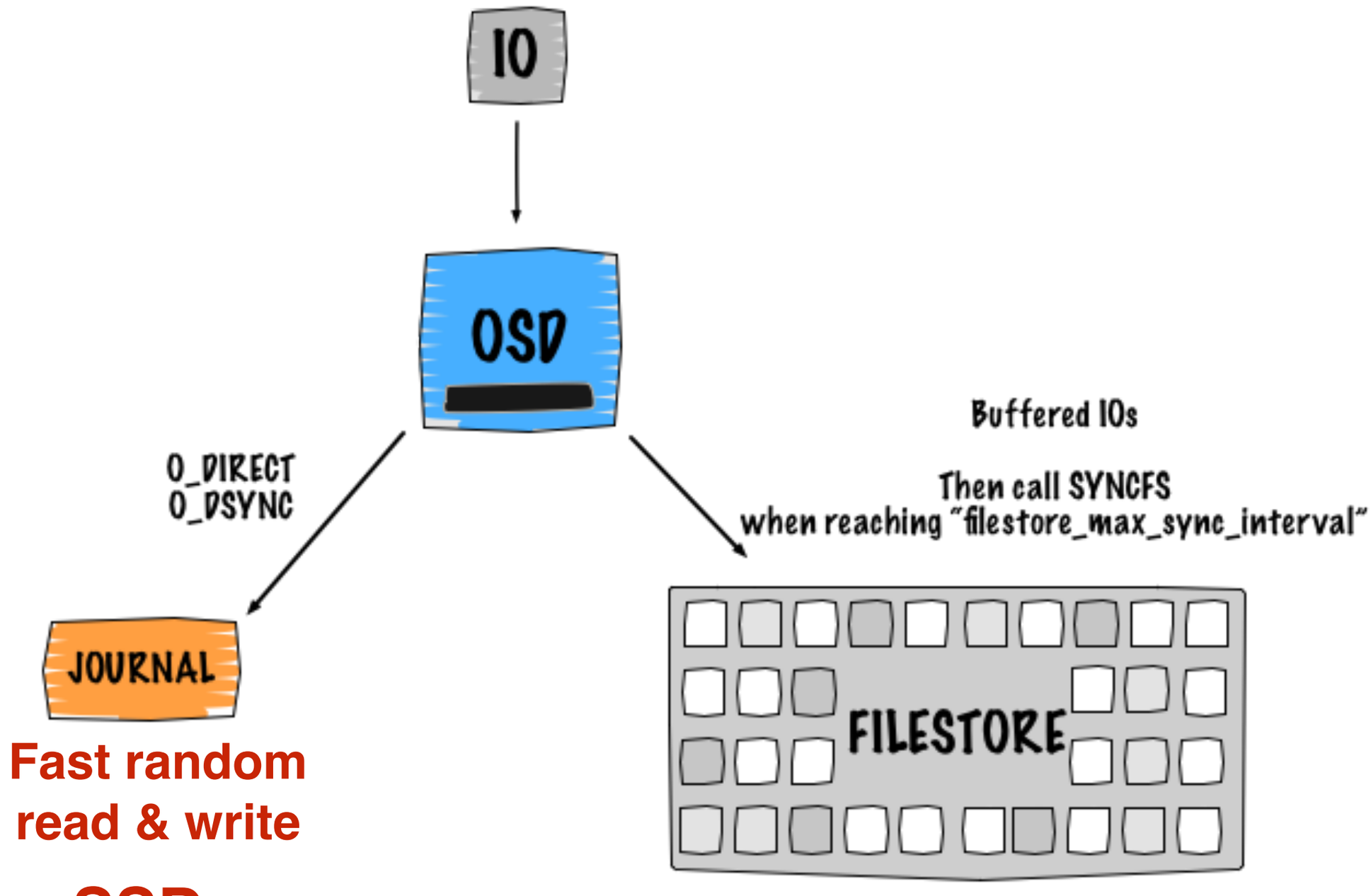
Hardware Design Workshop

Host	Role	Network	Interface	Network Address
ceph-node1	Ceph **Client** + Ceph **MON** + Ceph **OSD**	Virtualbox NAT	eth0	DHCP
		Ceph public network	eth1	192.168.1.101
		Ceph private network	eth2	192.168.2.101
ceph-node2	Ceph **MON** + Ceph **OSD**	Virtualbox NAT	eth0	DHCP
		Ceph public network	eth1	192.168.1.102
		Ceph private network	eth2	192.168.2.103
ceph-node3	Ceph **MON** + Ceph **OSD**	Virtualbox NAT	eth0	DHCP
		Ceph public network	eth1	192.168.1.103
		Ceph private network	eth2	192.168.2.103

Virtual Hardware	Specification
CPU	1 CPU
Memory	1300 MiB
Disk	**3 x 10 G** (VDI) **1 x 20 G** (VDI)
Network	3 X 1 Gib Network

Design scenarios

Ceph Journal



http://www.sebastien-han.fr/images/ceph-a-single-io.png

Scenarios

First scenario: (If OSD is SSD, Recommend !)

Journal and OSD data on the same device

Second scenario:

Single journal device for N OSDs

Third scenario: *(use in workshop)*

N journal devices for N OSDs

Fourth scenario:

Use directory instead of disk for OSDs

Hardware Design

Minimum Hardware

Process	Criteria	Minimum Recommended
`ceph-osd`	Processor	• 1x 64-bit AMD-64 • 1x 32-bit ARM dual-core or better • 1x i386 dual-core
	RAM	~1GB for 1TB of storage per daemon
	Volume Storage	1x storage drive per daemon
	Journal	1x SSD partition per daemon (optional)
	Network	2x 1GB Ethernet NICs
`ceph-mon`	Processor	• 1x 64-bit AMD-64/i386 • 1x 32-bit ARM dual-core or better • 1x i386 dual-core
	RAM	1 GB per daemon
	Disk Space	10 GB per daemon
	Network	2x 1GB Ethernet NICs

http://docs.ceph.com/docs/jewel/start/hardware-recommendations/

Hardware Planning

1TB

- 1GB Memory RAM (Metadata)
- 500MB Memory RAM (No Metadata)
- 0.5 Core CPUs
- Hard disk drives
  - Journal: **SSD**
  - OSD: SSD/Sata/SAS
- Network
  - at least 1Gbps for public
  - at least 1Gbps for data replication

PG Calculator

http://ceph.com/pgcalc/

OpenStack

Add Pool | Generate Commands

Pool Name	Size	OSD #	%Data	Target PGs per OSD	Suggested PG Count
cinder-backup	3	100	25.00	200	2048
cinder-volumes	3	100	53.00	200	4096
ephemeral-vms	3	100	15.00	200	1024
glance-images	3	100	7.00	200	512
Total Data Percentage: 100.00%					**PG Total Count: 7680**

Notes

- **"Total Data Percentage"** below table should be a multiple of 100%.
- **"Total PG Count"** below table will be the count of Primary PG copies. However, when calculating total PGs per OSD average, you must include all copies.
- It's also important to know that the PG count can be increased, but **NEVER** decreased without destroying / recreating the pool. However, increasing the PG Count of a pool is one of the most impactful events in a Ceph Cluster, and should be avoided for production clusters if possible.

Logic behind Suggested PG Count

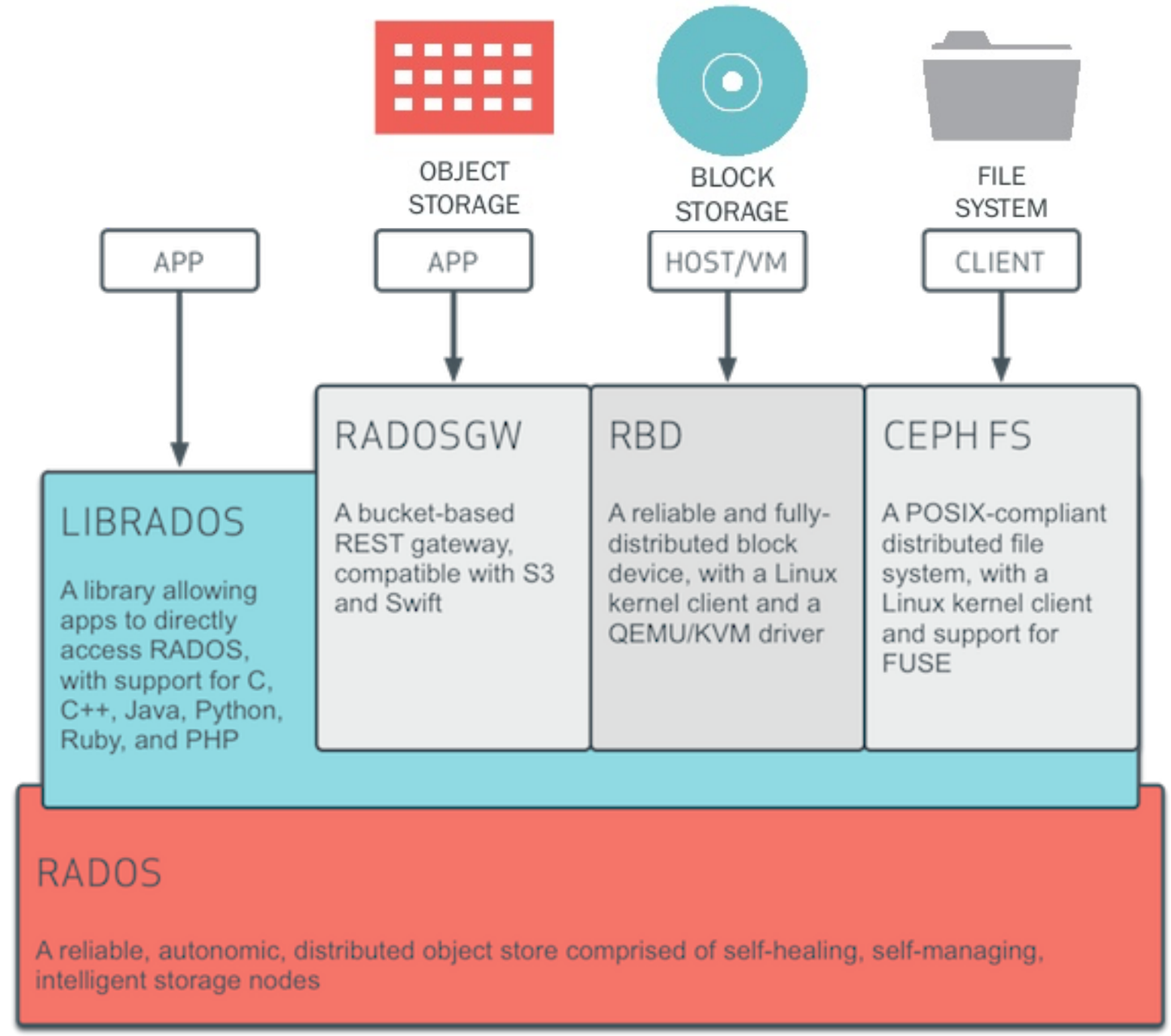
$$\frac{\text{(Target PGs per OSD)} * \text{(OSD \#)} * \text{(\%Data)}}{\text{(Size)}}$$

- If the value of the above calculation is less than the value of (OSD#) / (Size), then the value is updated to the value of ((OSD#) / (Size)). This is to ensure even load / data distribution by allocating at least one Primary or Secondary PG to every OSD for every Pool.
- The output value is then rounded to the nearest power of 2.
  Tip: The nearest power of 2 provides a marginal improvement in efficiency of the CRUSH algorithm.
- If the nearest power of 2 is more than **25%** below the original value, the next higher power of 2 is used.
- The objective of this calculation and the target ranges noted in the 'Key' section above are to ensure that there are sufficient Placement Groups for even data distribution throughout the cluster, while not going high enough on the PG per OSD ratio to cause problems during Recovery and/or Backfill operations.
- Effects of empty or non-active pools:
  - Empty or otherwise non-active pools should not be considered helpful toward even data distribution throughout the cluster.
  - However, the PGs associated with these empty / non-active pools still consume memory and CPU overhead.

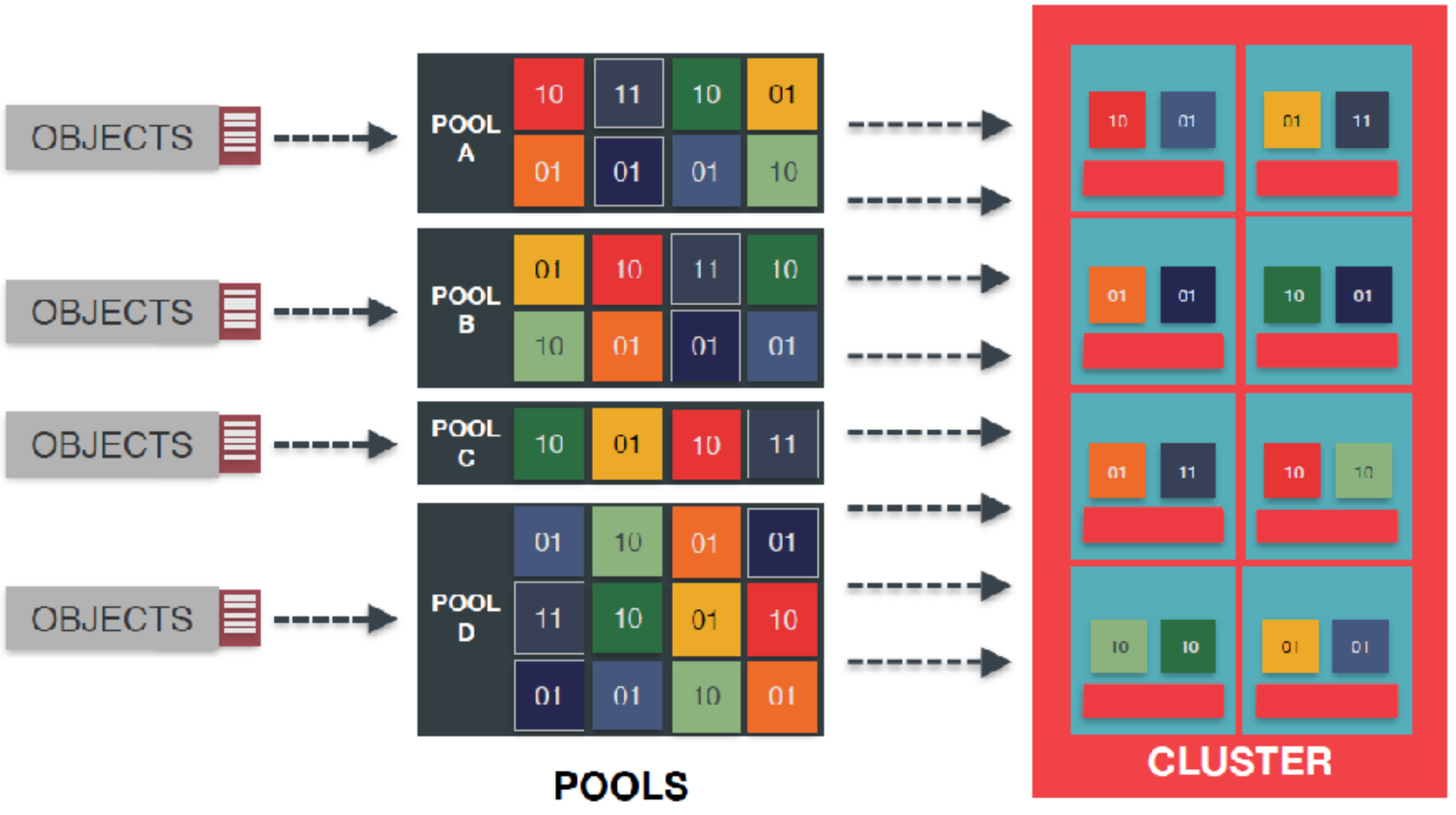
Conclusion

Ceph components

- **Rados**
  - Manage object store
- **LibRados**
  - Library for access Object store
- **RDB**
  - Linux kernel client
- **Ceph FS**
  - File system
  - Fuse
- **RadosGW**
  - REST gateway
  - S3 compatible

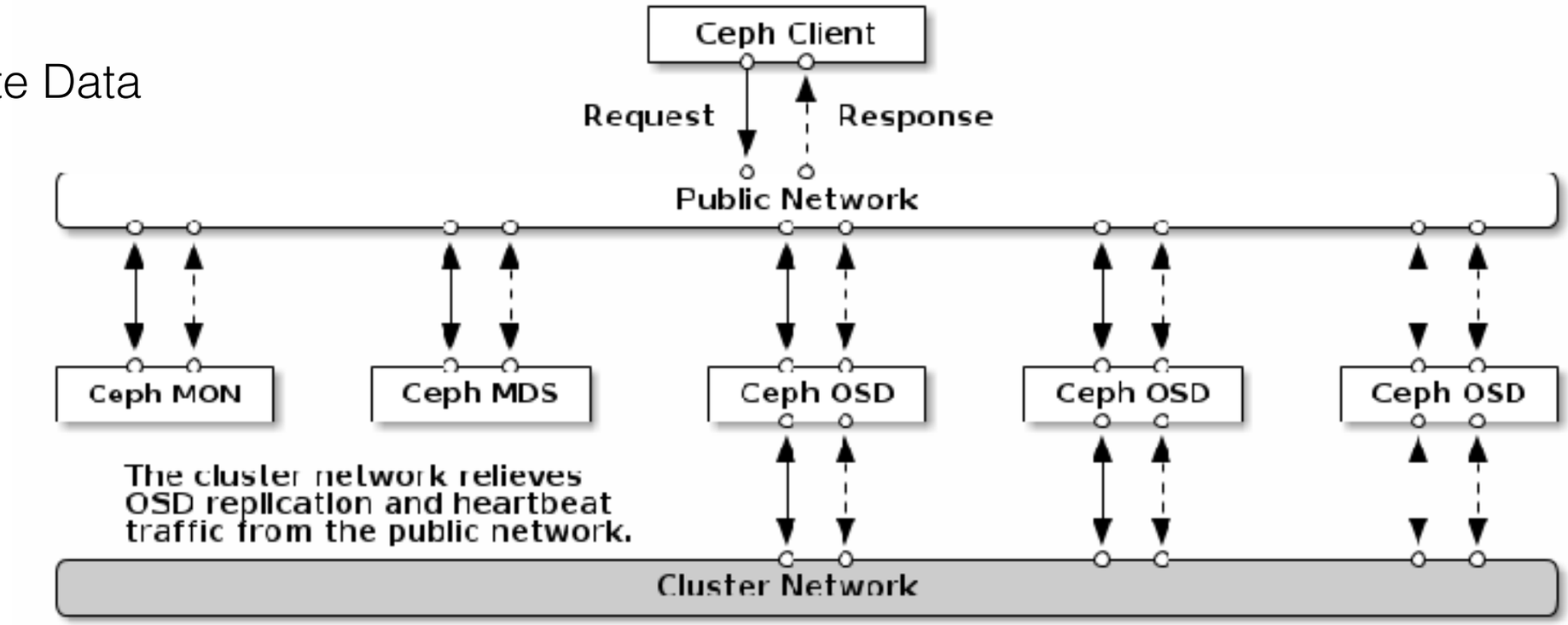


Pool


OBJECTS
OBJECTS
OBJECTS
POOL A
10 11 10 01
01 01 01 10
POOL B
01 10 11 10
10 01 01 01
POOL C
10 01 10 11
POOL D
01 10 01 01
11 10 01 10
01 01 10 01
POOLS
10 01
01 11
01 01
10 01
01 11
10 10
10 10
01 01
CLUSTER

Network Architecture

- **Public network**
  - Recommend MTU 9000
  - Uses for Read/Write Data
- **Cluster Network**
  - MTU <= 5000
  - Replication data
  - Cluster monitoring



Hardware Planning

1TB

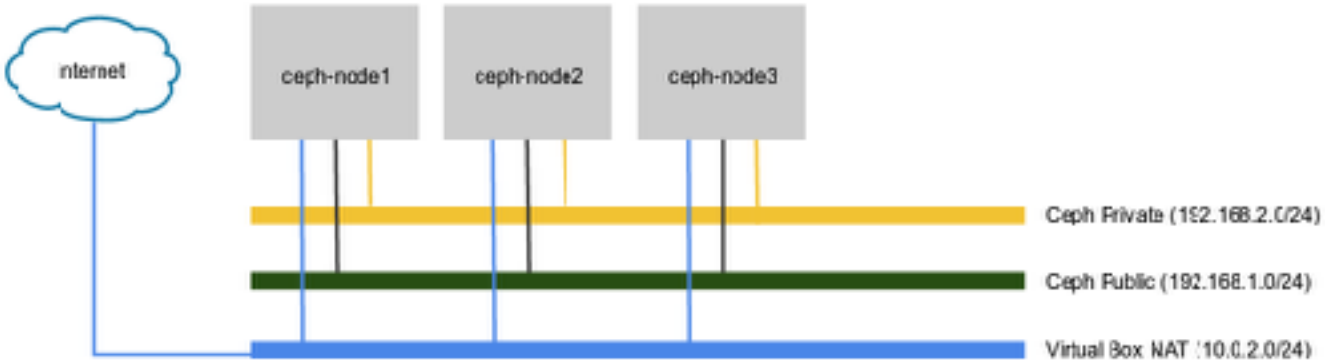
- 1GB Memory RAM (Metadata)
- 500MB Memory RAM (No Metadata)
- 0.5 Core CPUs
- Hard disk drives
  - Journal: **SSD**
  - OSD: SSD/Sata/SAS
- Network
  - at least 1Gbps for public
  - at least 1Gbps for data replication

Q&A

http://www.slideshare.net/openstackindia/ceph-openstackjun2015meetup
http://image.slidesharecdn.com/inktankdellopenstacksummitpreso-2013-04-12-130424120346-phpapp02/95/wicked-easy-ceph-block-storage-openstack-deployment-with-crowbar-9-638.jpg?cb=1366805568
http://www.slideshare.net/dalgaaf/linux-stammtisch-munich-ceph-overview-experiences-and-outlook

Ceph Ansible Workshop #1

Mr. Charnsilp Chinprasert



Host	Role	Network	Interface	Network Address
ceph-node1	Ceph Client + Ceph MON + Ceph OSD	Virtualbox NAT	eth0	DHCP
		Ceph public network	eth1	192.168.1.101
		Ceph private network	eth2	192.168.2.101
ceph-node2	Ceph MON + Ceph OSD	Virtualbox NAT	eth0	DHCP
		Ceph public network	eth1	192.168.1.102
		Ceph private network	eth2	192.168.2.103
ceph-node3	Ceph MON + Ceph OSD	Virtualbox NAT	eth0	DHCP
		Ceph public network	eth1	192.168.1.103
		Ceph private network	eth2	192.168.2.103

Virtual Machine Hardware Configuration

Ceph Node 1

Virtual Hardware	Specification
CPU	1 CPU
Memory	1300 MiB
Disk	3 x 10 G (VDI) 1 x 20 G (VDI)
Network	3 X 1 Gib Network

Ceph Node 2

Virtual Hardware	Specification
CPU	1 CPU
Memory	1024 MiB
Disk	3 x 10 G (VDI) 1 x 20 G (VDI)
Network	3 X 1 Gib Network

Ceph Node 3

Virtual Hardware	Specification
CPU	1 CPU
Memory	1024 MiB
Disk	3 x 10 G (VDI) 1 x 20 G (VDI)
Network	3 X 1 Gib Network

เข้าใช้งานเครื่อง Ceph node 1 เพื่อรัน Ansible

เราจะติดตั้ง Ceph โดยใช้ Ceph node 1 เป็นเครื่องที่รัน Ansible ดังนั้น ให้เรา SSH ไปยังเครื่อง Ceph node 1 ครับ

โดย VM ที่เตรียมให้ใช้

User: ubuntu

Password: Ubuntu

1) Linux/OSX ใช้คำสั่ง

```
$ ssh ubuntu@192.168.1.101
```

2) Windows ให้โหลด โปรแกรม putty มาใช้งานครับ

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Deploy Ceph Cluster with Ceph-Ansible

ตรวจสอบ host file ตรวจสอบทั้ง 3 เครื่อง

```
$ sudo vim /etc/hosts
```

ให้เป็นดังนี้

```
127.0.0.1       localhost
192.168.1.101   ceph-node1
192.168.1.102   ceph-node2
192.168.1.103   ceph-node3

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Prepare Ansible and Source file ให้ทำที่ ceph-node1

1) ติดตั้ง PPA ของ Ansible เพื่อให้สามารถติดตั้ง version ใหม่ได้

```
$ sudo add-apt-repository -y ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install -y ansible
```

2) ตรวจสอบ version

```
ansible 2.1.0.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = Default w/o overrides
```

3) ทำการส่ง pubilc key จาก client ไปยัง OSD+MON Server

ให้ทำทีเครื่องceph-client

```
$ ssh-copy-id ceph-node1
$ ssh-copy-id ceph-node2
$ ssh-copy-id ceph-node3
```

4) ทำการ clone source code ของ ceph-ansible เพื่อเตรียมพร้อมสำหรับการพัฒนา

```
$ git clone https://github.com/ceph/ceph-ansible.git
```

Component Versions

Component	Details
Host operating system	Ubuntu 14.04.5 LTS
Kernel	4.4.0-36-generic
Ansible	ansible 2.1.1.0
Ceph Ansible	Branch: master (19b3caef2ba2f894e0f147a645cb6597b619a9b5)
Ceph	ceph version 10.2.2 (Jewel)

Configure inventory, variable and playbook file

1) สร้างdirectory สำหรับการ deploy ceph

```
$ mkdir workspaces && cd  workspaces
```

2) สร้าง environment สำหรับการ deploy

```
$ mkdir dev-1 && cd dev-1
```

3) สร้างdirectory สำหรับการเก็บ variable

```
$ mkdir group_vars host_vars
```

4) สร้าง configuration ของ ansible ที่ใช้สำหรับ

```
$ vim ansible.cfg
```

เพิ่มเป็นดังนี้

```
[defaults]
```

```
roles_path = ../../roles
action_plugins = ../../plugins/actions
inventory = inventories
remote_port = 22
remote_user = ubuntu
host_key_checking = False
deprecation_warnings = False
display_skipped_hosts = False
```

5) สร้าง playbook ที่ใช้สำหรับการติดตั้ง cluster (client + osds + mons)

```
$ vim base.yml
```

แก้ไขเป็นดังนี้

```
---

- hosts: mons
  become: True
  roles:
  - ceph-mon

- hosts: osds
  become: True
  roles:
  - ceph-osd

- hosts: clients
  become: True
  roles:
  - ceph-client
```

6) สร้าง playbook ที่ใช้สำหรับการติดตั้งเฉพาะ ceph clients

```
$ vim clients-configure.yml
```

แก้ไขเป็นดังนี้

```
---
- hosts: mons
  become: True
  roles:
  - ceph-fetch-keys

- hosts: clients
  become: True
  roles:
  - ceph-client
```

7) สร้าง playbook ที่ใช้สำหรับการติดตั้งเฉพาะ ceph osds

```
$ vim osd-configure.yml
```

แก้ไขเป็นดังนี้

```
---
- hosts: mons
  become: True
  roles:
  - ceph-fetch-keys

- hosts: osd01
  become: True
  roles:
  - ceph-osd

- hosts: osd02
  become: True
  roles:
  - ceph-osd

- hosts: osd03
  become: True
```

```
roles:
- ceph-osd
```

8) สร้าง inventories สำหรับการเลือก host ที่จะ deploy

```
$ vim inventories
```

```
[mons]
mons01 ansible_host=ceph-node1
mons02 ansible_host=ceph-node2
mons03 ansible_host=ceph-node3

[osds]
osd01 ansible_host=ceph-node1
osd02 ansible_host=ceph-node2
osd03 ansible_host=ceph-node3

[clients]
admin ansible_host=ceph-node1
```

9) กำหนดตัวแปรให้กับ ceph ทั้ง cluster

```
$ vim group_vars/all
```

ให้เพิ่มดังต่อไปนี้

```
---
dummy:
############
# GENERAL #
############

fetch_directory: fetch/
cluster: ceph
fsid: 9848b7d5-3933-4ae9-8f8b-c775929e744d
#generate_fsid: true

############
# INSTALL #
############

#ชื่ออ้างอิงที่ใช้อ้างอิงใน inventory
mon_group_name: mons
osd_group_name: osds
client_group_name: clients
ceph_origin: 'upstream'

##########
# VERSION
##########

ceph_stable: true
ceph_stable_key: http://mirror.tek-me.com/ceph/keys/release.asc
ceph_stable_release: jewel
ceph_stable_repo: http://mirror.tek-me.com/ceph/debian-jewel

######################
# CEPH CONFIGURATION #
```

```
######################
monitor_interface: eth1
monitor_address: 0.0.0.0
journal_size: 2048
public_network: 192.168.1.0/24
cluster_network: 192.168.2.0/24

ceph_conf_overrides:
 global:
   rbd_default_features: 3
```

10) กำหนดตัวแปรให้กับ ceph mon

```
$ vim group_vars/mons
```

```
---
dummy:
############
# GENERAL #
############
fetch_directory: fetch/
cephx: true
```

11) กำหนดตัวแปรให้กับ ceph osds

```
$ vim group_vars/osds
```

```
---
dummy:
############
# GENERAL #
############
fetch_directory: fetch/
##############
# CEPH OPTIONS
##############
cephx: true
##############
# Disable Default Scenario
##############
osd_auto_discovery: false
osd_directory: false
journal_collocation: false
bluestore: false

##############
```

```
# Enable N JOURNAL N OSD Scenario
##############
raw_multi_journal: true
```

12) ตรวจสอบดูแต่ละเครื่องว่ามี Disk ครบถ้วน

```
ubuntu@ceph-node1:~$ ssh ceph-node1 'sudo lsblk'
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    10G  0 disk
 └─sda1   8:1    0    10G  0 part /
sdb      8:16   0    20G  0 disk
sdc      8:32   0    10G  0 disk
sdd      8:48   0    10G  0 disk
sde      8:64   0    10G  0 disk
sr0     11:0    1  1024M  0 rom
```

```
ubuntu@ceph-node1:~$ ssh ceph-node2 'sudo lsblk'
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    10G  0 disk
 └─sda1   8:1    0    10G  0 part /
sdb      8:16   0    20G  0 disk
sdc      8:32   0    10G  0 disk
sdd      8:48   0    10G  0 disk
sde      8:64   0    10G  0 disk
sr0     11:0    1  1024M  0 rom
```

```
ubuntu@ceph-node1:~$ ssh ceph-node3 'sudo lsblk'
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    10G  0 disk
 └─sda1   8:1    0    10G  0 part /
sdb      8:16   0    20G  0 disk
sdc      8:32   0    10G  0 disk
sdd      8:48   0    10G  0 disk
sde      8:64   0    10G  0 disk
sr0     11:0    1  1024M  0 rom
```

13) กำหนดตัวแปรให้กับ ceph osds ในแต่ล่ะ host

```
$ vim host_vars/osd01
```

```
---


devices:
  - /dev/sdc
  - /dev/sdd
  - /dev/sde

raw_journal_devices:              #N JOURNAL DEVICES FOR N OSDS
  - /dev/sdb
  - /dev/sdb
  - /dev/sdb
```

```
$ vim host_vars/osd02
```

```
---

devices:
  - /dev/sdc
  - /dev/sdd
  - /dev/sde

raw_journal_devices:          #N JOURNAL DEVICES FOR N OSDS
  - /dev/sdb
  - /dev/sdb
  - /dev/sdb
```

```
$ vim host_vars/osd03
```

```
---

devices:
  - /dev/sdc
  - /dev/sdd
  - /dev/sde

raw_journal_devices:          #N JOURNAL DEVICES FOR N OSDS
  - /dev/sdb
  - /dev/sdb
  - /dev/sdb
```

14) ทดสอบการ remote

```
$ ansible all -i inventories -m ping
```

Install Ceph Cluster by Ansible

1) เริ่มติดตั้ง Ceph Ansible

```
$ ansible-playbook base.yml
```

เมื่อติดตั้งเสร็จผลลัพธ์จะเป็นดังนี้

```
...
PLAY RECAP *********************************************************************
admin                      : ok=42   changed=0    unreachable=0    failed=0
mons01                     : ok=55   changed=14   unreachable=0    failed=0
mons02                     : ok=53   changed=12   unreachable=0    failed=0
mons03                     : ok=53   changed=12   unreachable=0    failed=0
osd01                      : ok=62   changed=1    unreachable=0    failed=0
osd02                      : ok=61   changed=1    unreachable=0    failed=0
osd03                      : ok=61   changed=1    unreachable=0    failed=0
```

2) ตรวจสอบ Ceph และตั้งค่า pg ให้เป็น 128

```
$ sudo ceph -s
    cluster 9848b7d5-3933-4ae9-8f8b-c775929e744d
     health HEALTH_WARN
            too few PGs per OSD (21 < min 30)
     monmap e1: 3 mons at {ceph-node1=192.168.1.101:6789/0,ceph-node2=192.168.1.102:6789/0,ceph-
node3=192.168.1.103:6789/0}
            election epoch 4, quorum 0,1,2 ceph-node1,ceph-node2,ceph-node3
     osdmap e39: 9 osds: 9 up, 9 in
            flags sortbitwise
      pgmap v76: 64 pgs, 1 pools, 0 bytes data, 0 objects
            305 MB used, 91755 MB / 92060 MB avail
                  64 active+clean
```

```
$ sudo ceph osd pool set rbd pg_num 128
set pool 0 pg_num to 128
```

```
$ sudo ceph osd pool set rbd pgp_num 128
set pool 0 pgp_num to 128
```

3) ตรวจสอบ ว่า Ceph OK แล้ว

```
$ sudo ceph -s
    cluster 9848b7d5-3933-4ae9-8f8b-c775929e744d
     health HEALTH_OK
     monmap e1: 3 mons at {ceph-node1=192.168.1.101:6789/0,ceph-node2=192.168.1.102:6789/0,ceph-
node3=192.168.1.103:6789/0}
            election epoch 4, quorum 0,1,2 ceph-node1,ceph-node2,ceph-node3
     osdmap e50: 9 osds: 9 up, 9 in
            flags sortbitwise
      pgmap v132: 128 pgs, 1 pools, 0 bytes data, 0 objects
            308 MB used, 91752 MB / 92060 MB avail
                  128 active+clean
```

Create a Ceph User and Mount it

1) ทดสอบ เชื่อต่อและใช้งาน Ceph เริ่มจากการสร้าง User

```
$ sudo su -
```

```
# ceph auth get-or-create client.rbd mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=rbd'
```

```
# ceph auth get-or-create client.rbd | sudo tee /etc/ceph/ceph.client.rbd.keyring
```

```
[client.rbd]
        key = AQDthtpXowj7HhAARz0VtV4yi5iEOmbL1MoDjA==
```

2) หลังจากสร้าง user แล้วเราจะได้ไฟล์ keyring มาด้านในจะมี key เพื่อใช้ในการ authen, ทดสอบการใช้งาน ceph ด้วย client ที่สร้าง

```
# ceph -s --name client.rbd
```

```
    cluster 9848b7d5-3933-4ae9-8f8b-c775929e744d
     health HEALTH_OK
     monmap e1: 3 mons at {ceph-node1=192.168.1.101:6789/0,ceph-node2=192.168.1.102:6789/0,ceph-node3=192.168.1.103:6789/0}
            election epoch 4, quorum 0,1,2 ceph-node1,ceph-node2,ceph-node3
     osdmap e50: 9 osds: 9 up, 9 in
            flags sortbitwise
      pgmap v132: 128 pgs, 1 pools, 0 bytes data, 0 objects
            308 MB used, 91752 MB / 92060 MB avail
                 128 active+clean
```

3) สร้าง RDB Block ใหม่ขึ้นมาขนาด 10GB โดยใช้ Client ที่สร้าง

```
# rbd create rbd1 --size 10240 --name client.rbd
```

4) ตรวจสอบ RBD Block ที่สร้างขึ้นมาด้วยคำสั่ง ls และ info

```
# rbd ls -p rbd --name client.rbd
```

```
# rbd --image rbd1 info --name client.rbd
```

```
rbd image 'rbd1':
        size 10240 MB in 2560 objects
        order 22 (4096 kB objects)
        block_name_prefix: rbd_data.1055238e1f29
        format: 2
        features: layering
        flags:
```

5) ทำการ map RBD Block เข้ากับเครื่องเรา โดยที่เราจะมองเห็นเป็นเหมือน Disk ลูกหนึ่ง

```
# rbd map --image rbd1 --name client.rbd

id pool image snap device
0  rbd  rbd1  -    /dev/rbd0
```

6) ใช้ fdisk เพื่อดู Disk ที่เรา map มา

```
# fdisk -l /dev/rbd0

Disk /dev/rbd0: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 4194304 bytes / 4194304 bytes
Disk identifier: 0x00000000
```

7) สร้าง directory และ สั่ง mount

```
# mkdir /mnt/ceph-block01
# mount /dev/rbd0 /mnt/ceph-block01
```

8) ทดสอบการ เขียนไฟล์

```
# cd /mnt/ceph-block01/
# touch test.txt
```