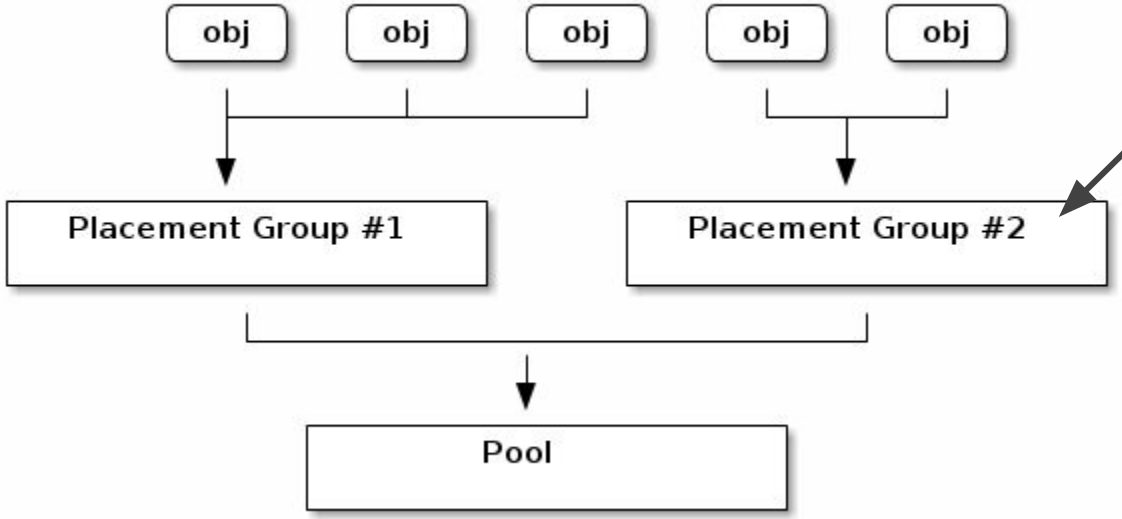
Ceph PGs

Jing, mqjing@gmail.com

Placement Group, PG

- Without PGs, it will be difficult to manage and track tens of millions of objects that are replicated and spread over hundreds of OSD
- Using PGs
  - Reduce computational penalty
  - Increasing PG number will reduce the per OSD load
- Each PG requires system resources, CPU, and memory

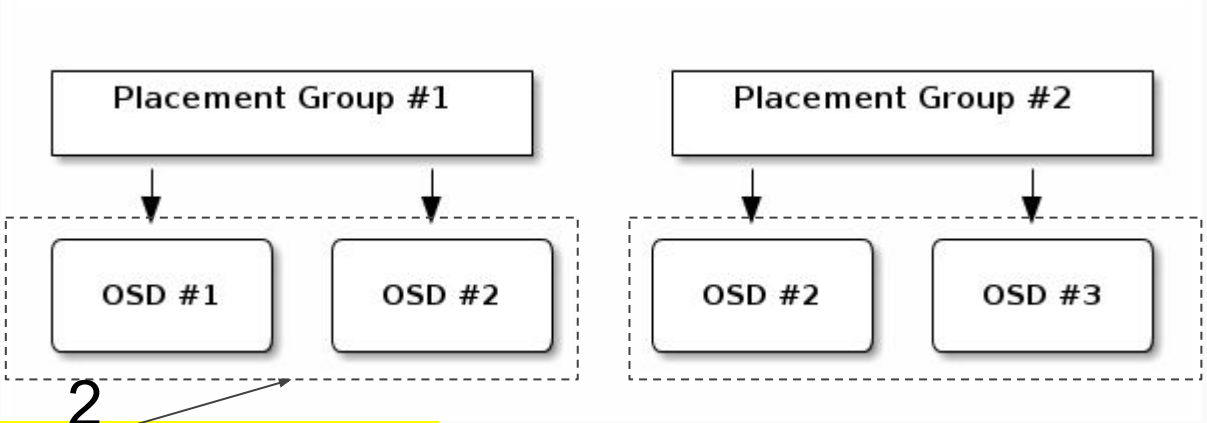
How PGs are Used?



A placement group (PG) aggregates objects within a pool

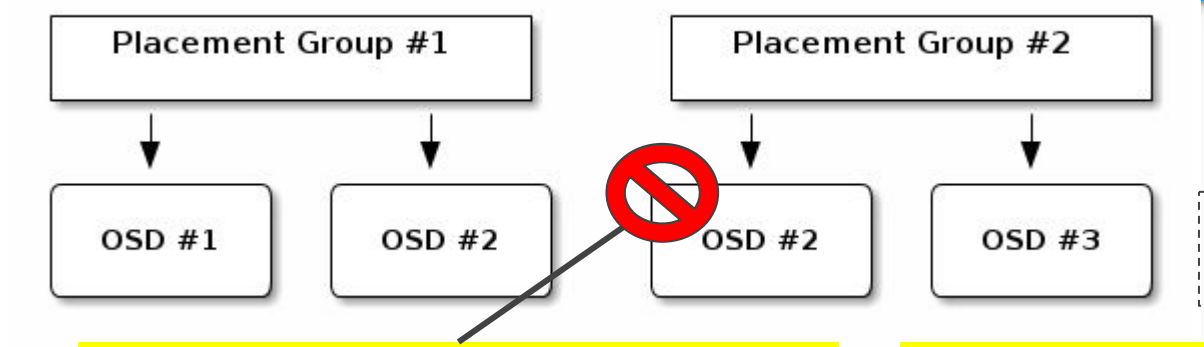
Usage: Tracking object placement and object metadata.

The object's contents within a PG are stored in a set of OSDs.



For instance, in a **replicated 2** pool, each PG will store objects on 2 OSDs

Should OSD#2 fail



When OSD # 2 fail

1. Another will be **assigned to PG #1**
2. Will be filled with copies of all objects in OSD #1

When Pool Size Changed: 2 --> 3

1. An additional OSD will be assigned to the PG
2. Will receive copies of all objects in the PG

- PG does not own the OSD, they share it with other PGs from the same/other pools
- When PG number increases,
  - The new PGs will be assigned OSDs
  - The CRUSH function will change and some objects move some objects from former PGs will be copied to the new PGs and remove the old one

Data Durability Issue

Cascading failures leading to the permanent loss of a Placement Group

The OSD fails and all copies of the object it contains are lost. For all objects within the **PG** the number of replica suddenly drops from **3 to 2**.

Ceph starts recovery for this **PG** by choosing a new OSD to re-create the third copy of all objects.

Another OSD, within the same placement group, fails before the new OSD is fully populated with the third copy.

Some objects will then only have one surviving copies.

Ceph picks yet another OSD and keeps copying objects to restore the desired number of copies.

A third OSD, within the same placement group, fails before recovery is complete.

If this OSD contained the only remaining copy of an object, it is permanently lost

Example

10 OSD Cluster

A cluster containing 10 OSDs with 512 PGs in a 3 replica pool

- CRUSH will give each PG 3 OSDs
- In the end, each OSDs will hosting (512 * 3) / 10 = ~150 PGs

1 OSD fail

- When the first OSD fails, the above scenario will therefore start recovery for all 150 PGs at the same time.
- The 150 PGs being recovered are likely to be homogeneously spread over the 9 remaining OSDs.

20 OSD Cluster

A cluster containing (10 ->20) OSDs with 512 PGs in a 3 replica pool

+ each OSD is hosted by a 1TB

- CRUSH will give each PG 3 OSDs
- In the end, each OSDs will hosting (512 * 3) / 20 = (~150 -> ~75) PGs

+ 1 OSD crashed
+ 19 OSD to do backfill operation

20 OSD Cluster:
1 OSD = 1TB (failure)
9 OSD to do backfill operation

- Copy size: 10 OSDs cluster had to copy ~ 100GB each, they now have to copy 50GB each instead.
- + Recovery will happen twice as fast
  + ==> In other words, recovery goes faster when the number of OSDs increases.

40 OSD Cluster

A cluster containing (10 -> 20 -> 40) OSDs with 512 PGs in a 3 replica pool

+ each OSD is hosted by a 1TB

- CRUSH will give each PG 3 OSDs
- In the end, each OSDs will hosting (512 * 3) / 40 = ~35 PGs

+ 1 OSD crashed
+ 39 OSD to do backfill operation

20 OSD Cluster:
1 OSD = 1TB (failure)
9 OSD to do backfill operation

- Copy size: 10 OSDs cluster had to copy ~ 100GB each, they now have to copy 25GB each instead.
- + Recovery will happen twice as fast
  + ==> In other words, recovery goes faster when the number of OSDs increases.

200 OSD Cluster

A cluster containing 200 OSDs with 512 PGs in a 3 replica pool

+ each OSD is hosted by a 1TB

- CRUSH will give each PG 3 OSDs
- In the end, each OSDs will hosting (512 * 3) / 200 = 7 PGs

+ 1 OSD crashed
+ 7 * 3 = 21 OSD to do backfill operation (pg * replica)

20 OSD Cluster:
1 OSD = 1TB (failure)
9 OSD to do backfill operation

- Copy size: 10 OSDs cluster had to copy ~ 100GB each, they now have to copy 47GB each instead.

1T / 9 = 999GB

1T / 21 = 47GB

+ Recovery will longer than 40 OSD
+ ==> In other words, recovery goes faster when the number of OSDs increases. --><--

+ Adjust PG_Num

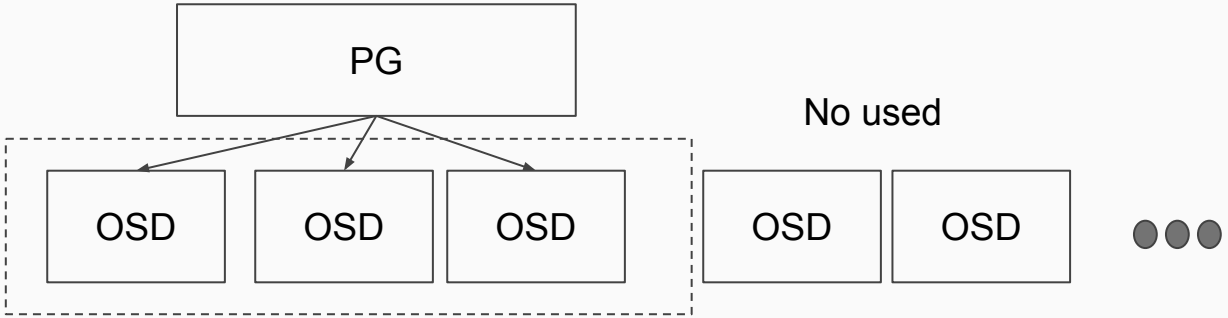
- In a nutshell, more OSDs mean faster recovery and a lower risk of cascading failures leading to the permanent loss of a PG.

Object Distribution Issue

if there was single a PG for 10 OSDs in a 3 replica pool

==> only three OSD would be used because CRUSH would have no other choice.

When more PGs are available, objects are more likely to be evenly spread among them.



Choosing the number of PGs

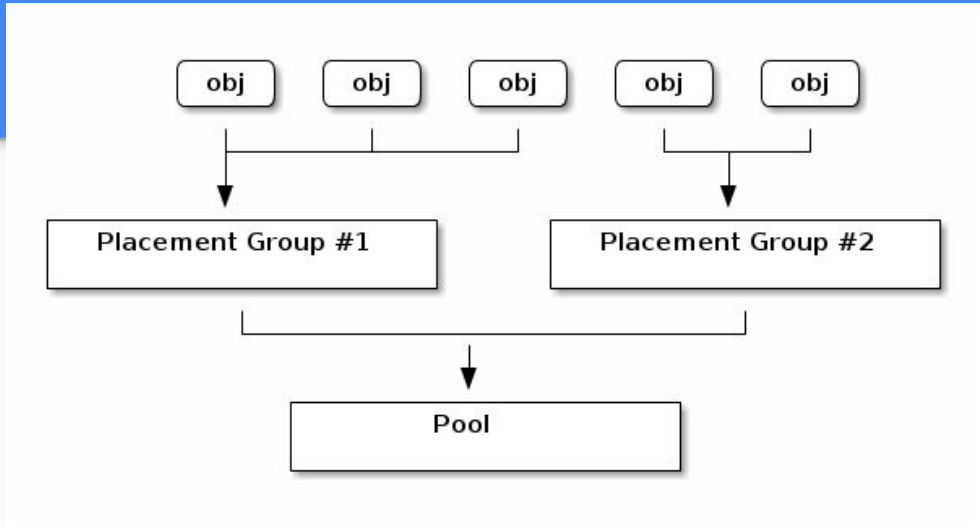
If you have **more than** 50 OSDs, we recommend approximately 50-100 PGs per OSD to balance out resource usage, data durability and distribution.

$$\text{Total PGs} = \frac{(\text{OSDs} * 100)}{\text{pool size}}$$

If you have **less than** 50 OSDs, choosing among the ***preselection*** (next page) is best

Choose PG NUMBER

- < 5 OSDs, set pg_num = 128
- 5 ~ 10 OSDs, set pg_num = 512
- 10 ~ 50 OSDs, set pg_num = 4096



ceph osd pool set {pool-name} pg_num

Example

A cluster containing 160 OSDs in a 3 replica pool

$$\text{Total PGs} = \frac{(\text{OSDs} * 100)}{\text{pool size}}$$

$$\frac{(160 * 100)}{3} = 5333.3333 \sim 8192 \text{ PGs}$$

It's important to balance the number of PGs per pool with the number of PGs per OSD (1. Balanced the PG # per pool 2. Balanced the PG # per OSD)

如何根據現狀調整 PG and PGP?

經過長期操作後, (1) OSD 可能會增加或減少, (2) Pool 的數量可能會增加或減少
如何的根據 cluster 的現狀, 調整 PG and PGP number?

- PGP is the PG for Placement purpose, which should be kept equal to the total number of PGs

- Step 1: Check the existing PG and PGP number
  - ceph osd pool get data-pool pg_num
  - ceph osd pool get data-pool pgp_num

- Step 2: Check 目前 Cluster 的各項重要參數 {OSD number, replication pool size, Pool count}
  - ceph osd dump | grep size

- Total OSD number = 9, replication pool size = 2, Pool count =3

$$\text{Total PGs} = \frac{(\text{OSDs} * 100)}{\text{pool size}}$$

$$\frac{(9 * 100)}{2} = 450$$

Total PGs per Pool = Total PGs / pool count = 450/3 = 150 ~ 256

- Step 3: Set the PG and PGP number for all other pool

Update data-pool

  - ceph osd pool set data-pool pg_num 256
  - ceph osd pool set data-pool pgp_num 256

Monitoring OSDs

- An OSD's status
  - **in**: in the cluster
  - **out**: out of the cluster
  - **up**: up and running
  - **down**: not running

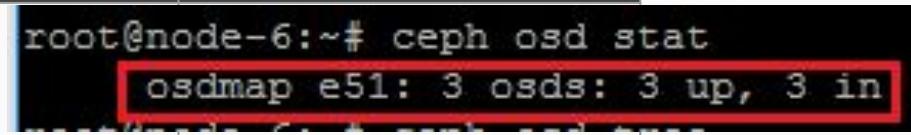
Previous	Currently	Behavior
in	out	Ceph will **migrate PGs** to other OSDs
in/out	out	Ceph will **not assign PGs** to the OSD

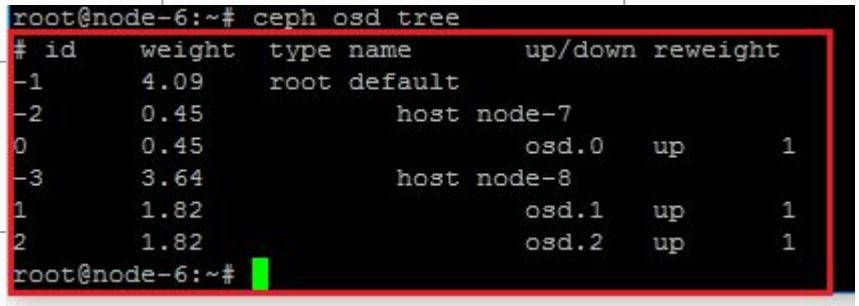
Ceph is NOT Healthy_OK

- You haven't started the cluster yet
- The OSDs are in the peering when you just started/restart the cluster
- You just added/removed an OSD
- You just have modified your cluster map

OSD Status Check

Command	Description	Note
ceph osd stat	• Check how many OSDs • Check how many are up • Check how many are in	
ceph osd tree	• Identify the ceph-osd daemons that aren't running	
sudo /etc/init.d/ceph -a start osd.1	• Start an OSD, if it down	





PG Sets

- If a POOL requires 3 replicas of a PG, CRUSH may assign them to osd.1, osd.2, osd.3
- CRUSH seeks a pseudo-random placement that will take into account failure domains you set in CRUSH MAP
- Acting Set
  - As a set of OSDs that contain the replicas of a PG

When A OSD in the Acting Set is down

Action	Ceph Behavior	Note
When You added or removed an OSD	Ceph reassigned the PGs to other OSDs ==> Changing the composition of the Acting Set ==> Spawning the migration of data with "backfill" process	
When OSD was down, was restarted	Ceph is now recovering	
An OSD in the Acting Set is down or unable to service requests	Another OSD has temporarily assumed it duties	

Up Set

- **Up Set**: the set of OSDs that will actually handle the requests
- The **Up Set** and **Acting Set** are virtually identical

When	Ceph Behavior	Note
Up Set != Acting Set	• Ceph is migrating Data • An OSD is recovering • The cluster is rebalancing itself • Problem: HEALTHY WARN with "stuck state"	

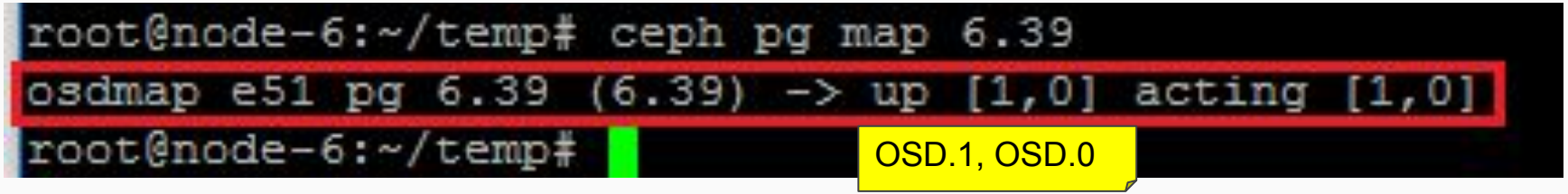
Check PG Status

PG ID Format:
{pool-num}.{pg-id}

Ex:

6.39 ==> pool 6, 39 id

Situation	Command	Note
List of **PG**	• ceph pg dump	
To view which OSDs are within the **Acting Set** or the **Up Set**	• ceph pg map {pg-num}	



Point

Peering

- **寫入 data 之前**, 這些PG 的狀態必須是 Active + Clean, 所以必須讓這個 PG 內的所有 OSD Daemons 都同意目前的狀態. 這個驗證同意的工作在 Ceph 的設計裡稱為 Peering
  - 作法: 就是讓 PG 中 Acting set 的第一個 OSD (primary OSD) 向 second OSD, 第三個 OSD 進行驗證 active + clean 狀態
- Peering **完畢後**, OSDs 也會回報狀態給 Monitor

Peering

When Ceph is **Peering** a placement group

- Ceph is bringing the OSDs that store the replicas of the PG into agreement about the state of the objects and metadata in the PG

簡單的說, 就是讓那些儲存 replica objects 的 OSDs, 都同意目前 primary OSD 裡面 pg 中 objects 和 metadata 的狀態

Peering: Establish Agreement of the PG status

Before you can write data to a PG, it must be in active state, and it should be in clean state

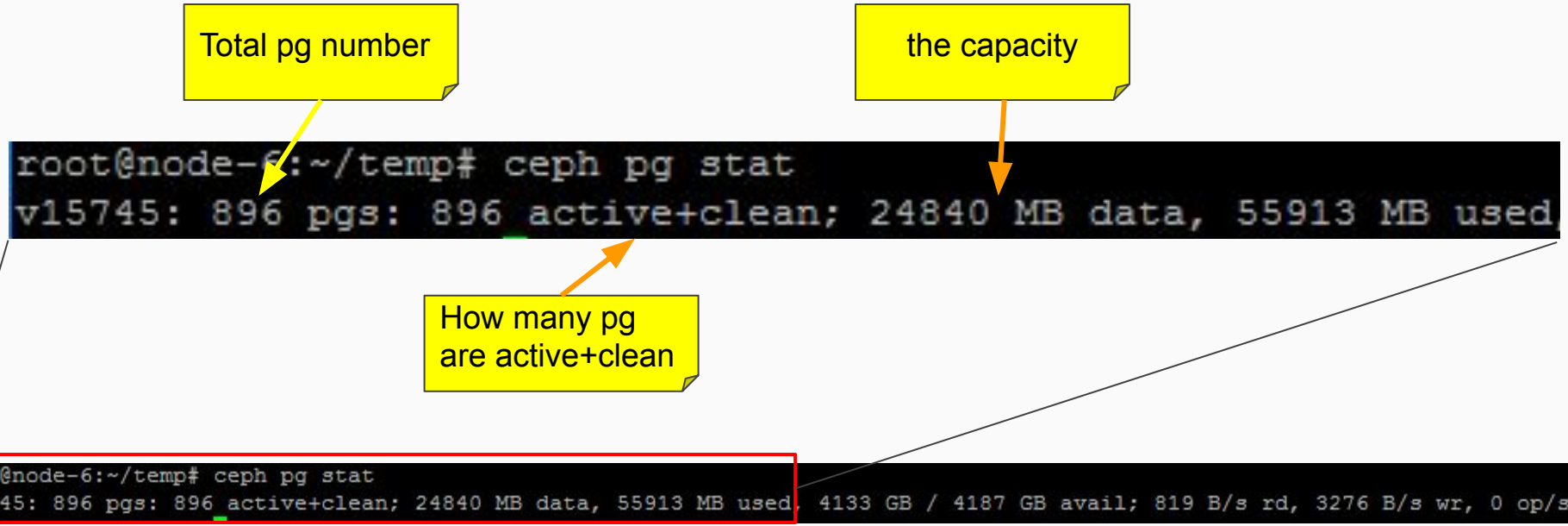
- For Ceph to determine the current state of a PG
  - The **primary OSD** of the PG (the first OSD in acting set), peers with the secondary and tertiary OSDs to establish agreement on the current state of the objects and metadata in the placement group
- The OSDs also report the status to the Monitor

Monitoring PG States

Ceph does NOT HEALTH_OK

When	Note
[**Pool**] Just create a pool and PG haven't peered yet	
[**Recovering**] The PG are recovering	
[**OSD**] Just added an OSD or removed an OSD	
[**CRUSHMap**] Just modified the CRUSH map and your PG are migrating	
[**Status error**] There is inconsistent data in different replicas of a PG	
Ceph is scrubbing a PG's replicas	

Check PG Stat



List Pool

ceph osd lspools

```
root@node-6:~/temp# ceph osd lspools
0 data,1 metadata,2 rbd,3 images,4 volumes,5 backups,6 .rgw.root,7 .rgw.control,8 compute,9 .rgw,10 .rgw.gc,11 .users.uid,12 .
gw.buckets.index,13 .rgw.buckets,
root@node-6:~/temp#
```

Total Pool = 14

PG IDs

Format: {pool-num}.{pg-id}

ex: 0.1f

pool-num = 0
pg-id = 1

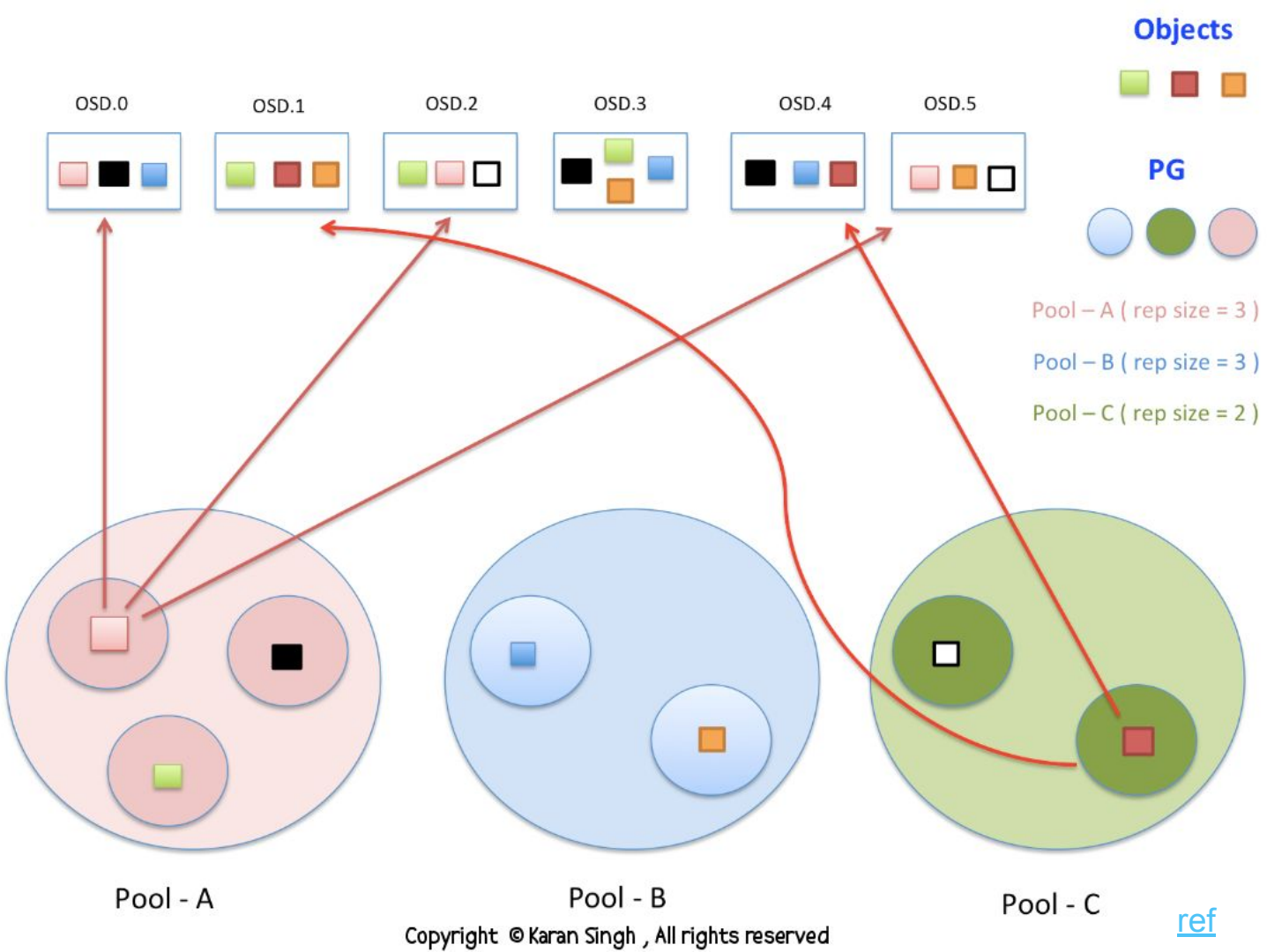
```
root@node-6:~/temp# ceph pg dump | more
dumped all in format plain
version 15985
stamp 2015-11-12 12:27:56.268109
last_osdmap_epoch 51
last_pg_scan 50
full_ratio 0.95
nearfull_ratio 0.85
pg_stat objects mip     degr    unf     bytes   log     disklog state   state_stamp     v       reported
ary     acting  acting_primary  last_scrub      scrub_stamp     last_deep_scrub deep_scrub_stamp
13.32   0       0       0       0       0       0       0       active+clean    2015-11-12 07:31:38.602648
2,0]    2       [2,0]   2       0'0     2015-11-12 07:31:33.542893      0'0     2015-11-12 07:31:33.542893
5.3a    0       0       0       0       0       0       0       active+clean    2015-11-12 09:29:57.996171
1,0]    1       [1,0]   1       0'0     2015-11-12 09:29:57.996144      0'0     2015-11-10 09:07:48.092730
```

The Output Format of the placement group

Format	Command
JSON format	ceph pg dump -o {filename} --format=json
Query PG	ceph pg {poolnum}.{pg-id} query

PG

If the Pool-A requires three replicas of a PG, CRUSH may assign them to osd.0, osd.2 and osd.5 respectively [ref]



Creating PG

When you create a pool, Ceph will create the number of placement groups you specified. Ceph will echo "creating" when it is creating PGs.

Step 1: Once PGs are created, the OSDs that are part of a PG's Acting Set will peer

Step 2: Once peering completed, the PG's status should be 'active + clean'

Ceph Client can begin write to the PG

Create A Pool

The total number of **PGs** for the pool. The default value 8 is NOT suitable for most systems.

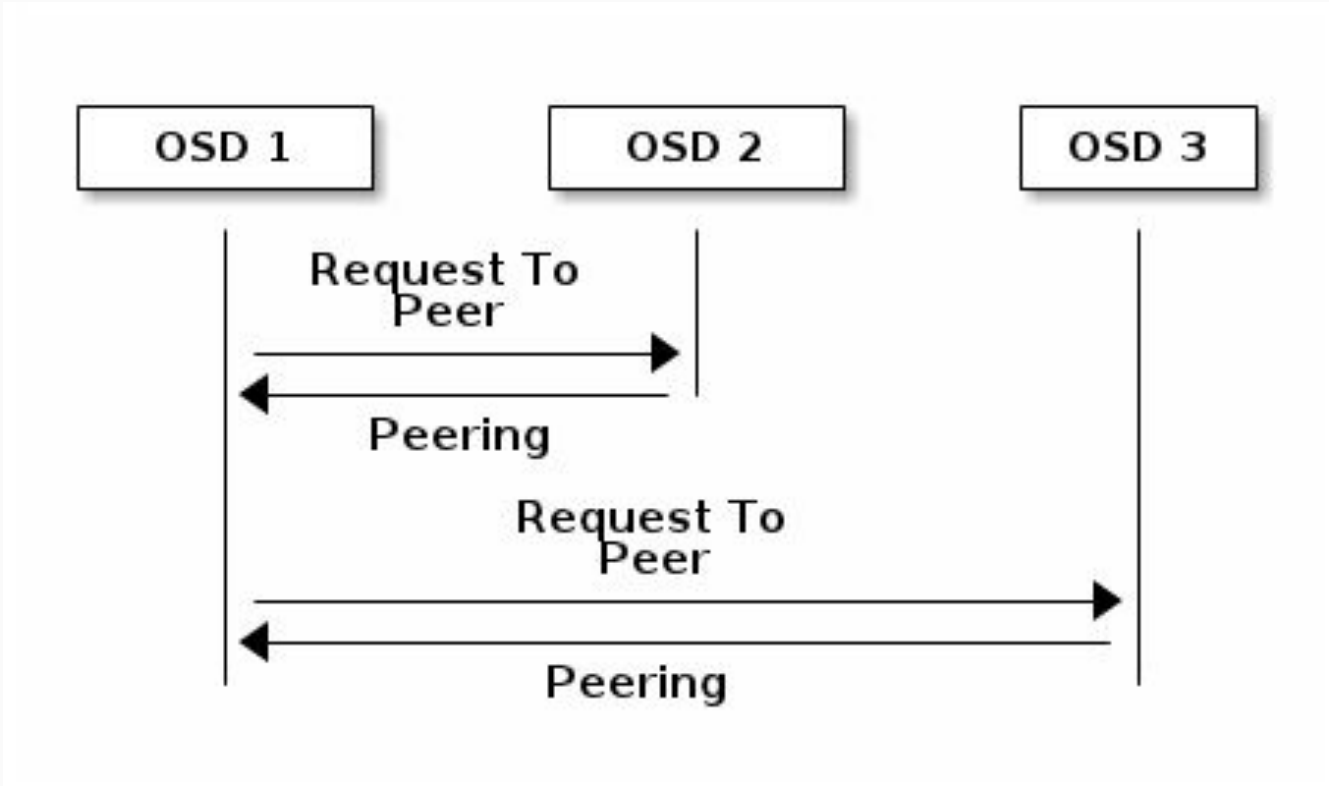
```
ceph osd pool create {pool-name} {pg-num} [{pgp-num}] [replicated] \
     [crush-ruleset-name]


ceph osd pool create {pool-name} {pg-num}  {pgp-num}   erasure \
     [erasure-code-profile] [crush-ruleset-name]
```

Peering

- When Ceph is Peering a placement group, Ceph is bring the OSDs that store the replicas of the placement group into agreement about the state of the object and metadata in the placement group

A Peering Process for a Pool with Replica 3



Active

在 pg 中的 data object 通常已經 **available** 在 primary 與 它的replica 中, 準備被讀寫

- When a **PG** is in the active state.
  - The data in the **PG** is generally available in the primary placement group and the replicas for read and write operations

Clean

若一個 pg 稱為 clean state
- 表示 primary OSD 與 replica OSDs 都**同意**了在 pg 中的 data objects與 metadata 狀態
- 這個 pg 面沒有 stray replicas

- When a **PG** is in the **clean state**,
  - The primary OSD and the replica OSDs have successfully peered
  - There are no stray replicas for the **PG**
  - Ceph replicated all objects in the **PG** the correct number of times

DEGRADED

- 當 replica OSD 還沒回報 replica object 已經成功寫入前, 這個 PG 就都會保持著 **degraded** 狀態

- A primary OSD writes the object to storage, the PG will remain in a degraded state until the primary OSD has received an acknowledgement from the replica OSDs that Ceph created the replica objects successfully

PG with {active + degraded}

例如資料 available for read/write 了，但是其中有些 replica objects 還不能存取

- An OSD may be active even though it doesn’t hold all of the objects yet
- If an OSD goes down, Ceph marks each **PG** assigned to the OSD as degraded

Recovering

- When an OSD goes down, its contents may fail behind the current state of other replicas in the PG
- Ceph was designed for fault-tolerance at a scale where hardware and software problems are ongoing

Backing Filling (1/2)

- When a new OSD joins the cluster, CRUSH will assign PG from OSDs in the cluster to the newly added OSD

State	Behavior
backfill_wait	a backfill operation is pending
backfill	a backfill operation is underway
backfill_too_full	a backfill operation was requested, but couldn't be completed due to insufficient storage capacity

Backing Filling (2/2)

Setup	Behavior	Default Value
osd max backfills	the maximum number of concurrent backfills to or from an OSD	10
osd backfill full ratio	Enables an OSD to refuse a backfill request if the OSD is approaching its its full ratio (85%, by default).	85%

all setting: ref

Remapped

- When the Acting Set that services a PG changes, the data migrates from the old acting set to the new acting set

Stale

- The ceph-osd daemons may also get into a stuck state where they aren't reporting statistics in a timely manner (e.g., a temporary network fault) while Ceph uses heartbeat to ensure the system are running
  - If the primary OSD of a placement group's acting set fails to report to the monitor or
  - If other OSDs have reported the primary OSDs down

==> the monitor will mark the placement group stale

Identifying Troubled PGs (1/2)

State	Behavior
Unclean	Placement groups contain objects that are not replicated the desired number of times. They should be recovering
InActive	Placement groups cannot process reads or writes because they are waiting for an OSD with the most up-to-date data to come back up.
Stale	Placement groups are in an unknown state, because the OSDs that host them have not reported to the monitor cluster in a while (configured by mon osd report timeout).

Identify Trouble PGs (2/2)

ceph pg dump_stuck [unclean | inactive | stale]

Finding An Object Location

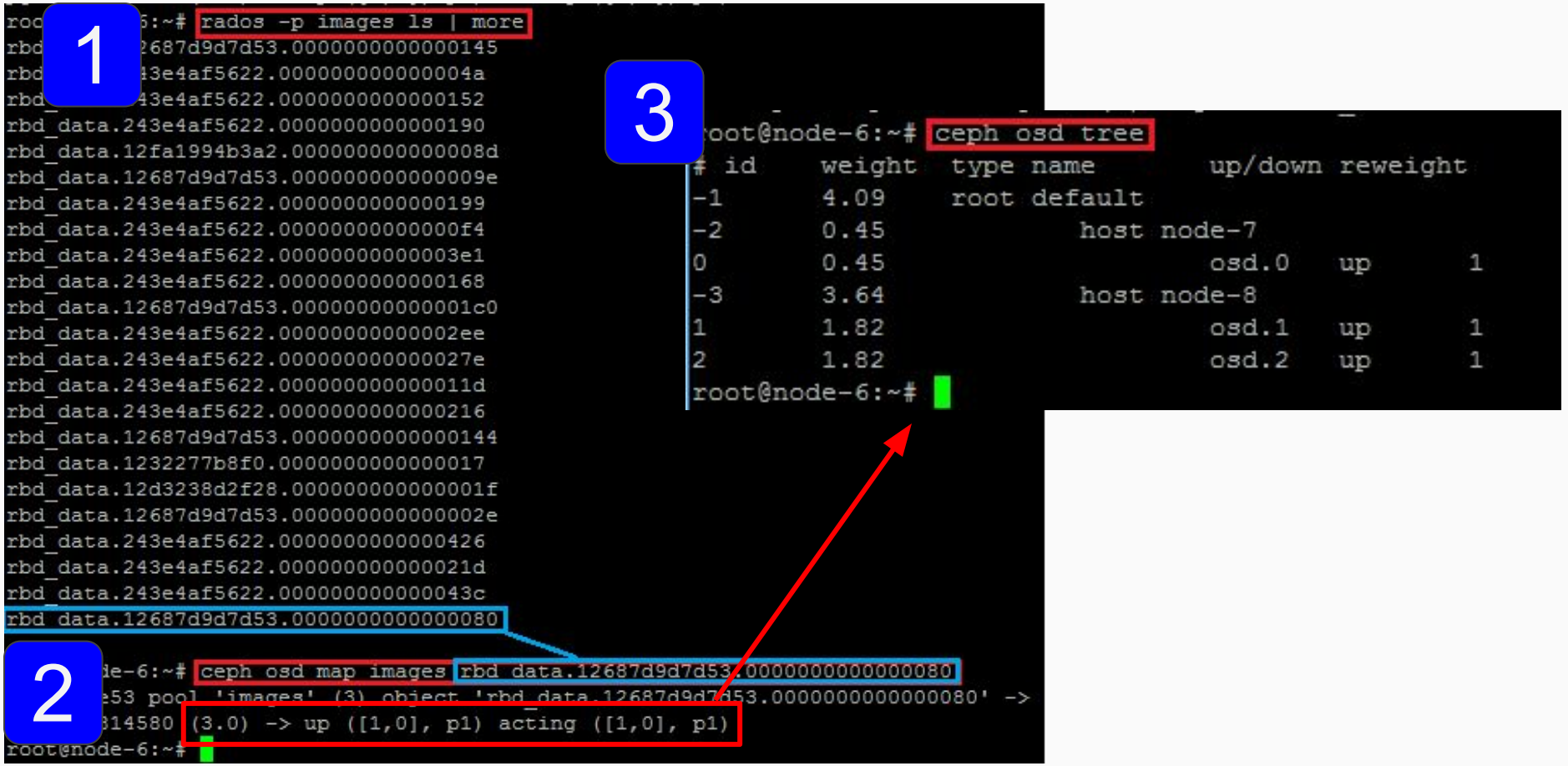
- To store object data in the Ceph Object Store, a Ceph client must:
  - Set an object name
  - Specify a pool

```
# list pools
ceph osd lspools

# list objects from a pool
rados -p {pool-name} ls

# identify the object location (OSDs)
ceph osd map {pool-name} {object-name}

# identify the object location (Hosts)
ceph osd tree
```


1
rados -p images ls | more
rbd_data.12687d9d7d53.0000000000000145
rbd_data.243e4af5622.000000000000004a
rbd_data.243e4af5622.0000000000000152
rbd_data.243e4af5622.0000000000000190
rbd_data.12fa1994b3a2.000000000000008d
rbd_data.12687d9d7d53.000000000000009e
rbd_data.243e4af5622.0000000000000199
rbd_data.243e4af5622.00000000000000f4
rbd_data.243e4af5622.00000000000003e1
rbd_data.243e4af5622.0000000000000168
rbd_data.12687d9d7d53.00000000000001c0
rbd_data.243e4af5622.00000000000002ee
rbd_data.243e4af5622.000000000000027e
rbd_data.243e4af5622.000000000000011d
rbd_data.243e4af5622.0000000000000216
rbd_data.12687d9d7d53.0000000000000144
rbd_data.1232277b8f0.0000000000000017
rbd_data.12d3238d2f28.000000000000001f
rbd_data.12687d9d7d53.000000000000002e
rbd_data.243e4af5622.0000000000000426
rbd_data.243e4af5622.000000000000021d
rbd_data.243e4af5622.000000000000043c
rbd_data.12687d9d7d53.0000000000000080
2
ceph osd map images rbd_data.12687d9d7d53.0000000000000080
pool 'images' (3) object 'rbd_data.12687d9d7d53.0000000000000080' ->
(3.0) -> up ([1,0], p1) acting ([1,0], p1)
3
ceph osd tree
id weight type name up/down reweight
-1 4.09 root default
-2 0.45 host node-7
0 0.45 osd.0 up 1
-3 3.64 host node-8
1 1.82 osd.1 up 1
2 1.82 osd.2 up 1

Reference

- http://docs.ceph.com/docs/v0.79/rados/operations/monitoring-osd-pg/
-