

从零开始学Ceph (Ceph From Scratch)



Ceph是目前非常流行的统一存储系统，所谓统一存储系统，就是通过Ceph集群同时提供块设备存储、对象存储以及文件系统服务。本书将从零开始，介绍Ceph的用法以及CRUSH、RADOS等底层技术。借助Ceph官方提...



下载手机APP
畅享精彩阅读

目 录

致谢

前言

前言

关于作者

致谢

Ceph简介

架构介绍

基础组件

Ceph命令

使用Ceph

更多介绍

Ceph用法

Ceph命令

Ceph容器

基本命令

RADOS操作

Bucket操作

Object操作

Ceph架构

论文介绍

CRUSH详解

CRUSH总结

RADOS详解

RBD

RBD命令

RBD快照

RBD克隆

RBD和Qemu

RBD和Virsh

RBD和OpenStack

Python librbd

RGW

RGW介绍

RGW用法

S3用法

Swift用法

CephFS

Ceph监控

Ceph-rest-api

Ceph-web

参与开发

贡献代码

Ceph-docker

结语

致谢

当前文档 《从零开始学Ceph (Ceph From Scratch) 》 由 进击的皇虫 使用 书栈网 (BookStack.CN) 进行构建, 生成于 2019-12-29。

书栈网仅提供文档编写、整理、归类等功能, 以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理, 书栈网难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候, 发现文档内容有不恰当的地方, 请向我们反馈, 让我们共同携手, 将知识准确、高效且有效地传递给每一个人。

同时, 如果您在日常工作、生活和学习中遇到有价值有营养的知识文档, 欢迎分享到书栈网, 为知识的传承献上您的一份力量!

如果当前文档生成时间太久, 请到书栈网获取最新的文档, 以跟上知识更新换代的步伐。

内容来源: [tobe https://github.com/tobegit3hub/ceph_from_scratch](https://github.com/tobegit3hub/ceph_from_scratch)

文档地址: http://www.bookstack.cn/books/ceph_from_scratch

书栈官网: <https://www.bookstack.cn>

书栈开源: <https://github.com/TruthHun>

分享, 让知识传承更久远! 感谢知识的创造者, 感谢知识的分享者, 也感谢每一位阅读到此处的读者, 因为我们都将成为知识的传承者。

Ceph From Scratch

Ceph是目前非常流行的统一存储系统，所谓统一存储系统，就是通过Ceph集群同时提供块设备存储、对象存储以及文件系统服务。



本书将从零开始，介绍Ceph的用法以及CRUSH、RADOS等底层技术。借助Ceph官方提供的容器，任何人都可以在本地体验此教程，学习分布式存储系统就像家庭作业一样简单。



前言

2006年Sage Weil发表了Ceph论文，启动一个伟大的开源项目，感谢作者。

2014年Redhat及其收购的Inktank在Ceph贡献全球第一，中国的UnitedStack全球第二也是国内第一，Ubuntu麒麟也有卓越贡献，感谢开发者。

还要感谢国内外关注Ceph的开发者以及运维人员！

本书使用GitBook编写，通过 `gitbook serve` 运行，也可以在本地运行Docker容器：

```
1. docker run -d -p 4000:4000 tobegit3hub/ceph_from_scratch
```



关于作者

[tobe](#)，前小米基础架构工程师，Docker 监控管理工具[seagull](#)和开源持续集成系统[ArchCI](#)作者，开源电子书[理解Linux进程](#)作者，现在UnitedStack负责[Docker](#)容器和[Ceph](#)存储。

致谢

1. `sudo docker run ubuntu echo "致谢读者们!"`

第一章 Ceph简介

本章将介绍Ceph的设计思路和基本组件，帮助你快速入门和了解Ceph。

阅读本章内容可以理解Ceph的架构设计、CRUSH、RADOS等概念，并且知道基于Ceph的RBD、RGW和CephFS等服务。

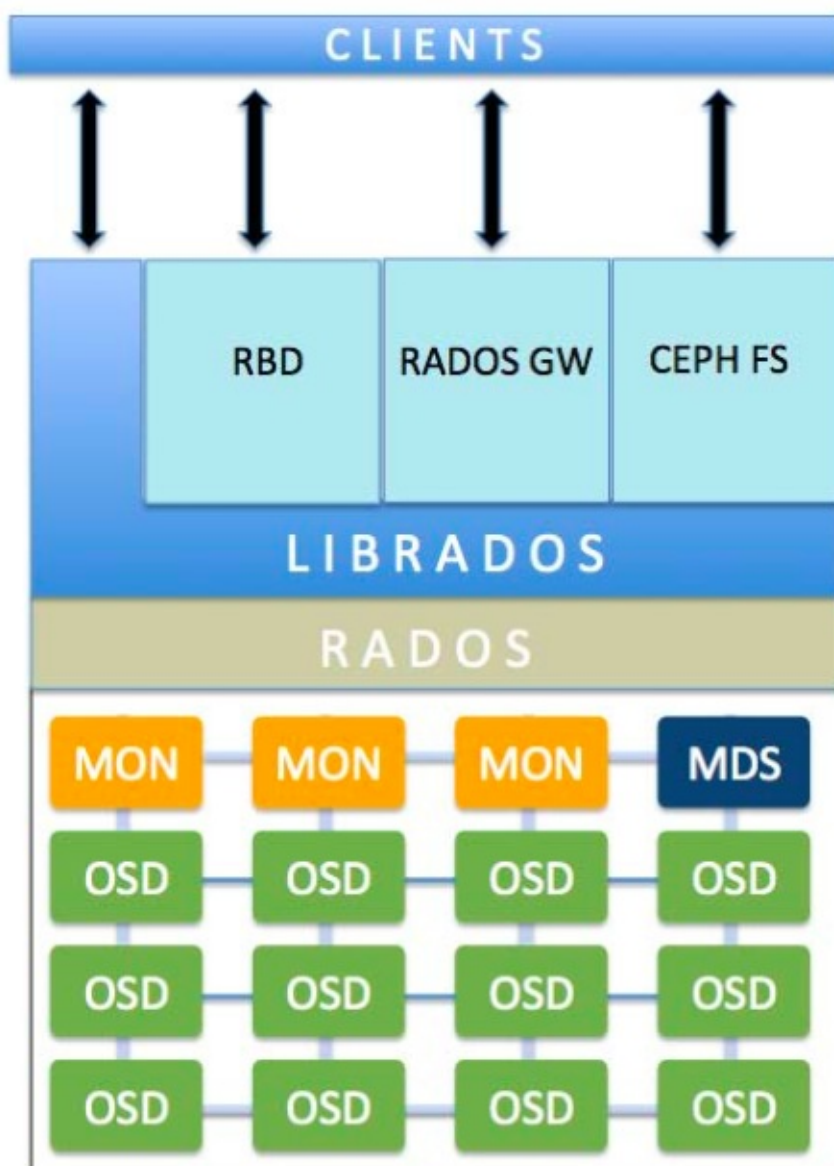
简介

Ceph is a distributed object, block, and file storage platform.

也就是说，使用Ceph系统我们可以提供对象存储、块设备存储和文件系统服务，更有趣的是基于Ceph的key-value存储和NoSQL存储也在开发中，让Ceph成为目前最流行的统一存储系统。

Ceph底层提供了分布式的RADOS存储，用与支撑上层的librados和RGW、RBD、CephFS等服务。Ceph实现了非常底层的object storage，是纯粹的SDS，并且支持通用的ZFS、Btrfs和Ext4文件系统，能轻易得Scale，没有单点故障。

接下来马上介绍Ceph的各个基础组件。



基础组件

Object

Ceph最底层的存储单元是Object对象，每个Object包含元数据和原始数据。

OSD

OSD全称Object Storage Device，也就是负责响应客户端请求返回具体数据的进程。一个Ceph集群一般都有很多个OSD。

PG

PG全称Placement Groups，是一个逻辑的概念，一个PG包含多个OSD。引入PG这一层其实是为了更好的分配数据和定位数据。

Monitor

一个Ceph集群需要多个Monitor组成的小集群，它们通过Paxos同步数据，用来保存OSD的元数据。

RADOS

RADOS全称Reliable Autonomic Distributed Object Store，是Ceph集群的精华，用户实现数据分配、Failover等集群操作。

Libradio

Librados是Rados提供库，因为RADOS是协议很难直接访问，因此上层的RBD、RGW和CephFS都是通过librados访问的，目前提供PHP、Ruby、Java、Python、C和C++支持。

CRUSH

CRUSH是Ceph使用的数据分布算法，类似一致性哈希，让数据分配到预期的地方。

RBD

RBD全称RADOS block device，是Ceph对外提供的块设备服务。

RGW

RGW全称RADOS gateway，是Ceph对外提供的对象存储服务，接口与S3和Swift兼容。

MDS

MDS全称Ceph Metadata Server，是CephFS服务依赖的元数据服务。

CephFS

CephFS全称Ceph File System，是Ceph对外提供的文件系统服务。

Ceph命令

运行 `ceph/demo` 容器后可以随心所欲执行Ceph命令了。

```
1. root@dev:/# ceph -w
2.   cluster 99ce2286-ac36-40c4-bdb7-c592893b6102
3.   health HEALTH_OK
4.   monmap e1: 1 mons at {dev=10.0.2.15:6789/0}
5.       election epoch 2, quorum 0 dev
6.   mdsmap e9: 1/1/1 up {0=0=up:active}
7.   osdmap e42: 1 osds: 1 up, 1 in
8.   pgmap v912: 120 pgs, 8 pools, 2810 bytes data, 63 objects
9.       3437 MB used, 14198 MB / 18603 MB avail
10.      120 active+clean
11.
    2015-06-29 15:17:07.137895 mon.0 [INF] pgmap v912: 120 pgs: 120 active+clean;
12. 2810 bytes data, 3437 MB used, 14198 MB / 18603 MB avail
```

使用Ceph

目前[Ceph官网](#)提供了多种安装方式，但是最简单的方法当然是使用Docker。

首先通过命令 `ifconfig` 找到当前主机的IP，如下。

```
1. eth0      Link encap:Ethernet  HWaddr 08:00:27:F9:89:52
2.          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
3.          inet6 addr: fe80::a00:27ff:fe9:8952/64  Scope:Link
4.          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
5.          RX packets:288605 errors:0 dropped:0 overruns:0 frame:0
6.          TX packets:55395 errors:0 dropped:0 overruns:0 carrier:0
7.          collisions:0 txqueuelen:1000
8.          RX bytes:307079524 (292.8 MiB)  TX bytes:3963682 (3.7 MiB)
```

记住本机IP为10.0.2.15，然后可以使用[ceph-docker](#)提供的命令。

```
1. docker run -d --net=host -e MON_IP=10.0.2.15 -e CEPH_PUBLIC_NETWORK=10.0.2.0/24
   ceph/demo
```

这样就启动一个单机版Ceph服务了，通过 `docker exec` 进入容器，开始体验Ceph命令吧。

更多介绍

前面对Ceph各个组件都简要介绍了以下，最重要的是RADOS的设计和CRUSH算法的使用，后面会有更详细的介绍。

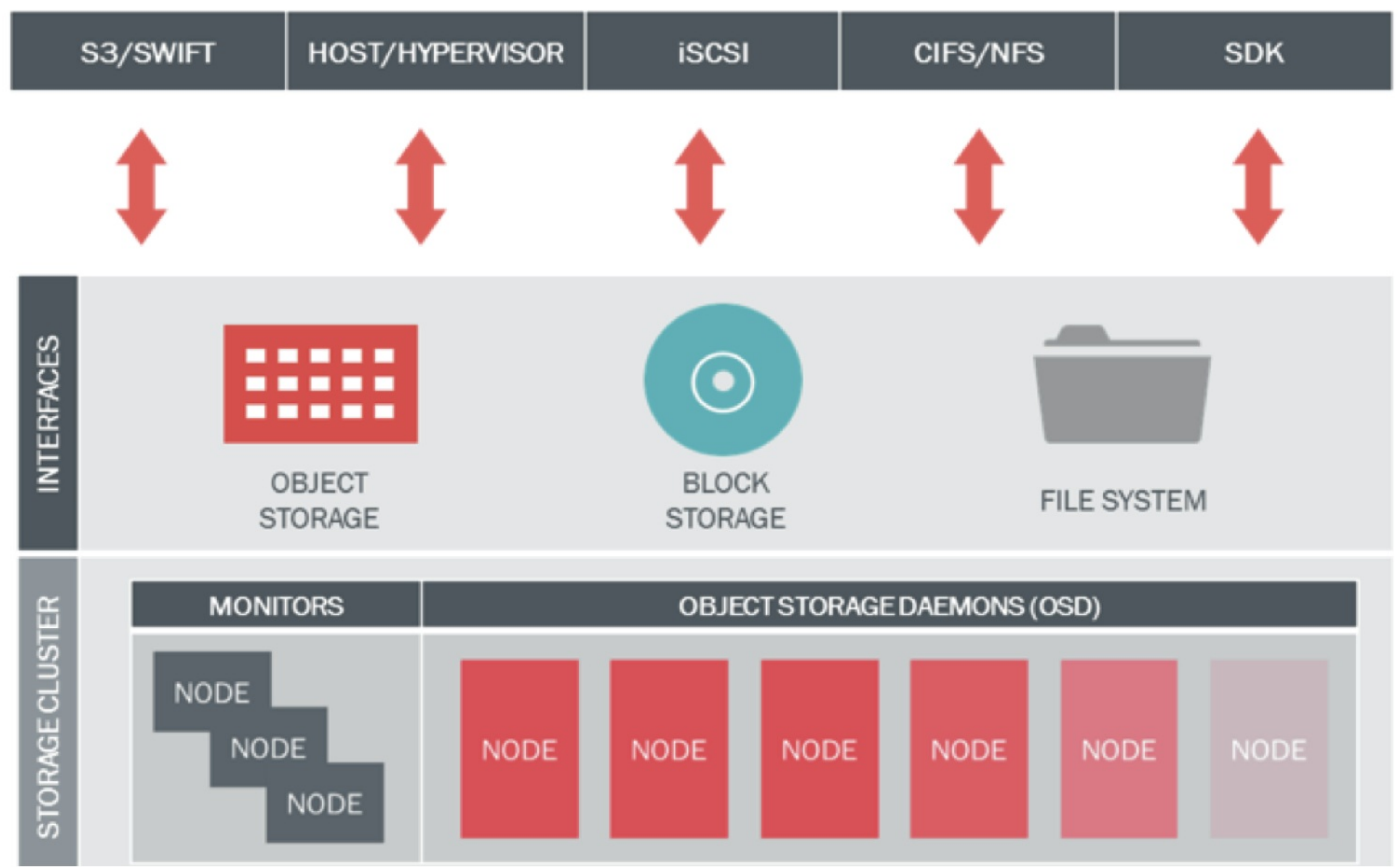
接下来要想学好Ceph，最重要是实践。



第二章 Ceph用法

本章将带领大家一步一步使用Ceph，分布式系统的安装和部署一般都是非常复杂的，而且很多教程不一定适用于本地的环境，我们本章所有代码与命令都使用官方提供Docker容器，保证任何人都能轻易地使用Ceph并且得到预期的结果。

通过本章大家都可以掌握Ceph的基本操作命令，基于Ceph搭建自己的存储系统。



Ceph命令

OSD状态

列举当前所有OSD。

```
1. root@dev:/# ceph osd ls
2. 0
```

查看OSD状态。

```
1. root@dev:/# ceph osd stat
2.      osdmap e21: 1 osds: 1 up, 1 in
```

查看OSD树形结构。

```
1. root@dev:/# ceph osd tree
2. ID WEIGHT  TYPE NAME      UP/DOWN REWEIGHT PRIMARY-AFFINITY
3. -1 1.00000 root default
4. -2 1.00000      host dev
5.  0 1.00000      osd.0    up  1.00000      1.00000
```

导出OSD详细信息。

```
1. root@dev:/# ceph osd dump
2. epoch 21
3. fsid fee30c76-aec4-44d4-8138-763969aaa562
4. created 2015-07-12 05:59:12.189734
5. modified 2015-07-12 11:57:56.706958
6. flags
   pool 0 'rbd' replicated size 1 min_size 1 crush_ruleset 0 object_hash rjenkins
7. pg_num 64 pgp_num 64 last_change 2 flags hashspool stripe_width 0
   pool 1 'cephfs_data' replicated size 1 min_size 1 crush_ruleset 0 object_hash
   rjenkins pg_num 8 pgp_num 8 last_change 7 flags hashspool
8. crash_replay_interval 45 stripe_width 0
   pool 2 'cephfs_metadata' replicated size 1 min_size 1 crush_ruleset 0
   object_hash rjenkins pg_num 8 pgp_num 8 last_change 6 flags hashspool
9. stripe_width 0
```

```

pool 3 '.rgw.root' replicated size 1 min_size 1 crush_ruleset 0 object_hash
rjenkins pg_num 8 pgp_num 8 last_change 8 owner 18446744073709551615 flags
10. hashspool stripe_width 0
pool 4 '.rgw.control' replicated size 1 min_size 1 crush_ruleset 0 object_hash
rjenkins pg_num 8 pgp_num 8 last_change 10 owner 18446744073709551615 flags
11. hashspool stripe_width 0
pool 5 '.rgw' replicated size 1 min_size 1 crush_ruleset 0 object_hash rjenkins
pg_num 8 pgp_num 8 last_change 12 owner 18446744073709551615 flags hashspool
12. stripe_width 0
pool 6 '.rgw.gc' replicated size 1 min_size 1 crush_ruleset 0 object_hash
rjenkins pg_num 8 pgp_num 8 last_change 13 owner 18446744073709551615 flags
13. hashspool stripe_width 0
pool 7 '.users.uid' replicated size 1 min_size 1 crush_ruleset 0 object_hash
rjenkins pg_num 8 pgp_num 8 last_change 14 owner 18446744073709551615 flags
14. hashspool stripe_width 0
15. max_osd 1
osd.0 up in weight 1 up_from 20 up_thru 20 down_at 19 last_clean_interval
[5,19) 10.0.2.15:6800/199 10.0.2.15:6801/2000199 10.0.2.15:6802/2000199
16. 10.0.2.15:6803/2000199 exists,up 5aaf2355-aa45-4452-a401-9add47541a88

```

Monitor状态

查看Monitor状态。

```

1. root@dev:/# ceph mon stat
2. e1: 1 mons at {dev=10.0.2.15:6789/0}, election epoch 2, quorum 0 dev

```

导出Monitor详细信息。

```

1. root@dev:/# ceph mon dump
2. dumped monmap epoch 1
3. epoch 1
4. fsid fee30c76-aec4-44d4-8138-763969aaa562
5. last_changed 2015-07-12 05:59:11.900924
6. created 2015-07-12 05:59:11.900924
7. 0: 10.0.2.15:6789/0 mon.dev

```

PG操作

导出PG详细信息。

```

1. root@dev:/# ceph pg dump
2. pool 0 0 0 0 0 0 0 0 0
3. pool 1 0 0 0 0 0 0 0 0
4. pool 2 20 0 0 0 0 1962 30 30
5. pool 3 3 0 0 0 0 848 3 3
6. pool 4 8 0 0 0 0 0 50 50
7. pool 5 0 0 0 0 0 0 0 0
8. pool 6 32 0 0 0 0 0 192 192
9. pool 7 0 0 0 0 0 0 0 0
10. sum 63 0 0 0 0 2810 275 275
11. osdstat kbused kbavail kb hb in hb out
12. 0 4630564 13428604 19049892 [] []
13. sum 4630564 13428604 19049892

```

CRUSH操作

导出CRUSH详细信息。

```
1. root@dev:/# ceph osd crush dump
```

将CURHS导出为不可读的配置文件。

```

1. root@dev:/# ceph osd getcrushmap -o crushmap.txt
2. got crush map from osdmap epoch 21

```

解码CRUSH配置文件为可读的文本文件。

```

1. root@dev:/# crushtool -d crushmap.txt -o crushmap-decompile.txt
2. root@dev:/# vim crushmap-decompile.txt

```

Ceph容器

使用Docker

试用Ceph最简单的方法是启动Ceph容器，使用前必须保证本地已经安装好Docker。Debian/Ubuntu用户可以通过 `apt-get install docker.io` 安装，CentOS/Redhat用户可以通过 `yum install docker` 安装，Mac和Windows建议下载boot2docker来使用。

基本的docker命令如下。

1. `docker images`
2. `docker pull ubuntu`
3. `docker run -i -t ubuntu /bin/bash`
4. `docker exec -i -t ubuntu /bin/bash`
5. `docker ps`

Ceph容器

Ceph社区提供了官方的docker镜像，代码与教程都托管到Github上
<https://github.com/ceph/ceph-docker>。

由于Ceph的配置文件必须指定IP地址，因此使用Ceph容器前我们必须获得本机IP，如果是boot2docker用户需要获得其虚拟机IP。

1. `docker@dev:~$ ifconfig | grep "eth0" -A 2`
2. `eth0` `Link encap:Ethernet HWaddr 08:00:27:F9:89:52`
3. `inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0`
4. `inet6 addr: fe80::a00:27ff:fe49:8952/64 Scope:Link`

启动Ceph

启动单机版ceph非常简单，使用下述命令。

- ```
docker@dev:~$ docker run -d --net=host -e MON_IP=10.0.2.15 -e
```
1. `CEPH_NETWORK=10.0.2.0/24 ceph/demo`
  2. `badaf5c8fed1e0edf6f2281539669d8f6522ba54b625076190fe4d6de79745ff`

然后可以通过 `docker ps` 来检查容器状态。

```
1. docker@dev:~$ docker ps
```

| CONTAINER ID    | IMAGE     | COMMAND                       | CREATED       |
|-----------------|-----------|-------------------------------|---------------|
| STATUS          | PORTS     | NAMES                         |               |
| badaf5c8fed1    | ceph/demo | <code>"/entrypoint.sh"</code> | 9 seconds ago |
| 3. Up 9 seconds |           | loving_pasteur                |               |

这里ceph容器的ID为“badaf5c8fed1”，可以快速进入容器。

```
1. docker@dev:~$ docker exec -i -t badaf5c8fed1 /bin/bash
2. root@dev:/#
```

# Ceph命令

## 检查状态

最简单的ceph命令是， `ceph -w` ，也就是watch整个ceph集群的状态。

```

1. root@dev:/# ceph -w
2. cluster fee30c76-aec4-44d4-8138-763969aaa562
3. health HEALTH_OK
4. monmap e1: 1 mons at {dev=10.0.2.15:6789/0}
5. election epoch 2, quorum 0 dev
6. mdsmmap e5: 1/1/1 up {0=0=up:active}
7. osdmap e18: 1 osds: 1 up, 1 in
8. pgmap v22: 120 pgs, 8 pools, 2810 bytes data, 63 objects
9. 4519 MB used, 13115 MB / 18603 MB avail
10. 120 active+clean
11.
 2015-07-12 06:53:58.454077 mon.0 [INF] pgmap v22: 120 pgs: 120 active+clean;
12. 2810 bytes data, 4519 MB used, 13115 MB / 18603 MB avail

```

或者通过 `ceph status` 命令。

```

1. root@dev:/# ceph status
2. cluster fee30c76-aec4-44d4-8138-763969aaa562
3. health HEALTH_OK
4. monmap e1: 1 mons at {dev=10.0.2.15:6789/0}
5. election epoch 2, quorum 0 dev
6. mdsmmap e5: 1/1/1 up {0=0=up:active}
7. osdmap e21: 1 osds: 1 up, 1 in
8. pgmap v30: 120 pgs, 8 pools, 2810 bytes data, 63 objects
9. 4521 MB used, 13114 MB / 18603 MB avail
10. 120 active+clean

```

## RADOS命令

### Pool简介

Pool是Ceph中的逻辑概念，不同的应用可以使用不同的Pool。

### Pool相关命令

```
1. root@dev:/# rados lspools
2. rbd
3. cephfs_data
4. cephfs_metadata
5. .rgw.root
6. .rgw.control
7. .rgw
8. .rgw.gc
9. .users.uid
```

如果想获得特定Pool的数据。

```
1. root@dev:/# rados -p .rgw ls
2. root@dev:/# rados -p .rgw.root ls
3. default.region
4. region_info.default
5. zone_info.default
```

### 容量相关

获得当前OSD所用容量。

```
1. root@dev:/# rados df
 pool name KB objects clones degraded
2. unfound rd rd KB wr wr KB
 .rgw 0 0 0 0
3. 0 0 0 0 0
 .rgw.control 0 8 0 0
4. 0 0 0 0 0
```



```

 .rgw.gc 0 32 0 0 0
5. 288 256 192 0
 .rgw.root 1 3 0 0 0
6. 0 0 3 3
 .users.uid 0 0 0 0 0
7. 0 0 0 0
 cephfs_data 0 0 0 0 0
8. 0 0 0 0
 cephfs_metadata 2 20 0 0 0
9. 0 0 31 8
 rbd 0 0 0 0 0
10. 0 0 0 0
11. total used 4630192 63
12. total avail 13428976
13. total space 19049892
```

# Bucket命令

## 创建Bucket

```

1. root@dev:/# ceph osd tree
2. ID WEIGHT TYPE NAME UP/DOWN REWEIGHT PRIMARY-AFFINITY
3. -1 1.000000 root default
4. -2 1.000000 host dev
5. 0 1.000000 osd.0 up 1.000000 1.000000
6. root@dev:/# ceph osd crush add-bucket rack01 rack
7. added bucket rack01 type rack to crush map
8. root@dev:/# ceph osd crush add-bucket rack02 rack
9. added bucket rack02 type rack to crush map
10. root@dev:/# ceph osd crush add-bucket rack03 rack
11. added bucket rack03 type rack to crush map
12. root@dev:/# ceph osd tree
13. ID WEIGHT TYPE NAME UP/DOWN REWEIGHT PRIMARY-AFFINITY
14. -5 0 rack rack03
15. -4 0 rack rack02
16. -3 0 rack rack01
17. -1 1.000000 root default
18. -2 1.000000 host dev
19. 0 1.000000 osd.0 up 1.000000 1.000000

```

## 移动Rack

```

1. root@dev:/# ceph osd crush move rack01 root=default
2. moved item id -3 name 'rack01' to location {root=default} in crush map
3. root@dev:/# ceph osd crush move rack02 root=default
4. moved item id -4 name 'rack02' to location {root=default} in crush map
5. root@dev:/# ceph osd crush move rack03 root=default
6. moved item id -5 name 'rack03' to location {root=default} in crush map
7. root@dev:/# ceph osd tree
8. ID WEIGHT TYPE NAME UP/DOWN REWEIGHT PRIMARY-AFFINITY
9. -1 1.000000 root default
10. -2 1.000000 host dev
11. 0 1.000000 osd.0 up 1.000000 1.000000
12. -3 0 rack rack01

```

```
13. -4 0 rack rack02
14. -5 0 rack rack03
```

## Object操作

### 创建Pool

```
1. root@dev:/# ceph osd pool create web-services 128 128
2. pool 'web-services' created
3. root@dev:/# rados lspools
4. rbd
5. cephfs_data
6. cephfs_metadata
7. .rgw.root
8. .rgw.control
9. .rgw
10. .rgw.gc
11. .users.uid
12. web-services
```

### 添加Object

```
1. root@dev:/# echo "Hello Ceph, You are Awesome like MJ" > /tmp/helloceph
2. root@dev:/# rados -p web-services put object1 /tmp/helloceph
3. root@dev:/# rados -p web-services ls
4. object1
5. root@dev:/# ceph osd map web-services object1
 osdmap e29 pool 'web-services' (8) object 'object1' -> pg 8.bac5debc (8.3c) ->
6. up ([0], p0) acting ([0], p0)
```

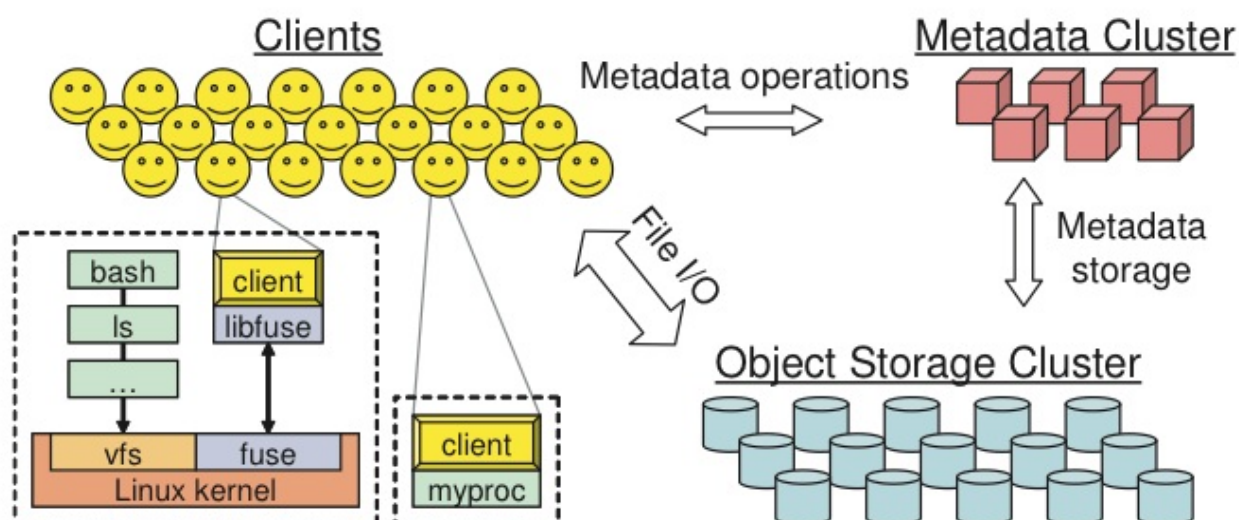
### 查看Object

```
1. root@dev:/# cd /var/lib/ceph/osd/
2. root@dev:/var/lib/ceph/osd# ls ceph-0/current/8.3c_head/
3. __head_0000003C__8 object1__head_BAC5DEBC__8
 root@dev:/var/lib/ceph/osd# cat ceph-
4. 0/current/8.3c_head/object1__head_BAC5DEBC__8
5. Hello Ceph, You are Awesome like MJ
```

## 第三章 Ceph架构

前面粗略地介绍了Ceph、RADOS、CRUSH等概念和用法，本章将重点剖析具体的架构与算法，详细介绍其设计和实现细节。

通过本章可以完全掌握Ceph的核心技术和算法。



# 论文

---

## 简介

---

Ceph是Sega本人的博士论文作品，想了解Ceph的架构设计最好的方式是阅读Sega的论文，其博士论文我们称之为长论文，后来整理成三篇较短的论文。

## 长论文

---

长论文包含了RADOS、CRUSH等所有内容的介绍，但篇幅相当长，如果感兴趣可以阅读，标题为《CEPH: RELIABLE, SCALABLE, AND HIGH-PERFORMANCE DISTRIBUTED STORAGE》，地址<https://ceph.com/wp-content/uploads/2016/08/weil-thesis.pdf>。

## CRUSH论文

---

CRUSH论文标题为《CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data》，地址<https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf>，介绍了CRUSH的设计与实现细节。

## RADOS论文

---

RADOS论文标题为《RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters》，地址为<https://ceph.com/wp-content/uploads/2016/08/weil-rados-pdsw07.pdf>，介绍了RADOS的设计与实现细节。

## CephFS论文

---

CephFS论文标题为《Ceph: A Scalable, High-Performance Distributed File System》，地址为<https://ceph.com/wp-content/uploads/2016/08/weil-ceph-osdi06.pdf>，介绍了Ceph的基本架构和Ceph的设计与实现细节。

# CRUSH详解

---

## CRUSH简介

---

CRUSH全称Controlled Replication Under Scalable Hashing，是一种数据分发算法，类似于哈希和一致性哈希。哈希的问题在于数据增长时不能动态加Bucket，一致性哈希的问题在于加Bucket时数据迁移量比较大，其他数据分发算法依赖中心的Metadata服务器来存储元数据效率较低，CRUSH则是通过计算、接受多维参数的来解决动态数据分发的场景。

## 算法基础

---

在学习CRUSH之前，需要了解以下内容。

CRUSH算法接受的参数包括cluster map，也就是硬盘分布的逻辑位置，例如这有多少个机房、多少个机柜、硬盘是如何分布的等等。cluster map是类似树的多层结果，子节点是真正存储数据的device，每个device都有id和权重，中间节点是bucket，bucket有多种类型用于不同的查询算法，例如一个机柜一个机架一个机房就是bucket。

另一个参数是placement rules，它指定了一份数据有多少备份，数据的分布有什么限制条件，例如同一份数据不能放在同一个机柜里等的功能。每个rule就是一系列操作，take操作就是就是选一个bucket，select操作就是选择n个类型是t的项，emit操作就是提交最后的返回结果。select要考虑的东西主要包括是否冲突、是否有失败和负载问题。

算法的还有一个输入是整数x，输出则是一个包含n个目标的列表R，例如三备份的话输出可能是[1, 3, 5]。

## 算法解读

---

```

1: procedure TAKE(a) ▷ Put item a in working vector \vec{i}
2: $\vec{i} \leftarrow [a]$
3: end procedure

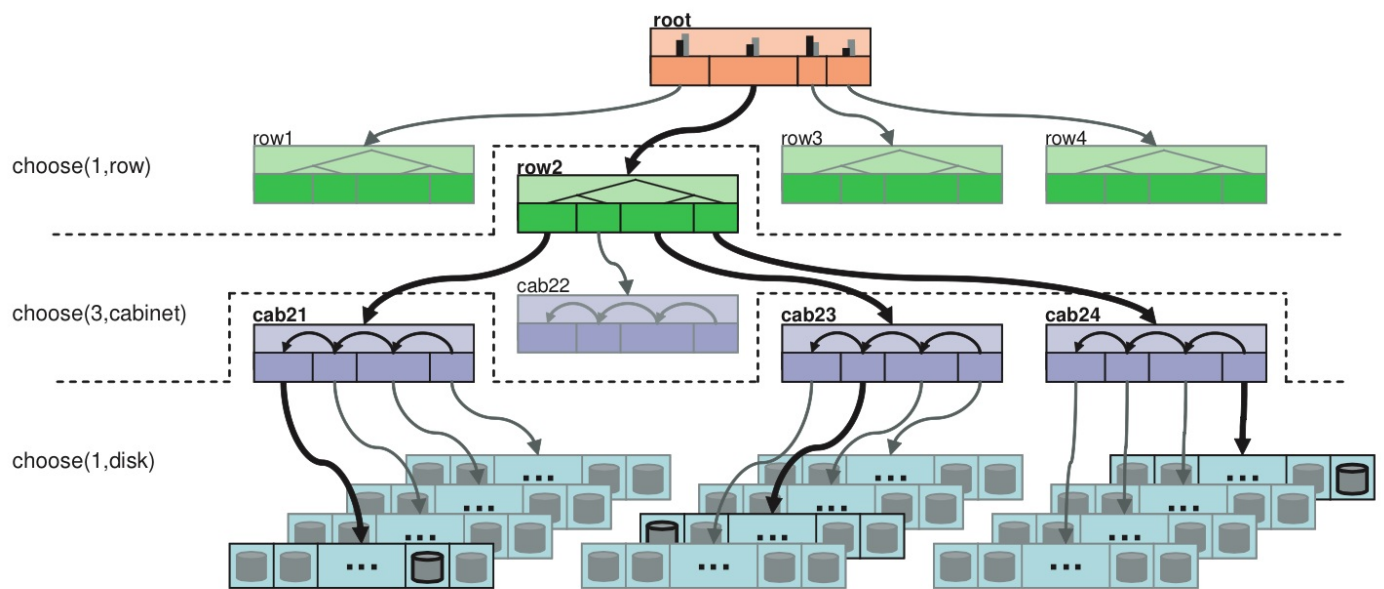
4: procedure SELECT(n, t) ▷ Select n items of type t
5: $\vec{o} \leftarrow \emptyset$ ▷ Our output, initially empty
6: for $i \in \vec{i}$ do ▷ Loop over input \vec{i}
7: $f \leftarrow 0$ ▷ No failures yet
8: for $r \leftarrow 1, n$ do ▷ Loop over n replicas
9: $f_r \leftarrow 0$ ▷ No failures on this replica
10: $retry_descent \leftarrow false$
11: repeat
12: $b \leftarrow bucket(i)$ ▷ Start descent at bucket i
13: $retry_bucket \leftarrow false$
14: repeat
15: if “first n ” then ▷ See Section 3.2.2
16: $r' \leftarrow r + f$
17: else
18: $r' \leftarrow r + f_r n$
19: end if
20: $o \leftarrow b.c(r', x)$ ▷ See Section 3.4
21: if $type(o) \neq t$ then
22: $b \leftarrow bucket(o)$ ▷ Continue descent
23: $retry_bucket \leftarrow true$
24: else if $o \in \vec{o}$ or $failed(o)$ or $overload(o, x)$
25: then
26: $f_r \leftarrow f_r + 1, f \leftarrow f + 1$
27: if $o \in \vec{o}$ and $f_r < 3$ then
28: $retry_bucket \leftarrow true$ ▷ Retry
29: collisions locally (see Section 3.2.1)
30: else
31: $retry_descent \leftarrow true$ ▷ Otherwise
32: end if
33: until $\neg retry_bucket$
34: $\vec{o} \leftarrow [\vec{o}, o]$ ▷ Add o to output \vec{o}
35: end repeat
36: end for
37: $\vec{i} \leftarrow \vec{o}$ ▷ Copy output back into \vec{i}
38: end procedure

39: procedure EMIT ▷ Append working vector \vec{i} to result
40: $\vec{R} \leftarrow [\vec{R}, \vec{i}]$
41: end procedure

```

图虽然很复杂，但如果理解了几个基本操作的含义就很好读下来了，这里是三个操作的伪代码，take和emit很好理解，select主要是遍历当前bucket，如果出现重复、失败或者超载就跳过，其中稍微复杂的“first  $n$ ”部分是一旦遇到失败，第一种情况是直接使用多备份，第二种情况是使用erasing code基本可以忽略。看着下面的图就更好理解具体的算法了。





# CRUSH总结

---

## 算法总结

---

CRUSH与一致性哈希最大的区别在于接受的参数多了cluster map和placement rules，这样就可以根据目前cluster的状态动态调整数据位置，同时通过算法得到一致的结果。

## 算法补充

---

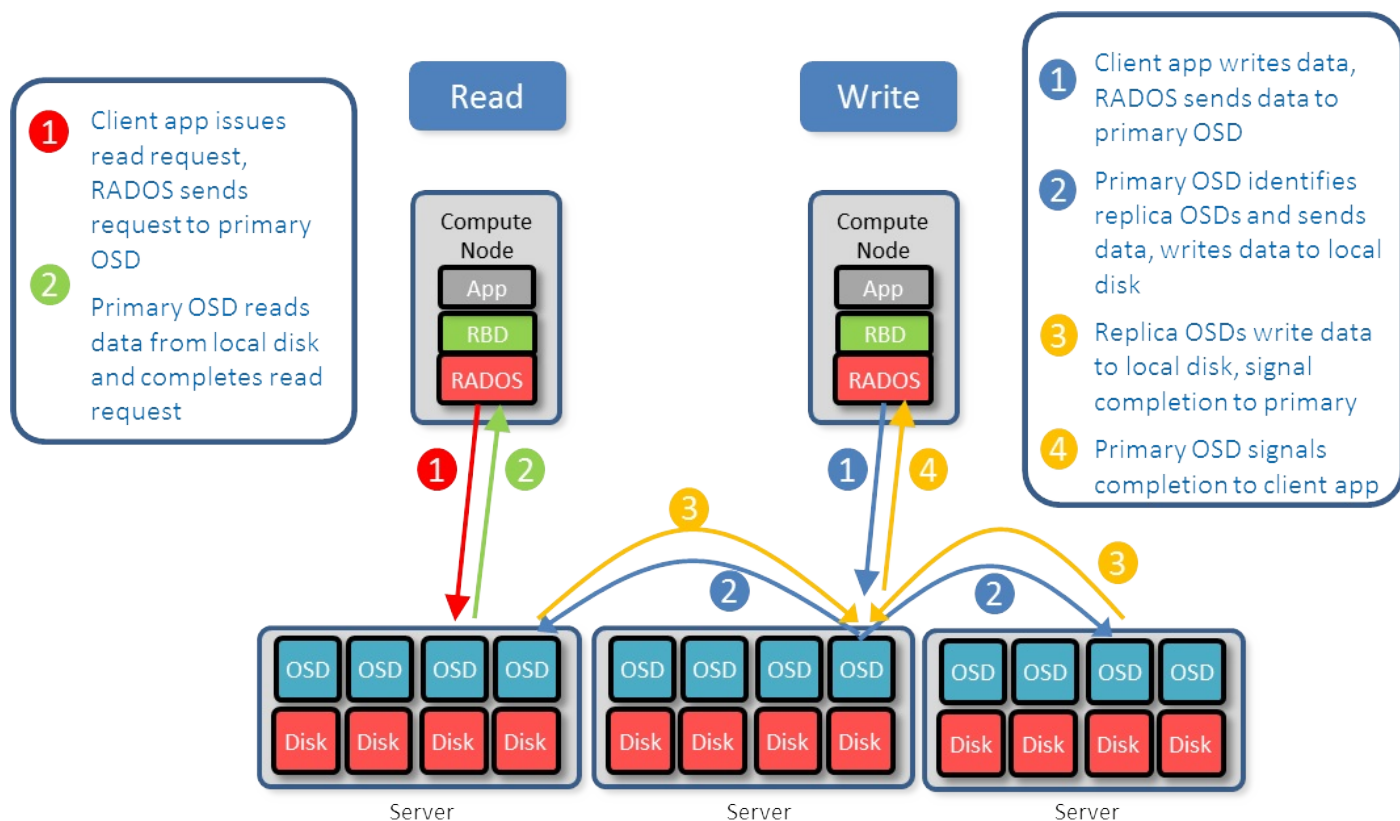
前面介绍了bucket根据不同场景有四种类型，分别是Uniform、List、Tree和Straw，他们对应运行数据和数据迁移量有不同的tradeoff，目前大家都在用Straw因此不太需要关系其他。

目前erasing code可以大大减小三备份的数据量，但除了会导致数据恢复慢，部分ceph支持的功能也是不能直接用的，而且功能仍在开发中不建议使用。

# RADOS详解

## RADOS简介

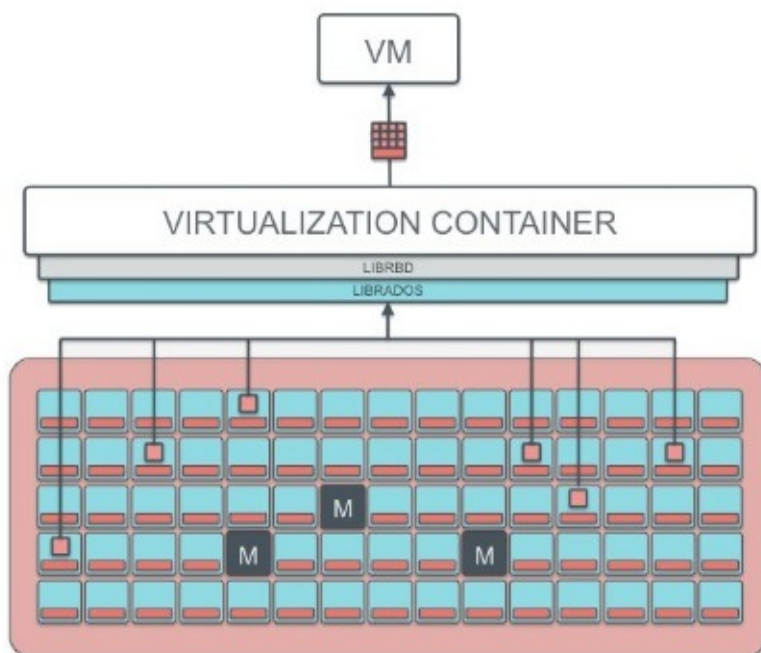
RADOS是ceph实现高可用、数据自动恢复的框架，设计的点比较多后面在详细介绍。



## 第四章 RBD

本章将介绍使用ceph搭建块设备服务。

# RADOS Block Device



## RBD命令

### 检查Pool

Ceph启动后默认创建rbd这个pool，使用rbd命令默认使用它，我们也可以创建新的pool。

1. `rados lspools`
2. `ceph osd pool create rbd_pool 1024`

### 创建Image

使用rbd命令创建image，创建后发现rbd这个pool会多一个 `rbd_directory` 的object。

1. `rbd create test_image --size 1024`
2. `rbd ls`
3. `rbd --image test_image info`
4. `rados -p rbd ls`

### 修改Image大小

增加Image大小可以直接使用 `resize` 子命令，如果缩小就需要添加 `--allow-shrink` 参数保证安全。

1. `rbd --image test_image resize --size 2000`
2. `rbd --image test_image resize --size 1000 --allow-shrink`

### 使用Image

通过 `map` 子命令可以把镜像映射成本地块设备，然后就可以格式化和 `mount` 了。

1. `rbd map test_image`
2. `rbd showmapped`
3. `mkfs.ext4 /dev/rbd0`
4. `mount /dev/rbd0 /mnt/`

### 移除Image

1. `umount /dev/rbd0`
2. `rbid unmap /dev/rbd0`
3. `rbid showmapped`

## 删除Image

---

删除和Linux类似使用 `rm` 命令即可。

1. `rbid --image test_image rm`

# RBD快照

## 创建快照

通过 `snap` 子命令可以创建和查看快照。

1. `rbd snap create --image test_image --snap test_snap`
2. `rbd snap ls --image test_image`

## 快照回滚

使用 `snap rollback` 就可以回滚快照，由于快照命名是镜像名后面加@，我们还可以下面的简便写法。

1. `rbd snap rollback --image test_image --snap test_snap`
2. `rbd snap rollback rbd/test_image@test_snap`

## 删除快照

删除快照也很简单，使用 `rm` 子命令，如果想清理所有快照可以使用 `purge` 子命令，注意Ceph删除是异步的不会立即释放空间。

1. `rbd snap rm --image test_image --snap test_snap`
2. `rbd snap purge --image test_image`

## 保护快照

保护快照可以防止用户误删数据，这是clone前必须做的。

1. `rbd snap protect --image test_image --snap test_snap`

要想不保护快照也很容易，使用 `unprotect` 子命令即可。

1. `rbd snap unprotect --image test_image --snap test_snap`

# RBD克隆

---

## 创建clone

---

RBD克隆就是通过快照克隆出新的可读可写的Image，创建前需要创建format为2的镜像快照。

1. `rbd create test_image2 --size 1024 --image-format 2`
2. `rbd snap create --image test_image2 --snap test_snap2`
3. `rbd snap protect --image test_image2 --snap test_snap2`

通过 `clone` 子命令就可以创建clone了。

1. `rbd clone --image test_image2 --snap test_snap2 test_clone`

## 列举clone

---

通过 `children` 子命令可以列举这个快照的所有克隆。

1. `rbd children --image test_image2 --snap test_snap2`

## 填充克隆

---

填充克隆也就是把快照数据flatten到clone中，如果你想删除快照你需要flatten所有的子Image。

1. `rbd flatten --image test_clone`



# RBD和Qemu

## 使用Qemu

官方Qemu已经支持librbd，使用Qemu创建镜像前需要安装工具。

```
1. apt-get install -y qemu-utils
```

## 创建镜像

创建镜像非常简单，使用 `qemu-img` 命令，注意目前RBD只支持raw格式镜像。

```
1. qemu-img create -f raw rbd:rbd/test_image3 1G
```

## 修改镜像大小

修改镜像大小可以使用 `resize` 子命令。

```
1. qemu-img resize rbd:rbd/test_image3 2G
```

## 查看镜像信息

通过 `info` 可以获取Qemu镜像信息。

```
1. qemu-img info rbd:rbd/test_image3
```

# RBD和Virsh

---

## 安装Virsh

---

Virsh是通用的虚拟化技术抽象层，可以统一管理Qemu/KVM、Xen和LXC等，要结合Virsh和RBD使用，我们需要安装对应工具。

```
1. apt-get install -y virt-manager
```

# RBD和OpenStack

## OpenStack介绍

OpenStack开源的云平台，其中Nova提供虚拟机服务，Glance提供镜像服务，Cinder提供块设备服务。因为OpenStack是Python实现的，因此RBD与OpenStack集成需要安装下面的软件。

```
1. apt-get install -y python-ceph
```

## Nova与RBD

修改nova.conf配置文件。

```
1. libvirt_images_type=rbd
2. libvirt_images_rbd_pool=volumes
3. libvirt_images_rbd_ceph_conf=/etc/ceph/ceph.conf
4. rbd_user=cinder
5. rbd_secret_uuid=457eb676-33da-42ec-9a8c-9293d545c337
6.
7. libvirt_inject_password=false
8. libvirt_inject_key=false
9. libvirt_inject_partition=-2
```

## Glance与RBD

修改glance-api.conf配置文件。

```
1. default_store=rbd
2. rbd_store_user=glance
3. rbd_store_pool=images
4. show_image_direct_url=True
```

## Cinder与RBD

修改cinder.conf配置文件。

```
1. volume_driver=cinder.volume.drivers.rbd.RBDDriver
```

```
2. rbd_pool=volumes
3. rbd_ceph_conf=/etc/ceph/ceph.conf
4. rbd_flatten_volume_from_snapshot=false
5. rbd_max_clone_depth=5
6. glance_api_version=2
7.
8. backup_driver=cinder.backup.drivers.ceph
9. backup_ceph_conf=/etc/ceph/ceph.conf
10. backup_ceph_user=cinder-backup
11. backup_ceph_chunk_size=134217728
12. backup_ceph_pool=backups
13. backup_ceph_stripe_unit=0
14. backup_ceph_stripe_count=0
15. restore_discard_excess_bytes=true
```

# Python librbd

## 安装librbd

Ceph官方提供Python库来访问RBD，通过以下命令可以安装。

```
1. apt-get install -y python-ceph
```

## 创建Image

使用librbd创建Image也很简单，下面是示例代码。

```
1. import rados
2. import rbd
3.
4. cluster = rados.Rados(conffile='/etc/ceph/ceph.conf')
5. cluster.connect()
6. ioctx = cluster.open_ioctx('rbd')
7.
8. rbd_inst = rbd.RBD()
9. size = 1024**3
10. image_name = "test_image"
11. rbd_inst.create(ioctx, image_name, size)
12.
13. image = rbd.Image(ioctx, image_name)
14. data = 'foo' * 200
15. image.write(data, 0)
16.
17. image.close()
18. ioctx.close()
19. cluster.shutdown()
```

也可以把下面代码保存成文件直接执行。

```
1. cluster = rados.Rados(conffile='/etc/ceph/ceph.conf')
2. try:
3. ioctx = cluster.open_ioctx('rbd')
4. try:
5. rbd_inst = rbd.RBD()
```

```
6. size = 1024**3
7. image_name = "test_image"
8. rbd_inst.create(ioctx, image_name, size)
9. image = rbd.Image(ioctx, image_name)
10. try:
11. data = 'foo' * 200
12. image.write(data, 0)
13. finally:
14. image.close()
15. finally:
16. ioctx.close()
17. finally:
18. cluster.shutdown()
```

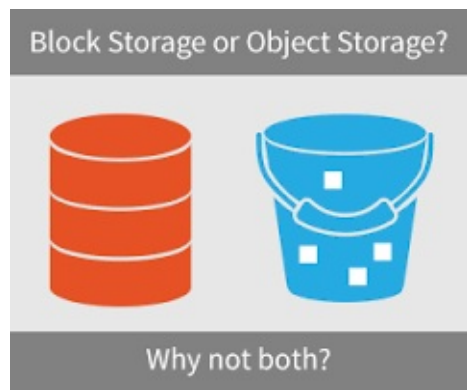
或者这样。

```
1. with rados.Rados(conffile='/etc/ceph/ceph.conf') as cluster:
2. with cluster.open_ioctx('rbd') as ioctx:
3. rbd_inst = rbd.RBD()
4. size = 1024**3
5. image_name = "test_image"
6. rbd_inst.create(ioctx, image_name, size)
7. with rbd.Image(ioctx, image_name) as image:
8. data = 'foo' * 200
9. image.write(data, 0)
```

## 第五章 RGW

RGW是ceph提供对象存储服务，本章将介绍RGW的搭建和使用，从而提供类似S3和Swift服务。

通过本章你可以在本地起ceph的S3服务，然后使用命令行或者SDK工具来访问对象存储服务，并且使用ceph管理用户和quota。

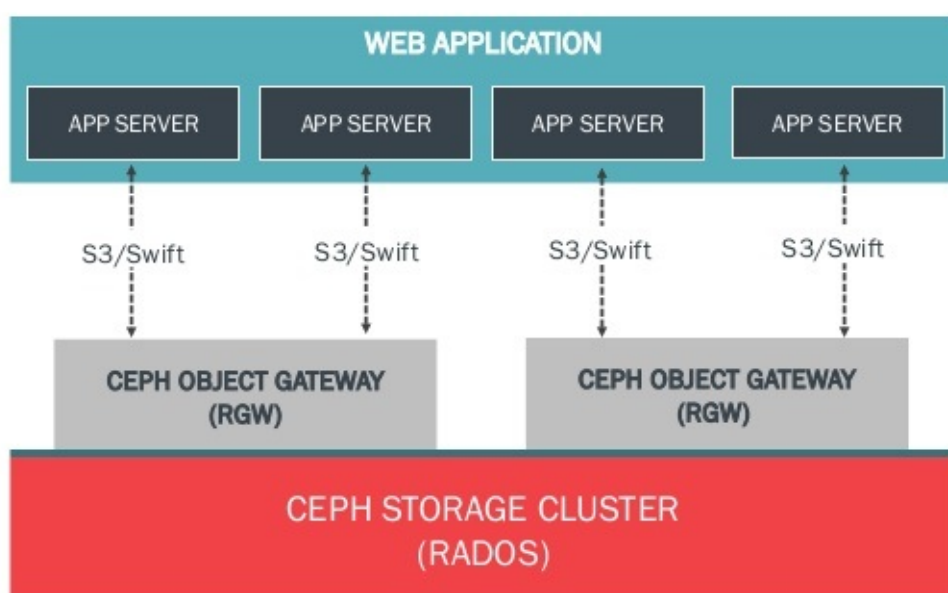


## RGW介绍

RGW全称Rados Gateway，是ceph封装RADOS接口而提供的gateway服务，并且实现S3和Swift兼容的接口，也就是说用户可以使用S3或Swift的命令行工具或SDK来使用RGW。

RGW对象存储也可以作为docker registry的后端，相对与本地存储，将docker镜像存储到RGW后端可以保证即使机器宕机或者操作系统crash也不会丢失数据。

### Ceph as Cloud Storage





# RGW用法

---

## 使用RGW

---

RGW同时提供了S3和Swift兼容的接口，因此只要启动了RGW服务，就可以像使用S3或Swift那样访问RGW的object和bucket了。

本地启动RGW的命令也很简单，启动ceph/demo镜像即可，命令如下。

```
docker run -d --net=host -e MON_IP=10.251.0.105 -e CEPH_NETWORK=10.251.0.0/24
1. ceph/demo
```

## 用户操作

---

查看用户信息。

```
1. radosgw-admin user info --uid=mona
```

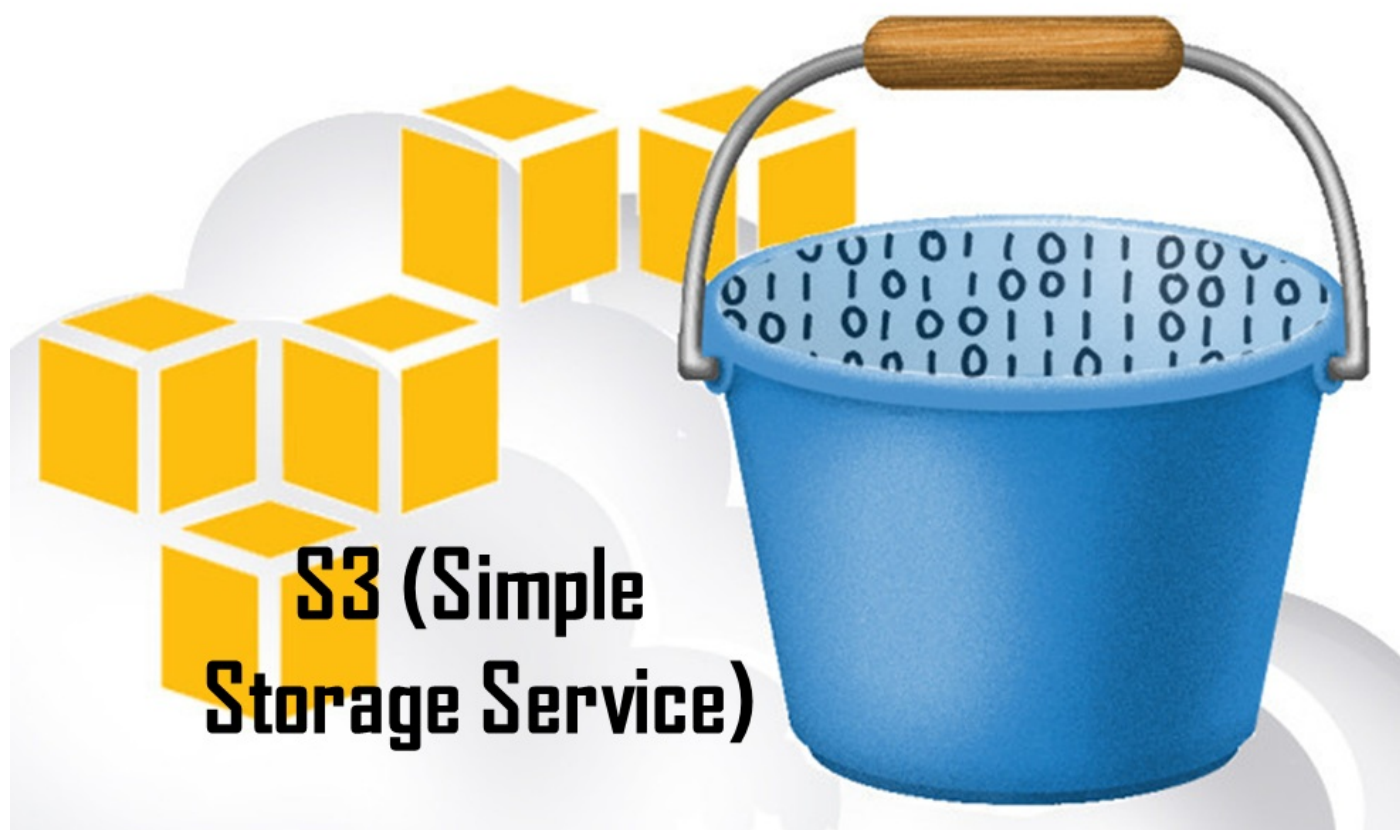
## Bucket操作

---

查看bucket信息。

```
1. root@dev:~# radosgw-admin bucket stats
2. []
```

## S3命令



## 创建用户

```
root@dev:/# radosgw-admin user create --uid=mona --display-name="Monika Singh"
1. --email=mona@example.com
2. {
3. "user_id": "mona",
4. "display_name": "Monika Singh",
5. "email": "mona@example.com",
6. "suspended": 0,
7. "max_buckets": 1000,
8. "auid": 0,
9. "subusers": [],
10. "keys": [
11. {
12. "user": "mona",
13. "access_key": "2196PJ0MA6FLHCVKVFWD",
14. "secret_key": "e01\dmE0EU5LlooexlWwcqJYZrt3Gzp\nBXsQCwz"
15. }
16.],
```

```
17. "swift_keys": [],
18. "caps": [],
19. "op_mask": "read, write, delete",
20. "default_placement": "",
21. "placement_tags": [],
22. "bucket_quota": {
23. "enabled": false,
24. "max_size_kb": -1,
25. "max_objects": -1
26. },
27. "user_quota": {
28. "enabled": false,
29. "max_size_kb": -1,
30. "max_objects": -1
31. },
32. "temp_url_keys": []
33. }
```

## 添加Capabilities

```
1. radosgw-admin caps add --uid=mona --caps="users=*"
2. radosgw-admin caps add --uid=mona --caps="buckets=*"
3. radosgw-admin caps add --uid=mona --caps="metadata=*"
4. radosgw-admin caps add --uid=mona --caps="zone=*"
```

## 安装s3cmd

```
1. apt-get install python-setuptools
2. git clone https://github.com/s3tools/s3cmd.git
3. cd s3cmd/
4. python setup.py install
```

## 使用s3cmd

必须提前设置.s3cfg文件。

```
1. s3cmd ls
```

# Swift命令

## 创建用户

```
root@dev:~# radosgw-admin subuser create --uid=mona --subuser=mona:swift --
1. access=full --secret=secretkey --key-type=swift
2. {
3. "user_id": "mona",
4. "display_name": "Monika Singh",
5. "email": "mona@example.com",
6. "suspended": 0,
7. "max_buckets": 1000,
8. "auid": 0,
9. "subusers": [
10. {
11. "id": "mona:swift",
12. "permissions": "<none>"
13. }
14.],
15. "keys": [
16. {
17. "user": "mona",
18. "access_key": "2196PJ0MA6FLHCVKVFWDW",
19. "secret_key": "e01\dmE0EU5LlooexlWwcqJYZrt3Gzp\nBXsQCwz"
20. },
21. {
22. "user": "mona:swift",
23. "access_key": "2FTDLNGGOWALF1ZS5XHY",
24. "secret_key": ""
25. }
26.],
27. "swift_keys": [
28. {
29. "user": "mona:swift",
30. "secret_key": "secretkey"
31. }
32.],
33. "caps": [
34. {
35. "type": "buckets",
```

```
36. "perm": "*"
37. },
38. {
39. "type": "metadata",
40. "perm": "*"
41. },
42. {
43. "type": "users",
44. "perm": "*"
45. },
46. {
47. "type": "zone",
48. "perm": "*"
49. }
50.],
51. "op_mask": "read, write, delete",
52. "default_placement": "",
53. "placement_tags": [],
54. "bucket_quota": {
55. "enabled": false,
56. "max_size_kb": -1,
57. "max_objects": -1
58. },
59. "user_quota": {
60. "enabled": false,
61. "max_size_kb": -1,
62. "max_objects": -1
63. },
64. "temp_url_keys": []
65. }
```

## 安装Swift客户端

1. `apt-get install python-pip`
2. `pip install python-swiftclient`

## Swift命令

```
swift -V 1.0 -A http://localhost/auth -U mona:swift -K secretkey post example-
1. bucket
```

```
1. swift -V 1.0 -A http://localhost/auth -U mona:swift -K secretkey list
```

## 第六章 CephFS

---

这一章计划介绍CephFS的搭建和使用。

由于CephFS仍未能上Production环境，本章内容将在CephFS稳定后继续完善。

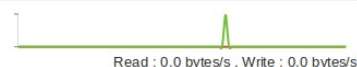
## 第七章 Ceph监控

这一章将介绍Ceph的监控与运维，搭建Ceph是一次性的工作，但运维Ceph却是长久的任务，幸运的是Ceph本身提供了很好的监控管理工具，方便我们管理Ceph集群。

通过本章我们可以学到Ceph官方提供的ceph-rest-api，并带领大家一步一步实现基于ceph-rest-api的Web监控管理工具。



### Cluster health at 09:33:17



14 pgs backfilling

#### history

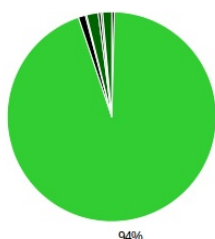
2014-07-09 09:33:02 : 14 pgs backfilling  
2014-07-09 09:32:40 : 1 pgs backfill

### Monitors status

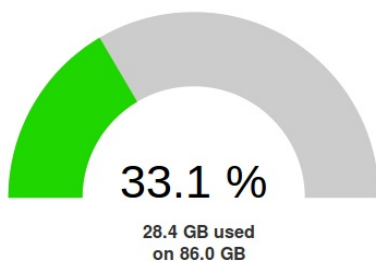
p-sbceph13 p-sbceph14 p-sbceph15

### 1008 Placement groups

- inactive
- peering
- active+remapped
- remapped+peering
- active+clean
- remapped
- active+recovering
- active+remapped+backfilling



Avail. capacity :  
57.6 GB



### 10 OSD

△ osds not verified since 48s

|      | UP   | DOWN |
|------|------|------|
| IN   | 9    | 0    |
| OUT  | 1    | 0    |
| Full | near | full |

### 18 pools

clean unclean  
15 3

### MDSs

up 0  
in 0  
max 1



# Ceph REST API

## 简介

Ceph-rest-api是Ceph官方提供的RESTful API接口，启动其进程后我们可以通过HTTP接口来收集Ceph集群状态与数据，并且进行起停OSD等管理操作。

详细的API文档可参考 <https://dmsimard.com/2014/01/01/documentation-for-ceph-rest-api/>。

## 启动API

因为ceph-rest-api需要管理一个ceph集群，我们建议通过ceph/demo来启动。

```
docker run -d --net=host -e MON_IP=10.0.2.15 -e CEPH_NETWORK=10.0.2.0/24
1. ceph/demo
```

```
1. ceph-rest-api -n client.admin
```

这样在启动单机版ceph的同时，也启动了ceph-rest-api。

## 测试API

通过简单的curl命令即可获得集群的状态信息。

```
1. root@dev:/# curl 127.0.0.1:5000/api/v0.1/health
2. HEALTH_OK
```

或者查询更复杂的数据。

```
1. root@dev:/# curl 127.0.0.1:5000/api/v0.1/osd/tree
2. ID WEIGHT TYPE NAME UP/DOWN REWEIGHT PRIMARY-AFFINITY
3. -1 1.00000 root default
4. -2 1.00000 host dev
5. 0 1.00000 osd.0 up 1.00000 1.00000
6. -3 0 rack rack01
7. -4 0 rack rack02
8. -5 0 rack rack03
```

# Ceph-web

## 监控工具

前面提到过的ceph-rest-api，为我们提供了HTTP接口来访问Ceph集群的状态信息，但是只有ceph-rest-api远远不够，我们需要更友好的Web管理工具。这里我们将介绍开源的ceph-web项目，是非常简单的Web前端，通过ceph-rest-api获得数据并展示。

## Ceph-web

为了不增加API的复杂性，ceph-web遵循官方ceph-rest-api的接口，只是提供HTTP服务器并展示Ceph的数据，开源地址 <https://github.com/tobegit3hub/ceph-web>。

目前ceph-web已经支持通过容器运行，执行下述命令即可一键启动Ceph监控工具。

```
1. docker run -d --net=host tobegit3hub/ceph-web
```

这样通过浏览器打开 <http://127.0.0.1:8080> 就可以看到以下管理界面。

Ceph-web Home Search Github More -

### Ceph Status

Overall status: HEALTH\_OK  
Epoch: 2  
Round status: finished

### OSD Disk Free


Total disk(KB): 19049892  
Used disk(KB): 4522632  
Available disk(KB): 13536536  
Average utilization: 23.740985  
Dev: 0  
Max var: 1  
Min var: 1

### OSD CRUSH Dump

OSD Profile: unknown  
OSD has\_v3\_rules: 0  
OSD has\_v4\_buckets: 0

# 第八章 参与开发

本章将介绍如何参与Ceph社区开发，以及Ceph相关项目的介绍。

 **ceph / ceph**

Watch 273

Unstar 1,457

Fork 842

Ceph is a distributed object, block, and file storage platform <http://ceph.com>

43,140 commits

562 branches

187 releases

292 contributors

Branch: master

ceph / +

Merge pull request #5445 from ceph/wip-12432

oritwas authored a day ago

latest commit ef86e29259

|                                   |                                                      |              |
|-----------------------------------|------------------------------------------------------|--------------|
| admin                             | admin/build-doc: fix dependency checks               | 2 months ago |
| bin                               | make_dist.sh: rename from bin/make_dist_tarball.sh   | 2 months ago |
| ceph-erasure-code-corpus @ ...    | erasure code: shec add ceph-erasure-code-corpus      | 16 days ago  |
| ceph-object-corpus @ 7bd9bbe      | ceph-object-corpus: drop coll_t 'foo' and 'bar'      | a month ago  |
| cmake/modules                     | CMake: add FindNSPR.cmake                            | 3 months ago |
| debian                            | remove rest-bench                                    | 3 days ago   |
| doc                               | doc: krbd supports clones since 3.10                 | 2 days ago   |
| etc/sysconfig/SuSEfirewall2.d/... | packaging: move SuSEfirewall2 templates out of src   | 3 months ago |
| examples                          | Fix typos in librados example code                   | 2 months ago |
| fusetrace                         | remove superfluous second semicolons at end of lines | a year ago   |
| keys                              | keys: renew autobuild.asc key                        | 3 years ago  |

Code

Pull requests 228

Pulse

Graphs

HTTPS clone URL

<https://github.com/ceph/ceph>

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

Download ZIP

## 贡献代码

---

目前Ceph代码托管到[Github](#)，任何开发者都可以给Ceph项目提交Pull-request，由committer决定是否merge你的代码。

而Ceph使用自己的Bug跟踪系统[tracker](#)，代码的Bug、功能改进以及版本发布都会在tracker中及时跟新。

如果你有志于参与Ceph的开发，请订阅他们的[邮件列表](#)和IRC频道。

# Ceph-docker

---

Ceph-docker是ceph官方团队维护的docker镜像，这些镜像都是每次有代码提交都会automated build生成的，你可以使用这些镜像快速启动ceph环境或者使用docker部署ceph集群。

如果你对官方docker镜像不满意，也可以通过pull-request提交你的修改，例如前面提到的ceph-rest-api也是我们提交到ceph/demo镜像的。

## 结语

---

最后，感谢读者关注本书以及ceph存储系统。

Ceph是开源的块设备、对象存储、文件系统服务，[UnitedStack](#)有全国最优秀的ceph团队，欢迎与各位大牛交流。

本书所有内容也开源在[Github](#)，如有勘误或者想要更详细的教程，欢迎提交issue与pull-request。