

Ceph



# ceph

ceph 是一个统一的、分布式的存储系统。

ceph 的统一体现在可以提供文件系统、块存储和对象存储，分布式体现在可以动态扩展。在国内一些公司的云环境中，通常会采用 ceph 作为 openstack 的唯一后端存储来提高数据转发效率。

# ceph

ceph 的特点:

## 1、高可用性

ceph 默认将数据存储三份，可以由管理员自定义，ceph 可以忍受多种故障场景并自动尝试并行修复。

## 2、高度自动化

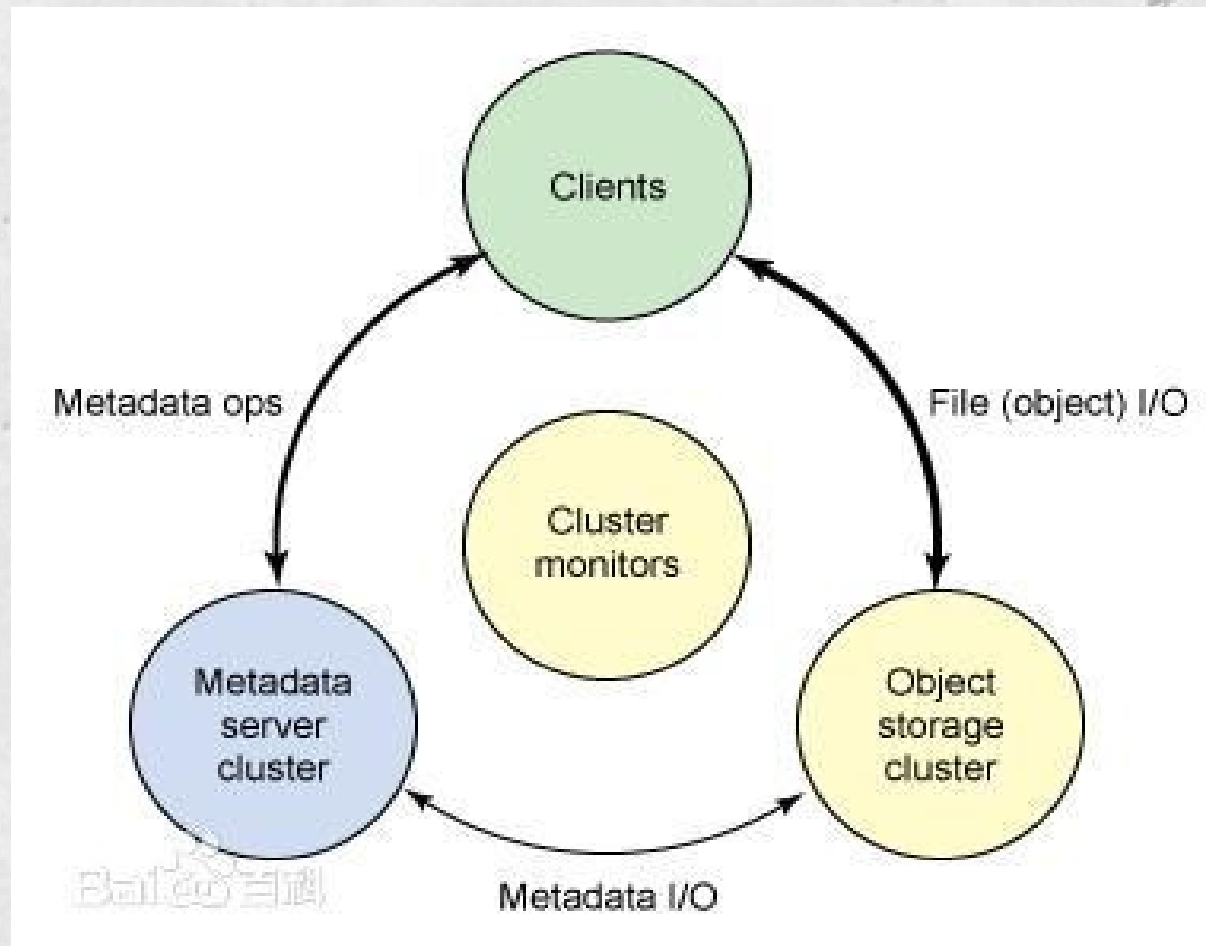
ceph 可以自动数据复制，自动数据平衡，自动错误检测和自动错误恢复

## 3、高扩展性

ceph 可以轻松扩展到数 PB 的容量



# ceph



# ceph

Ceph 生态系统架构可以划分为四部分：

1. Clients：客户端（数据用户）
2. cmds：Metadata server server，元数据服务器（缓存和同步分布式元数据）

# ceph

Ceph 生态系统架构可以划分为四部分：

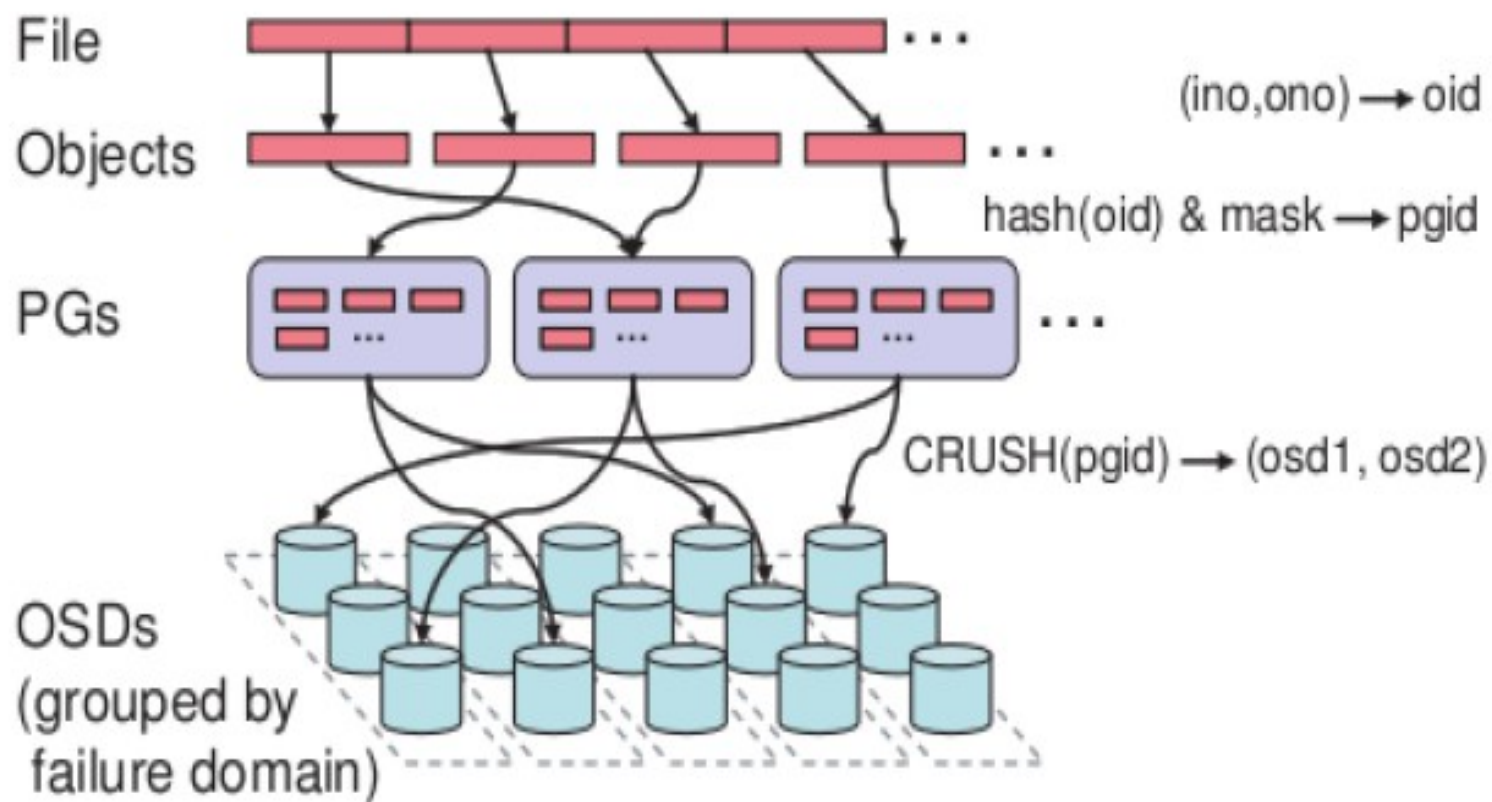
3. cosd : Object storage device, 对象存储集群（将数据和元数据作为对象存储，执行其他关键职能）

4. cmon : Cluster monitors, 集群监视器（执行监视功能）



# ceph

Ceph 数据存储过程：



无论使用哪种存储方式（对象、块、文件系统），  
存储的数据都会被切分成Objects。Objects size  
大小可以由管理员调整，通常为2M或4M。每个对象都会有一个唯一的OID，由ino与ono生成，虽然  
这些名词看上去很复杂，其实相当简单。

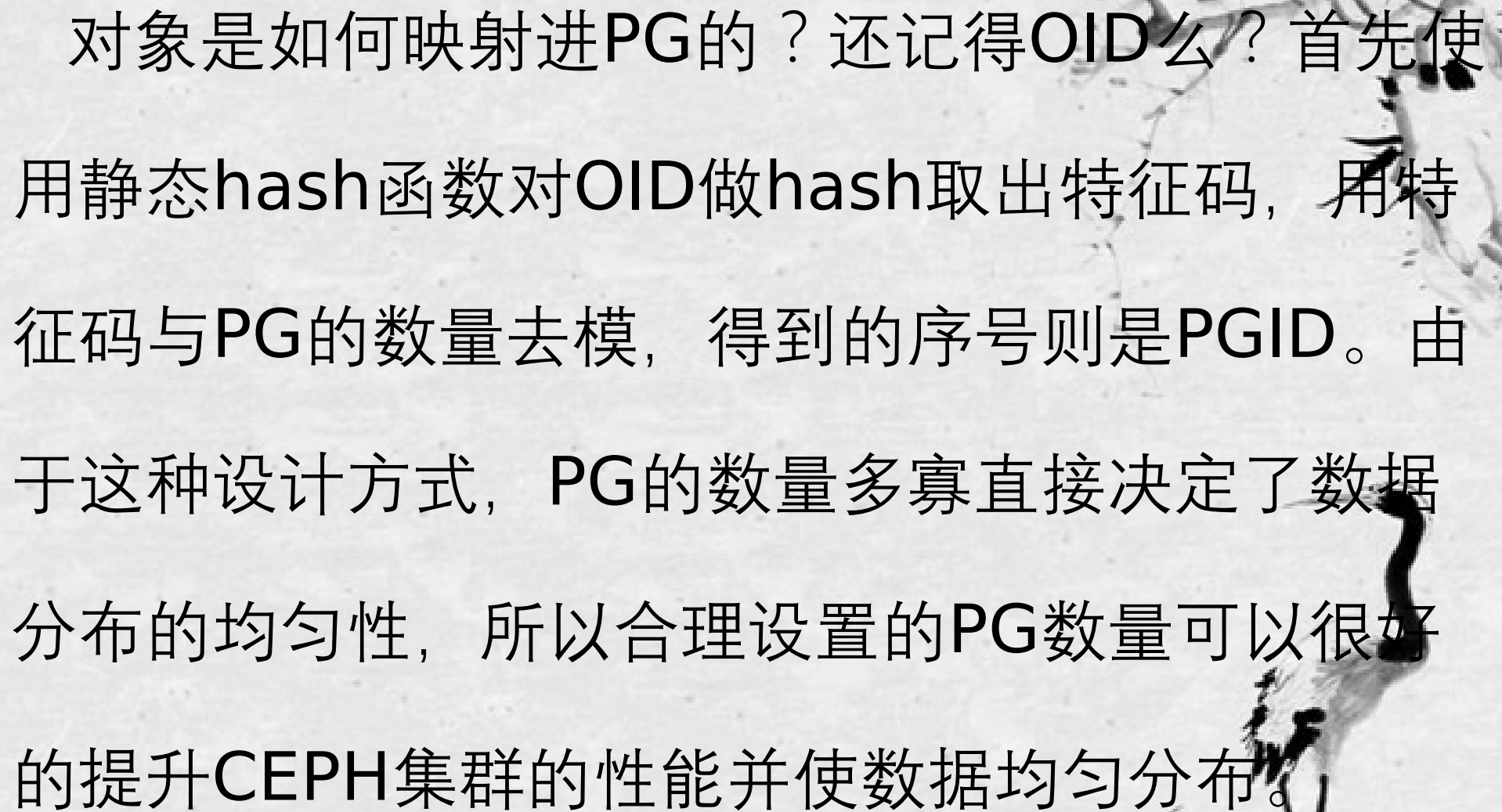




ino即是文件的File ID，用于在全局唯一标识每一个文件，而ono则是分片的编号。比如：一个文件FileID为A，它被切成了两个对象，一个对象编号0，另一个编号1，那么这两个文件的oid则为A0与A1。Oid的好处是可以唯一标示每个不同的对象，并且存储了对象与文件的从属关系。由于ceph的所有数据都虚拟成了整齐划一的对象，所以在读写时效率都会比较高。

但是对象并不会直接存储进OSD中，因为对象的size很小，在一个大规模的集群中可能有几百到几千万个对象。这么多对象光是遍历寻址，速度都是很缓慢的；并且如果将对象直接通过某种固定映射的哈希算法映射到osd上，当这个osd损坏时，对象无法自动迁移至其他osd上面（因为映射函数不允许）。为了解决这些问题，ceph引入了放置组的概念，即PG（placement group）。

PG是一个逻辑概念，我们linux系统中可以直接看到对象，但是无法直接看到PG。它在数据寻址时，类似于数据库中的索引：每个对象都会固定映射进一个PG中，所以当我们要寻找一个对象时，只需要先找到对象所属的PG，然后遍历这个PG就可以了，无需遍历所有对象。而且在数据迁移时，也是以PG作为基本单位进行迁移，ceph不会直接操作对象。



对象是如何映射进PG的？还记得OID么？首先使用静态hash函数对OID做hash取出特征码，用特征码与PG的数量去模，得到的序号则是PGID。由于这种设计方式，PG的数量多寡直接决定了数据分布的均匀性，所以合理设置的PG数量可以很好的提升CEPH集群的性能并使数据均匀分布。

最后PG会根据管理员设置的副本数量进行复制，  
然后通过crush算法存储到不同的OSD节点上（其  
实是把PG中的所有对象存储到节点上），第一个  
osd节点即为主节点，其余均为从节点。



# ceph 实验

环境准备（三个节点都操作）：

主机名	IP 地址	磁盘	功能
ceph-1	192.168.8.25	3x2T	mon+osd*3+depoly
ceph-2	192.168.8.26	3x2T	mon+osd*3
ceph-3	192.168.8.27	3x2T	mon+osd*3

# ceph

## 1、主机名

```
#hostnamectl set-hostname ceph-1
```

ceph-2 同上

ceph-3 同上



# ceph

## 2、统一 /etc/hosts

```
#vim /etc/hosts
```

```
192.168.8.25    ceph-1  
192.168.8.26    ceph-2  
182.168.8.27    ceph-3
```





# ceph

3, 时间同步

4, 准备磁盘, 每个节点添加三块硬盘, 每个 2T

5, ceph-1 节点能够免秘钥登录其他节点

```
#ssh-keygen
```

```
#ssh-copy-id ceph-2
```

```
#ssh-copy-id ceph-3
```



# ceph

## 6, 统一安装源

```
#rm -rf /etc/yum.repo.d/*  
#cd /etc/yum.repo.d  
#wget -O /etc/yum.repos.d/CentOS-  
Base.repo  
http://mirrors.aliyun.com/repo/Centos-7.repo
```



# ceph

```
#wget -O /etc/yum.repos.d/epel.repo  
http://mirrors.aliyun.com/repo/epel-7.repo  
#sed -i '/aliyuncs/d'  
/etc/yum.repos.d/CentOS-Base.repo  
#sed -i '/aliyuncs/d'  
/etc/yum.repos.d/epel.repo  
#sed -i 's/$releasever/7/g'  
/etc/yum.repos.d/CentOS-Base.repo
```



## ceph

```
#vim /etc/yum.repos.d/ceph.repo
```

```
[ceph]
```

```
name=ceph
```

```
baseurl=http://mirrors.163.com/ceph/rpm-jewel/el7/x86_64/
```

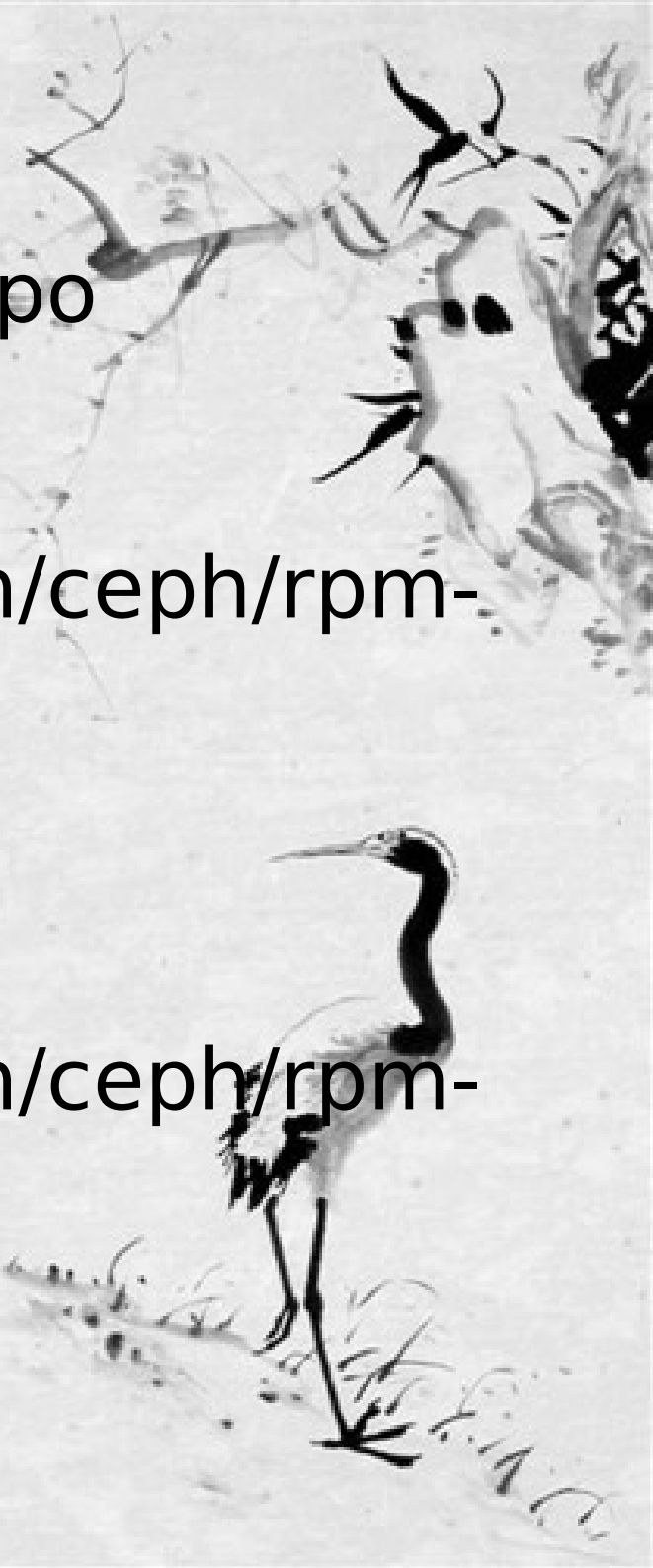
```
gpgcheck=0
```

```
[ceph-noarch]
```

```
name=cephnoarch
```

```
baseurl=http://mirrors.163.com/ceph/rpm-jewel/el7/noarch/
```

```
gpgcheck=0
```



ceph

```
#yum clean all  
#yum makecache  
#yum install ceph ceph-radosgw -y
```



## ceph

在部署节点操作 ( ceph-1 )

```
#yum -y install ceph-deploy
```

查看版本

```
#ceph-deploy -version
```

1.5.34

1、在部署节点创建部署目录并开始部署：

```
#cd
```

```
#mkdir cluster
```

```
#cd cluster
```

```
#ceph-deploy new ceph-1 ceph-2 ceph-3
```



# ceph

在部署节点操作 ( ceph-1 )

```
#ls
```

```
ceph.conf ceph-deploy-ceph.log
```

```
ceph.mon.keyring
```

```
#chown -R ceph. /root/cluster
```

开始部署 monitor:

```
#ceph-deploy mon create-initial
```

查看集群状态：

```
#ceph -s
```



# ceph

开始部署 OSD:

```
#ceph-deploy --overwrite-conf osd prepare  
ceph-1:/dev/sdb ceph-1:/dev/sdc ceph-  
1:/dev/sdd ceph-2:/dev/sdb ceph-  
2:/dev/sdc ceph-2:/dev/sdd ceph-  
3:/dev/sdb ceph-3:/dev/sdc ceph-  
3:/dev/sdd --zap-disk
```



# ceph

开始部署 OSD:

```
#ceph-deploy --overwrite-conf osd activate  
ceph-1:/dev/sdb1 ceph-1:/dev/sdc1 ceph-  
1:/dev/sdd1 ceph-2:/dev/sdb1 ceph-  
2:/dev/sdc1 ceph-2:/dev/sdd1 ceph-  
3:/dev/sdb1 ceph-3:/dev/sdc1 ceph-  
3:/dev/sdd1
```

# ceph

查看集群状态：

```
[root@ceph-1 cluster]# ceph -s
cluster 0248817a-b758-4d6b-a217-11248b098e10
health HEALTH_WARN
    too few PGs per OSD (21 < min 30)
monmap e1: 3 mons at {ceph-1=192.168.57.222:6789/0,ceph-2=192.168.57.223:6789/0,ceph-3=192.168.57.224:6789/0}
election epoch 22, quorum 0,1,2 ceph-1,ceph-2,ceph-3
osdmap e45: 9 osds: 9 up, 9 in
    flags sortbitwise
pgmap v82: 64 pgs, 1 pools, 0 bytes data, 0 objects
    273 MB used, 16335 GB / 16336 GB avail
    64 active+clean
```

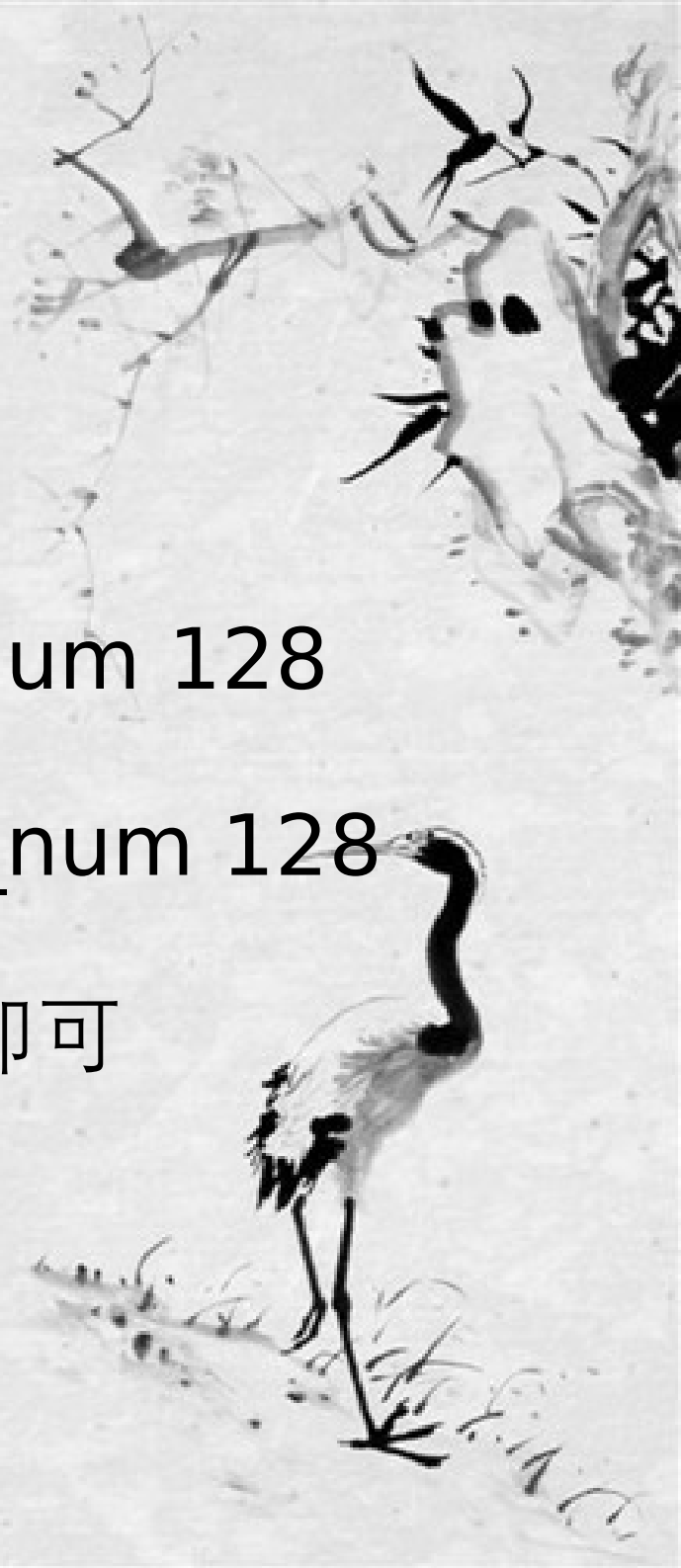
# ceph

增加 rbd 池的 PG 数量：

```
#ceph osd pool set rbd pg_num 128
```

```
#ceph osd pool set rbd pgp_num 128
```

如下图 health HEALTH\_OK 即可



# ceph

```
[root@ceph-1 cluster]# ceph -s
cluster 0248817a-b758-4d6b-a217-11248b098e10
health HEALTH_OK
monmap e1: 3 mons at {ceph-1=192.168.57.222:6789/0,ceph-2=192.168.57.223:6
election epoch 22, quorum 0,1,2 ceph-1,ceph-2,ceph-3
osdmap e49: 9 osds: 9 up, 9 in
flags sortbitwise
pgmap v99: 128 pgs, 1 pools, 0 bytes data, 0 objects
310 MB used, 18377 GB / 18378 GB avail
128 active+clean
```

# ceph

使用 ceph 作为文件系统使用（ceph-1 操作）：

建立元数据服务器

```
#cd cluster/
```

```
#ceph-deploy mds create ceph-1
```

# ceph

创建存储池和 ceph 文件系统

```
#ceph osd pool create test1 64
```

```
#ceph osd pool create test2 64
```

```
#ceph fs new cephfs test2 test1
```



# ceph

生成客户端挂载秘钥文件：

```
#ceph-authtool -p
```

```
ceph.client.admin.keyring > admin.key
```

```
#scp /root/cluster/admin.key
```

```
192.168.8.28:/root/
```

其中 192.168.8.28 是客户端的 IP 地址



# ceph

客户端操作：

```
#chmod 600 admin.key
```

安装客户端软件

```
#yum install ceph-fuse.x86_64 ceph -y
```

```
#mount -t ceph ceph-1:6789:/ /mnt/ -o  
name=admin,secretfile=admin.key
```

查看

```
#df -hT
```





## ceph

删除 ceph 文件系统（ceph-1 节点）

停止 mds 进程

```
# systemctl stop ceph-mds@ceph-1.service
```

将 mds 状态标记为失效

```
#ceph mds fail 0
```

删除文件系统

```
#ceph fs rm cephfs --yes-i-really-mean-it
```

查看文件系统

```
#ceph fs ls
```

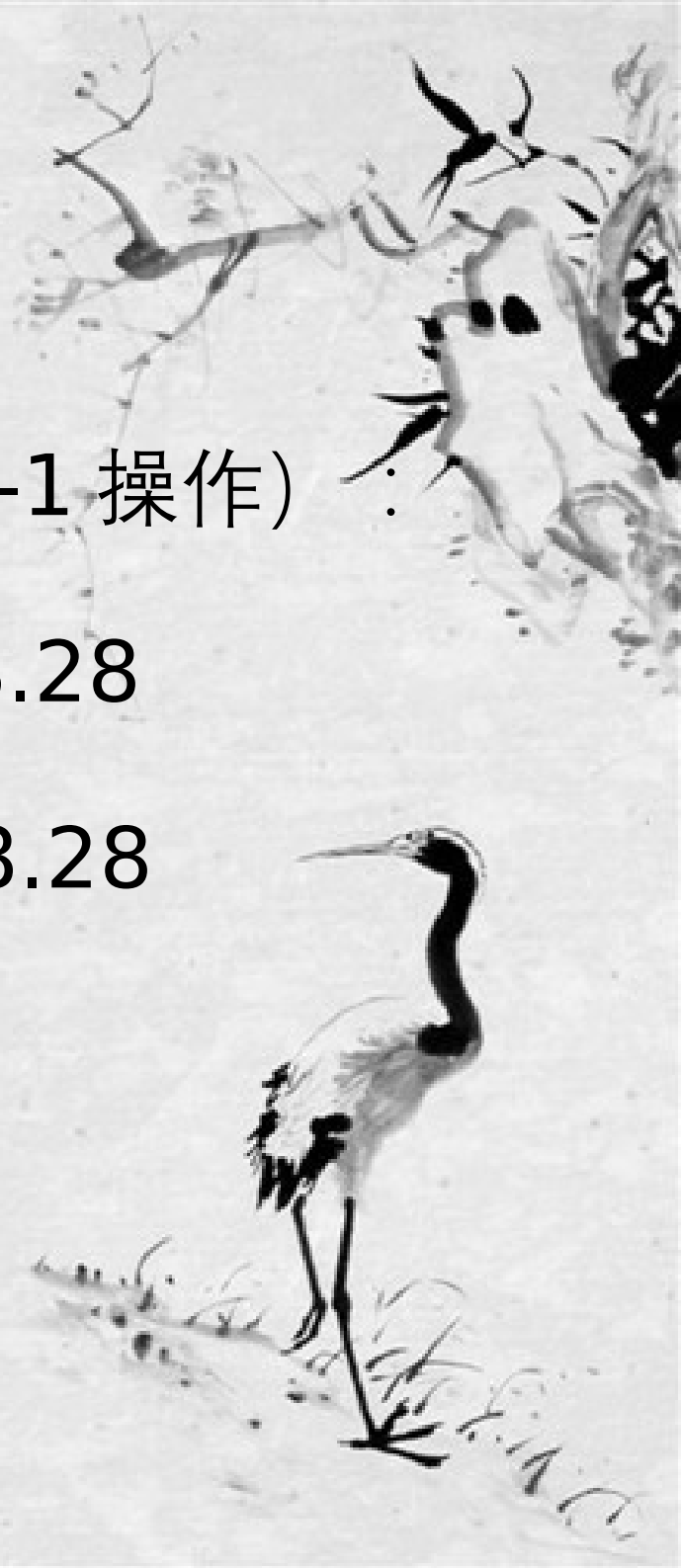


# ceph

使用 ceph 作为块存储使用（ceph-1 操作）：

```
#ceph-deploy install 192.168.8.28
```

```
#ceph-deploy admin 192.168.8.28
```



客户端操作：

ceph

创建磁盘

```
#rbd create disk01 --size 10G --image-  
feature layering
```

将磁盘映射至系统

```
#rbd map disk01
```

查看映射

```
#rbd showmapped
```

查看本地块设备

```
#lsblk
```

