# Ceph官方文档翻译

# Ceph 的介绍

Ceph集群包括文件，快，对象三种存储服务。一个ceph集群至少需要一个mon，一个mgr，一个osd（Object Storage Daemon）。

Monitor：ceph-mon维持集群状态的maps，包括monitor map，manager map，osd map，MDS map，CRUSH map。Monnitor 负责管理进程和客户端之间的认证。至少3个mon是需要来保证高可用服务，需要为奇数个。

Manager：ceph-mgr 进程负责追踪集群的当前状态指标信息，包括存储利用率，当前性能指标和系统负载。该进程也拥有基于python模块来管理集群信息，包括一个基于web的ceph Dashboard，REST API。高可用至少2个mgr。

OSD：ceph-osd负责存储数据，数据副本，数据恢复平衡，通过osd之间的心跳，提供一些监控信息给mgr和mon。至少3个osd用于高可用。

MDS：ceph-mds负责存储ceph 文件系统的元数据。对于块和对象服务，则不需要。允许POSIX文件系统用户执行基本的命令例如ls,find，而不会给集群造成太大的负担。

Ceph将数据存储为逻辑pool中的object，根据CRUSH算法，ceph计算哪个pg应该包含object，哪个osd进程应该存储pg。

## 硬件推荐

CPU: cephFs是CPU密集型的，应当单独拥有一个cpu核。OSD主要运行rados服务，可能需要1到2个多个cpu核。Mon和mgr则使用轻量的cpu处理器就行。

RAM：一般集群的MON/mgr节点的RAM应该是64GB，对于拥有数百个OSD大型集群来说，128Gb是一个合理的值。MON-MGR内存需求一般随着集群的大小而扩展。MDS一般1G作为最小的。OSD中的bluestore使用其自己的内存来缓存数据，而不是操作系统的页缓存。在bluestore中，可以通过参数osd\_memory\_target来配置OSD使用的内存数。将osd\_memory\_target设置为2-4GB之间可能会导致PG降级。一般默认值为4GB，大于4GB可以提高集群性能。

数据存储：每块硬盘最小为1T，用硬盘价格除以每千兆字节，得出每千兆字节价格。运行多个OSD在一个SAS/SATA硬盘上不是一个明智的选择，但是NVME盘可以这样做。

NETWORK：推荐使用10Gbps网络网卡，1TB的数据处理需要20分钟左右，PG恢复速度也会快很多。

## 文档关于ceph

git clone <https://gitub.com/ceph/ceph>

doc/rados: ceph集群

doc/rbd: 块设备

doc/radosgw: 对象服务

doc/cephfs: 文件存储服务

编译ceph源码安装ceph：

# 安装ceph

1. Cephadm: cephadm仅支持最新版本的ceph Octopus，使用cli和dashboard gui。
2. Rook部署和管理ceph在kubernetes平台上，仅支持Nautils。
3. Ceph-ansible部署和管理集群
4. 手动安装

主要介绍一下如何手动安装ceph：

可以选择下载编译好的ceph包，也可以选择下载源码自己编译。

1. Download Keys to avoid a security warning
2. sudo rpm --import 'https://download.ceph.com/keys/release.asc'
3. 也可以源下设置为不验证。
4. 配置ceph.repo，添加到/etc/yum.repos.d/

其中ceph-release可以为nautils, distro可以为el7

1. [ceph]
2. name=Ceph packages **for** $basearch
3. baseurl=https://download.ceph.com/rpm-{ceph-release}/{distro}/$basearch
4. enabled=1
5. gpgcheck=1
6. gpgkey=https://download.ceph.com/keys/release.asc
8. [ceph-noarch]
9. name=Ceph noarch packages
10. baseurl=https://download.ceph.com/rpm-{ceph-release}/{distro}/noarch
11. enabled=1
12. gpgcheck=1
13. gpgkey=https://download.ceph.com/keys/release.asc
15. [ceph-source]
16. name=Ceph source packages
17. baseurl=https://download.ceph.com/rpm-{ceph-release}/{distro}/SRPMS
18. enabled=0
19. gpgcheck=1
20. gpgkey=https://download.ceph.com/keys/release.asc
21. 安装epel源以及第三方库，可能很慢。
22. yum install snappy leveldb gdisk python-argparse gperftools-libs
23. yum install ceph ceph-radosgw

其中leveldb，若找不到包则需要自己编译安装。

# 配置ceph

## Add mon

/etc/ceph/ceph.conf的基本配置

Fsid是一个集群的唯一id，ceph含有一个集群名字，可通过ceph –cluster {cluster-name}来设置，不同集群，其配置文件为:cluster-name.conf。

每一个monitor实例都有一个唯一的name，一般是host name。

Monitor map要求fsid, 集群名，至少一个host name和其IP。

1. uuidgen  产生集群fsid
2. [global]
3. fsid = a7f64266-0894-4f1e-a635-d0aeaca0e993
4. mon initial members = node1
5. mon host = 192.168.56.7
6. cluster network = 192.168.56.0/24
7. auth cluster required = cephx
8. auth service required = cephx
9. auth client required = cephx
10. osd pool **default** size = 2
11. osd pool **default** min size = 1
12. osd pool **default** pg num = 64
13. osd pool **default** pgp num = 64
14. osd crush chooseleaf type = 1
15. 创建一个mon用户以及 keyring

Monitor之间通过密钥进行通信，在引导初始的monitor时需要提供密钥。

1. sudo ceph-authtool --create-keyring /tmp/ceph.mon.keyring --gen-key -n mon. --cap mon 'allow \*'
2. 创建一个client.admin keyring

为了使用ceph cli tools，必须要有一个client.admin user和keyring，用于登陆集群客户端。

1. sudo ceph-authtool --create-keyring /etc/ceph/ceph.client.admin.keyring --gen-key -n client.admin --cap mon 'allow \*' --cap osd 'allow \*' --cap mds 'allow \*' --cap mgr 'allow \*'
2. 创建一个bootstrap-osd keyring
3. sudo ceph-authtool --create-keyring /var/lib/ceph/bootstrap-osd/ceph.keyring --gen-key -n client.bootstrap-osd --cap mon 'profile bootstrap-osd' --cap mgr 'allow r'
4. 将bootstrap-osd和client.admin. keyring添加到mon中,并改变mon keyring的拥有者。
5. ceph-authtool /tmp/ceph.mon.keyring –import-keyring /etc/ceph/ceph.client. admin.keyring
6. ceph-authtool /tmp/ceph.mon.keyring –import-keyring /var/lib/ceph/bootstrap-osd/ceph.keyring
7. sudo chown ceph:ceph /tmp/ceph.mon.keyring
8. 新建monmap
9. monmaptool --create --add node1 192.168.0.1 --fsid a7f64266-0894-4f1e-a635-d0aeaca0e993 /tmp/monmap
10. 创建mon目录，存储mon密钥，以及填充monitor
11. sudo -u ceph mkdir /var/lib/ceph/mon/ceph-node1
12. sudo -u ceph ceph-mon --mkfs -i node1 --monmap /tmp/monmap --keyring /tmp/ceph.mon.keyring
13. 启动monitor
14. sudo systemctl start ceph-mon@node1
15. sudo systemctl enable ceph-mon@node1
16. ceph -s
17. add new mon from new node

sudo mkdir /var/lib/ceph/mon/$ceph-mon-id

mkdir $tmp

ceph auth get mon. -o $tmp/$keyring

ceph mon getmap -o $tmp/$map

ceph-mon -i $mon-id --mkfs --monmap $tmp/$map --keyring $tmp/$keyring

ceph-mon -i $mon-id --public-addr $ip

启动monitor

## rm mon

集群正常情况下

service ceph -a stop mon. $mon-id

ceph mon remove $mon-id

注意点：

Ceph clients and other Ceph daemons use ceph.conf to discover monitors. However, monitors discover each other using the monitor map, not ceph.conf.

如何修改mon的ip 地址：

方式一：rm mon, add mon.

方式二：修改monmap

1. ceph mon getmap -o $tmp/$map
2. monmaptool –print $tmp/$map
3. monmaptool –rm a –rm b –rm c $tmp/$map
4. monmaptool –add a 10.1.0.1 –add b 10.1.0.2 $tmp/$map
5. 停止所有的mon
6. Ceph-mon -i $mon-id –inject-monmap $tmp/map
7. 重启所有的mon.

## ADD mgr

1. ceph auth get-or-create mgr.node1 mon 'allow profile mgr' osd 'allow \*' mds 'allow \*' > /var/lib/ceph/mgr/ceph-node1/keyring
2. ceph-mgr -i node1
3. ceph mon enable-msgr2
4. sudo systemctl start ceph-mgr@node1
5. sudo systemctl enable ceph-mgr@node1

## ADD osd

Bluestore创建osd的过程:

ceph-volume lvm zap /dev/sdb –destroy:清楚osd上的pv,lv,vg信息

sudo ceph-volume lvm create --data /dev/sdb

1. 这是该命令的详细解析
2. Uuidgen //创建一个uuid给osd
3. ceph-authtool --gen-print-key  //创建一个osd key
4. 创建osd的fsid
5. ceph --cluster ceph --name client.bootstrap-osd --keyring /var/lib/ceph/bootstrap-osd/ceph.keyring -i - osd **new** ee661cef-02c2-43aa-94ba-5c1e5a05c180
6. 创建pv,vg，对/dev/sdb
7. vgcreate --force --yes ceph-e9e4dee8-ad57-4117-9818-a4a9ae318645 /dev/sdb
8. 创建lv，根据vg
9. lvcreate --yes -l 5119 -n osd-block-ee661cef-02c2-43aa-94ba-5c1e5a05c180 ceph-e9e4dee8-ad57-4117-9818-a4a9ae318645
10. 产生密钥osd
11. ceph-authtool --gen-print-key
12. 将tmpfs文件系统挂载到ceph-0目录下
13. mount -t tmpfs tmpfs /var/lib/ceph/osd/ceph-0
14. restorecon /var/lib/ceph/osd/ceph-0
15. 赋予拥有者ceph
16. chown -h ceph:ceph /dev/ceph-e9e4dee8-ad57-4117-9818-a4a9ae318645/osd-block-ee661cef-02c2-43aa-94ba-5c1e5a05c180
17. chown -R ceph:ceph /dev/dm-3
18. 建立Lvpath与osd block的软连接
19. ln -s /dev/ceph-e9e4dee8-ad57-4117-9818-a4a9ae318645/osd-block-ee661cef-02c2-43aa-94ba-5c1e5a05c180 /var/lib/ceph/osd/ceph-0/block
20. 产生activate.monmap
21. /bin/ceph --cluster ceph --name client.bootstrap-osd --keyring /var/lib/ceph/bootstrap-osd/ceph.keyring mon getmap -o /var/lib/ceph/osd/ceph-0/activate.monmap
22. 新建osd密钥文件
23. ceph-authtool /var/lib/ceph/osd/ceph-0/keyring --create-keyring --name osd.0 --add-key AQDe29hfWEc4NBAAju4Ec6WadXy3C26cYd9+0w==
24. chown -R ceph:ceph /var/lib/ceph/osd/ceph-0/keyring
25. chown -R ceph:ceph /var/lib/ceph/osd/ceph-0/
26. ceph-osd --cluster ceph --osd-objectstore bluestore --mkfs -i 0 --monmap /var/lib/ceph/osd/ceph-0/activate.monmap --keyfile - --osd-data /var/lib/ceph/osd/ceph-0/ --osd-**uuid** ee661cef-02c2-43aa-94ba-5c1e5a05c180 --setuser ceph --setgroup ceph
27. chown -R ceph:ceph /var/lib/ceph/osd/ceph-0
28. ceph-bluestore-tool --cluster=ceph prime-osd-dir --dev /dev/ceph-e9e4dee8-ad57-4117-9818-a4a9ae318645/osd-block-ee661cef-02c2-43aa-94ba-5c1e5a05c180 --path /var/lib/ceph/osd/ceph-0 --no-mon-config
29. ln -snf /dev/ceph-e9e4dee8-ad57-4117-9818-a4a9ae318645/osd-block-ee661cef-02c2-43aa-94ba-5c1e5a05c180 /var/lib/ceph/osd/ceph-0/block
30. chown -h ceph:ceph /var/lib/ceph/osd/ceph-0/block
31. chown -R ceph:ceph /dev/dm-3
32. chown -R ceph:ceph /var/lib/ceph/osd/ceph-0
33. systemctl enable ceph-volume@lvm-0-ee661cef-02c2-43aa-94ba-5c1e5a05c180
34. systemctl enable --runtime ceph-osd@0
35. systemctl start ceph-osd@0

添加osd的方式二：

Ceph-volume lvm prepare --data $/dev/path

Ceph-volume lvm list

Ceph-volume lvm activate $osd-id $osd-fsid

替换osd osd

While ! ceph osd safe-todestroy osd.$id ; do sleep 10; done

Ceph osd destroy $id –yes-i-really-mean-it

ceph-volume lvm create --osd-id $id –data $/dev/sdx

移除osd

Ceph osd out $id

Systemctl stop ceph-osd@$id

ceph osd purge $osd-id --yes-i-really-mean-it

ceph osd

# cephadm

cephadm是管理和部署集群的工具，只支持Octopus v15.2.0，老的版本不支持。

# ceph cluster

ceph cluster由两种进程组成：osd, mon。一个最小的集群应该有两个osd和一个mon。

osd的后端存储：filestore和bluestore。L版之后的ceph均用bluestore。Bluestore的关键特点：

1. 直接管理存储设备，无需文件系统作为抽象层。
2. 用rocksdb管理元数据，键值数据库映射对象名到存储地址。
3. 数据读写入时的数据校验。
4. 数据行压缩。
5. 元数据分层存储。
6. Copy-on-write：节省内存，复制对象时，内存中始终只有一份数据。快照所必须的技术。

Filestore:依靠标准的文件系统xfs和leveldb的组合。

Ceph.conf:

Sections: global mon mgr mds osd client, 例如client.rgw

运行时修改配置参数：

ceph tell <name> config set <option> <value>

ceph tell osd.1 config set debug\_osd 20

通过/var/run/ceph里面的套接字设置值

Ceph daemon osd.4 config set debug\_osd 20

显示参数设置

Ceph daemon osd.1 config show

Ceph network set:

Ceph Clients make requests directly to Ceph OSD Daemons. Ceph OSD Daemons perform data replication on behalf of Ceph Clients, which means replication and other factors impose additional loads on Ceph Storage Cluster networks。

By default, daemons [bind](https://docs.ceph.com/en/latest/rados/configuration/network-config-ref/#bind) to ports within the 6800:7300 range。Ceph Monitors listen on ports 3300 and 6789 by default。 Ceph assumes a public network with all hosts operating on it unless you specifically configure a cluster network.

Cephx: 认证

When cephx is enabled, Ceph will look for the keyring in the default search path, which includes /etc/ceph/$cluster.$name.keyring.

When you run Ceph with authentication enabled, ceph administrative commands and Ceph Clients require authentication keys to access the Ceph Storage Cluster.

the Ceph Storage Cluster daemons (i.e., ceph-mon, ceph-osd, ceph-mds and ceph-mgr) must authenticate with each other. Valid settings are cephx or none.

Ceph stores daemons keyrings inside their data directory.

Monitor settings:

Ceph client可以通过读取 cluster map，从而获取所有mon, osd, mds。在ceph client能够读写ceph osd时，均是利用cluster map以及crush rule，从而计算对象的location，从而直接与osd进行talk。

Mon负责维持cluster map以及logging service。Cluster map: mon map, osd map, pg map, mds map。

集群某个part发生问题，可能会导致某个对象不可访问，但并不意味着无法访问其它的对象。如果某一个osd down或者out， ceph将会将pg迁移到其它的osd上。

Crush map分配pg给osd，根据pool的副本数。比如3副本，则将pg的每个副本分配给不同的osd。Osd处在不同的故障域。

Ceph pg dump

Ceph ph map $pg-num

osdmap e27 pg 1.5f (1.5f) -> up [1] acting [1]

up set, act set,通常情况下两者是相同的，若集群pg处于迁移，osd处于恢复时，则可能不一致。Pg必须处于active的state，才可被写入数据。

创建pool后，会创建pgs，peering.一旦pgs创建完成，osd中含有的pg中对象的数据和元数据的状态会一致。但是peering完成不意味着每个副本的内容都是最新的。Peering完成之后就是active状态，当是clean时表明数据均保持一致。

当客户端写数据到主osd时，主osd负责将副本写入从osd，在得到从osd成功创建副本对象之前处于degraded状态。

当新的osd加入集群时，crush会给新的osd分配pgs，会给osd带来过大的负载。这个过程可以在后台运行，backfilling过程。

Backfilling\_wait: backfilling; backfill\_toofull表明因为存储容量不足的问题，从而无法完成。Osd\_max\_backfills表明当前最大数目，backfill ratio表明拒绝backfill时的osd容量。

Remapped: 当pgs改变时，数据会从老的acting set迁移到新的acting set。

Ceph osd crush dump

Ceph osd crush rule list

Ceph osd crush rule dump

查找对象的location

Ceph osd map $poolname $objectname $namespace

Rados put $objectname $object-path –pool=$poolname –namespace=$name

Rados -p $poolname ls

Rados rm $objectname –pool=$poolname –namespace=$name

Mon和osd之间interaction

Osd之间会检查心跳连接，以一个小于6秒的随机间隔时间。如果一个osd20s均没有心跳。Osd将会向mon报告，某一个osd down。参数Osd heart grace = 20。

如果osd 请求peer 另一个osd，失败，则会向mon报告请求更新map。

Osd mon heartbeat interval = 30

如果一个osd 不report mon, 则会认为osd down。

**Debug/logging**

ceph tell osd.0 config set debug\_osd 0/5

Bluestore:

只管理单个存储设备

ceph-volume lvm prepare --bluestore --data <device>

同时管理，wal和db设备：wal用于日志，db用于metadata

ceph-volume lvm prepare --bluestore --data <device> --block.wal <wal-device> --block.db <db-device>

--data can be a Logical Volume using vg/lv notation. Other devices can be existing logical volumes or GPT partitions.

要求wal, db均是ssd，而primary device是HDD.

如果设备的lv已经创建好了，以下操作也可以。

ceph-volume lvm create --bluestore --data ceph-vg/block-lv

推荐使用block.db和block一起用

例如使用4个osd均带有block.db

$ vgcreate ceph-block-0 /dev/sda

$ vgcreate ceph-block-1 /dev/sdb

$ vgcreate ceph-block-2 /dev/sdc

$ vgcreate ceph-block-3 /dev/sdd

$ lvcreate -l 100%FREE -n block-0 ceph-block-0

$ lvcreate -l 100%FREE -n block-1 ceph-block-1

$ lvcreate -l 100%FREE -n block-2 ceph-block-2

$ lvcreate -l 100%FREE -n block-3 ceph-block-3

$ vgcreate ceph-db-0 /dev/sdx

$ lvcreate -L 50GB -n db-0 ceph-db-0

$ lvcreate -L 50GB -n db-1 ceph-db-0

$ lvcreate -L 50GB -n db-2 ceph-db-0

$ lvcreate -L 50GB -n db-3 ceph-db-0

$ ceph-volume lvm create --bluestore --data ceph-block-0/block-0 --block.db ceph-db-0/db-0

$ ceph-volume lvm create --bluestore --data ceph-block-1/block-1 --block.db ceph-db-0/db-1

$ ceph-volume lvm create --bluestore --data ceph-block-2/block-2 --block.db ceph-db-0/db-2

$ ceph-volume lvm create --bluestore --data ceph-block-3/block-3 --block.db ceph-db-0/db-3

The general recommendation is to have block.db size in between 1% to 4% of block size

For RGW workloads, it is recommended that the block.db size isn’t smaller than 4% of block。For RBD workloads, 1% to 2% of block size is usually enough。

MANUAL CACHE SIZING

The amount of memory consumed by each OSD for BlueStore caches is determined by the bluestore\_cache\_size configuration option.

bluestore\_cache\_size\_hdd

Description

The default amount of memory BlueStore will use for its cache when backed by an HDD.

Type

Unsigned Integer

Required

Yes

Default

1 \* 1024 \* 1024 \* 1024 (1 GB)

BlueStore supports inline compression using snappy, zlib, or lz4.

ceph osd pool set <pool-name> compression\_algorithm <algorithm>

ceph osd pool set <pool-name> compression\_mode <mode>

ceph osd pool set <pool-name> compression\_required\_ratio <ratio>

ceph osd pool set <pool-name> compression\_min\_blob\_size <size>

ceph osd pool set <pool-name> compression\_max\_blob\_size <size>

CLUSTER OPERATION

sudo systemctl start ceph.target 启动所有进程

udo systemctl start ceph-osd.target 启动某一种类型的进程

sudo systemctl start ceph-mon.target

sudo systemctl start ceph-mds.target

sudo systemctl start ceph-osd@{id} 启动某一个进程

sudo systemctl start ceph-mon@{hostname}

sudo systemctl start ceph-mds@{hostname}