### ceph存储使用手册

**前言：**

1.蓝色字体表示命令行命令，正式执行时不要复制前面的#号，#号只是提示应该使用root权限操作

2.绿色字体表示注释，有时注释太多就不用绿色表示了

3.注意：本文档的所有操作请先在测环境进行实践，请不要直接在真实的服务器中操作！

**版权声明**：

本文档以开源的形式发布，所有条款如下：

1. 无担保：作者不保证文档内容的准确无误，亦不承担由于使用此文档所导致的任何后果

2. 自由使用：任何人可以出于任何目的而自由地 阅读/链接/打印/转载/引用/再创作 此文档，无需任何附加条件

若您 阅读/链接/打印/转载/引用/再创作 本文档，则说明接受以上2个条款。

作者：李茂福

更新日期：2023年10月31日

**★第0章、ceph各版本发布时间**

ceph官网： https://ceph.com/en/

版本查看： https://docs.ceph.com/en/latest/releases/

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 代号 | 版本 | 发布日期 | EOL日期 | 对应红帽ceph |  |
| Dumpling | 4 0.67.11 | 2013-08-01 | 2015-05-01 |  |  |
| Emperor | 5 0.72.2 | 2013-11-01 | 2014-05-01 |  |  |
| Firefly | 6 0.80.11 | 2014-05-01 | 2016-04-01 |  |  |
| Giant | 7 0.87.2 | 2014-10-01 | 2015-04-01 |  |  |
| Hammer | 8 0.94.10 | 2015-04-01 | 2017-08-01 |  |  |
| Infernalis | 9 | 2015-11-01 | 2016-04-01 |  |  |
| Jewel | 10 | 2016-04-01 | 2018-07-01 | rhcs 2 |  |
| Kraken | 11 | 2017-01-01 | 2017-08-01 |  |  |
| Luminous | 12 | 2017-08-01 | 2020-03-01 | rhcs 3 | 命令与前面版本开始有差异了 |
| Mimic | 13 | 2018-06-01 | 2020-07-22 |  |  |
| Nautilus | 14 | 2019-03-19 | 2021-06-30 | rhcs 4 | 开始支持自动调整PG数量 |
| Octopus | 15 | 2020-03-23 | 2022-08-09 |  | 开始支持使用cephadm部署 |
| Pacific | 16 | 2021-03-31 | 2023-10-01 | rhcs 5 |  |
| Quincy | 17 | 2022-04-19 | 2024-06-01 |  |  |
| Reef | 18 | 2023-08-07 | 2025-08-01 |  |  |
|  | 19 |  |  |  |  |

从Hammer之后版本命名使用新的命名策略，之前一直是 0.2，0.3，0.x

主版本号为8，9，10......，次版本号0表示开发版，1表示候选版，2表示稳定版

在Octopus版本后使用cephadm来部署ceph集群，如果使用cephadm部署，在后期新的版本升级时，可以做到完全自动化，并可以通过ceph -W cephadm查看升级进度，升级完成后，无法降级，升级时请不要跨版本升级

**★第1章、ceph各组件及原理**

**网络存储类型：**

|  |  |  |
| --- | --- | --- |
| 存储服务 | 类型 | 存储类型 |
| NFS，Samba，CephFS | NAS（Network Attacked Storage） | 目录共享（文件系统） |
| iSCSI，RBD | SAN（Storage Area Network） | 块设备，磁盘分区、lvm逻辑卷、文件io共享 |
| aws-S3，ceph-RGW | OBS（Object Storage Service） | 对象存储，以桶（bucket）来组织文件对象 |

Ceph是一个开源的分布式存储系统，可提供 对象存储、块设备、文件系统 的存储服务

ceph是一个对象（object）式存储系统，它把每一个待管理的数据流（文件等数据）切割为若干个固定大小（默认4MB）的对象数据，并以其为原子单元完成数据的读写。

对象数据的底层存储服务是由多个存储主机（host）组成的存储集群，该集群也被称为RADOS存储集群

RADOS（reliable automatic distributed object store），即可靠的、自动化的、分布式的存储集群

**★ceph的构成**

|  |  |
| --- | --- |
| pool | 存储池，存储池的大小取决于底层的存储空间大小（所有osd对应的磁盘的空间之和） |
| PG | （Placement Group）放置组，PG是组成存储池的基本单位。一个pool内部可以有多个PG存在，一个pool中有多少个PG可以通过公式计算。同一个PG内的所有对象都会放到相同的磁盘上。  一个对象只能属于一个PG，多副本情况下一个PG对应多个osd |
| OSD | （Object Storage Daemon）存储守护进程，ceph集群通过管理OSD来管理物理硬盘，服务器上的每一块数据盘都会相应地去运行一个OSD进程。一个osd对应一个数据磁盘。 |
| ceph-Mon | （Monitor）ceph监视器，维护OSD和PG的集群状态，一个ceph集群至少要有一个mon，可以是奇数个数量的mon进程实例，如一三五七等。每台服务器只起一个mon进程 |
| ceph-Mgr | Ceph Manager守护程序负责跟踪运行时指标和Ceph集群的当前状态，包括存储利用率，当前性能指标和系统负载。Ceph Manager守护程序还托管基于python的模块来管理和公开Ceph集群信息，包括基于Web的Ceph仪表板和REST API。高可用性通常至少需要两个mgr管理器。mgr组件不是必须的，只有需要查看信息状态时才部署mgr实例。 |
| MDS | ceph元数据服务器，代表ceph文件系统（NFS/CephFS）存储元数据，（即Ceph块设备和Ceph对象存储不使用MDS） |
| Ceph管理节点 | ceph的常用管理接口是一组命令行工具程序，例如rados、ceph、rbd等命令；安装了ceph-common工具命令的节点为管理节点，同时在ceph里将此节点标记为\_admin |
| librados | RADOS存储集群的API，支持大多数编程语言客户端调用 |

**OSD**

osd负责处理集群数据的复制、恢复、回填、再均衡，并向其他osd守护进程发送心跳，然后向Mon提供一些监控信息。当Ceph存储集群设定数据有两个副本时（一共存两份），则至少需要三个OSD守护进程即三个OSD节点，集群才能达到active+clean状态，实现冗余和高可用。

在Ceph中，如果要写数据，只能向主OSD写，然后再由主OSD向从OSD同步地写，只有当从OSD返回结果给主OSD后，主OSD才会向客户端报告写入完成的消息。如果要读数据，不会使用读写分离，而是也需要先向主OSD发请求，以保证数据的强一致性。

OSD权重是主/从OSD节点的选择 的唯一标准，而权重又以对应硬盘剩余存储容量作为唯一决定因素，所以磁盘剩余存储容量是数据分布和读写操作中选择主次OSD的唯一决定因素

**Monitor**

用于维护集群状态映射（maintains maps of the cluster state），比如ceph集群中有多少存储池、每个存储池有多少PG以及存储池和PG的映射关系等；此外监视器还负责管理守护程序和 客户端之间的身份验证（认证使用cephx协议），通常至少需要三个监视器才能实现冗余和高可用性

**CRUSH算法**

（Controllers replication under scalable hashing）可控的、可复制的、可伸缩的一致性hash算法

Ceph使用CRUSH算法来存放和管理数据，Ceph使用CRUSH算法来准确计算数据应该被保存到哪里，以及应该从哪里读取。和保存元数据不同，是CRUSH按需计算出元数据，因此它就消除了对中心式的服务器/网关的需求，它使得Ceph客户端能够计算出元数据，该过程也称为CRUSH查找，然后和OSD直接通信。

如果是把对象直接映射到OSD之上会导致对象与OSD的对应关系过于紧密和耦合，当OSD由于故障发生变更时将会对整个ceph集群产生影响。于是ceph将一个对象映射到RADOS集群的时候分为两步走：

首先使用一致性hash算法将对象名称映射到PG，

然后将PG ID基于CRUSH算法映射到OSD即可查到对象

**★Ceph相较于其他存储方案的优势**

CRUSH算法：

CURSH是ceph的两大创新之一（另一大就是去中心化），ceph摒弃了传统的集中式存储元数据寻址的方案，转而使用CRUSH算法计算的方式完成数据的寻址操作

crush算法有强大的扩展性（即高扩展性），理论上支持上千个存储节点规模

高可用：

数据副本数可以灵活调整

可以通过crush算法指定副本的物理存放位置以分割故障域，支持数据强一致性

支持多种故障场景自动尝试进行修复

支持多份强一致性副本，副本能够垮主机、机架、机房、数据中心存放，安全可靠

存储节点可以自管理、自动修复。无单点故障，容错性强

高性能：

因为是多个副本，因此在读写操作时能够做到高度并行化，理论上，节点越多，整个cpeh集群的IOPS和吞吐量就越高

ceph客户端读写数据可直接与存储设备OSD交互，在块存储和对象存储中无需元数据MDS服务

特性丰富：

支持三种存储接口：对象存储，块存储，文件存储

支持自定义接口，支持多种语言驱动

**★ceph上层接口**

基于LIBRADOS层开发的三个接口，其作用是在librados库的基础上提供抽象层次更高、更便于应用或客户端使用的上层接口。

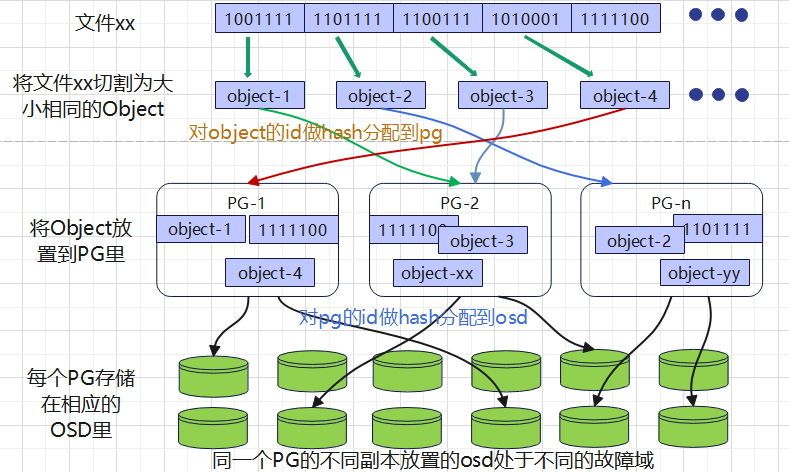
①RADOS GW（简称RGW）提供对象存储服务，是一套基于RESTFUL协议的网关，支持对象存储，兼容S3和Swift

②RBD提供分布式的块存储设备接口，主要面向虚拟机提供虚拟磁盘，可以被映射、格式化，像磁盘一样挂载到服务器使用。

③CephFS是一个POSIX兼容的分布式文件系统，依赖MDS来跟踪文件层次结构，基于librados封装原生接口，它跟传统的文件系统如Ext4是一个类型的，区别在于分布式存储提供了并行化的能力，像NFS等也是属于文件系统存储

**★ceph底层存储过程**

无论使用哪种存储方式（对象，块，文件），存储的数据在底层保存时，都会被切分成一个个大小固定的对象（Objects），对象大小可以由管理员自定义调整，RADOS中基本的存储单位就是Objects，一般为2MB或4MB（最后一个对象大小有可能不同）



文件（File）被切割成大小固定的Objects（每个默认4MB），再根据对象的id进行hash计算，得到的结果对pg个数取余，将这些对象分配到对应的PG中，然后各个PG会通过CRUSH算法分配到对应的osd上（每个PG：一副本就复制相同的数据(object)到对应的1个osd上，两副本就复制到对应的2个osd上，依此类推）

**★pg数量计算公式**

Total PGs = ( Total\_OSD\_number \* Target PGs per OSD ) / max\_replication\_count / pool\_count

（Target PGs per OSD 如果osd将来不扩展则置100，将来扩展则置200）

即：总PG数 = 总osd数量 \* 每osd目标pg数量 / 最大文件副本数 / pool数量

**例：**

24个磁盘，将来不扩展（100），最大副本3，pool数量8 ，则总pg数为

Total PGs = 24 x 100 /3 /8 = 100

常用值（3副本数）：

|  |  |  |
| --- | --- | --- |
| pg数量 | osd数量（将来不增加） | osd数量（将来增加） |
| 128 | 3.84 | 1.92 |
| 256 | 7.68 | 3.84 |
| 512 | 15.36 | 7.68 |
| 1024 | 30.72 | 15.36 |
| 2048 | 61.44 | 30.72 |
| 4096 | 122.88 | 61.44 |

大于50个osd，建议手动计算

**★从Nautilus版本开始，支持自动扩展pg数量，不需要再手动指定了。**

pg\_autoscaler模块最早在Nautilus（14.x）版本中引入，pg\_autoscaler可以根据池使用情况和用户设置的调整来自动调整集群中PG放置组的数量。

**★第2章、centos-stream8安装ceph集群**

**★规划及部署**

官方指导文档：https://access.redhat.com/documentation/zh-cn/red\_hat\_ceph\_storage/4/html/file\_system\_guide/index

**①规划主机名及ip**

ceph服务器使用固定ip地址，配置主机名，要求能解析相应的主机名到对应的ip地址，可以使用内网集群的dns服务器或**写入/etc/hosts**文件里。操作系统用centos-stream-8或RockyLinux 8，它们用的软件源是一样的

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | ip | 角色 | 操作系统 |
| ceph1.cof-lee.com | 10.99.1.41 | mon,mgr,osd,bootstrap | centos-stream-8或rocky 8 |
| ceph2.cof-lee.com | 10.99.1.42 | mon,mgr,osd,rgw,mds | centos-stream-8或rocky 8 |
| ceph3.cof-lee.com | 10.99.1.43 | mon,mgr,osd,rgw,mds | centos-stream-8或rocky 8 |

# cat >> /etc/hosts <<EOF

10.99.1.41 ceph1.cof-lee.com ceph1

10.99.1.42 ceph2.cof-lee.com ceph2

10.99.1.43 ceph3.cof-lee.com ceph3

EOF

# hostnamectl set-hostname ceph1.cof-lee.com

**②配置时间同步**

一定要配置ntp时间同步，可用chronyd服务或ntpd

# yum install chrony -y

# systemctl enable chronyd

# systemctl start chronyd

# vi /etc/chrony.conf

server 10.99.1.246 iburst prefer #配置可访问的ntp服务器

# systemctl restart chronyd

# chronyc sources -v

**③安装依赖包**

# yum install python3 lvm2 -y

# yum install podman -y #如果不想用podman，也可用docker-ce

以上操作在3台节点上都要操作，以下只用在ceph1节点上操作（安装cephadm进行bootstrap初始化）

**④配置cephadm软件仓库并安装cephadm工具**

# yum install epel-release -y

# yum search release-ceph #在自带的Extras仓库里

# yum install centos-release-ceph-quincy.noarch -y #自动安装了ceph-repo仓库：

# cat CentOS-Ceph-Quincy.repo

[centos-ceph-quincy]

name=CentOS-$stream - Ceph Quincy

mirrorlist=http://mirrorlist.centos.org/?release=$stream&arch=$basearch&repo=storage-ceph-quincy

#baseurl=http://mirror.centos.org/$contentdir/$stream/storage/$basearch/ceph-quincy/

gpgcheck=1

enabled=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Storage

#或者手动指定aliyun的源（**本例中使用ceph-quincy版本**）

# cat > /etc/yum.repos.d/cephadm-quincy-aliyun.repo <<EOF

[cephadm]

name=cephadm

baseurl=http://mirrors.aliyun.com/centos/8-stream/storage/x86\_64/ceph-quincy

enabled=1

gpgcheck=0

EOF

# yum search cephadm --showduplicates

# yum install cephadm -y

# yum install cephadm-2:17.2.6-3.el8s.noarch -y #安装时指定版本号，如果有多个版本的话

**⑤初始化第一台ceph节点**

# cat /usr/sbin/cephadm | grep "\_IMAGE =" #查看默认镜像名称

cephadm（quincy版本）初始化过程中默认会去下载以下容器镜像：  
quay.io/ceph/ceph:v17

quay.io/ceph/ceph-grafana:8.3.5

quay.io/prometheus/prometheus:v2.33.4

quay.io/prometheus/node-exporter:v1.3.1

quay.io/prometheus/alertmanager:v0.23.0

如果需要使用内网的私有镜像仓库，需要将内部镜像仓库域名设置为quay.io，或者将quay.io解析到内部的镜像仓库地址（暂时没找到修改默认镜像前缀的方法）。且在podman或docker里设置信任私有镜像仓库的ssl证书：

**★podman设置信任私有镜像仓库（不需要重启podman服务）**

# cat > /etc/containers/registries.conf <<EOF

unqualified-search-registries = [ "quay.io" ]

short-name-mode = "permissive"

[[registry]]

insecure = true

prefix = "quay.io"

location = "quay.io"

EOF

**★如果使用docker则设置信任私有镜像仓库（需要重启docker服务）**

# cat > /etc/docker/daemon.json <<EOF

{

"experimental": true,

"exec-opts": ["native.cgroupdriver=systemd"],

"insecure-registries": ["quay.io"]

}

EOF

# systemctl daemon-reload

# systemctl restart docker

# cat >> /etc/hosts <<EOF

#添加内部镜像仓库地址解析

10.99.1.246 quay.io

EOF

**★初始化命令**

# cephadm bootstrap --mon-ip 10.99.1.41 --allow-fqdn-hostname #部署第一台ceph节点（bootstrap）如果仅需要在单机主部署ceph，则需要带上参数--single-host-defaults 如果本地镜像仓库需要验证身份，则使用以下参数

--registry-url harbor.xxx.com --registry-username userxx --registry-password passwdxx

# cephadm shell -- ceph -s #创建容器并在容器里执行命令，默认只有容器里有ceph命令

# yum install ceph-common -y #安装ceph命令，在cephadm的软件仓库里有

# ceph orch host ls #查看集群主机

HOST ADDR LABELS STATUS

ceph1.cof-lee.com 10.99.1.41 \_admin

# ceph versions #查看集群版本

**★安装完成或添加新节点后，会自动配置防火墙**

**首结点：**

**services**: ceph ceph-mon

**ports:**  9283/tcp 8443/tcp 9093/tcp 9094/tcp 3000/tcp 9100/tcp 9095/tcp

**新加结点：**

**services:** ceph ceph-mon

**ports:**  443/tcp 9283/tcp 9100/tcp

**★添加其他ceph节点**

#其他节点也要添加hosts解析指向quay.io，以及配置podman/docker信任私有镜像仓库

# ceph cephadm get-pub-key > ~/ceph.pub

# ssh-copy-id -f -i ~/ceph.pub root@ceph2.cof-lee.com #复制ceph使用的公钥到目标主机

# ssh-copy-id -f -i ~/ceph.pub root@ceph3.cof-lee.com

# ceph orch host add ceph2.cof-lee.com 10.99.1.42 #主机名称和ip要对应得上

# ceph orch host add ceph3.cof-lee.com 10.99.1.43

# ceph orch host label add ceph2.cof-lee.com \_admin #将\_admin标签添加到指定主机，任何具有\_admin标签的主机都将获得/etc/ceph/ceph.conf和/etc/ceph/ceph.client.admin.keyring文件

**★修改时间差异阈值告警**

# ceph health #查看健康状态

# ceph config get mon mon\_clock\_drift\_allowed #查看时间差阈值（默认大于0.05秒就告警）

# ceph config set mon mon\_clock\_drift\_allowed 0.5 #设置时差阈值告警时间为0.5秒

**★配置dashboard**

# ceph dashboard ac-user-show admin

# echo "passwdxxyy" > pass.txt

# ceph dashboard ac-user-set-password admin -i pass.txt #设置dashboard的admin账号密码

# ceph mgr module enable dashboard

# ceph mgr module ls #查看dashboard是否启动

# ceph mgr services #查看dashboard监听端口，仪表板功能由mgr组件提供

{

"dashboard": "https://x.x.x.x:8443/",

"prometheus": "http://x.x.x.x:9283/"

}

**★配置full\_ratio参数**

# ceph osd set-full-ratio 0.97 #设置full阈值

# ceph osd set-nearfull-ratio 0.80

# ceph osd dump | grep ratio

**★★配置集群网络**

使用cephadm bootstrap命令的 --cluster-network x.x.x.x/24 参数在集群bootstrap阶段指定cluster网络

当然，如果在bootstrap的时候没有指定，后期也可以修改

# ceph config get mon public\_network #还可查看mgr，mds等服务的相应网络

# ceph config get mon cluster\_network

# ceph config get osd public\_network

# ceph config get osd cluster\_network

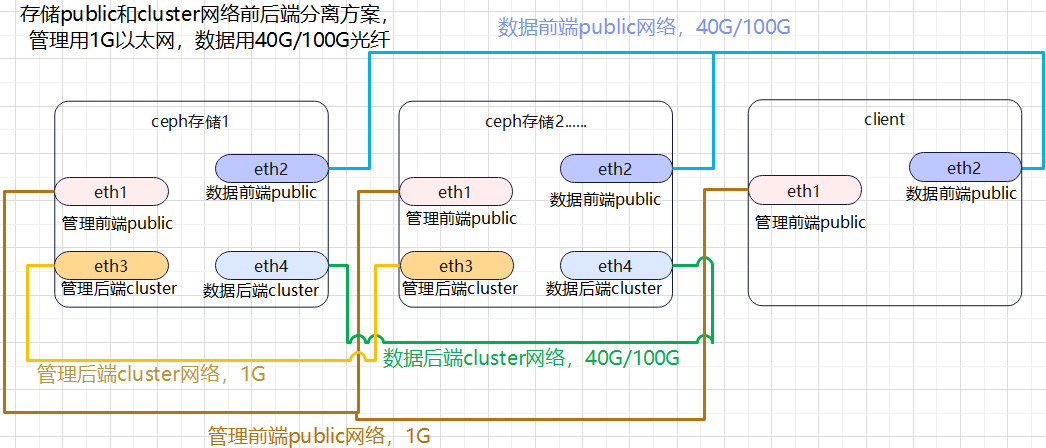
# ceph config dump | grep network #查看所有关于网络的配置

# cluster网络的作用：subnet to use for cluster replication, recovery and heartbeats (in CIDR notation network/mask)

**★ceph集群组网方案设计：**

存储节点有4个网口，客户端节点有2个网口，交换机使用分布式网关组网，服务器网口规划：

|  |  |  |  |
| --- | --- | --- | --- |
| eth1 | 管理前端 mon public\_network | 10.99.1.0/24 | bond 4 2\*1Gbit/s |
| eth2 | 数据前端 osd public\_network | 10.99.2.0/24 | bond 4 2\*40G/100G |
| eth3 | 管理后端 mon cluster\_network | 10.99.3.0/24 | bond 4 2\*1Gbit/s |
| eth4 | 数据后端 osd cluster\_network | 10.99.4.0/24 | bond 4 2\*40G/100G |



#设置网络命令：

# ceph config set mon public\_network 10.99.1.0/24 #供客户端身份验证及获取集群信息用的

# ceph config set mon cluster\_network 10.99.3.0/24

# ceph config set osd public\_network 10.99.2.0/24 #供客户端访问数据用的

# ceph config set osd cluster\_network 10.99.4.0/24

**★第3章、高可用配置**

**★部署mon/mgr服务到节点**

mon组件为必要组件，部署数量为奇数个，比如1，3，5，7

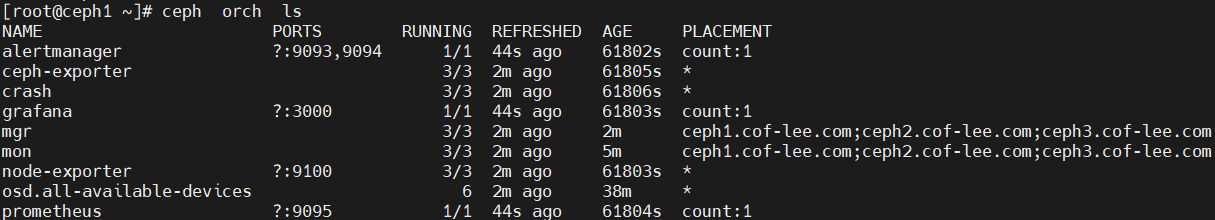
mgr组件为非必要组件，如果要监测状态信息及使用dashboard面板，则部署，高可用建议部署2个及以上数量

# ceph orch ls #查看服务组件基础情况，mgr提供dashboard功能

# ceph orch apply mon ceph1.cof-lee.com,ceph2.cof-lee.com,ceph3.cof-lee.com #将mon部署到这3台上

# ceph orch apply mgr ceph1.cof-lee.com,ceph2.cof-lee.com,ceph3.cof-lee.com #将mgr部署到这3台上

# ceph orch ls #查看服务组件基础情况



# ceph orch ps | grep mgr #查看mgr进程（一个服务进程对应一个docker容器）



# ceph orch daemon restart mgr.ceph1.xmxiig #重启ceph1上的mgr进程

**★添加osd**

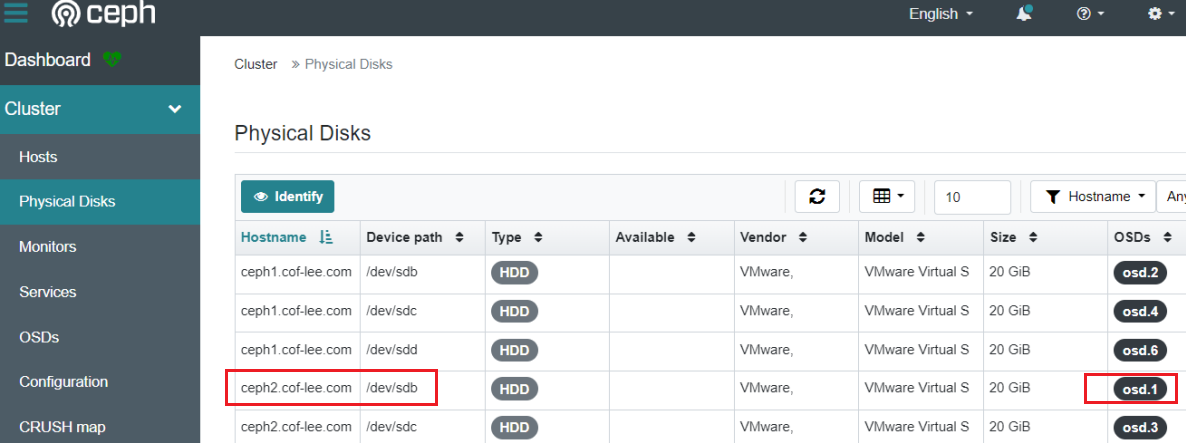
# ceph osd status #查看osd状态

# ceph orch daemon add osd ceph1.cof-lee.com:/dev/vdb,/dev/vdc,dev/vdd #手动添加osd

# ceph orch apply osd --all-available-devices #将集群中所有未使用的磁盘添加进来，开启此功能后，集群中的主机上插入新磁盘时，会自动将新的磁盘（干净可用的，无分区的磁盘）添加到集群中，自动创建相应osd进程

**★查看osd对应的磁盘**

**#方法1**，登录dashboard查看：Cluster→Physical Disks



**#方法2**，命令行查看：先查看目标osd的id，以及目标osd在哪个节点上

ceph osd tree

ceph osd dump | grep "osd\.0" | awk '{print$NF}'

147ef2bb-a717-4d0d-9be5-bf9ed23e895c # osd的id

#再到目标节点查看

osd\_id=147ef2bb-a717-4d0d-9be5-bf9ed23e895c

osd\_vg=$(lvs | grep ${osd\_id} | awk '{print$2}')

pvs | grep ${osd\_vg} | awk '{print$1}'

/dev/sdb #查询结果

**#方法3**，命令行查看：先查看目标osd在哪个节点上，以及ceph集群id

ceph osd tree

ceph -s | grep id | awk '{print$2}'

0eeaa2be-6aa5-11ee-8c6d-000c2943fd34 # ceph集群id

#再到目标节点操作

fsid=0eeaa2be-6aa5-11ee-8c6d-000c2943fd34

lvm\_map=$(ls -l /var/lib/ceph/${fsid}/osd.0/block | awk -F '/' '{print$NF}')

lsblk | grep -B1 ${lvm\_map} | grep disk | awk '{print$1}'

sdb #查询结果

**★移除osd**

**（1）移除正常的osd**

# ceph osd tree #查看目标osd的WEIGHT权重

# ceph osd crush reweight osd.0 0 #降低osd的权重为0，让数据自动迁移到其他osd

# ceph osd safe-to-destroy osd.0 #查看osd是否可安全移除（拔出硬盘）

OSD(s) 0 are safe to destroy without reducing data durability. #可安全移除此osd

# ceph orch apply osd --all-available-devices --unmanaged=true #先禁止自动添加osd

# ceph orch daemon stop osd.0 #停止目标osd进程

# ceph orch ps | grep osd\.0 #确认状态为stopped再删除此daemon

# ceph orch daemon rm osd.0 --force #删除daemon，会自动清除ceph auth

# ceph osd rm osd.0 #删除osdmap

# ceph osd crush remove osd.0 #删除crush map

# ceph auth ls | grep osd\.0 #有则删除，没有即可（删除daemon后，已自动清除ceph auth）

# ceph orch device zap ceph3.cof-lee.com /dev/sdb --force #最后清除磁盘上的lvm信息

#拔除磁盘后，再打开 --all-available-devices

# ceph orch apply osd --all-available-devices --unmanaged=false

**（2）移除故障的osd**

# ceph orch ps | grep osd\.0 #状态为error（由于磁盘损坏或意外拔除导致的）

# ceph orch daemon rm osd.0 --force #删除daemon后，已自动清除ceph auth

# ceph osd rm osd.0

# ceph osd crush remove osd.0

# ceph auth ls | grep osd\.0 #有则删除，没有即可（删除daemon后，已自动清除ceph auth）

#或者将osd daemon删除后，一次性清除osd的crush map，ceph auth，osdmap

# ceph osd purge osd.0 --yes-i-really-mean-it #一次性清除osd的crush map，ceph auth记录，osdmap

**#删除osd后，清理目标磁盘数据（前提是无法通过ceph命令自动清除，需要手动清除）**

# yum install psmisc -y #提供fuser命令

# lsblk #查看目标磁盘对应的vg和lv，先删除lv，后删除vg

# fuser -ck /dev/ceph-41098a76-986d-4025-ba6a-8c03d676fe8d/osd-block-74fa558e-7905-4d99-b432-dee4b656245b

# lvremove /dev/ceph-41098a76-986d-4025-ba6a-8c03d676fe8d/osd-block-74fa558e-7905-4d99-b432-dee4b656245b

# vgremove ceph-41098a76-986d-4025-ba6a-8c03d676fe8d

# dd if=/dev/zero of=/dev/sde count=1 bs=1M #最后清除MBR分区信息，如果磁盘有保密信息，需要做进一步的全盘低格处理

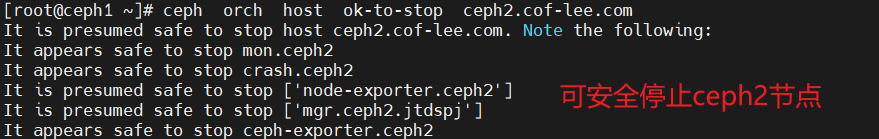
**★移除ceph节点**

**移除节点之前，要将mon/mgr等服务从已移除节点取消部署**

即： ceph orch apply mon/mgr 后面的节点不含要移除的节点即可

**①移除在线节点：**

# ceph orch host ok-to-stop ceph2.cof-lee.com #先查看如果停止主机，是否会影响到集群



# ceph orch host drain ceph2.cof-lee.com #排空主机中的所有守护进程，打上\_no\_schedule标签

# ceph orch osd rm status #检查移除OSD的状态

# ceph orch ps ceph2.cof-lee.com #检查所有守护进程是否已从主机中移除

No daemons reported #提示无守护进程时，才可删除此主机

# ceph orch host rm ceph2.cof-lee.com #删除主机

# ceph osd crush rm ceph2.cof-lee.com #从crush map里清除主机信息

# ceph osd crush rm ceph2 #从crush map里清除主机信息

**②移除离线节点：**

# ceph orch host rm ceph2.cof-lee.com --offline --force

# ceph osd crush rm ceph2.cof-lee.com

# ceph osd crush rm ceph2

**★ceph节点进入维护模式**

# ceph orch host ok-to-stop ceph2.cof-lee.com #先查看如果停止主机，是否会影响到集群

# ceph orch host maintenance enter ceph2.cof-lee.com #进入维护模式，停止节点上的所有服务

# ceph orch host maintenance exit ceph2.cof-lee.com #退出维护模式

**★查看集群存储信息**

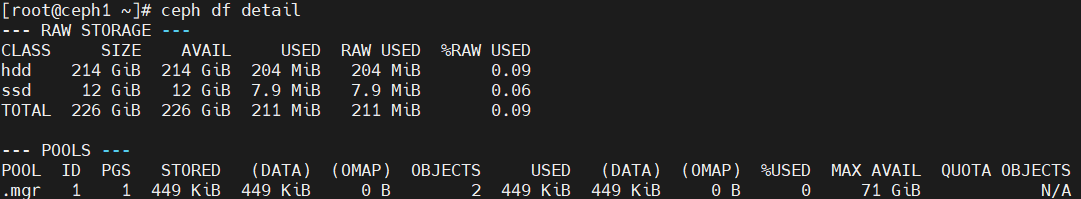
# ceph -s #查看集群存储状态信息（含集群id，即fsid）

cluster:

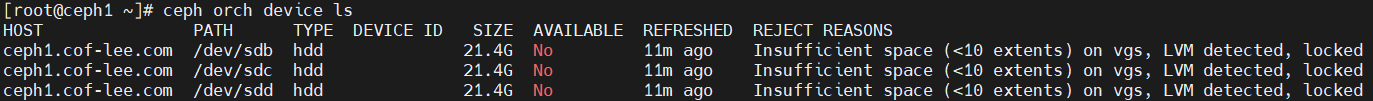
id: 0eeaa2be-6aa5-11ee-8c6d-000c2943fd34 # fsid

# ceph health #查看集群健康状态

# ceph df detail #查看存储情况



# ceph orch device ls #查看集群磁盘信息；AVAILABLE列显示NO表示磁盘已被使用



# ceph orch ps #查看所有节点的进程

# ceph orch ps ceph1.cof-lee.com #查看指定节点的进程

# ceph orch daemon osd.3 restart #重启指定进程容器

**★开启集群自动扩展功能**

默认是开启的，如果登录dashboard时不小心关闭了，需要手动打开，命令如下：

ceph orch apply alertmanager --unmanaged=false

ceph orch apply ceph-exporter --unmanaged=false

ceph orch apply crash --unmanaged=false

ceph orch apply grafana --unmanaged=false

ceph orch apply mgr --unmanaged=false

ceph orch apply mon --unmanaged=false

ceph orch apply node-exporter --unmanaged=false

ceph orch apply prometheus --unmanaged=false

**★第4章、pool存储池操作**

**★磁盘class操作**

磁盘class就是磁盘的分类，具有相同属性的磁盘分到同一类，默认自动根据ssd,hdd等介质来分类，也可自主分类

# ceph osd crush class ls #查看磁盘class

"hdd",

"ssd"

# ceph osd crush get-device-class osd.1 #查看目标磁盘的class

hdd

# ceph osd crush rm-device-class osd.1 #更改磁盘的class之前先删除其class

# ceph osd crush set-device-class ssd\_type2 osd.1 #更改磁盘的class，class名称可自定义

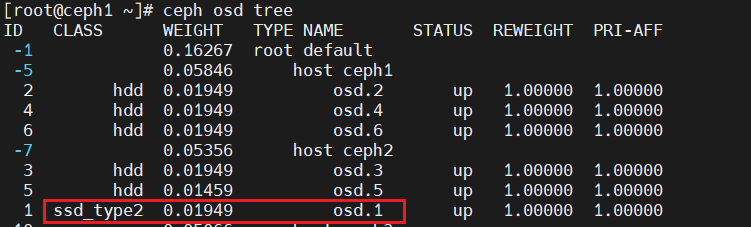
# ceph osd crush class ls

"hdd",

"ssd",

"ssd\_type2" #可见多了这个自定义的磁盘class

# ceph osd tree #查看所有磁盘class



**★crush rule故障域配置**

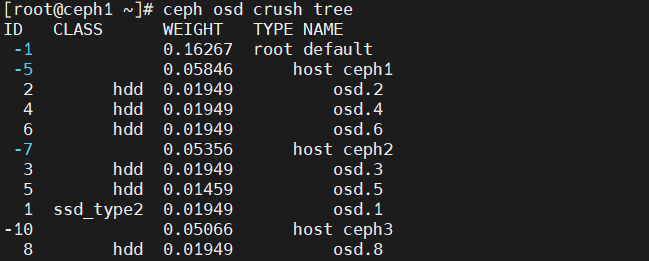
CRUSH层次结构将所有osd及所处的host等信息组成树状的Bucket结构，类似DNS域结构一样，最上层为根Bucket，然后是datacenter，rack，host等不同的层次级别的桶，最下层为osd结点，所以osd也称为Leaf叶子结点

|  |  |  |
| --- | --- | --- |
| 编号 | Bucket | 说明 |
| 10 | root | Bucket层次的根桶，根级别 |
| 9 | region | 可用区域 |
| 8 | datacenter | 物理数据中心 |
| 7 | room | 机房 |
| 6 | pod | 机房中的单间 |
| 5 | pdu | 为机柜分配电力的电源插排 |
| 4 | row | 一排机柜 |
| 3 | rack | 单个机柜 |
| 2 | chassis | 刀片服务器里的机箱 |
| 1 | host | 单台主机（服务器） |
| 0 | osd | 磁盘设备对应的osd守护进程 |

不是每个层级都要有，可选的，只有root根桶及host及osd是必须的。

# ceph osd tree #查看整个crush层次结构

# ceph osd crush tree #查看整个crush层次结构，最上面type为root，name为default，表示最上层的桶类型为root，名称为default，默认情况下所有host及osd都处在这个桶里



各个级别的bucket都有一个id，且为负数，osd结点的id为非负数（0到正整数N）

# ceph osd crush add-bucket datacenter1 root #创建一个名为datacenter1的根桶

# ceph osd crush add-bucket rack01 rack #创建一个名为rack01的rack桶

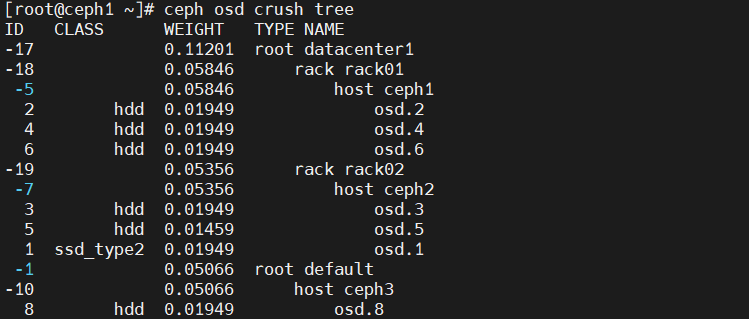
# ceph osd crush add-bucket rack02 rack #创建一个名为rack02的rack桶

# ceph osd crush move rack01 root=datacenter1 #将rack01移入datacenter1里

# ceph osd crush move rack02 root=datacenter1 #将rack02移入datacenter1里

# ceph osd crush move ceph1 rack=rack01 #将ceph1结点移入rack01里

# ceph osd crush move ceph2 rack=rack02 #将ceph2结点移入rack02里



**★crush rule规则表示副本存放的规则，可以是：**

基于osd的故障域存放，即不同的副本放在不同的osd里；

基于host的故障域存放，即不同的副本放在不同的host里；

基于rack的故障域存放，即不同的副本放在不同的rack里；

#创建一个名为rule\_rack\_hdd1的规则，故障域为rack，在datacenter1这个根桶里，使用磁盘class为hdd

# ceph osd crush rule create-replicated rule\_rack\_hdd1 datacenter1 rack hdd

# ceph osd crush rule dump rule\_rack\_hdd1

{

"rule\_id": 9,

"rule\_name": "rule\_rack\_hdd1",

"type": 1,

"steps": [

{

"op": "take",

"item": -24,

"item\_name": "datacenter1~hdd"

},

{

"op": "chooseleaf\_firstn",

"num": 0,

"type": "rack"

},

{

"op": "emit"

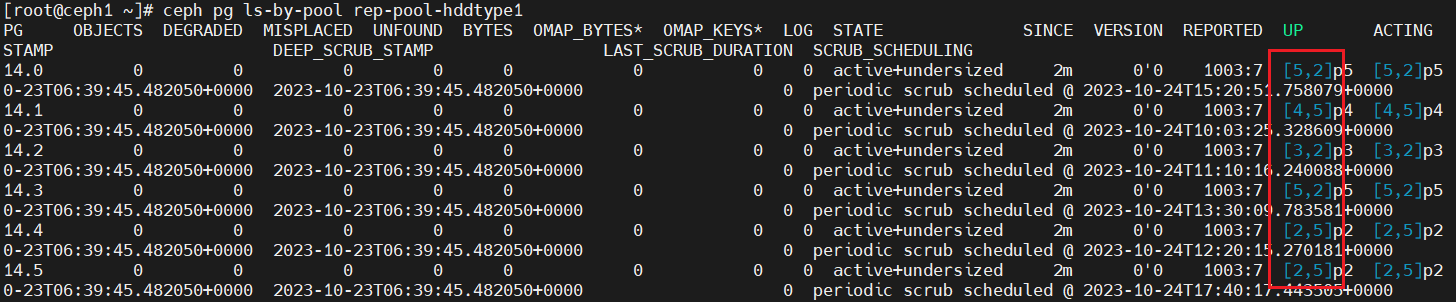
}

]

}

# ceph osd pool create rep-pool-hddtype1 32 32 replicated rule\_rack\_hdd1 #创建池时指定crush rule

# ceph pg ls-by-pool rep-pool-hddtype1 #查看池的pg信息，可见每个pg对应的osd都处于不同的rack里



**★创建复制池（副本池）**

池名称：rbd-pool-hddtype1，32个pg，默认是复制池，（集群为3副本，单机为2副本）

# ceph osd pool create rep-pool-xx 32 32 replicated #创建复制池

# ceph osd pool set rep-pool-xx size 3 #调整池的副本数量为3

#若不指定crush rule冗余策略，则默认为replicated\_rule

# ceph osd crush rule dump replicated\_rule #查看默认的crush rule

**#指定crush rule（放置规则）**

首先创建一个名为ssd\_type2\_rule的rule，根桶为default，故障域为host，磁盘class为ssd\_type2

# ceph osd crush rule create-replicated ssd\_type2\_rule default host ssd\_type2

# ceph osd pool create rep-pool-ssdtype2 32 32 replicated ssd\_type2\_rule #创建池时指定crush rule

**★创建纠删代码池（条带池）**

首先创建一个名为ecprofile-32的纠删代码池模板，3数据，2校验，根桶为default，故障域（冗余级别）为host

# ceph osd erasure-code-profile set ecprofile-32 k=3 m=2 \

crush-root=default crush-failure-domain=host crush-device-class=ssd

# ceph osd erasure-code-profile get ecprofile-32 #查看模板详情

#使用ecprofile-32模板创建一个名为ecpool-xx的纠删代码池，32个pg

# ceph osd pool create ecpool-xx 32 32 erasure ecprofile-32

# ceph osd pool get ecpool-xx all #查看池的详细信息

crush\_rule: ecpool-xx #自动创建crush\_rule，名称同池的名称，配置同池模板ecprofile-32的

# ceph osd pool application enable ecpool-xx rgw #将池用于rgw类似应用

# ceph osd pool application get ecpool-xx #查看池的应用

**★设置osd的pg数量上限**

# ceph config set mon mon\_max\_pg\_per\_osd 500 #设置每osd能创建的pg数量上限，默认250

**★pg数量自动缩放相关配置**

pg\_autoscaler模块最早在Nautilus（14.x）版本中引入，pg\_autoscaler可以根据池使用情况和用户设置的调整来自动调整集群中PG放置组的数量。

pg\_autoscaler自动缩放器带有一系列调整选项，包括--bulk、target\_size\_ratio、noautoscale、bias、pg\_num、pg\_num\_max和pg\_num\_min

相关设置请参考： https://docs.ceph.com/en/latest/rados/operations/placement-groups/

**★--bulk**

--bulk标志表示池的存储空间很大，需要一开始就设置较多的PG数量

# ceph osd pool create <pool-name> --bulk

# ceph osd pool set <pool-name> bulk <true/false/1/0> #在已有池上开启或关闭--bulk标志

# ceph osd pool get <pool-name> bulk #查看池的--bulk标志状态

**★--target\_size\_ratio**

可以使用--target\_size\_ratio设置指定池的目标比率（默认为0），给池设置的目标比率越高，则池的PG数量就越大

# ceph osd pool autoscale-status #查看TARGET RATIO列

# ceph osd pool set <pool\_name> target\_size\_ratio <ratio> #调整池的目标比率

# ceph osd pool create <pool-name> target-size-ratio <ratio>

**★设置池写入限额**

# ceph osd pool set-quota test\_pool max\_bytes 1099511627776 #设置test\_pool的限额为10GB

**★设置pool可被删除**

# ceph config set global mon\_allow\_pool\_delete true

# ceph config set mon mon\_allow\_pool\_delete true

# ceph config set mgr mon\_allow\_pool\_delete true

# ceph config set osd mon\_allow\_pool\_delete true

# ceph config set mds mon\_allow\_pool\_delete true

# ceph config set client mon\_allow\_pool\_delete true

# ceph config get mon

**★删除池**

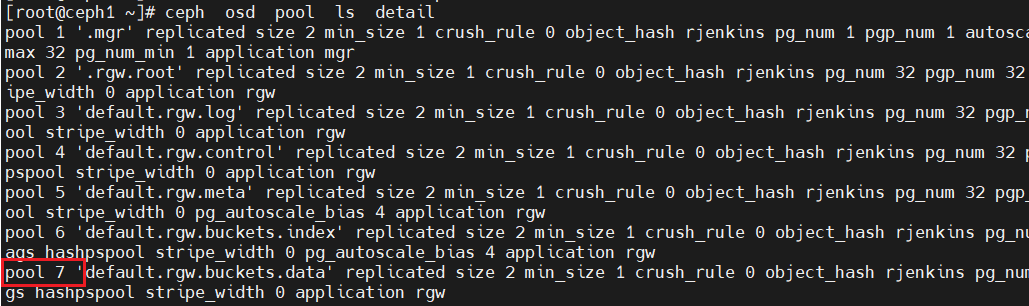
谨慎操作，不可在正式环境操作

# ceph osd pool delete ecpool-xx ecpool-xx --yes-i-really-really-mean-it #要求写2遍目标池的名称

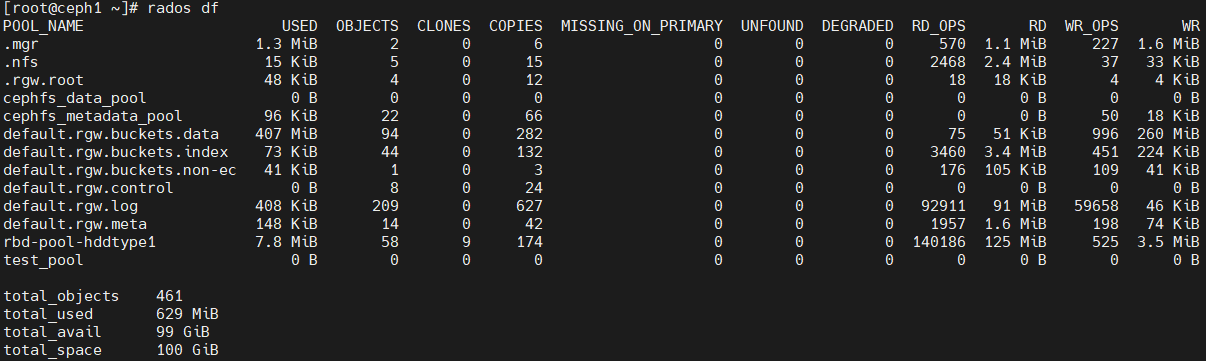
**★pool/pg/osd基本信息查看**

# ceph osd lspools #列出所有池及池id

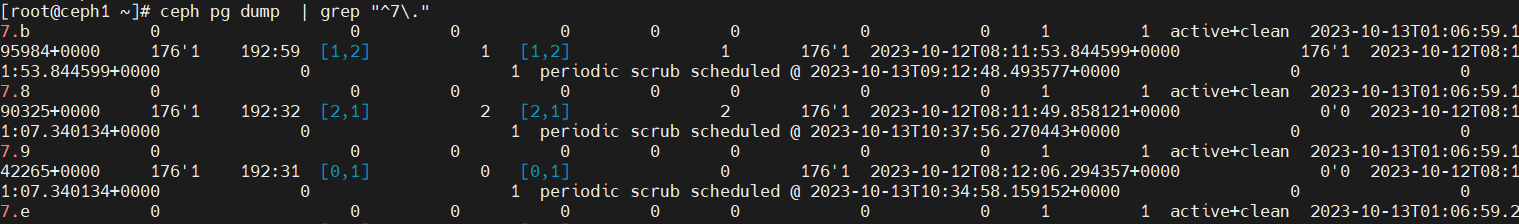
# ceph osd pool ls detail #查看池的详细信息，pool后面的数字为pool id



# rados df #查看所有池的使用情况



# ceph pg dump | grep "^7\." #查看pool id为7的池里的所有pg信息，一行为一个pg



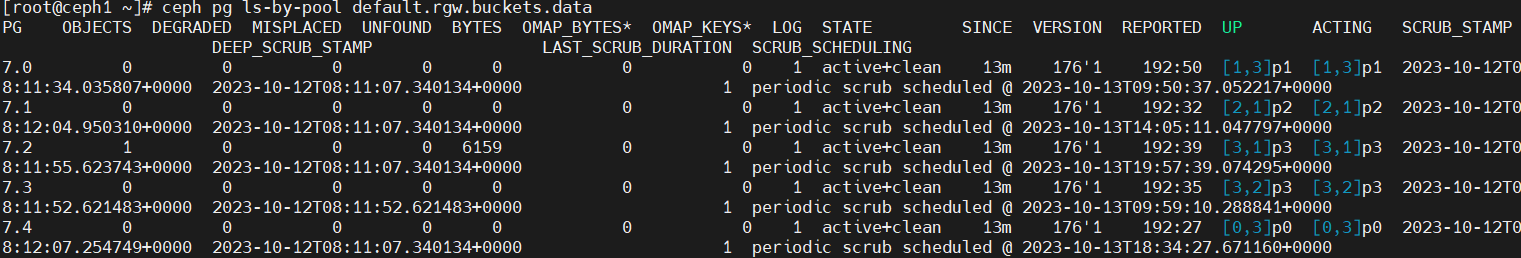
# ceph pg dump | grep "^7\." | wc -l

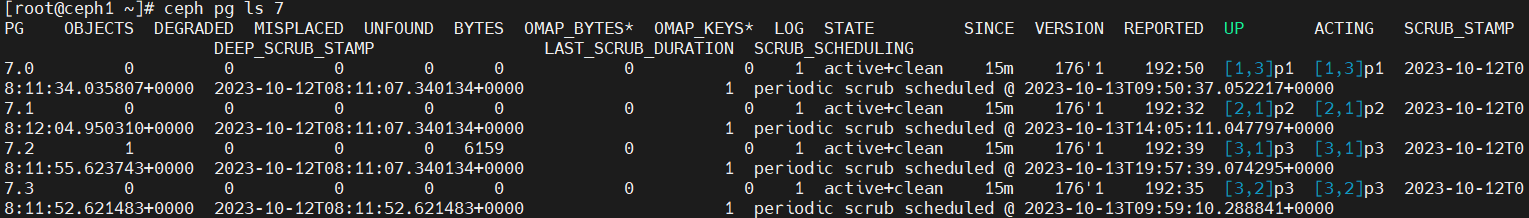
dumped all

32 #这个pool里一共有32个pg

# ceph pg ls-by-pool 池名称 #查看池里的所有pg，通过pool名称查询

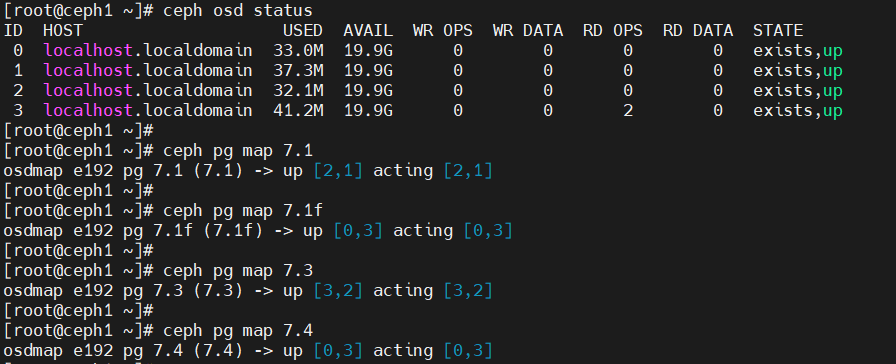
# ceph pg ls 池id #查看池里的所有pg，通过pool id查询





# ceph pg map <pg-id> #查看pg所在的osd

#若这个pg的pool为2副本，则这个pg正常时对应2个osd，pg就是osd上的一个目录

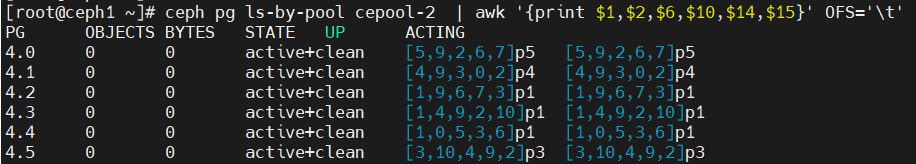


# ceph osd tree #查看osd id及状态等信息

# ceph osd status #查看osd id及状态等信息

# ceph pg ls-by-osd <osd-id> #查看osd上存在的pg，不同的osd上面所存在的pg数量不一定相同

# ceph pg ls-by-pool cepool-2 | awk '{print $1,$2,$6,$10,$14,$15}' OFS='\t'



**★pg状态说明**

Creating PG正在被创建

Peering 持有同一个PG的OSD之间互相比较数据的过程，peering过程中，PG不可读写

Active PG中的数据可以被读写，对该PG的操作请求都将会被处理

Clean PG中的所有对象都已经被复制了规定的副本数量

Scrubbing PG在做不一致性校验

Degraded PG中部分对象的副本数量未达到规定的数量

Undersized PG当前Acting Set小于存储池副本数

Recovering PG正在迁移或者同步对象及其副本

Back Filling 新OSD加入集群后，集群原有的一部分PG进行迁移或进行全量同步

Down PG处于失效离线状态，无法提供读写

Inconsistent PG副本出现不一致

Unfound 某个object已经不存在于任何活动的OSD上了

**★第5章、RBD块存储**

RBD是RADOS Block Device的简称，属于块存储

**★创建rbd池并创建磁盘镜像文件**

#首先创建一个名为rule\_host\_hdd的rule，根桶为default，故障域为host，磁盘class为hdd（根据实际需求来定）

# ceph osd crush rule create-replicated rule\_host\_hdd default host hdd

#创建复制池rbd-pool-hddtype1

# ceph osd pool create rbd-pool-hddtype1 32 32 replicated rule\_host\_hdd

# ceph osd pool application enable rbd-pool-hddtype1 rbd

# ceph osd pool get rbd-pool-hddtype1 all

# ceph osd pool application get rbd-pool-hddtype1

#在池中创建一个名为myrbd-1的磁盘镜像文件，大小512MB

# rbd create -s 512 rbd-pool-hddtype1/myrbd-1

# rbd ls -p rbd-pool-hddtype1 #列出池里的镜像

# rbd info -p rbd-pool-hddtype1 myrbd-1 #查看镜像信息

# rbd --image rbd-pool-hddtype1/myrbd-1 info #查看镜像信息

rbd image 'myrbd-1':

size 512 MiB in 128 objects

............

features: layering, exclusive-lock, object-map, fast-diff, deep-flatten #开启的特性

#关闭某些特性（有的客户端内核不支持的特性）

# rbd feature disable rbd-pool-hddtype1/myrbd-1 exclusive-lock object-map fast-diff deep-flatten

#或者创建镜像时指定开启的特性

# rbd create --size 512 rbd-pool-hddtype1/myrbd-2 --image-feature layering

# rbd resize --size 2G rbd-pool-hddtype1/myrbd-1 #扩容镜像大小

# rbd resize --size 1G rbd-pool-hddtype1/myrbd-1 --allow-shrink #缩容镜像大小，要求客户端文件系统支持

# rbd rm rbd-pool-hddtype1/myrbd-2 #删除rbd镜像，谨慎操作，非必要不删除

#创建rbd客户端用户，只能操作rbd-pool-hddtype1这个池里的rbd镜像

# ceph auth get-or-create client.rbduser1 mon 'profile rbd' osd 'profile rbd pool=rbd-pool-hddtype1' \

mgr 'profile rbd pool=rbd-pool-hddtype1' -o /etc/ceph/ceph.client.rbduser1.keyring

**★客户端挂载rbd磁盘**

客户端安装ceph-common软件

# yum install ceph-common -y

首先创建ceph.conf及keyring文件，客户端使用rbduser1身份操作，缩小权限范围，不能使用admin身份操作

**管理结点上** scp /etc/ceph/ceph.conf clientIPorHostName:/etc/ceph/

**管理结点上** scp /etc/ceph/ceph.client.rbduser1.keyring clientIPorHostName:/etc/ceph/

#客户端结点，以rbduser1身份操作

# rbd --id rbduser1 ls -p rbd-pool-hddtype1 #列出池里的镜像

# rbd --id rbduser1 map --image rbd-pool-hddtype1/myrbd-1 #创建rbd映射

# rbd --id rbduser1 map -p rbd-pool-hddtype1 myrbd-1 #同上，创建rbd映射（命令只可映射一次）

# rbd showmapped #查看rbd映射信息

id pool namespace image snap device

0 rbd-pool-hddtype1 myrbd-1 - /dev/rbd0

# vi /etc/ceph/rbdmap

rbd-pool-hddtype1/myrbd-1 id=rbduser1,keyring=/etc/ceph/ceph.client.rbduser1.keyring

# systemctl enable --now rbdmap

# lsblk

# mkfs.ext4 /dev/rbd0

# mkdir /mountpoint

# vi /etc/fstab

/dev/rbd/rbd-pool-hddtype1/myrbd-1(**最好是使用UUID**) /mountpoint ext4 \_netdev 0 0

# systemctl daemon-reload

# mount -a

# df -Th

/dev/rbd0 ext4 974M 24K 907M 1% /mountpoint

**★取消rbd映射**

# umount /mountpoint/ #先取消挂载

# rbd --id rbduser1 unmap /dev/rbd/rbd-pool-hddtype1/myrbd-1 #取消rbd映射

# rbd --id rbduser1 unmap /dev/rbd0 #同上，取消rbd映射

**★导出rbd磁盘镜像为文件**

# rbd --id rbduser1 export rbd-pool-hddtype1/myrbd-1 myrbd-1-export.img #导出镜像到文件

# ls -lh #导出后的文件大小同镜像的定义大小（即使镜像只使用了几MB，导出后的文件也是1GB）

-rw-r--r--. 1 root root 1.0G Oct 24 09:50 myrbd-1-export.img

**★导入磁盘镜像文件到rbd池中**

# rbd --id rbduser1 import myrbd-1-export.img rbd-pool-hddtype1/myrbd1-new

# rbd --id rbduser1 ls -p rbd-pool-hddtype1

myrbd-1

myrbd1-new #刚刚导入的镜像

**★rbd磁盘快照操作**

**★创建快照**

# rbd --id rbduser1 snap create rbd-pool-hddtype1/myrbd-1@snap-myrbd-1

# rbd --id rbduser1 snap ls rbd-pool-hddtype1/myrbd-1

**★恢复到指定快照**

#假如客户端误操作删除了某些文件，想通过恢复快照来还原；客户端必须先取消挂载及取消rbd映射

# rbd --id rbduser1 snap rollback rbd-pool-hddtype1/myrbd-1@snap-myrbd-1 #恢复到指定快照

**★删除快照**

# rbd --id rbduser1 snap rm rbd-pool-hddtype1/myrbd-1@snap-myrbd-1

**★创建rbd磁盘克隆**

#只能对快照进行克隆，快照只有处于保护状态才可创建克隆

# rbd --id rbduser1 snap protect rbd-pool-hddtype1/myrbd-1@snap-myrbd-1 #将快照设置为保护状态

# rbd --id rbduser1 clone rbd-pool-hddtype1/myrbd-1@snap-myrbd-1 rbd-pool-hddtype1/myrbd-1-clone

#克隆出来的镜像和其父镜像的UUID一样，所以不可在同一客户端挂载使用

# rbd --id rbduser1 children rbd-pool-hddtype1/myrbd-1 #查看此镜像的克隆（子镜像）

# rbd --id rbduser1 flatten rbd-pool-hddtype1/myrbd-1-clone #独立于父镜像

#独立出来的克隆镜像其UUID也和原来的父镜像一样

**★第6章、CephFS文件系统**

**★创建cephfs文件系统**

首先部署mds守护进程，2个及以上实例，可以一行命令操作，也可写入yaml配置文件再执行操作

**★命令行方式部署mds服务**

# ceph orch apply mds cephfs\_daemon\_xx --placement "2 ceph1.cof-lee.com ceph2.cof-lee.com"

#多个目标节点之间用空格隔开

**★使用yaml配置文件方式部署mds服务（与命令行方式二选一）**

# cat > mds\_service.yaml <<EOF

service\_type: mds

service\_id: cephfs\_daemon\_xx

service\_name: mds.cephfs\_daemon\_xx

placement:

count: 2

hosts:

- ceph1.cof-lee.com

- ceph2.cof-lee.com

EOF

# ceph orch apply -i mds\_service.yaml

#创建一个名为rule\_host\_hdd的rule，根桶为default，故障域为host，磁盘class为hdd（根据实际需求来定）

# ceph osd crush rule create-replicated rule\_host\_hdd default host hdd

#创建cephfs的数据池，及metadata池

# ceph osd pool create cephfs\_data\_pool 32 32 replicated rule\_host\_hdd

# ceph osd pool create cephfs\_metadata\_pool 32 32 replicated rule\_host\_hdd

# ceph osd pool application enable cephfs\_data\_pool cephfs #将池应用于cephfs

# ceph osd pool application enable cephfs\_metadata\_pool cephfs

#创建名为cephfs\_xx的文件系统，一个cephfs的data池及metadata池只能给一个文件系统用

# ceph fs new cephfs\_xx cephfs\_metadata\_pool cephfs\_data\_pool

#创建用户，并授予访问该池的相应权限

# ceph fs authorize cephfs\_xx client.fs\_user1 / rws -o /etc/ceph/ceph.client.fs\_user1.keyring

# ceph fs authorize cephfs\_xx client.fs\_user2 /dirxx r -o /etc/ceph/ceph.client.fs\_user2.keyring

#上面-o参数已经将keyring输出到文件了，如果未保存，则可使用ceph auth get client.xxx命令再次获取keyring

**★客户端挂载cephfs文件系统**

首先要在客户端创建用户keyring及ceph.conf文件

客户端 mkdir /etc/ceph

管理端 scp /etc/ceph/ceph.conf root@clientNode:/etc/ceph/

管理端 scp /etc/ceph/ceph.client.fs\_user1.keyring root@clientNode:/etc/ceph/

**①内核方式挂载**

#使用内核支持的挂载参数挂载，要求内核版本为4.x及以上版本，同时也需要安装ceph-fuse软件

# yum install ceph-fuse -y #在cephadm的软件仓库里有

# mkdir /mountpoint #创建挂载点

# mount -t ceph ceph1.cof-lee.com:/ /mountpoint -o name=fs\_user1,fs=cephfs\_xx

# df -Th | grep mountpoint #查看挂载情况

10.99.1.41:/ ceph 72G 0 72G 0% /mountpoint

#开机自动挂载

# vi /etc/fstab #添加以下一行

ceph1.cof-lee.com:6789,ceph2.cof-lee.com:6789:/ /mountpoint ceph \_netdev,defaults,name=fs\_user1,secret=AQANcCtlVDRQARAAMcHO4FQeV82LkbtMTfXIMQ== 0 0

**②ceph-fuse命令方式挂载**

# yum install ceph-fuse -y #在cephadm的软件仓库里有

# mkdir /mountpoint #创建挂载点

# ceph-fuse --keyring=/etc/ceph/ceph.client.fs\_user1.keyring \

--conf=/etc/ceph/ceph.conf \

--name=client.fs\_user1 \

--client-mountpoint=/ /mountpoint

# df -Th | grep mountpoint #查看挂载情况

ceph-fuse fuse.ceph-fuse 71G 2.0G 69G 3% /mountpoint

#开机自动挂载

# vi /etc/fstab #添加以下一行，以下为一行

none /mountpoint fuse.ceph \_netdev,defaults,ceph.name=client.fs\_user1,ceph.keyring=/etc/ceph/ceph.client.fs\_user1.keyring,ceph.client\_mountpoint=/,ceph.mon\_host=ceph1.cof-lee.com:6789,noatime 0 0

#ceph.mon\_host后面值只可写一个ip:port

**★第7章、RADOS Gateway对象存储**

**★部署rgw对象存储服务**

一套ceph存储集群可创建多个RGW网关，端口号不能冲突

例：在ceph1上部署Rados Gateway组件，服务id为myceph\_radosgw，2个服务实例

**★命令行方式部署rgw服务**

# ceph orch apply rgw myceph\_radosgw --port 80 --placement="2 ceph1.cof-lee.com ceph2.cof-lee.com"

#第一个进程监听--port指定的端口，比如80/tcp，第二个进程监听--port指定的值+1，比如81/tcp

#ceph会自动配置防火墙放通相应端口；多个目标节点之间用空格隔开

# ceph orch ls

# ceph orch ps | grep radosgw

**★使用yaml配置文件方式部署rgw服务（与命令行方式二选一）**

# cat > rgw\_service.yaml <<EOF

service\_type: rgw

service\_id: myceph\_radosgw

service\_name: rgw.myceph\_radosgw

placement:

count: 2

hosts:

- ceph1.cof-lee.com

- ceph2.cof-lee.com

spec:

rgw\_frontend\_port: 80

EOF

# ceph orch apply -i rgw\_service.yaml

**#调整服务进程数量**

# ceph orch apply rgw myceph\_radosgw --placement="4 ceph1.cof-lee.com ceph2.cof-lee.com"

**★创建rgw客户端用户**

# radosgw-admin user list #查看用户列表

#创建用户，自动生成用户的ak，sk，需要记下

# radosgw-admin user create --uid="cof" --display-name="cof" --access=full

#创建用户时指定ak，sk

# radosgw-admin user create --uid="cof" --display-name="cof" --access=full \

--access-key=OGYDOQX0LC36ZOXZCW22 --secret-key=lElclOdZBChddWRSJ7r9C5hZg6X54J5zoKK0aEI8

# radosgw-admin user info --uid=cof #查看用户信息，含ak,sk

# radosgw-admin user suspend --uid=cof #暂停用户使用权（禁用指定的用户）

# radosgw-admin user enable --uid=cof #恢复用户，解除禁用

# radosgw-admin user rm --uid=cof --purge-data #删除用户及其所有数据，谨慎操作

# radosgw-admin metadata list bucket #查看所有的bucket桶（用户创建的存储桶）

# radosgw-admin bucket stats --uid=cof #查看用户的所有桶的使用情况

# radosgw-admin bucket unlink --bucket=new-bucket-test --uid=cof #取消用户和桶的关联

# radosgw-admin bucket link --bucket=new-bucket-test --uid=cof #建立用户和桶的关联

**★用户使用量限额**

# radosgw-admin quota enable --uid=cof --quota-scope=user #开启限额功能

# radosgw-admin quota enable --uid=cof --quota-scope=bucket

#用户能创建的桶数量限额10个

# radosgw-admin quota set --uid=cof --quota-scope=user --max-bucket=10

#用户使用总量限额1GB，单位字节

# radosgw-admin quota set --uid=cof --quota-scope=user --max-size=1073741824

#每个桶最多文件对象50000个

# radosgw-admin quota set --uid=cof --quota-scope=bucket --max-objects=50000

#每个桶使用总量限额1GB，单位字节

# radosgw-admin quota set --uid=cof --quota-scope=bucket --max-size=1073741824

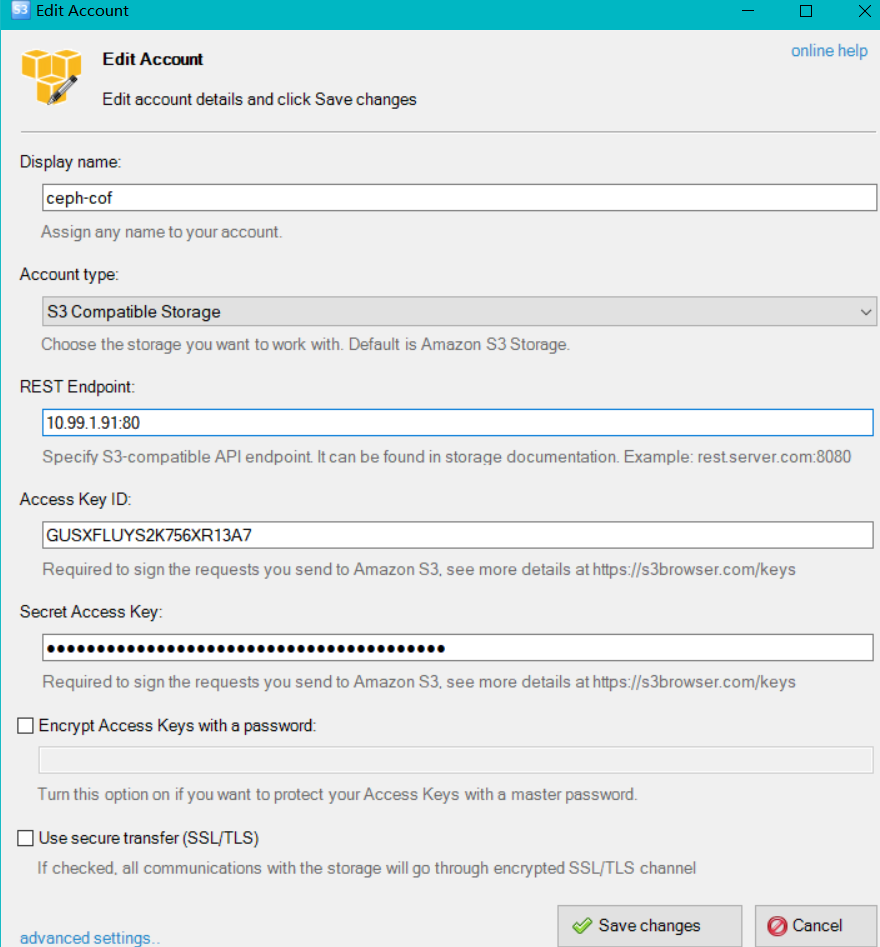
# radosgw-admin user info --uid=cof #查看用户限额情况

**★客户端使用rgw对象存储**

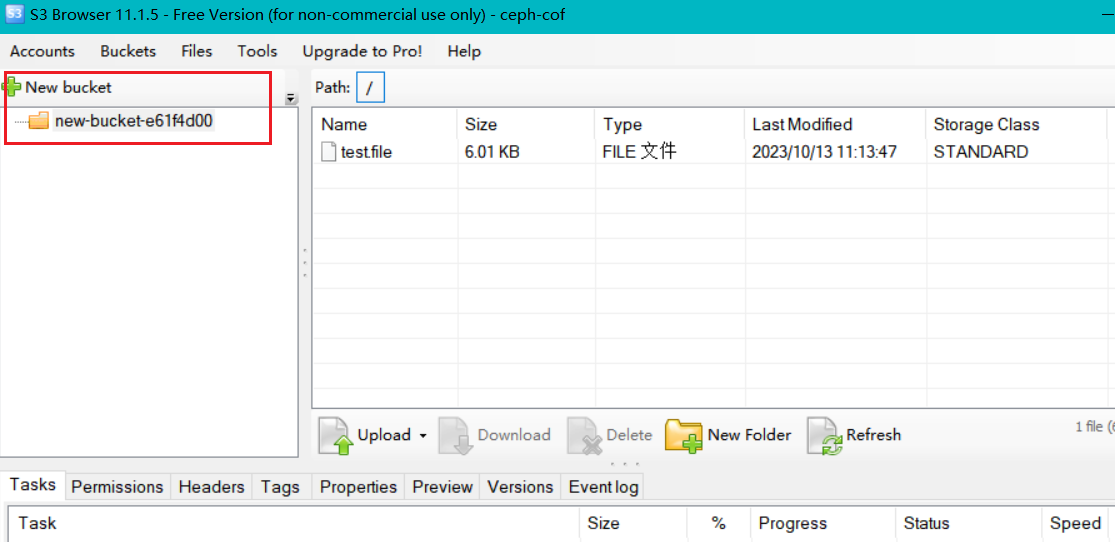
**①使用s3browser可视化客户端**

对象存储客户端可视化软件（windows版本）下载地址：https://s3browser.com/download.aspx

安装后，选择S3 Compatible Storage，输入rgw服务ip:port及用户的ak,sk



登录后，可创建bucket存储桶，在桶里上传或下载文件



**②使用aws命令行客户端**

# yum install epel-release -y

# yum install awscli -y #安装aws命令行客户端工具

或者下载再安装：

# curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

# unzip awscliv2.zip

# ./aws/install

# aws configure #配置用户ak，sk

AWS Access Key ID [None]: OGYDOQX0LC36ZOXZCW22

AWS Secret Access Key [None]: lElclOdZBChddWRSJ7r9C5hZg6X54J5zoKK0aEI8

Default region name [None]: 直接回车

Default output format [None]: 直接回车 #用户信息默认保存在 ~/.aws/credentials 文件里

# aws --endpoint-url http://ceph1.cof-lee.com s3 mb s3://bucket-test3 #创建桶，桶名为bucket-test3

# aws --endpoint-url http://ceph1.cof-lee.com s3 ls #查看本用户（前面ak,sk对应的用户）的所有桶

2023-10-24 14:20:49 bucket-test3

2023-10-24 11:20:30 new-bucket-test

# aws --endpoint-url http://ceph1.cof-lee.com s3 cp --acl public-read \

./localFileName s3://bucket-test3/remoteFileName #将本地文件上传到obs桶里

#权限可以设置为 private，public-read，public-read-write等

# aws --endpoint-url http://ceph1.cof-lee.com s3 ls s3://bucket-test3 #查看桶里的文件

2023-10-24 14:25:36 1028 remoteFileName

# aws --endpoint-url http://ceph1.cof-lee.com s3 cp \

s3://bucket-test3/remoteFileName ./localFileNamexx #从osb桶下载文件到本地

#从obs桶下载公开文件到本地（由于是public读权限，所以不用验证身份）

# curl http://ceph1.cof-lee.com/bucket-test3/remoteFileName -o fileName

方式二# 开启rgw DNS功能，会自动允许请求桶名做前缀的域名

（不建议用，会导致dashboard显示不了RGW信息，暂未找到原因）

存储服务端# ceph config set client.rgw rgw\_dns\_name ceph1.cof-lee.com

存储服务端# ceph orch restart rgw.myceph\_radosgw

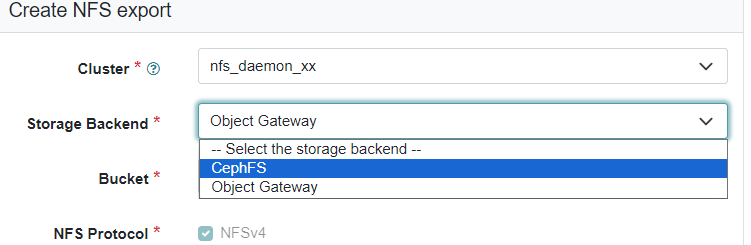
# curl http://bucket-test3.ceph1.cof-lee.com/remoteFileName -o fileName #使用桶名前缀域名获取文件

#前提是要在dns服务器添加对应的解析，或者在本地hosts文件里添加对应的解析

**★第8章、NFS文件系统**

NFS没有单独的存储池，后端存储是基于CephFS文件系统或RGW对象存储桶的

首先创建cephfs文件系统或RGW对象存储桶，再把目标文件系统或obs桶 以nfs的方式导出（export）



**★部署nfs服务**

★命令行方式部署nfs服务

# ceph orch apply nfs nfs\_daemon\_xx --placement "2 ceph1.cof-lee.com ceph2.cof-lee.com"

#多个目标节点之间用空格隔开

★使用yaml配置文件方式部署nfs服务（与命令行方式二选一）

# cat > nfs\_service.yaml <<EOF

service\_type: nfs

service\_id: nfs\_daemon\_xx

service\_name: nfs.nfs\_daemon\_xx

placement:

count: 2

hosts:

- ceph1.cof-lee.com

- ceph2.cof-lee.com

EOF

# ceph orch apply -i nfs\_service.yaml

#默认监听\*:2049端口，自动配置防火墙放通nfs服务（未监听mountd及rpc等服务端口，只可供nfs挂载）

**★创建nfs共享（后端为cephfs）**

首先参考第6章，创建名为cephfs\_xx的文件系统，然后再进行以下操作：

# ceph nfs export create cephfs --cluster-id nfs\_daemon\_xx --pseudo-path /nfsdir \

--squash no\_root\_squash --fsname cephfs\_xx --path=/

#参数说明：

--cluster-id 表示nfs服务的service\_id

--pseudo-path 表示导出的nfs共享路径（客户端那边可见的导出目录）

--squash 默认为no\_root\_squash，可指定为root\_squash，all\_squash

--fsname 表示后端cephfs的名称

--path 表示后端cephfs的路径

--readonly 表示只读，默认为：读写

# ceph nfs export info nfs\_daemon\_xx /nfsdir #查看nfs导出的共享信息

**★客户端挂载**

# showmount -e ceph1.cof-lee.com #查看服务端的nfs导出路径，由于未监听rpc，所以看不了

clnt\_create: RPC: Unable to receive

# mkdir /nfspoint

# mount -t nfs ceph1.cof-lee.com:/nfsdir /nfspoint -o nfsvers=4,proto=tcp,port=2049,nolock,sync #挂载

# df -Th

ceph1.cof-lee.com:/nfsdir nfs4 32G 0 32G 0% /nfspoint

**★创建nfs共享（后端为obs桶）**

首先参考第7章部署好RGW服务，创建用户（用户名为cof），并创建一个对象存储桶bucket-nfs

# ceph nfs export create rgw --cluster-id nfs\_daemon\_xx --pseudo-path /obs\_nfsdir \

--squash no\_root\_squash --bucket bucket-nfs

#参数说明：

--bucket #桶名称

--user-id cof #桶的用户，可选

# ceph nfs export info nfs\_daemon\_xx /obs\_nfsdir #查看nfs导出的共享信息

**★客户端挂载**

# showmount -e ceph1.cof-lee.com #查看服务端的nfs导出路径，由于未监听rpc，所以看不了

clnt\_create: RPC: Unable to receive

# mkdir /obs\_nfspoint

# mount -t nfs ceph1.cof-lee.com:/obs\_nfsdir /obs\_nfspoint -o nfsvers=4,proto=tcp,port=2049,nolock,sync

# df -Th

ceph1.cof-lee.com:/obs\_nfsdir nfs4 100G 666M 100G 1% /obs\_nfspoint

**★导出cof用户的所有桶**

# ceph nfs export create rgw --cluster-id nfs\_daemon\_xx --pseudo-path /cof\_obs\_nfsdir \

--squash no\_root\_squash --user-id cof

# mkdir /cof\_nfspoint

# mount -t nfs ceph1.cof-lee.com:/cof\_obs\_nfsdir /cof\_nfspoint -o nfsvers=4,proto=tcp,port=2049,nolock,sync

# df -Th

ceph1.cof-lee.com:/cof\_obs\_nfsdir nfs4 100G 682M 100G 1% /cof\_nfspoint

# ls -lh /cof\_nfspoint #可见cof用户创建的所有桶都在这个nfs导出共享路径里

drwxrwxrwx. 1 nobody nobody 0 Oct 25 15:16 bucket-nfs

drwxrwxrwx. 1 nobody nobody 0 Oct 25 15:16 bucket-test3

drwxrwxrwx. 1 nobody nobody 0 Oct 25 15:16 new-bucket-test

**★第9章、部署iSCSI Gateway服务**

**★安装ceph-iscsi及tcmu-runner**

官方指导文档： https://docs.ceph.com/en/quincy/rbd/iscsi-target-cli/

由于目前还没有el系列的rpm安装包，所以先用**源码安装**，需要安装以下5个软件：

tcmu-runner

rtslib-fb

configshell-fb

targetcli-fb

ceph-iscsi

指导文档链接： https://docs.ceph.com/en/quincy/rbd/iscsi-target-cli-manual-install/

在ceph存储结点上操作；要求存储服务器系统内核>=4.16，或系统为rhel系列且>=7.5

**★安装依赖**

# yum install python3-cryptography python3-flask python3-gobject python3-kmod \

python3-netifaces python3-pyOpenSSL python3-pyparsing python3-requests python3-urwid \

python3-pyudev libnl3 librbd1 kmod-libs git -y

# yum install ceph-common python3-rados python3-rbd -y #在 centos-ceph-quincy 仓库里有

**★tcmu-runner**

# git clone https://github.com/open-iscsi/tcmu-runner

# cd tcmu-runner

# vi /etc/os-release #将ID的值改为 fedora，rhel，centos 三个中的一个

ID="centos"

# ./extra/install\_dep.sh #安装12个依赖包，如下：

cmake gcc make libnl3-devel glib2-devel zlib-devel glusterfs-api \

gperftools-devel kmod-devel glusterfs-api-devel librados-devel librbd-devel #这5个包安装光盘里没有，需要手动下载安装，对应仓库如下：

gperftools-devel #在 epel 仓库里有

kmod-devel glusterfs-api-devel #在系统自带的 \*-PowerTools.repo 里有

librados-devel librbd-devel #在 centos-ceph-quincy 仓库里有

# ./extra/install\_dep.sh #安装完所有依赖包后再执行一次此脚本，所有的包都提示Complete!即可

# cmake -Dwith-glfs=false -Dwith-qcow=false -DSUPPORT\_SYSTEMD=ON -DCMAKE\_INSTALL\_PREFIX=/usr

# make install

# systemctl daemon-reload

# systemctl enable tcmu-runner

# systemctl start tcmu-runner

**★rtslib-fb**

# git clone https://github.com/open-iscsi/rtslib-fb.git

# cd rtslib-fb

# python3 setup.py install

**★configshell-fb**

# git clone https://github.com/open-iscsi/configshell-fb.git

# cd configshell-fb

# python3 setup.py install

**★targetcli-fb**

# git clone https://github.com/open-iscsi/targetcli-fb.git

# cd targetcli-fb

# python3 setup.py install

# mkdir /etc/target

# mkdir /var/target

**★ceph-iscsi**

ceph-iscsi提供了gwcli命令

# git clone https://github.com/ceph/ceph-iscsi.git

# cd ceph-iscsi

# python3 setup.py install --install-scripts=/usr/bin

# cp usr/lib/systemd/system/rbd-target-gw.service /lib/systemd/system

# cp usr/lib/systemd/system/rbd-target-api.service /lib/systemd/system

# systemctl daemon-reload

# systemctl enable rbd-target-gw

# systemctl start rbd-target-gw

# systemctl enable rbd-target-api

# systemctl start rbd-target-api #启动失败是正常的，因为未创建名为rbd的池

**★创建名为rbd的池给iSCSI用**

#创建一个名为rule\_host\_hdd的rule，根桶为default，故障域为host，磁盘class为hdd（根据实际需求来定）

# ceph osd crush rule create-replicated rule\_host\_hdd default host hdd

#创建iSCSI的数据池，默认需要名为rbd的池，也可自定义池名

# ceph osd pool create rbd 32 32 replicated rule\_host\_hdd

# ceph osd pool application enable rbd rbd

# systemctl restart rbd-target-api #重启此服务

# systemctl status rbd-target-api #状态为active (running)

**★创建ceph-iscsi配置文件**

# cat > /etc/ceph/iscsi-gateway.cfg <<EOF

[config]

cluster\_name = ceph

pool = rbd

# 需要将ceph.conf及admin的keyring文件复制到/etc/ceph目录下

gateway\_keyring = ceph.client.admin.keyring

# API settings.

api\_secure = false

# 如果要开启为true，则需要在/etc/ceph目录下创建密钥文件，文件名只能为：

# 'iscsi-gateway.crt' and 'iscsi-gateway.key' and placed in the '/etc/ceph/' directory

# Additional API configuration options are as follows, defaults shown.

api\_user = admin

# api\_password = admin

api\_port = 5000

# trusted\_ip\_list = 192.168.0.10,192.168.0.11 #允许访问iSCSI服务的客户端ip

EOF

# vi /usr/lib/systemd/system/rbd-target-api.service #新增RestartSec=60s及修改StartLimitInterval为30s

RestartSec=60s

StartLimitInterval=30s

# systemctl daemon-reload

# systemctl restart rbd-target-gw #重启此服务

# systemctl restart rbd-target-api #重启此服务

**★防火墙放通3260和5000端口**

# firewall-cmd --add-port=3260/tcp

# firewall-cmd --add-port=5000/tcp

# firewall-cmd --runtime-to-permanent

**★创建iscsi target及gateway**

gwcli will create and configure the iSCSI target and RBD images and copy the configuration across the gateways setup in the last section. Lower level tools including targetcli and rbd can be used to query the local configuration, but should not be used to modify it. This next section will demonstrate how to create a iSCSI target and export a RBD image as LUN 0

# gwcli #进入gwcli交互界面

/> cd /iscsi-targets

/> create iqn.2023-10.com.cof-lee.iscsi-gw:iscsi-igw

/> cd /iscsi-targets/iqn.2023-10.com.cof-lee.iscsi-gw:iscsi-igw/gateways

/> create skipchecks=true ceph1.cof-lee.com 10.99.1.41 #主机名称要写全，第一个节点必须为本主机（操作gwcli命令的这台服务器）如果操作系统不是cenots/rhel，则需要带skipchecks=true参数

/> create skipchecks=true ceph2.cof-lee.com 10.99.1.42 #如有多台，则分别添加，通过5000/tcp访问

/> cd /disks

/> create pool=rbd image=rbd\_disk\_01 size=10G #创建磁盘

/> cd /iscsi-targets/iqn.2023-10.com.cof-lee.iscsi-gw:iscsi-igw/hosts

/> create iqn.2023-10.com.cof-lee.client:clientnode1 #创建客户端验证，至少要有2个gateway才可创建

/> auth username=username1 password=passwdxxyyzz #创建CHAP用户及密码（必须），用户名要大于8字符长度，密码要大于12字符长度

/> disk add rbd/rbd\_disk\_01 #添加磁盘给此客户端iqn

/> exit #直接退出即可，自动保存

**★添加iscsi-gateway到dashboard中**

# vi /etc/ceph/iscsi-ceph1.api #添加api信息，http(s)://账号:密码@iSCSI-GW地址:port

http://admin:admin@10.99.1.41:5000

# vi /etc/ceph/iscsi-ceph2.api #添加api信息，http(s)://账号:密码@iSCSI-GW地址:port

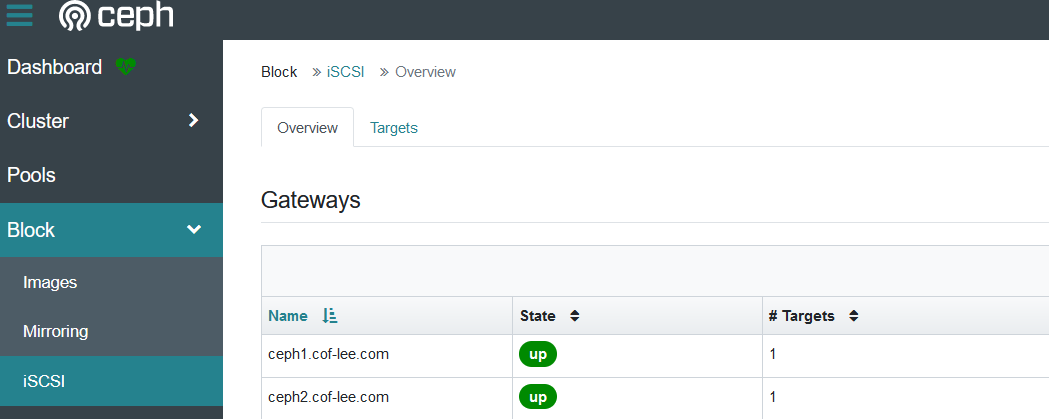
http://admin:admin@10.99.1.42:5000

# ceph dashboard iscsi-gateway-add -i /etc/ceph/iscsi-ceph1.api iscsi-ceph1

# ceph dashboard iscsi-gateway-add -i /etc/ceph/iscsi-ceph2.api iscsi-ceph2

# ceph dashboard iscsi-gateway-list #查看iscsi-gateway信息

{"gateways": {"ceph1.cof-lee.com": {"service\_url": "http://admin:admin@10.99.1.41:5000"}, "ceph2.cof-lee.com": {"service\_url": "http://admin:admin@10.99.1.42:5000"}}}



**★监测ceph-iSCSI性能**

参考链接： https://docs.ceph.com/en/quincy/rbd/iscsi-monitoring/

# git clone -b 2.2 --depth=1 https://github.com/ceph/ceph-iscsi-tools.git

**★客户端挂载iSCSI磁盘**

参考链接： https://docs.ceph.com/en/quincy/rbd/iscsi-initiator-linux/

# yum install iscsi-initiator-utils device-mapper-multipath -y

# mpathconf --enable --with\_multipathd y #启用iscsi多路径功能

# vi /etc/multipath.conf #在文件末尾新增以下一段内容

devices {

device {

vendor "LIO-ORG"

product "TCMU device"

hardware\_handler "1 alua"

path\_grouping\_policy "failover"

path\_selector "queue-length 0"

failback 60

path\_checker tur

prio alua

prio\_args exclusive\_pref\_bit

fast\_io\_fail\_tmo 25

no\_path\_retry queue

}

}

# systemctl reload multipathd

# vi /etc/iscsi/initiatorname.iscsi #删除默认那行，添加以下一行

InitiatorName=iqn.2023-10.com.cof-lee.client:clientnode1

# vi /etc/iscsi/iscsid.conf #在文件末尾新增以下一段内容，如果已有相同配置项，则修改之

node.session.auth.authmethod = CHAP

# set a CHAP username and password for initiator

node.session.auth.username = username1

node.session.auth.password = passwdxxyyzz

# set a CHAP username and password for target(s)，默认服务端未开启双向验证，不用写这个密码

#node.session.auth.username\_in = usernamexxx

#node.session.auth.password\_in = passwdxxxxxyyzz

# iscsiadm -m discovery -t sendtargets -p ceph1.cof-lee.com #发现iqn

10.99.1.41:3260,1 iqn.2023-10.com.cof-lee.iscsi-gw:iscsi-igw

10.99.1.42:3260,2 iqn.2023-10.com.cof-lee.iscsi-gw:iscsi-igw

# iscsiadm -m node -T iqn.2023-10.com.cof-lee.iscsi-gw:iscsi-igw -l #登录，连接iqn

# multipath -ll #查看iscsi多路径情况

mpatha (36001405cce9bbc857b648d8b1ac137e7) dm-2 LIO-ORG,TCMU device

size=10G features='1 queue\_if\_no\_path' hwhandler='1 alua' wp=rw

|-+- policy='queue-length 0' prio=50 status=active

| `- 3:0:0:0 sdb 8:16 active ready running

`-+- policy='queue-length 0' prio=10 status=enabled

`- 4:0:0:0 sdc 8:32 active ready running

# lsblk #生成多个盘符，对应 /dev/mapper/mpatha

sdb 8:16 0 10G 0 disk

└─mpatha 253:2 0 10G 0 mpath

sdc 8:32 0 10G 0 disk

└─mpatha 253:2 0 10G 0 mpath

然后可对 /dev/mapper/mpatha 进行正常的创建分区格式化挂载等操作

# mkfs -t xfs /dev/mapper/mpatha

# mkdir /iscsi-ceph-mountpoint

# mount -t xfs /dev/mapper/mpatha /iscsi-ceph-mountpoint

# df -Th

/dev/mapper/mpatha xfs 10G 104M 9.9G 2% /iscsi-ceph-mountpoint

**★第10章、升级ceph集群版本**

在Octopus版本后使用cephadm来部署ceph集群，如果使用cephadm部署，在后期新的版本升级时，可以做到完全自动化，并可以通过ceph -W cephadm查看升级进度，升级完成后，无法降级，升级时请不要跨版本升级

本章示例：从quincy版本升级到reef版本

# ceph orch upgrade start --ceph-version 18.2.0

# ceph -W cephadm

# cephadm shell -- ceph orch upgrade check --ceph-version 18.2.0 #检查是否有新版本可用，如果有，则会自动拉取镜像

# cephadm shell -- ceph orch upgrade start --ceph-version 18.2.0

# cephadm shell -- ceph -W cephadm