CEPH 官方文档

翻译人：张绍文<gongfan193@gmail.com>

版　本：v1.3

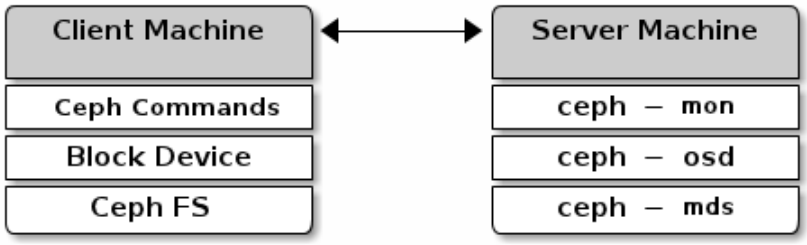
http://ceph.com/docs/master/start/

1 开始

1.1 5 分钟快速入门

Thank you for trying Ceph! Petabyte-scale data clusters are quite an undertaking. Before delving deeper into Ceph, we recommend setting up a cluster on a single host to explore some of the functionality. The Ceph **5-Minute Quick Start** deploys a Ceph object store cluster on one server machine and a Ceph client on a separate machine, each with a recent Debian/Ubuntu operating system. The intent of this **Quick Start** is to help you exercise Ceph object store functionality without the configuration and deployment overhead associated with a production-ready object store cluster. Once you complete this quick start, you may exercise Ceph commands on the command line. You may also proceed to the quick start guides for block devices, CephFS filesystems, and the RESTful gateway.

感谢您尝试 ceph！PB 级数据集群是很大的挑战，在深入 ceph 前，我们推荐您在单机上安装一个集群来发掘它的功能。5 分钟快速入门在单机上部署了一套 ceph 对象存储集群服务器端，在另一台机器上部署了 ceph 客户端，二者都基于最新的 Debian/Ubuntu 操作系统。这篇快速入门意在帮您练习 Ceph 对象存储功能，而无需配置、部署生产级的对象存储集群。完成本篇快速入门后，您就可以在命令行练习 ceph 命令了，也可以继续尝试块设备、CephFS 文件系统、和 RESTful 网关。



1.1.1 安装 Debian/Ubuntu

Install Debian/Ubuntu
Install a recent release of Debian or Ubuntu (e.g., 12.04 precise).
安装 Debian 或 Ubuntu 的最新版（如 12.10）

1.1.2 安装 ceph 软件包

Add Ceph Packages
To get the latest Ceph packages, add a release key to APT, add a source location to your /etc/apt/sources.list, update your system and install Ceph.
要安装最新的 ceph 包，先把发布公钥添加到 APT 中，再把源位置添加到/etc/apt/sources.list 中，更新系统、安装 ceph。以下是命令：

```
wget -q -O- https://raw.github.com/ceph/ceph/master/keys/release.asc | sudo apt-key add -
echo deb http://ceph.com/debian/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
sudo apt-get update && sudo apt-get install ceph
```

Check the Ceph version you are using and make a note of it so that you have the correct settings in your configuration file:
检查下你安装的 ceph 版本，记下来，以便稍后正确地配置：

```
ceph -v
```

If ceph -v reflects an earlier version from what you installed, your ceph-common library may be using the version distributed with the kernel. Once you've installed Ceph, you may also update and upgrade your packages to ensure you have the latest ceph-common library installed.
如果 ceph -v 显示您安装了较老的版本，但 ceph-common 库可能是随内核发布的。所以安装 ceph 后，最好更

新并升级软件包，以确保你安装了最新的 ceph-common 库。

```
sudo apt-get update && sudo apt-get upgrade
```

If you want to use a version other than the current release, see Installing Debian/Ubuntu Packages for further details.
如果你不想用当前发布的版本，参见 Installing Debian/Ubuntu Packages。

1.1.3 写配置文件

Add a Configuration File
The example configuration file will configure Ceph to operate a monitor, two OSD daemons and one metadata server on your Ceph server machine. To add a configuration file to Ceph, we suggest copying the contents of the example file below to an editor. Then, follow the steps below to modify it.
样板配置文件将为 ceph 集群配置 1 个监视器、2 个 OSD 守护进程、和 1 个元数据服务器。我们建议您把下面的配置样本拷贝到文本编辑器，然后按下列步骤修改它。

```
[global]

    # For version 0.55 and beyond, you must explicitly enable
    # or disable authentication with "auth" entries in [global].

    auth cluster required = cephx
    auth service required = cephx
    auth client required = cephx

[osd]
    osd journal size = 1000

    #The following assumes ext4 filesystem.
    filestore xattr use omap = true


    # For Bobtail (v 0.56) and subsequent versions, you may
    # add settings for mkcephfs so that it will create and mount
    # the file system on a particular OSD for you. Remove the comment `#`
    # character for the following settings and replace the values
    # in braces with appropriate values, or leave the following settings
    # commented out to accept the default values. You must specify the
    # --mkfs option with mkcephfs in order for the deployment script to
    # utilize the following settings, and you must define the 'devs'
    # option for each osd instance; see below.

    #osd mkfs type = {fs-type}
    #osd mkfs options {fs-type} = {mkfs options}   # default for xfs is "-f"
    #osd mount options {fs-type} = {mount options} # default mount option is "rw,noatime"

    # For example, for ext4, the mount option might look like this:

    #osd mkfs options ext4 = user_xattr,rw,noatime

    # Execute $ hostname to retrieve the name of your host,
    # and replace {hostname} with the name of your host.
    # For the monitor, replace {ip-address} with the IP
    # address of your host.

[mon.a]

    host = {hostname}
    mon addr = {ip-address}:6789

[osd.0]
    host = {hostname}

    # For Bobtail (v 0.56) and subsequent versions, you may
    # add settings for mkcephfs so that it will create and mount
    # the file system on a particular OSD for you. Remove the comment `#`
    # character for the following setting for each OSD and specify
    # a path to the device if you use mkcephfs with the --mkfs option.

    #devs = {path-to-device}

[osd.1]
    host = {hostname}
    #devs = {path-to-device}
```

```
[mds.a]
    host = {hostname}
```

1. Open a command line on your Ceph server machine and execute hostname -s to retrieve the name of your Ceph server machine.
   在 ceph 服务器机器上打开一个终端，执行 hostname -s 获取其主机名。
2. Replace {hostname} in the sample configuration file with your host name.
   用你的主机名替换配置样本中的{hostname}。
3. Execute ifconfig on the command line of your Ceph server machine to retrieve the IP address of your Ceph server machine.
   执行 ifconfig获取 ceph 服务器的 IP 地址。
4. Replace {ip-address} in the sample configuration file with the IP address of your Ceph server host.
   用你找到的 ceph 服务器 IP 地址替换配置样本中的{ip-address}。
5. Save the contents to /etc/ceph/ceph.conf on Ceph server host.
   把修改后的内容保存到 ceph 服务器的/etc/ceph/ceph.conf 文件。
6. Copy the configuration file to /etc/ceph/ceph.conf on your client host.
   把那份配置文件也拷贝到客户端的/etc/ceph/ceph.conf。

```
    sudo scp {user}@{server-machine}:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

Tip: Ensure the ceph.conf file has appropriate permissions set (e.g. chmod 644) on your client machine.
提示：确保客户端上的 ceph.conf 权限位设置得当，如：chmod 644。

New in version 0.55.
Ceph v0.55 and above have authentication enabled by default. You should explicitly enable or disable authentication with version 0.55 and above. The example configuration provides auth entries for authentication. For details on Ceph authentication see Cephx Configuration Reference and Cephx Guide.
0.55 新增。
ceph v0.55 及以上版本默认启用了认证，使用这些版本时应该显式地启用或禁用认证。配置样本提供了 auth 条目用于配置认证，详情参见 Cephx Configuration Reference 和 Cephx Guide。

1.1.4 部署配置

Deploy the Configuration
You must perform the following steps to deploy the configuration.
你必须执行下列步骤来实现之前的配置。

1. On your Ceph server host, create a directory for each daemon. For the example configuration, execute the following:
   在 ceph 服务器主机上，给每个进程创建相应目录。比如对样本配置，执行下面的命令：

```
sudo mkdir /var/lib/ceph/osd/ceph-0
sudo mkdir /var/lib/ceph/osd/ceph-1
sudo mkdir /var/lib/ceph/mon/ceph-a
sudo mkdir /var/lib/ceph/mds/ceph-a
```

2. Execute the following on the Ceph server host:
   在 ceph 服务器上执行下列命令：

```
cd /etc/ceph
sudo mkcephfs -a -c /etc/ceph/ceph.conf -k ceph.keyring
```

Among other things, mkcephfs will deploy Ceph and generate a client.admin user and key. For Bobtail and subsequent versions (v 0.56 and after), the mkcephfs script will create and mount the filesystem for you provided you specify osd mkfs osd mount and devs settings in your Ceph configuration file.

除此之外，mkcephfs 会部署并生成一个 client.admin 用户及其密钥。对于 Bobtail 及之后版本（v0.56 及以上），如果你在配置文件里配置了 osd mkfs、osd mount 和 devs，mkcephfs 脚本会帮你创建并挂载文件系统。

1.1.5 启动 ceph 集群

Start Ceph

Once you have deployed the configuration, start Ceph from the command line of your server machine.
配置、部署完成后，从服务器端命令行启动 ceph。

```
sudo service ceph -a start
```

Check the health of your Ceph cluster to ensure it is ready.
检查 ceph 集群健康状态，确保无误：

```
sudo ceph health
```

When your cluster echoes back HEALTH_OK, you may begin using Ceph.
如果显示 HEALTH_OK，你就可以试用集群了。

1.1.6 把密钥环拷贝到客户端

Copy The Keyring to The Client
The next step you must perform is to copy /etc/ceph/ceph.keyring, which contains the client.admin key, from the server machine to the client machine. If you don't perform this step, you will not be able to use the Ceph command line, as the example Ceph configuration requires authentication.
下一个必须做的是把/etc/ceph/ceph.keyring 从服务器复制到客户端，它包含 client.admin 的密钥。如果不做这步，你就不能用 ceph 命令行，因为样本配置要求认证。

```
sudo scp {user}@{server-machine}:/etc/ceph/ceph.keyring /etc/ceph/ceph.keyring
```

Tip: Ensure the ceph.keyring file has appropriate permissions set (e.g., chmod 644) on your client machine.
提示：确保客户端机器上的 ceph.keyring 文件设置了恰当的权限位。

1.1.7 继续其他快速入门

Proceed to Other Quick Starts
Once you have Ceph running with both a client and a server, you may proceed to the other Quick Start guides.
ceph 客户端和服务器都运行良好后，你可以继续其他快速入门文档。

1. For Ceph block devices, proceed to Block Device Quick Start.
   想了解 ceph 块设备，继续……
2. For the CephFS filesystem, proceed to CephFS Quick Start.
   想了解 CephFS 文件系统，继续……
3. For the RESTful Gateway, proceed to Gateway Quick Start.
   想了解 RESTful 网关，继续……

1.2 块设备快速入门

Block Device Quick Start
To use this guide, you must have executed the procedures in the 5-minute Quick Start guide first. Execute this quick start on the client machine.
要实践这篇指导，你必须先完成前面的 5 分钟快速入门。应该在客户端上执行这篇快速入门。

1. Create a block device image.
   创建一个块设备映像。

```
rbd create foo --size 4096
```

2. Load the rbd client module.
   载入 rbd 客户端模块：

```
sudo modprobe rbd
```

3. Map the image to a block device.
   把映像映射到块设备。

```
sudo rbd map foo --pool rbd --name client.admin
```

4. Use the block device. In the following example, create a file system.

使用块设备。在后面的例子中，创建了一个文件系统。

```
sudo mkfs.ext4 -m0 /dev/rbd/rbd/foo
```

5. Mount the file system.
   挂载文件系统。

```
sudo mkdir /mnt/myrbd
sudo mount /dev/rbd/rbd/foo /mnt/myrbd
```

Note: Mount the block device on the client machine, not the server machine. See FAQ for details.
注意：要从客户端挂载块设备，而不是从服务器。详情见 FAQ。

See block devices for additional details.
相关的更多信息见块设备。

1.3 CEPHFS 快速入门

CephFS Quick Start

To use this guide, you must have executed the procedures in the 5-minute Quick Start guide first. Execute this quick start on the client machine.

这篇向导基于前面的 5 分钟快速入门，以下步骤要在其他客户端主机上进行。

1.3.1 内核空间驱动

Kernel Driver

Mount Ceph FS as a kernel driver.

用内核驱动挂载 ceph 集群。

```
sudo mkdir /mnt/mycephfs
sudo mount -t ceph {ip-address-of-monitor}:6789:/ /mnt/mycephfs
```

Note: Mount the CephFS filesystem on the client machine, not the cluster machine. See FAQ for details.
注意：要在客户端主机上挂载 CephFS 文件系统，不能在集群主机上。详情参见 FAQ。

1.3.2 用户空间(fuse)

Mount Ceph FS as with FUSE. Replace {username} with your username.
和 FUSE 一样挂载 ceph 文件系统，用你自己的用户名替代{username}。

```
sudo mkdir /home/{username}/cephfs
sudo ceph-fuse -m {ip-address-of-monitor}:6789 /home/{username}/cephfs
```

1.3.3 额外信息

Additional Information

See CephFS for additional information. CephFS is not quite as stable as the block device and the object storage gateway. Contact Inktank for details on running CephFS in a production environment.

更多信息参见 CephFS。 CephFS 不如块设备和对象存储网关稳定，要在生产环境运行 CephFS 请联系 Inktank。

1.4 对象存储快速入门

Object Storage Quick Start

To use this guide, you must have executed the procedures in the 5-minute Quick Start guide first.

这篇向导基于前面的 5 分钟快速入门。

1.4.1 安装 Apache 和 FastCGI

Install Apache and FastCGI

The Ceph object storage gateway runs on Apache and FastCGI. Install them on the server machine. Use the following procedure:

ceph 对象存储网关运行在 Apache 和 FastCGI 之上。按下列步骤安装到服务器上：

1. Install Apache and FastCGI on the server machine.

   在服务器上安装 Apacha 和 FastCGI。

   ```
   sudo apt-get update && sudo apt-get install apache2 libapache2-mod-fastcgi
   ```

2. Enable the URL rewrite modules for Apache and FastCGI.

   为 Apache 和 FastCGI 启用 URL 重写模块。

   ```
   sudo a2enmod rewrite
   sudo a2enmod fastcgi
   ```

3. Add a line for the ServerName in the /etc/apache2/httpd.conf file. Provide the fully qualified domain name of the server machine.

   在/etc/apache2/httpd.conf 里添加一行 ServerName，设置为服务器的全资域名 FQDN。

   ServerName {fqdn}

4. Restart Apache so that the foregoing changes take effect.

   重启 Apacha，以确保前面的更改生效。

   ```
   sudo service apache2 restart
   ```

1.4.2 安装 RADOS 网关

Install RADOS Gateway

Once you have installed and configured Apache and FastCGI, you may install Ceph's RADOS Gateway.

安装、配置好 Apache 和 FastCGI 后，还要安装 ceph 的 RADOS 网关。

```
sudo apt-get install radosgw
```

For details on the preceding steps, see RADOS Gateway Manual Install.

前述步骤的详情请参见……

1.4.3 修改 ceph 配置文件

Modify the Ceph Configuration File

On the server machine, perform the following steps:

在服务器上执行下面的步骤：

1. Open the Ceph configuration file.

   打开 ceph 配置文件。

   ```
   cd /etc/ceph
   vim ceph.conf
   ```

2. Add the following settings to the Ceph configuration file:

   把下面的配置添加到 ceph 配置文件里：

```
[client.radosgw.gateway]
host = {host-name}
keyring = /etc/ceph/keyring.radosgw.gateway
rgw socket path = /tmp/radosgw.sock
log file = /var/log/ceph/radosgw.log
```

3. Go to the client machine and copy the configuration file from the server machine to `/etc/ceph/ceph.conf` on your client machine.

   把服务器上的配置文件复制到客户端的`/etc/ceph/ceph.conf`。

```
sudo scp {user}@{cluster-machine}:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

Tip: Ensure the `ceph.conf` file has appropriate permissions set (e.g. `chmod 644`) on your client machine.
提示：确保客户端的 ceph.conf 设置了合适的权限位，如：chmod 644。

1.4.4 创建数据目录

Create a Data Directory

Create a data directory on the cluster server for the instance of `radosgw`.

在服务器上为 radosgw 例程创建数据目录。

```
sudo mkdir -p /var/lib/ceph/radosgw/ceph-radosgw.gateway
```

1.4.5 创建网关配置文件

Create a Gateway Configuration File

The example configuration file will configure the gateway to operate with the Apache FastCGI module, a rewrite rule for OpenStack Swift, and paths for the log files. To add a configuration file for the Ceph Gateway, we suggest copying the contents of the example file below to an editor. Then, follow the steps below to modify it.

下例会配置网关与 Apache FastCGI 模块运作、一个 OpenStack Swift 重写规则、日志文件路径。要配置 ceph 网关，我们建议把下面的样本复制到编辑器，然后按后面的步骤修改。

```
FastCgiExternalServer /var/www/s3gw.fcgi -socket /tmp/radosgw.sock

<VirtualHost *:80>
        ServerName {fqdn}
        ServerAdmin {email.address}
        DocumentRoot /var/www
</VirtualHost>

RewriteEngine On
RewriteRule ^/([a-zA-Z0-9-_.]*)([/]?.*) /s3gw.fcgi?page=$1&params=$2&%{QUERY_STRING}
[E=HTTP_AUTHORIZATION:%{HTTP:Authorization},L]

<VirtualHost *:80>

        <IfModule mod_fastcgi.c>
                <Directory /var/www>
                        Options +ExecCGI
                        AllowOverride All
                        SetHandler fastcgi-script
                        Order allow,deny
                        Allow from all
                        AuthBasicAuthoritative Off
                </Directory>
        </IfModule>

        AllowEncodedSlashes On
        ErrorLog /var/log/apache2/error.log
        CustomLog /var/log/apache2/access.log combined
        ServerSignature Off
</VirtualHost>
```

Replace the `{fqdn}` entry with the fully-qualified domain name of the server server.

用服务器的全资域名替换{fqdn}。

1. Replace the `{email.address}` entry with the email address for the server administrator.

   用管理员邮件地址替换{email.address}。

2. Save the contents to the `/etc/apache2/sites-available` directory on the server machine.

   把内容保存到服务器的/etc/apache2/sites-avaiable 目录下。

3. Enable the site for `rgw.conf`.

   启用 rgw.conf 配置的站点。

   ```
   sudo a2ensite rgw.conf
   ```

4. Disable the default site.

   禁用默认站点。

   ```
   sudo a2dissite default
   ```

See Create rgw.conf for additional details.

详情参见 Create rgw.conf。

1.4.6 添加 FastCGI 脚本

Add a FastCGI Script

FastCGI requires a script for the S3-compatible interface. To create the script, execute the following procedures on the server machine.

S3 兼容接口需要一个 FastCGI 脚本，执行下列步骤创建脚本：

1. Go to the `/var/www` directory.

   进入/var/www 目录。

   ```
   cd /var/www
   ```

2. Open an editor with the file name `s3gw.fcgi`.

   在编辑器里创建 s3gw.fcgi 空文件。

   ```
   sudo vim s3gw.fcgi
   ```

3. Copy the following into the editor.

   把下面的内容复制到编辑器。

   ```
   #!/bin/sh
   exec /usr/bin/radosgw -c /etc/ceph/ceph.conf -n client.radosgw.gateway
   ```

4. Save the file.

   保存文件。

5. Change the permissions on the file so that it is executable.

   给文件增加可执行权限位。

   ```
   sudo chmod +x s3gw.fcgi
   ```

1.4.7 生成密钥环和密钥

Generate a Keyring and Key

Perform the following steps on the server machine.

在服务器上执行下列步骤。

1. Create a keyring for the RADOS Gateway.

给 RADOS 网关创建密钥环。

```
sudo ceph-authtool --create-keyring /etc/ceph/keyring.radosgw.gateway
sudo chmod +r /etc/ceph/keyring.radosgw.gateway
```

2. Create a key for the RADOS Gateway to authenticate with the cluster.

   创建一个密钥用于 RADOS 到集群的认证。

```
sudo ceph-authtool /etc/ceph/keyring.radosgw.gateway -n client.radosgw.gateway --gen-key
sudo ceph-authtool -n client.radosgw.gateway --cap osd 'allow rwx' --cap mon 'allow r'
/etc/ceph/keyring.radosgw.gateway
```

3. Add the key to the Ceph keyring.

   把密钥添加到 ceph 密钥环。

```
sudo ceph -k /etc/ceph/ceph.keyring auth add client.radosgw.gateway -i
/etc/ceph/keyring.radosgw.gateway
```

1.4.8 创建用户

Create a User

To use the Gateway, you must create a Gateway user. First, create a gateway user for the S3-compatible interface; then, create a subuser for the Swift-compatible interface.

要使用网关，必须有网关用户。首先给 S3 兼容接口创建一个网关用户，然后给 Swift 兼容接口创建一个子用户。

S3 网关用户

Gateway (S3) User

First, create a Gateway user for the S3-compatible interface.

首先，给 S3 兼容接口创建网关用户。

```
sudo radosgw-admin user create --uid="{username}" --display-name="{Display Name}"
```

For example:

例如：

```
radosgw-admin user create --uid=johndoe --display-name="John Doe" --email=john@example.com
```

```
{ "user_id": "johndoe",
  "rados_uid": 0,
  "display_name": "John Doe",
  "email": "john@example.com",
  "suspended": 0,
  "subusers": [],
  "keys": [
    { "user": "johndoe",
      "access_key": "QFAMEDSJP5DEKJO0DDXY",
      "secret_key": "iaSFLDVvDdQt6lkNzHyW4fPLZugBAI1g17LO0+87"}],
  "swift_keys": []
 }
```

Creating a user creates an access_key and secret_key entry for use with any S3 API-compatible client.

创建用户时也创建了给 S3 兼容 API 客户端用的对应访问密钥和私钥。

Important: Check the key output. Sometimes radosgw-admin generates a key with an escape () character, and some clients do not know how to handle escape characters. Remedies include removing the escape character (), encapsulating the string in quotes, or simply regenerating the key and ensuring that it does not have an escape character.
重要：验证下密钥输出。有时候 radosgw-admin 生成了包含 escape 字符的密钥，而有些客户端不知道如何处理它们。矫正包括删除 escape 字符、把字符串放入引号，或者干脆重新生成密钥，并再次确认。

子用户

Subuser

Next, create a subuser for the Swift-compatible interface.

接下来，创建给 Swift 兼容接口用的子用户。

```
sudo radosgw-admin subuser create --uid=johndoe --subuser=johndoe:swift --access=full
{ "user_id": "johndoe",
  "rados_uid": 0,
  "display_name": "John Doe",
  "email": "john@example.com",
  "suspended": 0,
  "subusers": [
    { "id": "johndoe:swift",
      "permissions": "full-control"}],
  "keys": [
    { "user": "johndoe",
      "access_key": "QFAMEDSJP5DEKJO0DDXY",
      "secret_key": "iaSFLDVvDdQt6lkNzHyW4fPLZugBAI1g17LO0+87"}],
  "swift_keys": []}
```

```
sudo radosgw-admin key create --subuser=johndoe:swift --key-type=swift
{ "user_id": "johndoe",
  "rados_uid": 0,
  "display_name": "John Doe",
  "email": "john@example.com",
  "suspended": 0,
  "subusers": [
     { "id": "johndoe:swift",
       "permissions": "full-control"}],
  "keys": [
    { "user": "johndoe",
      "access_key": "QFAMEDSJP5DEKJO0DDXY",
      "secret_key": "iaSFLDVvDdQt6lkNzHyW4fPLZugBAI1g17LO0+87"}],
  "swift_keys": [
    { "user": "johndoe:swift",
      "secret_key": "E9T2rUZNu2gxUjcwUBO8n\/Ev4KX6\/GprEuH4qhu1"}]}
```

This step enables you to use any Swift client to connect to and use RADOS Gateway via the Swift-compatible API.

这一步使得你能用任何 Swift 客户端通过 Swift 兼容 API 连接和使用 RADOS 网关。

RGW's `user:subuser` tuple maps to the `tenant:user` tuple expected by Swift.

RGW 的 user:subuser 元组和 Swift 期望的一样映射到了 tenant:user 元组。

Note: RGW's Swift authentication service only supports built-in Swift authentication (`-V 1.0`) at this point. See RGW Configuration for Keystone integration details.
注意：当前 RGW 的 Swift 认证服务只支持内建的 Swift 认证（-V 1.0），要和 Keystone 集成请参见……

1.4.9 启用 SSL

Enable SSL

Some REST clients use HTTPS by default. So you should consider enabling SSL for Apache on the server machine.

有些 REST 客户端默认使用 HTTPS，所以你应该考虑开启 Apache 上的 SSL。

```
sudo a2enmod ssl
```

Once you enable SSL, you should generate an SSL certificate.

启用 SSL 后，应该生成一个 SSL 证书。

```
sudo mkdir /etc/apache2/ssl
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key
-out  /etc/apache2/ssl/apache.crt
```

Then, restart Apache.

然后重启 Apache。

```
service apache2 restart
```

1.5 加入 ceph 社区

Get Involved in the Ceph Community!

These are exciting times in the Ceph community! Get involved!
在 ceph 社区里有激动人心的时刻，加入吧！

频道	简介	联系信息
Blog	Check the Ceph Blog periodically to keep track of Ceph progress and important announcements. 经常检查 blog 来跟踪 ceph 进展和重要通告。	http://ceph.com/community/blog/
IRC	As you delve into Ceph, you may have questions or feedback for the Ceph development team. Ceph developers are often available on the `#ceph` IRC channel particularly during daytime hours in the US Pacific Standard Time zone. 随着您对 ceph 的深入了解，您也许有问题或回馈给 ceph 开发团队。ceph 开发者经常在 IRC 的#ceph 频道，尤其是美国太平洋标准时区白天工作时间。	• **Domain:** `irc.oftc.net` • **Channel:** `#ceph`
Email List **邮件列表**	Keep in touch with developer activity by subscribing to the email list at ceph-devel@vger.kernel.org. You can opt out of the email list at any time byunsubscribing. A simple email is all it takes! If you would like to view the archives, go to Gmane. 你可以订阅位于 ceph-devel@vger.kernel.org 的邮件列表来和开发者保持联系，也可以随时离开。您也可以到 Gmane 查看历史存档。	• Subscribe • Unsubscribe • Gmane
Bug Tracker **问题跟踪**	You can help keep Ceph production worthy by filing and tracking bugs, and providing feature requests using the Bug Tracker. 您可以使用问题跟踪系统来帮助我们提升 ceph 稳定性，或提出新功能申请。	http://tracker.newdream.net/projects/ceph
Source Code **源码**	If you would like to participate in development, bug fixing, or if you just want the very latest code for Ceph, you can get it at http://github.com. See Ceph Source Code for details on cloning from github. 如果您想参与开发、问题修正，或者想要最新源码，您可以从 http://github.com 获取，参见 Ceph Source Code 如何从 github 克隆源码。	• http://github.com:ceph/ceph • http://ceph.com/download
Support **支持**	If you have a very specific problem, an immediate need, or if your deployment requires significant help, consider commercial support. 如果您有特殊问题、急切的需求、或者您的部署需要大量帮助，请考虑商业支持。	http://inktank.com

1.6 手动安装 ceph

Installing Ceph Manually
Ceph is intended for large-scale deployments, but you may install Ceph on a single host. This guide is intended for Debian/Ubuntu Linux distributions.
ceph 为大规模部署设计，但是仍然能安装在单机上。这篇手册适用于 Debian/Ubuntu Linux 发行版。

1. Install Ceph packages
2. Create a `ceph.conf` file. See Ceph Configuration Files for details.
3. Deploy the Ceph configuration. See Deploy with mkcephfs for details.
4. Start a Ceph cluster. See Starting a Cluster for details.
5. Mount Ceph FS. See Ceph FS for details.

2 安装

Installation

The Ceph Object Store is the foundation of all Ceph clusters, and it consists primarily of two types of daemons: Object Storage Daemons (OSDs) and monitors. The Ceph Object Store is based upon the concept of RADOS, which eliminates single points of failure and delivers infinite scalability. For details on the architecture of Ceph and RADOS, refer to Ceph Architecture. All Ceph deployments have OSDs and monitors, so you should prepare your Ceph cluster by focusing first on the object storage cluster.
ceph 对象存储是所有 ceph 集群的基础，它主要包含两种类型的守护进程：对象存储守护进程（Object Storage Daemons, OSDs）和监视器。ceph 对象存储运行于 RADOS 概念之上，它避免了单点故障、实现了无限的扩展能力。要了解 ceph 体系结构和 RADOS 的详情，请参见 ceph 体系结构部分。所有 ceph 的部署都有 OSD 和监视器，所以你首先应该关注对象存储集群，再准备 ceph 集群。
Recommendations

To begin using Ceph in production, you should review our hardware recommendations and operating system recommendations. Many of the frequently-asked questions in our mailing list involve hardware-related questions and how to install Ceph on various distributions.

忠告

要在生产环境下使用 ceph，你得重新审阅硬件推荐和操作系统推荐部分，这些问题经常在我们的邮件列表里提及，以及如何在各种发行版上安装 ceph。

2.1 硬件推荐

Hardware Recommendations

Ceph was designed to run on commodity hardware, which makes building and maintaining petabyte-scale data clusters economically feasible. When planning out your cluster hardware, you will need to balance a number of considerations, including failure domains and potential performance issues. Hardware planning should include distributing Ceph daemons and other processes that use Ceph across many hosts. Generally, we recommend running Ceph daemons of a specific type on a host configured for that type of daemon. We recommend using other hosts for processes that utilize your data cluster (e.g., OpenStack, CloudStack, etc).
ceph 为普通硬件设计，这可使构建、维护 PB 级数据集群的费用相对低廉。规划集群硬件时，需要均衡几方面的因素，包括区域失效和潜在的性能问题。硬件规划要包含把使用 ceph 集群的 ceph 守护进程和其他进程恰当分布。通常，我们推荐在一台机器上只运行一种类型的守护进程。我们推荐把使用数据集群的进程（如 OpenStack、CloudStack 等）安装在别的机器上。

Inktank provides excellent premium support for hardware planning.
Inktank 可提供优秀的硬件规划支持。

Tip: Check out the Ceph blog too. Articles like Ceph Write Throughput 1, Ceph Write Throughput 2, Argonaut v. Bobtail Performance Preview, Bobtail Performance - I/O Scheduler Comparison and others are an excellent source of information.

提示：关于 Ceph 的高品质 blog 文章也值得参考，如 Ceph Write Throughput 1, Ceph Write Throughput 2, Argonaut v. Bobtail Performance Preview, Bobtail Performance - I/O Scheduler Comparison。

2.1.1 CPU

Ceph metadata servers dynamically redistribute their load, which is CPU intensive. So your metadata servers should have significant processing power (e.g., quad core or better CPUs). Ceph OSDs run the RADOS service, calculate data placement with CRUSH, replicate data, and maintain their own copy of the cluster map. Therefore, OSDs should have a reasonable amount of processing power (e.g., dual core processors). Monitors simply maintain a master copy of the cluster map, so they are not CPU intensive. You must also consider whether the host machine will run CPU-intensive processes in addition to Ceph daemons. For example, if your hosts will run computing VMs (e.g., OpenStack Nova), you will need to ensure that these other processes leave sufficient processing power for Ceph daemons. We recommend running additional CPU-intensive processes on separate hosts.

ceph 元数据服务器对 CPU 敏感，它会动态地重分布它们的负载，所以你的元数据服务器应该有足够的处理能力（如 4 核或更强悍的 CPU）。ceph 的 OSD 运行着 RADOS 服务、用 CRUSH 计算数据存放位置、复制数据、维护它自己的集群运行图副本，因此 OSD 需要一定的处理能力（如双核 CPU）。监视器只简单地维护着集群运行图的副本，因此对 CPU 不敏感；但必须考虑机器以后是否还会运行 ceph 监视器以外的 CPU 密集型任务。例如，如果服务器以后要运行用于计算的虚拟机（如 OpenStack Nova），你就要确保给 ceph 进程保留了足够的处理能力，所以我们推荐在其他机器上运行 CPU 密集型任务。

2.1.2 内存

RAM

Metadata servers and monitors must be capable of serving their data quickly, so they should have plenty of RAM (e.g., 1GB of RAM per daemon instance). OSDs do not require as much RAM for regular operations (e.g., 200MB of RAM per daemon instance); however, during recovery they need significantly more RAM (e.g., 500MB-1GB). Generally, more RAM is better.

元数据服务器和监视器必须可以尽快地提供它们的数据，所以他们应该有足够的内存，至少每进程 1GB。运行 OSD 的服务器不需要那么多的内存，每进程 500MB 差不多了。通常内存越多越好。

2.1.3 数据存储

Data Storage

Plan your data storage configuration carefully. There are significant cost and performance tradeoffs to consider when planning for data storage. Simultaneous OS operations, and simultaneous request for read and write operations from multiple daemons against a single drive can slow performance considerably. There are also file system limitations to consider: btrfs is not quite stable enough for production, but it has the ability to journal and write data simultaneously, whereas XFS and ext4 do not.

要谨慎地规划数据存储配置，因为其间涉及明显的成本和性能折衷。来自操作系统的并行操作和到单个硬盘的多个守护进程并发读、写请求操作会极大地降低性能。文件系统局限性也要考虑：btrfs 尚未稳定到可以用于生产环境的程度，但它可以同时记日志并写入数据，而 xfs 和 ext4 却不能。

Important: Since Ceph has to write all data to the journal before it can send an ACK (for XFS and EXT4 at least), having the journals and OSD performance in balance is really important!
重要：因为 ceph 发送 ACK 前必须把所有数据写入日志（至少对 xfs 和 ext4 来说是），因此均衡日志和 OSD 性能相当重要。

2.1.3.1 硬盘

Hard Disk Drives

OSDs should have plenty of hard disk drive space for object data. We recommend a minimum hard disk drive size of 1 terabyte. Consider the cost-per-gigabyte advantage of larger disks. We recommend dividing the price of the hard disk drive by the number of gigabytes to arrive at a cost per gigabyte, because larger drives may have a significant impact on the cost-per-gigabyte. For example, a 1 terabyte hard disk priced at $75.00 has a

cost of $0.07 per gigabyte (i.e., $75 / 1024 = 0.0732). By contrast, a 3 terabyte hard disk priced at $150.00 has a cost of $0.05 per gigabyte (i.e., $150 / 3072 = 0.0488). In the foregoing example, using the 1 terabyte disks would generally increase the cost per gigabyte by 40%–rendering your cluster substantially less cost efficient.

OSD 应该有足够的空间用于存储对象数据。考虑到大硬盘的每 GB 成本，我们建议用至少 1TB 的硬盘。建议用 GB 数除以硬盘价格来计算每 GB 成本，因为较大的硬盘通常会对每 GB 成本有较大影响，例如，单价为$75 的 1TB 硬盘其每 GB 价格为$0.07（$75/1024=0.0732），又如单价为$150 的 3TB 硬盘其每 GB 价格为$0.05（$150/3072=0.0488），这样使用 1TB 硬盘会增加 40%的每 GB 价格，它将表现为较低的经济性。

Tip: Running multiple OSDs on a single disk–irrespective of partitions–is **NOT** a good idea.
提示：不顾分区而在单个硬盘上运行多个 OSD，这样不明智！

Tip: Running an OSD and a monitor or a metadata server on a single disk–irrespective of partitions–is **NOT** a good idea either.
提示：不顾分区而在运行了 OSD 的硬盘上同时运行监视器或元数据服务器也不明智！

Storage drives are subject to limitations on seek time, access time, read and write times, as well as total throughput. These physical limitations affect overall system performance–especially during recovery. We recommend using a dedicated drive for the operating system and software, and one drive for each OSD daemon you run on the host. Most “slow OSD” issues arise due to running an operating system, multiple OSDs, and/or multiple journals on the same drive. Since the cost of troubleshooting performance issues on a small cluster likely exceeds the cost of the extra disk drives, you can accelerate your cluster design planning by avoiding the temptation to overtax the OSD storage drives.

存储驱动器受限于寻道时间、访问时间、读写时间、还有总吞吐量，这些物理局限性影响着整体系统性能，尤其在系统恢复期间。因此我们推荐独立的驱动器用于安装操作系统和软件，另外每个 OSD 守护进程占用一个驱动器。大多数“slow OSD”问题的起因都是在相同的硬盘上运行了操作系统、多个 OSD、和/或多个日志文件。鉴于解决性能问题的成本差不多会超过另外增加磁盘驱动器，你应该在设计时就避免增加 OSD 存储驱动器的负担来提升性能。

You may run multiple OSDs per hard disk drive, but this will likely lead to resource contention and diminish the overall throughput. You may store a journal and object data on the same drive, but this may increase the time it takes to journal a write and ACK to the client. Ceph must write to the journal before it can ACK the write. The btrfs filesystem can write journal data and object data simultaneously, whereas XFS and ext4 cannot.

ceph 允许你在每块硬盘驱动器上运行多个 OSD，但这会导致资源竞争并降低总体吞吐量；ceph 也允许把日志和对象数据存储在相同驱动器上，但这会增加记录写日志并回应客户端的延时，因为 ceph 必须先写入日志才会回应确认了写动作。btrfs 文件系统能同时写入日志数据和对象数据，xfs 和 ext4 却不能。

Ceph best practices dictate that you should run operating systems, OSD data and OSD journals on separate drives.

ceph 最佳实践指示，你应该分别在单独的硬盘运行操作系统、OSD 数据和 OSD 日志。

2.1.3.2 固态硬盘

Solid State Drives

One opportunity for performance improvement is to use solid-state drives (SSDs) to reduce random access time and read latency while accelerating throughput. SSDs often cost more than 10x as much per gigabyte when compared to a hard disk drive, but SSDs often exhibit access times that are at least 100x faster than a hard disk drive.

一种提升性能的方法是使用固态硬盘（SSD）来降低随机访问时间和读延时，同时增加吞吐量。SSD 和硬盘相比每 GB 成本通常要高 10 倍以上，但访问时间至少比硬盘快 100 倍。

SSDs do not have moving mechanical parts so they aren’t necessarily subject to the same types of limitations as hard disk drives. SSDs do have significant limitations though. When evaluating SSDs, it is important to

consider the performance of sequential reads and writes. An SSD that has 400MB/s sequential write throughput may have much better performance than an SSD with 120MB/s of sequential write throughput when storing multiple journals for multiple OSDs.

SSD 没有可移动机械部件，所以不存在和硬盘一样的局限性。但 SSD 也有局限性，评估 SSD 时，顺序读写性能很重要，在为多个 OSD 存储日志时，有着 400MB/s 顺序读写吞吐量的 SSD 其性能远高于 120MB/s 的。

> Important: We recommend exploring the use of SSDs to improve performance. However, before making a significant investment in SSDs, we **strongly recommend** both reviewing the performance metrics of an SSD and testing the SSD in a test configuration to gauge performance.
> 重要：我们建议发掘 SSD 的用法来提升性能。然而在大量投入 SSD 前，我们**强烈建议**核实 SSD 的性能指标，并在测试环境下衡量性能。

Since SSDs have no moving mechanical parts, it makes sense to use them in the areas of Ceph that do not use a lot of storage space. Relatively inexpensive SSDs may appeal to your sense of economy. Use caution. Acceptable IOPS are not enough when selecting an SSD for use with Ceph. There are a few important performance considerations for journals and SSDs:

正因为 SSD 没有移动机械部件，所以它很适合 ceph 里不需要太多存储空间的地方。相对廉价的 SSD 很诱人，慎用！可接受的 IOPS 指标对选择用于 ceph 的 SSD 还不够，用于日志和 SSD 时还有几个重要考量：

- **Write-intensive semantics:** Journaling involves write-intensive semantics, so you should ensure that the SSD you choose to deploy will perform equal to or better than a hard disk drive when writing data. Inexpensive SSDs may introduce write latency even as they accelerate access time, because sometimes high performance hard drives can write as fast or faster than some of the more economical SSDs available on the market!
  写密集语义：记日志涉及写密集语义，所以你要确保选用的 SSD 写入性能和硬盘相当或好于硬盘。廉价 SSD 可能在加速访问的同时引入写延时，有时候高性能硬盘的写入速度可以和便宜 SSD 相媲美。
- **Sequential Writes:** When you store multiple journals on an SSD you must consider the sequential write limitations of the SSD too, since they may be handling requests to write to multiple OSD journals simultaneously.
  顺序写入：在一个 SSD 上为多个 OSD 存储多个日志时也必须考虑 SSD 的顺序写入极限，因为它们要同时处理多个 OSD 日志的写入请求。
- **Partition Alignment:** A common problem with SSD performance is that people like to partition drives as a best practice, but they often overlook proper partition alignment with SSDs, which can cause SSDs to transfer data much more slowly. Ensure that SSD partitions are properly aligned.

  分区对齐：采用了 SSD 的一个常见问题是人们喜欢分区，却常常忽略了分区对齐，这会导致 SSD 的数据传输速率慢很多，所以请确保分区对齐了。

While SSDs are cost prohibitive for object storage, OSDs may see a significant performance improvement by storing an OSD's journal on an SSD and the OSD's object data on a separate hard disk drive. The `osd journal` configuration setting defaults to `/var/lib/ceph/osd/$cluster-$id/journal`. You can mount this path to an SSD or to an SSD partition so that it is not merely a file on the same disk as the object data.

SSD 用于对象存储太昂贵了，但是把 OSD 的日志存到 SSD、把对象数据存储到独立的硬盘可以明显提升性能。osd journal 选项的默认值是/var/lib/ceph/osd/$cluster-$id/journal，你可以把它挂载到一个 SSD 或 SSD 分区，这样它就不再是和对象数据一样存储在同一个硬盘上的文件了。

One way Ceph accelerates CephFS filesystem performance is to segregate the storage of CephFS metadata from the storage of the CephFS file contents. Ceph provides a default `metadata` pool for CephFS metadata. You will never have to create a pool for CephFS metadata, but you can create a CRUSH map hierarchy for your CephFS metadata pool that points only to a host's SSD storage media. See Mapping Pools to Different Types of OSDs for details.

提升 CephFS 文件系统性能的一种方法是从 CephFS 文件内容里分离出元数据。ceph 提供了默认的 metadata 存储池来存储 CephFS 元数据，所以你不需要给 CephFS 元数据创建存储池，但是可以给它创建一个仅指向某主机 SSD 的 CRUSH 运行图。详情见 Mapping Pools to Different Types of OSDs。

2.1.3.3 控制器

Controllers

Disk controllers also have a significant impact on write throughput. Carefully, consider your selection of disk controllers to ensure that they do not create a performance bottleneck.

硬盘控制器对写吞吐量也有显著影响，要谨慎地选择，以免产生性能瓶颈。

> Tip: The Ceph blog is often an excellent source of information on Ceph performance issues. See Ceph Write Throughput 1 and Ceph Write Throughput 2 for additional details.
> 提示：ceph blog 通常是优秀的 ceph 性能问题来源，见 Ceph Write Throughput 1 和 Ceph Write Throughput 2。

2.1.3.4 其他注意事项

Additional Considerations

You may run multiple OSDs per host, but you should ensure that the sum of the total throughput of your OSD hard disks doesn't exceed the network bandwidth required to service a client's need to read or write data. You should also consider what percentage of the overall data the cluster stores on each host. If the percentage on a particular host is large and the host fails, it can lead to problems such as exceeding the `full ratio`, which causes Ceph to halt operations as a safety precaution that prevents data loss.

你可以在同一主机上运行多个 OSD，但要确保 OSD 硬盘总吞吐量不超过为客户端提供读写服务所需的网络带宽；还要考虑集群在每台主机上所存储的数据占总体的百分比，如果一台主机所占百分比太大而它挂了，就可能导致诸如超过 full ratio 的问题，此问题会使 ceph 中止运作以防数据丢失。

When you run multiple OSDs per host, you also need to ensure that the kernel is up to date. See OS Recommendations for notes on `glibc` and `syncfs(2)` to ensure that your hardware performs as expected when running multiple OSDs per host.

如果每台主机运行多个 OSD，也得保证内核是最新的。参阅 OS Recommendations 里关于 glibc 和 syncfs(2)的部分，确保硬件性能可达期望值。

2.1.4 网络

Networks

We recommend that each host have at least two 1Gbps network interface controllers (NICs). Since most commodity hard disk drives have a throughput of approximately 100MB/second, your NICs should be able to handle the traffic for the OSD disks on your host. We recommend a minimum of two NICs to account for a public (front-side) network and a cluster (back-side) network. A cluster network (preferably not connected to the internet) handles the additional load for data replication and helps stop denial of service attacks that prevent the cluster from achieving `active + clean` states for placement groups as OSDs replicate data across the cluster. Consider starting with a 10Gbps network in your racks. Replicating 1TB of data across a 1Gbps network takes 3 hours, and 3TBs (a typical drive configuration) takes 9 hours. By contrast, with a 10Gbps network, the replication times would be 20 minutes and 1 hour respectively. In a petabyte-scale cluster, failure of an OSD disk should be an expectation, not an exception. System administrators will appreciate PGs recovering from a `degraded` state to an `active + clean` state as rapidly as possible, with price / performance tradeoffs taken into consideration. Additionally, some deployment tools (e.g., Dell's Crowbar) deploy with five different networks, but employ VLANs to make hardware and network cabling more manageable. VLANs using 802.1q protocol require VLAN-capable NICs and Switches. The added hardware expense may be offset by the operational cost savings for network setup and maintenance. When using VLANs to handle VM traffic between between the cluster and compute stacks (e.g., OpenStack, CloudStack, etc.), it is also worth considering using 10G Ethernet. Top-of-rack routers for each network also need to be able to communicate with spine routers that have even faster throughput–e.g., 40Gbps to 100Gbps.

建议每台机器最少两个千兆网卡，现在大多数机械硬盘都能达到大概 100MB/s 的吞吐量，网卡应该能处理所有 OSD 硬盘总吞吐量，所以推荐最少两个千兆网卡，分别用于公网（前端）和集群网络（后端）。集群网络（最好别连接到国际互联网）用于处理由数据复制产生的额外负载，而且可防止拒绝服务攻击，拒绝服务攻击会干扰数据归置组，使之在 OSD 数据复制时不能回到 active+clean 状态。请考虑部署万兆网卡。通过 1Gbps

网络复制 1TB 数据耗时 3 小时，而 3TB（典型配置）需要 9 小时，相比之下，如果使用 10Gbps 复制时间可分别缩减到 20 分钟和 1 小时。在一个 PB 级集群中，OSD 磁盘失败是常态，而非异常；在性价比合理的的前提下，系统管理员想让 PG 尽快从 degraded（降级）状态恢复到 active+clean 状态。另外，一些部署工具（如 Dell 的 Crowbar）部署了 5 个不同的网络，但使用了 VLAN 以提高网络和硬件可管理性。VLAN 使用 802.1q 协议，还需要采用支持 VLAN 功能的网卡和交换机，增加的硬件成本可用节省的运营（网络安装、维护）成本抵消。使用 VLAN 来处理集群和计算栈（如 OpenStack、CloudStack 等等）之间的 VM 流量时，采用 10G 网卡仍然值得。每个网络的机架路由器到核心路由器应该有更大的带宽，如 40Gbps 到 100Gbps。

Your server hardware should have a Baseboard Management Controller (BMC). Administration and deployment tools may also use BMCs extensively, so consider the cost/benefit tradeoff of an out-of-band network for administration. Hypervisor SSH access, VM image uploads, OS image installs, management sockets, etc. can impose significant loads on a network. Running three networks may seem like overkill, but each traffic path represents a potential capacity, throughput and/or performance bottleneck that you should carefully consider before deploying a large scale data cluster.

服务器应配置底板管理控制器（Baseboard Management Controller, BMC），管理和部署工具也应该大规模使用 BMC，所以请考虑带外网络管理的成本/效益平衡，此程序管理着 SSH 访问、VM 映像上传、操作系统安装、端口管理、等等，会徒增网络负载。运营 3 个网络有点过分，但是每条流量路径都指示了部署一个大型数据集群前要仔细考虑的潜能力、吞吐量、性能瓶颈。

2.1.5 故障域

A failure domain is any failure that prevents access to one or more OSDs. That could be a stopped daemon on a host; a hard disk failure, an OS crash, a malfunctioning NIC, a failed power supply, a network outage, a power outage, and so forth. When planning out your hardware needs, you must balance the temptation to reduce costs by placing too many responsibilities into too few failure domains, and the added costs of isolating every potential failure domain.

故障域指任何导致不能访问一个或多个 OSD 的故障，可以是主机上停止的进程、硬盘故障、操作系统崩溃、有问题的网卡、损坏的电源、断网、断电等等。规划硬件需求时，要在多个需求间寻求平衡点，像付出很多努力减少故障域带来的成本削减、隔离每个潜在故障域增加的成本。

2.1.6 最低硬件推荐

Minimum Hardware Recommendations

Ceph can run on inexpensive commodity hardware. Small production clusters and development clusters can run successfully with modest hardware.

ceph 可以运行在廉价的普通硬件上，小型生产集群和开发集群可以在一般的硬件上。

Process	Criteria	Minimum Recommended
`ceph-osd`	Processor	1x 64-bit AMD-64/i386 dual-core
	RAM	500 MB per daemon
	Volume Storage	1x Disk per daemon
	Network	2x 1GB Ethernet NICs
`ceph-mon`	Processor	1x 64-bit AMD-64/i386
	RAM	1 GB per daemon
	Disk Space	10 GB per daemon
	Network	2x 1GB Ethernet NICs
`ceph-mds`	Processor	1x 64-bit AMD-64/i386 quad-core
	RAM	1 GB minimum per daemon
	Disk Space	1 MB per daemon
	Network	2x 1GB Ethernet NICs

Tip: If you are running an OSD with a single disk, create a partition for your volume storage that is separate from the partition containing the OS. Generally, we recommend separate disks for the OS and the volume storage.

提示：如果在只有一块硬盘的机器上运行 OSD，要把数据和操作系统分别放到不同分区；一般来说，我们推荐操作系统和数据分别使用不同的硬盘。

2.1.7 生产集群实例

Production Cluster Example

Production clusters for petabyte scale data storage may also use commodity hardware, but should have considerably more memory, processing power and data storage to account for heavy traffic loads.

PB 级生产集群也可以使用普通硬件，但应该配备更多内存、CPU 和数据存储空间来解决流量压力。

A recent (2012) Ceph cluster project is using two fairly robust hardware configurations for Ceph OSDs, and a lighter configuration for monitors.

一个最新（2012）的 ceph 集群项目使用了 2 个相当强悍的 OSD 硬件配置，和稍逊的监视器配置。

Configuration	Criteria	Minimum Recommended
Dell PE R510	Processor	2x 64-bit quad-core Xeon CPUs
	RAM	16 GB
	Volume Storage	8x 2TB drives. 1 OS, 7 Storage
	Client Network	2x 1GB Ethernet NICs
	OSD Network	2x 1GB Ethernet NICs
	Mgmt. Network	2x 1GB Ethernet NICs
Dell PE R515	Processor	1x hex-core Opteron CPU
	RAM	16 GB
	Volume Storage	12x 3TB drives. Storage
	OS Storage	1x 500GB drive. Operating System.
	Client Network	2x 1GB Ethernet NICs
	OSD Network	2x 1GB Ethernet NICs
	Mgmt. Network	2x 1GB Ethernet NICs

2.2 推荐操作系统

OS Recommendations

2.2.1 CEPH 依赖

As a general rule, we recommend deploying Ceph on newer releases of Linux.

在较新的 Linux 发行版上部署 ceph，这是我们推荐的通用法则。

Linux 内核

- **Ceph Kernel Client:** We currently recommend:
  - v3.6.6 or later in the v3.6 stable series
  - v3.4.20 or later in the v3.4 stable series
- **btrfs**: If you use the `btrfs` file system with Ceph, we recommend using a recent Linux kernel (v3.5 or later).

  btrfs 文件系统：如果您想在 btrfs 上运行 ceph，我们推荐使用一个最新的 Linux 内核（v3.5 或更新）。

GLIBC

- **syncfs(2)**: For non-btrfs filesystems such as XFS and ext4 where more than one `ceph-osd` daemon is used on a single server, Ceph performs signficantly better with the `syncfs(2)` system call (added in kernel 2.6.39 and glibc 2.14). New versions of Ceph (v0.55 and later) do not depend on glibc support.

  syncfs(2):对非 btrfs 文件系统（像 XFS 和 ext4）而言，在一台服务器上运行了多个 ceph-osd 守护进程时，ceph 使用 syncfs(2) 系统调用时效率高得多（此功能在 2.6.39 内核和 glibc-2.14 加入）。

2.2.2 系统平台

Platforms

The charts below show how Ceph's requirements map onto various Linux platforms. Generally speaking, there is very little dependence on specific distributions aside from the kernel and system initialization package (i.e., sysvinit, upstart, systemd).

下面的表格展示了 ceph 需求和各种 Linux 发行版的对应关系。一般来说，ceph 对内核和系统初始化阶段的依赖很少（如 sysvinit, upstart, systemd）。

ARGONAUT (0.48)

Distro	Release	Code Name	Kernel	Notes	Testing
Ubuntu	11.04	Natty Narwhal	linux-2.6.38	1, 2, 3	B
Ubuntu	11.10	Oneric Ocelot	linux-3.0.0	1, 2, 3	B
Ubuntu	12.04	Precise Pangolin	linux-3.2.0	1, 2	B, I, C
Ubuntu	12.10	Quantal Quetzal	linux-3.5.4	2	B
Debian	6.0	Squeeze	linux-2.6.32	1, 2	B
Debian	7.0	Wheezy	linux-3.2.0	1, 2	B

BOBTAIL (0.56)

Distro	Release	Code Name	Kernel	Notes	Testing
Ubuntu	11.04	Natty Narwhal	linux-2.6.38	1, 2, 3	B
Ubuntu	11.10	Oneric Ocelot	linux-3.0.0	1, 2, 3	B
Ubuntu	12.04	Precise Pangolin	linux-3.2.0	1, 2	B, I, C
Ubuntu	12.10	Quantal Quetzal	linux-3.5.4	2	B
Debian	6.0	Squeeze	linux-2.6.32	1, 2	B
Debian	7.0	Wheezy	linux-3.2.0	1, 2	B
CentOS	6.3	N/A	linux-2.6.32	1, 2, 3	B, I
Fedora	17.0	Beefy Miracle	linux-3.3.4	1, 2	B
Fedora	18.0	Spherical Cow	linux-3.6.0		B
OpenSuse	12.2	N/A	linux-3.4.0	2	B

NOTES

附注

1: The default kernel has an older version of btrfs that we do not recommend for ceph-osd storage nodes. Upgrade to a recommended kernel or use XFS or ext4.

2: The default kernel has an old Ceph client that we do not recommend for kernel client (kernel RBD or the Ceph file system). Upgrade to a recommended kernel.

3: The installed version of glibc does not support the syncfs(2) system call. Putting multiple ceph-osd daemons using XFS or ext4 on the same host will not perform as well as they could.

1、默认内核 btrfs 版本较老，不推荐用于 ceph-osd 存储节点；要升级到推荐的内核，或者改用 xfs、ext4。

2、默认内核带的 ceph 客户端较老，不推荐做内核空间客户端（内核 RBD 或 ceph 文件系统），请升级到推荐内核。

3、已安装的 glibc 版本不支持 syncfs(2) 系统调用，同一台机器上使用 xfs 或 ext4 的 ceph-osd 守护进程性能一般，它可以更好。

TESTING

测试

B: We continuously build all branches on this platform and exercise basic unit tests. We build release packages for this platform.

I: We do basic installation and functionality tests of releases on this platform.

C: We run a comprehensive functional, regression, and stress test suite on this platform on a continuous basis. This includes development branches, pre-release, and released code.

B：我们持续地在这个平台上编译所有分支、做基本单元测试；也为这个平台构建可发布软件包。

I：我们在这个平台上做基本的安装和功能测试。

C：我们在这个平台上持续地做全面的功能、退化、压力测试，包括开发分支、预发布版本、正式发布版本。

2.3 Debian/Ubuntu 包的安装

Installing Debian/Ubuntu Packages

You may install stable release packages (for stable deployments), development release packages (for the latest features), or development testing packages (for development and QA only). Do not add multiple package sources at the same time.

你可以选择安装稳定发行包（用于稳定部署）、开发发行包（用于发掘最新功能）、或开发测试包（仅用于开发和质检）。不要同时添加多个软件源。

2.3.1 安装发布密钥

Install Release Key

Packages are cryptographically signed with the `release.asc` key. Add our release key to your system's list of trusted keys to avoid a security warning:

软件包都用 release.asc 密钥加密签名过，把我们的发布密钥加到您系统的可信密钥列表中可避免安全警告：

```
wget -q -O- 'https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc' | sudo apt-key add -
```

2.3.2 添加软件源

Add Release Packages

2.3.2.1 稳定版——Bobtail

Bobtail is the most recent major release of Ceph. These packages are recommended for anyone deploying Ceph in a production environment. Critical bug fixes are backported and point releases are made as necessary.

bobtail 是 ceph 最新的主要发布，这些软件包适合任何想在生产环境下部署 ceph 的人。重大缺陷的修复都移植过来了，次版本会适时推出。

Add our package repository to your system's list of APT sources. See the bobtail Debian repository for a complete list of distributions supported.

把我们的软件库加到 APT 源列表，所支持发行版的完整列表见 the bobtail Debian repository。

```
echo deb http://ceph.com/debian-bobtail/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
```

For the European users there is also a mirror in the Netherlands at http://eu.ceph.com/

欧洲用户可以访问位于荷兰的镜像 http://eu.ceph.com/。

```
echo deb http://eu.ceph.com/debian-bobtail/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
```

2.3.2.2 稳定版——Argonaut

Argonaut is the previous major release of Ceph. These packages are recommended for those who have already deployed argonaut in production and are not yet ready to upgrade.

Argonaut 是 ceph 的前一个主要发布，这些包适用于那些已经在生产环境部署了 argonaut 又没准备好升级的。

Add our package repository to your system's list of APT sources. See the argonaut Debian repository for a complete list of distributions supported.

把我们的软件库加到 APT 源列表，所支持发行版的完整列表见 the argonaut Debian repository。

```
echo deb http://ceph.com/debian-argonaut/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
```

For the European users there is also a mirror in the Netherlands at http://eu.ceph.com/

欧洲用户可以访问位于荷兰的镜像 http://eu.ceph.com/。

```
echo deb http://eu.ceph.com/debian-argonaut/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
```

2.3.2.3 开发版软件包

Development Release Packages

Our development process generates a new release of Ceph every 3-4 weeks. These packages are faster-moving than the stable releases, as they get new features integrated quickly, while still undergoing several weeks of QA prior to release.

在我们开发过程中，每 3-4 周会发布一次，这些包比稳定版变迁得快，因为它们经常集成进新功能，然后经历几周时间的质检考验才能正式发布。

Add our package repository to your system's list of APT sources. See the testing Debian repository for a complete list of distributions supported.

把我们的软件库加到 APT 源列表，所支持发行版的完整列表见 the testing Debian repository。

```
echo deb http://ceph.com/debian-testing/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
```

For the European users there is also a mirror in the Netherlands at http://eu.ceph.com/

欧洲用户可以访问位于荷兰的镜像 http://eu.ceph.com/。

```
echo deb http://eu.ceph.com/debian-testing/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
```

2.3.2.4 开发测试软件包

Development Testing Packages

We automatically build Debian and Ubuntu packages for current development branches in the Ceph source code repository. These packages are intended for developers and QA only.

我们自动构建当前开发分支的 Debian 和 Ubuntu 包，这些包是给开发者和 QA 用的。

Packages are cryptographically signed with the `autobuild.asc` key. Add our autobuild key to your system's list of trusted keys to avoid a security warning:

这些软件包用 autobuild.asc 加密签名过，把我们的 autobuild 密钥加到您系统的可信密钥列表中可避免安全警告：

```
wget -q -O- 'https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/autobuild.asc' | sudo apt-key add
-
```

Add our package repository to your system's list of APT sources, but replace `{BRANCH}` with the branch you'd like to use (e.g., chef-3, wip-hack, master, stable). See the gitbuilder page for a complete list of distributions we build.

把我们的软件库加到 APT 源列表，并把{branch}替换为你想用的分支（如：chef-3、wip-hack、master、stable）。我们构建的的完整发行版列表见 the gitbuilder page。

```
echo deb http://gitbuilder.ceph.com/ceph-deb-$(lsb_release -sc)-x86_64-basic/ref/{BRANCH} $
(lsb_release -sc) main | sudo tee /etc/apt/sources.list.d/ceph.list
```

2.3.3 安装软件包

Installing Packages

Once you have added either release or development packages to APT, you should update APT's database and install Ceph:

把正式发布或开发包加到 APT 后，要更新 APT 数据库，再安装 ceph：

```
sudo apt-get update && sudo apt-get install ceph
```

2.4 RPM 包的安装

Installing RPM Packages

You may install stable release packages (for stable deployments), development release packages (for the latest features), or development testing packages (for development and QA only). Do not add multiple package sources at the same time.

你可以选择安装稳定发行包（用于稳定部署）、开发发行包（用于发掘最新功能）、或开发测试包（仅用于开发和质检）。不要同时添加多个软件源。

2.4.1 安装发布密钥

Install Release Key

Packages are cryptographically signed with the `release.asc` key. Add our release key to your system's list of trusted keys to avoid a security warning:

软件包都用 release.asc 密钥加密签名过，把我们的发布密钥加到您系统的可信密钥列表中可避免安全警告：

```
sudo rpm --import 'https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc'
```

2.4.2 添加发布包

Add Release Packages

2.4.2.1 稳定版——Bobtail

Bobtail is the most recent major release of Ceph. These packages are recommended for anyone deploying Ceph in a production environment. Critical bug fixes are backported and point releases are made as necessary.

bobtail 是 ceph 最新的主要发布，这些软件包适合任何想在生产环境下部署 ceph 的人。重大缺陷的修复都移植过来了，次版本会适时推出。

Packages are currently built for the RHEL/CentOS6 (`el6`), Fedora 17 (`f17`), OpenSUSE 12 (`opensuse12`), and SLES (`sles11`) platforms. The repository package installs the repository details on your local system for use with `yum` or `up2date`.

现在我们给 RHEL/CentOS6(el6)、Fedora 17(f17)、OpenSUSE 12 (`opensuse12`)、和 SLES (`sles11`)构建包，软件库包会给你安装软件仓库，具体到系统可能用于 yum 或 up2date。

Replace the``{DISTRO}`` below with the distro codename:

用发行版代码名替换下面命令中的{DISTRO}：

```
su -c 'rpm -Uvh http://ceph.com/rpm-bobtail/{DISTRO}/x86_64/ceph-release-1-0.el6.noarch.rpm'
```

For example, for CentOS 6 or other RHEL6 derivatives (`el6`):

例如，拿 CentOS 6 或其他 RHEL6 衍生物(e16)来说：

```
su -c 'rpm -Uvh http://ceph.com/rpm-bobtail/el6/x86_64/ceph-release-1-0.el6.noarch.rpm'
```

You can download the RPMs directly from:

你也能直接从下列地址下载 RPM：

```
http://ceph.com/rpm-bobtail
```

2.4.2.2 开发版包

Development Release Packages

Our development process generates a new release of Ceph every 3-4 weeks. These packages are faster-moving than the stable releases. Development packages have new features integrated quickly, while still undergoing several weeks of QA prior to release.

在我们开发过程中，每 3-4 周会发布一次，这些包比稳定版变迁得快，因为它们经常集成进新功能，然后经历几周时间的质检考验才能正式发布。

Packages are cryptographically signed with the `release.asc` key. Add our release key to your system's list of trusted keys to avoid a security warning:

软件包用 release.asc 加密签名过，把我们的发布密钥放入可信密钥列表中可避免安全警告：

```
sudo rpm --import 'https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/autobuild.asc'
```

Packages are currently built for the CentOS-6 and Fedora 17 platforms. The repository package installs the repository details on your local system for use with `yum` or `up2date`.

当前我们给 CentOS-6 和 Fedora 17 平台构建软件包，软件库包会安装软件仓库，具体到系统可能用于 yum 或 up2date。

For CentOS-6:

对 CentOS 6 来说：

```
su -c 'rpm -Uvh http://ceph.com/rpms/el6/x86_64/ceph-release-1-0.el6.noarch.rpm'
```

For Fedora 17:

对 Fedora 17 来说：

```
su -c 'rpm -Uvh http://ceph.com/rpms/fc17/x86_64/ceph-release-1-0.fc17.noarch.rpm'
```

You can download the RPMs directly from:

你可以从下列地址直接下载 RPM：

```
http://ceph.com/rpm-testing
```

2.4.3 安装软件包

Installing Packages

Once you have added either release or development packages to `yum`, you can install Ceph:

把发布版或开发版源加入 yum 后，可以这样安装 ceph：

```
sudo yum install ceph
```

2.5 升级 ceph

Upgrading Ceph

You can upgrade daemons in your Ceph cluster one-by-one while the cluster is online and in service! The upgrade process is relatively simple:

你可以在集群在线且提供服务时，逐个升级集群中的守护进程！升级过程相对简单：

1. Login to a host and upgrade the Ceph package.
   登录主机并升级 ceph 包。
2. Restart the daemon.
   重启守护进程。
3. Ensure your cluster is healthy.

   确认集群是健康的。

Important: Once you upgrade a daemon, you cannot downgrade it.
重要：一旦升级了守护进程，就不能再降级。

Certain types of daemons depend upon others. For example, metadata servers and RADOS gateways depend upon Ceph monitors and OSDs. We recommend upgrading daemons in this order:

特定类型的守护进程依赖于其它的，例如元数据服务器和 RADOS 网关依赖于 ceph 监视器和 OSD，所以我们推荐按下列顺序升级：

1. Monitors (or OSDs)
2. OSDs (or Monitors)
3. Metadata Servers
4. RADOS Gateway

As a general rule, we recommend upgrading all the daemons of a specific type (e.g., all `ceph-osd` daemons, all `ceph-mon` daemons, etc.) to ensure that they are all on the same release. We also recommend that you upgrade all the daemons in your cluster before you try to excercise new functionality in a release.

作为一般规则，我们推荐一次性升级同一类型的所有守护进程（如所有 ceph-osd 守护进程、所有 ceph-mon 等等）以保证它们都处于同一版本。同时建议您升级完集群中的所有守护进程后再练习那个版本的新功能。

The following sections describe the upgrade process.

下面几段描述了升级过程。

> Important: Each release of Ceph may have some additional steps. Refer to release-specific sections for details BEFORE you begin upgrading daemons.
> 重要：每次版本升级都可能有例外步骤，**升级前**请参考与此版本相关的部分。

2.5.1 升级 OSD

Upgrading an OSD

To upgrade an OSD peform the following steps:

要升级 OSD，执行下列步骤：

1. Upgrade the OSD package:

   升级 OSD 包：

   ```
   ssh {osd-host}
   sudo apt-get update && sudo apt-get install ceph
   ```

2. Restart the OSD, where `N` is the OSD number:

   重启 OSD，N 是 OSD 号：

   ```
   service ceph restart osd.N
   ```

3. Ensure the upgraded OSD has rejoined the cluster:

   确认升级后的 OSD 加入了集群：

   ```
   ceph osd stat
   ```

Once you have successfully upgraded an OSD, you may upgrade another OSD until you have completed the upgrade cycle for all of your OSDs.

成功升级一个 OSD 后，再继续其它的，直到完成所有 OSD。

2.5.2 升级监视器

Upgrading a Monitor

To upgrade a monitor, perform the following steps:

执行下列步骤升级监视器：

1. Upgrade the ceph package:

   升级 ceph 包：

   ```
   ssh {mon-host}
   sudo apt-get update && sudo apt-get install ceph
   ```

2. Restart the monitor:

   重启监视器：

```
service ceph restart mon.{name}
```

3. Ensure the monitor has rejoined the quorum.

   确认监视器重入法定人数。

```
ceph mon stat
```

Once you have successfully upgraded a monitor, you may upgrade another monitor until you have completed the upgrade cycle for all of your monitors.

成功升级一个监视器后，再继续其它的，直到完成所有监视器。

2.5.3 升级元数据服务器

Upgrading a Metadata Server

To upgrade an MDS, perform the following steps:

执行下列步骤升级 MDS：

1. Upgrade the ceph package:

   升级 ceph 包：

```
ssh {mds-host}
sudo apt-get update && sudo apt-get install ceph
```

2. Restart the metadata server:

   重启元数据服务器：

```
service ceph restart mds.{name}
```

3. Ensure the metadata server is up and running:

   确认元数据服务器启动且运行着：

```
ceph mds stat
```

Once you have successfully upgraded a metadata, you may upgrade another metadata server until you have completed the upgrade cycle for all of your metadata servers.

成功升级一个元数据服务器后，再继续其它的，直到完成所有元数据服务器。

2.5.4 升级客户端

Upgrading a Client

Once you have upgraded the packages and restarted daemons on your Ceph cluster, we recommend upgrading `ceph-common` and client libraries (`librbd1` and `librados2`) on your client nodes too.

完成 ceph 集群上软件包和守护进程的升级后，我们建议也升级一下客户端节点上的 ceph-common 和客户端库（librbd1 和 librados2）。

1. Upgrade the package:

   升级软件包：

```
ssh {client-host}
apt-get update && sudo apt-get install ceph-common librados2 librbd1 python-ceph
```

2. Ensure that you have the latest version:

   确认你升级到了最新版本：

```
ceph --version
```

2.5.5 从 Argonaut 升级到 Bobtail

Upgrading from Argonaut to Bobtail

When upgrading from Argonaut to Bobtail, you need to be aware of three things:

从 Argonaut 升级到 Bobtail 时，要注意三件事：

1. Authentication now defaults to **ON**, but used to default to off.
   现在默认开启了认证，但以前默认关闭。
2. Monitors use a new internal on-wire protocol
   监视器使用了新的内部 on-wire 协议（一个时间同步协议？）
3. RBD `format2` images require updgrading all OSDs before using it.

   使用 RBD format2 格式的映像前要先升级完所有 OSD。

See the following sections for details.

详情参见下列段落。

2.5.5.1 认证

Authentication

The Ceph Bobtail release enables authentication by default. Bobtail also has finer-grained authentication configuration settings. In previous versions of Ceph (i.e., actually v 0.55 and earlier), you could simply specify:

ceph 的 bobtail 版默认启用认证，可配置粒度也更细。在之前版本的 ceph 中（如 v0.55 及更早），你可以简单地指定：

```
auth supported = [cephx | none]
```

This option still works, but is deprecated. New releases support `cluster`, `service` and `client` authentication settings as follows:

这个选项仍然可用，但已过时。新版可分别支持 cluster、service 和 client 认证设置：

```
auth cluster required = [cephx | none]  # default cephx
auth service required = [cephx | none] # default cephx
auth client required = [cephx | none] # default cephx,none
```

Important: If your cluster does not currently have an `auth supported` line that enables authentication, you must explicitly turn it off in Bobtail using the settings below.:
重要：如果你现在的集群上没有用 auth supported 启用认证，在 bobtail 里你必须用下列配置显式地关闭认证：

```
auth cluster required = none
auth service required = none
```

This will disable authentication on the cluster, but still leave clients with the default configuration where they can talk to a cluster that does enable it, but do not require it.
这会在集群上关闭认证，但默认启用了认证的客户端仍然可以和集群通讯，不是必需的。

Important: If your cluster already has an `auth supported` option defined in the configuration file, no changes are necessary.
重要：如果你的集群已经配置了 auth supported，那就没必要改了。

See Ceph Authentication - Backward Compatibility for details.

详情参见 Ceph Authentication - Backward Compatibility。

2.5.5.2 监视器 on-wire 协议

Monitor On-wire Protocol

We recommend upgrading all monitors to Bobtail. A mixture of Bobtail and Argonaut monitors will not be able to use the new on-wire protocol, as the protocol requires all monitors to be Bobtail or greater. Upgrading only a majority of the nodes (e.g., two out of three) may expose the cluster to a situation where a single

additional failure may compromise availability (because the non-upgraded daemon cannot participate in the new protocol). We recommend not waiting for an extended period of time between `ceph-mon` upgrades.

我们建议把所有监视器都升级到 bobtail 版， Bobtail 和 Argonaut 版的监视器混合将不能使用新的 on-wire 协议，因为此协议要求所有监视器是 bobtail 或更高版本。只升级大部分节点（如 3 个中的 2 个）会使集群陷入一种窘境——再失败一个节点就可能危及可用性，因为未升级的守护进程不能参与新协议。我们建议在 ceph-mon 升级时不要间隔太长时间。

2.5.5.3 RBD 映像

RBD Images

The Bobtail release supports `format 2` images! However, you should not create or use `format 2` RBD images until after all `ceph-osd` daemons have been upgraded. Note that `format 1` is still the default. You can use the new `ceph osd ls` and `ceph tell osd.N version` commands to doublecheck your cluster. `ceph osd ls` will give a list of all OSD IDs that are part of the cluster, and you can use that to write a simple shell loop to display all the OSD version strings:

bobtail 版支持 format 2 格式的映像！然而你在升级完所有 ceph-osd 守护进程前不应该创建或使用 format 2 格式的 RBD 映像。注意，默认仍是 format 1。你可以用 ceph osd ls 和 ceph tell osd.N version 命令检查集群版本，ceph osd ls 会列出集群里所有 OSD 的 ID，你可以把它们代进一个简单的 shell 循环来显示所有 OSD 版本号：

```
for i in $(ceph osd ls); do
    ceph tell osd.${i} version
done
```

2.6 构建前提

Build Prerequisites

> Tip: Check this section to see if there are specific prerequisites for your Linux/Unix distribution.
> 提示：检查下这段，看看你的 Linux/Unix 发行版是否具备了必要前提。

Before you can build Ceph source code, you need to install several libraries and tools. Ceph provides `autoconf` and `automake` scripts to get you started quickly. Ceph build scripts depend on the following:

在构建 ceph 源码前，你得先安装几个库和工具。ceph 用 autoconf 和 automake 脚本来简化构建，这些脚本依赖下列：

- `autotools-dev`
- `autoconf`
- `automake`
- `cdbs`
- `gcc`
- `g++`
- `git`
- `libboost-dev`
- `libedit-dev`
- `libssl-dev`
- `libtool`
- `libfcgi`
- `libfcgi-dev`
- `libfuse-dev`
- `linux-kernel-headers`
- `libcrypto++-dev`
- `libcrypto++`
- `libexpat1-dev`
- `pkg-config`
- `libcurl4-gnutls-dev`

On Ubuntu, execute `sudo apt-get install` for each dependency that isn't installed on your host.

在 Ubuntu 上，执行 sudo apt-get install 安装缺失依赖。

```
sudo apt-get install autotools-dev autoconf automake cdbs gcc g++ git libboost-dev libedit-dev
libssl-dev libtool libfcgi libfcgi-dev libfuse-dev linux-kernel-headers libcrypto++-dev libcrypto++
libexpat1-dev
```

On Debian/Squeeze, execute `aptitude install` for each dependency that isn't installed on your host.

在 Debian/Squeeze 上，执行 aptitude install 安装缺失依赖。

```
aptitude install autotools-dev autoconf automake cdbs gcc g++ git libboost-dev libedit-dev libssl-
dev libtool libfcgi libfcgi-dev libfuse-dev linux-kernel-headers libcrypto++-dev libcrypto++
libexpat1-dev pkg-config libcurl4-gnutls-dev
```

On Debian/Wheezy, you may also need:

在 Debian/Wheezy 上，你可能还需要：

```
keyutils-dev libaio and libboost-thread-dev
```

Note: Some distributions that support Google's memory profiler tool may use a different package name (e.g., `libgoogle-perftools4`).
注意：有些支持 Google 内存分析器工具的发行版可能用了不同的软件包名字（如 libgoogle-perftools4）。

2.6.1 Ubuntu

- `uuid-dev`
- `libkeyutils-dev`
- `libgoogle-perftools-dev`
- `libatomic-ops-dev`
- `libaio-dev`
- `libgdata-common`
- `libgdata13`
- `libsnappy-dev`
- `libleveldb-dev`

Execute `sudo apt-get install` for each dependency that isn't installed on your host.

执行 sudo apt-get install 安装缺失依赖。

```
sudo apt-get install uuid-dev libkeyutils-dev libgoogle-perftools-dev libatomic-ops-dev libaio-dev
libgdata-common libgdata13 libsnappy-dev libleveldb-dev
```

2.6.2 Debian

Alternatively, you may also install:

另外可能还要安装：

```
aptitude install fakeroot dpkg-dev
aptitude install debhelper cdbs libexpat1-dev libatomic-ops-dev
```

2.6.3 openSUSE 11.2（及更高版）

- `boost-devel`
- `gcc-c++`
- `libedit-devel`
- `libopenssl-devel`
- `fuse-devel` (optional)

Execute `zypper install` for each dependency that isn't installed on your host.

执行 zypper install 安装缺失依赖。

```
zypper install boost-devel gcc-c++ libedit-devel libopenssl-devel fuse-devel
```

2.7 下载 ceph 发布压缩包

Downloading a Ceph Release Tarball

As Ceph development progresses, the Ceph team releases new versions of the source code. You may download source code tarballs for Ceph releases here:

随着开发的推进，ceph 团队会经常发布新版源码，你可以从下列地点下载压缩包：

Ceph Release Tarballs

Ceph Release Tarballs (EU mirror)

2.8 安装 git 版

Set Up Git

To clone the Ceph git repository, you must have `git` installed on your local host.

要克隆 ceph 的 git 仓库，你得先在本机安装 git。

2.8.1 安装 git

Install Git

To install `git`, execute:

执行下列命令安装 git：

```
sudo apt-get install git
```

You must also have a `github` account. If you do not have a `github` account, go to github.com and register. Follow the directions for setting up git at Set Up Git.

你还得有一个 github 帐户，如果你没有，去 github.com 注册一个，然后继续随着安装 git 版配置 git。

2.8.2 生成 SSH 密钥对

Generate SSH Keys

If you intend to commit code to Ceph or to clone using SSH (`git@github.com:ceph/ceph.git`), you must generate SSH keys for github.

如果你要向 ceph 贡献代码，或者通过 SSH 克隆（git@github.com:ceph/ceph.git），就必须生成用于 github 的 SSH 密钥对。

Tip: If you only intend to clone the repository, you may use `git clone --recursive https://github.com/ceph/ceph.git` without generating SSH keys.
提示：如果你只想克隆仓库，可以用 git clone --recursive https://github.com/ceph/ceph.git，而不必生成 SSH 密钥对。

To generate SSH keys for `github`, execute:

要为 github 生成 SSH 密钥对，执行：

```
ssh-keygen
```

Get the key to add to your `github` account (the following example assumes you used the default file path):

把生成的密钥加进你的 github 帐户（下例假设你用的是默认路径）：

```
cat .ssh/id_rsa.pub
```

Copy the public key.

拷贝公钥。

2.8.3 添加密钥

Add the Key

Go to your your `github` account, click on “Account Settings” (i.e., the ‘tools’ icon); then, click “SSH Keys” on the left side navbar.

进入你的 github 帐户，点击“Account Settings”；然后点击左边导航条的“SSH Keys”。

Click “Add SSH key” in the “SSH Keys” list, enter a name for the key, paste the key you generated, and press the “Add key” button.

点击“SSH Keys”列表中的“Add SSH key”、给密钥输入名字、粘贴你生成的密钥、并按下“Add key”按钮。

2.9 克隆 ceph 源码仓库

Cloning the Ceph Source Code Repository

To clone the source, you must install Git. See Set Up Git for details.

要克隆源码，必须安装 git，详情参见安装 git 版。

2.9.1 克隆源码

Clone the Source

To clone the Ceph source code repository, execute:

要克隆 ceph 源码仓库，执行：

```
git clone --recursive https://github.com/ceph/ceph.git
```

Once `git clone` executes, you should have a full copy of the Ceph repository.

git clone 执行完毕后，你就得到了 ceph 仓库的完整副本。

Tip: Make sure you maintain the latest copies of the submodules included in the repository. Running `git status` will tell you if the submodules are out of date.
提示：要保证你维护着仓库中子模块的最新副本，用 git status 查看子模块是否过期。

```
cd ceph
git status
```

If your submodules are out of date, run:

如果你的子模块过期了，运行：

```
git submodule update
```

2.9.2 选择分支

Choose a Branch

Once you clone the source code and submodules, your Ceph repository will be on the `master` branch by default, which is the unstable development branch. You may choose other branches too.

克隆源码和子模块后，ceph 仓库默认会位于 master 分支，它是不稳定开发分支。你也可以选择其它分支：

- `master`: The unstable development branch.
- `stable`: The bugfix branch.
- `next`: The release candidate branch.

```
git checkout master
```

2.10 构建 ceph

Building Ceph

Ceph provides `automake` and `configure` scripts to streamline the build process. To build Ceph, navigate to your cloned Ceph repository and execute the following:

ceph 提供了 automake 和 configure 脚本来简化构建过程。要构建 ceph，进入你克隆的仓库、执行下列命令：

```
cd ceph
./autogen.sh
./configure
make
```

Hyperthreading

You can use `make -j` to execute multiple jobs depending upon your system. For example, `make -j4` for a dual core processor may build faster.

超线程

根据你的系统可以用 make -j 同时运行多个任务，例如 make -j4 在双核 CPU 上会编译得更快。

To install Ceph locally, you may also use:

要把 ceph 安装到本地，你可以用：

```
sudo make install
```

If you install Ceph locally, `make` will place the executables in `usr/local/bin`. You may add the Ceph configuration file to the `usr/local/bin` directory to run an evaluation environment of Ceph from a single directory.

如果你在本地安装 ceph，make 会把可执行文件放到 usr/local/bin 下。你可以把 ceph 配置文件放到 usr/local/bin 目录下，以搭建 ceph 评估环境。

2.11 安装 oprofile

Installing Oprofile

The easiest way to profile Ceph's CPU consumption is to use the oprofile system-wide profiler.

分析 ceph CPU 消耗情况的最简方法就是用系统级的 oprofile。

2.11.1 安装

Installation

If you are using a Debian/Ubuntu distribution, you can install `oprofile` by executing the following:

如果你在用 Debian/Ubuntu 发行版，可以用下列命令安装 oprofile：

```
sudo apt-get install oprofile oprofile-gui
```

2.11.2 编译适合分析的 ceph

Compiling Ceph for Profiling

To compile Ceph for profiling, first clean everything.

要编译适合分析的 ceph，首先清理干净。

```
make distclean
```

Then, export the following settings so that you can see callgraph output.

然后，执行下列命令才能看到输出的调用图。

```
export CFLAGS="-fno=omit-frame-pointer -O2 -g"
```

Finally, compile Ceph.

最后，编译 ceph。

```
./autogen.sh
./configure
make
```

You can use `make -j` to execute multiple jobs depending upon your system. For example:

你可以用 make -j 同时执行多个任务，例如：

```
make -j4
```

2.11.3 ceph 配置

Ceph Configuration

Ensure that you disable `lockdep`. Consider setting logging to levels appropriate for a production cluster. See Ceph Logging and Debugging for details.

确保禁用了 lockdep。生产集群应设置合理的日志级别，参见日志和调试配置参考。

See the CPU Profiling section of the RADOS Operations documentation for details on using Oprofile.

关于 Oprofile 的使用，请参考 RADOS Operations 文档的 CPU profiling 部分。

2.12 构建 ceph 包

Build Ceph Packages

To build packages, you must clone the Ceph repository. You can create installation packages from the latest code using `dpkg-buildpackage` for Debian/Ubuntu or `rpmbuild` for the RPM Package Manager.

要构建包，比必须克隆 ceph 仓库。

Tip: When building on a multi-core CPU, use the `-j` and the number of cores * 2. For example, use `-j4` for a dual-core processor to accelerate the build.
提示：在多核 CPU 上构建时，使用-j 加 CPU 核心数乘 2 可更快地编译，比如双核 CPU 上用-j4。

2.12.1 高级包管理器（APT）

Advanced Package Tool (APT)

To create `.deb` packages for Debian/Ubuntu, ensure that you have cloned the Ceph repository, installed the build prerequisites and installed `debhelper`:

要为 Debian/Ubuntu 创建.deb 包，先克隆 ceph 仓库、完成构建前提并安装 debhelper：

```
sudo apt-get install debhelper
```

Once you have installed debhelper, you can build the packages:

安装 debhelper 后，就可以构建包了：

```
sudo dpkg-buildpackage
```

For multi-processor CPUs use the `-j` option to accelerate the build.

有多 CPU 时用-j 选项加快构建。

2.12.2 RPM 包管理器

RPM Package Manager

To create `.rpm` packages, ensure that you have cloned the Ceph repository, installed the build prerequisites and installed `rpm-build` and `rpmdevtools`:

要创建.rpm 包，先克隆 ceph 仓库、完成构建前提并安装 rpm-build 和 rpmdevtools：

```
yum install rpm-build rpmdevtools
```

Once you have installed the tools, setup an RPM compilation environment:

安装完工具后，设置 RPM 编译环境：

```
rpmdev-setuptree
```

Fetch the source tarball for the RPM compilation environment:

把源码包下载到 RPM 编译环境：

```
wget -P ~/rpmbuild/SOURCES/ http://ceph.com/download/ceph-<version>.tar.gz
```

Or from the EU mirror:

或者从欧洲镜像：

```
wget -P ~/rpmbuild/SOURCES/ http://eu.ceph.com/download/ceph-<version>.tar.gz
```

Build the RPM packages:

构建 RPM 包：

```
rpmbuild -tb ~/rpmbuild/SOURCES/ceph-<version>.tar.gz
```

For multi-processor CPUs use the `-j` option to accelerate the build.

有多 CPU 时用-j 选项加快构建。

2.13 贡献代码

Contributing Source Code

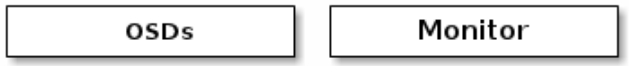
If you are making source contributions to the Ceph project, you must be added to the Ceph project on github.

如果你打算为 ceph 贡献代码，必须加入 github 上的 ceph 项目。

3 RADOS 对象存储

Ceph's RADOS Object Store is the foundation for all Ceph clusters. When you use object store clients such as the CephFS filesystem, the RESTful Gateway or Ceph block devices, Ceph reads data from and writes data to the object store. Ceph's RADOS Object Stores consist of two types of daemons: Object Storage Daemons (OSDs) store data as objects on storage nodes; and Monitors maintain a master copy of the cluster map. A Ceph cluster may contain thousands of storage nodes. A minimal system will have at least two OSDs for data replication.

ceph 的 RADOS 对象存储是所有 ceph 集群的基础，当你使用类似 CephFS 文件系统的对象存储客户端（RESTful 网关或 ceph 块设备）时，ceph 从对象存储里读取和写入数据。ceph 的 RADOS 对象存储包括两类守护进程：对象存储守护进程（OSD）把存储节点上的数据存储为对象；监视器维护着集群运行图的主拷贝。一个 ceph 集群可以包含数千个存储节点，最简的系统至少需要二个 OSD 才能做到数据复制。



CONFIG AND DEPLOY

配置和部署

Once you have installed Ceph packages, you must configure. There are a a few required settings, but most configuration settings have default values. Following the initial configuration, you must deploy Ceph. Deployment consists of creating and initializing data directories, keys, etc.

安装 ceph 包后必须配置，大多数选项都有默认值，但有几个地方必须更改才能使用；最初的步骤有初始化数据目录、密钥等。

3.1 配置

Ceph can run with a cluster containing thousands of Object Storage Devices (OSDs). A minimal system will have at least two OSDs for data replication. To configure OSD clusters, you must provide settings in the configuration file. Ceph provides default values for many settings, which you can override in the configuration file. Additionally, you can make runtime modification to the configuration using command-line utilities.
ceph 集群可以包含数千个存储节点，最小系统至少需要二个 OSD 才能做到数据复制。要配置 OSD 集群，你得把配置写入配置文件，ceph 对很多选项提供了默认值，你可以在配置文件里覆盖它们；另外，你可以使用命令行工具修改运行时配置。

When Ceph starts, it activates three daemons:
ceph 启动时要激活三种守护进程：

ceph-osd (mandatory)
ceph-mon (mandatory)
ceph-mds (mandatory for cephfs only)
Each process, daemon or utility loads the host's configuration file. A process may have information about more than one daemon instance (i.e., multiple contexts). A daemon or utility only has information about a single daemon instance (a single context).

Note Ceph can run on a single host for evaluation purposes.
注意：评估时 Ceph 可运行在单机上。

3.1.1 硬盘和文件系统推荐

HARD DISK PREP
准备硬盘

Ceph aims for data safety, which means that when the application receives notice that data was written to the disk, that data was actually written to the disk. For old kernels (<2.6.33), disable the write cache if the journal is on a raw disk. Newer kernels should work fine.
ceph 注重数据安全，就是说它收到数据已经写入硬盘的通知时，数据确实已写入硬盘。使用较老的内核（版本小于 2.6.33）时，如果文件系统日志在原始硬盘上，就要禁用写缓存；较新的内核没问题。

Use hdparm to disable write caching on the hard disk:
用 hdparm 禁用硬盘的写缓存功能。

```
sudo hdparm -W 0 /dev/hda 0
```

In production environments, we recommend running OSDs with an operating system disk, and a separate disk(s) for data. If you run data and an operating system on a single disk, create a separate partition for your data before configuring your OSD cluster.
在生产环境，建议您在系统盘运行 OSD，在另外的硬盘里放数据。如果必须把数据和系统放在一个硬盘里，最好给数据分配一个单独的分区。

FILE SYSTEMS
文件系统

Ceph OSDs rely heavily upon the stability and performance of the underlying file system.
ceph 的 OSD 依赖于底层文件系统的稳定性和性能。

Note We currently recommend XFS for production deployments. We recommend btrfs for testing, development, and any non-critical deployments. We believe that btrfs has the correct feature set and roadmap to serve Ceph in the long-term, but XFS and ext4 provide the necessary stability for today's deployments. btrfs development is proceeding rapidly: users should be comfortable installing the latest released upstream kernels and be able to track development activity for critical bug fixes.

注意：当前，我们推荐部署生产系统时使用 xfs 文件系统，建议用 btrfs 做测试、开发和其他不太要紧的部署。我们相信，长期来看 btrfs 适合 ceph 的功能需求和发展方向，但是 xfs 和 ext4 能提供当前部署所必需的稳定性。btrfs 开发在迅速推进，用户应该有能力经常更新到最新内核发布，而且能跟踪严重 bug 的修正进度。

Ceph OSDs depend on the Extended Attributes (XATTRs) of the underlying file system for various forms of internal object state and metadata. The underlying file system must provide sufficient capacity for XATTRs. btrfs does not bound the total xattr metadata stored with a file. XFS has a relatively large limit (64 KB) that most deployments won't encounter, but the ext4 is too small to be usable. To use these file systems, you should add the following like to the [osd] section of your ceph.conf file.:
ceph 的 OSD 有赖于底层文件系统的扩展属性（XATTR）存储各种内部对象状态和元数据。底层文件系统必须给 XATTR 提供足够容量，btrfs 没有限制随文件存储的 xattr 元数据；xfs 的限制相对大(64KB)，多数部署都不会有瓶颈；ext4 的则太小而不可用。要用这些文件系统，你得把下面这行写入 ceph.conf 的[osd]段里。

```
filestore xattr use omap = true
```

FS BACKGROUND INFO
文件系统背景知识

The XFS and btrfs file systems provide numerous advantages in highly scaled data storage environments when compared to ext3 and ext4. Both XFS and btrfs are journaling file systems, which means that they are more robust when recovering from crashes, power outages, etc. These filesystems journal all of the changes they will make before performing writes.
xfs 和 btrfs 相比较 ext3/4 而言，在高伸缩性数据存储方面有几个优势，xfs 和 btrfs 都是日志文件系统，这使得在崩溃、断电后恢复时更健壮，因为文件系统在写入数据前会先记录所有变更。

XFS was developed for Silicon Graphics, and is a mature and stable filesystem. By contrast, btrfs is a relatively new file system that aims to address the long-standing wishes of system administrators working with large scale data storage environments. btrfs has some unique features and advantages compared to other Linux filesystems.
xfs 由 Silicon Graphics 开发，是一个成熟、稳定的文件系统。相反，btrfs 是相对年轻的文件系统，它致力于实现系统管理员梦寐以求的大规模数据存储基础，和其他 Linux 文件系统相比它有独一无二的功能和优势。

btrfs is a copy-on-write filesystem. It supports file creation timestamps and checksums that verify metadata integrity, so it can detect bad copies of data and fix them with the good copies. The copy-on-write capability means that btrfs can support snapshots that are writable. btrfs supports transparent compression and other features.
btrfs 是写时复制（copy-on-write, cow）文件系统，它支持文件创建时间戳和校验和（可校验元数据完整性）功能，所以它能探测到数据坏副本，并且用好副本修复。写时复制功能是说 btrfs 支持可写文件系统快照。btrfs 也支持透明压缩和其他功能。

btrfs also incorporates multi-device management into the file system, which enables you to support heterogeneous disk storage infrastructure, data allocation policies. The community also aims to provide fsck, deduplication, and data encryption support in the future. This compelling list of features makes btrfs the ideal choice for Ceph clusters.
btrfs 也集成了多设备管理功能，据此可以在底层支持异质硬盘存储，和数据分配策略。未来开发社区还会提供 fsck、拆分、数据加密功能，这些诱人的功能正是 ceph 集群的理想选择。

3.1.2 ceph 的配置

When you start the Ceph service, the initialization process activates a series of daemons that run in the background. The hosts in a typical Ceph cluster run at least one of four daemons:
你启动 ceph 服务的时候，初始化进程会把一系列守护进程放到后台运行。典型的 ceph 集群中至少要运行下面四种守护进程中的一种。

Object Storage Device (ceph-osd)
Monitor (ceph-mon)
Metadata Server (ceph-mds)

Ceph Gateway (radosgw)
For your convenience, each daemon has a series of default values (i.e., many are set by ceph/src/common/config_opts.h). You may override these settings with a Ceph configuration file.
对象存储设备(ceph-osd)
监视器(ceph-mon)
元数据服务器(ceph-mds)
ceph 网关(radosgw)
为方便起见，每个守护进程都有一系列默认值（参见 ceph/src/common/config_opts.h），你可以用 ceph 配置文件覆盖这些设置。

3.1.2.1 配置文件 ceph.conf

When you start a Ceph cluster, each daemon looks for a ceph.conf file that provides its configuration settings. For manual deployments, you need to create a ceph.conf file to configure your cluster. For third party tools that create configuration files for you (e.g., Chef), you may use the information contained herein as a reference. The ceph.conf file defines:
启动 ceph 集群时，每个守护进程都从 ceph.conf 里查找它自己的配置。手动配置时，你需要创建 ceph.conf 来配置集群，使用第三方工具（如 chef）创建配置文件时可以参考下面的信息。ceph.conf 文件定义了：

Cluster membership
Host names
Host addresses
Paths to keyrings
Paths to journals
Paths to data
Other runtime options
集群成员
主机名
主机 IP 地址
密钥路径
日志路径
数据路径
其它运行时选项

The default ceph.conf locations in sequential order include:
默认的 ceph.conf 位置相继排列如下：

- $CEPH_CONF (i.e., the path following the $CEPH_CONF environment variable)
- -c path/path (i.e., the -c command line argument)
- /etc/ceph/ceph.conf
- ~/.ceph/config
- ./ceph.conf (i.e., in the current working directory)

The ceph.conf file uses an ini style syntax. You can add comments to the ceph.conf file by preceding comments with a semi-colon (;) or a pound sign (#). For example:
ceph 文件使用 ini 风格的语法，以分号(;)和井号(#)开始的行是注释，如下：

```
# <--A number (#) sign precedes a comment.
; A comment may be anything.
# Comments always follow a semi-colon (;) or a pound (#) on each line.
# The end of the line terminates a comment.
# We recommend that you provide comments in your configuration file(s).
```

3.1.2.2 ceph.conf 设置

The ceph.conf file can configure all daemons in a cluster, or all daemons of a particular type. To configure a series of daemons, the settings must be included under the processes that will receive the configuration as follows:

ceph.conf 可配置集群里的所有守护进程，或者一种特定类型的守护进程

[global]
全局选项

Description: Settings under [global] affect all daemons in a Ceph cluster.
描述：[global]下的配置影响 ceph 集群里的所有守护进程。
Example: auth supported = cephx

[osd]
Description: Settings under [osd] affect all ceph-osd daemons in the cluster.
描述：[osd]下的配置影响集群里的所有 ceph-osd 进程。
Example: osd journal size = 1000

[mon]
Description: Settings under [mon] affect all ceph-mon daemons in the cluster.
描述：[mon]下的配置影响集群里的所有 ceph-mon 进程。
Example: mon addr = 10.0.0.101:6789

[mds]
Description: Settings under [mds] affect all ceph-mds daemons in the cluster.
描述：[mds]下的配置影响集群里的所有 ceph-mds 进程。
Example: host = myserver01

Global settings affect all instances of all daemon in the cluster. Use the [global] setting for values that are common for all daemons in the cluster. You can override each [global] setting by:
全局设置影响集群内所有守护进程的例程，但可以用下面的设置覆盖[global]设置：

Changing the setting in a particular process type (e.g., [osd], [mon], [mds]).
在[osd]、[mon]、[mds]下设置某一类的进程。
Changing the setting in a particular process (e.g., [osd.1])
修改特定进程的设置，如[osd.1]。

Overriding a global setting affects all child processes, except those that you specifically override.
覆盖全局设置会影响所有子进程，明确剔除的例外。

A typical global setting involves activating authentication. For example:
典型的全局设置包括激活认证，例如：

```
[global]
    # Enable authentication between hosts within the cluster.
    # 在集群内的主机间认证
        auth supported = cephx
```

You can specify settings that apply to a particular type of daemon. When you specify settings under [osd], [mon] or [mds] without specifying a particular instance, the setting will apply to all OSDs, monitors or metadata daemons respectively.
你可以在[osd]、[mon]、[mds]下设置特定类型的守护进程，而无须指定特定例程，这些设置会分别影响所有 OSD、监视器、元数据进程。

You may specify settings for particular instances of a daemon. You may specify an instance by entering its type, delimited by a period (.) and by the instance ID. The instance ID for an OSD is always numeric, but it may be alphanumeric for monitors and metadata servers.
你也可以设置一个守护进程的特定例程，一个例程由类型和它的例程编号（ID）确定，OSD 的例程 ID 只能是数字，监视器和元数据服务器的 ID 可包含字母和数字。

```
[osd.1]
        # settings affect osd.1 only.
```

```
[mon.a]
        # settings affect mon.a only.

[mds.b]
        # settings affect mds.b only.
```

3.1.2.3 元变量

METAVARIABLES

Metavariables simplify cluster configuration dramatically. When a metavariable is set in a configuration value, Ceph expands the metavariable into a concrete value. Metavariables are very powerful when used within the [global], [osd], [mon] or [mds] sections of your configuration file. Ceph metavariables are similar to Bash shell expansion.
元变量大大简化了集群配置，ceph 会把配置的元变量展开为具体值；元变量功能很强大，可以用在[global]、[osd]、[mon]、[mds]段里，类似于 bash 的 shell 扩展。

Ceph supports the following metavariables:
ceph 支持下面的元变量：

$cluster
Description: Expands to the cluster name. Useful when running multiple clusters on the same hardware.
描述：展开为集群名字，在同一套硬件上运行多个集群时有用。
Example: /etc/ceph/$cluster.keyring
Default: ceph

$type
Description: Expands to one of mds, osd, or mon, depending on the type of the current daemon.
描述：展开为 mds、osd、mon 中的一个，有赖于当前守护进程的类型。
Example: /var/lib/ceph/$type

$id
Description: Expands to the daemon identifier. For osd.0, this would be 0; for mds.a, it would be a.
描述：展开为守护进程标识；对 osd.0 来说标识为 0，mds.a 的标识是 a。
Example: /var/lib/ceph/$type/$cluster-$id

$host
Description: Expands to the host name of the current daemon.
描述：展开为当前守护进程的主机名。

$name
Description: Expands to $type.$id.
Example: /var/run/ceph/$cluster-$name.asok

3.1.2.4 共有设置

COMMON SETTINGS
The Hardware Recommendations section provides some hardware guidelines for configuring the cluster. It is possible for a single host to run multiple daemons. For example, a single host with multiple disks or RAIDs may run one ceph-osd for each disk or RAID. Additionally, a host may run both a ceph-mon and an ceph-osd daemon on the same host. Ideally, you will have a host for a particular type of process. For example, one host may run ceph-osd daemons, another host may run a ceph-mds daemon, and other hosts may run ceph-mon daemons.
硬件推荐段提供了一些配置集群的硬件指导。一台机器可以运行多个进程，例如一台机器有多个硬盘或 RAID，可以为每个硬盘和 RAID 配置一个守护进程。另外，在同一台主机上也可以同时运行 ceph-mon 和 ceph-osd，理想情况下一台主机应该只运行一类进程，例如：一台主机运行着 ceph-osd 进程，另一台主机运行着 ceph-mds 进程，ceph-mon 进程又在另外一台主机上。

Each host has a name identified by the host setting. Monitors also specify a network address and port (i.e., domain name or IP address) identified by the addr setting. A basic configuration file will typically specify only minimal settings for each instance of a daemon. For example:
每个主机都有一个标识名称（系统配置），监视器可用 addr 选项指定网络地址和端口（如域名或 IP 地址），基本配置可以只指定最小配置。例如：

```
[mon.a]
        host = hostName
        mon addr = 150.140.130.120:6789
[osd.0]
        host = hostName
```

Important The host setting is the short name of the host (i.e., not an fqdn). It is NOT and IP address either. Enter hostname -s on the command line to retrieve the name of the host. Also, this setting is ONLY for mkcephfs and manual deployment. It MUST NOT be used with chef or ceph-deploy.
重要：主机名设置是主机的短名字，不是正式域名 FQDN，也不是 IP 地址；在执行 hostname -s 就可以得到短名字。此设置只在 mkcephfs 和手动部署时用到，用 chef 部署 ceph 时不必要。

3.1.2.5 网络

NETWORKS
Monitors listen on port 6789 by default, while metadata servers and OSDs listen on the first available port beginning at 6800. Ensure that you open port 6789 on hosts that run a monitor daemon, and open one port beginning at port 6800 for each OSD or metadata server that runs on the host. Ports are host-specific, so you don't need to open any more ports open than the number of daemons running on that host, other than potentially a few spares. You may consider opening a few additional ports in case a daemon fails and restarts without letting go of the port such that the restarted daemon binds to a new port. If you set up separate public and cluster networks, you may need to make entries for each network. 监视器默认监听 6789 端口，元数据服务器和 OSD 监听从 6800 开始的第一个可用端口；要确保监听 6789 端口的是监视器进程，每个 OSD 或元数据服务器进程监听了从 6800 开始的端口。每个主机使用的端口数量都是确定的，所以你没必要多开端口。但有些时候进程失败后重启却没释放旧端口，就会绑定到了新端口，所以要多留些端口作备用。如果你的部署区分了集群网和公网，你也许需要给每个网络写相应配置。

For example:
iptables -A INPUT -m multiport -p tcp -s {ip-address}/{netmask} --dports 6789,6800:6810 -j ACCEPT
In our hardware recommendations section, we recommend having at least two NIC cards, because Ceph can support two networks: a public (front-side) network, and a cluster (back-side) network. Ceph functions just fine with a public network only. You only need to specify the public and cluster network settings if you use both public and cluster networks.
例如：

```
iptables -A INPUT -m multiport -p tcp -s {ip-address}/{netmask} --dports 6789,6800:6810 -j ACCEPT
```

在硬件推荐一段，我们推荐至少 2 个网卡，因为 ceph 支持两个网络：位于前端的公网、位于后端的集群网。只有一个公网也可以，只有在公网、集群网都使用的情况下才需要分别配置。

There are several reasons to consider operating two separate networks. First, OSDs handle data replication for the clients. When OSDs replicate data more than once, the network load between OSDs easily dwarfs the network load between clients and the Ceph cluster. This can introduce latency and create a performance problem. Second, while most people are generally civil, a very tiny segment of the population likes to engage in what's known as a Denial of Service (DoS) attack. When traffic between OSDs gets disrupted, placement groups may no longer reflect an active + clean state, which may prevent users from reading and writing data. A great way to defeat this type of attack is to maintain a completely separate cluster network that doesn't connect directly to the internet.
建议使用两个分开的网络，首先，OSD 给客户端做数据复制时，如果副本数大于 1，客户端和 ceph 集群间的网络负载会很容易增大，这会导致延时增大和性能下降；其次，大多数人都是文明的，只有一小撮人喜欢所谓的拒绝服务攻击（DoS），当攻击 OSD 间的流量中断时，归置组就再也不能进入 active+clean 状态了，它也会阻止用户读写数据。杜绝此类攻击的最佳方法就是彻底分开集群网和公网。

To configure the networks, add the following options to the [global] section of your ceph.conf file.

要配置这样的网络，把下面的选项加到 ceph.conf 里的[global]段下。

```
[global]
        public network {public-network-ip-address/netmask}
        cluster network {enter cluster-network-ip-address/netmask}
```

To configure Ceph hosts to use the networks, you should set the following options in the daemon instance sections of your ceph.conf file.
要让 ceph 主机使用分开的网络，把下面的选项设置在守护进程例程下面。

```
[osd.0]
        public addr {host-public-ip-address}
        cluster addr {host-cluster-ip-address}
```

3.1.2.6 监视器

MONITORS
Ceph production clusters typically deploy with a minimum 3 monitors to ensure high availability should a monitor instance crash. An odd number of monitors (3) ensures that the Paxos algorithm can determine which version of the cluster map is the most recent from a quorum of monitors.
典型的 ceph 生产集群至少部署 3 个监视器来确保高可靠性，它允许一个监视器例程崩溃。奇数个监视器确保 PAXOS 算法能确定一批监视器里哪个版本的集群运行图是最新的。

Note You may deploy Ceph with a single monitor, but if the instance fails, the lack of a monitor may interrupt data service availability.
注：一个 ceph 集群可以只有一个监视器，但是如果它失败了，因没有监视器数据服务就会中断。

Ceph monitors typically listen on port 6789. For example:
ceph 监视器默认监听 6789 端口，例如：

```
[mon.a]
        host = hostName
        mon addr = 150.140.130.120:6789
```

By default, Ceph expects that you will store a monitor's data under the following path:
默认情况下，ceph 会在下面的路径存储监视器数据：

```
/var/lib/ceph/mon/$cluster-$id
```

You must create the corresponding directory yourself. With metavariables fully expressed and a cluster named "ceph", the foregoing directory would evaluate to:
你必须自己创建对应的目录，前述的元变量必须先展开，名为 ceph 的集群将展开为：

```
/var/lib/ceph/mon/ceph-a
```

You may override this path using the mon data setting. We don't recommend changing the default location. Create the default directory on your new monitor host.
你可以用 mon data 选项更改默认路径，但我们不推荐修改。用下面的命令在新监视器主机上创建默认目录：

```
ssh {new-mon-host}
sudo mkdir /var/lib/ceph/mon/ceph-{mon-letter}
```

3.1.2.7 OSDs

Ceph production clusters typically deploy OSDs where one host has one OSD daemon running a filestore on one data disk. A typical deployment specifies a journal size and whether the file store's extended attributes (XATTRs) use an object map (i.e., when running on the ext4 filesystem). For example:
典型的生产集群里，一个数据盘上运行一个 OSD 进程；典型部署要指定日志尺寸、文件存储的扩展属性（XATTR）是否使用对象图（如运行在 ext4 之上），例如：

```
[osd]
        osd journal size = 10000
        filestore xattr use omap = true   #enables the object map. Only if running ext4.
```

```
[osd.0]
        hostname = {hostname}
```

By default, Ceph expects that you will store an OSD's data with the following path:
默认 ceph 认为你把 OSD 数据存储在下面的路径下：

```
/var/lib/ceph/osd/$cluster-$id
```

You must create the corresponding directory yourself. With metavariables fully expressed and a cluster named "ceph", the foregoing directory would evaluate to:
你必须创建对应目录，名字为 ceph 的集群其元变量完全展开后，前述的目录将是：

```
/var/lib/ceph/osd/ceph-0
```

You may override this path using the osd data setting. We don't recommend changing the default location. Create the default directory on your new OSD host.
你可以用 osd data 选项更改默认值，但我们不建议修改。用下面的命令在新 OSD 主机上创建默认目录：

```
ssh {new-osd-host}
sudo mkdir /var/lib/ceph/osd/ceph-{osd-number}
```

The osd data path ideally leads to a mount point with a hard disk that is separate from the hard disk storing and running the operating system and daemons. If the OSD is for a disk other than the OS disk, prepare it for use with Ceph, and mount it to the directory you just created:
osd data 路径应该指向一个硬盘的挂载点，这个硬盘应该独立于操作系统和守护进程所在硬盘。按下列步骤准备好并挂载：

```
ssh {new-osd-host}
sudo mkfs -t {fstype} /dev/{disk}
sudo mount -o user_xattr /dev/{hdd} /var/lib/ceph/osd/ceph-{osd-number}
```

We recommend using the xfs file system or the btrfs file system when running command:mkfs.
我们推荐用 xfs 或 btrfs 文件系统，命令是 mkfs。

By default, Ceph expects that you will store an OSDs journal with the following path:
ceph 默认把 OSD 日志存储在下面的路径：

```
/var/lib/ceph/osd/$cluster-$id/journal
```

Without performance optimization, Ceph stores the journal on the same disk as the OSDs data. An OSD optimized for performance may use a separate disk to store journal data (e.g., a solid state drive delivers high performance journaling).
没有性能优化时，ceph 把日志和 OSD 数据存储相同的硬盘上；要优化 OSD 性能，可以把日志分离到单独的硬盘上（例如，固态硬盘能提供高日志性能）。

Ceph's default osd journal size is 0, so you will need to set this in your ceph.conf file. A journal size should find the product of the filestore min sync interval and the expected throughput, and multiple the product by two (2):
ceph 的 osd journal size 默认值是 0，所以你得在 ceph.conf 里设置，日志尺寸应该至少 2 倍于 filestore min sync interval 的值和预计吞吐量的乘积：

```
osd journal size = {2 * (expected throughput * filestore min sync interval)}
```

The expected throughput number should include the expected disk throughput (i.e., sustained data transfer rate), and network throughput. For example, a 7200 RPM disk will likely have approximately 100 MB/s. Taking the min() of the disk and network throughput should provide a reasonable expected throughput. Some users just start off with a 10GB journal size. For example:
预计吞吐量应该包括硬盘吞吐量（持续的数据传输速度）和网络吞吐量，例如，7200 转的硬盘速度大概是 100MB/s，取硬盘和网络吞吐量中较小的一个应该能提供合理的预计吞吐量。一些用户以 10GB 起步，例如：

```
osd journal size = 10000
```

3.1.2.8 日志、调试

LOGS / DEBUGGING
Ceph is still on the leading edge, so you may encounter situations that require modifying logging output and using Ceph's debugging. To activate Ceph's debugging output (i.e., dout()), you may add debug settings to your configuration. Ceph's logging levels operate on a scale of 1 to 20, where 1 is terse and 20 is verbose. Subsystems common to each daemon may be set under [global] in your configuration file. Subsystems for particular daemons are set under the daemon section in your configuration file (e.g., [mon], [osd], [mds]). For example:
ceph 仍在前沿，所以你可能碰到一些情况，需要修改日志和调试信息。为打开 ceph 的调试输出（例如，dout()），你可以在配置文件里添加调试选项。ceph 的日志级别在 1 到 20 之间，1 是简洁、20 是详细。对每个进程都相同的子系统可以在[global]下配置，对特定守护进程的子系统要配置在进程段下，如[mon]、[osd]、[mds]下。例如：

```
[global]
        debug ms = 1

[mon]
        debug mon = 20
        debug paxos = 20
        debug auth = 20

[osd]
        debug osd = 20
        debug filestore = 20
        debug journal = 20
        debug monc = 20

[mds]
        debug mds = 20
        debug mds balancer = 20
        debug mds log = 20
        debug mds migrator = 20
```

When your system is running well, choose appropriate logging levels and remove unnecessary debugging settings to ensure your cluster runs optimally. Logging debug output messages is relatively slow, and a waste of resources when operating your cluster.
你的系统运行良好的时候，应该选择合适的日志级别、关闭不必要的调试设置来确保集群运行在最佳状态。记录调试输出相对慢，且浪费资源。

Each subsystem has a logging level for its output logs, and for its logs in-memory. You may set different values for each of these subsystems by setting a log file level and a memory level for debug logging. For example:
每个子系统的日志都有它自己的输出级别、和内存存留级别。你可以给每个子系统分别设置不同的日志文件和内存日志级别，例如：

```
debug {subsystem} {log-level}/{memory-level}
#for example
debug mds log 1/20
```

Subsystem	Log Level	Memory Level
default	0	5
lockdep	0	5
context	0	5
crush	1	5
mds	1	5
mds balancer	1	5
mds locker	1	5
mds log	1	5
mds log expire	1	5
mds migrator	1	5
buffer	0	0
timer	0	5
filer	0	5
objecter	0	0

Subsystem	Log Level	Memory Level
rados	0	5
rbd	0	5
journaler	0	5
objectcacher	0	5
client	0	5
osd	0	5
optracker	0	5
objclass	0	5
filestore	1	5
journal	1	5
ms	0	5
mon	1	5
monc	0	5
paxos	0	5
tp	0	5
auth	1	5
finisher	1	5
heartbeatmap	1	5
perfcounter	1	5
rgw	1	5
hadoop	1	5
asok	1	5
throttle	1	5

3.1.2.9 ceph.conf 实例

EXAMPLE CEPH.CONF

```
[global]
    auth supported = cephx

[osd]
    osd journal size = 1000
    # uncomment the following line if you are mounting with ext4
    # filestore xattr use omap = true

[mon.a]
    host = myserver01
    mon addr = 10.0.0.101:6789

[mon.b]
    host = myserver02
    mon addr = 10.0.0.102:6789

[mon.c]
    host = myserver03
    mon addr = 10.0.0.103:6789

[osd.0]
    host = myserver01

[osd.1]
    host = myserver02

[osd.2]
    host = myserver03

[mds.a]
    host = myserver01
```

3.1.2.10 运行时更改

RUNTIME CHANGES

Ceph allows you to make changes to the configuration of an ceph-osd, ceph-mon, or ceph-mds daemon at runtime. This capability is quite useful for increasing/decreasing logging output, enabling/disabling debug settings, and even for runtime optimization. The following reflects runtime configuration usage:

ceph 可以在运行时更改 ceph-osd、ceph-mon、ceph-mds 守护进程的配置，这种功能在增加/降低日志输出、启用/禁用调试设置、甚至是运行时优化的时候非常有用，下面是运行时配置的用法：

```
ceph {daemon-type} tell {id or *} injectargs '--{name} {value} [--{name} {value}]'
```

Replace {daemon-type} with one of osd, mon or mds. You may apply the runtime setting to all daemons of a particular type with *, or specify a specific daemon's ID (i.e., its number or letter). For example, to increase debug logging for a ceph-osd daemon named osd.0, execute the following:
用 osd、mon、mds 中的一个替代{daemon-type}，你可以用星号(*)或具体进程 ID（其数字或字母）把运行时配置应用到一类进程的所有例程，例如增加名为 osd.0 的 ceph-osd 进程的调试级别的命令如下：

```
ceph osd tell 0 injectargs '--debug-osd 20 --debug-ms 1'
```

In your ceph.conf file, you may use spaces when specifying a setting name. When specifying a setting name on the command line, ensure that you use an underscore or hyphen (_ or -) between terms (e.g., debug osd becomes debug-osd).
在 ceph.conf 文件里配置时用空格分隔关键词，但在命令行使用的时候要用下划线或连字符(_或-)分隔，例如 debug osd 变成 debug-osd。

3.1.2.11 查看运行时配置

VIEWING A CONFIGURATION AT RUNTIME
If your Ceph cluster is running, and you would like to see the configuration settings from a running daemon, execute the following:
如果你的 ceph 集群在运行，而你想看一个在运行进程的配置，用下面的命令：

```
ceph --admin-daemon {/path/to/admin/socket} config show | less
```

The default path for the admin socket for each daemon is:
每个守护进程的管理套接字默认路径如下：

```
/var/run/ceph/$cluster-$name.asok
```

At real time, the metavariables will evaluate to the actual cluster name and daemon name. For example, if the cluster name is ceph (it is by default) and you want to retrieve the configuration for osd.0, use the following:
同时，元变量将展开为实际的集群名和进程名，例如如果集群名是 ceph（默认值），你可以用下面的命令检索 osd.0 的配置：

```
ceph --admin-daemon /var/run/ceph/ceph-osd.0.asok config show | less
```

3.1.3 通用配置选项参考

General config reference

auth supported
Deprecated since version 0.51.
Description: Indicates the type of authentication used. Currently cephx only. If not specified, it defaults to none.
Type: String
Required: No
Default: none
描述：指定认证类型，当前只有 cephx。如果没指定，默认为 none。

auth cluster required
New in version 0.51.
Description: Enables authentication for the cluster. Valid setting is cephx.
Type: String
Required: No
Default: none
描述：启用集群认证，有效设置为 cephx。

auth service required
New in version 0.51.
Description: Enables authentication for the service. Valid setting is cephx.

Type: String
Required: No
Default: none
描述：启用服务认证，有效设置为 cephx。

auth client required
New in version 0.51.
Description: Enables authentication for the client. Valid setting is cephx.
Type: String
Required: No
Default: none
描述：启用客户端认证，有效设置为 cephx。

keyring
Description: The path to the cluster's keyring file.
Type: String
Required: No
Default: /etc/ceph/$cluster.$name.keyring, /etc/ceph/$cluster.keyring, /etc/ceph/keyring, /etc/ceph/keyring.bin
描述：集群密钥环的路径。

fsid
Description: The filesystem ID. One per cluster.
Type: UUID
Required: No.
Default: N/A. Generated if not specified.
描述：文件系统 ID，每集群一个。

mon host
Description: A list of {hostname}:{port} entries that clients can use to connect to a Ceph monitor. If not set, Ceph searches [mon.*] sections.
Type: String
Required: No
Default: N/A
描述：一个{hostname}:{port}条目的列表，可以让客户端连接到 ceph 监视器，如果没设置 ceph 会查找[mon.*]段。

host
Description: The hostname. Use this setting for specific daemon instances (e.g., [osd.0]).
Type: String
Required: Yes, for daemon instances.
Default: localhost
Tip Do not use localhost. To get your host name, execute hostname -s on your command line and use the name of your host (to the first period, not the fully-qualified domain name).
描述：主机名。这个设置可以用于具体守护进程例程，例如[osd.0]。
提示：不要使用 localhost。在命令行执行 hostname -s 获取主机名，并且使用主机的实际名字。

public network
Description: The IP address and netmask of the public (front-side) network (e.g., 10.20.30.40/24). Set in [global].
Type: {ip-address}/{netmask}
Required: No
Default: N/A
描述：公网（前端）的 IP 地址和掩码，如 10.20.30.40/24，在[global]下设置。

public addr
Description: The IP address for the public (front-side) network. Set for each daemon.
Type: IP Address
Required: No

Default: N/A
描述：公网（前端）的 IP 地址，可以给每个进程设置。

cluster network
Description: The IP address and netmask of the cluster (back-side) network (e.g., 10.20.30.41/24). Set in [global].
Type: {ip-address}/{netmask}
Required: No
Default: N/A
描述：集群网（后端）的 IP 地址和掩码，如 10.20.30.41/24，在[global]里设置。

cluster addr
Description: The IP address for the cluster (back-side) network. Set for each daemon.
Type: Address
Required: No
Default: N/A
描述：集群网（后端）的 IP 地址，每个进程都要设置。

admin socket
Description: The socket for executing administrative commands irrespective of whether Ceph monitors have established a quorum.
Type: String
Required: No
Default: /var/run/ceph/$cluster-$name.asok
描述：执行管理命令的套接字，不管 ceph 监视器团体是否已经建立。

pid file
Description: Each running Ceph daemon has a running process identifier (PID) file.
Type: String
Required: No
Default: N/A. The default path is /var/run/$cluster/$name.pid. The PID file is generated upon start-up.
描述：每个运行的 ceph 进程都有一个 PID 文件。

chdir
Description: The directory Ceph daemons change to once they are up and running. Default / directory recommended.
Type: String
Required: No
Default: /
描述：ceph 进程一旦启动、运行就进入这个目录，默认推荐/。

max open files
Description: If set, when the Ceph service starts, Ceph sets the max open fds at the OS level (i.e., the max # of file descriptors). It helps prevents OSDs from running out of file descriptors.
Type: 64-bit Integer
Required: No
Default: 0
描述：如果设置了，ceph 服务启动的时候会设置操作系统级的最大打开文件描述符（如最大数量的文件描述符），这有助于防止耗尽文件描述符。

fatal signal handlers
Description: If set, we will install signal handlers for SEGV, ABRT, BUS, ILL, FPE, XCPU, XFSZ, SYS signals to generate a useful log message
Type: Boolean
Default: true
描述：如果设置了，将安装 SEGV、ABRT、BUS、ILL、FPE、XCPU、XFSZ、SYS 信号处理器，用于产生有用的日志信息。

3.1.4 监视器配置选项参考

mon data
Description: The monitor's data location.
Type: String
Default: /var/lib/ceph/mon/$cluster-$id
描述：监视器的数据位置。

mon initial members
Description: The IDs of initial monitors in a cluster during startup. If specified, Ceph requires an odd number of monitors to form an initial quorum.
Type: String
Default: None
描述：集群启动时最初的监视器 ID，如果指定了，ceph 需要奇数个监视器来确定最初的法人团。

mon sync fs threshold
Description: Synchronize with the filesystem when writing the specified number of objects. Set it to 0 to disable it.
Type: 32-bit Integer
Default: 5
描述：数量达到设定值时和文件系统同步，0 为禁用。

mon tick interval
Description: A monitor's tick interval in seconds.
Type: 32-bit Integer
Default: 5
描述：监视器的心跳间隔，单位为秒。

mon subscribe interval
Description: The refresh interval (in seconds) for subscriptions. The subscription mechanism enables obtaining the cluster maps and log informations.
Type: Double
Default: 300
描述：同步的刷新间隔（秒），同步机制允许获取集群运行图和日志信息。

mon osd auto mark in
Description: Ceph will mark any booting OSDs as in the cluster.
Type: Boolean
Default: false
描述：把任何在 booting 状态的 OSD 标记为 in（在对象存储集群内）。

mon osd auto mark auto out in
Description: Ceph will mark booting OSDs auto marked out of the cluster as in the cluster.
Type: Boolean
Default: true
描述：把在 booting 状态、且为 auto marked out（不在对象存储集群内）状态的 OSD 标记为 in。

mon osd auto mark new in
Description: Ceph will mark booting new OSDs as in the cluster.
Type: Boolean
Default: true
描述：把 booting 状态的新 OSD 标记为 in。
mon osd down out interval
Description: The number of seconds Ceph waits before marking an OSD down and out if it doesn't respond.
Type: 32-bit Integer
Default: 300
描述：在 OSD 停止响应多少秒后把它标记为 down。

mon osd min up ratio

Description: The minimum ratio of up OSDs before Ceph will mark OSDs down.
Type: Double
Default: .3
描述：在把 OSD 标记为 down 前，保持处于 up 状态的 OSD 最小比例

mon osd min in ratio
Description: The minimum ratio of in OSDs before Ceph will mark OSDs out.
Type: Double
Default: .3
描述：在把 OSD 标记为 out 前，保持处于 in 状态的 OSD 最小比例

mon lease
Description: Length (in seconds) of the lease on the monitor's versions.
Type: Float
Default: 5
描述：监视器版本租期（秒）

mon lease renew interval
Description: The interval (in seconds) for the Leader to renew the other monitor's leases.
Type: Float
Default: 3
描述：监视器 leader（头领）刷新其他监视器租期的间隔。

mon lease ack timeout
Description: Number of seconds the Leader will wait for the Peons to acknowledge the lease extension.
Type: Float
Default: 10.0
描述：leader 在等到 peons（随从）确认延长租期前等待的时间

mon clock drift allowed
Description: The clock drift in seconds allowed between monitors.
Type: Float
Default: .050
描述：监视器间允许的时钟漂移量

mon clock drift warn backoff
Description: Exponential backoff for clock drift warnings
Type: Float
Default: 5
描述：时钟偏移警告的退避指数

mon accept timeout
Description: Number of seconds the Leader will wait for the Peons to accept a Paxos update. It is also used during the Paxos recovery phase for similar purposes.
Type: Float
Default: 10.0
描述：leader 等待 peons 接受 PAXOS 更新的时间，出于同样的目的此值也用于 PAXOS 恢复阶段。

mon pg create interval
Description: Number of seconds between PG creation in the same OSD.
Type: Float
Default: 30.0
描述：在同一个 OSD 里创建 PG 的间隔秒数。

mon pg stuck threshold
Description: Number of seconds after which PGs can be considered as being stuck.
Type: 32-bit Integer
Default: 300
描述：多长时间无响应的 PG 才认为它卡住了。

mon osd full ratio
Description: The percentage of disk space used before an OSD is considered full.
Type: Float
Default: .95
描述：OSD 硬盘使用率达到多少就认为它满了。

mon osd nearfull ratio
Description: The percentage of disk space used before an OSD is considered nearfull.
Type: Float
Default: .85
描述：OSD 硬盘使用率达到多少就认为它快满了。

mon globalid prealloc
Description: The number of global IDs to pre-allocate for the cluster.
Type: 32-bit Integer
Default: 100
描述：为集群预分配的全局 ID 数量。

mon osd report timeout
Description: The grace period in seconds before declaring unresponsive OSDs down.
Type: 32-bit Integer
Default: 900
描述：宣布 OSD 失效前的宽容时间。

mon force standby active
Description: should mons force standby-replay mds to be active
Type: Boolean
Default: true
描述：监视器是否应该强制处于 standby-replay 的元数据服务器活跃。

mon min osdmap epochs
Description: Minimum number of OSD map epochs to keep at all times.
Type: 32-bit Integer
Default: 500
描述：一直保存的 OSD 图元素最小数量。

mon max pgmap epochs
Description: Maximum number of PG map epochs the monitor should keep.
Type: 32-bit Integer
Default: 500
描述：监视器应该一直保存的 PG 图元素最大数量。

mon max log epochs
Description: Maximum number of Log epochs the monitor should keep.
Type: 32-bit Integer
Default: 500
描述：监视器应该保留的最大日志数量。

mon max osd
Description: The maximum number of OSDs allowed in the cluster.
Type: 32-bit Integer
Default: 10000
描述：集群允许的最大 OSD 数量。

mon probe timeout
Description: Number of seconds the monitor will wait to find peers before bootstrapping.
Type: Double
Default: 2.0

描述：监视器自举无效，搜寻节点前等待的时间。

mon slurp timeout
Description: Number of seconds the monitor has to recover using slurp before the process is aborted and the monitor bootstraps.
Type: Double
Default: 10.0
描述：监视器进程终止后、自举前，要等待多长时间才开始发出显式修复通告。

mon slurp bytes
Description: Limits the slurp messages to the specified number of bytes.
Type: 32-bit Integer
Default: 256 * 1024
描述：显式修复消息尺寸限制。

mon client bytes
Description: The amount of client message data allowed in memory (in bytes).
Type: 64-bit Integer Unsigned
Default: 100ul << 20
描述：内存中允许存留的客户端消息数量。

mon daemon bytes
Description: The message memory cap for metadata server and OSD messages (in bytes).
Type: 64-bit Integer Unsigned
Default: 400ul << 20
描述：给元数据服务器和 OSD 的消息使用的内存空间。

mon max log entries per event
Description: The maximum number of log entries per event.
Type: Integer
Default: 4096
描述：每个事件允许的最大日志条数。

3.1.5 OSD 配置选项参考

osd config reference
osd uuid
Description: The universally unique identifier (UUID) for the OSD.
Type: UUID
Default: None
描述：OSD 的全局唯一标识符（UUID）。

osd data
Description: The path to the OSDs data. You must create the directory. You should mount a data disk at this mount point. We do not recommend changing the default.
Type: String
Default: /var/lib/ceph/osd/$cluster-$id
描述：OSD 数据存储位置，你得创建并把数据盘挂载到其下。我们不推荐更改默认值。

osd journal
Description: The path to the OSD's journal. This may be a path to a file or a block device (such as a partition of an SSD). If it is a file, you must create the directory to contain it.
Type: String
Default: /var/lib/ceph/osd/$cluster-$id/journal
描述：OSD 日志路径。可以是一个文件或块设备（SSD 的一个分区），如果是文件，要先创建相应目录。

osd journal size
Description: The size of the journal in megabytes. If this is 0, and the journal is a block device, the entire block device is used. Since v0.54, this is ignored if the journal is a block device, and the entire block device is

used.
Type: 32-bit Integer
Default: 1024
Recommended: Begin with 1GB. Should at least twice the product of the expected speed multiplied by filestore min sync interval.
描述：日志尺寸（MB）。如果是 0 且日志文件是块设备，它会使用整个块设备，从 v0.54 起，如果日志文件是块设备，这个选项会被忽略，且会使用整个块设备。

osd max write size
Description: The maximum size of a write in megabytes.
Type: 32-bit Integer
Default: 90
描述：一次写入的最大尺寸，MB。

osd client message size cap
Description: The largest client data message allowed in memory.
Type: 64-bit Integer Unsigned
Default: 500MB default. 500*1024L*1024L
描述：内存里允许的最大客户端数据消息。

osd stat refresh interval
Description: The status refresh interval in seconds.
Type: 64-bit Integer Unsigned
Default: .5
描述：状态刷新间隔。

osd pg bits
Description: Placement group bits per OSD.
Type: 32-bit Integer
Default: 6
描述：每个 OSD 的归置组位数。

osd pgp bits
Description: The number of bits per OSD for PGPs.
Type: 32-bit Integer
Default: 4
描述：每个 OSD 为 PGP 留的位数。

osd pg layout
Description: Placement group layout.
Type: 32-bit Integer
Default: 2
描述：归置组布局图。

osd pool default crush rule
Description: The default CRUSH rule to use when creating a pool.
Type: 32-bit Integer
Default: 0
描述：创建一个存储池的时候使用的 CRUSH 规则。

osd pool default size
Description: The default size of an OSD pool in gigabytes. The default value is the same as --size 2 with mkpool.
Type: 32-bit Integer
Default: 2
描述：一个 OSD 存储池的默认大小（GB），默认值等同于 mkpool 的--size 2 参数。

osd pool default pg num
Description: The default number of placement groups for a pool. The default value is the same as pg_num

with mkpool.
Type: 32-bit Integer
Default: 8
描述：一个存储池的默认归置组数量，默认值即是 mkpool 的 pg_num 参数。

osd pool default pgp num
Description: The default number of placement groups for placement for a pool. The default value is the same as pgp_num with mkpool. PG and PGP should be equal (for now).
Type: 32-bit Integer
Default: 8
描述：一个存储池里，为归置使用的归置组数量，默认值等同于 mkpool 的 pgp_num 参数。当前 PG 和 PGP 应该相同。

osd map dedup
Description: Enable removing duplicates in the OSD map.
Type: Boolean
Default: true
描述：允许删除 OSD 图里的副本。

osd map cache size
Description: The size of the OSD map cache in megabytes.
Type: 32-bit Integer
Default: 500
描述：OSD 图缓存尺寸，MB。

osd map cache bl size
Description: The size of the in-memory OSD map cache in OSD daemons.
Type: 32-bit Integer
Default: 50
描述：OSD 进程中，驻留内存的 OSD 图缓存尺寸。

osd map cache bl inc size
Description: The size of the in-memory OSD map cache incrementals in OSD daemons.
Type: 32-bit Integer
Default: 100
描述：OSD 进程中，驻留内存的 OSD 图缓存增量尺寸。

osd map message max
Description: The maximum map entries allowed per MOSDMap message.
Type: 32-bit Integer
Default: 100
描述：每个 MOSDMap 图消息允许的最大条目数量。

osd op threads
Description: The number of OSD operation threads. Set to 0 to disable it. Increasing the number may increase the request processing rate.
Type: 32-bit Integer
Default: 2
描述：OSD 线程数，0 为禁用。增大数量可以增加请求处理速度。

osd op thread timeout
Description: The OSD operation thread timeout in seconds.
Type: 32-bit Integer
Default: 30
描述：OSD 线程超时秒数。

osd disk threads
Description: The number of disk threads, which are used to perform background disk intensive OSD operations such as scrubbing and snap trimming.

Type: 32-bit Integer
Default: 1
描述：硬盘线程数，用于在后台执行 IO 密集操作，像数据洗刷和快照修复。

osd recovery threads
Description: The number of threads for recovering data.
Type: 32-bit Integer
Default: 1
描述：数据恢复时的线程数。

osd recover clone overlap
Description: Preserves clone overlap during recovery and data migration.
Type: Boolean
Default: false
描述：数据恢复和迁移时保留的克隆重叠率。

osd backfill scan min
Description: The scan interval in seconds for backfill operations.
Type: 32-bit Integer
Default: 64
描述：回填操作时扫描间隔。

osd backfill scan max
Description: The maximum scan interval in seconds for backfill operations.
Type: 32-bit Integer
Default: 512
描述：回填操作时最大扫描间隔。

osd backlog thread timeout
Description: The maximum time in seconds before timing out a backlog thread.
Type: 32-bit Integer
Default: 60*60*1
描述：积压线程最大死亡时值。

osd recovery thread timeout
Description: The maximum time in seconds before timing out a recovery thread.
Type: 32-bit Integer
Default: 30
描述：恢复线程最大死亡时值。

osd snap trim thread timeout
Description: The maximum time in seconds before timing out a snap trim thread.
Type: 32-bit Integer
Default: 60*60*1
描述：快照修复线程最大死亡时值。

osd scrub thread timeout
Description: The maximum time in seconds before timing out a scrub thread.
Type: 32-bit Integer
Default: 60
描述：洗刷线程最大死亡时值。

osd scrub finalize thread timeout
Description: The maximum time in seconds before timing out a scrub finalize thread.
Type: 32-bit Integer
Default: 60*10
描述：洗刷终结线程最大超时值。

osd remove thread timeout

Description: The maximum time in seconds before timing out a remove OSD thread.
Type: 32-bit Integer
Default: 60*60
描述：OSD 删除线程的最大死亡时值。

osd command thread timeout
Description: The maximum time in seconds before timing out a command thread.
Type: 32-bit Integer
Default: 10*60
描述：命令线程最大超时值。

osd heartbeat address
Description: An OSD's network address for heartbeats.
Type: Address
Default: The host address.
描述：OSD 用于心跳的地址。

osd heartbeat interval
Description: How often an OSD pings its peers (in seconds).
Type: 32-bit Integer
Default: 6
描述：OSD 探测其他节点的频率。

osd heartbeat grace
Description: The elapsed time when an OSD hasn't shown a heartbeat that the cluster considers it down.
Type: 32-bit Integer
Default: 20
描述：OSD 多长时间没发出心跳信号则被集群认为死亡了。

osd _mon_heartbeat interval
Description: How often the OSD pings a monitor if it has no OSD peers.
Type: 32-bit Integer
Default: 30
描述：一个 OSD 如果没有 OSD 节点了，它 ping 监视器的频率。

osd mon report interval max
Description: The maximum time in seconds for an OSD to report to a monitor before the monitor considers the OSD down.
Type: 32-bit Integer
Default: 120
描述：OSD 向监视器报告的最大时间间隔。

osd mon report interval min
Description: The number of minutes between reports that include pg stats, up thru, boot and failures.
Type: 32-bit Integer
Default: 5
描述：报告间隔时间（分钟），报告内容包括 pg 状态、up 状态、启动和失败状态。

osd mon ack timeout
Description: The number of seconds to wait for a monitor to acknowledge a request for statistics.
Type: 32-bit Integer
Default: 30
描述：监视器确认一个统计请求的等待超时。

osd min down reporters
Description: The minimum number of OSDs required to report a down OSD.
Type: 32-bit Integer
Default: 1
描述：确认一个 OSD 为 down 至少要收到多少 down 报告。

osd min down reports
Description: The minimum number of times an OSD must report that another is down.
Type: 32-bit Integer
Default: 3
描述：一个 OSD 死亡后，另一个 OSD 必须报告的死亡次数。

osd recovery delay start
Description: After peering completes, Ceph will delay for the specified number of seconds before starting to recover objects.
Type: Float
Default: 15
描述：对等关系建立完毕后，ceph 开始对象恢复前等待的时间（秒）。

osd recovery max active
Description: The number of active recovery requests per OSD at one time. More accelerates recovery, but places an increased load on the cluster.
Type: 32-bit Integer
Default: 5
描述：每个 OSD 一次处理的活跃恢复请求，增大此值能加速恢复，但增加了集群的负载。

osd recovery max chunk
Description: The maximum size of a recovered chunk of data to push.
Type: 64-bit Integer Unsigned
Default: 1 << 20
描述：一次推送的数据块的最大尺寸。

osd max scrubs
Description: The maximum number of scrub operations for an OSD.
Type: 32-bit Int
Default: 1
描述：一个 OSD 的最大（并行？）洗刷操作数。

osd scrub load threshold
Description: The maximum CPU load. Ceph will not scrub when the CPU load is higher than this number. Default is 50%.
Type: Float
Default: 0.5
描述：最大 CPU 负载。CPU 负载高于此阀值时 ceph 将停止洗刷。默认是 50%。

osd scrub min interval
Description: The maximum interval in seconds for scrubbing the OSD.
Type: Float
Default: 5 minutes. 300
描述：OSD 洗刷最小间隔。

osd scrub max interval
Description: The maximum interval in seconds for scrubbing the OSD.
Type: Float
Default: Once per day. 60*60*24
描述：OSD 洗刷最大间隔。

osd deep scrub interval
Description: The interval for “deep” scrubbing (fully reading all data)
Type: Float
Default: Once per week. 60*60*24*7
描述：深度洗刷间隔，会读取所有数据。

osd deep scrub stride

Description: Read size when doing a deep scrub
Type: 32-bit Int
Default: 512 KB. 524288
描述：深度洗刷时的读尺寸。

osd class dir
Description: The class path for RADOS class plug-ins.
Type: String
Default: $libdir/rados-classes
描述：RADOS 类插件的路径。

osd check for log corruption
Description: Check log files for corruption. Can be computationally expensive.
Type: Boolean
Default: false
描述：根据日志文件查找数据腐败，会耗费大量计算时间。

osd default notify timeout
Description: The OSD default notification timeout (in seconds).
Type: 32-bit Integer Unsigned
Default: 30
描述：OSD 默认通告超时。

osd min pg log entries
Description: The minimum number of placement group logs to maintain when trimming log files.
Type: 32-bit Int Unsigned
Default: 1000
描述：清理日志文件的时候保留的归置组日志量。

osd op complaint time
Description: An operation becomes complaint worthy after the specified number of seconds have elapsed.
Type: Float
Default: 30
描述：一个操作进行多久后开始抱怨。

osd command max records
Description: Limits the number of lost objects to return.
Type: 32-bit Integer
Default: 256
描述：限制返回的丢失对象数量。

osd auto upgrade tmap
Description: Uses tmap for omap on old objects.
Type: Boolean
Default: true
描述：在旧对象上给 omap 使用 tmap。

osd tmapput sets users tmap
Description: Uses tmap for debugging only.
Type: Boolean
Default: false
描述：只在调试时使用 tmap。

osd kill backfill at
Description: For debugging only.
Type: 32-bit Integer
Default: 0
描述：仅为调试。

3.1.6 文件存储配置参考

Filestore config reference
filestore debug omap check
Description: Debugging check on synchronization. Expensive. For debugging only.
Type: Boolean
Required: No
Default: 0
描述：同步检查。昂贵的调试选项。

3.1.6.1 扩展属性

EXTENDED ATTRIBUTES
Extended Attributes (XATTRs) are an imporant aspect in your configuration. Some file systems have limits on the number of bytes stored in XATTRS. Additionally, in some cases, the filesystem may not be as fast as an alternative method of storing XATTRs. The following settings may help improve performance by using a method of storing XATTRs that is extrinsic to the underlying filesystem.
扩展属性（XATTR）是配置里的重要方面。一些文件系统对 XATTR 大小有限制，而且在一些情况下文件系统存储 XATTR 的速度不如其他方法，下面的设置通过使用独立于文件系统的存储方法能帮你提升性能。

filestore xattr use omap
Description: Use object map for XATTRS. Set to true for ext4 file systems.
Type: Boolean
Required: No
Default: false
描述：为 XATTR 使用对象图，使用 ext4 文件系统的时候要设置。

filestore max inline xattr size
Description: The maximimum size of an XATTR stored in the filesystem (i.e., XFS, btrfs, ext4, etc.) per object. Should not be larger than the filesytem can handle.
Type: Unsigned 32-bit Integer
Required: No
Default: 512
描述：每个对象在文件系统里存储的 XATTR 最大尺寸，应该小于文件系统支持的尺寸。

filestore max inline xattrs
Description: The maximum number of XATTRs stored in the fileystem per object.
Type: 32-bit Integer
Required: No
Default: 2
描述：每个对象存储在文件系统里的 XATTR 数量。

3.1.6.2 同步间隔

SYNCHRONIZATION INTERVALS
Periodically, the filestore needs to quiesce writes and synchronize the filesystem, which creates a consistent commit point. It can then free journal entries up to the commit point. Synchronizing more frequently tends to reduce the time required perform synchronization, and reduces the amount of data that needs to remain in the journal. Less frequent synchronization allows the backing filesystem to coalesce small writes and metadata updates more optimally–potentially resulting in more efficient synchronization.
filestore 需要周期性地静默写入、同步文件系统，这创建了一个提交点，然后就能释放相应的日志条目了。较大的同步频率可减小执行同步的时间及保存在日志里的数据量；较小的频率使得后端的文件系统能优化归并较小的数据和元数据写入，因此可能使同步更有效。

filestore max sync interval
Description: The maximum interval in seconds for synchronizing the filestore.
Type: Double
Required: No
Default: 5
描述：同步 filestore 的最大间隔秒数。

filestore min sync interval
Description: The minimum interval in seconds for synchronizing the filestore.
Type: Double
Required: No
Default: .01
描述：同步 filestore 的最小间隔秒数。

3.1.6.3 同步器

FLUSHER
The filestore flusher forces data from large writes to be written out using sync file range before the sync in order to (hopefully) reduce the cost of the eventual sync. In practice, disabling 'filestore flusher' seems to improve performance in some cases.
文件存储回写器强制使用同步文件排列来写出大块数据，这样处理有望减小最终同步的代价。实践中，禁用"filestore flusher"有时候能提升性能。

filestore flusher
Description: Enables the filestore flusher.
Type: Boolean
Required: No
Default: false
描述：启用 filestore flusher 功能。

filestore flusher max fds
Description: Sets the maximum number of file descriptors for the flusher.
Type: Integer
Required: No
Default: 512
描述：设置回写器的最大文件描述符数量。

filestore sync flush
Description: Enables the synchronization flusher.
Type: Boolean
Required: No
Default: false
描述：启用同步回写器。

filestore fsync flushes journal data
Description: Flush journal data during filesystem synchronization.
Type: Boolean
Required: No
Default: false
描述：文件系统同步时也回写日志数据。

3.1.6.4 队列

QUEUE
The following settings provide limits on the size of filestore queue.
下面的配置能限制文件存储队列的尺寸。

filestore queue max ops
Description: Defines the maximum number of in progress operations the file store accepts before blocking on queuing new operations.
Type: Integer
Required: No. Minimal impact on performance.
Default: 500
描述：文件存储操作接受的最大并发数，超过此设置的请求会被拒绝。

filestore queue max bytes

Description: The maximum number of bytes for an operation.
Type: Integer
Required: No
Default: 100 << 20
描述：一个操作的最大字节数。

filestore queue committing max ops
Description: The maximum number of operations the filestore can commit.
Type: Integer
Required: No
Default: 500
描述：文件存储能提交的最大操作数。

filestore queue committing max bytes
Description: The maximum number of bytes the filestore can commit.
Type: Integer
Required: No
Default: 100 << 20
描述：文件存储器能提交的最大字节数。

3.1.6.5 超时选项

TIMEOUTS
filestore op threads
Description: The number of filesystem operation threads that execute in parallel.
Type: Integer
Required: No
Default: 2
描述：最大并行文件系统操作线程数。

filestore op thread timeout
Description: The timeout for a filesystem operation thread (in seconds).
Type: Integer
Required: No
Default: 60
描述：文件系统操作线程超时值，单位为秒。

filestore op thread suicide timeout
Description: The timeout for a commit operation before cancelling the commit (in seconds).
Type: Integer
Required: No
Default: 180
描述：提交操作超时值（秒），超时后会取消。

3.1.6.6 B-tree 文件系统

B-TREE FILESYSTEM
filestore btrfs snap
Description: Enable snapshots for a btrfs filestore.
Type: Boolean
Required: No. Only used for btrfs.
Default: true
描述：对 btrfs 文件存储器启用快照功能。

filestore btrfs clone range
Description: Enable cloning ranges for a btrfs filestore.
Type: Boolean
Required: No. Only used for btrfs.
Default: true
描述：允许 btrfs 文件存储克隆动作排队。

3.1.6.7 日志

JOURNAL
filestore journal parallel
Description:
Type: Boolean
Required: No
Default: false

filestore journal writeahead
Description:
Type: Boolean
Required: No
Default: false

filestore journal trailing
Description:
Type: Boolean
Required: No
Default: false

3.1.6.8 其他选项

MISC
filestore merge threshold
Description:
Type: Integer
Required: No
Default: 10

filestore split multiple
Description:
Type: Integer
Required: No
Default: 2

filestore update to
Description:
Type: Integer
Required: No
Default: 1000

filestore blackhole
Description: Drop any new transactions on the floor.
Type: Boolean
Required: No
Default: false
描述：丢弃任何讨论中的事务。

filestore dump file
Description: File onto which store transaction dumps?
Type: Boolean
Required: No
Default: false
描述：存储事务转储目的文件。

filestore kill at
Description: inject a failure at the n'th opportunity
Type: String
Required: No
Default: false

描述：在第 N 次机会后注入一个失效。

filestore fail eio
Description: Fail/Crash on eio.
Type: Boolean
Required: No
Default: true
描述：在 IO 错误的时候失败或崩溃。

3.1.7 日志配置参考

Journal config reference
Ceph OSDs use a journal for two reasons: speed and consistency.
ceph 的 OSD 使用日志的原因有二：速度和一致性。

Speed: The journal enables the OSD to commit small writes quickly. Ceph writes small, random i/o to the journal sequentially, which tends to speed up bursty workloads by allowing the backing filesystem more time to coalesce writes. The OSD journal, however, can lead to spiky performance with short spurts of high-speed writes followed by periods without any write progress as the filesystem catches up to the journal.
速度：日志允许 OSD 快速地提交小块数据的写入，ceph 把小片、随机 IO 依次写入日志，这样，后端文件系统就有机会归并写入动作，并最终提升并发承载力。因此，使用 OSD 日志能展现出优秀的瞬间写性能，实际上却没有任何写动作，因为文件系统把它们捕捉到了日志。

Consistency: Ceph OSDs requires a filesystem interface that guarantees atomic compound operations. Ceph OSDs write a description of the operation to the journal and apply the operation to the filesystem. This enables atomic updates to an object (for example, placement group metadata). Every few seconds–between filestore max sync interval and filestore min sync interval–the OSD stops writes and synchronizes the journal with the filesystem, allowing OSDs to trim operations from the journal and reuse the space. On failure, OSDs replay the journal starting after the last synchronization operation.
Ceph OSDs support the following journal settings:
一致性：ceph 的 OSD 需要一个能保证原子操作的文件系统接口。OSD 把一个操作的描述写入日志，然后把操作应用到文件系统，这需要原子更新一个对象（例如归置组元数据）。每隔一段 filestore max sync interval 和 filestore min sync interval 之间的时间，OSD 停止写入、把日志同步到文件系统，这样允许 OSD 修整日志里的操作并重用空间。失败后，OSD 从上次的同步点开始重放日志。OSD 支持下面的日志设置：

journal dio
Description: Enables direct i/o to the journal. Requires journal block align set to true.
Type: Boolean
Required: Yes when using aio.
Default: true
描述：启用径直 IO 到日志，需要 journal block align 设置为 true。

journal aio
Description: Enables using libaio for asynchronous writes to the journal. Requires journal dio set to true.
Type: Boolean
Required: No.
Default: false
描述：异步写入日志时用 libaio 库，需要 journal dio 设为 true。

journal block align
Description: Block aligns writes. Required for dio and aio.
Type: Boolean
Required: Yes when using dio and aio.
Default: true
描述：块对齐写，dio 和 aio 需要。

journal max write bytes
Description: The maximum number of bytes the journal will write at any one time.
Type: Integer

Required: No
Default: 10 << 20
描述：一次写入日志的最大尺寸。

journal max write entries
Description: The maximum number of entries the journal will write at any one time.
Type: Integer
Required: No
Default: 100
描述：一次写入日志的最大数量。

journal queue max ops
Description: The maximum number of operations allowed in the queue at any one time.
Type: Integer
Required: No
Default: 500
描述：队列里一次允许的最大操作数量。

journal queue max bytes
Description: The maximum number of bytes allowed in the queue at any one time.
Type: Integer
Required: No
Default: 10 << 20
描述：队列里一次允许的最大字节数。

journal align min size
Description: Align data payloads greater than the specified minimum.
Type: Integer
Required: No
Default: 64 << 10
描述：对齐大于指定最小值的数据有效载荷。

journal zero on create
Description: Causes the file store to overwrite the entire journal with 0‘s during mkfs.
Type: Boolean
Required: No
Default: false
描述：在创建文件系统期间用 0 填充整个日志。

3.1.8 日志和调试配置参考

Logging and debugging config reference
Logging and debugging settings are not required, but you may override default settings as needed. Ceph supports the following settings:
日志记录和调试不是必需选项，但需要时可以覆盖。ceph 支持下面的设置：

3.1.8.1 日志记录

LOGGING
log file
Desription: The location of the logging file for your cluster.
Type: String
Required: No
Default: /var/log/ceph/$cluster-$name.log
描述：集群日志文件的位置。

log max new
Description: The maximum number of new log files.
Type: Integer
Required: No

Default: 1000
描述：新日志文件的最大数量。

log max recent
Description: The maximum number of recent events to include in a log file.
Type: Integer
Required: No
Default: 1000000
描述：一个日志文件包含的最新事件的最大数量。

log to stderr
Description: Determines if logging messages should appear in stderr.
Type: Boolean
Required: No
Default: true
描述：设置日志消息是否输出到标准错误。

err to stderr
Description: Determines if error messages should appear in stderr.
Type: Boolean
Required: No
Default: true
描述：设置错误消息是否输出到标准错误。

log to syslog
Description: Determines if logging messages should appear in syslog.
Type: Boolean
Required: No
Default: false
描述：设置日志消息是否输出到 syslog。

err to syslog
Description: Determines if error messages should appear in syslog.
Type: Boolean
Required: No
Default: false
描述：设置错误消息是否输出到 syslog。

log flush on exit
Description: Determines if Ceph should flush the log files after exit.
Type: Boolean
Required: No
Default: true
描述：设置 ceph 退出后是否回写日志文件。

clog to monitors
Description: Determines if clog messages should be sent to monitors.
Type: Boolean
Required: No
Default: true
描述：设置是否把 clog 消息发送给监视器。

clog to syslog
Description: Determines if clog messages should be sent to syslog.
Type: Boolean
Required: No
Default: false
描述：设置是否把 clog 输出到 syslog。

mon cluster log to syslog
Description: Determines if the cluster log should be output to the syslog.
Type: Boolean
Required: No
Default: false
描述：设置集群日志是否输出到 syslog。

mon cluster log file
Description: The location of the cluster's log file.
Type: String
Required: No
Default: /var/log/ceph/$cluster.log
描述：集群日志位置。

3.1.8.2 OSD 调试选项

osd debug drop ping probability
Description: ?
Type: Double
Required: No
Default: 0

osd debug drop ping duration
Description: The duration ?
Type: Integer
Required: No
Default: 0

osd debug drop pg create probability
Description:
Type: Integer
Required: No
Default: 0

osd debug drop pg create duration
Description: ?
Type: Double
Required: No
Default: 1

osd preserve trimmed log
Description: ?
Type: Boolean
Required: No
Default: false

osd tmapput sets uses tmap
Description: For debug only. ???
Type: Boolean
Required: No
Default: false

osd min pg log entries
Description: The minimum number of log entries for placement groups.
Type: 32-bit Unsigned Integer
Required: No
Default: 1000
描述：归置组日志最小条数。

osd op log threshold
Description: How many op log messages to show up in one pass.

Type: Integer
Required: No
Default: 5
描述：一次发送多少操作日志消息。

3.1.8.3 FILESTORE 调试选项

filestore debug omap check
Description: Checks the omap. This is an expensive operation.
Type: Boolean
Required: No
Default: 0
描述：检查 omap，这是昂贵的操作。

3.1.8.4 MDS 调试选项

mds debug scatterstat
Description: ?
Type: Boolean
Required: No
Default: false

mds debug frag
Description:
Type: Boolean
Required: No
Default: false

mds debug auth pins
Description: ?
Type: Boolean
Required: No
Default: false

mds debug subtrees
Description: ?
Type: Boolean
Required: No
Default: false

3.1.8.5 RADOS 网关

RADOS GATEWAY
rgw log nonexistent bucket
Description: Should we log a non-existent buckets?
Type: Boolean
Required: No
Default: false
描述：记录不存在的桶?

rgw log object name
Description: Should an object's name be logged. // man date to see codes (a subset are supported)
Type: String
Required: No
Default: %Y-%m-%d-%H-%i-%n
描述：是否记录对象名称。

rgw log object name utc
Description: Object log name contains UTC?
Type: Boolean
Required: No

Default: false
描述：对象日志名称包含 UTC?

rgw enable ops log
Description: Enables logging of every RGW operation.
Type: Boolean
Required: No
Default: true
描述：允许记录 RGW 的每一个操作。

rgw enable usage log
Description: Enable logging of RGW's bandwidth usage.
Type: Boolean
Required: No
Default: true
描述：允许记录 RGW 的带宽使用。

rgw usage log flush threshold
Description: Threshold to flush pending log data.
Type: Integer
Required: No
Default: 1024
描述：回写未决的日志数据阀值。

rgw usage log tick interval
Description: Flush pending log data every s seconds.
Type: Integer
Required: No
Default: 30
描述：未决日志回写间隔。

rgw intent log object name
Description:
Type: String
Required: No
Default: %Y-%m-%d-%i-%n

rgw intent log object name utc
Description: Include a UTC timestamp in the intent log object name.
Type: Boolean
Required: No
Default: false
描述：日志对象名字里包含 UTC 时间戳。

rgw cluster root pool
Description: RADOS pool to store radosgw metadata for this instance
Type: String
Required: No
Default: .rgw.root
描述：给这个例程扯出 radosgw 元数据的 RADOS 存储池。

rgw gc max objs
Description: Number of objects to collect garbage collection data
Type: 32-bit Integer
Default: 32
描述：用于存储垃圾回收数据的对象数量。

rgw gc obj min wait
Description: Minimum time to wait before object's removal and its processing by the garbage collector
Type: 32-bit Integer

Default: 2 hours. 2*60*60
描述：对象删除，且被垃圾回收器处理前等待的最小时间。

rgw gc processor max time
Description: Max time for a single garbage collection process run
Type: 32-bit Integer
Default: 1 hour. 60*60
描述：单个垃圾回收进程运行的最大时间。

rgw gc processor max period
Description: Max time between the beginning of two consecutive garbage collection processes run
Type: 32-bit Integer
Default: 1 hour. 60*60
描述：两次连续的垃圾回收进程开始运行时间的最大间隔。

3.1.9 消息传递

Messaging

ms tcp nodelay
Description:
Type: Boolean
Required: No
Default: true

ms initial backoff
Description:
Type: Double
Required: No
Default: .2

ms max backoff
Description:
Type: Double
Required: No
Default: 15.0

ms nocrc
Description:
Type: Boolean
Required: No
Default: false

ms die on bad msg
Description:
Type: Boolean
Required: No
Default: false

ms dispatch throttle bytes
Description:
Type: 64-bit Unsigned Integer
Required: No
Default: 100 << 20

ms bind ipv6
Description:
Type: Boolean
Required: No
Default: false

ms rwthread stack bytes
Description:
Type: 64-bit Unsigned Integer
Required: No
Default: 1024 << 10

ms tcp read timeout
Description:
Type: 64-bit Unsigned Integer
Required: No
Default: 900

ms inject socket failures
Description:
Type: 64-bit Unsigned Integer
Required: No
Default: 0

3.2 CEPH 的部署

Ceph deployment
You can deploy Chef using many different deployment systems including Chef, Juju, Puppet, and Crowbar. If you are just experimenting, Ceph provides some minimal deployment tools that rely only on SSH and DNS to deploy Ceph. You need to set up the SSH and DNS settings manually.
你可以用很多其它部署系统（如 Chef、Juju、Puppet、Crowbar）安装 chef，如果你只想测试一下，ceph 提供了最小化安装工具，它只依赖 SSH 和 DNS。你得手动设置 SSH 和 DNS。

CEPH DEPLOYMENT SCRIPTS
We provide light-weight deployment scripts to help you evaluate Ceph. For professional deployment, you should consider professional deployment systems such as Juju, Puppet, Chef or Crowbar.
ceph 部署脚本
我们提供了轻量级部署脚本供您评估 ceph，专业的部署应该考虑用专业的部署系统，像 Chef、Juju、Puppet、Crowbar。

3.2.1 用 mkcephfs 配置

Deploying with mkcephfs
To deploy a test or development cluster, you can use the mkcephfs tool. We do not recommend using this tool for production environments.
要配置一个测试或开发集群，你可以用 mkcephfs 工具，生产环境我们不推荐。

3.2.1.1 允许以root 身份登录集群主机

ENABLE LOGIN TO CLUSTER HOSTS AS ROOT
To deploy with mkcephfs, you will need to be able to login as root on each host without a password. For each host, perform the following:
用 mkcephfs 配置，需要以 root 身份登录且不需要密码，在每台主机上执行下面的命令：

```
sudo passwd root
```

Enter a password for the root user.
给 root 设置一个密码。

On the admin host, generate an ssh key without specifying a passphrase and use the default locations.
在管理主机上生成一个 ssh 密钥，不要指定通行码，密钥文件使用默认位置。

```
ssh-keygen
Generating public/private key pair.
Enter file in which to save the key (/ceph-admin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Your identification has been saved in /ceph-admin/.ssh/id_rsa.
Your public key has been saved in /ceph-admin/.ssh/id_rsa.pub.
```

You may use RSA or DSA keys. Once you generate your keys, copy them to each OSD host. For example:
你可以用 RSA 或 DSA 密钥，生成后把公钥拷贝到每个 OSD 主机。例如：

```
ssh-copy-id root@myserver01
ssh-copy-id root@myserver02
```

Modify your ~/.ssh/config file to login as root, as follows:
更改你的~/.ssh/config 文件来登录为 root，如下：

```
Host myserver01
        Hostname myserver01.fully-qualified-domain.com
        User root
Host myserver02
        Hostname myserver02.fully-qualified-domain.com
        User root
```

3.2.1.2 把配置文件拷贝到所有主机

COPY CONFIGURATION FILE TO ALL HOSTS
Ceph's mkcephfs deployment script does not copy the configuration file you created from the Administration host to the OSD Cluster hosts. Copy the configuration file you created (i.e., mycluster.conf in the example below) from the Administration host to etc/ceph/ceph.conf on each OSD Cluster host if you are using mkcephfs to deploy Ceph.
ceph 的 mkcephfs 脚本不会把管理主机上创建的配置文件拷贝到 OSD 主机，所以你得手动拷贝，例如：

```
ssh myserver01 sudo tee /etc/ceph/ceph.conf < /etc/ceph/ceph.conf
ssh myserver02 sudo tee /etc/ceph/ceph.conf < /etc/ceph/ceph.conf
ssh myserver03 sudo tee /etc/ceph/ceph.conf < /etc/ceph/ceph.conf
```

3.2.1.3 创建默认目录

CREATE THE DEFAULT DIRECTORIES
The mkcephfs deployment script does not create the default server directories. Create server directories for each instance of a Ceph daemon (if you haven't done so already). The host variables in the ceph.conf file determine which host runs each instance of a Ceph daemon. Using the exemplary ceph.conf file, you would perform the following:
mkcephfs 配置脚本不会创建默认服务器目录（貌似现在的版本可以），要为每个 ceph 进程手动创建目录。ceph.conf 配置文件里的 host 变量指定哪个主机运行哪个进程。用 ceph.conf 样板文件的话你要执行下面的命令：

On myserver01:

```
sudo mkdir /var/lib/ceph/osd/ceph-0
sudo mkdir /var/lib/ceph/mon/ceph-a
```

On myserver02:

```
sudo mkdir /var/lib/ceph/osd/ceph-1
sudo mkdir /var/lib/ceph/mon/ceph-b
```

On myserver03:

```
sudo mkdir /var/lib/ceph/osd/ceph-2
sudo mkdir /var/lib/ceph/mon/ceph-c
sudo mkdir /var/lib/ceph/mds/ceph-a
```

3.2.1.4 把硬盘挂载到数据目录

MOUNT DISKS TO THE DATA DIRECTORIES
If you are running multiple OSDs per host and one hard disk per OSD, you should mount the disk under the OSD data directory (if you haven't done so already).
如果你在每台主机上都要运行多个 OSD 进程，应该先把这些硬盘分别挂载到其数据目录下。

3.2.1.5 运行 mkcephfs

RUN MKCEPHFS
Once you have copied your Ceph Configuration to the OSD Cluster hosts and created the default directories, you may deploy Ceph with the mkcephfs script.
配置文件拷贝完毕、默认目录创建完毕后就可以用 mkcephfs 配置 ceph 集群了。

Note mkcephfs is a quick bootstrapping tool. It does not handle more complex operations, such as upgrades.
注意：mkcephfs 是快速起步工具，它不能处理复杂的操作，如升级。

For production environments, deploy Ceph using Chef cookbooks. To run mkcephfs, execute the following:
生产环境应该用 chef 配置 ceph。mkcephfs 命令的用法如下：

```
cd /etc/ceph
sudo mkcephfs -a -c /etc/ceph/ceph.conf -k ceph.keyring
```

The script adds an admin key to the ceph.keyring, which is analogous to a root password. See Authentication when running with cephx enabled.
这个脚本把一个管理密钥添加到了 ceph.keyring 里，它和 root 密码的功能类似。启用了 cephx 时参见认证部分。

When you start or stop your cluster, you will not have to use sudo or provide passwords. For example:
启动或关闭集群时不必使用 sudo 或者提供密码。

```
service ceph -a start
```

See Operating a Cluster for details.
详情参见 Operating a cluster。

3.2.2 ceph 部署

Ceph deploy
Coming soon.

3.2.3 安装 chef

Installing chef
Chef defines three types of entities:

- Chef Nodes: Run chef-client, which installs and manages software.
- Chef Server: Interacts with chef-client on Chef nodes.
- Chef Workstation: Manages the Chef server.

See Chef Architecture Introduction for details.
chef 定义了三种实体：

- chef 节点：运行 chef-client 客户端，用来安装和管理软件；
- chef 服务器：在 chef 节点上和 chef-client 交互；
- chef 工作站：管理 chef 服务器。

详情参见 chef 体系结构介绍。

3.2.3.1 创建一个 chef 用户

CREATE A CHEF USER
The chef-client command requires the proper privileges to install and manage installations. On each Chef node, we recommend creating a chef user with full root privileges. For example:
chef-client 命令需要合适的特权才能安装和管理软件，我们建议在每个 chef 节点上都创建一个 chef 用户，他应该有所有 root 特权，例如：

```
ssh user@chef-node
sudo useradd -d /home/chef -m chef
sudo passwd chef
```

To provide full privileges, add the following to /etc/sudoers.d/chef.

执行下面的命令给 chef 用户分配所有特权：

```
echo "chef ALL = (root) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/chef
sudo chmod 0440 /etc/sudoers.d/chef
```

If you are using a version of sudo that doesn't support includes, you will need to add the following to the /etc/sudoers file:
如果你所用的 sudo 不能包含其它配置文件，你要把下面这行添加到/etc/sudoers

```
chef ALL = (root) NOPASSWD:ALL
```

Important: Do not change the file permissions on /etc/sudoers. Use a suitable tool such as visudo.
重要：不要更改/etc/sudoers 的权限位，请用 visudo 或 sudoedit 编辑。

3.2.3.2 为 chef 客户端生成 ssh 密钥

GENERATE SSH KEYS FOR CHEF CLIENTS
Chef's knife tool can run ssh. To streamline deployments, we recommend generating an SSH key pair without a passphrase for your Chef nodes and copying the public key(s) to your Chef nodes so that you can connect to them from your workstation using ssh from knife without having to provide a password. To generate a key pair without a passphrase, execute the following on your Chef workstation.
chef 的 knife 工具可运行 ssh，为使部署流水化，建议您给 chef 节点生成一个无口令的 ssh 密钥对，并且把公钥拷贝到 chef 节点，这样你无需密码就可以用 knife 的 ssh 从工作站连接过去了，在工作站执行下面的命令：

```
ssh-keygen
Generating public/private key pair.
Enter file in which to save the key (/ceph-admin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /ceph-admin/.ssh/id_rsa.
Your public key has been saved in /ceph-admin/.ssh/id_rsa.pub.
```

You may use RSA or DSA keys. Once you generate your keys, copy them to each OSD host. For example:
你可以用 RSA 或 DSA 密钥，生成后拷贝到每个 OSD 主机，例如：

```
ssh-copy-id chef@your-node
```

Consider modifying your ~/.ssh/config file so that it defaults to logging in as chef when no username is specified.
请修改你的~/.ssh/config 配置文件，以便在没有指定用户名时默认用 chef 登录。

```
Host myserver01
        Hostname myserver01.fqdn-or-ip-address.com
        User chef
Host myserver02
        Hostname myserver02.fqdn-or-ip-address.com
        User chef
```

3.2.3.3 安装 Ruby

INSTALLING RUBY
Chef requires you to install Ruby. Use the version applicable to your current Linux distribution and install Ruby on all of your hosts.
chef 依赖 Ruby，选择适合您当前发行版的 Ruby 版本并且安装到所有主机。

```
sudo apt-get update
sudo apt-get install ruby
```

3.2.3.4 安装 chef 和 chef 服务器

INSTALLING CHEF AND CHEF SERVER ON A SERVER
If you plan on hosting your Chef Server at Opscode you may skip this step, but you must make a note of the the fully qualified domain name or IP address of your Chef Server for knife and chef-client.
如果你计划把 chef 服务器托管在 Opscode，那可以跳过这步，但你必须记录 chef 服务器的正式域名或 IP 地址，knife 和 chef-client 要用到。

First, add Opscode packages to your APT configuration. For example:
首先，在 APT 配置里添加 Opscode，例如：

```
sudo tee /etc/apt/sources.list.d/chef.list << EOF
deb http://apt.opscode.com/ $(lsb_release -cs)-0.10 main
deb-src http://apt.opscode.com/ $(lsb_release -cs)-0.10 main
EOF
```

Next, you must request keys so that APT can verify the packages. Copy and paste the following line into your command line:
然后，为 APT 申请密钥用于校验 Opscode 软件包，执行下列命令：

```
sudo touch /etc/apt/trusted.gpg.d/opscode-keyring.gpg && sudo gpg --fetch-key
http://apt.opscode.com/packages@opscode.com.gpg.key && sudo gpg --export 83EF826A | sudo apt-key
--keyring /etc/apt/trusted.gpg.d/opscode-keyring.gpg add - && sudo gpg --yes --delete-key 83EF826A
```

The key is only used by apt, so remove it from the root keyring by typing Y when prompted to delete it.
只有 apt 用这个密钥，所以提示删除的时候可以回答 Y 删除掉。

Install the Opscode keyring, Chef and Chef server on the host designated as your Chef Server.
在 chef 服务器上安装 Opscode 密钥环、chef 和 chef 服务器。

```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get install opscode-keyring chef chef-
server
```

Enter the fully qualified domain name or IP address for your Chef server. For example:
输入 chef 服务器的正式域名或 IP 地址，例如：

```
http://fqdn-or-ip-address.com:4000
```

The Chef server installer will prompt you to enter a temporary password. Enter a temporary password (e.g., foo) and proceed with the installation.
chef 服务器安装器会提示你输入一个临时密码，输入一个临时密码，然后继续安装。

Tip When prompted for a temporary password, you may press OK. The installer wants you to re-enter the password to confirm it. To re-enter the password, you must press the ESC key.
提示：提示要一个临时密码的时候，你可以按下 OK，安装器让你重新输入密码时必须按 ESC 键。

Once the installer finishes and activates the Chef server, you may enter the fully qualified domain name or IP address in a browser to launch the Chef web UI. For example:
安装器完成、启动 chef 服务器后，你可以在浏览器输入正式域名或者 IP 地址启动 chef 的 WEB 界面，例如：

```
http://fqdn-or-ip-address.com:4000
```

The Chef web UI will prompt you to enter the username and password.
chef 的 WEB 界面会提示你输入用户名和密码。

```
login: admin
password: foo
```

Once you have entered the temporary password, the Chef web UI will prompt you to enter a new password.
输入临时密码后，chef 的 WEB 界面会提示你输入新密码。

3.2.3.5 在剩余机器上安装 chef

INSTALL CHEF ON ALL REMAINING HOSTS
Install Chef on all Chef Nodes and on the Chef Workstation (if it is not the same host as the Chef Server). See Installing Chef Client on Ubuntu or Debian for details.
在所有 chef 节点和工作站（如果和 chef 服务器不是同一台主机）上安装 chef 程序。详情参见 Installing Chef Client on Ubuntu or Debian

First, add Opscode packages to your APT configuration. For example:
首先在 APT 源里添加 Opscode，例如：

```
sudo tee /etc/apt/sources.list.d/chef.list << EOF
deb http://apt.opscode.com/ $(lsb_release -cs)-0.10 main
deb-src http://apt.opscode.com/ $(lsb_release -cs)-0.10 main
EOF
```

Next, you must request keys so that APT can verify the packages. Copy and paste the following line into your command line:
然后，要申请 APT 校验软件包的密钥，执行下面的命令：

```
sudo touch /etc/apt/trusted.gpg.d/opscode-keyring.gpg && sudo gpg --fetch-key
http://apt.opscode.com/packages@opscode.com.gpg.key && sudo gpg --export 83EF826A | sudo apt-key
--keyring /etc/apt/trusted.gpg.d/opscode-keyring.gpg add - && sudo gpg --yes --delete-key 83EF826A
```

The key is only used by apt, so remove it from the root keyring by typing Y when prompted to delete it.
只有 apt 使用这个密钥，所以最后提示删除的时候可以按 Y 删除。

Install the Opscode keyring and Chef on all hosts other than the Chef Server.
在除 chef 服务器之外的所有主机上安装 Opscode 密钥环。

```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get install opscode-keyring chef
```

Enter the fully qualified domain name or IP address for your Chef server. For example:
输入 chef 服务器的正式域名或 IP 地址，例如：

```
http://fqdn-or-ip-address.com:4000
```

3.2.3.6 配置 knife

CONFIGURING KNIFE
Once you complete the Chef server installation, install knife on the your Chef Workstation. If the Chef server is a remote host, use ssh to connect.
chef 服务器安装完后，要在 chef 工作站安装 knife。如果 chef 服务器是远程主机，先 ssh 登录：

```
ssh chef@fqdn-or-ip-address.com
```

In the /home/chef directory, create a hidden Chef directory.
在/home/chef 目录下创建隐藏目录：

```
mkdir -p ~/.chef
```

The server generates validation and web UI certificates with read/write permissions for the user that installed the Chef server. Copy them from the /etc/chef directory to the ~/.chef directory. Then, change their ownership to the current user.
服务器会给 chef 服务器安装者生成认证和 web UI 证书，并设置合适的读写权限位，把它们从/etc/chef 拷贝到 ~/.chef，然后把所有者改为当前用户。

```
sudo cp /etc/chef/validation.pem /etc/chef/webui.pem ~/.chef && sudo chown $(id -u):$(id -g)
~/.chef/*.pem
```

From the current user's home directory, configure knife with an initial API client.
在当前用户的家目录下，用一个初始化 API 客户端配置 knife：

```
knife configure -i
```

The configuration will prompt you for inputs. Answer accordingly:
上面的配置会提问，相应回答：

```
Where should I put the config file? [~/.chef/knife.rb] Press Enter to accept the default value.

Please enter the chef server URL: If you are installing the client on the same host as the server,
enter http://localhost:4000. Otherwise, enter an appropriate URL for the server.

Please enter a clientname for the new client: Press Enter to accept the default value.
```

```
Please enter the existing admin clientname: Press Enter to accept the default value.

Please enter the location of the existing admin client's private key: Override the default value so
that it points to the .chef directory. (e.g., /home/chef/.chef/webui.pem)

Please enter the validation clientname: Press Enter to accept the default value.

Please enter the location of the validation key: Override the default value so that it points to
the .chef directory. (e.g., /home/chef/.chef/validation.pem)

Please enter the path to a chef repository (or leave blank): Leave the entry field blank and press
Enter.
```

3.2.3.7 菜谱路径

ADD A COOKBOOK PATH
Add cookbook_path to the ~/.chef/knife.rb configuration file on your Chef workstation. For example:
在你的 chef 工作站上把 cookbook_path 添加到~/.chef/knife.rb 配置文件里，例如：

```
cookbook_path '/home/{user-name}/chef-cookbooks/'
```

Then create the path if it doesn't already exist.
路径不存在的话先创建：

```
mkdir /home/{user-name}/chef-cookbooks
```

This is where you will store local copies of cookbooks before uploading them to the Chef server.
这是菜谱的本地副本位置，稍后上传到 chef 服务器。

3.2.3.8 把validation.pem 拷贝到所有节点

COPY VALIDATION.PEM TO NODES
Copy the /etc/chef/validation.pem file from your Chef server to each Chef Node. In a command line shell on the Chef Server, for each node, replace {nodename} in the following line with the node's host name and execute it.
把 chef 服务器上的/etc/chef/validation.pem 文件拷贝到每个 chef 节点，在 chef 服务器上的命令行 shell 执行下面的命令（用节点主机名替代{nodename}）。

```
sudo cat /etc/chef/validation.pem | ssh {nodename} "exec sudo tee /etc/chef/validation.pem
>/dev/null"
```

3.2.3.9 在每个节点运行chef-client

RUN CHEF-CLIENT ON EACH CHEF NODE
Run the chef-client on each Chef Node so that the nodes register with the Chef server.
在每个 chef 节点启动 chef-client，以注册到 chef 服务器。

```
ssh chef-node
sudo chef-client
```

3.2.3.10 检查节点

VERIFY NODES
Verify that you have setup all the hosts you want to use as Chef nodes.
检查下是否安装好了所有 chef 节点。

```
knife node list
```

A list of the nodes you've configured should appear.
应该显示你已经配置的节点列表。

See the Deploy With Chef section for information on using Chef to deploy your Ceph cluster.
关于如何用 chef 部署集群，请参见用 chef 部署。

3.2.4 用 chef 部署

Deploying with chef
We use Chef cookbooks to deploy Ceph. See Managing Cookbooks with Knife for details on using knife. For Chef installation instructions, see Installing Chef.
我们用 chef 菜谱部署 ceph，knife 用法参见 Managing Cookbooks with Knife，chef 安装参见安装 chef。

3.2.4.1 克隆必需的菜谱

CLONE THE REQUIRED COOKBOOKS
To get the cookbooks for Ceph, clone them from git.:
从 github 克隆 ceph 的菜谱：

```
cd ~/chef-cookbooks
git clone https://github.com/opscode-cookbooks/apache2.git
git clone https://github.com/ceph/ceph-cookbooks.git ceph
```

3.2.4.2 添加必需的菜谱路径

ADD THE REQUIRED COOKBOOK PATHS
If you added a default cookbook path when you installed Chef, knife may be able to upload the cookbook you've cloned to your cookbook path directory without further configuration. If you used a different path, or if the cookbook repository you cloned has a different tree structure, add the required cookbook path to your knife.rb file. The cookbook_path setting takes a string or an array of strings. For example, you can replace a string path with an array of string paths:
如果你安装 chef 的时候添加了默认菜谱路径，knife 无需过多配置就可以上传克隆的菜谱了。如果你要使用另一个路径，或者克隆的菜谱仓库树结构不同，把路径添加到你的 knife.rb 文件里，cookbook_path 可以是字符串或字符串阵列。例如，你可以用字符串阵列替代字符串：

```
cookbook_path '/home/{user-name}/chef-cookbooks/'
```

Becomes:
改为：

```
cookbook_path [
        '/home/{user-name}/chef-cookbooks/',
        '/home/{user-name}/chef-cookbooks/{another-directory}/',
        '/some/other/path/to/cookbooks/'
]
```

3.2.4.3 安装菜谱

INSTALL THE COOKBOOKS
To install Ceph, you must upload the Ceph cookbooks and the Apache cookbooks (for use with RADOSGW) to your Chef server.
要安装 ceph，你必须把 ceph 和 Apache 菜谱（RADOSGW 依赖）上传到 chef 服务器。

```
knife cookbook upload apache2 ceph
```

3.2.4.4 配置 ceph 环境

CONFIGURE YOUR CEPH ENVIRONMENT
The Chef server can support installation of software for multiple environments. The environment you create for Ceph requires an fsid, the secret for your monitor(s) if you are running Ceph with cephx authentication, and the host name (i.e., short name) for your monitor hosts.
chef 支持多种环境下的软件安装，你给 ceph 创建的环境需要一个 fsid、监视器的密钥（如果启用了 cephx 认证）、监视器的短主机名。

For the filesystem ID, use uuidgen from the uuid-runtime package to generate a unique identifier.
至于文件系统 ID，可以用 uuid-runtime 包里的 uuidgen 工具生成一个唯一的标识符：

```
uuidgen -r
```

For the monitor(s) secret(s), use ceph-authtool to generate the secret(s):
用 ceph-authtool 为监视器生成密钥：

```
sudo apt-get update
sudo apt-get install ceph-common
ceph-authtool /dev/stdout --name=mon. --gen-key
```

The secret is the value to the right of "key =", and should look something like this:
密钥是"key = "右边的值，长相如下：

```
AQBAMuJPINJgFhAAziXIrLvTvAz4PRo5IK/Log==
```

To create an environment for Ceph, set a command line editor. For example:
要给 ceph 创建环境，先设置一个命令行编辑器，例如：

```
export EDITOR=vim
```

Then, use knife to create an environment.
然后，用 knife 创建环境：

```
knife environment create {env-name}
```

For example:
例如：

```
knife environment create Ceph
```

A JSON file will appear. Perform the following steps:
将会出现一个 JSON 文件，依次完成下面的步骤：

Enter a description for the environment.
In "default_attributes": {}, add "ceph" : {}.
Within "ceph" : {}, add "monitor-secret":.
Immediately following "monitor-secret": add the key you generated within quotes, followed by a comma.
Within "ceph":{} and following the monitor-secret key-value pair, add "config": {}
Within "config": {} add "fsid":.
Immediately following "fsid":, add the unique identifier you generated within quotes, followed by a comma.
Within "config": {} and following the fsid key-value pair, add "mon_initial_members":
Immediately following "mon_initial_members":, enter the initial monitor host names.
For example:
为这个环境输入一个描述，在"default_attributes": {}里添加"ceph" : {}；
在"ceph" : {}里添加"monitor-secret":，紧接着在引号里加上生成的密钥，逗号结尾；
还是在"ceph":{}里，接着前面添加"config": {}，在"config": {}里添加"fsid":，然后在引号里填上生成的唯一标识符，逗号结尾；
还是在"config": {}里，紧接上面添加"mon_initial_members":，输入初始的监视器主机名。
例如：

```
"default_attributes" : {
        "ceph": {
                "monitor-secret": "{replace-with-generated-secret}",
                "config": {
                        "fsid": "{replace-with-generated-uuid}",
                        "mon_initial_members": "{replace-with-monitor-hostname(s)}"
                }
        }
}
```

Advanced users (i.e., developers and QA) may also add "ceph_branch": "{branch}" to default-attributes, replacing {branch} with the name of the branch you wish to use (e.g., master).
高级用户（如开发者和质检人员）也可以把"ceph_branch": "{branch}"添加到默认属性里，起个分支名字（如master）替代{branch}。

3.2.4.5 配置角色

CONFIGURE THE ROLES
Navigate to the Ceph cookbooks directory.
进入 ceph 菜谱目录:

```
cd ~/chef-cookbooks/ceph
```

Create roles for OSDs, monitors, metadata servers, and RADOS Gateways from their respective role files.
分别用它们的角色文件给 OSD、监视器、元数据服务器、RADOS 网关创建角色。

```
knife role from file roles/ceph-osd.rb
knife role from file roles/ceph-mon.rb
knife role from file roles/ceph-mds.rb
knife role from file roles/ceph-radosgw.rb
```

3.2.4.6 配置节点

CONFIGURE NODES
You must configure each node you intend to include in your Ceph cluster. Identify nodes for your Ceph cluster.
每个规划到 ceph 集群里的节点都必需配置，用下面的命令标识节点:

```
knife node list
```

For each node you intend to use in your Ceph cluster, configure the node as follows:
对规划的 ceph 集群的每个节点，都要用下面的命令配置:

```
knife node edit {node-name}
```

The node configuration should appear in your text editor. Change the chef_environment value to Ceph (or whatever name you set for your Ceph environment).
要配置的节点的配置信息应该出现在文本编辑器里，把 chef_environment 的值改为 ceph（你给 ceph 环境设置的名字）。

In the run_list, add "recipe[ceph::apt]", to all nodes as the first setting, so that Chef can install or update the necessary packages. Then, add at least one of:
在所有节点的 run_list 里添加"recipe[ceph::apt]"，这是第一个配置，这样 chef 就能安装或更新必要的软件包了。至少然后添加下面几行中的一个:

```
"role[ceph-mon]"
"role[ceph-osd]"
"role[ceph-mds]"
"role[ceph-radosgw]"
```

If you add more than one role, separate them with a comma. Run hostname on your command line, and replace the {hostname} setting of the name key to the host name for the node.
如果你添加的角色不止一个，要用逗号分隔。用 hostname 命令的结果替换配置里 name 的值{hostname}作为节点的主机名。

```
{
        "chef_environment": "Ceph",
        "name": "{hostname}",
        "normal": {
        "tags": [

        ]
        },
         "run_list": [
                "recipe[ceph::apt]",
                "role[ceph-mon]",
                "role[ceph-mds]"
        ]
}
```

3.2.4.7 准备OSD 硬盘

PREPARE OSD DISKS

Configuring a node with an OSD role tells Chef that the node will run at least one OSD. However, you may run many OSDs on one host. For example, you may run one ceph-osd daemon for each data disk on the system. This step prepares the OSD disk(s) and tells Chef how many OSDs the node will be running.
给一个节点配置了 OSD 角色意味着这个节点要运行至少一个 OSD 进程，你可以在一台主机上运行很多 OSD，例如，你可以给系统里每个数据盘配置一个 ceph-osd 守护进程。这个步骤准备好 OSD 硬盘并且告诉 chef 这个节点将运行多少个 OSD。

For the Ceph 0.48 Argonaut release, install gdisk:
ceph 版本是 0.48 时要安装 gdisk:

```
sudo apt-get install gdisk
```

For the Ceph 0.48 Argonaut release, on each hard disk that will store data for an OSD daemon, configure the hard disk for use with Ceph. Replace {fsid} with the UUID you generated while using uuidgen -r.
对 ceph 0.48 版来说，用下面的命令配置每个 OSD 所用的存储数据硬盘，用 uuidgen -r 命令生成的 UUID 替代 {fsid}:

```
sudo sgdisk /dev/{disk} --zap-all --clear --mbrtogpt --largest-new=1 --change-name=1:'ceph data'
--typecode=1:{fsid}
```

Create a file system and allocate the disk to your cluster. Specify a filesystem (e.g., ext4, xfs, btrfs). When you execute ceph-disk-prepare, remember to replace {fsid} with the UUID you generated while using uuidgen -r:
创建文件系统、然后把它分配给集群，执行 ceph-disk-prepare 的时候要指定一个文件系统（如 ext4、xfs 或 btrfs），记得用 uuidgen -r 生成的 UUID 替代{fsid}:

```
sudo mkfs -t ext4 /dev/{disk}
sudo mount -o user_xattr /dev/{disk} /mnt
sudo ceph-disk-prepare --cluster-uuid={fsid} /mnt
sudo umount /mnt
```

Finally, simulate a hotplug event.
最后，模拟一个热插拔事件:

```
sudo udevadm trigger --subsystem-match=block --action=add
```

3.2.4.8 在每个节点运行chef-client

RUN CHEF-CLIENT ON EACH NODE
Once you have completed the preceding steps, you must run chef-client on each node. For example:
完成前面的步骤后，必须在每个节点运行 chef-client，例如:

```
sudo chef-client
```

3.2.4.9 继续集群运维 :)

PROCEED TO OPERATING THE CLUSTER
Once you complete the deployment, you may begin operating your cluster. See Operating a Cluster for details.
完成部署后，就可以开始集群运维了，详情参见

3.3 运维

Once you have a deployed Ceph cluster, you may begin operating your cluster.
HIGH-LEVEL OPERATIONS
High-level cluster operations consist primarily of starting, stopping, and restarting a cluster with the ceph service; checking the cluster's health; and, monitoring an operating cluster.
ceph 集群部署完毕后，就开始集群运维了。

高级操作
高级集群操作主要包括用服务管理脚本启动、停止、重启集群、和集群健康状态检查、监控和操作集群。

3.3.1 操作一个集群

Operating a cluster
The ceph service provides functionality to start, restart, and stop your Ceph cluster. Each time you execute ceph processes, you must specify at least one option and one command. You may also specify a daemon type or a daemon instance. For most newer Debian/Ubuntu distributions, you may use the following syntax:
ceph 服务提供了启动、重启、停止 ceph 集群，每次执行的时候必须指定至少一个选项和一个命令，还必须指定一个守护进程类型或例程名称。对最新的 Debian/Ubuntu 发行版，你可以用下面的语法：

```
sudo service ceph [options] [commands] [daemons]
```

For older distributions, you may wish to use the /etc/init.d/ceph path:
对较老的发行版，你也许想用/etc/init.d/ceph：

```
sudo /etc/init.d/ceph [options] [commands] [daemons]
```

The ceph service options include:
ceph 服务的选项包括：

Option	Shortcut	Description
`--verbose`	`-v`	Use verbose logging. 详细的日志。
`--valgrind`	`N/A`	(Dev and QA only) Use Valgrind debugging. （只适用于开发者和品质保证人员）使用 Valgrind 调试。
`--allhosts`	`-a`	Execute on all hosts in `ceph.conf.` Otherwise, it only executes on `localhost`. 在 ceph.conf 里配置的所有主机上执行，否则它只在本机执行。
`--restart`	`N/A`	Automatically restart daemon if it core dumps. 核心转储后自动重启。
`--norestart`	`N/A`	Don't restart a daemon if it core dumps. 核心转储后不自动重启。
`--conf`	`-c`	Use an alternate configuration file. 使用另外一个配置文件。

The ceph service commands include:
ceph 服务的命令包括：

Command	Description
`start`	Start the daemon(s).
`stop`	Stop the daemon(s).
`forcestop`	Force the daemon(s) to stop. Same as `kill -9`
`killall`	Kill all daemons of a particular type.
`cleanlogs`	Cleans out the log directory.
`cleanalllogs`	Cleans out **everything** in the log directory.

For subsystem operations, the ceph service can target specific daemon types by adding a particular daemon type for the [daemons] option. Daemon types include:
至于子系统操作，ceph 服务能指定守护进程类型，在[daemons]处指定守护进程类型就行了，守护进程类型包括：

- mon
- osd
- mds

The ceph service's [daemons] setting may also target a specific instance:
ceph 服务的[daemons]设置也可以指定一个具体例程：

```
sudo /etc/init.d/ceph -a start osd.0
```

Where osd.0 is the first OSD in the cluster.
这里 osd.0 是集群里的第一个 OSD。

3.3.1.1 启动集群

STARTING A CLUSTER
To start your Ceph cluster, execute ceph with the start command. The usage may differ based upon your Linux distribution. For example, for most newer Debian/Ubuntu distributions, you may use the following syntax:
要启动 ceph 集群，执行 ceph 的时候加上 start 命令，用法可能因你的 Linux 发行版而有所不同，例如对大多数较新的 Debian/Ubuntu 你可以用下面的语法：

```
sudo service ceph start [options] [start|restart] [daemonType|daemonID]
```

For older distributions, you may wish to use the /etc/init.d/ceph path:
对较老的发行版可能要用/etc/init.d/ceph：

```
sudo /etc/init.d/ceph [options] [start|restart] [daemonType|daemonID]
```

The following examples illustrates a typical use case:
下面的命令展示了典型用法：

```
sudo service ceph -a start
sudo /etc/init.d/ceph -a start
```

Once you execute with -a, Ceph should begin operating. You may also specify a particular daemon instance to constrain the command to a single instance. For example:
使用-a 选项可操作整个集群，你也可以指定一个具体的守护进程例程把操作限制到一个单独的例程，例如：

```
sudo /etc/init.d/ceph start osd.0
```

3.3.1.2 停止集群

STOPPING A CLUSTER
To stop your Ceph cluster, execute ceph with the stop command. The usage may differ based upon your Linux distribution. For example, for most newer Debian/Ubuntu distributions, you may use the following syntax:
要停止 ceph 集群，可以在执行 ceph 时加上 stop 命令，用法因 Linux 发行版不同而有所差异。例如，在最新的 Debian/Ubuntu 上可以用下面的语法：

```
sudo service ceph [options] stop [daemonType|daemonID]
```

For example:
例如：

```
sudo service -a ceph stop
```

For older distributions, you may wish to use the /etc/init.d/ceph path:
在较老的发行版上可以用/etc/init.d/ceph：

```
sudo /etc/init.d/ceph -a stop
```

Ceph should shut down the operating processes.
ceph 会关闭运行的进程。

3.3.2 监控集群

Monitoring a cluster
Once you have a running cluster, you may use the ceph tool to monitor your cluster. Monitoring a cluster typically involves checking OSD status, monitor status, placement group status and metadata server status.
集群运行起来后，你可以用 ceph 工具来监控，典型的监控包括检查 OSD 状态、监视器状态、归置组状态和元数据服务器状态。

3.3.2.1 交互模式

要在交互模式下运行 ceph，不要带参数运行 ceph，例如：

```
ceph
```

```
ceph> health
ceph> status
ceph> quorum_status
ceph> mon_status
```

3.3.2.2 检查集群健康状况

CHECKING CLUSTER HEALTH
After you start your cluster, and before you start reading and/or writing data, check your cluster's health first. You can check on the health of your Ceph cluster with the following:
启动集群后、读写数据前，先检查下集群的健康状态。你可以用下面的命令检查：

```
ceph health
```

If you specified non-default locations for your configuration or keyring, you may specify their locations:
如果你的 ceph.conf 或密钥环不在默认路径下，你得指定：

```
ceph -c /path/to/conf -k /path/to/keyring health
```

Upon starting the Ceph cluster, you will likely encounter a health warning such as HEALTH_WARN XXX num placement groups stale. Wait a few moments and check it again. When your cluster is ready, ceph health should return a message such as HEALTH_OK. At that point, it is okay to begin using the cluster.
集群起来的时候，你也许会碰到像 HEALTH_WARN XXX num placement groups stale 这样的健康告警，等一会再检查下。集群准备好的话 ceph health 会给出像 HEALTH_OK 一样的消息，这时候就可以开始使用集群了。

3.3.2.3 观察集群

To watch the cluster's ongoing events, open a new terminal. Then, enter:
要观察集群内正发生的事件，打开一个新终端，然后输入：

```
ceph -w
```

Ceph will print each version of the placement group map and their status. For example, a tiny Ceph cluster consisting of one monitor, one metadata server and two OSDs may print the following:
ceph 会打印每个归置组版本和它们的状态，例如一个包括 1 个监视器、1 个元数据服务器和 2 个 OSD 的小型 ceph 集群可能会打印下面的：

```
health HEALTH_OK
monmap e1: 1 mons at {a=192.168.0.1:6789/0}, election epoch 0, quorum 0 a
osdmap e13: 2 osds: 2 up, 2 in
 placement groupmap v9713: 384 placement groups: 384 active+clean; 8730 bytes data, 22948 MB used,
264 GB / 302 GB avail
mdsmap e4: 1/1/1 up {0=a=up:active}

     2012-08-01 11:33:53.831268 mon.0 [INF] placement groupmap v9712: 384 placement groups: 384
active+clean; 8730 bytes data, 22948 MB used, 264 GB / 302 GB avail
     2012-08-01 11:35:31.904650 mon.0 [INF] placement groupmap v9713: 384 placement groups: 384
active+clean; 8730 bytes data, 22948 MB used, 264 GB / 302 GB avail
     2012-08-01 11:35:53.903189 mon.0 [INF] placement groupmap v9714: 384 placement groups: 384
active+clean; 8730 bytes data, 22948 MB used, 264 GB / 302 GB avail
     2012-08-01 11:37:31.865809 mon.0 [INF] placement groupmap v9715: 384 placement groups: 384
active+clean; 8730 bytes data, 22948 MB used, 264 GB / 302 GB avail
```

3.3.2.4 检查一个集群的状态

CHECKING A CLUSTER'S STATUS
To check a cluster's status, execute the following:
要检查集群的状态，执行下面的命令：

```
ceph status
```

或者：

```
ceph -s
```

In interactive mode, type status and press Enter.
在交互模式下，输入 status 然后按回车：

```
ceph> status
```

Ceph will print the cluster status. For example, a tiny Ceph cluster consisting of one monitor, one metadata server and two OSDs may print the following:
ceph 将打印集群状态，例如一个包括 1 个监视器、1 个元数据服务器和 2 个 OSD 的小型 ceph 集群可能打印：

```
health HEALTH_OK
monmap e1: 1 mons at {a=192.168.0.1:6789/0}, election epoch 0, quorum 0 a
osdmap e13: 2 osds: 2 up, 2 in
 placement groupmap v9754: 384 placement groups: 384 active+clean; 8730 bytes data, 22948 MB used,
264 GB / 302 GB avail
mdsmap e4: 1/1/1 up {0=a=up:active}
```

3.3.2.5 检查 OSD 状态

CHECKING OSD STATUS
An OSD's status is either in the cluster (in) or out of the cluster (out); and, it is either up and running (up), or it is down and not running (down). If an OSD is up, it may be either in in the cluster (you can read and write data) or it is out of the cluster out. If it is down, it should also be out. If an OSD is down and in, there is a problem.
OSD 状态是集群内(in)或集群外(out)状态，而且是活着且在运行(up)或挂了且不在运行(down)。如果一个 OSD 活着它也可以是在集群内（你可以读写数据）或者不在集群内；如果它挂了，它应该在集群外，如果它挂了且在集群内，肯定有问题。

You can check OSDs to ensure they are up and in by executing:
你可以执行下列命令来确定 OSD 活着且在集群里：

```
ceph osd stat
```

或者：

```
ceph osd dump
```

You can also check view OSDs according to their position in the CRUSH map.
你也可以根据 OSD 在 CRUSH 图里的位置来查看：

```
ceph osd tree
```

Ceph will print out a CRUSH tree with a host, its OSDs, whether they are up and their weight.
ceph 会打印 CRUSH 的树状态、它的 OSD 例程、状态、权重：

```
# id    weight  type name       up/down reweight
-1      3       pool default
-3      3               rack mainrack
-2      3                       host osd-host
0       1                               osd.0   up      1
1       1                               osd.1   up      1
2       1                               osd.2   up      1
```

3.3.2.6 检查监视器状态

CHECKING MONITOR STATUS
If your cluster has multiple monitors (likely), you should check the monitor quorum status after you start the cluster before reading and/or writing data. A quorum must be present when multiple monitors are running. You should also check monitor status periodically to ensure that they are running.
如果你有多个监视器（很可能），你启动集群后、读写数据前应该检查监视器法定人数状态。多个监视器必须活着、且在运行，要周期性检查监视器状态来确定它们在运行：

To see display the monitor map, execute the following:
要查看监视器图，执行下面的命令：

```
ceph mon stat
```

或者：

```
ceph mon dump
```

To check the quorum status for the monitor cluster, execute the following:
要检查监视器的法定人数状态，执行下面的命令：

```
ceph quorum_status
```

Ceph will return the quorum status. For example, a Ceph cluster consisting of three monitors may return the following:
ceph 会返回法定人数状态，例如，包含 3 个监视器的 ceph 集群可能返回下面的：

```
{ "election_epoch": 10,
  "quorum": [
        0,
        1,
        2],
  "monmap": { "epoch": 1,
      "fsid": "444b489c-4f16-4b75-83f0-cb8097468898",
      "modified": "2011-12-12 13:28:27.505520",
      "created": "2011-12-12 13:28:27.505520",
      "mons": [
            { "rank": 0,
              "name": "a",
              "addr": "127.0.0.1:6789\/0"},
            { "rank": 1,
              "name": "b",
              "addr": "127.0.0.1:6790\/0"},
            { "rank": 2,
              "name": "c",
              "addr": "127.0.0.1:6791\/0"}
           ]
    }
}
```

3.3.2.7 检查 MDS 状态

CHECKING MDS STATUS
Metadata servers provide metadata services for Ceph FS. Metadata servers have two sets of states: up | down and active | inactive. To ensure your metadata servers are up and active, execute the following:
元数据服务器为 ceph 文件系统提供元数据服务，元数据服务器有两种状态：up|down 和活跃|不活跃，执行下面的命令来确保元数据服务器活着且活跃：

```
ceph mds stat
```

To display details of the metadata cluster, execute the following:
要展示元数据集群的详细状态，执行下面的命令：

```
ceph mds dump
```

3.3.2.8 检查归置组状态

CHECKING PLACEMENT GROUP STATES
Placement groups map objects to OSDs. When you monitor your placement groups, you will want them to be active and clean. For other PG states, see Placement Group States.
归置组把对象映射到 OSD，归置组的状态应该是活跃且未污染的，其它的 PG 状态请参见{Placement Group States}。

3.3.3 CPU 剖析

If you built Ceph from source and compiled Ceph for use with oprofile you can profile Ceph's CPU usage. See Installing Oprofile for details.
如果你从源码编译、且启用了 oprofile，那么你就可以剖析 ceph 的 CPU 用法了，详情参见 Installing Oprofile。

INITIALIZING OPROFILE
The first time you use oprofile you need to initialize it. Locate the vmlinux image corresponding to the kernel you are now running.
你首次使用 oprofile 的时候要初始化，找到对应于当前运行内核的 vmlinux 位置：

```
ls /boot
```

```
sudo opcontrol --init
sudo opcontrol --setup --vmlinux={path-to-image} --separate=library --callgraph=6
```

3.3.3.1 启动 oprofile

STARTING OPROFILE
To start oprofile execute the following command:
执行下面的命令启动 opeofile：

```
opcontrol --start
```

Once you start oprofile, you may run some tests with Ceph.
启动 oprofile 后，你可以运行一些 ceph 测试：

3.3.3.2 停止 oprofile

STOPPING OPROFILE
To stop oprofile execute the following command:
执行下面的命令停止 oprofile：

```
opcontrol --stop
```

3.3.3.3 查看 oprofile 运行结果

RETRIEVING OPROFILE RESULTS
To retrieve the top cmon results, execute the following command:
要查看 cmon 最近的结果，执行下面的命令：

```
opreport -gal ./cmon | less
```

To retrieve the top cmon results with call graphs attached, execute the following command:
要检索 cmon 最近的调用图结果，执行下面的命令：

```
opreport -cal ./cmon | less
```

Important After reviewing results, you should reset oprofile before running it again. Resetting oprofile removes data from the session directory.
重要：回顾结果后，重新剖析前应该先重置，重置动作从会话目录里删除了数据。

3.3.3.4 重置 oprofile

RESETTING OPROFILE
To reset oprofile, execute the following command:
要重置 oprofile，执行下面的命令：

```
sudo opcontrol --reset
```

Important You should reset oprofile after analyzing data so that you do not commingle results from different tests.
重要：你应该分析后再重置，以免混合不同的剖析结果。

3.3.4 故障排除

When monitoring your cluster, you may receive health warnings and you may also notice that not all of your daemons are running properly. The following sections will help you identify and resolve daemon operations issues.

3.3.4.1 OSD 失败

RECOVERING FROM OSD FAILURES
监控集群的时候，你可能收到健康告警或者某些例程运行不正常，下面的一些段落会帮你识别和解决守护进程

运维问题。

3.3.4.1.1 单个 osd 失败

When a ceph-osd process dies, the monitor will learn about the failure from surviving ceph-osd daemons and report it via the ceph health command:
ceph-osd 挂的时候，监视器将了解 ceph-osd 的存活情况，且通过 ceph health 命令报告：

```
ceph health
HEALTH_WARN 1/3 in osds are down
```

Specifically, you will get a warning whenever there are ceph-osd processes that are marked in and down. You can identify which ceph-osds are down with:
而且，有 ceph-osd 进程标记为 in 且 down 的时候，你会得到警告，你可以用下面的命令得知哪个 ceph-osd 进程挂了：

```
ceph health detail
HEALTH_WARN 1/3 in osds are down
osd.0 is down since epoch 23, last address 192.168.106.220:6800/11080
```

Under normal circumstances, simply restarting the ceph-osd daemon will allow it to rejoin the cluster and recover. If there is a disk failure or other fault preventing ceph-osd from functioning or restarting, an error message should be present in its log file in /var/log/ceph.
一般情况下，简单地重启 ceph-osd 进程可使之重新加入集群并且恢复，如果有个硬盘失败或其它错误使 ceph-osd 不能正常运行或重启，一条错误信息将会出现在日志文件/var/log/ceph/里。

If the daemon stopped because of a heartbeat failure, the underlying kernel file system may be unresponsive. Check dmesg output for disk or other kernel errors.
如果守护进程因心跳失败、或者底层文件系统无响应而停止，查看 dmesg 获取硬盘或者内核错误。

If the problem is a software error (failed assertion or other unexpected error), it should be reported to the mailing list.
如果是软件错误（失败的插入或其它意外错误），就应该回馈到邮件列表。

3.3.4.1.2 集群没有空闲硬盘空间

THE CLUSTER HAS NO FREE DISK SPACE
If the cluster fills up, the monitor will prevent new data from being written. The system puts ceph-osds in two categories: nearfull and full, with configurable threshholds for each (80% and 90% by default). In both cases, full ceph-osds will be reported by ceph health:
如果集群填满了，监视器将阻止新数据写入，系统会把 ceph-osd 分为两类：快满了和满了，它们都有可配置的阀值（默认分别是 80%和 90%），满了的 ceph-osd 会被 ceph health 报告：

```
ceph health
HEALTH_WARN 1 nearfull osds
osd.2 is near full at 85%
```

或者：

```
ceph health
HEALTH_ERR 1 nearfull osds, 1 full osds
osd.2 is near full at 85%
osd.3 is full at 97%
```

The best way to deal with a full cluster is to add new ceph-osds, allowing the cluster to redistribute data to the newly available storage.
处理这种情况的最好方法就是增加新的 ceph-osd，这允许集群把数据重分布到新 OSD 里。

3.3.4.1.3 无根归置组

HOMELESS PLACEMENT GROUPS
It is possible for all OSDs that had copies of a given placement groups to fail. If that's the case, that subset of the object store is unavailable, and the monitor will receive no status updates for those placement groups. To detect this situation, the monitor marks any placement group whose primary OSD has failed as stale. For example:

拥有归置组拷贝的 OSD 都可以失败，在这种情况下，那一部分的对象存储不可用，监视器就不会收到那些归置组的状态更新了。为检测这种情况，监视器把任何主 OSD 失败的归置组标记为 stale（不新鲜），例如：

```
ceph health
HEALTH_WARN 24 pgs stale; 3/300 in osds are down
```

You can identify which placement groups are stale, and what the last OSDs to store them were, with:
你能找出哪些归置组不新鲜、和存储这些归置组的最新 OSD，命令如下：

```
ceph health detail
HEALTH_WARN 24 pgs stale; 3/300 in osds are down
...
pg 2.5 is stuck stale+active+remapped, last acting [2,0]
...
osd.10 is down since epoch 23, last address 192.168.106.220:6800/11080
osd.11 is down since epoch 13, last address 192.168.106.220:6803/11539
osd.12 is down since epoch 24, last address 192.168.106.220:6806/11861
```

If we want to get placement group 2.5 back online, for example, this tells us that it was last managed by osd.0 and osd.2. Restarting those ceph-osd daemons will allow the cluster to recover that placement group (and, presumably, many others).
如果想使归置组 2.5 重新在线，例如，上面的输出告诉我们它最后由 osd.0 和 osd.2 处理，重启这些 ceph-osd 将恢复之（还有其它的很多 PG）。

3.3.4.1.4 卡住的归置组

STUCK PLACEMENT GROUPS
It is normal for placement groups to enter states like “degraded” or “peering” following a failure. Normally these states indicate the normal progression through the failure recovery process. However, if a placement group stays in one of these states for a long time this may be an indication of a larger problem. For this reason, the monitor will warn when placement groups get “stuck” in a non-optimal state. Specifically, we check for:
有失败时归置组进入“degraded”（降级）或“peering”（连接建立中）状态，这事时有发生，通常这些状态意味着正常的失败恢复正在进行。然而，如果一个归置组长时间处于某个这些状态就意味着有更大的问题，因此监视器在归置组卡(stuck)在非最优状态时会警告，具体地，我们检查：

inactive - The placement group has not been active for too long (i.e., it hasn’t been able to service read/write requests).
unclean - The placement group has not been clean for too long (i.e., it hasn’t been able to completely recover from a previous failure).
stale - The placement group status has not been updated by a ceph-osd, indicating that all nodes storing this placement group may be down.
inactive（不活跃）——归置组长时间无活跃（例如它不能提供读写服务了）；
unclean（不干净）——归置组长时间不干净（例如它未能从前面的失败完全恢复）；
stale（不新鲜）——归置组状态没有被 ceph-osd 更新，表明存储这个归置组的所有节点可能都挂了。

You can explicitly list stuck placement groups with one of:
你可以明确列出卡住的归置组：

```
ceph pg dump_stuck stale
ceph pg dump_stuck inactive
ceph pg dump_stuck unclean
```

For stuck stale placement groups, it is normally a matter of getting the right ceph-osd daemons running again. For stuck inactive placement groups, it is usually a peering problem (see Placement Group Down - Peering Failure). For stuck unclean placement groups, there is usually something preventing recovery from completing, like unfound objects (see Unfound Objects);
处于 stuck stale 状态的归置组通过修复 ceph-osd 进程通常可以修复；处于 stuck inactive 状态的归置组通常是连接建立问题（参见归置组挂了——连接建立失败）；处于 stuck unclean 状态的归置组通常是由于某些东西阻止了恢复的完成，像未找到的对象（参见未找到的对象）。

3.3.4.1.5 归置组挂了——连接建立失败

PLACEMENT GROUP DOWN - PEERING FAILURE
In certain cases, the ceph-osd Peering process can run into problems, preventing a PG from becoming active and usable. For example, ceph health might report:
在某些情况下，ceph-osd 连接建立进程会遇到问题，使 PG 不能活跃、可用，例如 ceph health 也许显示：

```
ceph health detail
HEALTH_ERR 7 pgs degraded; 12 pgs down; 12 pgs peering; 1 pgs recovering; 6 pgs stuck unclean;
114/3300 degraded (3.455%); 1/3 in osds are down
...
pg 0.5 is down+peering
pg 1.4 is down+peering
...
osd.1 is down since epoch 69, last address 192.168.106.220:6801/8651
```

We can query the cluster to determine exactly why the PG is marked down with:
可以查询到 PG 为何被标记为 down：

```
ceph pg 0.5 query
{ "state": "down+peering",
  ...
  "recovery_state": [
        { "name": "Started\/Primary\/Peering\/GetInfo",
          "enter_time": "2012-03-06 14:40:16.169679",
          "requested_info_from": []},
        { "name": "Started\/Primary\/Peering",
          "enter_time": "2012-03-06 14:40:16.169659",
          "probing_osds": [
                0,
                1],
          "blocked": "peering is blocked due to down osds",
          "down_osds_we_would_probe": [
                1],
          "peering_blocked_by": [
                { "osd": 1,
                  "current_lost_at": 0,
                  "comment": "starting or marking this osd lost may let us proceed"}]},
        { "name": "Started",
          "enter_time": "2012-03-06 14:40:16.169513"}
    ]
}
```

The recovery_state section tells us that peering is blocked due to down ceph-osd daemons, specifically osd.1. In this case, we can start that ceph-osd and things will recover.
recovery_state 段告诉我们连接建立因 ceph-osd 进程挂了而被阻塞，本例是 osd.1 挂了，启动这个进程应该就可以恢复。

Alternatively, if there is a catastrophic failure of osd.1 (e.g., disk failure), we can tell the cluster that it is lost and to cope as best it can.
另外，如果 osd.1 是灾难性的失败（如硬盘损坏），我们可以告诉集群它丢失了，让集群尽力完成副本拷贝。

> Important This is dangerous in that the cluster cannot guarantee that the other copies of the data are consistent and up to date.
> 重要：集群不能保证其它数据副本是一致且最新就危险了！

To instruct Ceph to continue anyway:
无论如何让 ceph 继续：

```
ceph osd lost 1
```

Recovery will proceed.
恢复将继续。

3.3.4.1.6 未找到的对象

UNFOUND OBJECTS
Under certain combinations of failures Ceph may complain about unfound objects:

某几种失败相组合可能导致 ceph 抱怨有对象丢失：

```
ceph health detail
HEALTH_WARN 1 pgs degraded; 78/3778 unfound (2.065%)
pg 2.4 is active+degraded, 78 unfound
```

This means that the storage cluster knows that some objects (or newer copies of existing objects) exist, but it hasn't found copies of them. One example of how this might come about for a PG whose data is on ceph-osds 1 and 2:
这意味着存储集群知道一些对象（或者存在对象的较新副本）存在，却没有找到它们的副本。下例展示了这种情况是如何发生的，一个 PG 的数据存储在 ceph-osd 1 和 2 上：

1 goes down
2 handles some writes, alone
1 comes up
1 and 2 repeer, and the objects missing on 1 are queued for recovery.
Before the new objects are copied, 2 goes down.
Now 1 knows that these object exist, but there is no live ceph-osd who has a copy. In this case, IO to those objects will block, and the cluster will hope that the failed node comes back soon; this is assumed to be preferable to returning an IO error to the user.
1 挂了；2 独自处理一些写动作；1 启动了；1 和 2 重新建立连接，1 上面丢失的对象加入队列准备恢复；新对象还未拷贝完，2 挂了。这时，1 知道这些对象存在，但是活着的 ceph-osd 都没有副本，这种情况下，读写这些对象的 IO 就会被阻塞，集群只能指望节点早点恢复。这时我们假设用户希望先得到一个 IO 错误。

First, you can identify which objects are unfound with:
首先，你应该确认哪些对象找不到了：

```
ceph pg 2.4 list_missing [starting offset, in json]
{ "offset": { "oid": "",
     "key": "",
     "snapid": 0,
     "hash": 0,
     "max": 0},
 "num_missing": 0,
 "num_unfound": 0,
 "objects": [
    { "oid": "object 1",
      "key": "",
      "hash": 0,
      "max": 0 },
    ...
 ],
 "more": 0}
```

If there are too many objects to list in a single result, the more field will be true and you can query for more. (Eventually the command line tool will hide this from you, but not yet.)
如果在一次查询里列出的对象太多，more 这个域将为 true，因此你可以查询 more。（命令行工具可能隐藏了，但这里没有）

Second, you can identify which OSDs have been probed or might contain data:
其次，你可以找出哪些 OSD 上探测到、或可能包含数据：

```
ceph pg 2.4 query
"recovery_state": [
     { "name": "Started\/Primary\/Active",
       "enter_time": "2012-03-06 15:15:46.713212",
       "might_have_unfound": [
             { "osd": 1,
               "status": "osd is down"}]},
```

In this case, for example, the cluster knows that osd.1 might have data, but it is down. The full range of possible states include:
本例中，集群知道 osd.1 可能有数据，但它挂了。所有可能的状态有 L

- already probed（已经探测到了）
- querying（在查询）

- osd is down（OSD 挂了）
- not queried (yet)（尚未查询）

Sometimes it simply takes some time for the cluster to query possible locations.
有时候集群要花一些时间来查询可能的位置。

It is possible that there are other locations where the object can exist that are not listed. For example, if a ceph-osd is stopped and taken out of the cluster, the cluster fully recovers, and due to some future set of failures ends up with an unfound object, it won't consider the long-departed ceph-osd as a potential location to consider. (This scenario, however, is unlikely.)
还有一种可能性，对象存在于其它位置却未被列出，例如，集群里的一个 ceph-osd 停止且被剔出，然后完全恢复了；后来的失败、恢复后仍有未找到的对象，它也不会觉得早已死亡的 ceph-osd 上仍可能包含这些对象。（这种情况几乎不太可能发生）。

If all possible locations have been queried and objects are still lost, you may have to give up on the lost objects. This, again, is possible given unusual combinations of failures that allow the cluster to learn about writes that were performed before the writes themselves are recovered. To mark the "unfound" objects as "lost":
如果所有位置都查询过了仍有对象丢失，那就得放弃丢失的对象了。这仍可能是罕见的失败组合导致的，集群在写入完成前，未能得知写入是否已执行。以下命令把未找到的(unfound)对象标记为丢失(lost)。

```
ceph pg 2.5 mark_unfound_lost revert
```

This the final argument specifies how the cluster should deal with lost objects. Currently the only supported option is "revert", which will either roll back to a previous version of the object or (if it was a new object) forget about it entirely. Use this with caution, as it may confuse applications that expected the object to exist.
上述最后一个参数告诉集群应如何处理丢失的对象。当前只支持 revert 选项，它使得回滚到对象的前一个版本（如果它是新对象）或完全忽略它。要谨慎使用，它可能迷惑那些期望对象存在的应用程序。

3.3.4.1.7 龟速或反应迟钝的 OSD

SLOW OR UNRESPONSIVE OSD
If, for some reason, a ceph-osd is slow to respond to a request, it will generate log messages complaining about requests that are taking too long. The warning threshold defaults to 30 seconds, and is configurable via the osd op complaint time option. When this happens, the cluster log will receive messages like:
如果某些因素导致 ceph-osd 对别人的请求响应缓慢，它将生成日志，抱怨请求花的时间太长，这个告警阀值是 30 秒。可通过 osd op complaint time 选项配置，这事发生的时候，集群日志系统将收到类似下面的消息：

```
osd.0 192.168.106.220:6800/18813 312 : [WRN] old request osd_op(client.5099.0:790
fatty_26485_object789 [write 0~4096] 2.5e54f643) v4 received at 2012-03-06 15:42:56.054801
currently waiting for sub ops
```

Possible causes include:
可能的起因包括：

- bad disk (check dmesg output)
- kernel file system bug (check dmesg output)
- overloaded cluster (check system load, iostat, etc.)
- ceph-osd bug

3.3.4.1.8 打摆子的 OSD

FLAPPING OSDS
If something is causing OSDs to "flap" (repeatedly getting marked down and then up again), you can force the monitors to stop with:
如果有东西导致 OSD 摆动（反复地被标记为 down，然后又 up），你可以强制监视器停止：

```
ceph osd set noup      # prevent osds from getting marked up
ceph osd set nodown    # prevent osds from getting marked down
```

These flags are recorded in the osdmap structure:

这些标记记录在 osdmap 数据结构里：

```
ceph osd dump | grep flags
flags no-up,no-down
```

You can clear the flags with:
下列命令可清除标记：

```
ceph osd unset noup
ceph osd unset nodown
```

Two other flags are supported, noin and noout, which prevent booting OSDs from being marked in (allocated data) or down ceph-osds from eventually being marked out (regardless of what the current value for mon osd down out interval is).
还支持其它两个标记 noin 和 noout，它们分别可阻止 OSD 被标记为 in、死亡的 ceph-osd 被标记为 out（不管 mon osd down out interval 的值是什么）。

Note that noup, noout, and noout are temporary in the sense that once the flags are cleared, the action they were blocking should occur shortly after. The noin flag, on the other hand, prevents ceph-osds from being marked in on boot, and any daemons that started while the flag was set will remain that way.
注意，noup、noout 和 nodown 从某种意义上说是临时的，一旦标记清除了，它们被阻塞的动作短时间内就会发生；相反，noin 标记阻止 ceph-osd 启动时进入集群，任何设置了此标记的守护进程启动后都维持原样。

3.3.4.2 监视器失败恢复

Recovering from monitor failures

In production clusters, we recommend running the cluster with a minimum of three monitors. The failure of a single monitor should not take down the entire monitor cluster, provided a majority of the monitors remain available. If the majority of nodes are available, the remaining nodes will be able to form a quorum.
在生产集群，我们推荐至少要运行 3 个监视器。这样单个监视器失败不会拖垮整个监视器集群，因为大部分仍可用，这样剩余节点仍能形成法定人数。

When you check your cluster's health, you may notice that a monitor has failed. For example:
检查集群健康状况的时候，也许会看到一个监视器失败了，例如：

```
ceph health
HEALTH_WARN 1 mons down, quorum 0,2
```

For additional detail, you may check the cluster status:
额外详情可检查集群状态：

```
ceph status
HEALTH_WARN 1 mons down, quorum 0,2
mon.b (rank 1) addr 192.168.106.220:6790/0 is down (out of quorum)
```

In most cases, you can simply restart the affected node. For example:
大多情况下，都可以简单地重启相应节点，例如：

```
service ceph -a restart {failed-mon}
```

If there are not enough monitors to form a quorum, the ceph command will block trying to reach the cluster. In this situation, you need to get enough ceph-mon daemons running to form a quorum before doing anything else with the cluster.
如果监视器数量不足以形成法定人数，ceph 命令将拦截你的操作尝试，你得先启动足够的 ceph-mon 守护进程来形成法定人数，才能在集群里做其它的事。

3.3.5 调试和日志记录

Debugging and logging

You may view Ceph log files under /var/log/ceph (the default location).
你可以在默认位置/var/log/ceph 下翻阅 ceph 的日志。

Ceph is still on the leading edge, so you may encounter situations that require using Ceph's debugging and logging. To activate and configure Ceph's debug logging, refer to Ceph Logging and Debugging. For additional logging settings, refer to the Logging and Debugging Config Reference.
ceph 仍在技术前沿，所以你也许会碰到一些状况，需要使用 ceph 的调试和日志功能。激活和配置 ceph 的调试日志，参见日志、调试；额外的日志设置参见日志和调试配置参考。

You can change the logging settings at runtime so that you don't have to stop and restart the cluster. Refer to Ceph Configuration - Runtime Changes for additional details.
可以在运行时更改日志选项，这样你就不必停止、重启集群了，详情参见运行时更改。

Debugging may also require you to track down memory and threading issues. You can run a single daemon, a type of daemon, or the whole cluster with Valgrind. You should only use Valgrind when developing or debugging Ceph. Valgrind is computationally expensive, and will slow down your system otherwise. Valgrind messages are logged to stderr.
调试也许要求你捕捉内存和线程问题，你可以在 Valgrind 下运行单个、一类守护进程、或者整个集群。你应该只在开发、调试期间使用 Valgrind，它需要大量计算，会减慢你的系统。Valgrind 消息记录在标准错误。

3.3.5.1 数据归置

DATA PLACEMENT
Once you have your cluster up and running, you may begin working with data placement. Ceph supports petabyte-scale data storage clusters, with storage pools and placement groups that distribute data across the cluster using Ceph's CRUSH algorithm.
你的集群启动并运行后，你就可以研究数据放置了，ceph 支持 PB 级数据存储集群，是因为存储池和归置组用 CRUSH 算法在集群内分布数据。

3.3.6 数据归置概览

Data placement overview

Ceph stores, replicates and rebalances data objects across a RADOS cluster dynamically. With many different users storing objects in different pools for different purposes on countless OSDs, Ceph operations require some data placement planning. The main data placement planning concepts in Ceph include:
ceph 通过 RADOS 集群动态地存储、复制和重新均衡数据对象。很多不同用户因不同目的把对象存储在不同的存储池里，而它们都坐落于无数的 OSD 之上，所以 ceph 的运营需要些数据归置计划。ceph 的数据归置计划概念主要有：

Pools: Ceph stores data within pools, which are logical groups for storing objects. Pools manage the number of placement groups, the number of replicas, and the ruleset for the pool. To store data in a pool, you must have an authenticated user with permissions for the pool. Ceph can snapshot pools. Future versions of Ceph will support namespaces within pools.
存储池：ceph 在存储池内存储数据，它是对象存储的逻辑组；存储池管理着归置组数量、复制数量、和存储池规则集。要往存储池里存数据，用户必须认证过、且权限合适，存储池可做快照，它未来将支持名称空间功能。

Placement Groups: Ceph maps objects to placement groups (PGs). Placement groups (PGs) are shards or fragments of a logical object pool that place objects as a group into OSDs. Placement groups reduce the amount of per-object metadata when Ceph stores the data in OSDs. A larger number of placement groups (e.g., 100 per OSD) leads to better balancing.
归置组：ceph 把对象映射到归置组（PG），归置组是一系列逻辑对象池的片段，这些对象分组后再存储到 OSD，归置组减少了每对象元数据数量，更多的归置组（如每 OSD 100 个）使得均衡更好。

CRUSH Maps: CRUSH is a big part of what allows Ceph to scale without performance bottlenecks, without limitations to scalability, and without a single point of failure. CRUSH maps provide the physical topology of the cluster to the CRUSH algorithm to determine where the data for an object and its replicas should be stored, and how to do so across failure domains for added data safety among other things.
CRUSH 图：CRUSH 是使 ceph 能伸缩自如而没有性能瓶颈、没有扩展限制、没有单点故障，它为 CRUSH 算法

提供集群的物理拓扑，以此确定一个对象的数据及它的副本应该在哪里、怎样才能越过故障域保证数据安全。

When you initially set up a test cluster, you can use the default values. Once you begin planning for a large Ceph cluster, refer to pools, placement groups and CRUSH for data placement operations. If you find some aspects challenging, Inktank provides excellent premium support for Ceph.
起初安装测试集群的时候，可以使用默认值。但开始规划一个大型 ceph 集群，做数据归置操作的时候会涉及存储池、归置组、和 CRUSH。

3.3.7 存储池

Pools

When you first deploy a cluster without creating a pool, Ceph uses the default pools for storing data. A pool differs from CRUSH's location-based buckets in that a pool doesn't have a single physical location, and a pool provides you with some additional functionality, including:
开始部署集群时没有创建存储池，ceph 则用默认存储池存数据。存储池不同于 CRUSH 基于位置的桶，它没有单独的物理位置，而且存储池提供了一些额外的功能：

Replicas: You can set the desired number of copies/replicas of an object. A typical configuration stores an object and one additional copy (i.e., size = 2), but you can determine the number of copies/replicas.
复制：你可以设置一个对象期望的副本数量。典型配置存储一个对象和一个它的副本（如 size = 2），但你可以更改副本的数量。

Placement Groups: You can set the number of placement groups for the pool. A typical configuration uses approximately 100 placement groups per OSD to provide optimal balancing without using up too many computing resources. When setting up multiple pools, be careful to ensure you set a reasonable number of placement groups for both the pool and the cluster as a whole.
归置组：你可以设置一个存储池的归置组数量。典型配置在每个 OSD 上使用大约 100 个归置组，这样，不用过多计算资源就得到了较优的均衡。设置多个存储池的时候，要注意为这些存储池和集群设置合理的归置组数量。

CRUSH Rules: When you store data in a pool, a CRUSH ruleset mapped to the pool enables CRUSH to identify a rule for the placement of the primary object and object replicas in your cluster. You can create a custom CRUSH rule for your pool.
CRUSH 规则：当你在存储池里存数据的时候，映射到存储池的 CRUSH 规则集使得 CRUSH 确定一条规则，用于集群内主对象的归置和其副本的复制。你可以给存储池定制 CRUSH 规则。

Snapshots: When you create snapshots with ceph osd pool mksnap, you effectively take a snapshot of a particular pool.
快照：你用 ceph osd pool mksnap 创建快照的时候，实际上创建了一小部分存储池的快照。

Set Ownership: You can set a user ID as the owner of a pool.
设置所有者：你可以设置一个用户 ID 为一个存储池的所有者。

To organize data into pools, you can list, create, and remove pools. You can also view the utilization statistics for each pool.
要把数据组织到存储池里，你可以列出、创建、删除存储池，也可以查看每个存储池的利用率。

3.3.7.1 列出存储池

LIST POOLS
To list your cluster's pools, execute:
要列出集群的存储池，命令如下：

```
ceph osd lspools
```

The default pools include:
默认存储池有：

```
data
```

```
metadata
rbd
```

3.3.7.2 创建一个存储池

CREATE A POOL
To create a pool, execute:
要创建一个存储池，执行：

```
ceph osd pool create {pool-name} {pg-num} [{pgp-num}]
```

Where:
参数含义如下：

{pool-name}
Description: The name of the pool. It must be unique.
Type: String
Required: Yes
描述：存储池名称，必须唯一。

{pg-num}
Description: The total number of placement groups for the pool
Type: Integer
Required: No
描述：存储池拥有的归置组总数。

{pgp-num}
Description: The total number of placement groups for placement purposes.
Type: Integer
Required: No
描述：用于归置的归置组总数。

When you create a pool, you should consider setting the number of placement groups.
创建存储池时应该设置归置组数量。

> Important You cannot change the number of placement groups in a pool after you create it.
> 重要：创建存储池后，内中包含的归置组数量不可更改。

See Placement Groups for details on calculating an appropriate number of placement groups for your pool.
如何计算存储池合适的归置组数量请参见归置组。

3.3.7.3 删除一个存储池

DELETE A POOL
To delete a pool, execute:
要删除一个存储池，执行：

```
ceph osd pool delete {pool-name}
```

If you created your own rulesets and rules for a pool you created, you should consider removing them when you no longer need your pool. If you created users with permissions strictly for a pool that no longer exists, you should consider deleting those users too.
如果你给自建的存储池创建了定制的规则集，你不需要存储池时最好删除它。如果你曾严格地创建了用户及其权限给一个存储池，但存储池已不存在，最好也删除那些用户。

3.3.7.4 重命名一个存储池

RENAME A POOL
To rename a pool, execute:

要重命名一个存储池，执行：

```
ceph osd pool rename {current-pool-name} {new-pool-name}
```

If you rename a pool and you have per-pool capabilities for an authenticated user, you must update the user's capabilities (i.e., caps) with the new pool name.
如果重命名了一个存储池，且认证用户有每存储池能力，那你必须用新存储池名字更新用户的能力（如 caps）。

3.3.7.5 显示存储池统计信息

SHOW POOL STATISTICS
To show a pool's utilization statistics, execute:
要查看某存储池的使用统计信息，执行命令：

```
rados df
```

3.3.7.6 拍下存储池快照

MAKE A SNAPSHOT OF A POOL
To make a snapshot of a pool, execute:
要拍下某存储池的快照，执行命令：

```
ceph osd pool mksnap {pool-name} {snap-name}
```

3.3.7.7 删除存储池快照

REMOVE A SNAPSHOT OF A POOL
To remove a snapshot of a pool, execute:
要删除某存储池的一个快照，执行命令：

```
ceph osd pool rmsnap {pool-name} {snap-name}
```

3.3.7.8 设置存储池的值

SET POOL VALUES
To set a value to a pool, execute the following:
要设置一个存储池的选项值，执行命令：

```
ceph osd pool set {pool-name} {key} {value}
```

You may set values for the following keys:
你可以设置下列键的值：

size
Description: Sets the number of replicas for objects in the pool. See Set the Number of Object Replicas for further details.
Type: Integer
描述：设置存储池中对象的副本数，详情参见设置对象副本数。

min_size
Description: Sets the minimum number of replicas required for io. See Set the Number of Object Replicas for further details
Type: Integer
描述：设置 IO 需要的最小副本数，详情参见设置对象副本数。

crash_replay_interval
Description: The number of seconds to allow clients to replay acknowledged, but uncommitted requests.
Type: Integer
描述：允许客户端重放确认而未提交请求的秒数。

pgp_num
Description: The effective number of placement groups to use when calculating data placement.

Type: Integer
Valid Range: Equal to or less than pg_num.
描述：计算数据归置时使用的有效归置组数量。

crush_ruleset
Description: The ruleset to use for mapping object placement in the cluster.
Type: Integer
描述：集群内映射对象归置时使用的规则集。

3.3.7.9 获取存储池的值

GET POOL VALUES
To set a value to a pool, execute the following:
要给一个存储池设置值，执行命令：

```
ceph osd pool get {pool-name} {key}
```

pg_num
Description: The number of placement groups for the pool.
Type: Integer
描述：存储池的归置组数量。

pgp_num
Description: The effective number of placement groups to use when calculating data placement.
Type: Integer
Valid Range: Equal to or less than pg_num.
描述：计算数据归置时使用的归置组有效数量。

3.3.7.10 设置对象副本数

SET THE NUMBER OF OBJECT REPLICAS
To set the number of object replicas, execute the following:
要设置对象副本数，执行命令：

```
ceph osd pool set {poolname} size {num-replicas}
```

For example:
例如：

```
ceph osd pool set data size 3
```

You may execute this command for each pool.
你可以在每个存储池上执行这个命令。

Note, however, that pool size is more of a best-effort setting: an object might accept IOs in degraded mode with fewer than size replicas. To set a minimum number of required replicas for io, you should use the min_size setting.
注意：pool size 是倾向于尽最大努力的设置：一个处于降级模式的对象其副本数小于设置值，但仍可接受 IO 请求。min_size 选项可设置必需给 IO 的最小副本数。

For example:
例如：

```
ceph osd pool set data min_size 2
```

This ensures that no object in the data pool will receive IO with fewer than min_size replicas.
这确保数据存储池里任何副本数小于 min_size 的对象都不会收到 IO 了。

3.3.7.11 获取对象副本数

GET THE NUMBER OF OBJECT REPLICAS

To get the number of object replicas, execute the following:
要获取对象副本数，执行命令：

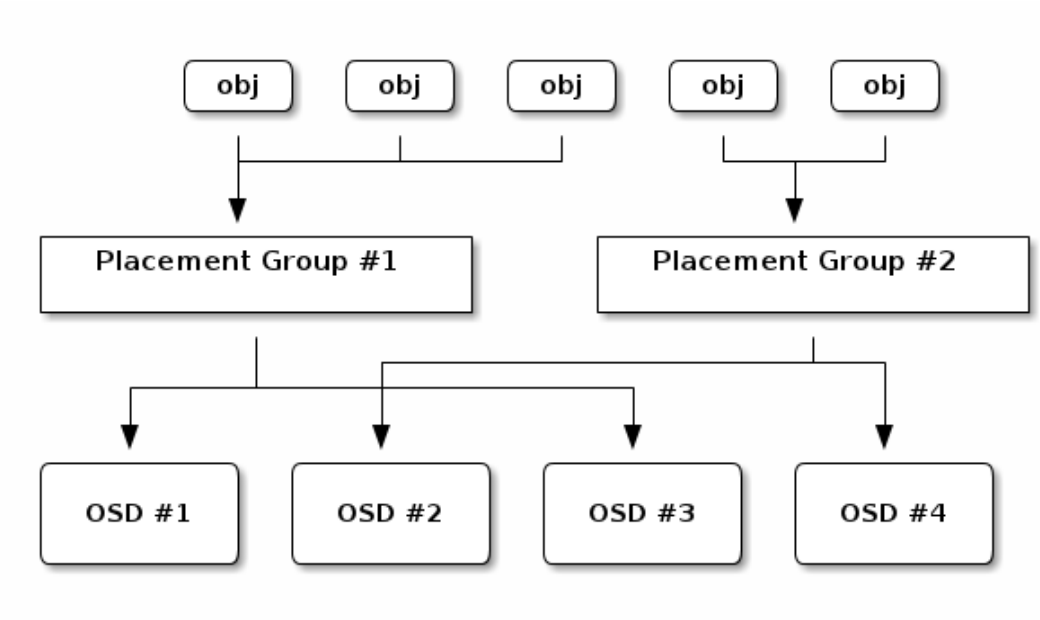
```
ceph osd dump | grep 'rep size'
```

Ceph will list the pools, with the rep size attribute highlighted. By default, Ceph creates two replicas of an object (two copies).
ceph 会列出存储池，且高亮'rep size'属性，它创建了一个对象的两个副本（两份拷贝）。

3.3.8 归置组

Placement groups

A Placement Group (PG) aggregates a series of objects into a group, and maps the group to a series of OSDs. Tracking object placement and object metadata on a per-object basis is computationally expensive–i.e., a system with millions of objects cannot realistically track placement on a per-object basis. Placement groups address this barrier to performance and scalability. Additionally, placement groups reduce the number of processes and the amount of per-object metadata Ceph must track when storing and retrieving data.
一个归置组(PG)把一系列对象汇聚到一组，并且把这个组映射到一系列 OSD。跟踪每个对象的位置和元数据需要大量计算。例如，一个拥有数百万对象的系统，不可能在每对象级追踪位置。归置组可应对这个影响性能和扩展性的问题，另外，归置组减小了 ceph 存储、检索数据时必须追踪的每对象元数据的处理量和尺寸。



Each placement group requires some amount of system resources:
每个归置组都需要一定量系统资源：

Directly: Each PG requires some amount of memory and CPU.
Indirectly: The total number of PGs increases the peering count.
Increasing the number of placement groups reduces the variance in per-OSD load across your cluster. We recommend approximately 50-100 placement groups per OSD to balance out memory and CPU requirements and per-OSD load. For a single pool of objects, you can use the following formula:
直接地：每个 PG 需要一些内存和 CPU；
间接地：PG 总量增加了连接建立数量；
增加 PG 数量能减小集群内每个 OSD 间的变迁，我们推荐每个 OSD 大约 50-100 个归置组，以均衡内存、CPU 需求、和每 OSD 负载。对于单存储池里的对象，你可用下面的公式：

```
            (OSDs * 100)
Total PGs = ------------
              Replicas
```

When using multiple data pools for storing objects, you need to ensure that you balance the number of placement groups per pool with the number of placement groups per OSD so that you arrive at a reasonable total number of placement groups that provides reasonably low variance per OSD without taxing system

resources or making the peering process too slow.
当用了多个数据存储池来存储数据时，你得确保均衡每个存储池的归置组数量、且归置组数量分摊到每个 OSD，这样才能达到较合理的归置组总量，并因此使得每个 OSD 无需耗费过多系统资源或拖慢连接进程就能实现较小变迁。

3.3.8.1 设置归置组数量

SET THE NUMBER OF PLACEMENT GROUPS

To set the number of placement groups in a pool, you must specify the number of placement groups at the time you create the pool.
你必须在创建存储池时设置一个存储池的归置组数量。

See Create a Pool for details.
详情参见创建一个存储池。

3.3.8.2 获取归置组数量

GET THE NUMBER OF PLACEMENT GROUPS

To get the number of placement groups in a pool, execute the following:
要获取一个存储池的归置组数量，执行命令：

```
ceph osd pool get {pool-name} pg_num
```

3.3.8.3 获取归置组统计信息

GET A CLUSTER'S PG STATISTICS
To get the statistics for the placement groups in your cluster, execute the following:
要获取集群里归置组的统计信息，执行命令：

```
ceph pg dump [--format {format}]
```

Valid formats are plain (default) and json.
可用格式有纯文本（默认）和 json。

3.3.8.4 获取卡住的归置组统计信息

GET STATISTICS FOR STUCK PGS

To get the statistics for all placement groups stuck in a specified state, execute the following:
要获取所有卡在某状态的归置组统计信息，执行命令：

```
ceph pg dump_stuck inactive|unclean|stale [--format <format>] [-t|--threshold <seconds>]
```

Inactive Placement groups cannot process reads or writes because they are waiting for an OSD with the most up-to-date data to come up and in.
inactive（不活跃）归置组不能处理读写，因为它们在等待一个有最新数据的 OSD 复活且进入集群。

Unclean Placement groups contain objects that are not replicated the desired number of times. They should be recovering.
unclean（不干净）归置组含有复制数未达到期望数量的对象，它们应该在恢复中。

Stale Placement groups are in an unknown state - the OSDs that host them have not reported to the monitor cluster in a while (configured by mon_osd_report_timeout).
stale（不新鲜）归置组处于未知状态：存储它们的 OSD 有段时间没向监视器报告了（由 mon_osd_report_timeout 配置）。

Valid formats are plain (default) and json. The threshold defines the minimum number of seconds the placement group is stuck before including it in the returned statistics (default 300 seconds).
可用格式有纯文本（默认）和 json。阀值定义的是，归置组被认为卡住前等待的最小时间（默认 300 秒）。

3.3.8.5 获取归置组图

GET A PG MAP

To get the placement group map for a particular placement group, execute the following:
要获取一个具体归置组的归置组图，执行命令：

```
ceph pg map {pg-id}
```

For example:
例如：

```
ceph pg map 1.6c
```

Ceph will return the placement group map, the placement group, and the OSD status:
ceph 将返回归置组图、归置组、和 OSD 状态：

```
osdmap e13 pg 1.6c (1.6c) -> up [1,0] acting [1,0]
```

3.3.8.6 获取一个PG 的统计信息

GET A PGS STATISTICS
To retrieve statistics for a particular placement group, execute the following:
要查看一个具体归置组的统计信息，执行命令：

```
ceph pg {pg-id} query
```

3.3.8.7 洗刷一个归置组

Scrub a placement group
To scrub a placement group, execute the following:
要洗刷一个归置组，执行命令：

```
ceph pg scrub {pg-id}
```

Ceph checks the primary and any replica nodes, generates a catalog of all objects in the placement group and compares them to ensure that no objects are missing or mismatched, and their contents are consistent. Assuming the replicas all match, a final semantic sweep ensures that all of the snapshot-related object metadata is consistent. Errors are reported via logs.
ceph 检查原始的和任何复制节点，生成归置组里所有对象的目录，然后再对比，确保没有对象丢失或不匹配，并且它们的内容一致。

3.3.8.8 恢复丢失的

REVERT LOST
If the cluster has lost one or more objects, and you have decided to abandon the search for the lost data, you must mark the unfound objects as lost.
如果集群丢了一或多个对象，而且必须放弃搜索这些数据，你就要把未找到的对象标记为丢失。

If all possible locations have been queried and objects are still lost, you may have to give up on the lost objects. This is possible given unusual combinations of failures that allow the cluster to learn about writes that were performed before the writes themselves are recovered.
如果所有可能的位置都查询过了，而仍找不到这些对象，你也许得放弃它们了。这可能是罕见的失败组合导致的，集群在写入完成前，未能得知写入是否已执行。

Currently the only supported option is “revert”, which will either roll back to a previous version of the object or (if it was a new object) forget about it entirely. To mark the “unfound” objects as “lost”, execute the following:
当前只支持 revert 选项，它使得回滚到对象的前一个版本（如果它是新对象）或完全忽略它。要把 unfound 对象标记为 lost，执行命令：

```
ceph pg {pg-id} mark_unfound_lost revert
```

Important Use this feature with caution, because it may confuse applications that expect the object(s) to exist.
重要：要谨慎使用，它可能迷惑那些期望对象存在的应用程序。

3.3.9 CRUSH 图

CRUSH MAPS
The CRUSH algorithm determines how to store and retrieve data by computing data storage locations. CRUSH empowers Ceph clients to communicate with OSDs directly rather than through a centralized server or broker. With an algorithmically determined method of storing and retrieving data, Ceph avoids a single point of failure, a performance bottleneck, and a physical limit to its scalability.
CRUSH 算法通过计算数据存储位置来确定如何存储和检索。CRUSH 授权 ceph 客户端直接连接 OSD，而非通过一个中央服务器或经纪人。数据存储、检索算法的使用，使 ceph 避免了单点失败、性能瓶颈、和伸缩的物理限制。

CRUSH requires a map of your cluster, and uses the CRUSH map to pseudo-randomly store and retrieve data in OSDs with a uniform distribution of data across the cluster. For a detailed discussion of CRUSH, see CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data
CRUSH 需要一张集群的地图，且使用 CRUSH 把数据伪随机地存储、检索于整个集群的 OSD 里。CRUSH 的讨论详情参见 CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data。

CRUSH Maps contain a list of OSDs, a list of 'buckets' for aggregating the devices into physical locations, and a list of rules that tell CRUSH how it should replicate data in a Ceph cluster's pools. By reflecting the underlying physical organization of the installation, CRUSH can model—and thereby address—potential sources of correlated device failures. Typical sources include physical proximity, a shared power source, and a shared network. By encoding this information into the cluster map, CRUSH placement policies can separate object replicas across different failure domains while still maintaining the desired distribution. For example, to address the possibility of concurrent failures, it may be desirable to ensure that data replicas are on devices in different shelves, racks, power supplies, controllers, and/or physical locations.
CRUSH 图包含 OSD 列表、把设备汇聚为物理位置的“桶”列表、和指示 CRUSH 如何复制存储池里的数据的规则列表。由于对所安装底层物理组织的表达，CRUSH 能模型化、并因此定位到潜在的相关失败设备源头，典型的源头有物理距离、共享电源、和共享网络，把这些信息编码到集群图里，CRUSH 归置策略可把对象副本分离到不同的失败域，却仍能保持期望的分布。例如，要定位同时失败的可能性，可能希望保证数据复制到的设备位于不同机架、不同托盘、不同电源、不同控制器、甚至不同物理位置。

When you create a configuration file and deploy Ceph with mkcephfs, Ceph generates a default CRUSH map for your configuration. The default CRUSH map is fine for your Ceph sandbox environment. However, when you deploy a large-scale data cluster, you should give significant consideration to developing a custom CRUSH map, because it will help you manage your Ceph cluster, improve performance and ensure data safety.
当你写好配置文件，用 mkcephfs 部署 ceph 后，它生成了一个默认的 CRUSH 图，对于你的沙盒环境来说它很好。然而，部署一个大规模数据集群的时候，应该好好设计自己的 CRUSH 图，因为它帮你管理 ceph 集群、提升性能、和保证数据安全性。

For example, if an OSD goes down, a CRUSH Map can help you can locate the physical data center, room, row and rack of the host with the failed OSD in the event you need to use onsite support or replace hardware.
例如，如果一个 OSD 挂了，CRUSH 图可帮你定位此事件中 OSD 所在主机的物理数据中心、房间、行和机架，据此你可以请求在线支持或替换硬件。

Similarly, CRUSH may help you identify faults more quickly. For example, if all OSDs in a particular rack go down simultaneously, the fault may lie with a network switch or power to the rack or the network switch rather than the OSDs themselves.
类似地，CRUSH 可帮你更快地找出问题。例如，如果一个机架上的所有 OSD 同时挂了，问题可能在于机架的交换机或电源，而非 OSD 本身。

A custom CRUSH map can also help you identify the physical locations where Ceph stores redundant copies of data when the placement group(s) associated with a failed host are in a degraded state.

定制的 CRUSH 图也能在归置组降级时，帮你找出冗余副本所在主机的物理位置。

Inktank provides excellent premium support for developing CRUSH maps.
Inktank 提供优秀的商业支持，帮您开发 CRUSH 图。

Note Lines of code in example boxes may extend past the edge of the box. Please scroll when reading or copying longer examples.
注意：文本框里的代码实例可能超出了边界，读或拷贝时注意滚动。

3.3.9.1 编辑 CRUSH 图

EDITING A CRUSH MAP
To edit an existing CRUSH map:
要编辑现有的 CRUSH 图：

Get the CRUSH Map.
Decompile the CRUSH Map.
Edit at least one of Devices, Buckets and Rules.
Recompile the CRUSH Map.
Set the CRUSH Map.
获取 CRUSH 图；反编译 CRUSH 图；至少编辑一个设备、桶、规则；重编译 CRUSH 图；应用 CRUSH 图。

To activate CRUSH Map rules for a specific pool, identify the common ruleset number for those rules and specify that ruleset number for the pool. See Set Pool Values for details.
要激活 CRUSH 图里某存储池的规则，找到通用规则集编号，然后把它指定到那个规则集。详情参见设置存储池的值。

3.3.9.1.1 获取 CRUSH 图

GET A CRUSH MAP
To get the CRUSH Map for your cluster, execute the following:
要获取集群的 CRUSH 图，执行命令：

```
ceph osd getcrushmap -o {compiled-crushmap-filename}
```

Ceph will output (-o) a compiled CRUSH Map to the filename you specified. Since the CRUSH Map is in a compiled form, you must decompile it first before you can edit it.
ceph 将把 CRUSH 输出(-o)到你指定的文件，由于 CRUSH 图是已编译的，所以编辑前必须先反编译。

3.3.9.1.2 反编译 CRUSH 图

DECOMPILE A CRUSH MAP
To decompile a CRUSH Map, execute the following:
要反编译 CRUSH 图，执行命令：

```
crushtool -d {compiled-crushmap-filename} -o {decompiled-crushmap-filename}
```

Ceph will decompile (-d) the compiled CRUSH map and output (-o) it to the filename you specified.
ceph 将反编译(-d)二进制 CRUSH 图，且输出(-o)到你指定的文件。

3.3.9.1.3 编译 CRUSH 图

COMPILE A CRUSH MAP
To compile a CRUSH Map, execute the following:
要编译 CRUSH 图，执行命令：

```
crushtool -c {decompiled-crush-map-filename} -o {compiled-crush-map-filename}
```

Ceph will store a compiled CRUSH map to the filename you specified.
ceph 将把已编译的 CRUSH 图保存到你指定的文件。

3.3.9.1.4 设置 CRUSH 图

SET A CRUSH MAP
To set the CRUSH Map for your cluster, execute the following:
要把 CRUSH 图应用到集群，执行命令：

```
ceph osd setcrushmap -i  {compiled-crushmap-filename}
```

Ceph will input the compiled CRUSH Map of the filename you specified as the CRUSH Map for the cluster.
ceph 将把你指定的已编译 CRUSH 图输入到集群。

3.3.9.2 CRUSH 图参数

CRUSH MAP PARAMETERS
There are three main sections to a CRUSH Map.
CRUSH 图主要有 3 个主要段落。

Devices consist of any object storage device–i.e., the hard disk corresponding to a ceph-osd daemon.
Buckets consist of a hierarchical aggregation of storage locations (e.g., rows, racks, hosts, etc.) and their assigned weights.
Rules consist of the manner of selecting buckets
devices 由任意对象存储设备组成，如对应一个 ceph-osd 进程的硬盘；
buckets 由分级汇聚的存储位置（如行、机架、主机等）及其权重组成；
rules 由选择桶的方法组成。

3.3.9.2.1 CRUSH 图之设备

CRUSH MAP DEVICES
To map placement groups to OSDs, a CRUSH Map requires a list of OSD devices (i.e., the name of the OSD daemon). The list of devices appears first in the CRUSH Map.
为把归置组映射到 OSD，CRUSH 图需要 OSD 列表（如 OSD 守护进程名称），所以它们首先出现在 CRUSH 图里。

```
#devices
device {num} {osd.name}
```

For example:
例如：

```
#devices
device 0 osd.0
device 1 osd.1
device 2 osd.2
device 3 osd.3
```

As a general rule, an OSD daemon maps to a single disk or to a RAID.
一般来说，一个 OSD 映射到一个单独的硬盘或 RAID。

3.3.9.2.2 CRUSH 图之桶

CRUSH MAP BUCKETS
CRUSH maps support the notion of 'buckets', which may be thought of as nodes that aggregate other buckets into a hierarchy of physical locations, where OSD devices are the leaves of the hierarchy. The following table lists the default types.
CRUSH 图支持 bucket 概念，可以认为是把其他桶汇聚为分级物理位置的节点，OSD 设备是此分级结构的叶子，下表列出了默认类型。

Type	Location | Description	
0	OSD	An OSD daemon (e.g., osd.1, osd.2, etc).
1	Host	A host name containing one or more OSDs.
2	Rack	A computer rack. The default is `unknownrack`.

Type	Location	Description
3	Row	A row in a series of racks.
4	Room	A room containing racks and rows of hosts.
5	Data Center	A physical data center containing rooms.
6	Pool	A data storage pool for storing objects.

> Tip You can remove these types and create your own bucket types.
> 提示：你可以删除这些类型，并创建自己的桶类型。

Ceph's deployment tools generate a CRUSH map that contains a bucket for each host, and a pool named "default," which is useful for the default data, metadata and rbd pools. The remaining bucket types provide a means for storing information about the physical location of nodes/buckets, which makes cluster administration much easier when OSDs, hosts, or network hardware malfunction and the administrator needs access to physical hardware.
ceph 部署工具生成的 CRUSH 图包含：每主机一个桶、名为 default 的存储池（对默认数据、元数据、rbd 存储池有用），其余桶类型提供的存储信息是关于节点、桶的物理位置的，它简化了 OSD、主机、网络硬件故障时集群管理员处理的步骤。

A bucket has a type, a unique name (string), a unique ID expressed as a negative integer, a weight relative to the total capacity/capability of its item(s), the bucket algorithm (straw by default), and the hash (0 by default, reflecting CRUSH Hash rjenkins1). A bucket may have one or more items. The items may consist of other buckets or OSDs. Items may have a weight that reflects the relative weight of the item.
一个桶有类型、唯一的名字（字符串）、表示为负数的唯一 ID、和能力相关的权重、桶算法（默认微不足道）、和哈希（默认 0，反映 CRUSH Hash rjenkins1）。一个桶可能有一或多个条目。条目由其他桶或 OSD 组成，也可以有反映条目相对权重的权重。

```
[bucket-type] [bucket-name] {
        id [a unique negative numeric ID]
        weight [the relative capacity/capability of the item(s)]
        alg [the bucket type: uniform | list | tree | straw ]
        hash [the hash type: 0 by default]
        item [item-name] weight [weight]
}
```

The following example illustrates how you can use buckets to aggregate a pool and physical locations like a datacenter, a room, a rack and a row.
下例阐述了你该如何用桶汇聚存储池和物理位置，像数据中心、房间、机架、机架行。

```
host ceph-osd-server-1 {
        id -17
        alg straw
        hash 0
        item osd.0 weight 1.00
        item osd.1 weight 1.00
}

row rack-1-row-1 {
        id -16
        alg straw
        hash 0
        item ceph-osd-server-1 2.00
}

rack rack-3 {
        id -15
        alg straw
        hash 0
        item rack-3-row-1 weight 2.00
        item rack-3-row-2 weight 2.00
        item rack-3-row-3 weight 2.00
        item rack-3-row-4 weight 2.00
        item rack-3-row-5 weight 2.00
}

rack rack-2 {
        id -14
```

```
        alg straw
        hash 0
        item rack-2-row-1 weight 2.00
        item rack-2-row-2 weight 2.00
        item rack-2-row-3 weight 2.00
        item rack-2-row-4 weight 2.00
        item rack-2-row-5 weight 2.00
}

rack rack-1 {
        id -13
        alg straw
        hash 0
        item rack-1-row-1 weight 2.00
        item rack-1-row-2 weight 2.00
        item rack-1-row-3 weight 2.00
        item rack-1-row-4 weight 2.00
        item rack-1-row-5 weight 2.00
}

room server-room-1 {
        id -12
        alg straw
        hash 0
        item rack-1 weight 10.00
        item rack-2 weight 10.00
        item rack-3 weight 10.00
}

datacenter dc-1 {
        id -11
        alg straw
        hash 0
        item server-room-1 weight 30.00
        item server-room-2 weight 30.00
}

pool data {
        id -10
        alg straw
        hash 0
        item dc-1 weight 60.00
        item dc-2 weight 60.00
}
```

3.3.9.2.3 CRUSH 图之规则

CRUSH MAP RULES

CRUSH maps support the notion of 'CRUSH rules', which are the rules that determine data placement for a pool. For large clusters, you will likely create many pools where each pool may have its own CRUSH ruleset and rules. The default CRUSH map has a rule for each pool, and one ruleset assigned to each of the default pools, which include:

CRUSH 图支持 CRUSH 规则概念，用以确定一个存储池里数据的归置。对大型集群来说，你可能创建很多存储池，且每个存储池都有它自己的 CRUSH 规则集和规则。默认的 CRUSH 图里，每个存储池有一条规则、一个规则集被分配到每个默认存储池，它们有：

- data
- metadata
- rbd

Note In most cases, you will not need to modify the default rules. When you create a new pool, its default ruleset is 0.

注意：大多数情况下，你都不需要修改默认规则。新创建存储池的默认规则集是 0。

A rule takes the following form:

规则格式如下：

```
rule [rulename] {
        ruleset [ruleset]
        type [type]
        min_size [min-size]
        max_size [max-size]
```

```
        step [step]
}
```

ruleset
Description: A means of classifying a rule as belonging to a set of rules. Activated by setting the ruleset in a pool.
Purpose: A component of the rule mask.
Type: Integer
Required: Yes
Default: 0
描述：区分一条规则属于某个规则集合的手段。在存储池里设置 ruleset 后激活。

type
Description: Describes a rule for either a hard disk (replicated) or a RAID.
Purpose: A component of the rule mask.
Type: String
Required: Yes
Default: replicated
Valid Values: Currently only replicated
描述：为硬盘（复制的）或 RAID 写一条规则。

min_size
Description: If a placement group makes fewer replicas than this number, CRUSH will NOT select this rule.
Type: Integer
Purpose: A component of the rule mask.
Required: Yes
Default: 1
描述：如果一个归置组副本数小于此数，CRUSH 将不应用此规则。

max_size
Description: If a placement group makes more replicas than this number, CRUSH will NOT select this rule.
Type: Integer
Purpose: A component of the rule mask.
Required: Yes
Default: 10
描述：如果一个归置组副本数大于此数，CRUSH 将不应用此规则。

step take {bucket}
Description: Takes a bucket name, and begins iterating down the tree.
Purpose: A component of the rule.
Required: Yes
Example: step take data
描述：选取桶名并类推到树底。

step choose firstn {num} type {bucket-type}
Description: Selects the number of buckets of the given type. Where N is the number of options available, if {num} > 0 && < N, choose that many buckets; if {num} < 0, it means N - {num}; and, if {num} == 0, choose N buckets (all available).
Purpose: A component of the rule.
Prerequisite: Follows step take or step choose.
Example: step choose firstn 1 type row
描述：选取指定类型的桶数量，这里 N 是可用选择的数量，如果{num}>0 && <N 就选择那么多的桶；如果{num}<0 它意为 N-{num}；如果{num}==0 选择 N 个桶（所有可用的）。

step emit
Description: Outputs the current value and empties the stack. Typically used at the end of a rule, but may also be used to from different trees in the same rule.
Purpose: A component of the rule.
Prerequisite: Follows step choose.

Example: step emit
描述：输出当前值并清空堆栈，通常用于规则末尾，也适用于相同规则应用到不同树的情况。

Important To activate one or more rules with a common ruleset number to a pool, set the ruleset number to the pool.
重要：要把规则集编号设置到存储池，才能用一个通用规则集编号激活一或多条规则。

3.3.9.3 增加/移动OSD

ADD/MOVE AN OSD
To add or move an OSD in the CRUSH map of a running cluster, execute the following:
要增加或删除在线集群里的 OSD 对应的 CRUSH 图条目，执行命令：

```
ceph osd crush set {id} {name} {weight} pool={pool-name}  [{bucket-type}={bucket-name} ...]
```

Where:

id
Description: The numeric ID of the OSD.
Type: Integer
Required: Yes
Example: 0
描述：OSD 的数字标识符。

name
Description: The full name of the OSD.
Type: String
Required: Yes
Example: osd.0
描述：OSD 的全名。

weight
Description: The CRUSH weight for the OSD.
Type: Double
Required: Yes
Example: 2.0
描述：OSD 的 CRUSH 权重。

pool
Description: By default, the CRUSH hierarchy contains the pool default as its root.
Type: Key/value pair.
Required: Yes
Example: pool=default
描述：默认情况下，CRUSH 分级结构的根是 default 存储池。

bucket-type
Description: You may specify the OSD's location in the CRUSH hierarchy.
Type: Key/value pairs.
Required: No
Example: datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1
描述：定义 OSD 在 CRUSH 分级结构中的位置。

The following example adds osd.0 to the hierarchy, or moves the OSD from a previous location.
下例把 osd.0 添加到分级结构里、或者说从前一个位置挪动一下。

```
ceph osd crush set 0 osd.0 1.0 pool=data datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1
```

3.3.9.4 调整OSD 的CRUSH 权重

ADJUST AN OSD'S CRUSH WEIGHT

To adjust an OSD's crush weight in the CRUSH map of a running cluster, execute the following:
要调整在线集群中 OSD 的 CRUSH 权重，执行命令：

```
ceph osd crush reweight {name} {weight}
```

Where:

name
Description: The full name of the OSD.
Type: String
Required: Yes
Example: osd.0
描述：OSD 的全名。

weight
Description: The CRUSH weight for the OSD.
Type: Double
Required: Yes
Example: 2.0
描述：OSD 的 CRUSH 权重。

3.3.9.5 删除OSD

REMOVE AN OSD
To remove an OSD from the CRUSH map of a running cluster, execute the following:
要从在线集群里把 OSD 踢出 CRUSH 图，执行命令：

```
ceph osd crush remove {name}
```

Where:

name
Description: The full name of the OSD.
Type: String
Required: Yes
Example: osd.0
描述：OSD 全名。

3.3.9.6 移动桶

MOVE A BUCKET
To move a bucket to a different location or position in the CRUSH map hierarchy, execute the following:
要把一个桶挪动到 CRUSH 图里的不同位置，执行命令：

```
ceph osd crush move {bucket-name} {bucket-type}={bucket-name}, [...]
```

Where:

bucket-name
Description: The name of the bucket to move/reposition.
Type: String
Required: Yes
Example: foo-bar-1
描述：要移动到的桶名。

bucket-type
Description: You may specify the bucket's location in the CRUSH hierarchy.
Type: Key/value pairs.
Required: No
Example: datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1
描述：你可以指定桶在 CRUSH 分级结构里的位置。

3.3.9.7 可调选项

TUNABLES
New in version 0.48.
0.48 时加入。

There are several magic numbers that were used in the original CRUSH implementation that have proven to be poor choices. To support the transition away from them, newer versions of CRUSH (starting with the v0.48 argonaut series) allow the values to be adjusted or tuned.
在 CRUSH 最初实现时加入的几个幻数，现在看来已成问题。作为过渡方法，较新版的 CRUSH（从 0.48 起）允许调整这些值。

Clusters running recent Ceph releases support using the tunable values in the CRUSH maps. However, older clients and daemons will not correctly interact with clusters using the "tuned" CRUSH maps. To detect this situation, there is now a feature bit CRUSH_TUNABLES (value 0x40000) to reflect support for tunables.
最近发布的 ceph 允许 CRUSH 图里的值可调，然而老客户端和守护进程不会正确地和调整过的 CRUSH 图交互，为应对这种情况，现在多了个功能位（值 0x40000）来反映是否支持可调值。

If the OSDMap currently used by the ceph-mon or ceph-osd daemon has non-legacy values, it will require the CRUSH_TUNABLES feature bit from clients and daemons who connect to it. This means that old clients will not be able to connect.
如果 ceph-mon 或 ceph-osd 进程现在用的 OSDMap 有非遗留值，它将要求连接它的客户端和守护进程有 CRUSH_TUNABLES 功能位。

At some future point in time, newly created clusters will have improved default values for the tunables. This is a matter of waiting until the support has been present in the Linux kernel clients long enough to make this a painless transition for most users.
将来，新建集群的可调值其默认值会更好。这要等到此功能进入内核客户端的时间足够长，对大多数用户来说已是无痛的过渡。

3.3.9.7.1 遗留值的影响

IMPACT OF LEGACY VALUES
The legacy values result in several misbehaviors:
遗留值导致几个不当行为：

For hiearchies with a small number of devices in the leaf buckets, some PGs map to fewer than the desired number of replicas. This commonly happens for hiearchies with "host" nodes with a small number (1-3) of OSDs nested beneath each one.
如果分级结构的支部只有少量设备，一些 PG 的副本数小于期望值，这通常出现在一些子结构里，host 节点下少数 OSD 嵌套到了其他 OSD 里。

For large clusters, some small percentages of PGs map to less than the desired number of OSDs. This is more prevalent when there are several layers of the hierarchy (e.g., row, rack, host, osd).
When some OSDs are marked out, the data tends to get redistributed to nearby OSDs instead of across the entire hierarchy.
大型集群里，小部分 PG 映射到的 OSD 数目小于期望值，有多层结构（如：机架行、机架、主机、OSD）时这种情况更普遍。当一些 OSD 标记为 out 时，数据倾向于重分布到附近 OSD 而非整个分级结构。

3.3.9.7.2 哪个客户端版本支持可调参数

WHICH CLIENT VERSIONS SUPPORT TUNABLES
argonaut series, v0.48.1 or later
v0.49 or later
Linux kernel version v3.5 or later (for the file system and RBD kernel clients)
argonaut 系列，v0.48.1 及更高；v0.49 及更高；Linux 内核版本在 v3.5 及更高（文件系统和 RBD 内核客户端）。

3.3.9.7.3 一些要点

A FEW IMPORTANT POINTS
Adjusting these values will result in the shift of some PGs between storage nodes. If the Ceph cluster is already storing a lot of data, be prepared for some fraction of the data to move.
The ceph-osd and ceph-mon daemons will start requiring the CRUSH_TUNABLES feature of new connections as soon as they get the updated map. However, already-connected clients are effectively grandfathered in, and will misbehave if they do not support the new feature.
调整这些值将使一些 PG 在存储节点间移位，如果 ceph 集群已经存储了大量数据，可能移动一部分数据。一旦 ceph-osd 和 ceph-mon 收到了更新的地图，它们对新建连接就开始要求 CRUSH_TUNABLES 功能了，然而，之前已经连接的客户端如果不支持新功能将行为失常。

If the CRUSH tunables are set to non-legacy values and then later changed back to the defult values, ceph-osd daemons will not be required to support the feature. However, the OSD peering process requires examining and understanding old maps. Therefore, you should not run old (pre-v0.48) versions of the ceph-osd daemon if the cluster has previosly used non-legacy CRUSH values, even if the latest version of the map has been switched back to using the legacy defaults.
如果 CRUSH 可调值更改过、然后又改回了默认值，ceph-osd 守护进程将不要求支持此功能，然而，OSD 连接建立进程要能检查和理解旧地图。因此，集群如果用过非默认 CRUSH 值就不应该再运行版本小于 0.48.1 的 ceph-osd，即使最新版地图已经回滚到了遗留默认值。

3.3.9.7.4 调整 CRUSH

TUNING CRUSH
If you can ensure that all clients are running recent code, you can adjust the tunables by extracting the CRUSH map, modifying the values, and reinjecting it into the cluster.
如果你能保证所有客户端都运行最新代码，你可以这样调整可调值：从集群抽取 CRUSH 图、修改值、重注入。

Extract the latest CRUSH map:
抽取最新 CRUSH 图：

```
ceph osd getcrushmap -o /tmp/crush
```

Adjust tunables. These values appear to offer the best behavior for both large and small clusters we tested with. You will need to additionally specify the --enable-unsafe-tunables argument to crushtool for this to work. Please use this option with extreme care.:
调整可调参数。这些值在我们测试过的大、小型集群上都有最佳表现。在极端情况下，你需要给 crushtool 额外指定--enable-unsafe-tunables 参数才行：

```
crushtool -i /tmp/crush --set-choose-local-tries 0 --set-choose-local-fallback-tries 0 --set-
choose-total-tries 50 -o /tmp/crush.new
```

Reinject modified map:
重注入修改的地图：

```
ceph osd setcrushmap -i /tmp/crush.new
```

3.3.9.7.5 遗留值

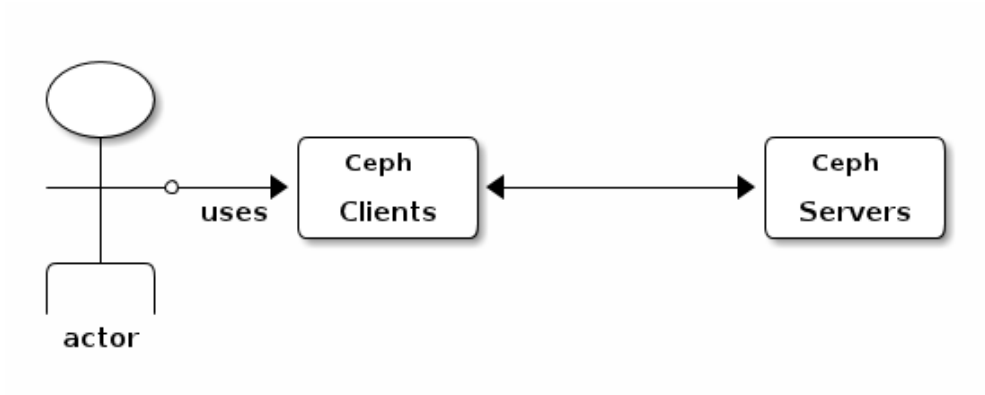
LEGACY VALUES
For reference, the legacy values for the CRUSH tunables can be set with:
CRUSH 可调参数的遗留值可以用下面命令设置：

```
crushtool -i /tmp/crush --set-choose-local-tries 2 --set-choose-local-fallback-tries 5 --set-
choose-total-tries 19 -o /tmp/crush.legacy
```

Again, the special --enable-unsafe-tunables option is required. Further, as noted above, be careful running old versions of the ceph-osd daemon after reverting to legacy values as the feature bit is not perfectly enforced.
再次申明，--enable-unsafe-tunables 是必需的，而且前面也提到了，回退到遗留值后慎用旧版 ceph-osd 进程，因为此功能位不是完全强制的。

3.3.10 ceph 认证及授权

Ceph authentication & authorization
Ceph is a distributed storage system where a typical deployment involves a relatively small quorum of monitors, scores of metadata servers (MDSs) and many thousands of OSD daemons operating across many hosts/nodes–representing the server portion of the Ceph object store. Ceph clients such as CephFS, Ceph block device and Ceph Gateway interact with the Ceph object store. All Ceph object store clients use the librados library to interact with the Ceph object store. The following diagram illustrates an abstract client/server technology stack.
ceph 是一个分布式存储系统，其典型部署包含相对少量的监视器、许多元数据服务器（MDS）和数千 OSD 守护进程，它们运行于很多主机或节点，共同构成了 ceph 对象存储的服务器部分；ceph 客户端如 CephFS、Ceph 块设备和 Ceph 网关与 ceph 对象存储交互，所有客户端都用 librados 库和 ceph 对象存储交互，下面的图抽象地展示了客户端/服务器技术。



Users are either individuals or system actors such as applications, which use Ceph clients to interact with Ceph server daemons.
用户可以是个人或系统角色，像应用程序，它们用 ceph 客户端和 ceph 服务器守护进程交互。

For additional information, see our Cephx Guide and ceph-authtool manpage.
更多信息参见 Cephx Guide 和 ceph-authtool 手册。

3.3.10.1 ceph 认证（cephx）

CEPH AUTHENTICATION (CEPHX)
Cryptographic authentication has some computational costs, though they should generally be quite low. If the network environment connecting your client and server hosts is very safe and you cannot afford authentication, you can use a Ceph option to turn it off. This is not generally recommended, but should you need to do so, details can be found in the Disable Cephx section.
加密认证要耗费一定计算资源，但通常很低。如果您的客户端和服务器网络环境相当安全，而且认证的负面效应更大，你可以关闭它，通常不推荐您这么做，但必要时可以。详情参见 Disable Cephx。

> Important Remember, if you disable authentication, you are at risk of a man-in-the-middle attack altering your client/server messages, which could lead to disastrous security effects.
> 重要：记住，如果禁用了认证，就会有篡改客户端/服务器消息这样的中间人攻击风险，这会导致灾难性后果。

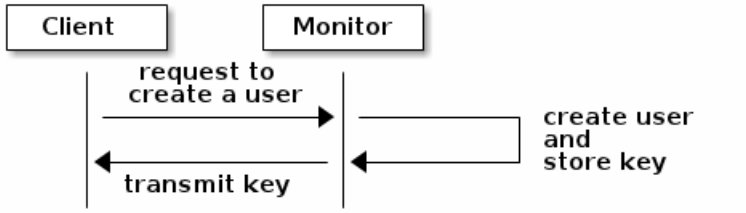
A key scalability feature of Ceph is to avoid a centralized interface to the Ceph object store, which means that Ceph clients must be able to interact with OSDs directly. To protect data, Ceph provides its cephx authentication system, which authenticates users operating Ceph clients. The cephx protocol operates in a manner with behavior similar to Kerberos.
ceph 一个主要伸缩功能就是避免了对象存储的中央接口，这就要求 ceph 客户端能直接和 OSD 交互。Ceph 通过 cephx 认证系统保护数据，它也认证运行 ceph 客户端的用户，cephx 协议运行机制类似 Kerberos。

A user/actor invokes a Ceph client to contact a monitor. Unlike Kerberos, each monitor can authenticate users and distribute keys, so there is no single point of failure or bottleneck when using cephx. The monitor returns

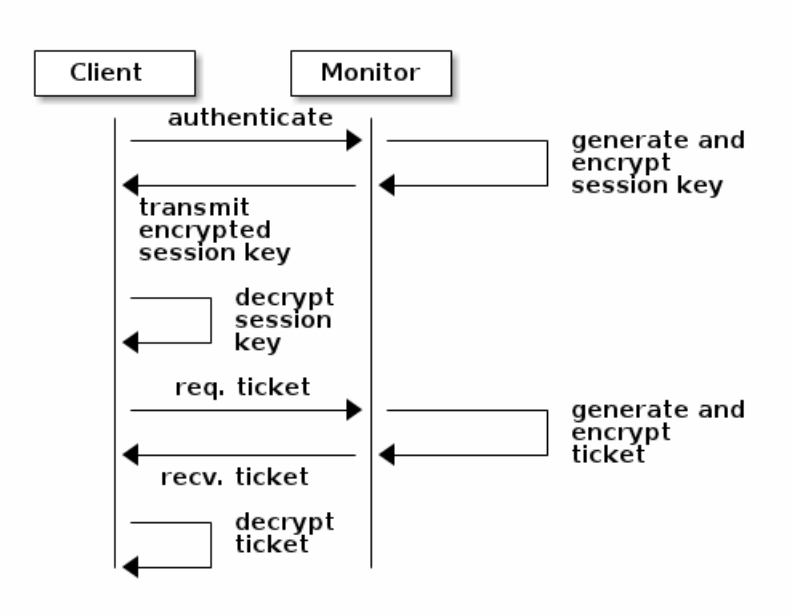
an authentication data structure similar to a Kerberos ticket that contains a session key for use in obtaining Ceph services. This session key is itself encrypted with the user's permanent secret key, so that only the user can request services from the Ceph monitor(s). The client then uses the session key to request its desired services from the monitor, and the monitor provides the client with a ticket that will authenticate the client to the OSDs that actually handle data. Ceph monitors and OSDs share a secret, so the client can use the ticket provided by the monitor with any OSD or metadata server in the cluster. Like Kerberos, cephx tickets expire, so an attacker cannot use an expired ticket or session key obtained surreptitiously. This form of authentication will prevent attackers with access to the communications medium from either creating bogus messages under another user's identity or altering another user's legitimate messages, as long as the user's secret key is not divulged before it expires.
用户/参与者通过调用 ceph 客户端来联系监视器，不像 Kerberos，每个监视器都能认证用户、发布密钥，所以使用 cephx 时不会有单点故障或瓶颈。监视器返回一个类似 Kerberos 票据的认证数据结构，它包含一个可用于获取 ceph 服务的会话密钥，会话密钥是用户的永久私钥自加密过的，只有此用户能从 ceph 监视器请求服务。客户端用会话密钥向监视器请求需要的服务，然后监视器给客户端一个凭证用以向实际持有数据的 OSD 认证。ceph 的监视器和 OSD 共享相同的密钥，所以集群内任何 OSD 或元数据服务器都认可客户端从监视器获取的凭证，像 Kerberos 一样 cephx 凭证也会过期，以使攻击者不能用暗中得到的过期凭证或会话密钥。只要用户的私钥过期前没有泄露，这种认证形式就可防止中间线路攻击者以别人的 ID 发送垃圾消息、或修改用户的正常消息。

To use cephx, an administrator must set up users first. In the following diagram, the client.admin user invokes ceph auth get-or-create-key from the command line to generate a username and secret key. Ceph's auth subsystem generates the username and key, stores a copy with the monitor(s) and transmits the user's secret back to the client.admin user. This means that the client and the monitor share a secret key.
要使用 cephx，管理员必须先设置好用户。在下面的图解里，client.admin 用户从命令行调用 ceph auth get-or-create-key 来生成一个用户及其密钥，ceph 的认证子系统生成了用户名和密钥、副本存到监视器然后把此用户的密钥回传给 client.admin 用户，也就是说客户端和监视器共享着相同的密钥。

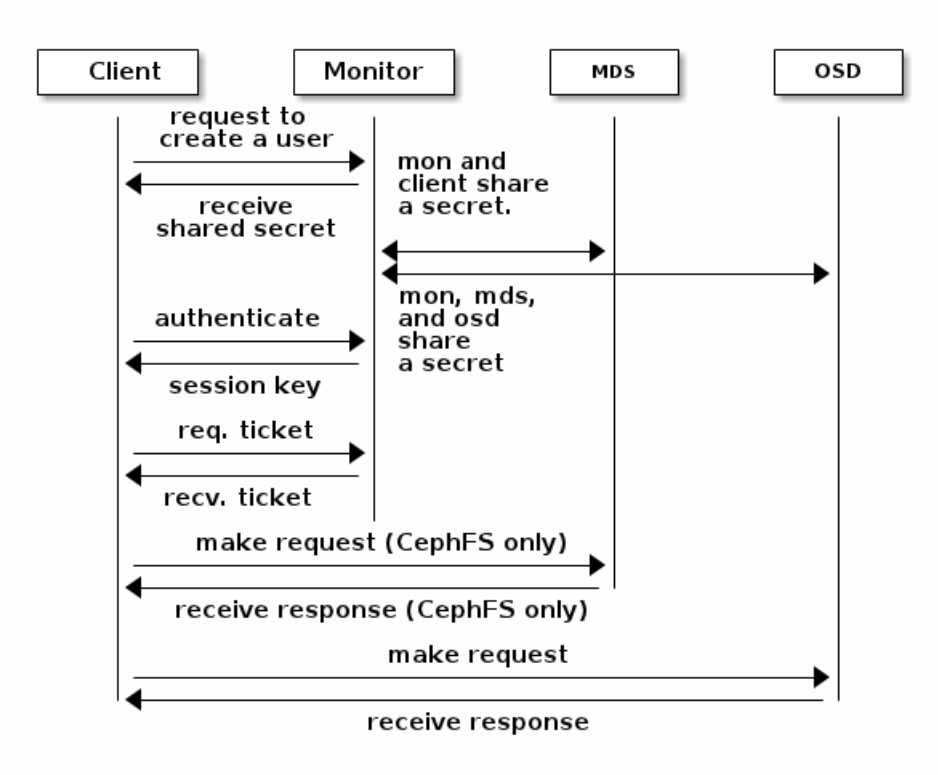
Note The client.admin user must provide the user ID and secret key to the user in a secure manner.
注意：client.admin 用户必须以安全方式把此用户 ID 和密钥交给用户。



To authenticate with the monitor, the client passes in the user name to the monitor, and the monitor generates a session key and encrypts it with the secret key associated to the user name. Then, the monitor transmits the encrypted ticket back to the client. The client then decrypts the payload with the shared secret key to retrieve the session key. The session key identifies the user for the current session. The client then requests a ticket on behalf of the user signed by the session key. The monitor generates a ticket, encrypts it with the user's secret key and transmits it back to the client. The client decrypts the ticket and uses it to sign requests to OSDs and metadata servers throughout the cluster.
要和监视器认证，客户端得把用户名传给监视器，然后监视器生成一个会话密钥、并且用此用户的密钥加密它，然后把加密的凭证回传给客户端，客户端用共享密钥解密载荷就可获取会话密钥。会话密钥在当前会话中标识了此用户，客户端再用此会话密钥签署过的用户名请求一个凭证，监视器生成一个凭证、用客户端的密钥加密它，然后回传给客户端，客户端解密此凭证，然后用它签署连接集群内 OSD 和元数据服务器的请求。



The cephx protocol authenticates ongoing communications between the client machine and the Ceph servers. Each message sent between a client and server, subsequent to the initial authentication, is signed using a ticket that the monitors, OSDs and metadata servers can verify with their shared secret.
cephx 协议认证客户端机器和 ceph 服务器间正在进行的通讯，二者间认证完成后的每条消息都用凭证签署过，监视器、OSD、元数据服务器可用共享的密钥来校验。



The protection offered by this authentication is between the Ceph client and the Ceph server hosts. The authentication is not extended beyond the Ceph client. If the user accesses the Ceph client from a remote host, Ceph authentication is not applied to the connection between the user's host and the client host.
认证提供的保护位于 ceph 客户端和服务器间，没有扩展到 ceph 客户端之外。如果用户从远程主机访问 ceph 客户端，ceph 认证就不管用了，它不会影响到用户主机和客户端主机间的通讯。

3.3.10.2 ceph 授权（能力）

CEPH AUTHORIZATION (CAPS)
Ceph uses the term "capabilities" (caps) to describe authorizing an authenticated user to exercise the functionality of the monitors, OSDs and metadata servers. Capabilities can also restrict access to data within one or more pools.

ceph 用能力(caps)来描述给认证用户的授权，这样才能使用监视器、OSD、和元数据服务器的功能。能力也能限制到一或多个存储池的访问。

Note Ceph uses the capabilities discussed here for setting up and controlling access between various Ceph client and server instances, and are relevant regardless of what type of client accesses the Ceph object store. CephFS uses a different type of capability for files and directories internal to the CephFS filesystem. CephFS filesystem access controls are relevant to CephFS, but not block devices or the RESTful gateway.
注意：ceph 使用能力来设置、控制 ceph 客户端和服务器例程间的相互访问，并且不管哪种客户端访问 ceph 对象存储。CephFS 内部给文件和目录用了不同于 CephFS 文件系统的另外一种能力，CephFS 访问控制和 CephFS 有关，但不是块设备或 RESTful 网关。

A Ceph client.admin user sets a user's capabilities when creating the user.
ceph 的 client.admin 用户在创建用户时设置了用户的能力。

allow
Description: Precedes access settings for a daemon. Implies rw for MDS only.
Example: ceph-authtool -n client.foo --cap mds 'allow'
描述：在守护进程的访问设置之前，仅对 MDS 隐含 rw。

r
Description: Gives the user read access. Required with monitors to retrieve the CRUSH map.
Example: ceph-authtool -n client.foo --cap mon 'allow r'
描述：授予用户读权限，监视器需要它才能搜刮 CRUSH 图。

w
Description: Gives the user write access to objects.
Example: ceph-authtool -n client.foo --cap osd 'allow w'
描述：授予用户写对象的权限。

x
Description: Gives the user the capability to call class methods (i.e., both read and write).
Example: ceph-authtool -n client.foo --cap osd 'allow x'
描述：授予用户调用类方法的能力，如同时有读和写。

class-read
Descriptions: Gives the user the capability to call class read methods. Subset of x.
Example: ceph-authtool -n client.foo --cap osd 'allow class-read'
描述：授予用户调用类读取方法的能力，x 的子集。

class-write
Description: Gives the user the capability to call class write methods. Subset of x.
Example: ceph-authtool -n client.foo --cap osd 'allow class-write'
描述：授予用户调用类写入方法的能力，x 的子集。

*
Description: Gives the user read, write and execute permissions for a particular daemon/pool, and the ability to execute admin commands.
Example: ceph-authtool -n client.foo --cap osd 'allow *'
描述：授权用户读、写和执行某守护进程/存储池，且允许执行管理命令。

When setting capabilities for a user, Ceph also supports restricting the capabilities to a particular pool. This means you can have full access to some pools, and restricted (or no) access to other pools for the same user. For example:
给用户设置能力的时候，ceph 也支持把能力限制于某存储池。意思是你可以完全地访问一些存储池、访问其他存储池却是受限的（或未受限）。

```
ceph-authtool -n client.foo --cap osd 'allow rwx' pool=customer-pool
```

3.3.10.3 cephx 的局限性

CEPHX LIMITATIONS

The cephx protocol authenticates Ceph clients and servers to each other. It is not intended to handle authentication of human users or application programs run on their behalf. If that effect is required to handle your access control needs, you must have another mechanism, which is likely to be specific to the front end used to access the Ceph object store. This other mechanism has the role of ensuring that only acceptable users and programs are able to run on the machine that Ceph will permit to access its object store.
cephx 协议提供 ceph 客户端和服务器间的相互认证，并没打算认证人类用户或者应用程序。如果有访问控制需求，那必须用另外一种机制，它对于前端用户访问 ceph 对象存储可能是特定的，其任务是确保只有此机器上可接受的用户和程序才能访问 ceph 的对象存储。

The keys used to authenticate Ceph clients and servers are typically stored in a plain text file with appropriate permissions in a trusted host.
用于认证 ceph 客户端和服务器的密钥通常以纯文本存储在权限合适的文件里，此文件保存于可信主机上。

> Important Storing keys in plaintext files has security shortcomings, but they are difficult to avoid, given the basic authentication methods Ceph uses in the background. Those setting up Ceph systems should be aware of these shortcomings.
> 重要：密钥存储为纯文本文件有安全缺陷，但很难避免，它给了 ceph 可用的基本认证方法，设置 ceph 时应该注意这些缺陷。

In particular, arbitrary user machines, especially portable machines, should not be configured to interact directly with Ceph, since that mode of use would require the storage of a plaintext authentication key on an insecure machine. Anyone who stole that machine or obtained surreptitious access to it could obtain the key that will allow them to authenticate their own machines to Ceph.
尤其是任意用户、特别是移动机器不应该和 ceph 直接交互，因为这种用法要求把明文认证密钥存储在不安全的机器上，这些机器的丢失、或盗用将泄露可访问 ceph 集群的密钥。

Rather than permitting potentially insecure machines to access a Ceph object store directly, users should be required to sign in to a trusted machine in your environment using a method that provides sufficient security for your purposes. That trusted machine will store the plaintext Ceph keys for the human users. A future version of Ceph may address these particular authentication issues more fully.
相比于允许潜在的欠安全机器直接访问 ceph 对象存储，应该要求用户先登录安全有保障的可信机器，这台可信机器会给人们存储明文密钥。未来的 ceph 版本也许会更彻底地解决这些特殊认证问题。

At the moment, none of the Ceph authentication protocols provide secrecy for messages in transit. Thus, an eavesdropper on the wire can hear and understand all data sent between clients and servers in Ceph, even if he cannot create or alter them. Further, Ceph does not include options to encrypt user data in the object store. Users can hand-encrypt and store their own data in the Ceph object store, of course, but Ceph provides no features to perform object encryption itself. Those storing sensitive data in Ceph should consider encrypting their data before providing it to the Ceph system.
当前，没有任何 ceph 认证协议保证传送中消息的私密性。所以，即使物理线路窃听者不能创建用户或修改它们，但可以听到、并理解客户端和服务器间发送过的所有数据。此外，ceph 没有可加密用户数据的选项，当然，用户可以手动加密、然后把它们存在对象库里，但 ceph 没有自己加密对象的功能。在 ceph 里存储敏感数据的用户应该考虑存入 ceph 集群前先加密。

3.3.11 cephx 手册

cephx guide

Ceph provides two authentication modes:
ceph 提供了两种认证模式：

None: Any user can access data without authentication.
Cephx: Ceph requires user authentication in a manner similar to Kerberos.
none：任何用户无需认证就能访问数据；

cephx：ceph 要求用户认证，机制类似 Kerberos。

If you disable cephx, you do not need to generate keys using the procedures described here. If you re-enable cephx and have already generated keys, you do not need to generate the keys again.
如果你要禁用 cephx，就不需要按前述生成密钥；如果你重新启用 cephx 且已然生成了密钥，也无需再次生成。

For additional information, see our Cephx Intro and ceph-authtool manpage.
额外信息要参见 Cephx Intro 和 ceph-authtool 手册。

3.3.11.1 配置 cephx

CONFIGURING CEPHX

There are several important procedures you must follow to enable the cephx protocol for your Ceph cluster and its daemons. First, you must generate a secret key for the default client.admin user so the administrator can execute Ceph commands. Second, you must generate a monitor secret key and distribute it to all monitors in the cluster. Finally, you can follow the remaining steps in Enabling Cephx to enable authentication.
要为集群及其守护进程启用 cephx 协议，有几个重要步骤必须遵循。首先，必须给默认的 client.admin 生成密钥，以便管理员执行 ceph 命令；其次，必须生成一个监视器密钥，并发布到集群内的所有监视器；最后，你要完成启用 cephx 里的剩余步骤。

3.3.11.1.1 client.admin 密钥

THE CLIENT.ADMIN KEY

When you first install Ceph, each Ceph command you execute on the command line assumes that you are the client.admin default user. When running Ceph with cephx enabled, you need to have a key for the client.admin user to run ceph commands as the administrator.
你首次安装 ceph 后，你执行的每个 ceph 命令都先假设你是默认用户 client.admin。集群如果启用了 cephx，client.admin 用户就需要一个密钥才能运行 ceph 命令。

The following command will generate and register a client.admin key on the monitor with admin capabilities and write it to a keyring on the local file system. If the key already exists, its current value will be returned.
下列命令将在监视器上生成并注册一个具有管理能力的 client.admin 密钥，并将之写入本地文件系统的 keyring，如果密钥已存在，将返回其值。

```
sudo ceph auth get-or-create client.admin mds 'allow' osd 'allow *' mon 'allow *' >
/etc/ceph/keyring
```

See Enabling Cephx step 1 for stepwise details to enable cephx.
启用 cephx 的手把手教程，请参见启用 cephx 步骤 1。

3.3.11.1.2 监视器密钥环

MONITOR KEYRINGS

Ceph requires a keyring for the monitors. Use the ceph-authtool command to generate a secret monitor key and keyring.
ceph 的监视器需要一个密钥环，用 ceph-authtool 命令生成密钥和密钥环。

```
sudo ceph-authtool {keyring} --create-keyring --gen-key -n mon.
```

A cluster with multiple monitors must have identical keyrings for all monitors. So you must copy the keyring to each monitor host under the following directory:
有多个监视器的集群，其所有监视器的密钥环必须相同，所以你必须把下列目录里的密钥环拷到每个监视器：

```
/var/lib/ceph/mon/$cluster-$id
```

See Enabling Cephx step 2 and 3 for stepwise details to enable cephx.
启用 cephx 的详细步骤参见启用 cephx 的第 2、3 步。

3.3.11.1.3 启用 cephx

ENABLING CEPHX

When cephx is enabled, Ceph will look for the keyring in the default search path, which includes /etc/ceph/keyring. You can override this location by adding a keyring option in the [global] section of your Ceph configuration file, but this is not recommended.
启用 cephx 后，ceph 将在默认搜索路径（/etc/ceph/keyring）里查找密钥环。你可以在配置文件的[global]段里添加 keyring 选项来修改，但不推荐。

Execute the following procedures to enable cephx on a cluster with cephx disabled. If you (or your deployment utility) have already generated the keys, you may skip the steps related to generating keys.
在禁用了 cephx 的集群上执行下面的步骤来启用它，如果你（或者部署工具）已经生成了密钥，你可以跳过相关步骤。

Create a client.admin key, and save a copy of the key for your client host:
创建 client.admin 密钥，并为客户端保存此密钥的副本：

```
ceph auth get-or-create client.admin mon 'allow *' mds 'allow *' osd 'allow *' -o /etc/ceph/keyring
**Warning:** This will clobber any existing ``/etc/ceph/keyring`` file. Be careful!
```

Generate a secret monitor mon. key:
生成给监视器的 mon.密钥：

```
ceph-authtool --create --gen-key -n mon. /tmp/monitor-key
```

Copy the mon keyring into a keyring file in every monitor's mon data directory:
把 mon 密钥环拷贝到 keyring 文件，再拷到每个监视器的 mon 数据目录下：

```
cp /tmp/monitor-key /var/lib/ceph/mon/ceph-a/keyring
```

Generate a secret key for every OSD, where {$id} is the OSD number:
为每个 OSD 生成密钥，{id}是 OSD 编号：

```
ceph auth get-or-create osd.{$id} mon 'allow rwx' osd 'allow *' -o /var/lib/ceph/osd/ceph-
{$id}/keyring
```

Generate a secret key for every MDS, where {$id} is the MDS letter:
为每个 MDS 生成密钥，{id}是 MDS 的标识字母：

```
ceph auth get-or-create mds.{$id} mon 'allow rwx' osd 'allow *' mds 'allow *' -o
/var/lib/ceph/mds/ceph-{$id}/keyring
```

Enable cephx authentication for versions 0.51 and above by setting the following options in the [global] section of your Ceph configuration file:
0.51 以上版本启用 cephx 时，要在配置文件的[global]段下添加如下内容：

```
auth cluster required = cephx
auth service required = cephx
auth client required = cephx
```

Or, enable cephx authentication for versions 0.50 and below by setting the following option in the [global] section of your Ceph configuration file:
或者，0.50 以下版本启用 cephx 时，要在配置文件的[global]段下添加以下：

```
auth supported = cephx
```

Deprecated since version 0.51.
从 0.51 起死亡了。

Start or restart the Ceph cluster.
启动或重启 ceph 集群：

```
sudo service ceph -a start
sudo service ceph -a restart
```

3.3.11.1.4 禁用 cephx

DISABLING CEPHX

The following procedure describes how to disable Cephx. If your cluster environment is relatively safe, you can offset the computation expense of running authentication. We do not recommend it. However, it may be easier during setup and/or troubleshooting to temporarily disable authentication.
前面的步骤描述了如何禁用 cephx，如果你的集群环境相对安全，你可以去掉认证带来的计算消耗，然而我们不推荐。但是临时禁用认证会使安装、和/或排障更简单。

Disable cephx authentication for versions 0.51 and above by setting the following options in the [global] section of your Ceph configuration file:
在 0.51 及以上禁用 cephx 认证，要在配置文件的[global]段下设置：

```
auth cluster required = none
auth service required = none
auth client required = none
```

Or, disable cephx authentication for versions 0.50 and below (deprecated as of version 0.51) by setting the following option in the [global] section of your Ceph configuration file:
或者，在 0.50（从 0.51 时消失）及以下禁用 cephx，要在配置文件的[global]段下设置：

```
auth supported = none
```

Start or restart the Ceph cluster.
启动或重启 ceph 集群：

```
sudo service ceph -a start
sudo service ceph -a restart
```

3.3.11.1.5 守护进程密钥环

DAEMON KEYRINGS

With the exception of the monitors, daemon keyrings are generated in the same way that user keyrings are. By default, the daemons store their keyrings inside their data directory. The default keyring locations, and the capabilities necessary for the daemon to function, are shown below.
除监视器外，守护进程密钥环和用户密钥环生成方法一样。默认情况下，守护进程把密钥环保存在它们的数据目录下，默认密钥环位置、和守护进程发挥作用必需的能力展示如下：

ceph-mon
Location: $mon_data/keyring
Capabilities: N/A

ceph-osd
Location: $osd_data/keyring
Capabilities: mon 'allow rwx' osd 'allow *'

ceph-mds
Location: $mds_data/keyring
Capabilities: mds 'allow rwx' mds 'allow *' osd 'allow *'

radosgw
Location: $rgw_data/keyring
Capabilities: mon 'allow r' osd 'allow rwx'

Note that the monitor keyring contains a key but no capabilities, and is not part of the cluster auth database.
注意，监视器密钥环包含一个密钥，但没有能力，且不是集群认证数据库的一部分。

The daemon data directory locations default to directories of the form:
守护进程数据目录位置默认格式如下：

```
/var/lib/ceph/$type/$cluster-$id
```

For example, osd.12 would be:
例如，osd.12 的目录会是：

```
/var/lib/ceph/osd/ceph-12
```

You can override these locations, but it is not recommended.
你可以覆盖这些位置，但不推荐。

3.3.11.2 cephx 管理

CEPHX ADMINISTRATION

Cephx uses shared secret keys for authentication, meaning both the client and the monitor cluster have a copy of the client's secret key. The authentication protocol is such that both parties are able to prove to each other they have a copy of the key without actually revealing it. This provides mutual authentication, which means the cluster is sure the user possesses the secret key, and the user is sure that the cluster has a copy of the secret key.
cephx 用共享密钥来认证，即客户端和监视器集群各自都有客户端密钥的副本。这样的认证协议使所有参与者没有展现密钥就能相互证明，就是说集群确信用户可处理密钥、而且用户相信集群有密钥的副本。

Default users and pools are suitable for initial testing purposes. For test bed and production environments, you should create users and assign pool access to the users.
默认用户和存储池只适合最初的测试，在试验台和生产环境下，应该创建用户并给其分配存储池访问权限。

3.3.11.2.1 增加一个密钥

ADD A KEY
Keys enable a specific user to access the monitor, metadata server and cluster according to capabilities assigned to the key. Capabilities are simple strings specifying some access permissions for a given server type. Each server type has its own string. All capabilities are simply listed in {type} and {capability} pairs on the command line:
具体用户根据分配给密钥的能力可获得监视器、元数据服务器和集群的访问。能力是字符串，它给某类服务器指定访问权限，每种服务器都各有其字符串。在命令行下，所有能力都可成对地表示为{type}和{capability}。

```
sudo ceph auth get-or-create-key client.{username} {daemon1} {cap1} {daemon2} {cap2} ...
```

For example, to create a user client.foo with access 'rw' for daemon type 'osd' and 'r' for daemon type 'mon':
例如，要创建一个用户，他对 osd 有 rw 权限、对 mon 有 r 权限：
sudo ceph auth get-or-create-key client.foo osd rw mon r > keyring.foo

3.3.11.2.2 删除密钥

DELETE A KEY

To delete a key for a user or a daemon, use ceph auth del:
要删除一个用户或守护进程的密钥，用 ceph auth del：

```
ceph auth del {daemon-type}.{ID|username}
```

Where {daemon-type} is one of client, osd, mon, or mds, and {ID|username} is the ID of the daemon or the username.
{daemon-type}是客户端、osd、mon、mds 中一个，{ID|username}是守护进程或用户名的 ID。

3.3.11.2.3 列出集群内的密钥

LIST KEYS IN YOUR CLUSTER

To list the keys registered in your cluster:
要列出集群内注册的密钥：

```
sudo ceph auth list
```

3.3.11.3 cephx 命令行选项

CEPHX COMMANDLINE OPTIONS

When Ceph runs with Cephx enabled, you must specify a user name and a secret key on the command line. Alternatively, you may use the CEPH_ARGS environment variable to avoid re-entry of the user name and secret.
在启用了 cephx 的集群上，你必须在命令行指定用户名及其密钥；另外你也可以用 CEPH_ARGS 环境变量来避免多次输入用户名和密钥。

```
ceph --id {user-name} --keyring=/path/to/secret [commands]
```

For example:
例如：

```
ceph --id client.admin --keyring=/etc/ceph/ceph.keyring [commands]
```

Ceph supports the following usage for user name and secret:
ceph 支持用户名和密钥的下列用法：

--id | --user
Description:
Ceph identifies users with a type and an ID (e.g., TYPE.ID or client.admin, client.user1). The id, name and -n options enable you to specify the ID portion of the user name (e.g., admin, user1, foo, etc.). You can specify the user with the --id and omit the type. For example, to specify user client.foo enter the following:
ceph 用一个类型和 ID（如 TYPE.ID 或 client.admin、client.user1）来标识用户，id、name、和-n 选项可用于指定用户名（如 admin、user1、foo 等）的 ID 部分，你可以用--id 指定用户并忽略类型，例如可用下列命令指定 client.foo 用户：

```
ceph --id foo --keyring /path/to/keyring health
ceph --user foo --keyring /path/to/keyring health
```

--name
Description:
Ceph identifies users with a type and an ID (e.g., TYPE.ID or client.admin, client.user1). The --name and -n options enables you to specify the fully qualified user name. You must specify the user type (typically client) with the user ID. For example:
ceph 用一个类型和 ID（如 TYPE.ID 或 client.admin、client.user1）来标识用户，--name 和-n 选项可用于指定完整的用户名，但必须指定用户类型（一般是 client）和用户 ID，例如：

```
ceph --name client.foo --keyring /path/to/keyring health
ceph -n client.foo --keyring /path/to/keyring health
```

--keyring
Description:
The path to the keyring containing one or more user name and secret. The --secret option provides the same functionality, but it does not work with Ceph RADOS Gateway, which uses --secret for another purpose. You may retrieve a keyring with ceph auth get-or-create and store it locally. This is a preferred approach, because you can switch user names without switching the keyring path. For example:
包含一或多个用户名、密钥的密钥环路径。--secret 选项提供了相同功能，但它不能用于 RADOS 网关，其--secret 另有用途。你可以用 ceph auth get 获取密钥环并保存在本地，然后您就可以改用其他用户而无需重指定密钥环路径了。

```
sudo rbd map foo --pool rbd myimage --id client.foo --keyring /path/to/keyring
```

--keyfile
Description:
The path to the key file containing the secret key for the user specified by --id, --name, -n, or --user. You may retrieve the key for a specific user with ceph auth get and store it locally. Then, specify the path to the keyfile. For example:
包含用户（用--id、--name、-n、或--user 指定）密钥的文件路径。你可以用 ceph auth get 获取密钥并保存在本地，然后指定给--keyfile，例如：

```
sudo rbd map foo --pool rbd myimage --id client.foo --keyfile /path/to/file
```

Note Add the user and secret to the CEPH_ARGS environment variable so that you don't need to enter them each time. You can override the environment variable settings on the command line.
注意：把用户和密钥添加到 CEPH_ARGS 环境变量就不用每次都输入了。命令行下可以覆盖环境变量设置。

3.3.11.4 向后兼容性

BACKWARD COMPATIBILITY

New in version Bobtail.
bobtail 版新功能。

In Ceph Argonaut v0.48 and earlier versions, if you enable cephx authentication, Ceph only authenticates the initial communication between the client and daemon; Ceph does not authenticate the subsequent messages they send to each other, which has security implications. In Ceph Bobtail and subsequent versions, Ceph authenticates all ongoing messages between the entities using the session key set up for that initial authentication.
在 ceph-0.48 及更早版本，启用 cephx 认证后，ceph 仅认证客户端和守护进程间的最初通讯，不会认证后续相互发送的消息，这导致了安全隐患；bobtail 及后续版本会用认证后生成的会话密钥来认证所有消息。

We identified a backward compatibility issue between Argonaut v0.48 (and prior versions) and Bobtail (and subsequent versions). During testing, if you attempted to use Argonaut (and earlier) daemons with Bobtail (and later) daemons, the Argonaut daemons did not know how to perform ongoing message authentication, while the Bobtail versions of the daemons insist on authenticating message traffic subsequent to the initial request/response–making it impossible for Argonaut (and prior) daemons to interoperate with Bobtail (and subsequent) daemons.
我们确定了一个向后兼容性问题，在 Argonaut v0.48（及之前版本）和 Bobtail（及后续版本）之间。测试发现，如果你混用 Argonaut（及更小版）和 Bobtail 的守护进程，Argonaut 的守护进程将对正在进行的消息不知所措，因为 Bobtail 进程坚持要求认证最初请求/响应之后的消息，导致二者无法交互。

We have addressed this potential problem by providing a means for Argonaut (and prior) systems to interact with Bobtail (and subsequent) systems. Here's how it works: by default, the newer systems will not insist on seeing signatures from older systems that do not know how to perform them, but will simply accept such messages without authenticating them. This new default behavior provides the advantage of allowing two different releases to interact. We do not recommend this as a long term solution. Allowing newer daemons to forgo ongoing authentication has the unfortunate security effect that an attacker with control of some of your machines or some access to your network can disable session security simply by claiming to be unable to sign messages.
我们已经提供了一种方法，解决了 Argonaut（及之前）和 Bobtail（及后续）系统间交互的潜在的问题。是这样解决的，默认情况下，较新系统不会再坚持要求较老系统的签名，只是简单地接收这些消息而不对其认证。这个默认行为使得两个不同版本可以交互，但我们不推荐作为长期方案。允许较新进程不认证正在进行的消息会导致安全问题，因为攻击者如果能控制你的机器、或访问你的网络，就可以宣称不能签署消息，从而禁用会话安全。

Note Even if you don't actually run any old versions of Ceph, the attacker may be able to force some messages to be accepted unsigned in the default scenario. While running Cephx with the default scenario, Ceph still authenticates the initial communication, but you lose desirable session security.
If you know that you are not running older versions of Ceph, or you are willing to accept that old servers and new servers will not be able to interoperate, you can eliminate this security risk. If you do so, any Ceph system that is new enough to support session authentication and that has Cephx enabled will reject unsigned messages. To preclude new servers from interacting with old servers, include the following line into the [global] section of your Ceph configuration file directly below the line that specifies the use of Cephx for authentication:
注意：即使你没有使用旧版的 ceph，在默认配置下，攻击者也可以强制一些未签署消息被接受；虽然初始通讯认证通过了，但你失去了会话安全。如果你确定不会使用旧版 ceph、或者新旧服务器不能交互无所谓，那就可以排除这个安全风险；如果你这样做了，任何支持会话认证、启用了 cephx 的 ceph 系统都会拒绝未

签名的消息。要防止新服务器和旧服务器交互，在配置文件的[global]下添加下列这行，要加到启用 cephx 之后。

```
cephx require signatures = true
```

We recommend migrating all daemons to the newer versions and enabling the foregoing flag at the nearest practical time so that you may avail yourself of the enhanced authentication.
我们推荐把所有进程迁移到较新版本，然后关闭兼容选项，以增强认证安全性。

3.3.12 添加/删除 OSD

如果您的集群已经在运行，你可以在运行时添加或删除 OSD。

3.3.12.1 添加 OSD

When you want to expand a cluster, you may add an OSD at runtime. With Ceph, an OSD is generally one Ceph ceph-osd daemon for one storage disk within a host machine. If your host has multiple storage disks, you may map one ceph-osd daemon for each disk.
你迟早要扩容集群，ceph 允许在运行时增加 OSD。在 ceph 里，一个 OSD 一般是一个 ceph-osd 守护进程，它运行在硬盘之上，如果你有多个硬盘，可以给每个硬盘启动一个 ceph-osd 守护进程。

Generally, it's a good idea to check the capacity of your cluster to see if you are reaching the upper end of its capacity. As your cluster reaches its near full ratio, you should add one or more OSDs to expand your cluster's capacity.
通常，你应该监控集群容量，看是否达到了容量上限，因为达到了它的 near full ratio 值后，要增加一或多个 OSD 来扩容。

Warning Do not let your cluster reach its full ratio before adding an OSD. OSD failures that occur after the cluster reaches its near full ratio may cause the cluster to exceed its full ratio.
警告：不要等空间满了再增加 OSD，空间使用率达到 near full ratio 值后，OSD 失败可能导致集群空间占满。

3.3.12.1.1 部署硬件

DEPLOY YOUR HARDWARE

If you are adding a new host when adding a new OSD, see Hardware Recommendations for details on minimum recommendations for OSD hardware. To add a OSD host to your cluster, first make sure you have an up-to-date version of Linux installed (typically Ubuntu 12.04 precise), and you have made some initial preparations for your storage disks. See Filesystem Recommendations for details.
如果你通过增加主机来增加 OSD，关于 OSD 服务器硬件的配置请参见硬件推荐。要把一台 OSD 主机加入到集群，首先要安装最新版的 Linux（如 Ubuntu 12.04），而且存储硬盘要做好必要的准备，详情参见硬盘和文件系统推荐。

Add your OSD host to a rack in your cluster, connect it to the network and ensure that it has network connectivity.
把 OSD 主机添加到集群机架上，连接好网络、确保网络通畅。

3.3.12.1.2 安装推荐软件

INSTALL THE REQUIRED SOFTWARE

For manually deployed clusters, you must install Ceph packages manually. See Installing Debian/Ubuntu Packages for details. You should configure SSH to a user with password-less authentication and root permissions.
在手动部署的集群里，你必须手动安装 ceph 软件包，详情参见安装 ceph 软件包。你应该配置一个无密码登录 SSH 的用户，且他有 root 权限。

For clusters deployed with Chef, create a chef user, configure SSH keys, install Ruby and install the Chef client on your host. See Installing Chef for details.
对于用 chef 部署的集群，要在主机上创建一个 cheg 用户、配置 SSH 密钥对、安装 Ruby、安装 chef 客户端，详情参见安装 chef。

3.3.12.1.3 添加 OSD（手动）

ADDING AN OSD (MANUAL)

This procedure sets up an ceph-osd daemon, configures it to use one disk, and configures the cluster to distribute data to the OSD. If your host has multiple disks, you may add an OSD for each disk by repeating this procedure.
此过程要设置一个 ceph-osd 守护进程，让它使用一个硬盘，且让集群把数据发布到 OSD。如果一台主机有多个硬盘，可以重复此过程，把每个硬盘配置为一个 OSD。

To add an OSD, create a data directory for it, mount a disk to that directory, add the OSD to your configuration file, add the OSD to the cluster, and then add it to the CRUSH map.
要添加 OSD，要依次创建数据目录、把硬盘挂载到目录、把 OSD 添加到配置文件、把 OSD 加入集群、把 OSD 加入 CRUSH 图。

When you add the OSD to the CRUSH map, consider the weight you give to the new OSD. Hard disk capacity grows 40% per year, so newer OSD hosts may have larger hard disks than older hosts in the cluster (i.e., they may have greater weight).
往 CRUSH 图里添加 OSD 时建议设置权重，硬盘容量每年增长 40%，所以较新的 OSD 主机拥有更大的空间（如他们可以有更大的权重）。

Create the default directory on your new OSD.
在新 OSD 主机上创建默认目录。

```
ssh {new-osd-host}
sudo mkdir /var/lib/ceph/osd/ceph-{osd-number}
```

If the OSD is for a disk other than the OS disk, prepare it for use with Ceph, and mount it to the directory you just created:
如果给 OSD 用的是单独的而非系统盘，先把它挂载到刚创建的目录下：

```
ssh {new-osd-host}
sudo mkfs -t {fstype} /dev/{disk}
sudo mount -o user_xattr /dev/{hdd} /var/lib/ceph/osd/ceph-{osd-number}
```

Navigate to the host where you keep the master copy of the cluster's ceph.conf file.
登录到保存 ceph.conf 主拷贝的主机上：

```
ssh {admin-host}
cd /etc/ceph
vim ceph.conf
```

Add the new OSD to your ceph.conf file.
把新 OSD 添加到 ceph.conf 文件里：

```
[osd.123]
    host = {hostname}
```

From the host where you keep the master copy of the cluster's ceph.conf file, copy the updated ceph.conf file to your new OSD's /etc/ceph directory and to other hosts in your cluster.
从保存集群 ceph.conf 主拷贝的主机上，把更新过的 ceph.conf 拷贝到新 OSD 主机和其他主机的/etc/ceph/目录下。

```
ssh {new-osd} sudo tee /etc/ceph/ceph.conf < /etc/ceph/ceph.conf
```

Create the OSD.
创建 OSD。

```
ceph osd create {osd-num}
ceph osd create 123 #for example
```

Initialize the OSD data directory.
初始化 OSD 数据目录。

```
ssh {new-osd-host}
ceph-osd -i {osd-num} --mkfs --mkkey
```

The directory must be empty before you can run ceph-osd.
运行 ceph-osd 时目录必须是空的。

Register the OSD authentication key. The value of ceph for ceph-{osd-num} in the path is the $cluster-$id. If your cluster name differs from ceph, use your cluster name instead.:
注册 OSD 认证密钥，ceph-{osd-num}路径里的 ceph 值应该是$cluster-$id，如果你的集群名字不是 ceph，那就用改过的名字：

```
ceph auth add osd.{osd-num} osd 'allow *' mon 'allow rwx' -i /var/lib/ceph/osd/ceph-{osd-
num}/keyring
```

Add the OSD to the CRUSH map so that it can begin receiving data. You may also decompile the CRUSH map, add the OSD to the device list, add the host as a bucket (if it's not already in the CRUSH map), add the device as an item in the host, assign it a weight, recompile it and set it. See Add/Move an OSD for details.
把 OSD 加入 CRUSH 图以便它接收数据，也可以反编译 CRUSH 图、把 OSD 加入 device list、以桶的形式加入主机（如果它没在 CRUSH 图里）、以条目形式把设备加入主机、分配权重、重编译并应用它。详情参见增加/移动 OSD。

```
ceph osd crush set {id} {name} {weight} pool={pool-name}  [{bucket-type}={bucket-name} ...]
```

0.48 版最佳实践

Argonaut (v0.48) Best Practices
To limit impact on user I/O performance, add an OSD to the CRUSH map with an initial weight of 0. Then, ramp up the CRUSH weight a little bit at a time. For example, to ramp by increments of 0.2, start with:
为降低对用户 I/O 性能的影响，加入 CRUSH 图时应该把 OSD 的初始权重设为 0，然后每次增大一点、逐步增大 CRUSH 权重。例如每次增加 0.2：

```
ceph osd crush reweight {osd-id} .2
```

and allow migration to complete before reweighting to 0.4, 0.6, and so on until the desired CRUSH weight is reached.
迁移完成前，可以依次把权重重置为 0.4、0.6 等等，直到达到期望权重。

To limit the impact of OSD failures, you can set:
为降低 OSD 失败的影响，你可以设置：

```
mon osd down out interval = 0
```

which prevents down OSDs from automatically being marked out, and then ramp them down manually with:
它防止挂了的 OSD 自动被标记为 out，然后逐步降低其权重：

```
ceph osd reweight {osd-num} .8
```

Again, wait for the cluster to finish migrating data, and then adjust the weight further until you reach a weight of 0. Note that this problem prevents the cluster to automatically re-replicate data after a failure, so please ensure that sufficient monitoring is in place for an administrator to intervene promptly.
还是等着集群完成数据迁移，然后再次调整权重，直到权重为 0。注意，这会阻止集群在发生故障时自动重复制数据，所以要确保监控的及时性，以便管理员迅速介入。

Note that this practice will no longer be necessary in Bobtail and subsequent releases.
注意：以上经验在 Bobtail 及后续版本已不再必要。

3.3.12.1.4 添加 OSD（chef）

ADDING AN OSD (CHEF)

This procedure configures your OSD using chef-client. If your host has multiple disks, you may need to execute the procedure for preparing an OSD disk for each data disk on your host.
这个步骤用 chef-client 配置 OSD，如果主机有多个硬盘，每个硬盘都要重复此步骤。

When you add the OSD to the CRUSH map, consider the weight you give to the new OSD. Hard disk capacity grows 40% per year, so newer OSD hosts may have larger hard disks than older hosts in the cluster.
往 CRUSH 图里添加 OSD 时建议设置权重，硬盘容量每年增长 40%，所以较新的 OSD 主机拥有更大的空间（如他们可以有更大的权重）。

Execute chef-client to register it with Chef as a Chef node.
执行 chef-client 来注册为 chef 节点。

Edit the node. See Configure Nodes for details. Change its environment to your Chef environment. Add "role[ceph-osd]" to the run list.
编辑节点，详情参见配置节点。进入 chef 环境，把"role[ceph-osd]"添加到运行列表。

Execute Prepare OSD Disks for each disk.
在每个硬盘上执行 Prepare OSD Disks。

Execute chef-client to invoke the run list.
执行 chef-client 调用运行列表。

Add the OSD to the CRUSH map so that it can begin receiving data. You may also decompile the CRUSH map edit the file, recompile it and set it. See Add/Move an OSD for details.
把 OSD 加入 CRUSH 图以便它接收数据，也可以反编译 CRUSH 图、编辑文件、重编译并应用它。详情参见增加/移动 OSD。

```
ceph osd crush set {id} {name} {weight} pool={pool-name}  [{bucket-type}={bucket-name} ...]
```

3.3.12.1.5 启动 OSD

STARTING THE OSD

After you add an OSD to Ceph, the OSD is in your configuration. However, it is not yet running. The OSD is down and out. You must start your new OSD before it can begin receiving data. You may use service ceph from your admin host or start the OSD from its host machine:
把 OSD 加入 ceph 后，OSD 就在配置里了。然而，它还没运行，其状态为 down 且 out，开始收数据前必须先启动。可以用管理主机上的服务、或从 OSD 所在主机启动。

```
service ceph -a start osd.{osd.num}
#or alternatively
ssh {new-osd-host}
sudo /etc/init.d/ceph start osd.{osd-num}
```

Once you start your OSD, it is up.
只要启动了 OSD，它就是 up 状态。

3.3.12.1.6 把 OSD 推进集群

PUT THE OSD IN THE CLUSTER

After you start your OSD, it is up and out. You need to put it in to the cluster so that Ceph can begin writing data to it.
启动 OSD 后，它是 up 且 out 的，你得把它推入集群，ceph 才能向其写入数据。

```
ceph osd in {osd-num}
```

3.3.12.1.7 观察数据迁移

OBSERVE THE DATA MIGRATION

Once you have added your new OSD to the CRUSH map, Ceph will begin rebalancing the server by migrating placement groups to your new OSD. You can observe this process with the ceph tool.
把新 OSD 加入 CRUSH 图后，ceph 会重新均衡服务器，一些归置组会迁移到新 OSD 里，你可以用 ceph 命令观察此过程。

```
ceph -w
```

You should see the placement group states change from active+clean to active, some degraded objects, and finally active+clean when migration completes. (Control-c to exit.)
你会看到归置组状态从 active+clean 变为 active、一些降级的对象、且迁移完成后回到 active+clean 状态。（Ctrl-c 退出）

3.3.12.2 删除OSD

When you want to reduce the size of a cluster or replace hardware, you may remove an OSD at runtime. With Ceph, an OSD is generally one Ceph ceph-osd daemon for one storage disk within a host machine. If your host has multiple storage disks, you may need to remove one ceph-osd daemon for each disk. Generally, it's a good idea to check the capacity of your cluster to see if you are reaching the upper end of its capacity. Ensure that when you remove an OSD that your cluster is not at its near full ratio.
要想缩减集群尺寸或替换硬件，可在运行时删除 OSD。在 ceph 里，一个 OSD 通常是一台主机上的一个 ceph-osd 守护进程、它运行在一个硬盘之上。如果一台主机上有多个数据盘，你得挨个删除其对应 ceph-osd。通常，操作前应该检查集群容量，看是否快达到上限了，确保删除 OSD 后不会使集群达到 near full ratio 值。

Warning Do not let your cluster reach its full ratio when removing an OSD. Removing OSDs could cause the cluster to reach or exceed its full ratio.
警告：删除 OSD 时不要让集群达到其最大利用率，删除 OSD 可能导致集群达到或超过最大利用率。

3.3.12.2.1 把 OSD 踢出集群

TAKE THE OSD OUT OF THE CLUSTER

Before you remove an OSD, it is usually up and in. You need to take it out of the cluster so that Ceph can begin rebalancing and copying its data to other OSDs.
删除 OSD 前，它通常是 up 且 in 的，要先把它踢出集群，以使 ceph 启动重新均衡、把数据拷贝到其他 OSD。

```
ceph osd out {osd-num}
```

3.3.12.2.2 观察数据迁移

OBSERVE THE DATA MIGRATION

Once you have taken your OSD out of the cluster, Ceph will begin rebalancing the cluster by migrating placement groups out of the OSD you removed. You can observe this process with the ceph tool.
一旦把 OSD 踢出集群，ceph 就会开始重新均衡集群、把归置组迁移出删除的 OSD。你可以用 ceph 程序观察此过程。

```
ceph -w
```

You should see the placement group states change from active+clean to active, some degraded objects, and finally active+clean when migration completes. (Control-c to exit.)
你会看到归置组状态从 active+clean 变为 active、有一些降级的对象、迁移完成后最终回到 active+clean 状态。（Ctrl-c 中止）

3.3.12.2.3 停止 OSD

STOPPING THE OSD
After you take an OSD out of the cluster, it may still be running. That is, the OSD may be up and out. You must stop your OSD before you remove it from the configuration.

把 OSD 踢出集群后，它可能仍在运行，就是说其状态为 up 且 out。删除前要先停止 OSD 进程。

```
ssh {new-osd-host}
sudo /etc/init.d/ceph stop osd.{osd-num}
```

Once you stop your OSD, it is down.
停止 OSD 后，状态变为 down。

3.3.12.2.4 删除一个 OSD（手动）

This procedure removes an OSD from a cluster map, removes its authentication key, removes the OSD from the OSD map, and removes the OSD from the ceph.conf file. If your host has multiple disks, you may need to remove an OSD for each disk by repeating this procedure.
此步骤依次把一个 OSD 移出集群 CRUSH 图、删除认证密钥、删除 OSD 图条目、删除 ceph.conf 条目。如果主机有多个硬盘，每个硬盘对应的 OSD 都得重复此步骤。

Remove the OSD from the CRUSH map so that it no longer receives data. You may also decompile the CRUSH map, remove the OSD from the device list, remove the device as an item in the host bucket or remove the host bucket (if it's in the CRUSH map and you intend to remove the host), recompile the map and set it. See Remove an OSD for details.
删除 CRUSH 图的对应 OSD 条目，它就不再接收数据了。你也可以反编译 CRUSH 图、删除 device 列表条目、删除对应的 host 桶条目或删除 host 桶（如果它在 CRUSH 图里，而且你想删除主机），重编译 CRUSH 图并应用它。详情参见删除 OSD。

```
ceph osd crush remove {name}
```

Remove the OSD authentication key.
删除 OSD 认证密钥：

```
ceph auth del osd.{osd-num}
```

The value of ceph for ceph-{osd-num} in the path is the $cluster-$id. If your cluster name differs from ceph, use your cluster name instead.
ceph-{osd-num}路径里的 ceph 值是$cluster-$id，如果集群名字不是 ceph，这里要更改。

Remove the OSD.
删除 OSD。

```
ceph osd rm {osd-num}
#for example
ceph osd rm 123
```

Navigate to the host where you keep the master copy of the cluster's ceph.conf file.
登录到保存 ceph.conf 主拷贝的主机：

```
ssh {admin-host}
cd /etc/chef
vim ceph.conf
```

从 ceph.conf 配置文件里删除对应条目。

```
[osd.123]
        host = {hostname}
```

From the host where you keep the master copy of the cluster's ceph.conf file, copy the updated ceph.conf file to the /etc/ceph directory of other hosts in your cluster.
从保存 ceph.conf 主拷贝的主机，把更新过的 ceph.conf 拷贝到集群其他主机的/etc/ceph 目录下。

```
ssh {osd} sudo tee /etc/ceph/ceph.conf < /etc/ceph/ceph.conf
```

3.3.13 增加/删除监视器

Adding/removing monitors
When you have a cluster up and running, you may add or remove monitors from the cluster at runtime.

你的集群启动并运行后，可以在运行时增加、或删除监视器。

3.3.13.1 增加监视器

ADDING MONITORS
Ceph monitors are light-weight processes that maintain a master copy of the cluster map. You can run a cluster with 1 monitor. We recommend at least 3 monitors for a production cluster. Ceph monitors use PAXOS to establish consensus about the master cluster map, which requires a majority of monitors running to establish a quorum for consensus about the cluster map (e.g., 1; 3 out of 5; 4 out of 6; etc.).
ceph 监视器是轻量级进程，它维护着集群运行图的副本。一个集群可以只有一个监视器，我们推荐生产环境至少 3 个监视器。ceph 使用 PAXOS 算法对主集群运行图达成共识，它要求大部分监视器在运行，以达到关于集群运行图建立共识所需的法定人数（如 1、5 个中的 3 个、6 个中的 4 个等等）。

Since monitors are light-weight, it is possible to run them on the same host as an OSD; however, we recommend running them on separate hosts.
正因为监视器是轻量级的，所以有可能在作为 OSD 的主机上同时运行它；然而，我们推荐运行于单独主机。

> Important A majority of monitors in your cluster must be able to reach each other in order to establish a quorum.
> 重要：要确立法定人数，集群内的大部分监视器必须互相可达。

3.3.13.1.1 部署硬件

DEPLOY YOUR HARDWARE
If you are adding a new host when adding a new monitor, see Hardware Recommendations for details on minimum recommendations for monitor hardware. To add a monitor host to your cluster, first make sure you have an up-to-date version of Linux installed (typically Ubuntu 12.04 precise).
如果你增加新监视器时要新增一台主机，关于其最低硬件配置请参见硬件推荐。要增加一个监视器主机，首先要安装最新版的 Linux（如 Ubuntu 12.04）。

Add your monitor host to a rack in your cluster, connect it to the network and ensure that it has network connectivity.
把监视器主机安装上架，连通网络。

3.3.13.1.2 安装必要软件

INSTALL THE REQUIRED SOFTWARE
For manually deployed clusters, you must install Ceph packages manually. See Installing Debian/Ubuntu Packages for details. You should configure SSH to a user with password-less authentication and root permissions.
手动部署的集群，ceph 软件包必须手动装，详情参见安装 ceph 软件包。应该配置一个用户，使之可以无密码登录 SSH、且有 root 权限。

For clusters deployed with Chef, create a chef user, configure SSH keys, install Ruby and install the Chef client on your host. See Installing Chef for details.
chef 部署的集群，创建一个 chef 用户、配置 SSH 密钥、安装 Ruby 并且在主机上安装 chef 客户端。详情参见安装 chef。

3.3.13.1.3 增加监视器（手动）

ADDING A MONITOR (MANUAL)
This procedure creates a ceph-mon data directory, retrieves the monitor map and monitor keyring, and adds a ceph-mon daemon to your cluster. If this results in only two monitor daemons, you may add more monitors by repeating this procedure until you have a sufficient number of ceph-mon daemons to achieve a quorum.
本步骤创建 ceph-mon 数据目录、获取监视器运行图和监视器密钥环、增加一个 ceph-mon 守护进程。如果这导致只有 2 个监视器守护进程，你可以重演此步骤来增加一或多个监视器，直到你拥有足够多 ceph-mon 建立法定人数。

Create the default directory on your new monitor.
在新监视器上创建默认目录:

```
ssh {new-mon-host}
sudo mkdir /var/lib/ceph/mon/ceph-{mon-letter}
```

Create a temporary directory {tmp} to keep the files needed during this process. This directory should be different from monitor's default directory created in the previous step, and can be removed after all the steps are taken.
创建临时目录，用以保存此过程中用到的文件。此目录应不同于前面步骤创建的监视器数据目录，且完成后可删除。

```
mkdir {tmp}
```

Retrieve the keyring for your monitors, where {tmp} is the path to the retrieved keyring, and {filename} is the name of the file containing the retrieved monitor key.
获取监视器密钥环，{tmp}是密钥环文件保存路径、{filename}是包含密钥的文件名。

```
ceph auth get mon. -o {tmp}/{filename}
```

Retrieve the monitor map, where {tmp} is the path to the retrieved monitor map, and {filename} is the name of the file containing the retrieved monitor monitor map.
获取监视器运行图，{tmp}是获取到的监视器运行图、{filename}是包含监视器运行图的文件名。

```
ceph mon getmap -o {tmp}/{filename}
```

Prepare the monitor's data directory created in the first step. You must specify the path to the monitor map so that you can retrieve the information about a quorum of monitors and their fsid. You must also specify a path to the monitor keyring:
准备第一步创建的监视器数据目录。必须指定监视器运行图路径，这样才能获得监视器法定人数和它们 fsid 的信息；还要指定监视器密钥环路径。

```
sudo ceph-mon -i {mon-letter} --mkfs --monmap {tmp}/{filename} --keyring {tmp}/{filename}
```

Add a [mon.{letter}] entry for your new monitor in your ceph.conf file.
把新监视器的[mon.{letter}]条目添加到 ceph.conf 文件里。

```
[mon.c]
        host = new-mon-host
        addr = ip-addr:6789
```

Add the new monitor to the list of monitors for you cluster (runtime). This enables other nodes to use this monitor during their initial startup.
把新监视器添加到集群的监视器列表里（运行时），这允许其它节点开始启动时使用这个节点。

```
ceph mon add <name> <ip>[:<port>]
```

Start the new monitor and it will automatically join the cluster. The daemon needs to know which address to bind to, either via --public-addr {ip:port} or by setting mon addr in the appropriate section of ceph.conf. For example:
启动新监视器，它会自动加入机器。守护进程需知道绑定到哪个地址，通过--public-addr {ip:port}或在 ceph.conf 里的相应段设置 mon addr 可以指定。

```
ceph-mon -i newname --public-addr {ip:port}
```

3.3.13.2 删除监视器

REMOVING MONITORS
When you remove monitors from a cluster, consider that Ceph monitors use PAXOS to establish consensus about the master cluster map. You must have a sufficient number of monitors to establish a quorum for consensus about the cluster map.
从集群删除监视器时，必须认识到，ceph 监视器用 PASOX 算法关于主集群运行图达成共识。必须有足够多的监视器才能对集群运行图达成共识。

3.3.13.2.1 删除监视器（手动）

REMOVING A MONITOR (MANUAL)
This procedure removes a ceph-mon daemon from your cluster. If this procedure results in only two monitor daemons, you may add or remove another monitor until you have a number of ceph-mon daemons that can achieve a quorum.
本步骤从集群删除 ceph-mon 守护进程，如果此步骤导致只剩 2 个监视器了，你得另外增加一或多个监视器，直到达成法定人数所必需的 ceph-mon 数量。

Stop the monitor.
停止监视器。

```
service ceph -a stop mon.{mon-letter}
```

Remove the monitor from the cluster.
从集群删除监视器。

```
ceph mon remove {mon-letter}
```

Remove the monitor entry from ceph.conf.
删除 ceph.conf 对应条目。

3.3.13.2.2 从不健康集群删除监视器

REMOVING MONITORS FROM AN UNHEALTHY CLUSTER

This procedure removes a ceph-mon daemon from an unhealhty cluster–i.e., a cluster that has placement groups that are persistently not active + clean.
本步骤从不健康集群删除 ceph-mon，即集群有些归置组永久偏离 active+clean。

Identify a surviving monitor.
找出活着的监视器。

```
ceph mon dump
```

Navigate to a surviving monitor's monmap directory.
进入幸存监视器的 monmap 目录。

```
ssh {mon-host}
cd /var/lib/ceph/mon/ceph-{mon-letter}/monmap
```

List the directory contents and identify the last commmitted map. Directory contents will show a numeric list of maps.
列出目录内容，找出最后提交的运行图。目录内容是运行图的数字列表。

```
ls
1  2  3  4  5  first_committed  last_committed  last_pn  latest
```

Identify the most recently committed map.
找出最近提交的运行图。

```
sudo cat last_committed
```

Copy the most recently committed file to a temporary directory.
把最近提交的文件拷到一个临时目录。

```
cp /var/lib/ceph/mon/ceph-{mon-letter}/monmap/{last_committed} /tmp/surviving_map
```

Remove the non-surviving monitors. For example, if you have three monitors, mon.a, mon.b, and mon.c, where only mon.a will survive, follow the example below:
删除死亡监视器。例如，假设你有 3 个监视器，mon.a、mon.b 和 mon.c，其中只有 mon.a 要保留，如下：

```
monmaptool /tmp/surviving_map --rm {mon-letter}
#for example
monmaptool /tmp/surviving_map --rm b
```

```
monmaptool /tmp/surviving_map --rm c
```

Stop all monitors.
停止所有监视器。

```
service ceph -a stop mon
```

Inject the surviving map with the removed monitors into the surviving monitors. For example, to inject a map into monitor mon.a, follow the example below:
把找出的幸存运行图注入保留的监视器，例如，要把运行图注入 mon.a，模仿如下：
ceph-mon -i {mon-letter} --inject-monmap {map-path}

```
#for example
ceph-mon -i a --inject-monmap /etc/surviving_map
```

3.3.14 更改监视器 IP 地址

Changing a Monitor's IP Address

> Important Existing monitors are not supposed to change their IP addresses.
> **重要**：现有监视器不应该更改其 IP 地址。

Monitors are critical components of a Ceph cluster, and they need to maintain a quorum for the whole system to work properly. To establish a quorum, the monitors need to discover each other. Ceph has strict requirements for discovering monitors.
监视器是 ceph 集群的关键组件，它们要维护一个法定人数，这样整个系统才能正常工作。要确立法定人数，监视器得互相发现对方，ceph 对监视器的发现要求严格。

Ceph clients and other Ceph daemons use ceph.conf to discover monitors. However, monitors discover each other using the monitor map, not ceph.conf. For example, if you refer to Adding a Monitor (Manual) you will see that you need to obtain the current monmap for the cluster when creating a new monitor, as it is one of the required arguments of ceph-mon -i {mon-id} --mkfs. The following sections explain the consistency requirements for Ceph monitors, and a few safe ways to change a monitor's IP address.
ceph 客户端及其它 ceph 守护进程用 ceph.conf 发现监视器，然而，监视器之间用监视器运行图发现对方，而非 ceph.conf。例如，如果你参照了增加监视器，会发现创建新监视器时得获取当前集群的 monmap，因为它是 ceph-mon -i {mon-id} --mkfs 命令的必要参数。下面几段解释了 ceph 监视器的一致性要求，和几种改 IP 的安全方法。

3.3.14.1 一致性要求

Consistency Requirements
A monitor always refers to the local copy of the monmap when discovering other monitors in the cluster. Using the monmap instead of ceph.conf avoids errors that could break the cluster (e.g., typos in ceph.conf when specifying a monitor address or port). Since monitors use monmaps for discovery and they share monmaps with clients and other Ceph daemons, the monmap provides monitors with a strict guarantee that their consensus is valid.
监视器发现集群内的其它监视器时总是参照 monmap 的本地副本，用 monmap 而非 ceph.conf 可避免因配置错误（例如在 ceph.conf 里指定监视器地址或端口时拼写错误）而损坏集群。正因为监视器用 monmaps 相互发现、且共享于客户端和其它 ceph 守护进程间，所以 monmap 以苛刻的一致性提供监视器。

Strict consistency also applies to updates to the monmap. As with any other updates on the monitor, changes to the monmap always run through a distributed consensus algorithm called Paxos. The monitors must agree on each update to the monmap, such as adding or removing a monitor, to ensure that each monitor in the quorum has the same version of the monmap. Updates to the monmap are incremental so that monitors have the latest agreed upon version, and a set of previous versions, allowing a monitor that has an older version of the monmap to catch up with the current state of the cluster.
苛刻的一致性要求也适用于 monmap 的更新，因为任何有关监视器的更新、monmap 的更改都通过名为 Pasox 的分布式一致性算法运行。为保证法定人数里的所有监视器都持有同版本 monmap，所有监视器都要赞成 monmap 的每一次更新，像增加、删除监视器。monmap 的更新是增量的，这样监视器都有最近商定的版本

以及一系列之前版本，这样可使一个有较老 monmap 的监视器赶上集群当前的状态。

If monitors discovered each other through the Ceph configuration file instead of through the monmap, it would introduce additional risks because the Ceph configuration files aren't updated and distributed automatically. Monitors might inadvertantly use an older `ceph.conf` file, fail to recognize a monitor, fall out of a quorum, or develop a situation where Paxos isn't able to determine the current state of the system accurately. Consequently, making changes to an existing monitor's IP address must be done with great care.
如果监视器通过 ceph 配置文件而非 monmap 相互发现，就会引进额外风险，因为 ceph 配置文件不会自动更新和发布。监视器有可能用了较老的 ceph.conf 而导致不能识别某监视器、掉出法定人数、或者发展为一种 Paxos 不能精确确定当前系统状态的情形。总之，更改现有监视器的 IP 地址必须慎之又慎。

3.3.14.2 更改监视器IP 地址（正确方法）

Changing a Monitor's IP address (The Right Way)
Changing a monitor's IP address in `ceph.conf` only is not sufficient to ensure that other monitors in the cluster will receive the update. To change a monitor's IP address, you must add a new monitor with the IP address you want to use (as described in Adding a Monitor (Manual)), ensure that the new monitor successfully joins the quorum; then, remove the monitor that uses the old IP address. Then, update the `ceph.conf` file to ensure that clients and other daemons know the IP address of the new monitor.
仅仅在 ceph.conf 里更改监视器的 IP 不足以让集群内的其它监视器接受更新。要更改一个监视器的 IP 地址，你必须以先以想用的 IP 地址增加一个监视器（见增加监视器），确保新监视器成功加入法定人数，然后删除用旧 IP 的监视器，最后更新 ceph.conf 以确保客户端和其它守护进程得知新监视器的 IP 地址。

For example, lets assume there are three monitors in place, such as
例如，我们假设有 3 个监视器，如下：

```
[mon.a]
        host = host01
        addr = 10.0.0.1:6789
[mon.b]
        host = host02
        addr = 10.0.0.2:6789
[mon.c]
        host = host03
        addr = 10.0.0.3:6789
```

To change mon.c to host04 with the IP address 10.0.0.4, follow the steps in Adding a Monitor (Manual) by adding a new monitor mon.d. Ensure that mon.d is running before removing mon.c, or it will break the quorum. Remove mon.c as described on Removing a Monitor (Manual). Moving all three monitors would thus require repeating this process as many times as needed.
要把 host04 上 mon.c 的 IP 改为 10.0.0.4，按照增加监视器（手动）里的步骤增加一个新监视器 mon.d，确认它运行正常后再删除 mon.c，否则会破坏法定人数；最后依照删除监视器（手动）删除 mon.c。3 个监视器都要更改的话，每次都要重复一次。

3.3.14.3 更改监视器IP 地址（错误方法）

Changing a Monitor's IP address (The Messy Way)
There may come a time when the monitors must be moved to a different network, a different part of the datacenter or a different datacenter altogether. While it is possible to do it, the process becomes a bit more hazardous.
可能有时候监视器不得不挪到不同的网络、数据中心的不同位置、甚至不同的数据中心，这是可能的，但过程有点惊险。

In such a case, the solution is to generate a new monmap with updated IP addresses for all the monitors in the cluster, and inject the new map on each individual monitor. This is not the most user-friendly approach, but we do not expect this to be something that needs to be done every other week. As it is clearly stated on the top of this section, monitors are not supposed to change IP addresses.
在这种情形下，一种方法是用所有监视器的新 IP 地址生成新 monmap，并注入到集群内的所有监视器。对大多数用户来说，这并不简单，好在它不常见。再次重申，监视器不应该更改 IP 地址。

Using the previous monitor configuration as an example, assume you want to move all the monitors from the

10.0.0.x range to 10.1.0.x, and these networks are unable to communicate. Use the following procedure:
以前面的监视器配置为例，假设你想把所有监视器的 IP 从 10.0.0.x 改为 10.1.0.x，并且两个网络互不相通，步骤如下：

1. Retrieve the monitor map, where {tmp} is the path to the retrieved monitor map, and {filename} is the name of the file containing the retrieved monitor monitor map.
   获取监视器运行图，其中{tmp}是所获取的运行图路径，{filename}是监视器运行图的文件名。
   ceph mon getmap -o {tmp}/{filename}
2. The following example demonstrates the contents of the monmap.
   下面是一个 monmap 内容示例：
   $ monmaptool --print {tmp}/{filename}

   monmaptool: monmap file {tmp}/{filename}
   epoch 1
   fsid 224e376d-c5fe-4504-96bb-ea6332a19e61
   last_changed 2012-12-17 02:46:41.591248
   created 2012-12-17 02:46:41.591248
   0: 10.0.0.1:6789/0 mon.a
   1: 10.0.0.2:6789/0 mon.b
   2: 10.0.0.3:6789/0 mon.c
3. Remove the existing monitors.
   删除现有监视器。
   $ monmaptool --rm a --rm b --rm c {tmp}/{filename}

   monmaptool: monmap file {tmp}/{filename}
   monmaptool: removing a
   monmaptool: removing b
   monmaptool: removing c
   monmaptool: writing epoch 1 to {tmp}/{filename} (0 monitors)
4. Add the new monitor locations.
   添加新监视器位置。
   $ monmaptool --add a 10.1.0.1:6789 --add b 10.1.0.2:6789 --add c 10.1.0.3:6789 {tmp}/{filename}

   monmaptool: monmap file {tmp}/{filename}
   monmaptool: writing epoch 1 to {tmp}/{filename} (3 monitors)
5. Check new contents.
   检查新内容。
   $ monmaptool --print {tmp}/{filename}

   monmaptool: monmap file {tmp}/{filename}
   epoch 1
   fsid 224e376d-c5fe-4504-96bb-ea6332a19e61
   last_changed 2012-12-17 02:46:41.591248
   created 2012-12-17 02:46:41.591248
   0: 10.1.0.1:6789/0 mon.a
   1: 10.1.0.2:6789/0 mon.b
   2: 10.1.0.3:6789/0 mon.c

At this point, we assume the monitors (and stores) are installed at the new location. The next step is to propagate the modified monmap to the new monitors, and inject the modified monmap into each new monitor.
从这里开始，假设监视器（及存储）已经被安装到了新位置。下一步把修正的 monmap 散播到新监视器，并且注入每个监视器。

1. First, make sure to stop all your monitors. Injection must be done while the daemon is not running.
   首先，停止所有监视器，注入必须在守护进程停止时进行。
2. Inject the monmap.
   注入 monmap。
   ceph-mon -i {mon-id} --inject-monmap {tmp}/{filename}
3. Restart the monitors.

重启监视器。
After this step, migration to the new location is complete and the monitors should operate successfully.
到这里，到新位置的迁移完成，监视器应该照常运行了。

3.3.15 控制命令

Control commands

3.3.15.1 监视器命令

MONITOR COMMANDS
Monitor commands are issued using the ceph utility:
监视器命令用 ceph 工具执行：

```
ceph [-m monhost] {command}
```

The command is usually (though not always) of the form:
命令格式通常是（不总是）：

```
ceph {subsystem} {command}
```

3.3.15.2 系统命令

SYSTEM COMMANDS
Execute the following to display the current status of the cluster.
下列命令显示集群状态：

```
ceph -s
ceph status
```

Execute the following to display a running summary of the status of the cluster, and major events.
下列命令显示集群状态的运行摘要、及主要事件：

```
ceph -w
```

Execute the following to show the monitor quorum, including which monitors are participating and which one is the leader.
下列命令显示监视器法定人数状态，包括哪些监视器参与着、哪个是首领。

```
ceph quorum_status
```

Execute the following to query the status of a single monitor, including whether or not it is in the quorum.
下列命令查询单个监视器状态，包括是否在法定人数里。

```
ceph [-m monhost] mon_status
```

3.3.15.3 认证子系统

AUTHENTICATION SUBSYSTEM
To add a keyring for an OSD, execute the following:
要添加一个 OSD 的密钥环，执行下列命令：

```
ceph auth add {osd} {--in-file|-i} {path-to-osd-keyring}
```

To list the cluster's keys and their capabilities, execute the following:
要列出集群的密钥及其能力，执行下列命令：

```
ceph auth list
```

3.3.15.4 归置组子系统

PLACEMENT GROUP SUBSYSTEM
To display the statistics for all placement groups, execute the following:
要显示所有归置组的统计信息，执行下列命令：

```
ceph -- pg dump [--format {format}]
```

The valid formats are plain (default) and json.
可用输出格式有纯文本（默认）和 json。

To display the statistics for all placement groups stuck in a specified state, execute the following:
要显示卡在某状态的所有归置组，执行下列命令：

```
ceph -- pg dump_stuck inactive|unclean|stale [--format {format}] [-t|--threshold {seconds}]
```

--format may be plain (default) or json
--threshold defines how many seconds “stuck” is (default: 300)
--format 可以是 plain（默认）或 json
--threshold 定义了多久算“卡”（默认 300 秒）

Inactive Placement groups cannot process reads or writes because they are waiting for an OSD with the most up-to-date data to come back.
Unclean Placement groups contain objects that are not replicated the desired number of times. They should be recovering.
Stale Placement groups are in an unknown state - the OSDs that host them have not reported to the monitor cluster in a while (configured by mon_osd_report_timeout).
inactive 归置组不能处理读或写，因为它们在等待数据及时更新的 OSD 回来。
unclean 归置组包含副本数未达期望值的对象，它们应该在恢复中。
stale 归置组处于未知状态——归置组所托付的 OSD 有一阵没向监视器报告了（由 mon osd report timeout 配置）。

Revert “lost” objects to their prior state, either a previous version or delete them if they were just created.
把“丢失”对象恢复到其先前状态，可以是前一版本、或如果刚创建就干脆删除。

```
ceph pg {pgid} mark_unfound_lost revert
```

3.3.15.5 OSD 子系统

OSD SUBSYSTEM
Query osd subsystem status.
查询 OSD 子系统状态：

```
ceph osd stat
```

Write a copy of the most recent osd map to a file. See osdmaptool.
把最新的 OSD 运行图拷贝到一个文件，参见 osdmaptool：

```
ceph osd getmap -o file
```

Write a copy of the crush map from the most recent osd map to file.
从最新 OSD 运行图拷出 CRUSH 图：

```
ceph osd getcrushmap -o file
```

The foregoing functionally equivalent to
前述功能等价于：

```
ceph osd getmap -o /tmp/osdmap
osdmaptool /tmp/osdmap --export-crush file
```

Dump the OSD map. Valid formats for -f are plain and json. If no --format option is given, the OSD map is dumped as plain text.
转储 OSD 运行图，-f 的可用格式有 plain 和 json，如未指定--format 则转储为纯文本。

```
ceph osd dump [--format {format}]
```

Dump the OSD map as a tree with one line per OSD containing weight and state.
把 OSD 运行图转储为树，每个 OSD 一行、包含权重和状态。

```
ceph osd tree [--format {format}]
```

Find out where a specific object is or would be stored in the system:
找出某对象在哪里或应该在哪里：

```
ceph osd map <pool-name> <object-name>
```

Add or move a new item (OSD) with the given id/name/weight at the specified location.
增加或挪动一个新 OSD 条目，要给出 id/name/weight、和位置参数。

```
ceph osd crush set {id} {weight} [{loc1} [{loc2} ...]]
```

Remove an existing item from the CRUSH map.
从现有 CRUSH 图删除存在的条目：

```
ceph osd crush remove {id}
```

Move an existing bucket from one position in the hierarchy to another.
把有效的桶从分级结构里的一个位置挪到另一个：

```
ceph osd crush move {id} {loc1} [{loc2} ...]
```

Set the weight of the item given by {name} to {weight}.
设置{name}所指条目的权重为{weight}：

```
ceph osd crush reweight {name} {weight}
```

Create a cluster snapshot.
创建集群快照：

```
ceph osd cluster_snap {name}
```

Mark an OSD as lost. This may result in permanent data loss. Use with caution.
把 OSD 标记为丢失，有可能导致永久性数据丢失，慎用！

```
ceph osd lost [--yes-i-really-mean-it]
```

Create a new OSD. If no ID is given, a new ID is automatically selected if possible.
创建新 OSD。如果未指定 ID，有可能的话将自动分配个新 ID：

```
ceph osd create [{id}]
```

Remove the given OSD(s).
删除指定 OSD：

```
ceph osd rm [{id}...]
```

Query the current max_osd parameter in the osd map.
查询 OSD 运行图里的 max_osd 参数。

```
ceph osd getmaxosd
```

Import the given OSD map. Note that this can be a bit dangerous, since the OSD map includes dynamic state about which OSDs are current on or offline; only do this if you've just modified a (very) recent copy of the map.
导入指定 OSD 运行图。注意，此动作有风险，因为 OSD 运行图包含当前 OSD 在线、离线的动态状态，只可用于相当新的副本。

```
ceph osd setmap -i file
```

Import the given crush map.
导入指定 CRUSH 图。

```
ceph osd setcrushmap -i file
```

Set the max_osd parameter in the OSD map. This is necessary when expanding the storage cluster.
设置 OSD 运行图的 max_osd 参数，扩展存储集群时有必要。

```
ceph osd setmaxosd
```

Mark OSD {osd-num} down.
把 ID 为{osd-num}的 OSD 标记为 down。

```
ceph osd down {osd-num}
```

Mark OSD {osd-num} out of the distribution (i.e. allocated no data).
把 OSD {osd-num}标记为数据分布之外（如不给分配数据）。

```
ceph osd out {osd-num}
```

Mark {osd-num} in the distribution (i.e. allocated data).
把 OSD {osd-num}标记为数据分布之内（如分配了数据）。

```
ceph osd in {osd-num}
```

List classes that are loaded in the ceph cluster.
列出 ceph 集群载入的类:

```
ceph class list
```

Set or clear the pause flags in the OSD map. If set, no IO requests will be sent to any OSD. Clearing the flags via unpause results in resending pending requests.
设置或清空 OSD 运行图里的暂停标记。若设置了，不会有 IO 请求发送到任何 OSD；用 unpause 清空此标记会导致重发未决的请求。

```
ceph osd pause
ceph osd unpause
```

Set the weight of {osd-num} to {weight}. Two OSDs with the same weight will receive roughly the same number of I/O requests and store approximately the same amount of data.
把{osd-num}的权重设置为{weight}，权重相同的两个 OSD 大致会收到相同的 I/O 请求、并存储相同数量的数据。

```
ceph osd reweight {osd-num} {weight}
```

Reweights all the OSDs by reducing the weight of OSDs which are heavily overused. By default it will adjust the weights downward on OSDs which have 120% of the average utilization, but if you include threshold it will use that percentage instead.
重设所有滥用 OSD 的权重，它默认向下调整达到 120%利用率的 OSD，除非你指定了 threshold 值。

```
ceph osd reweight-by-utilization [threshold]
```

Adds/removes the address to/from the blacklist. When adding an address, you can specify how long it should be blacklisted in seconds; otherwise, it will default to 1 hour. A blacklisted address is prevented from connecting to any OSD. Blacklisting is most often used to prevent a lagging metadata server from making bad changes to data on the OSDs.
增加、删除黑名单里的地址。增加地址的时候可以指定有效期，否则有效期为 1 小时。黑名单里的地址不允许连接任何 OSD，此技术常用于防止滞后的元数据服务器“错爱”OSD 上的数据。

These commands are mostly only useful for failure testing, as blacklists are normally maintained automatically and shouldn't need manual intervention.
这些命令大多只在故障测试时有用，因为黑名单是自动维护的，无需手动干涉。

```
ceph osd blacklist add ADDRESS[:source_port] [TIME]
ceph osd blacklist rm ADDRESS[:source_port]
```

Creates/deletes a snapshot of a pool.
创建/删除存储池快照。

```
ceph osd pool mksnap {pool-name} {snap-name}
```

```
ceph osd pool rmsnap {pool-name} {snap-name}
```

Creates/deletes/renames a storage pool.
创建/删除/重命名存储池。

```
ceph osd pool create {pool-name} pg_num [pgp_num]
ceph osd pool delete {pool-name}
ceph osd pool rename {old-name} {new-name}
```

Changes a pool setting.
更改存储池设置。

```
ceph osd pool set {pool-name} {field} {value}
```

Valid fields are:
size: Sets the number of copies of data in the pool.
crash_replay_interval: The number of seconds to allow clients to replay acknowledged but uncommited requests.
pg_num: The placement group number.
pgp_num: Effective number when calculating pg placement.
crush_ruleset: rule number for mapping placement.
可用的 field 有:
size ——设置存储池内数据的副本数;
crash_replay_interval ——允许客户端重放确认而未提交的请求前等待的时间，秒;
pg_num ——归置组数量;
pgp_num ——计算归置组存放的有效数量;
crush_ruleset ——用于归置映射的规则号。

Get the value of a pool setting.
获取存储池配置值。

```
ceph osd pool get {pool-name} {field}
```

Valid fields are:
pg_num: The placement group number.
pgp_num: Effective number of placement groups when calculating placement.
lpg_num: The number of local placement groups.
lpgp_num: The number used for placing the local placement groups.
可用的 field 有:
pg_num ——归置组数量;
pgp_num ——计算归置组存放的有效数量;
lpg_num ——本地归置组数量;
lpgp_num ——用于存放本地归置组的数量。

Sends a scrub command to OSD {osd-num}. To send the command to all OSDs, use *.
向 OSD {osd-num}下达一个洗刷命令，用通配符*把命令下达到所有 OSD。

```
ceph osd scrub {osd-num}
```

Sends a repair command to osdN. To send the command to all osds, use *.
向 osdN 下达修复命令，用*下达到所有 OSD。

```
ceph osd repair N
```

Runs a simple throughput benchmark against osdN, writing TOTAL_BYTES in write requests of BYTES_PER_WRITE each. By default, the test writes 1 GB in total in 4-MB increments.
在 osdN 上进行个简单的吞吐量测试，每次写入 BYTES_PER_WRITE、一共写入 TOTAL_BYTES。默认以 4MB 增量写入 1GB。

```
ceph osd tell N bench [BYTES_PER_WRITE] [TOTAL_BYTES]
```

3.3.15.6 MDS 子系统

MDS SUBSYSTEM
Change configuration parameters on a running mds.
更改在运行 mds 的参数:

```
ceph mds tell {mds-id} injectargs '--{switch} {value} [--{switch} {value}]'
```

Example:
例如:

```
ceph mds tell 0 injectargs '--debug_ms 1 --debug_mds 10'
```

Enables debug messages.
打开了调试消息。

```
ceph mds stat
```

Displays the status of all metadata servers.
显示所有元数据服务器状态。

Todo ceph mds subcommands missing docs: set_max_mds, dump, getmap, stop, setmap
待完成：ceph mds 子命令缺少文档：set_max_mds、dump、getmap、stop、setmap。

3.3.15.7 监视器子系统

MON SUBSYSTEM
Show monitor stats:
查看监视器状态:

```
ceph mon stat
2011-12-14 10:40:59.044395 mon {- [mon,stat]
2011-12-14 10:40:59.057111 mon.1 -} 'e3: 5 mons at
{a=10.1.2.3:6789/0,b=10.1.2.4:6789/0,c=10.1.2.5:6789/0,d=10.1.2.6:6789/0,e=10.1.2.7:6789/0},
election epoch 16, quorum 0,1,2,3' (0)
```

The quorum list at the end lists monitor nodes that are part of the current quorum.
末尾的法定人数列表列出了当前法定人数里的监视器节点。

This is also available more directly:
也可以更直接地获取:

```
$ ./ceph quorum_status
2011-12-14 10:44:20.417705 mon {- [quorum_status]
2011-12-14 10:44:20.431890 mon.0 -}
'{ "election_epoch": 10,
  "quorum": [
        0,
        1,
        2],
  "monmap": { "epoch": 1,
      "fsid": "444b489c-4f16-4b75-83f0-cb8097468898",
      "modified": "2011-12-12 13:28:27.505520",
      "created": "2011-12-12 13:28:27.505520",
      "mons": [
            { "rank": 0,
              "name": "a",
              "addr": "127.0.0.1:6789\/0"},
            { "rank": 1,
              "name": "b",
              "addr": "127.0.0.1:6790\/0"},
            { "rank": 2,
              "name": "c",
              "addr": "127.0.0.1:6791\/0"}]}}' (0)
```

The above will block until a quorum is reached.
如果法定人数未形成，上述命令会一直等待。

For a status of just the monitor you connect to (use -m HOST:PORT to select):
你刚刚连接的监视器的状态（用-m HOST:PORT 另外指定）：

```
ceph mon_status
2011-12-14 10:45:30.644414 mon {- [mon_status]
2011-12-14 10:45:30.644632 mon.0 -}
'{ "name": "a",
  "rank": 0,
  "state": "leader",
  "election_epoch": 10,
  "quorum": [
        0,
        1,
        2],
  "outside_quorum": [],
  "monmap": { "epoch": 1,
      "fsid": "444b489c-4f16-4b75-83f0-cb8097468898",
      "modified": "2011-12-12 13:28:27.505520",
      "created": "2011-12-12 13:28:27.505520",
      "mons": [
            { "rank": 0,
              "name": "a",
              "addr": "127.0.0.1:6789\/0"},
            { "rank": 1,
              "name": "b",
              "addr": "127.0.0.1:6790\/0"},
            { "rank": 2,
              "name": "c",
              "addr": "127.0.0.1:6791\/0"}]}}' (0)
```

A dump of the monitor state:
监视器状态转储:

```
ceph mon dump
2011-12-14 10:43:08.015333 mon {- [mon,dump]
2011-12-14 10:43:08.015567 mon.0 -} 'dumped monmap epoch 1' (0)
epoch 1
fsid 444b489c-4f16-4b75-83f0-cb8097468898
last_changed 2011-12-12 13:28:27.505520
created 2011-12-12 13:28:27.505520
0: 127.0.0.1:6789/0 mon.a
1: 127.0.0.1:6790/0 mon.b
2: 127.0.0.1:6791/0 mon.c
```

3.4 手册页

Manual Pages

3.5 API 接口

APIS

Most Ceph deployments use Ceph block devices, the gateway and/or the CephFS filesystem. You may also develop applications that talk directly to the Ceph object store.
大多数 ceph 部署都要用到 ceph 块设备、网关和/或 CephFS 文件系统，你也可以开发程序直接和 ceph 对象存储交互。

RADOS object store APIs
RADOS 对象存储 API
Ceph's RADOS Object Store has a messaging layer protocol that enables clients to interact with Ceph monitors and OSDs. librados provides this functionality to object store clients in the form of a library. All Ceph clients either use librados or the same functionality encapsulated in librados to interact with the object store. For example, librbd and libcephfs leverage this functionality. You may use librados to interact with Ceph directly (e.g., an application that talks to Ceph, your own interface to Ceph, etc.).
ceph 的 RADOS 对象存储提供了消息传递层协议，用于客户端和 ceph 监视器和 OSD 交互，librados 以库形式为对象存储客户端提供了这个功能。所有 ceph 客户端可以用 librados 或 librados 里封装的相同功能和对象存储交互，例如 librbd 和 libcephfs 就利用了此功能。你可以用 librados 直接和 ceph 交互（如和 ceph 兼容的应用程序、你自己的 ceph 接口、等等）。

For an overview of where librados appears in the technology stack, see Architecture.

要大致了解 librados 在技术栈里的地位，参见 Architecture。

3.5.1 librados (C)

librados provides low-level access to the RADOS service. For an overview of RADOS, see Architecture.
librados 提供了 RADOS 服务的底层访问功能，RADOS 概览参见 Architecture。

3.5.1.1 实例：连接并写入一个对象

Example: connecting and writing an object
To use Librados, you instantiate a rados_t variable (a cluster handle) and call rados_create() with a pointer to it:
要使用 librados，先实例化一个 rados_t 变量（集群句柄）、再用指向它的指针调用 rados_create()：

```
int err;
rados_t cluster;

err = rados_create(&cluster, NULL);
if (err < 0) {
        fprintf(stderr, "%s: cannot create a cluster handle: %s\n", argv[0], strerror(-err));
        exit(1);
}
```

Then you configure your rados_t to connect to your cluster, either by setting individual values (rados_conf_set()), using a configuration file (rados_conf_read_file()), using command line options (rados_conf_parse_argv()), or an environment variable (rados_conf_parse_env()):
然后配置 rados_t 以连接集群，可以挨个设置值（rados_conf_set()）、或者用配置文件（rados_conf_read_file()）、或者用命令行选项（rados_conf_parse_argv()）、或者环境变量（rados_conf_parse_env()）：

```
err = rados_conf_read_file(cluster, "/path/to/myceph.conf");
if (err < 0) {
        fprintf(stderr, "%s: cannot read config file: %s\n", argv[0], strerror(-err));
        exit(1);
}
```

Once the cluster handle is configured, you can connect to the cluster with rados_connect():
集群句柄配置好后，就可以用 rados_connect()连接了：

```
err = rados_connect(cluster);
if (err < 0) {
        fprintf(stderr, "%s: cannot connect to cluster: %s\n", argv[0], strerror(-err));
        exit(1);
}
```

Then you open an “IO context”, a rados_ioctx_t, with rados_ioctx_create():
然后打开一个"IO 上下文"，用 rados_ioctx_create()打开 rados_ioctx_t：

```
rados_ioctx_t io;
char *poolname = "mypool";

err = rados_ioctx_create(cluster, poolname, &io);
if (err < 0) {
        fprintf(stderr, "%s: cannot open rados pool %s: %s\n", argv[0], poolname, strerror(-err));
        rados_shutdown(cluster);
        exit(1);
}
```

Note that the pool you try to access must exist.
注意，你访问的存储池必须存在。

Then you can use the RADOS data manipulation functions, for example write into an object called greeting with rados_write_full():
这时你能用 RADOS 数据修改函数了，例如用 rados_write_full()写入名为 greeting 的对象：

```
err = rados_write_full(io, "greeting", "hello", 5);
if (err < 0) {
        fprintf(stderr, "%s: cannot write pool %s: %s\n", argv[0], poolname, strerror(-err));
```

```
        rados_ioctx_destroy(io);
        rados_shutdown(cluster);
        exit(1);
}
```

In the end, you'll want to close your IO context and connection to RADOS with rados_ioctx_destroy() and rados_shutdown():
最后，用 rados_ioctx_destroy()和 rados_shutdown()分别关闭 IO 上下文、到 RADOS 的连接。

```
rados_ioctx_destroy(io);
rados_shutdown(cluster);
```

3.5.1.2 异步IO

Asychronous IO
When doing lots of IO, you often don't need to wait for one operation to complete before starting the next one. Librados provides asynchronous versions of several operations:
处理大量 IO 时，通常不必等一个完成再开始下一个。librados 提供了几种操作的异步版本：

```
rados_aio_write()
rados_aio_append()
rados_aio_write_full()
rados_aio_read()
```

For each operation, you must first create a rados_completion_t that represents what to do when the operation is safe or complete by calling rados_aio_create_completion(). If you don't need anything special to happen, you can pass NULL:
对每种操作，都必须先创建一个 rados_completion_t 数据结构来表达做什么、何时安全或显式地调用 rados_aio_create_completion()来结束，如果没什么特殊需求，可以仅传递 NULL：

```
rados_completion_t comp;
err = rados_aio_create_completion(NULL, NULL, NULL, &comp);
if (err < 0) {
        fprintf(stderr, "%s: could not create aio completion: %s\n", argv[0], strerror(-err));
        rados_ioctx_destroy(io);
        rados_shutdown(cluster);
        exit(1);
}
```

Now you can call any of the aio operations, and wait for it to be in memory or on disk on all replicas:
现在你可以调用任意一种 aio 操作了，然后等它出现在内存、所有复制所在的硬盘里：

```
err = rados_aio_write(io, "foo", comp, "bar", 3, 0);
if (err < 0) {
        fprintf(stderr, "%s: could not schedule aio write: %s\n", argv[0], strerror(-err));
        rados_aio_release(comp);
        rados_ioctx_destroy(io);
        rados_shutdown(cluster);
        exit(1);
}
rados_wait_for_complete(comp); // in memory
rados_wait_for_safe(comp); // on disk
```

Finally, we need to free the memory used by the completion with rados_aio_release():
最后，用 rados_aio_release()释放内存：

```
rados_aio_release(comp);
```

You can use the callbacks to tell your application when writes are durable, or when read buffers are full. For example, if you wanted to measure the latency of each operation when appending to several objects, you could schedule several writes and store the ack and commit time in the corresponding callback, then wait for all of them to complete using rados_aio_flush() before analyzing the latencies:
你可以用 callback 告知应用程序何时可以持续写入、或何时读缓冲是满的。例如，如果你追加几个对象时想衡量每个操作的延时，可以调度几个写操作、并把确认和提交时间保存到相应 callback，然后用 rados_aio_flush()等它们完成，然后就可以分析延时了：

```
typedef struct {
        struct timeval start;
```

```
        struct timeval ack_end;
        struct timeval commit_end;
} req_duration;

void ack_callback(rados_completion_t comp, void *arg) {
        req_duration *dur = (req_duration *) arg;
        gettimeofday(&dur->ack_end, NULL);
}

void commit_callback(rados_completion_t comp, void *arg) {
        req_duration *dur = (req_duration *) arg;
        gettimeofday(&dur->commit_end, NULL);
}

int output_append_latency(rados_ioctx_t io, const char *data, size_t len, size_t num_writes) {
        req_duration times[num_writes];
        rados_completion_t comps[num_writes];
        for (size_t i = 0; i < num_writes; ++i) {
                gettimeofday(&times[i].start, NULL);
                int err = rados_aio_create_completion((void*) &times[i], ack_callback,
commit_callback, &comps[i]);
                if (err < 0) {
                        fprintf(stderr, "Error creating rados completion: %s\n", strerror(-err));
                        return err;
                }
                char obj_name[100];
                snprintf(obj_name, sizeof(obj_name), "foo%ld", (unsigned long)i);
                err = rados_aio_append(io, obj_name, comps[i], data, len);
                if (err < 0) {
                        fprintf(stderr, "Error from rados_aio_append: %s", strerror(-err));
                        return err;
                }
        }
        // wait until all requests finish *and* the callbacks complete
        rados_aio_flush(io);
        // the latencies can now be analyzed
        printf("Request # | Ack latency (s) | Commit latency (s)\n");
        for (size_t i = 0; i < num_writes; ++i) {
                // don't forget to free the completions
                rados_aio_release(comps[i]);
                struct timeval ack_lat, commit_lat;
                timersub(&times[i].ack_end, &times[i].start, &ack_lat);
                timersub(&times[i].commit_end, &times[i].start, &commit_lat);
                printf("%9ld | %8ld.%06ld | %10ld.%06ld\n", (unsigned long) i, ack_lat.tv_sec,
ack_lat.tv_usec, commit_lat.tv_sec, commit_lat.tv_usec);
        }
        return 0;
}
```

Note that all the rados_completion_t must be freed with rados_aio_release() to avoid leaking memory.
注意，所有 rados_completion_t 都必须用 rados_aio_release()释放内存，以免造成内存泄漏。

3.5.1.3 API 调用

API calls

3.5.1.3.1 rados_pool_stat_t 数据结构

STRUCT RADOS_POOL_STAT_T
struct **rados_pool_stat_t**
Usage information for a pool.
存储池用法信息。

成员

MEMBERS
uint64_t **num_bytes**
space used in bytes
已用空间，字节数

uint64_t **num_kb**
space used in KB

已用空间，KB；

uint64_t **num_objects**
number of objects in the pool
存储池里的对象数量；

uint64_t **num_object_clones**
number of clones of objects
对象的克隆数量；

uint64_t **num_object_copies**
num_objects * num_replicas

uint64_t **num_objects_missing_on_primary**
uint64_t **num_objects_unfound**
number of objects found on no OSDs
未在 OSD 上找到的对象数量；

uint64_t **num_objects_degraded**
number of objects replicated fewer times than they should be (but found on at least one OSD)
复制次数小于规定值的对象数量（但是发现于至少一个 OSD）

uint64_t **num_rd**
uint64_t **num_rd_kb**
uint64_t **num_wr**
uint64_t **num_wr_kb**

3.5.1.3.2 rados_cluster_stat_t 数据结构

STRUCT RADOS_CLUSTER_STAT_T
struct **rados_cluster_stat_t**
Cluster-wide usage information.
集群范畴的用法信息。

成员

MEMBERS

uint64_t **kb**
uint64_t **kb_used**
uint64_t **kb_avail**
uint64_t **num_objects**

3.5.1.3.3 定义

DEFINES

CEPH_OSD_TMAP_HDR

CEPH_OSD_TMAP_SET

CEPH_OSD_TMAP_CREATE

CEPH_OSD_TMAP_RM

LIBRADOS_VER_MAJOR

LIBRADOS_VER_MINOR

LIBRADOS_VER_EXTRA

LIBRADOS_VERSION

LIBRADOS_VERSION_CODE

LIBRADOS_SUPPORTS_WATCH

3.5.1.3.4 类

TYPES

rados_t

A handle for interacting with a RADOS cluster.
It encapsulates all RADOS client configuration, including username, key for authentication, logging, and debugging. Talking different clusters – or to the same cluster with different users – requires different cluster handles.
和 RADOS 集群交互的句柄。
它封装了所有 RADOS 客户端配置，包括用户名、认证的密钥、日志、调试的配置。与不同集群交互、或以不同用户身份访问同一集群要用不同句柄。

rados_config_t

A handle for the ceph configuration context for the rados_t cluster instance. This can be used to share configuration context/state (e.g., logging configuration) between librados instance.
rados_t 集群例程的配置上下文句柄，可用于 librados 之间共享配置内容/状态（例如日志配置）。

Warning The config context does not have independent reference counting. As such, a rados_config_t handle retrieved from a given rados_t is only valid as long as that rados_t.
警告：配置上下文没有独立的引用计数，这样，从某 rados_t 获取的 rados_config_t 句柄仅在 rados_t 存活的时候有效。

rados_ioctx_t

An io context encapsulates a few settings for all I/O operations done on it:
一个 IO 上下文封装了所有和其相关的 I/O 操作的一些配置：

- pool - set when the io context is created (see rados_ioctx_create())
- snapshot context for writes (see rados_ioctx_selfmanaged_snap_set_write_ctx())
- snapshot id to read from (see rados_ioctx_snap_set_read())
- object locator for all single-object operations (see rados_ioctx_locator_set_key())
- 存储池——上下文创建时就设置了（见 rados_ioctx_create()）
- 用于写的快照上下文（见 rados_ioctx_selfmanaged_snap_set_write_ctx()）
- 要读的快照 ID（见 rados_ioctx_snap_set_read()）
- 用于所有单对象操作的对象定位器（见 rados_ioctx_locator_set_key()）

Warning changing any of these settings is not thread-safe - librados users must synchronize any of these changes on their own, or use separate io contexts for each thread.
警告：这些设置的更改不是线程安全的，librados 用户必须自己同步任何更改、或为每个线程使用单独的 IO 上下文。

rados_list_ctx_t

An iterator for listing the objects in a pool.
Used with rados_objects_list_open(),rados_objects_list_next(), andrados_objects_list_close().
用于列出存储池中对象的迭代器。
和 rados_objects_list_open()、rados_objects_list_next()、andrados_objects_list_close()一起使用。

rados_snap_t

The id of a snapshot.
快照的 ID。

rados_xattrs_iter_t

An iterator for listing extended attrbutes on an object.
Used with rados_getxattrs(),rados_getxattrs_next(), andrados_getxattrs_end().
列出一个对象扩展属性的迭代器。
和 rados_getxattrs()、rados_getxattrs_next()、andrados_getxattrs_end()一起使用。

rados_completion_t

Represents the state of an asynchronous operation - it contains the return value once the operation completes, and can be used to block until the operation is complete or safe.

表达异步操作状态，操作完成后会包含返回值，也可用于阻塞，直到操作完成或变安全了。

rados_callback_t

Callbacks for asynchrous operations, take two parameters:

异步操作回调，需 2 个参数：

- cb the completion that has finished
- arg application defined data made available to the callback function
- cb 刚完成的操作；
- arg 回调函数可访问的应用程序数据

rados_watchcb_t

Callback activated when a notify is received on a watched object.

关于被监视对象的通知收到时，激活回调。

Parameters are:

参数有：

- opcode undefined
- ver version of the watched object
- arg application-specific data
- opcode 未定义；
- ver 被监视对象的版本；
- arg 具体应用程序数据。

Note BUG: opcode is an internal detail that shouldn't be exposed

注意：缺陷：opcode 是不应该暴露的内部细节。

3.5.1.3.5 函数

FUNCTIONS

void **rados_version**(int *major, int *minor, int *extra)

Get the version of librados.

The version number is major.minor.extra. Note that this is unrelated to the Ceph version number.

TODO: define version semantics, i.e.:

获取 librados 版本。

版本号依次为：主.次.额外。注意这和 ceph 版本无关。

待完成：定义版本语义，例如：

- incrementing major is for backwards-incompatible changes
- incrementing minor is for backwards-compatible changes
- incrementing extra is for bug fixes
- 递增的主版本是向后不兼容的变更；
- 递增的次版本是向后兼容的变更；
- 递增的额外版本是缺陷修正。

Parameters:

- major – where to store the major version number
- minor – where to store the minor version number
- extra – where to store the extra version number

int **rados_create**(rados_t *cluster, const char *const id)

Create a handle for communicating with a RADOS cluster.

创建句柄用于和 RADOS 集群通信。

Ceph environment variables are read when this is called, so if $CEPH_ARGS specifies everything you

need to connect, no further configuration is necessary.
调用此函数时会读取 ceph 环境变量，所以如果$CEPH_ARGS 给足了用于连接的信息，其他配置就不必要了。

Parameters:
- cluster – where to store the handle
- id – the user to connect as (i.e. admin, not client.admin)

Returns:
- 0 on success, negative error code on failure

int **rados_create_with_context**(rados_t *cluster, rados_config_t cct)
Initialize a cluster handle from an existing configuration.
根据已有配置初始化一个集群句柄。

Share configuration state with another rados_t instance.
和另外一个 rados_t 例程共享配置状态。

Parameters:
- cluster – where to store the handle
- cct_ – the existing configuration to use

Returns:
- 0 on success, negative error code on failure

int **rados_connect**(rados_t cluster)
Connect to the cluster.
连接到集群。

Note BUG: Before calling this, calling a function that communicates with the cluster will crash.
注意：缺陷：调用此函数前，调用和集群通信的函数将崩溃。

Precondition: The cluster handle is configured with at least a monitor address. If cephx is enabled, a client name and secret must also be set.
Postcondition: If this succeeds, any function in librados may be used
前提条件：集群句柄必须配置至少一个监视器地址。如果启用了 cephx，客户端名字和密钥也必须设置。
后置条件：如果此函数成功了，librados 里的任何函数都可用。

Parameters:
- cluster – The cluster to connect to.

Returns:
- 0 on sucess, negative error code on failure

void **rados_shutdown**(rados_t cluster)
Disconnects from the cluster.
For clean up, this is only necessary after rados_connect()has succeeded.
断开集群连接。
要清理的话，这仅在 rados_connect()成功时必要。

Warning This does not guarantee any asynchronous writes have completed. To do that, you must call rados_aio_flush()on all open io contexts.
警告：此函数不保证任何异步写已完成。要确认，必须对所有已开 io 上下文调用 rados_aio_flush()。

Postcondition the cluster handle cannot be used again
后置条件：集群句柄不能再次使用。

Parameters:
- cluster – the cluster to shutdown

int **rados_conf_read_file**(rados_t cluster, const char *path)

Configure the cluster handle using a Ceph config file.

If path is NULL, the default locations are searched, and the first found is used. The locations are:

用配置文件配置集群句柄。

如果路径为空，就搜索默认路径，并使用第一个找到的。搜索位置有：

- $CEPH_CONF (environment variable)
- /etc/ceph/ceph.conf
- ~/.ceph/config
- ceph.conf (in the current working directory)

Precondition rados_connect()has not been called on the cluster handle

前提条件：集群句柄尚未调用 rados_connect()。

Parameters:

- cluster – cluster handle to configure
- path – path to a Ceph configuration file

Returns:

- 0 on success, negative error code on failure

int **rados_conf_parse_argv**(rados_t cluster, int argc, const char **argv)

Configure the cluster handle with command line arguments.

用命令行参数配置集群句柄。

argv can contain any common Ceph command line option, including any configuration parameter prefixed by '–' and replacing spaces with dashes or underscores. For example, the following options are equivalent:

argv 可包含任何通用的 ceph 命令行选项，包括任何以'-'打头的配置参数、和用连字符或下划线替代空格。例如，下列选项是等价的：

- --mon-host 10.0.0.1:6789
- --mon_host 10.0.0.1:6789
- -m 10.0.0.1:6789

Precondition rados_connect()has not been called on the cluster handle

前提条件：集群句柄尚未调用 rados_connect()。

Parameters:

- cluster – cluster handle to configure
- argc – number of arguments in argv
- argv – arguments to parse

Returns:

- 0 on success, negative error code on failure

int **rados_conf_parse_env**(rados_t cluster, const char *var)

Configure the cluster handle based on an environment variable.

The contents of the environment variable are parsed as if they were Ceph command line options. If var is NULL, the CEPH_ARGS environment variable is used.

用环境变量配置集群句柄。

如果环境变量内容像 ceph 命令行选项，它就会被分析。若 var 值为 NULL，就会用到 CEPH_ARGS 环境变量。

Precondition rados_connect()has not been called on the cluster handle

前提条件：集群句柄尚未调用 rados_connect()。

Note BUG: this is not threadsafe - it uses a static buffer

注意：缺陷：此函数不是线程安全的，它用静态缓冲区。

Parameters:

- cluster – cluster handle to configure
- var – name of the environment variable to read

Returns:

- 0 on success, negative error code on failure

int **rados_conf_set**(rados_t cluster, const char *option, const char *value)

Set a configuration option.

设置选项。

Precondition rados_connect()has not been called on the cluster handle

前提条件：集群句柄尚未调用 rados_connect()。

Parameters:

- cluster – cluster handle to configure
- option – option to set
- value – value of the option

Returns:

- 0 on success, negative error code on failure
- -ENOENT when the option is not a Ceph configuration option

int **rados_conf_get**(rados_t cluster, const char *option, char *buf, size_t len)

Get the value of a configuration option.

获取配置选项值。

Parameters:

- cluster – configuration to read
- option – which option to read
- buf – where to write the configuration value
- len – the size of buf in bytes

Returns:

- 0 on success, negative error code on failure
- -ENAMETOOLONG if the buffer is too short to contain the requested value

int **rados_cluster_stat**(rados_t cluster, struct rados_cluster_stat_t *result)

Read usage info about the cluster.

This tells you total space, space used, space available, and number of objects. These are not updated immediately when data is written, they are eventually consistent.

读取集群的用法信息。

它会告诉你总空间、已用空间、可用空间、和对象数量。这些不会在数据写入时立即更新，但最终会一致。

Parameters:

- cluster – cluster to query
- result – where to store the results

Returns:

- 0 on success, negative error code on failure

int **rados_cluster_fsid**(rados_t cluster, char *buf, size_t len)

Get the fsid of the cluster as a hexadecimal string.

The fsid is a unique id of an entire Ceph cluster.

获取集群的 fsid，十六进制格式。

fsid 是整个 ceph 集群的唯一 ID。

Parameters:

- cluster – where to get the fsid
- buf – where to write the fsid
- len – the size of buf in bytes (should be 37)

Returns:

- 0 on success, negative error code on failure
- -ERANGE if the buffer is too short to contain the fsid

int **rados_pool_list**(rados_t cluster, char *buf, size_t len)

List objects in a pool.

Gets a list of pool names as NULL-terminated strings. The pool names will be placed in the supplied buffer one after another. After the last pool name, there will be two 0 bytes in a row.

If len is too short to fit all the pool name entries we need, we will fill as much as we can.

列出存储池里的对象。

获取存储池列表，它将以 NULL 结尾的字符串挨个放置于指定缓冲区。最后一个存储池名字后面会有一行、2 个 0 字节。

如果 len 太小，放不下存储池名称，它会尽可能多地填充。

Parameters:

- cluster – cluster handle
- buf – output buffer
- len – output buffer length

Returns:

- length of the buffer we would need to list all pools

rados_config_t **rados_cct**(rados_t cluster)

Get a configuration handle for a rados cluster handle.

This handle is valid only as long as the cluster handle is valid.

为 rados 集群句柄获取一个配置句柄。

仅在集群句柄有效时此句柄才有效。

Parameters:

- cluster – cluster handle

Returns:

- config handle for this cluster

uint64_t **rados_get_instance_id**(rados_t cluster)

Get a global id for current instance.

This id is a unique representation of current connection to the cluster

为当前例程申请个全局 ID。

此 ID 是当前和集群连接的唯一标识符。

Parameters:

- cluster – cluster handle

Returns:

- instance global id

int **rados_ioctx_create**(rados_t cluster, const char *pool_name, rados_ioctx_t *ioctx)

Create an io context.

The io context allows you to perform operations within a particular pool. For more details see rados_ioctx_t.

创建一个 io 上下文。

此 io 上下文允许你在特定存储池内进行操作，更多细节见 rados_ioctx_t。

Parameters:

- cluster – which cluster the pool is in
- pool_name – name of the pool
- ioctx – where to store the io context

Returns:

- 0 on success, negative error code on failure

void **rados_ioctx_destroy**(rados_ioctx_t io)

The opposite of rados_ioctx_create.

This just tells librados that you no longer need to use the io context. It may not be freed immediately if there are pending asynchronous requests on it, but you should not use an io context again after calling this function on it.

和 rados_ioctx_create 相反。
这只是告诉 librados 你不再需要 io 上下文了。如果还有关于它的未决异步请求，它就不会被立即释放，但是调用此函数后就不应该再在其上使用 io 上下文了。

Warning This does not guarantee any asynchronous writes have completed. You must call rados_aio_flush()on the io context before destroying it to do that.
警告：这不保证任何异步写已完成，要那样做，就得在杀死 io 上下文前先在其上调用 rados_aio_flush()。

Parameters:
- io – the io context to dispose of

rados_config_t **rados_ioctx_cct**(rados_ioctx_t io)
Get configuration handle for a pool handle.
为一个存储池获取配置句柄。

Parameters:
- io – pool handle

Returns:
- rados_config_t for this cluster

rados_t **rados_ioctx_get_cluster**(rados_ioctx_t io)
Get the cluster handle used by this rados_ioctx_t. Note that this is a weak reference, and should not be destroyed via rados_destroy().
获取 rados_ioctx_t 用着的集群句柄。注意这是一个弱引用，不应该通过 rados_destroy()杀死。

Parameters:
- io – the io context

Returns:
- the cluster handle for this io context

int **rados_ioctx_pool_stat**(rados_ioctx_t io, struct rados_pool_stat_t *stats)
Get pool usage statistics.
Fills in a rados_pool_stat_t after querying the cluster.
获取存储池利用统计信息。
查询集群后填充 rados_pool_stat_t。

Parameters:
- io – determines which pool to query
- stats – where to store the results

Returns:
- 0 on success, negative error code on failure

int64_t **rados_pool_lookup**(rados_t cluster, const char *pool_name)
Get the id of a pool.
获取存储池的 ID。

Parameters:
- cluster – which cluster the pool is in
- pool_name – which pool to look up

Returns:
- id of the pool
- -ENOENT if the pool is not found

int **rados_pool_reverse_lookup**(rados_t cluster, int64_t id, char *buf, size_t maxlen)
Get the name of a pool.
获取存储池名字。

Parameters:

- cluster – which cluster the pool is in
- id – the id of the pool
- buf – where to store the pool name
- maxlen – size of buffer where name will be stored

Returns:

- length of string stored, or -ERANGE if buffer too small

int **rados_pool_create**(rados_t cluster, const char *pool_name)

Create a pool with default settings.

The default owner is the admin user (auid 0). The default crush rule is rule 0.

用默认配置创建一个存储池。

默认所有者是 admin 用户（auid 0），默认 crush 规则是 rule 0。

Parameters:

- cluster – the cluster in which the pool will be created
- pool_name – the name of the new pool

Returns:

- 0 on success, negative error code on failure

int **rados_pool_create_with_auid**(rados_t cluster, const char *pool_name, uint64_t auid)

Create a pool owned by a specific auid.

The auid is the authenticated user id to give ownership of the pool. TODO: document auid and the rest of the auth system

创建所有者为指定 auid 的存储池。

auid 是存储池所有者的已认证用户 ID。待完成：写作认证系统的 auid 和及其它。

Parameters:

- cluster – the cluster in which the pool will be created
- pool_name – the name of the new pool
- auid – the id of the owner of the new pool

Returns:

- 0 on success, negative error code on failure

int **rados_pool_create_with_crush_rule**(rados_t cluster, const char *pool_name, __u8 crush_rule_num)

Create a pool with a specific CRUSH rule.

创建有指定 CRUSH 规则的存储池。

Parameters:

- cluster – the cluster in which the pool will be created
- pool_name – the name of the new pool
- crush_rule_num – which rule to use for placement in the new pool1

Returns:

- 0 on success, negative error code on failure

int **rados_pool_create_with_all**(rados_t cluster, const char *pool_name, uint64_t auid, __u8 crush_rule_num)

Create a pool with a specific CRUSH rule and auid.

This is a combination of rados_pool_create_with_crush_rule() and rados_pool_create_with_auid().

创建有指定 CRUSH 规则和 auid 的存储池。

这是 rados_pool_create_with_crush_rule()和 rados_pool_create_with_auid()的组合。

Parameters:

- cluster – the cluster in which the pool will be created
- pool_name – the name of the new pool
- crush_rule_num – which rule to use for placement in the new pool2
- auid – the id of the owner of the new pool

Returns:

- 0 on success, negative error code on failure

int **rados_pool_delete**(rados_t cluster, const char *pool_name)

Delete a pool and all data inside it.

The pool is removed from the cluster immediately, but the actual data is deleted in the background.

删除存储池及其内所有数据。

存储池会立即从集群里删除，但是实际的数据将在后台删除。

Parameters:

- cluster – the cluster the pool is in
- pool_name – which pool to delete

Returns:

- 0 on success, negative error code on failure

int **rados_ioctx_pool_set_auid**(rados_ioctx_t io, uint64_t auid)

Attempt to change an io context's associated auid "owner.".

Requires that you have write permission on both the current and new auid.

尝试更改一个和 io 上下文相关的所有者 auid。

要求你同时具有当前和新 auid 的写权限。

Parameters:

- io – reference to the pool to change.
- auid – the auid you wish the io to have.

Returns:

- 0 on success, negative error code on failure

int **rados_ioctx_pool_get_auid**(rados_ioctx_t io, uint64_t *auid)

Get the auid of a pool.

获取一个存储池的 auid。

Parameters:

- io – pool to query
- auid – where to store the auid

Returns:

- 0 on success, negative error code on failure

int64_t **rados_ioctx_get_id**(rados_ioctx_t io)

Get the pool id of the io context.

获取 io 上下文的存储池 ID。

Parameters:

- io – the io context to query

Returns:

- the id of the pool the io context uses

int **rados_ioctx_get_pool_name**(rados_ioctx_t io, char *buf, unsigned maxlen)

Get the pool name of the io context.

获取 io 上下文的存储池名字。

Parameters:

- io – the io context to query
- buf – pointer to buffer where name will be stored
- maxlen – size of buffer where name will be stored

Returns:

- length of string stored, or -ERANGE if buffer too small

void **rados_ioctx_locator_set_key**(rados_ioctx_t io, const char *key)

Set the key for mapping objects to pgs within an io context.

The key is used instead of the object name to determine which placement groups an object is put in. This affects all subsequent operations of the io context - until a different locator key is set, all objects in this io

context will be placed in the same pg.
设置键名，用于把 io 上下文内的对象映射到归置组。
ceph 用键名而非对象名来确定对象放到了哪个归置组，这影响 io 上下文的所有后续操作，除非设置了另一个定位键名，此 io 上下文里的所有对象都将放到相同归置组。

This is useful if you need to do clone_range operations, which must be done with the source and destination objects in the same pg.
这在做 clone_range 操作的时候有用，因为它只能在源、目的对象都在相同归置组时完成。

Parameters:
- io – the io context to change
- key – the key to use as the object locator, or NULL to discard any previously set key

int **rados_objects_list_open**(rados_ioctx_t io, rados_list_ctx_t *ctx)
Start listing objects in a pool.
开列存储池里的对象。

Parameters:
- io – the pool to list from
- ctx – the handle to store list context in

Returns:
- 0 on success, negative error code on failure

int **rados_objects_list_next**(rados_list_ctx_t ctx, const char **entry, const char **key)
Get the next object name and locator in the pool.
*entry and *key are valid until next call to rados_objects_list_*
获取存储池里下个对象的名字和定位符。
*entry 和*key 在再次调用 rados_objects_list_*前都是有效的。

Parameters:
- ctx – iterator marking where you are in the listing
- entry – where to store the name of the entry
- key – where to store the object locator (set to NULL to ignore)

Returns:
- 0 on success, negative error code on failure
- -ENOENT when there are no more objects to list

void **rados_objects_list_close**(rados_list_ctx_t ctx)
Close the object listing handle.
This should be called when the handle is no longer needed. The handle should not be used after it has been closed.
关闭列出对象句柄。
句柄不再需要时应该调用此函数，且调用后不应该再使用此句柄。

Parameters:
- ctx – the handle to close

int **rados_ioctx_snap_create**(rados_ioctx_t io, const char *snapname)
Create a pool-wide snapshot.
创建存储池快照。

Parameters:
- io – the pool to snapshot
- snapname – the name of the snapshot

Returns:
- 0 on success, negative error code on failure

int **rados_ioctx_snap_remove**(rados_ioctx_t io, const char *snapname)

Delete a pool snapshot.
删除存储池快照。

Parameters:
- io – the pool to delete the snapshot from
- snapname – which snapshot to delete

Returns:
- 0 on success, negative error code on failure

int **rados_rollback**(rados_ioctx_t io, const char *oid, const char *snapname)
Rollback an object to a pool snapshot.
The contents of the object will be the same as when the snapshot was taken.
根据存储池快照回滚一个对象。
对象内容应该和拍快照时的内容相同。

Parameters:
- io – the pool in which the object is stored
- oid – the name of the object to rollback
- snapname – which snapshot to rollback to

Returns:
- 0 on success, negative error code on failure

void **rados_ioctx_snap_set_read**(rados_ioctx_t io, rados_snap_t snap)
Set the snapshot from which reads are performed.
Subsequent reads will return data as it was at the time of that snapshot.
指定从哪个快照执行读操作。
后续读操作将返回拍下快照时的数据。

Parameters:
- io – the io context to change
- snap – the id of the snapshot to set, or CEPH_NOSNAP for no snapshot (i.e. normal operation)

int **rados_ioctx_selfmanaged_snap_create**(rados_ioctx_t io, rados_snap_t *snapid)
Allocate an ID for a self-managed snapshot.
Get a unique ID to put in the snaphot context to create a snapshot. A clone of an object is not created until a write with the new snapshot context is completed.
给自管理的快照分配一个 ID。
获取一个唯一 ID，用于创建快照时放入上下文，在完成快照上下文的写操作前不会克隆对象。

Parameters:
- io – the pool in which the snapshot will exist
- snapid – where to store the newly allocated snapshot ID

Returns:
- 0 on success, negative error code on failure

int **rados_ioctx_selfmanaged_snap_remove**(rados_ioctx_t io, rados_snap_t snapid)
Remove a self-managed snapshot.
This increases the snapshot sequence number, which will cause snapshots to be removed lazily.
删除自管理的快照。
这会增加快照序列号，将导致懒散地删除快照。

Parameters:
- io – the pool in which the snapshot will exist
- snapid – where to store the newly allocated snapshot ID

Returns:
- 0 on success, negative error code on failure

int **rados_ioctx_selfmanaged_snap_rollback**(rados_ioctx_t io, const char *oid, rados_snap_t snapid)
Rollback an object to a self-managed snapshot.

The contents of the object will be the same as when the snapshot was taken.
把对象回滚到一个自管理快照。
对象内容将和拍下快照时一样。

Parameters:
- io – the pool in which the object is stored
- oid – the name of the object to rollback
- snapid – which snapshot to rollback to

Returns:
- 0 on success, negative error code on failure

int **rados_ioctx_selfmanaged_snap_set_write_ctx**(rados_ioctx_t io, rados_snap_t seq, rados_snap_t *snaps, int num_snaps)

Set the snapshot context for use when writing to objects.
This is stored in the io context, and applies to all future writes.
设置写入对象时的快照上下文。
这会存储在 io 上下文内，并且应用到未来所有写操作里。

Parameters:
- io – the io context to change
- seq – the newest snapshot sequence number for the pool
- snaps – array of snapshots in sorted by descending id
- num_snaps – how many snaphosts are in the snaps array

Returns:
- 0 on success, negative error code on failure
- -EINVAL if snaps are not in descending order

int **rados_ioctx_snap_list**(rados_ioctx_t io, rados_snap_t *snaps, int maxlen)

List all the ids of pool snapshots.
If the output array does not have enough space to fit all the snapshots, -ERANGE is returned and the caller should retry with a larger array.
列出存储池快照的所有 ID。
如果输出阵列没有足够空间容纳所有快照，就会返回-ERANGE，调用者应该尝试更大的阵列。

Parameters:
- io – the pool to read from
- snaps – where to store the results
- maxlen – the number of rados_snap_t that fit in the snaps array

Returns:
- number of snapshots on success, negative error code on failure
- -ERANGE is returned if the snaps array is too short

int **rados_ioctx_snap_lookup**(rados_ioctx_t io, const char *name, rados_snap_t *id)

Get the id of a pool snapshot.
获取存储池快照的 ID。

Parameters:
- io – the pool to read from
- name – the snapshot to find
- id – where to store the result

Returns:
- 0 on success, negative error code on failure

int **rados_ioctx_snap_get_name**(rados_ioctx_t io, rados_snap_t id, char *name, int maxlen)

Get the name of a pool snapshot.
获取存储池快照名字。

Parameters:
- io – the pool to read from

- id – the snapshot to find
- name – where to store the result
- maxlen – the size of the name array

Returns:

- 0 on success, negative error code on failure
- -ERANGE if the name array is too small

int **rados_ioctx_snap_get_stamp**(rados_ioctx_t io, rados_snap_t id, time_t *t)

Find when a pool snapshot occurred.

查出存储池快照时间戳。

Parameters:

- io – the pool the snapshot was taken in
- id – the snapshot to lookup
- t – where to store the result

Returns:

- 0 on success, negative error code on failure

uint64_t **rados_get_last_version**(rados_ioctx_t io)

Return the version of the last object read or written to.

This exposes the internal version number of the last object read or written via this io context

返回最后读取或写入对象的版本。

揭露通过此 io 上下文最后读取或写入对象的内部版本号。

Parameters:

- io – the io context to check

Returns:

- last read or written object version

int **rados_write**(rados_ioctx_t io, const char *oid, const char *buf, size_t len, uint64_t off)

Write data to an object.

把数据写入对象。

Parameters:

- io – the io context in which the write will occur
- oid – name of the object
- buf – data to write
- len – length of the data, in bytes
- off – byte offset in the object to begin writing at

Returns:

- number of bytes written on success, negative error code on failure

int **rados_write_full**(rados_ioctx_t io, const char *oid, const char *buf, size_t len)

Write an entire object.

The object is filled with the provided data. If the object exists, it is atomically truncated and then written.

写入一个完整对象。

用准备好的数据填充对象。如果对象已存在，它会自动删节、然后写入。

Parameters:

- io – the io context in which the write will occur
- oid – name of the object
- buf – data to write
- len – length of the data, in bytes

Returns:

- 0 on success, negative error code on failure

int **rados_clone_range**(rados_ioctx_t io, const char *dst, uint64_t dst_off, const char *src, uint64_t src_off, size_t len)

Efficiently copy a portion of one object to another.

If the underlying filesystem on the OSD supports it, this will be a copy-on-write clone.
高效地把一对象的一部分拷贝到别处。
如果 OSD 的底层文件系统支持，这会是写时复制克隆。

The src and dest objects must be in the same pg. To ensure this, the io context should have a locator key set (see rados_ioctx_locator_set_key()).
源和目的对象必须在同一归置组内，要保证这点，io 上下文应该设置一个键定位器（见 rados_ioctx_locator_set_key()）。

Parameters:
- io – the context in which the data is cloned
- dst – the name of the destination object
- dst_off – the offset within the destination object (in bytes)
- src – the name of the source object
- src_off – the offset within the source object (in bytes)
- len – how much data to copy

Returns:
- 0 on success, negative error code on failure

int **rados_append**(rados_ioctx_t io, const char *oid, const char *buf, size_t len)
Append data to an object.
把数据追加到一对象。

Parameters:
- io – the context to operate in
- oid – the name of the object
- buf – the data to append
- len – length of buf (in bytes)

Returns:
- number of bytes written on success, negative error code on failure

int **rados_read**(rados_ioctx_t io, const char *oid, char *buf, size_t len, uint64_t off)
Read data from an object.
The io context determines the snapshot to read from, if any was set by rados_ioctx_snap_set_read().
从一对象读数据。
如果 rados_ioctx_snap_set_read()设置过了，那就由 io 上下文决定从哪个快照读。

Parameters:
- io – the context in which to perform the read
- oid – the name of the object to read from
- buf – where to store the results
- len – the number of bytes to read
- off – the offset to start reading from in the object

Returns:
- number of bytes read on success, negative error code on failure

int **rados_remove**(rados_ioctx_t io, const char *oid)
Delete an object.
删除一对象。

Note This does not delete any snapshots of the object.
注意：这不会删除任何快照里的对应对象。

Parameters:
- io – the pool to delete the object from
- oid – the name of the object to delete

Returns:
- 0 on success, negative error code on failure

int **rados_trunc**(rados_ioctx_t io, const char *oid, uint64_t size)

Resize an object.

If this enlarges the object, the new area is logically filled with zeroes. If this shrinks the object, the excess data is removed.

调整对象大小。

若是扩大对象，新区域是逻辑上用 0 填充的；若是缩小对象，多余数据将被删除。

Parameters:

- io – the context in which to truncate
- oid – the name of the object
- size – the new size of the object in bytes

Returns:

- 0 on success, negative error code on failure

int **rados_getxattr**(rados_ioctx_t io, const char *o, const char *name, char *buf, size_t len)

Get the value of an extended attribute on an object.

获取一对象的扩展属性值。

Parameters:

- io – the context in which the attribute is read
- o – name of the object
- name – which extended attribute to read
- buf – where to store the result
- len – size of buf in bytes

Returns:

- length of xattr value on success, negative error code on failure

int **rados_setxattr**(rados_ioctx_t io, const char *o, const char *name, const char *buf, size_t len)

Set an extended attribute on an object.

设置一对象的扩展属性。

Parameters:

- io – the context in which xattr is set
- o – name of the object
- name – which extended attribute to set
- buf – what to store in the xattr
- len – the number of bytes in buf

Returns:

- 0 on success, negative error code on failure

int **rados_rmxattr**(rados_ioctx_t io, const char *o, const char *name)

Delete an extended attribute from an object.

删除一对象的扩展属性。

Parameters:

- io – the context in which to delete the xattr
- o – the name of the object
- name – which xattr to delete

Returns:

- 0 on success, negative error code on failure

int **rados_getxattrs**(rados_ioctx_t io, const char *oid, rados_xattrs_iter_t *iter)

Start iterating over xattrs on an object.

Postcondition iter is a valid iterator

开始递归一对象的 xattr。

后置条件：iter 是合法的递归器。

Parameters:

- io – the context in which to list xattrs

- oid – name of the object
- iter – where to store the iterator

Returns:

- 0 on success, negative error code on failure

int **rados_getxattrs_next**(rados_xattrs_iter_t iter, const char **name, const char **val, size_t *len)

Get the next xattr on the object.

获取对象的下个 xattr。

Precondition iter is a valid iterator

Postcondition name is the NULL-terminated name of the next xattr, and val contains the value of the xattr, which is of length len. If the end of the list has been reached, name and val are NULL, and len is 0.

前提条件：iter 是合法递归器。

后置条件：name 是 NULL 结尾的下一个 xattr 名字；val 包含 xattr 的值，它是长度 len。到达列表末尾后，name 和 val 为 NULL、len 为 0。

Parameters:

- iter – iterator to advance
- name – where to store the name of the next xattr
- val – where to store the value of the next xattr
- len – the number of bytes in val

Returns:

- 0 on success, negative error code on failure

void **rados_getxattrs_end**(rados_xattrs_iter_t iter)

Close the xattr iterator.

iter should not be used after this is called.

关闭 xattr 递归器。

调用此函数后 iter 寿终。

Parameters:

- iter – the iterator to close

int **rados_stat**(rados_ioctx_t io, const char *o, uint64_t *psize, time_t *pmtime)

Get object stats (size/mtime)

TODO: when are these set, and by whom? can they be out of date?

获取对象状态：尺寸、修改时间。

待完成：何时设置、谁设置的、它们可以失真么？

Parameters:

- io – ioctx
- o – object name
- psize – where to store object size
- pmtime – where to store modification time

Returns:

- 0 on success, negative error code on failure

int **rados_tmap_update**(rados_ioctx_t io, const char *o, const char *cmdbuf, size_t cmdbuflen)

Update tmap (trivial map)

Do compound update to a tmap object, inserting or deleting some number of records. cmdbuf is a series of operation byte codes, following by command payload. Each command is a single-byte command code, whose value is one of CEPH_OSD_TMAP_*.

更新 tmap（普通 map）。

混合更新一个 tmap 对象，插入或删除一些记录。cmdbuf 是一系列操作字节码，其后是命令载荷。每个命令都是一个单字节命令代码，其值是 CEPH_OSD_TMAP_*之一。

- update tmap ‘header’
  - 1 byte = CEPH_OSD_TMAP_HDR

        - 4 bytes = data length (little endian)
        - N bytes = data
    - insert/update one key/value pair
        - 1 byte = CEPH_OSD_TMAP_SET
        - 4 bytes = key name length (little endian)
        - N bytes = key name
        - 4 bytes = data length (little endian)
        - M bytes = data
    - insert one key/value pair; return -EEXIST if it already exists.
        - 1 byte = CEPH_OSD_TMAP_CREATE
        - 4 bytes = key name length (little endian)
        - N bytes = key name
        - 4 bytes = data length (little endian)
        - M bytes = data
    - remove one key/value pair
        - 1 byte = CEPH_OSD_TMAP_RM
        - 4 bytes = key name length (little endian)
        - N bytes = key name

    Restrictions:
    - The HDR update must preceed any key/value updates.
    - All key/value updates must be in lexicographically sorted order in cmdbuf.
    - You can read/write to a tmap object via the regular APIs, but you should be careful not to corrupt it. Also be aware that the object format may change without notice.

    限制条件：
    - HDR 更新必须先于 key/value 更新。
    - 所有 key/value 更新必须按 cmdbuf 里的字典顺序进行。
    - 你可以通过正常的 API 读写 tmap 对象，但要注意不要伤害它。注意，对象格式可能在未通知的情况下更改。

    Parameters:
    - io – ioctx
    - o – object name
    - cmdbuf – command buffer
    - cmdbuflen – command buffer length in bytes

    Returns:
    - 0 on success, negative error code on failure

int **rados_tmap_put**(rados_ioctx_t io, const char *o, const char *buf, size_t buflen)

    Store complete tmap (trivial map) object.

    Put a full tmap object into the store, replacing what was there.

    存储完整的 tmap（寻常 map）对象。

    放入一个完整的 tmap 对象，取代先前的。

    The format of buf is:

    buf 格式为：
    - 4 bytes - length of header (little endian)
    - N bytes - header data
    - 4 bytes - number of keys (little endian)

    and for each key,

    以及每个键：
    - 4 bytes - key name length (little endian)
    - N bytes - key name
    - 4 bytes - value length (little endian)
    - M bytes - value data

    Parameters:
    - io – ioctx

- o – object name
- buf – buffer
- buflen – buffer length in bytes

Returns:

- 0 on success, negative error code on failure

int **rados_tmap_get**(rados_ioctx_t io, const char *o, char *buf, size_t buflen)

Fetch complete tmap (trivial map) object.

Read a full tmap object. See rados_tmap_put() for the format the data is returned in.

取来完整的 tmap（寻常 map）对象。

读取一个完整 tmap 对象，返回的数据格式见 rados_tmap_put()。

Parameters:

- io – ioctx
- o – object name
- buf – buffer
- buflen – buffer length in bytes

Returns:

- 0 on success, negative error code on failure
- -ERANGE if buf isn't big enough

int **rados_exec**(rados_ioctx_t io, const char *oid, const char *cls, const char *method, const char *in_buf, size_t in_len, char *buf, size_t out_len)

Execute an OSD class method on an object.

The OSD has a plugin mechanism for performing complicated operations on an object atomically. These plugins are called classes. This function allows librados users to call the custom methods. The input and output formats are defined by the class. Classes in ceph.git can be found in src/cls_*.cc

对一对象执行 OSD 类方法。

OSD 有个插件机制，用于在一个对象上原子地执行复杂操作，这些插件叫类。此函数允许 librados 用户调用定制的方法。输入和输出格式由类定义，类位于 ceph.git 的 src/cls_*.cc 下。

Parameters:

- io – the context in which to call the method
- oid – the object to call the method on
- cls – the name of the class
- method – the name of the method
- in_buf – where to find input
- in_len – length of in_buf in bytes
- buf – where to store output
- out_len – length of buf in bytes

Returns:

- the length of the output, or -ERANGE if out_buf does not have enough space to store it (For methods that return data). For methods that don't return data, the return value is method-specific.

int **rados_aio_create_completion**(void *cb_arg, rados_callback_t cb_complete, rados_callback_t cb_safe, rados_completion_t *pc)

Constructs a completion to use with asynchronous operations.

The complete and safe callbacks correspond to operations being acked and committed, respectively. The callbacks are called in order of receipt, so the safe callback may be triggered before the complete callback, and vice versa. This is affected by journalling on the OSDs.

构造异步操作需要的完成。

操作对应的 complete 和 safe 回调会被分别确认、提交，回调按照其收到顺序调用，所以 safe 回调可能早于 complete 而触发，反之亦然。它会被 OSD 上的日志影响。

TODO: more complete documentation of this elsewhere (in the RADOS docs?)

待完成：关于 complete 的更详细文档（在 RADOS 文档里？）

Note: Read operations only get a complete callback.
BUG: this should check for ENOMEM instead of throwing an exception
注意：读操作只能得到一个 complete 回调。
缺陷：此函数应该检查 ENOMEM 而不是抛出异常。

Parameters:
- cb_arg – application-defined data passed to the callback functions
- cb_complete – the function to be called when the operation is in memory on all relpicas
- cb_safe – the function to be called when the operation is on stable storage on all replicas
- pc – where to store the completion

Returns:
- 0

int **rados_aio_wait_for_complete**(rados_completion_t c)
Block until an operation completes.
This means it is in memory on all replicas.
操作完成前一直阻塞。
这意味着有关它的所有复制都在内存里。

Note BUG: this should be void
注意：缺陷：这应该是空的。

Parameters:
- c – operation to wait for

Returns:
- 0

int **rados_aio_wait_for_safe**(rados_completion_t c)
Block until an operation is safe.
This means it is on stable storage on all replicas.
某操作安全前一直阻塞。
这意味着有关它的所有复制都在稳定的存储器上。

Note BUG: this should be void
注意：缺陷：这应该为空。

Parameters:
- c – operation to wait for

Returns:
- 0

int **rados_aio_is_complete**(rados_completion_t c)
Has an asynchronous operation completed?
异步操作是否完成?

Warning This does not imply that the complete callback has finished
警告：这并不意味着完成回调已结束。

Parameters:
- c – async operation to inspect

Returns:
- whether c is complete

int rados_aio_is_safe(rados_completion_t c)
Is an asynchronous operation safe?
某异步操作是否安全?

Warning This does not imply that the safe callback has finished
警告：这并不意味着安全回调已完成。

Parameters:
- c – async operation to inspect

Returns:
- whether c is safe

int **rados_aio_wait_for_complete_and_cb**(rados_completion_t c)
Block until an operation completes and callback completes.
This means it is in memory on all replicas and can be read.
在操作完成、其回调完成前一直阻塞。
这意思是关于它的所有复制都在内存里，且可读。

Note BUG: this should be void
注意：缺陷：这应该为空。

Parameters:
- c – operation to wait for

Returns:
- 0

int **rados_aio_wait_for_safe_and_cb**(rados_completion_t c)
Block until an operation is safe and callback has completed.
在操作安全且其回调完成前一直阻塞。

This means it is on stable storage on all replicas.
这意思是关于它的所有复制都在稳定存储器上。

Note BUG: this should be void
注意：缺陷：这应该为空。

Parameters:
- c – operation to wait for

Returns:
- 0

int **rados_aio_is_complete_and_cb**(rados_completion_t c)
Has an asynchronous operation and callback completed.
异步操作及其回调是否完成。

Parameters:
- c – async operation to inspect

Returns:
- whether c is complete

int **rados_aio_is_safe_and_cb**(rados_completion_t c)
Is an asynchronous operation safe and has the callback completed.
异步操作是否安全、其回调是否完成。

Parameters:
- c – async operation to inspect

Returns:
- whether c is safe

int **rados_aio_get_return_value**(rados_completion_t c)
Get the return value of an asychronous operation.

The return value is set when the operation is complete or safe, whichever comes first.
获取一异步操作的返回值。
操作完成或安全时就设置返回值，先到为准。

Precondition The operation is safe or complete
前提条件：操作安全或完成。

Note BUG: complete callback may never be called when the safe message is received before the complete message
注意：缺陷：在安全消息先于完成消息收到前，不该再调用完成回调。

Parameters:
- c – async operation to inspect

Returns:
- return value of the operation

void **rados_aio_release**(rados_completion_t c)
Release a completion.
Call this when you no longer need the completion. It may not be freed immediately if the operation is not acked and committed.
释放一个完成。
你不再需要完成回馈时可调用这个。如果操作未被确认和提交，它就不会立即释放。

Parameters:
- c – completion to release

int **rados_aio_write**(rados_ioctx_t io, const char *oid, rados_completion_t completion, const char *buf, size_t len, uint64_t off)
Write data to an object asynchronously.
Queues the write and returns. The return value of the completion will be 0 on success, negative error code on failure.
异步地把数据写入对象。
把写加入队列然后返回，完成时若返回 0，则成功；负数错误代码则失败。

Parameters:
- io – the context in which the write will occur
- oid – name of the object
- completion – what to do when the write is safe and complete
- buf – data to write
- len – length of the data, in bytes
- off – byte offset in the object to begin writing at

Returns:
- 0 on success, -EROFS if the io context specifies a snap_seq other than CEPH_NOSNAP

int **rados_aio_append**(rados_ioctx_t io, const char *oid, rados_completion_t completion, const char *buf, size_t len)
Asychronously append data to an object.
Queues the append and returns.
The return value of the completion will be 0 on success, negative error code on failure.
异步地追加数据到对象。
把追加放入队列然后返回。
成功时返回值为 0，失败时为负错误代码。

Parameters:
- io – the context to operate in
- oid – the name of the object
- completion – what to do when the append is safe and complete
- buf – the data to append

- len – length of buf (in bytes)

Returns:

- 0 on success, -EROFS if the io context specifies a snap_seq other than CEPH_NOSNAP

int **rados_aio_write_full**(rados_ioctx_t io, const char *oid, rados_completion_t completion, const char *buf, size_t len)

Asychronously write an entire object.

The object is filled with the provided data. If the object exists, it is atomically truncated and then written.

Queues the write_full and returns.

The return value of the completion will be 0 on success, negative error code on failure.

异步写整个对象。

用提供的数据填充对象，若对象存在，它会被自动删节、然后写入。把 write_full 排队并返回。

成功时完成返回值为 0，失败时为负数错误代码。

Parameters:

- io – the io context in which the write will occur
- oid – name of the object
- completion – what to do when the write_full is safe and complete
- buf – data to write
- len – length of the data, in bytes

Returns:

- 0 on success, -EROFS if the io context specifies a snap_seq other than CEPH_NOSNAP

int **rados_aio_remove**(rados_ioctx_t io, const char *oid, rados_completion_t completion)

Asychronously remove an object.

Queues the remove and returns.

The return value of the completion will be 0 on success, negative error code on failure.

异步删除对象。

排列删除并返回。

成功时完成返回值为 0，失败时为负数错误代码。

Parameters:

- io – the context to operate in
- oid – the name of the object
- completion – what to do when the remove is safe and complete

Returns:

- 0 on success, -EROFS if the io context specifies a snap_seq other than CEPH_NOSNAP

int **rados_aio_read**(rados_ioctx_t io, const char *oid, rados_completion_t completion, char *buf, size_t len, uint64_t off)

Asychronously read data from an object.

The io context determines the snapshot to read from, if any was set by rados_ioctx_snap_set_read().

The return value of the completion will be number of bytes read on success, negative error code on failure.

从一对象异步读数据。

如果 rados_ioctx_snap_set_read()设置过，io 上下文可决定从哪个快照读取。

完成返回值是成功读取的字节数，失败时为负数错误代码。

Note only the 'complete' callback of the completion will be called.

注意：只有完成的'complete'回调会被调用。

Parameters:

- io – the context in which to perform the read
- oid – the name of the object to read from
- completion – what to do when the read is complete
- buf – where to store the results
- len – the number of bytes to read
- off – the offset to start reading from in the object

Returns:

- 0 on success, negative error code on failure

int **rados_aio_flush**(rados_ioctx_t io)

Block until all pending writes in an io context are safe.
This is not equivalent to calling rados_aio_wait_for_safe()on all write completions, since this waits for the associated callbacks to complete as well.
确定 io 上下文里的未决写入安全前一直阻塞着。
此函数和在所有完成上调用 rados_aio_wait_for_safe()不等价，因为它也等着相关回调完成。

Note BUG: always returns 0, should be void or accept a timeout
注意：缺陷：总是返回 0，应该为空或接受超时。

Parameters:

- io – the context to flush

Returns:

- 0 on success, negative error code on failure

int **rados_watch**(rados_ioctx_t io, const char *o, uint64_t ver, uint64_t *handle, rados_watchcb_t watchcb, void *arg)

Register an interest in an object.
A watch operation registers the client as being interested in notifications on an object. OSDs keep track of watches on persistent storage, so they are preserved across cluster changes by the normal recovery process. If the client loses its connection to the primary OSD for a watched object, the watch will be removed after 30 seconds. Watches are automatically reestablished when a new connection is made, or a placement group switches OSDs.
关注对象。
关注操作注册了客户端对关于某对象的通知感兴趣。OSD 把关注记录在永久存储器上，所以在常规的集群恢复后仍会保留。如果客户端到关注对象所在 OSD 的连接断开，则相应关注会在 30 秒后删除。新连接建立时、或归置组跑到另外 OSD 时关注会自动重建。

Note BUG: watch timeout should be configurable
BUG: librados should provide a way for watchers to notice connection resets
注意：缺陷：关注超时值应该可配置。
缺陷：librados 库应该提供一种方法让关注者通知连接重置。

BUG: the ver parameter does not work, and -ERANGE will never be returned (
缺陷：ver 参数无效，且-ERANGE 永不返回。

Warning asphyxiate: No renderer found for doxygen tag ‘ulink’
<ulink xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" url="http://www.tracker.newdream.net/issues/2592">http://www.tracker.newdream.net/issues/2592</ulink>)
)

Parameters:

- io – the pool the object is in
- o – the object to watch
- ver – expected version of the object
- handle – where to store the internal id assigned to this watch
- watchcb – what to do when a notify is received on this object
- arg – application defined data to pass when watchcb is called

Returns:

- 0 on success, negative error code on failure
- -ERANGE if the version of the object is greater than ver

int **rados_unwatch**(rados_ioctx_t io, const char *o, uint64_t handle)

Unregister an interest in an object.

Once this completes, no more notifies will be sent to us for this watch. This should be called to clean up unneeded watchers.
注销到对象的关注。
它完成后，不会再有相关通知发来。应该用于清理不必要的关注。

Parameters:
- io – the pool the object is in
- o – the name of the watched object
- handle – which watch to unregister

Returns:
- 0 on success, negative error code on failure

int **rados_notify**(rados_ioctx_t io, const char *o, uint64_t ver, const char *buf, int buf_len)
Sychronously notify watchers of an object.
This blocks until all watchers of the object have received and reacted to the notify, or a timeout is reached.
同步通知某对象的关注者。
它会一直阻塞着，直到对象的所有关注者收到并回馈了通知、或超时。

Note BUG: the timeout is not changeable via the C API
BUG: the bufferlist is inaccessible in a rados_watchcb_t
注意：缺陷：通过 C API 不能更改超时值。
缺陷：rados_watchcb_t 内的缓冲区列表不可访问。

Parameters:
- io – the pool the object is in
- o – the name of the object
- ver – obsolete - just pass zero
- buf – data to send to watchers
- buf_len – length of buf in bytes

Returns:
- 0 on success, negative error code on failure

3.5.2 libradospp (C++)

Todo write me!
待完成。

4 CEPH FS

The Ceph FS file system is a POSIX-compliant file system that uses a RADOS cluster to store its data. Ceph FS uses the same RADOS object storage device system as RADOS block devices and RADOS object stores such as the RADOS gateway with its S3 and Swift APIs, or native bindings. Using Ceph FS requires at least one metadata server in your ceph.conf configuration file.
Ceph FS 文件系统是个 POSIX 兼容的文件系统，它使用 RADOS 集群存储其数据。Ceph FS 使用的对象存储设备和 RADOS 块设备及 RADOS 对象存储相同，就像 RADOS 网关也有 S3 和 Swift API、或本地 API 一样。使用 Ceph FS 要求 ceph.conf 配置文件里至少有一个元数据服务器。

4.1 用内核驱动挂载 ceph 文件系统

Mount ceph fs with the kernel driver
To mount the Ceph file system you may use the mount command if you know the monitor host IP address(es), or use the mount.ceph utility to resolve the monitor host name(s) into IP address(es) for you. For example:
要挂载 ceph 文件系统，如果你知道监视器 IP 地址可以用 mount 命令、或者用 mount.ceph 工具来自动解析监视器 IP 地址。例如：

```
sudo mkdir /mnt/mycephfs
```

```
sudo mount -t ceph 192.168.0.1:6789:/ /mnt/mycephfs
```

To mount the Ceph file system with cephx authentication enabled, you must specify a user name and a secret.
要挂载启用了 cephx 认证的 ceph 文件系统，你必须指定用户名、密码。

```
sudo mount -t ceph 192.168.0.1:6789:/ /mnt/mycephfs -o
name=admin,secret=AQATSKdNGBnwLhAAnNDKnH65FmVKpXZJVasUeQ==
```

The foregoing usage leaves the secret in the Bash history. A more secure approach reads the secret from a file. For example:
前述用法会把密码遗留在 bash 历史里，更安全的方法是从文件读密码。例如：

```
sudo mount -t ceph 192.168.0.1:6789:/ /mnt/mycephfs -o name=admin,secretfile=/etc/ceph/admin.secret
```

See Authentication for details on cephx.
关于 cephx 参见 ceph 认证及授权。

To unmount the Ceph file system, you may use the umount command. For example:
要卸载 ceph 文件系统，可以用 unmount 命令，例如：

```
sudo umount /mnt/mycephfs
```

Tip Ensure that you are not within the file system directories before executing this command.
提示：执行此命令前确保你不在此文件系统目录下。

See mount.ceph for details.
详情参见 mount.ceph。

4.2 用户空间挂载 ceph 文件系统

Mount ceph FS as a fuse

To mount the Ceph file system as a File System in User Space (FUSE), you may use the ceph-fuse command. For example:
要把 ceph 文件系统挂载为用户空间文件系统（File System in User Space, FUSE），可以用 ceph-fuse 命令，例如：

```
sudo mkdir /home/usernname/cephfs
sudo ceph-fuse -m 192.168.0.1:6789 /home/username/cephfs
```

If cephx authentication is on, ceph-fuse will retrieve the name and secret from the key ring automatically.
如果启用了 cephx 认证，ceph-fuse 会从密钥环自动检索用户名、密码。

See ceph-fuse for details.
详情参见 ceph-fuse——ceph 的用户空间客户端。

4.3 从 fstab 挂载

Mount ceph fs in your file systems table

If you mount Ceph FS in your file systems table, the Ceph file system will mount automatically on startup. To mount Ceph FS in your file systems table, add the following to /etc/fstab:
如果你从文件系统表挂载，ceph 文件系统将在启动时自动挂载。要从文件系统表挂载 Ceph FS，按下列格式添加到/etc/fstab：

```
{ipaddress}:{port}:/ {mount}/{mountpoint} {filesystem-name}     [name=username,secret=secretkey|
secretfile=/path/to/secretfile],[{mount.options}]
```

For example:
例如：

```
10.10.10.10:6789:/     /mnt/ceph     ceph
name=admin,secretfile=/etc/ceph/secret.key,noauto,rw,noexec,nodev,noatime,nodiratime    0       2
```

Important The name and secret or secretfile options are mandatory when you have Ceph authentication running. See Authentication for details.
重要：启用了认证时，name 及 secret 或 secretfile 选项是强制的。详情参见 ceph 认证及授权。

4.4 让 hadoop 使用 cephfs

Using hadoop with cephfs

4.4.1 hadoop 配置

HADOOP CONFIGURATION
This section describes the Hadoop configuration options used to control Ceph. These options are intended to be set in the Hadoop configuration file conf/core-site.xml.
本段描述了用于控制 ceph 的 hadoop 选项，这些选项应该写于 Hadoop 配置文件 conf/core-site.xml。

Property	Value	Notes
fs.default.name	Ceph URI	ceph:///
ceph.conf.file	Local path to ceph.conf	/etc/ceph/ceph.conf
ceph.conf.options	Comma separated list of Ceph configuration key/value pairs	opt1=val1,opt2=val2
ceph.root.dir	Mount root directory	Default value: /
ceph.object.size	Default file object size in bytes	Default value (64MB): 67108864
ceph.data.pools	List of Ceph data pools for storing file.	Default value: default Ceph pool.
ceph.localize.reads	Allow reading from file replica objects	Default value: true

4.4.2 对每文件定制复制的支持

Support For Per-file Custom Replication
Hadoop users may specify a custom replication factor (e.g. 3 copies of each block) when creating a file. However, object replication factors are controlled on a per-pool basis in Ceph, and by default a Ceph file system will contain a pre-configured pool. In order to support per-file replication Hadoop can be configured to select from alternative pools when creating new files.
Hadoop 用户可在创建文件时指定一个定制的复制因子（例如每块 3 个副本）。然而对象复制因子在 ceph 里是以每存储池为基数进行控制的，并且 ceph 文件系统默认会包含一个预配置的存储池。为支持每文件复制策略，Hadoop 可配置为创建新文件时选择另一个存储池。

Additional data pools can be specified using the ceph.data.pools configuration option. The value of the option is a comma separated list of pool names. The default Ceph pool will be used automatically if this configuration option is omitted or the value is empty. For example, the following configuration setting will consider the three pools listed.
额外的数据存储池可用 ceph.data.pools 指定，此选项的值是逗号分隔的一溜存储池名字。此选项被忽略或为空时将使用默认 ceph 存储池，例如，下面配置了 3 个存储池：

```
<property>
  <name>ceph.data.pools</name>
  <value>pool1,pool2,pool5</value>
</property>
```

Hadoop will not create pools automatically. In order to create a new pool with a specific replication factor use the ceph osd pool create command, and then set the size property on the pool using the ceph osd pool set command. For more information on creating and configuring pools see the RADOS Pool documentation.
Hadoop 不会自动创建存储池，要创建有指定复制因子的存储池，可用 ceph osd pool create 命令、然后用 ceph osd pool set 命令设置存储池的 size 属性。更多的创建、配置手册见存储池。

Once a pool has been created and configured the metadata service must be told that the new pool may be used to store file data. A pool can be made available for storing file system data using the ceph mds add_data_pool command.
存储池创建、配置完毕后，新存储池可用于存储文件数据的消息必须告知元数据服务，可用 ceph mds add_data_pool 命令告知，这样存储池就可存储文件系统数据了。

First, create the pool. In this example we create the hadoop1 pool with replication factor 1.
首先，创建存储池。本例中，我们创建 hadoop1 存储池，其复制因子为 1。
ceph osd pool create hadoop1 100
ceph osd pool set hadoop1 size 1

Next, determine the pool id. This can be done using the ceph osd dump command. For example, we can look for the newly created hadoop1 pool.
下一步，找出存储池 ID，命令为 ceph osd dump。例如，找出刚创建的 hadoop1 存储池：
ceph osd dump | grep hadoop1

The output should resemble:
输出应该类似：
pool 3 'hadoop1' rep size 1 min_size 1 crush_ruleset 0...

where 3 is the pool id. Next we will use the pool id reference to register the pool as a data pool for storing file system data.
其中，3 是存储池 id。下面我们用前述 ID 把存储池注册为数据存储池，用于存储文件系统数据。
ceph mds add_data_pool 3

The final step is to configure Hadoop to consider this data pool when selecting the target pool for new files.
最后配置 Hadoop，让它在为新文件选择目标存储池时考虑此存储池。
```
<property>
    <name>ceph.data.pools</name>
    <value>hadoop1</value>
</property>
```

4.4.3 存储池选择语义

Pool Selection Semantics
The following semantics describe the rules by which Hadoop will choose a pool given a desired replication factor and the set of pools specified using the ceph.data.pools configuration option.
下面的语义描述了 Hadoop 根据期望复制因子用以从 ceph.data.pools 配置中选择一个存储池的规则。

1. When no custom pools are specified the default Ceph data pool is used.
   未指定存储池时用 ceph 的默认 data 存储池。
2. A custom pool with the same replication factor as the default Ceph data pool will override the default.
   复制因子相同时，定制存储池优先于 ceph 的默认 data 存储池。
3. A pool with a replication factor that matches the desired replication will be chosen if it exists.
   复制因子和期望值相同的存储池会被选择。
4. Otherwise, a pool with at least the desired replication factor will be chosen, or the maximum possible.
   否则，选择复制因子和期望值最接近的存储池，或者复制因子最大的。

4.4.4 存储池选择调试

Debugging Pool Selection
Hadoop will produce log file entry when it cannot determine the replication factor of a pool (e.g. it is not configured as a data pool). The log message will appear as follows:
Hadoop 不确定存储池复制因子时会产生日志（如它未被配置为数据存储池），日志消息长相如下：
Error looking up replication of pool: <pool name>

Hadoop will also produce a log entry when it wasn't able to select an exact match for replication. This log

entry will appear as follows:
未能选到复制数准确匹配的存储池时 Hadoop 也会产生日志，其长相如下:
selectDataPool path=<path> pool:repl=<name>:<value> wanted=<value>

4.5 mds 配置参考

mds config reference

```
mds max file size
```

Description:
Type: 64-bit Integer Unsigned
Default: 1ULL << 40

```
mds cache size
```

Description:
Type: 32-bit Integer
Default: 100000

```
mds cache mid
```

Description:
Type: Float
Default: 0.7

```
mds mem max
```

Description: // KB
Type: 32-bit Integer
Default: 1048576

```
mds dir commit ratio
```

Description:
Type: Float
Default: 0.5

```
mds dir max commit size
```

Description: // MB
Type: 32-bit Integer
Default: 90

```
mds decay halflife
```

Description:
Type: Float
Default: 5

```
mds beacon interval
```

Description:
Type: Float
Default: 4

```
mds beacon grace
```

Description:
Type: Float
Default: 15

```
mds blacklist interval
```

Description: // how long to blacklist failed nodes
Type: Float
Default: 24.0*60.0

描述：多久把失败节点加入黑名单。

```
mds session timeout
```

Description: // cap bits and leases time out if client idle
Type: Float
Default: 60
描述：客户端空闲的能力位和租期。

```
mds session autoclose
```

Description: // autoclose idle session
Type: Float
Default: 300
描述：自动关闭空闲会话。

```
mds reconnect timeout
```

Description: // secs to wait for clients during mds restart
Type: Float
Default: 45
描述：mds 重启期间客户端等待时间，秒。

```
mds tick interval
```

Description:
Type: Float
Default: 5

```
mds dirstat min interval
```

Description: //try to avoid propagating more often than x
Type: Float
Default: 1
描述：试图使传播频率低于 x。

```
mds scatter nudge interval
```

Description: // how quickly dirstat changes propagate up
Type: Float
Default: 5
描述：dirstat 变更传播多快。

```
mds client prealloc inos
```

Description:
Type: 32-bit Integer
Default: 1000

```
mds early reply
```

Description:
Type: Boolean
Default: true

```
mds use tmap
```

Description: // use trivialmap for dir updates
Type: Boolean
Default: true

```
mds default dir hash
```

Description: CEPH STR HASH RJENKINS
Type: 32-bit Integer
Default:

```
mds log
```
Description:
Type: Boolean
Default: true

```
mds log skip corrupt events
```
Description:
Type: Boolean
Default: false

```
mds log max events
```
Description:
Type: 32-bit Integer
Default: -1

```
mds log max segments
```
Description: // segment size defined by FileLayout above
Type: 32-bit Integer
Default: 30
描述：上述 FileLayout 定义的段尺寸。

```
mds log max expiring
```
Description:
Type: 32-bit Integer
Default: 20

```
mds log eopen size
```
Description: // # open inodes per log entry
Type: 32-bit Integer
Default: 100
描述：每个日志条目打开一个 inode。

```
mds bal sample interval
```
Description: // every 5 seconds
Type: Float
Default: 3

```
mds bal replicate threshold
```
Description:
Type: Float
Default: 8000

```
mds bal unreplicate threshold
```
Description:
Type: Float
Default: 0

```
mds bal frag
```
Description:
Type: Boolean
Default: false

```
mds bal split size
```
Description:
Type: 32-bit Integer

Default: 10000

```
mds bal split rd
```

Description:
Type: Float
Default: 25000

```
mds bal split wr
```

Description:
Type: Float
Default: 10000

```
mds bal split bits
```

Description:
Type: 32-bit Integer
Default: 3

```
mds bal merge size
```

Description:
Type: 32-bit Integer
Default: 50

```
mds bal merge rd
```

Description:
Type: Float
Default: 1000

```
mds bal merge wr
```

Description:
Type: Float
Default: 1000

```
mds bal interval
```

Description: // seconds
Type: 32-bit Integer
Default: 10

```
mds bal fragment interval
```

Description: // seconds
Type: 32-bit Integer
Default: 5

```
mds bal idle threshold
```

Description:
Type: Float
Default: 0

```
mds bal max
```

Description:
Type: 32-bit Integer
Default: -1

```
mds bal max until
```

Description:
Type: 32-bit Integer
Default: -1

```
mds bal mode
```
Description:
Type: 32-bit Integer
Default: 0

```
mds bal min rebalance
```
Description: // must be x above avg before we export
Type: Float
Default: 0.1
描述：导出前必须大于 avg。

```
mds bal min start
```
Description: // if we need less x. we don't do anything
Type: Float
Default: 0.2
描述：如果我们需要小点的 x，则放任自流。

```
mds bal need min
```
Description: // take within this range of what we need
Type: Float
Default: 0.8
描述：拉回到我们需要的范围。

```
mds bal need max
```
Description:
Type: Float
Default: 1.2

```
mds bal midchunk
```
Description: // any sub bigger than this taken in full
Type: Float
Default: 0.3
描述：把小于这个的子块当作满了。

```
mds bal minchunk
```
Description: // never take anything smaller than this
Type: Float
Default: 0.001

```
mds bal target removal min
```
Description: // min bal iters before old target is removed
Type: 32-bit Integer
Default: 5

```
mds bal target removal max
```
Description: // max bal iters before old target is removed
Type: 32-bit Integer
Default: 10

```
mds replay interval
```
Description: // time to wait before starting replay again
Type: Float
Default: 1

```
mds shutdown check
```
Description:

Type: 32-bit Integer
Default: 0

`mds thrash exports`

Description:
Type: 32-bit Integer
Default: 0

`mds thrash fragments`

Description:
Type: 32-bit Integer
Default: 0

`mds dump cache on map`

Description:
Type: Boolean
Default: false

`mds dump cache after rejoin`

Description:
Type: Boolean
Default: false

`mds verify scatter`

Description:
Type: Boolean
Default: false

`mds debug scatterstat`

Description:
Type: Boolean
Default: false

`mds debug frag`

Description:
Type: Boolean
Default: false

`mds debug auth pins`

Description:
Type: Boolean
Default: false

`mds debug subtrees`

Description:
Type: Boolean
Default: false

`mds kill mdstable at`

Description:
Type: 32-bit Integer
Default: 0

`mds kill export at`

Description:
Type: 32-bit Integer
Default: 0

```
mds kill import at
```
Description:
Type: 32-bit Integer
Default: 0

```
mds kill link at
```
Description:
Type: 32-bit Integer
Default: 0

```
mds kill rename at
```
Description:
Type: 32-bit Integer
Default: 0

```
mds wipe sessions
```
Description:
Type: Boolean
Default: 0

```
mds wipe ino prealloc
```
Description:
Type: Boolean
Default: 0

```
mds skip ino
```
Description:
Type: 32-bit Integer
Default: 0

```
max mds
```
Description:
Type: 32-bit Integer
Default: 1

```
mds standby for name
```
Description:
Type: String
Default:

```
mds standby for rank
```
Description:
Type: 32-bit Integer
Default: -1

```
mds standby replay
```
Description:
Type: Boolean
Default: false

4.6 cephfs——ceph 文件系统选项工具

cephfs – ceph file system options utility

4.6.1 概述

SYNOPSIS

```
cephfs [ path command options ]
```

4.6.2 描述

DESCRIPTION
cephfs is a control utility for accessing and manipulating file layout and location data in the Ceph distributed file system.
cephfs 是个控制工具，用于访问和修改 ceph 分布式文件系统内的文件布局及位置信息。

Choose one of the following three commands:
可用命令有下面 3 个：

- show_layout View the layout information on a file or directory
- set_layout Set the layout information on a file or directory
- show_location View the location information on a file

- show_layout 查看一个文件或目录的布局信息；
- set_layout 设置一个文件或目录的布局信息；
- show_location 查看一个文件的位置信息；
- map 查看一个文件分片信息，包括其对象及所在归置组、OSD。

4.6.3 选项

OPTIONS
Your applicable options differ depending on whether you are setting or viewing layout/location.
可用选项因你是否在设置或查看布局、位置而不同。

查看选项

VIEWING OPTIONS:
-l --offset
Specify an offset for which to retrieve location data
指定检索位置数据的偏移量。

设置选项

SETTING OPTIONS:
-u --stripe_unit
Set the size of each stripe
设置每个条带的大小

-c --stripe_count
Set the number of stripes per object
设置每对象条带数量

-s --object_size
Set the size of the objects to stripe across
设置条带化的对象大小。

-p --pool
Set the pool (by numeric value, not name!) to use
指定要使用的存储池（数字，不是名字！）

-o --osd
Set the preferred OSD to use as the primary

设置优先用的主 OSD。

4.6.4 限制条件

LIMITATIONS
When setting layout data, the specified stripe unit and stripe count must multiply to the size of an object. Any parameters you don't set explicitly are left at the system defaults.
设置布局数据时，指定的条带单位和条带数必须乘以对象大小。任何未显式设置的参数都会按默认计。

Obviously setting the layout of a file and a directory means different things. Setting the layout of a file specifies exactly how to place the individual file. This must be done before writing any data to it. Truncating a file does not allow you to change the layout either.
很明显，设置一文件和一目录的布局含义不同。设置一文件的布局指示如何放置此文件，这必须在写入数据前完成；删节文件也不会更改其布局。

Setting the layout of a directory sets the "default layout", which is used to set the file layouts on any files subsequently created in the directory (or any subdirectory). Pre-existing files do not have their layouts changed.
设置一目录的布局实际上设置了“默认布局”，此设置会影响此目录下后续创建的所有文件、子目录。已经存在的文件其布局不会变动。

You'll notice that the layout information allows you to specify a preferred OSD for placement. This is allowed but is not recommended since it can dramatically unbalance your storage cluster's space utilization.
你也许注意到了，布局信息允许你指定归置的优先 OSD，这是可以的但不推荐，因为它会极大地破坏集群的使用空间均衡。

4.6.5 可用范围

AVAILABILITY
cephfs is part of the Ceph distributed file system. Please refer to the Ceph documentation at http://ceph.com/docs for more information.
cephfs 是 ceph 分布式文件系统的一部分，请参考位于 http://ceph.com/docs 的 ceph 文档。

SEE ALSO¶
ceph(8)

4.7 ceph-fuse——ceph 的用户空间客户端

ceph-fuse – fuse-based client for ceph

4.7.1 概述

SYNOPSIS

```
ceph-fuse [ -m monaddr:port ] mountpoint [ fuse options ]
```

4.7.2 描述

DESCRIPTION
ceph-fuse is a FUSE (File system in USErspace) client for Ceph distributed file system. It will mount a ceph file system (specified via the -m option for described by ceph.conf (see below) at the specific mount point.
ceph-fuse 是一个 Ceph 分布式文件系统的用户空间（FUSE, File system in USErspace）客户端，它会把 ceph 文件系统（用-m 选项指定）挂载到指定挂载点。

The file system can be unmounted with:
用下面的命令卸载文件系统：

```
fusermount -u mountpoint
```

or by sending SIGINT to the ceph-fuse process.
或者向 ceph-fuse 进程发送 SIGINT 信号。

4.7.3 选项

OPTIONS
Any options not recognized by ceph-fuse will be passed on to libfuse.
ceph-fuse 不认识的选项会接着传向 libfuse。

-d
Detach from console and daemonize after startup.
启动后从控制台分离，作为守护进程。

-c ceph.conf, --conf=ceph.conf
Use ceph.conf configuration file instead of the default /etc/ceph/ceph.conf to determine monitor addresses during startup.
用指定 ceph.conf 配置文件而非默认的/etc/ceph/ceph.conf 来找出启动时需要的监视器地址。

-m monaddress[:port]
Connect to specified monitor (instead of looking through ceph.conf).
连接到指定监视器（而非通过 ceph.conf 寻找）。

-r root_directory
Use root_directory as the mounted root, rather than the full Ceph tree.
以 root_directory 作为挂载的根，而不是整个 Ceph 树。

4.7.4 可用范围

AVAILABILITY
ceph-fuse is part of the Ceph distributed file system. Please refer to the Ceph documentation at http://ceph.com/docs for more information.
ceph-fuse 是 Ceph 分布式文件系统的一部分，更多信息参见 http://ceph.com/docs。

SEE ALSO
fusermount(8), ceph(8)

4.8 mount.ceph——挂载 ceph 文件系统

mount.ceph – mount a ceph file system

4.8.1 概述

SYNOPSIS

```
mount.ceph monaddr1[,monaddr2,...]:/[subdir] dir [ -o options ]
```

4.8.2 描述

DESCRIPTION
mount.ceph is a simple helper for mounting the Ceph file system on a Linux host. It serves to resolve monitor hostname(s) into IP addresses and read authentication keys from disk; the Linux kernel client component does most of the real work. In fact, it is possible to mount a non-authenticated Ceph file system without mount.ceph by specifying monitor address(es) by IP:
mount.ceph 是个简单的助手程序，用于在 Linux 主机上挂载 Ceph 文件系统。其作用是把监视器主机名解析为 IP 地址、并从硬盘读取认证密钥，Linux 内核客户端组件完成了大多数实际工作。事实上，如果用 IP 指定监视器，不需要 mount.ceph 就可以挂载无需认证的 ceph 文件系统：

```
mount -t ceph 1.2.3.4:/ mountpoint
```

Each monitor address monaddr takes the form host[:port]. If the port is not specified, the Ceph default of

6789 is assumed.
每个监视器地址 monaddr 格式都是 host[:port]，如果未指定端口，就用默认端口 6789。

Multiple monitor addresses can be separated by commas. Only one responsible monitor is needed to successfully mount; the client will learn about all monitors from any responsive monitor. However, it is a good idea to specify more than one in case one happens to be down at the time of mount.
多个监视器地址用逗号分隔。要成功地挂载，只要有一个可响应的监视器即可；客户端会从有响应的监视器学习到所有监视器。即便如此，最好还是指定多个监视器，以防挂载时它碰巧挂了。

A subdirectory subdir may be specified if a subset of the file system is to be mounted.
如果只想挂载一个文件系统子集，可以指定子目录 subdir。

Mount helper application conventions dictate that the first two options are device to be mounted and destination path. Options must be passed only after these fixed arguments.
mount 助手程序惯例要求头两个选项必须是要挂载的设备和目的路径，可选项必须在这些固定参数之后传入。

4.8.3 选项

OPTIONS
wsize
int, max write size. Default: none (writeback uses smaller of wsize and stripe unit)
整数，最大写尺寸。默认：无（回写用了较小的 wsize 和条带单元）
rsize
int (bytes), max readahead, multiple of 1024, Default: 524288 (512*1024)
整数（字节），最大预读，1024 的倍数。默认：524288（512*1024）
osdtimeout
int (seconds), Default: 60
整数，秒。默认 60
osdkeepalivetimeout
int, Default: 5
mount_timeout
int (seconds), Default: 60
osd_idle_ttl
int (seconds), Default: 60
caps_wanted_delay_min
int, cap release delay, Default: 5
整数，能力释放延时。默认 5
caps_wanted_delay_max
int, cap release delay, Default: 60
cap_release_safety
int, Default: calculated
readdir_max_entries
int, Default: 1024
readdir_max_bytes
int, Default: 524288 (512*1024)
write_congestion_kb
int (kb), max writeback in flight. scale with available memory. Default: calculated from available memory
整数（kb），行进中最大回写。随可用内存伸缩。默认：根据可用内存计算。
snapdirname
string, set the name of the hidden snapdir. Default: .snap
字符串，设置隐藏 snapdir 的名字。默认：.snap
name
RADOS user to authenticate as when using cephx. Default: guest
使用 cephx 是用以认证的 RADOS 用户名，默认：guest。
secret
secret key for use with cephx. This option is insecure because it exposes the secret on the command line. To avoid this, use the secretfile option.
用于 cephx 的密码，此选项不安全，因为它在命令行展示密码。要避免这种情况，改用 cecretfile 选项。
secretfile

path to file containing the secret key to use with cephx
用于 cephx、包含着密码的文件路径。

ip
my ip

noshare
create a new client instance, instead of sharing an existing instance of a client mounting the same cluster
创建新客户端例程，而不是共享挂载了相同集群的已存在客户端例程。

dirstat
funky cat dirname for stats, Default: off
臭名昭著的 cat dirname，用以获取状态。默认：off

nodirstat
no funky cat dirname for stats
不用臭名昭著的 cat dirname 获取状态信息。

rbytes
Report the recursive size of the directory contents for st_size on directories. Default: on
向目录的 st_size 报告目录内容的递归尺寸，默认：on

norbytes
Do not report the recursive size of the directory contents for st_size on directories.
不向目录的 st_size 报告目录内容的递归尺寸。

nocrc
no data crc on writes
写时不做数据 crc 校验

noasyncreaddir
no dcache readdir

4.8.4 实例

EXAMPLES
Mount the full file system:
挂载整个文件系统：

```
mount.ceph monhost:/ /mnt/foo
```

If there are multiple monitors:
若有多个监视器：

```
mount.ceph monhost1,monhost2,monhost3:/ /mnt/foo
```

If ceph-mon(8) is running on a non-standard port:
如果 ceph-mon(8)运行在非标准端口上：

```
mount.ceph monhost1:7000,monhost2:7000,monhost3:7000:/ /mnt/foo
```

To mount only part of the namespace:
只挂载名字空间的一部分：

```
mount.ceph monhost1:/some/small/thing /mnt/thing
```

Assuming mount.ceph(8) is installed properly, it should be automatically invoked by mount(8) like so:
假设 mount.ceph(8)正确安装了，它会被 mount(8)自动调用，像这样：

```
mount -t ceph monhost:/ /mnt/foo
```

4.8.5 可用范围

AVAILABILITY
mount.ceph is part of the Ceph distributed file system. Please refer to the Ceph documentation at http://ceph.com/docs for more information.

SEE ALSO
ceph-fuse(8), ceph(8)

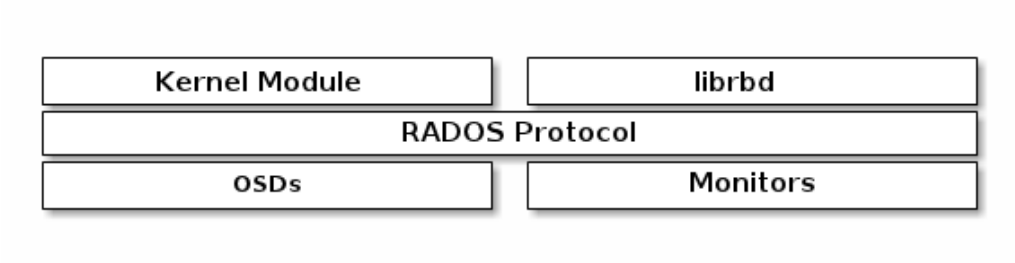
4.9 libcephfs (javadoc)

View the auto-generated JavaDoc pages for the CephFS Java bindings.
浏览自动生成的文档： JavaDoc pages for the CephFS Java bindings。

5 块设备

Block devices
A block is a sequence of bytes (for example, a 512-byte block of data). Block-based storage interfaces are the most common way to store data with rotating media such as hard disks, CDs, floppy disks, and even traditional 9-track tape. The ubiquity of block device interfaces makes a virtual block device an ideal candidate to interact with a mass data storage system like Ceph.
块是一个字节序列（例如，一个 512 字节的一块数据），基于块的存储接口是最常见的存储数据方法，它们基于旋转媒体，像硬盘、CD、软盘、甚至传统的 9 磁道磁带。无处不在的块设备接口使虚拟块设备成为与 ceph 这样的海量存储系统交互的理想之选。

Ceph block devices are thin-provisioned, resizable and store data striped over multiple OSDs in a Ceph cluster. Ceph block devices leverage RADOS capabilities such as snapshotting, replication and consistency. Ceph's RADOS Block Devices (RBD) interact with OSDs using kernel modules or the librbd library.
ceph 块设备是精简的、大小可调且数据条带化到集群内的多个 OSD。ceph 块设备均衡多个 RADOS 能力，如快照、复制和一致性，ceph 的 RADOS 块设备（RADOS Block Devices, RBD）用内核模块或 librbd 库与 OSD 交互。



Note: Kernel modules can use Linux page caching. For librbd-based applications, Ceph supports RBD Caching.
注意：内核模块可使用 Linux 页缓存。对基于 librbd 的应用程序，ceph 可提供 RBD 缓存。

Ceph's block devices deliver high performance with infinite scalability to kernel modules, or to KVMs such as Qemu, and cloud-based computing systems like OpenStack and CloudStack that rely on libvirt and Qemu to integrate with Ceph block devices. You can use the same cluster to operate the Ceph RADOS Gateway, the Ceph FS filesystem, and Ceph block devices simultaneously.
Ceph 块设备靠无限伸缩性提供了高性能，如向内核模块、或向 KVM（如 Qemu、依赖 libvirt 和 Qemu 的 OpenStack 和 CloudStack 云计算系统都可与 Ceph 块设备集成）。你可以用同一个集群同时运行 Ceph RADOS 网关、CephFS 文件系统、和 ceph 块设备。

Important To use RBD, you must have a running Ceph cluster.
重要：要使用 RBD，你必须有一个在运行的 ceph 集群。

- Commands
- Kernel Modules
- Snapshots
- QEMU
- libvirt
- Cache Settings
- OpenStack
- CloudStack
- Manpage rbd
- Manpage ceph-rbdnamer
- librbd

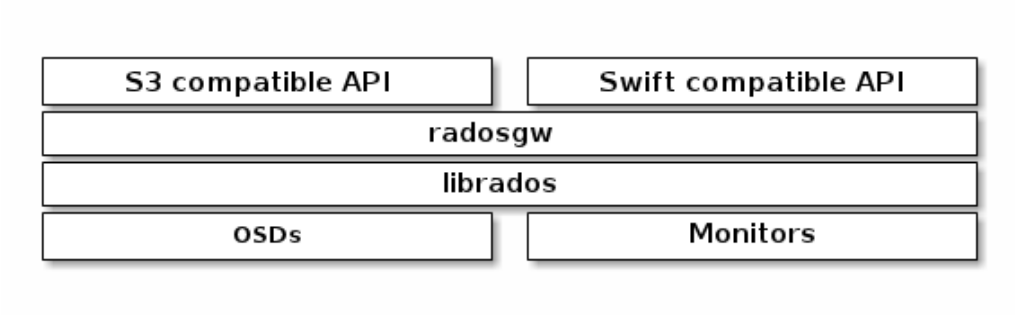
后续将补。。。

6 RADOS 网关

RADOS gateway
RADOS Gateway is an object storage interface built on top of librados to provide applications with a RESTful gateway to RADOS clusters. The RADOS Gateway supports two interfaces:
RADOS 网关是构建于 librados 之上的对象存储接口，可为应用程序提供到 RADOS 集群的 RESTful 网关，它现在提供了 2 个接口。

S3-compatible: Provides block storage functionality with an interface that is compatible with a large subset of the Amazon S3 RESTful API.
Swift-compatible: Provides block storage functionality with an interface that is compatible with a large subset of the OpenStack Swift API.
S3 兼容：用兼容大量亚马逊 S3 RESTful API 的接口提供了块存储功能。
Swift 兼容：用兼容大量 OpenStack Swift API 的接口提供了块存储功能。

RADOS Gateway is a FastCGI module for interacting with librados. Since it provides interfaces compatible with OpenStack Swift and Amazon S3, RADOS Gateway has its own user management. RADOS Gateway can store data in the same RADOS cluster used to store data from Ceph FS clients or RADOS block devices. The S3 and Swift APIs share a common namespace, so you may write data with one API and retrieve it with the other.
RADOS 网关是个与 librados 交互的 FastCGI 模块。因为它提供了与 OpenStack Swift 和 Amazon S3 兼容的接口，RADOS 要有它自己的用户管理。RADOS 网关可在存储 Ceph FS 客户端或 RADOS 块设备数据的集群里存储数据。S3 和 Swift API 共用一个名字空间，所以你可以用一个 API 写、然后用另一个检索。



Note RADOS Gateway does NOT use the CephFS metadata server.
注意：RADOS 网关不使用 CephFS 元数据服务器。

- Manual Install
- Configuration
- Config Reference
- Purging Temp Data
- S3 API
- Swift API
- Admin API
- Troubleshooting
- Manpage radosgw
- Manpage radosgw-admin

后续将补。。。

7 API 文档

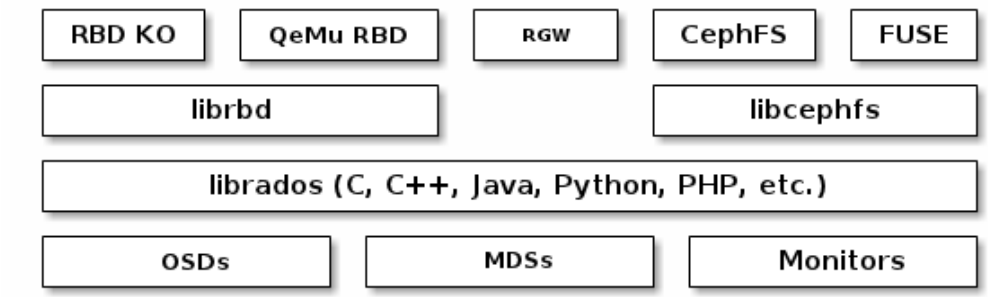
后续将补。。。

8 体系结构

Architecture
Ceph provides an infinitely scalable object storage system. It is based upon RADOS, which you can read about in RADOS - A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters. Its high-level features include providing a native interface to the object storage system via librados, and a number of service interfaces built on top of librados. These include:
CEPH 提供了一个可无限伸缩的对象存储系统，它基于 RADOS，见论文 RADOS - A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters。它的高级功能包括：基于 librados 的对象存储系统原生接口、和多种服务接口，它们有：

- Block Devices: The RADOS Block Device (RBD) service provides resizable, thin-provisioned block devices with snapshotting and cloning. Ceph stripes a block device across the cluster for high performance. Ceph supports both kernel objects (KO) and a QEMU hypervisor that uses librbd directly–avoiding the kernel object overhead for virtualized systems.
  块设备：RBD 服务提供了大小可调、精炼、支持快照和克隆的块设备。为提供高性能，ceph 把块设备条带化。ceph 同时支持直接使用 librbd 的内核对象（KO）和 QEMU 管理程序——避免了虚拟系统上的内核模块开销。
- RESTful Gateway: The RADOS Gateway (RGW) service provides RESTful APIs with interfaces that are compatible with Amazon S3 and OpenStack Swift.
  RESTful 网关： RADOS 网关（RADOS Gateway, RGW）服务提供了和 Amazon S3 和 OpenStack Swift 兼容的 RESTful API。
- Ceph FS: The Ceph Filesystem (CephFS) service provides a POSIX compliant filesystem usable with mount or as a filesytem in user space (FUSE).
  Ceph 文件系统：Ceph 文件系统兼容 POSIX，可以直接挂载或挂载为用户空间文件系统（FUSE）。

Ceph OSDs store all data–whether it comes through RBD, RGW, or CephFS–as objects in the object storage system. Ceph can run additional instances of OSDs, MDSs, and monitors for scalability and high availability. The following diagram depicts the high-level architecture.
Ceph 的 OSD 把所有数据以对象形式保存在对象存储系统中——不论它来自 RBD、RGW 还是 CephFS。Ceph 可运行多个 OSD、MDS 和监视器例程，以保证伸缩性和高可用性。下面图解了其高级体系结构。



8.1 消除局限性

Removing limitations

Today's storage systems have demonstrated an ability to scale out, but with some significant limitations: interfaces, session managers, and stateful sessions with a centralized point of access often limit the scalability of today's storage architectures. Furthermore, a centralized interface that dispatches requests from clients to server nodes within a cluster and subsequently routes responses from those server nodes back to clients will hit a scalability and/or performance limitation.
当今的存储系统已经表现出了扩张力，但是有着明显的局限性：接口、会话管理、和有状态会话都基于中央访问点，它通常限制了存储系统架构的伸缩性，另外，中央化的接口要把客户端请求调度到集群内的服务器节点，并把后续响应从服务器路由给客户端，这通常会碰到伸缩和/或性能问题。

Another problem for storage systems is the need to manually rebalance data when increasing or decreasing the size of a data cluster. Manual rebalancing works fine on small scales, but it is a nightmare at larger scales because hardware additions are common and hardware failure becomes an expectation rather than an exception when operating at the petabyte scale and beyond.
存储系统的另一个问题是增加、或减少数据集群尺寸时要手动重均衡数据，对小型集群来说手动均衡没什么问题，但对大型集群来说是噩梦，因为在 PB 级及更大的集群里，硬件增加和失败都是常态而非异常。

The operational challenges of managing legacy technologies with the burgeoning growth in the demand for unstructured storage makes legacy technologies inadequate for scaling into petabytes. Some legacy technologies (e.g., SAN) can be considerably more expensive, and more challenging to maintain when compared to using commodity hardware. Ceph uses commodity hardware, because it is substantially less expensive to purchase (or to replace), and it only requires standard system administration skills to use it.
急剧增长的非结构化存储需求使得用老技术不足以扩展到 PB 级。一些老技术（如 SAN）和普通硬件相比相当昂贵、且不易维护，而 ceph 用普通硬件，因为购买（或替换）它很便宜，而且只需要基本的系统管理技能即可。

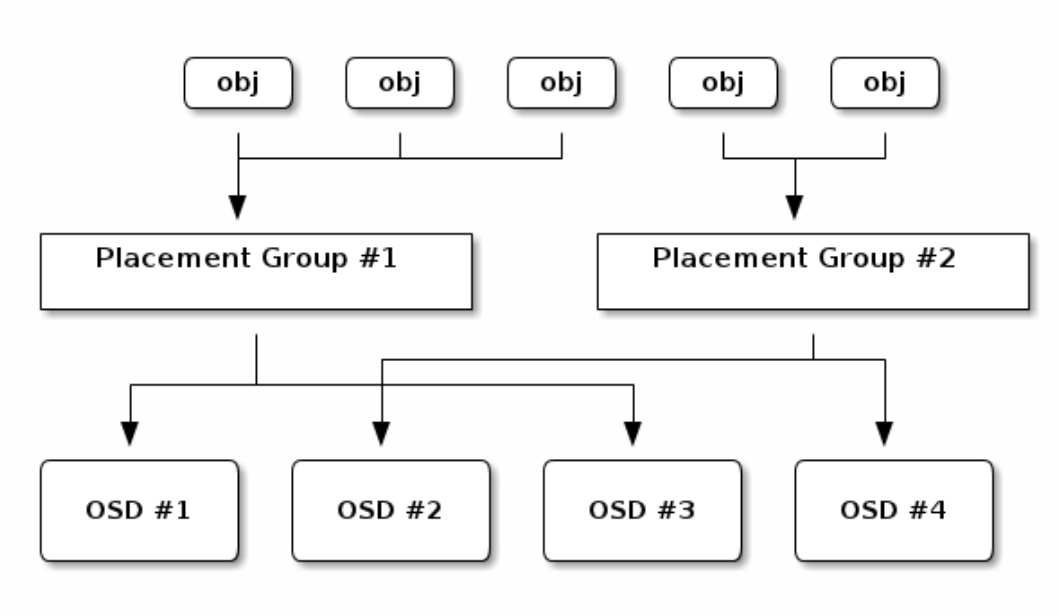
8.2 ceph 如何伸缩

How ceph scales

In traditional architectures, clients talk to a centralized component (e.g., a gateway, broker, API, facade, etc.), which acts as a single point of entry to a complex subsystem. This imposes a limit to both performance and scalability, while introducing a single point of failure (i.e., if the centralized component goes down, the whole system goes down, too).
在传统架构里，客户端沟通中央化的组件（如网关、中间件、API、facade 等等），它作为一个复杂子系统的单接触点，它引入单故障点的同时，也压制了性能和伸缩性（例如，如果中央化组件挂了，整个系统就挂了）。

Ceph uses a new and innovative approach. Ceph clients contact a Ceph monitor and retrieve a copy of the cluster map. The CRUSH algorithm allows a client to compute where data should be stored, and enables the client to contact the primary OSD to store or retrieve the data. The OSD also uses the CRUSH algorithm, but the OSD uses it to compute where replicas of data should be stored (and for rebalancing). For a detailed discussion of CRUSH, see CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data
Ceph 用了全新的方法，其客户端联系 ceph 监视器并获得一份集群运行图拷贝。CRUSH 算法让客户端计算数据应该存在哪里、并允许它联系主 OSD 来存储或检索数据。OSD 也用 CRUSH 算法，但是用于计算数据副本应该存到哪里（或用于重均衡）。关于 CRUSH 的详细讨论，见 CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data。

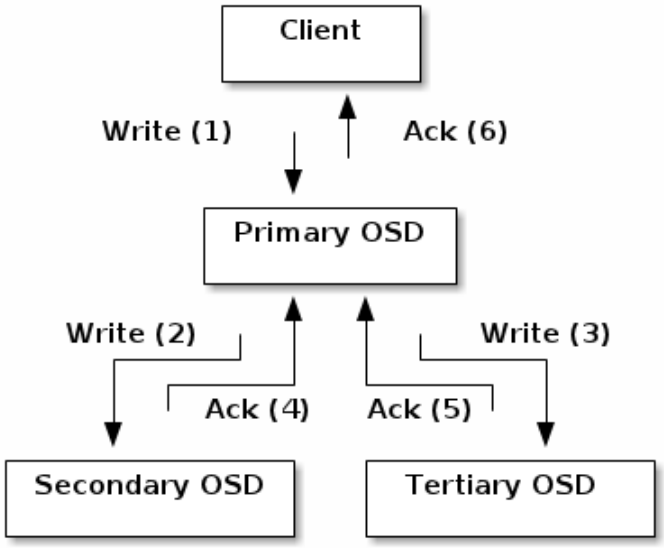
The Ceph storage system supports the notion of ‘Pools’, which are logical partitions for storing object data. Pools set ownership/access, the number of object replicas, the number of placement groups, and the CRUSH rule set to use. Each pool has a number of placement groups that are mapped dynamically to OSDs. When clients store data, CRUSH maps the object data to placement groups. The following diagram depicts how CRUSH maps objects to placement groups, and placement groups to OSDs.
Ceph 存储系统支持“池”概念，它是存储对象数据的逻辑分区。存储池设置了所有者及访问权限、对象副本数、归置组数量、要用的 CRUSH 规则集。每个存储池都有一定数量的归置组动态地映射到 OSD，客户端存数据时，CRUSH 把对象数据映射到归置组。下图描述了 CRUSH 如何把对象映射到归置组、然后归置组到 OSD。



Mapping objects to placement groups instead of directly to OSDs creates a layer of indirection between the OSD and the client. The cluster must be able to grow (or shrink) and rebalance data dynamically. If the client "knew" which OSD had the data, that would create a tight coupling between the client and the OSD. Instead, the CRUSH algorithm maps the data to a placement group and then maps the placement group to one or more OSDs. This layer of indirection allows Ceph to rebalance dynamically when new OSDs come online.
把对象映射到归置组而不是直接到 OSD，由此在 OSD 和客户端间产生了一个间接层。由于集群必须能增大或缩小、并动态地重均衡数据，如果客户端“知道”哪个 OSD 有数据，这将会导致客户端和 OSD 间密耦合，相反，CRUSH 算法把数据映射到归置组、然后再把归置组映射到一或多个 OSD。这个间接层可以让 ceph 在 OSD 上线时动态地重均衡。

With a copy of the cluster map and the CRUSH algorithm, the client can compute exactly which OSD to use when reading or writing a particular piece of data.
用一份集群运行图的拷贝和 CRUSH 算法，客户端能准确计算出到哪个 OSD 读、写数据片段。

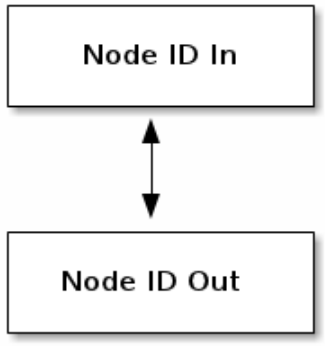
In a typical write scenario, a client uses the CRUSH algorithm to compute where to store data, maps the data to a placement group, then looks at the CRUSH map to identify the primary OSD for the placement group. Clients write data to the identified placement group in the primary OSD. Then, the primary OSD with its own copy of the CRUSH map identifies the secondary and tertiary OSDs for replication purposes, and replicates the data to the appropriate placement groups in the secondary and tertiary OSDs (as many OSDs as additional replicas), and responds to the client once it has confirmed the data was stored successfully.
在一个典型的写入场景中，一客户端用 CRUSH 算法计算往哪里存数据、映射数据到归置组、然后参阅 CRUSH 图找到归置组的主 OSD；客户端把数据写入目标归置组的主 OSD，然后这个主 OSD 再用它的 CRUSH 图副本找出用于放副本的第二、第三个 OSD，并把数据复制到适当的归置组所对应的第二、第三 OSD（要多少副本就有多少 OSD），最终，确认数据成功存储后反馈给客户端。



Since any network device has a limit to the number of concurrent connections it can support, a centralized system has a low physical limit at high scales. By enabling clients to contact nodes directly, Ceph increases both performance and total system capacity simultaneously, while removing a single point of failure. Ceph clients can maintain a session when they need to, and with a particular OSD instead of a centralized server.
由于任何网络设备都有其支持的最大并发连接限制，规模巨大时一个中央化的系统其物理局限性就暴露了。Ceph 允许客户端直接和节点联系，这在消除单故障点的同时，提升了性能和系统总容量。Ceph 客户端可按需维护和某 OSD 的会话，而不是一个中央服务器。

8.3 邻居感应节点

Peer-aware nodes
Ceph's cluster map determines whether a node in a network is in the Ceph cluster or out of the Ceph cluster.
Ceph 的集群图可确定网络中的一个节点是在集群内还是在集群外。



In many clustered architectures, the primary purpose of cluster membership is so that a centralized interface knows which hosts it can access. Ceph takes it a step further: Ceph's nodes are cluster aware. Each node knows about other nodes in the cluster. This enables Ceph's monitor, OSD, and metadata server daemons to interact directly with each other. One major benefit of this approach is that Ceph can utilize the CPU and RAM of its nodes to easily perform tasks that would bog down a centralized server.
在很多集群化体系结构中，设计集群成员的主要目的是让中央接口知道它能访问哪些主机。ceph 向前迈进了一步：ceph 的节点会感知集群，每个节点都知道集群内的其它节点，这使得 ceph 的监视器、OSD、和元数据服务器守护进程可以直接互相沟通。这种方法的一个主要优势是，ceph 能很容易地利用其节点的 CPU 和内存资源执行任务，而这样的任务会拖垮中央服务器。

Todo Explain OSD maps, Monitor Maps, MDS maps
待完成：解释 OSD 图、监视器图、MDS 图。

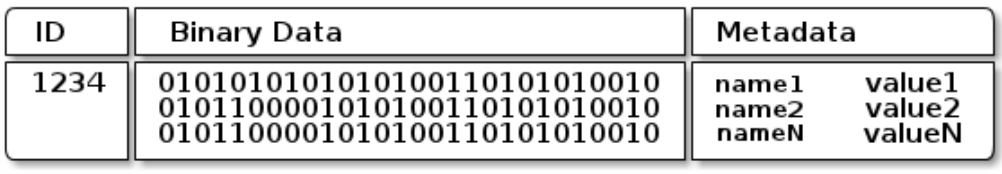
8.4 智能 OSD

Smart osds
Ceph OSDs join a cluster and report on their status. At the lowest level, the OSD status is up or down reflecting whether or not it is running and able to service requests. If an OSD is down and in the cluster, this status may indicate the failure of the OSD.
Ceph 的 OSD 加入集群后会持续报告自己的状态。在最低水平，OSD 状态为 up 或 down，反应了它是否在运行、能否提供服务。如果一个 OSD 状态为 down 且 in，表明 OSD 可能有故障。

With peer awareness, OSDs can communicate with other OSDs and monitors to perform tasks. OSDs take client requests to read data from or write data to pools, which have placement groups. When a client makes a request to write data to a primary OSD, the primary OSD knows how to determine which OSDs have the placement groups for the replica copies, and then update those OSDs. This means that OSDs can also take requests from other OSDs. With multiple replicas of data across OSDs, OSDs can also "peer" to ensure that the placement groups are in sync. See Placement Group States and Placement Group Concepts for details.
有了邻居感知功能，为执行任务 OSD 能和其它 OSD 和监视器通讯。OSD 根据客户端请求向存储池内的归置组读取或写入数据。当一客户端向某个主 OSD 请求写入数据时，主 OSD 知道如何确定哪个 OSD 持有用于副本的归置组，继而更新那些 OSD。这意味着 OSD 也能接受其他 OSD 的请求，用散布于多个 OSD 的多份数据副本，OSD 也能通过相互查询来确保归置组同步无误。详情见 Placement Group States 和 Placement Group Concepts。

If an OSD is not running (e.g., it crashes), the OSD cannot notify the monitor that it is down. The monitor can ping an OSD periodically to ensure that it is running. However, Ceph also empowers OSDs to determine if a neighboring OSD is down, to update the cluster map and to report it to the monitor(s). When an OSD is down, the data in the placement group is said to be degraded. If the OSD is down and in, but subsequently taken out of the cluster, the OSDs receive an update to the cluster map and rebalance the placement groups within the cluster automatically.
如果一 OSD 没在运行（比如，它崩溃了），这个 OSD 就不能告诉监视器它挂了。监视器可以周期性地 ping 一个 OSD 来确保它在运行，即便这样，ceph 也授权 OSD 们探测它的邻居 OSD 是否还活着，以此来更新集群运行图并报告给监视器。一个 OSD 挂了时，对应归置组里的数据就降级了，如果 OSD 状态为 down 且 in，但随后被踢出了集群，其余 OSD 会收到一个集群运行图更新，并自动重均衡集群内的归置组。

OSDs store all data as objects in a flat namespace (e.g., no hierarchy of directories). An object has an identifier, binary data, and metadata consisting of a set of name/value pairs. The semantics are completely up to the client. For example, CephFS uses metadata to store file attributes such as the file owner, created date, last modified date, and so forth.
OSD 在扁平的名字空间里把所有数据存储为对象（例如，没有目录层次）。对象包含一个标识符、二进制数据、和由名字/值配对组成的元数据，语义完全取决于客户端。例如，CephFS 用元数据存储文件属性，如文件所有者、创建日期、最后修改日期等等。



As part of maintaining data consistency and cleanliness, Ceph OSDs can also scrub the data. That is, Ceph OSDs can compare object metadata across replicas to catch OSD bugs or filesystem errors (daily). OSDs can also do deeper scrubbing by comparing data in objects bit-for-bit to find bad sectors on a disk that weren't apparent in a light scrub (weekly).
作为维护工作的一部分，不但要保证数据的一致性和清洁性，OSD 也能洗刷数据，就是说，Ceph OSD 能比较

对象元数据的几个副本以捕捉 OSD 缺陷或文件系统错误（每天）。OSD 也能做深度洗刷，即按字节比较对象中的数据以找出轻度洗刷（每周）时未发现的硬盘坏扇区。

Todo explain “classes”
待完成：解释“classes”

8.5 监视器法定人数

Monitor quorums
Ceph’s monitors maintain a master copy of the cluster map. So Ceph daemons and clients merely contact the monitor periodically to ensure they have the most recent copy of the cluster map. Ceph monitors are light-weight processes, but for added reliability and fault tolerance, Ceph supports distributed monitors. Ceph must have agreement among various monitor instances regarding the state of the cluster. To establish a consensus, Ceph always uses an odd number of monitors (e.g., 1, 3, 5, 7, etc) and the Paxos algorithm in order to establish a consensus.
ceph 的监视器维护着集群运行图的原稿，所以 ceph 守护进程和客户端只是周期性地联系监视器以保证他们有运行图的最新副本。ceph 监视器是轻量级进程，但是增强了可靠性和容错能力，ceph 支持分布式监视器。不管 Ceph 集群状态如何，其不同监视器例程必须达成一致，为此，ceph 总是使用奇数个监视器（如：1、3、5、7……）和 Paxos 算法来达成一致意见。

8.6 mds

The Ceph filesystem service is provided by a daemon called ceph-mds. It uses RADOS to store all the filesystem metadata (directories, file ownership, access modes, etc), and directs clients to access RADOS directly for the file contents. The Ceph filesystem aims for POSIX compatibility. ceph-mds can run as a single process, or it can be distributed out to multiple physical machines, either for high availability or for scalability.
Ceph 文件系统服务由名为 ceph-mds 的守护进程提供，它使用 RADOS 存储所有文件系统元数据（目录、文件所有者、访问权限等等），并指导客户端直接访问 RADOS 获取文件内容。Ceph 力争兼容 POSIX。ceph-mds 可以只运行一个，也可以分布于多台物理机器，以获得高可用性或伸缩性。

- High Availability: The extra ceph-mds instances can be standby, ready to take over the duties of any failed ceph-mds that was active. This is easy because all the data, including the journal, is stored on RADOS. The transition is triggered automatically by ceph-mon.
  高可用性：多余的 ceph-mds 例程可处于 standby（待命）状态，随时准备替下之前处于 active（活跃）状态的失败 ceph-mds。这可以轻易做到，因为所有数据、包括日志都存储在 RADOS 上，这个转换过程由 ceph-mon 自动触发。
- Scalability: Multiple ceph-mds instances can be active, and they will split the directory tree into subtrees (and shards of a single busy directory), effectively balancing the load amongst all active servers.
  可伸缩性：多个 ceph-mds 例程可以处于 active 状态，并且它们会把目录树分割为子树（和单个忙碌目录的碎片），在所有活跃服务器间高效地均衡负载。

译者曰：虽然文档这么说，但实践中还不推荐这样做，mds 稳定性尚不理想。多个活跃的 mds 远没一个稳定，即便如此，您也应该先配置起几个 mds 备用。

Combinations of standby and active etc are possible, for example running 3 active ceph-mds instances for scaling, and one standby intance for high availability.
待命和活跃 MDS 可组合，例如，运行 3 个活跃 ceph-mds 例程以实现扩展、和 1 个待命例程以实现高可用性。

8.7 客户端接口

Client interfaces

8.7.1 认证和授权

Authentication and authorization
Ceph clients can authenticate their users with Ceph monitors, OSDs and metadata servers. Authenticated

users gain authorization to read, write and execute Ceph commands. The Cephx authentication system is similar to Kerberos, but avoids a single point of failure to ensure scalability and high availability. For details on Cephx, see Ceph Authentication and Authorization.
ceph 客户端能通过 ceph 监视器、OSD 和元数据服务器认证它们的用户，通过认证的用户获得读、写和执行 ceph 命令的授权。cephx 认证系统和 Kerberos 相似，但消除了单故障点，进而保证了可扩展性和高可用性。关于 cephx 的细节，参见 ceph 认证及授权。

8.7.2 librados

Todo Snapshotting, Import/Export, Backup
Todo native APIs
待完成：拍快照、导入/导出、备份。
待完成：原生 API。

8.7.3 RBD

RBD stripes a block device image over multiple objects in the cluster, where each object gets mapped to a placement group and distributed, and the placement groups are spread across separate ceph-osd daemons throughout the cluster.
RBD 把一个设备映像条带化到集群内的多个对象，其中每个对象都被映射到一个归置组并分布出去，这些归置组会散播到整个集群的某些 ceph-osd 守护进程。

> Important Striping allows RBD block devices to perform better than a single server could!
> 重要：条带化会使 RBD 块设备比单台服务器运行的更好。

RBD's thin-provisioned snapshottable block devices are an attractive option for virtualization and cloud computing. In virtual machine scenarios, people typically deploy RBD with the rbd network storage driver in Qemu/KVM, where the host machine uses librbd to provide a block device service to the guest. Many cloud computing stacks use libvirt to integrate with hypervisors. You can use RBD thin-provisioned block devices with Qemu and libvirt to support OpenStack and CloudStack among other solutions.
RBD 的简配、可快照块设备对虚拟化和云计算很有吸引力。在虚拟机场景中，人们一般会用 Qemu/KVM 中的 rbd 网络驱动部署 RBD，虚拟主机用 librbd 向客户提供块设备服务；很多云计算堆栈用 libvirt 和管理程序集成，你可以用 RBD 的简配块设备搭配 Qemu 和 libvirt 来支持 OpenStack 和 CloudStack。

While we do not provide librbd support with other hypervisors at this time, you may also use RBD kernel objects to provide a block device to a client. Other virtualization technologies such as Xen can access the RBD kernel object(s). This is done with the command-line tool rbd.
现在我们还没对其它虚拟机管理程序提供 librbd 支持，你可以用 RBD 内核对象把块设备提供给客户端。其它虚拟化技术，像 Xen 能访问 RBD 内核对象，这可以用命令行工具 rbd 来实现。

8.7.4 RGW

The RADOS Gateway daemon, radosgw, is a FastCGI service that provides a RESTful HTTP API to store objects and metadata. It layers on top of RADOS with its own data formats, and maintains its own user database, authentication, and access control. The RADOS Gateway uses a unified namespace, which means you can use either the OpenStack Swift-compatible API or the Amazon S3-compatible API. For example, you can write data using the S3-comptable API with one application and then read data using the Swift-compatible API with another application.
RADOS 网关守护进程是 radosgw，它是一个 FastCGI 服务，提供了 RESTful HTTP API 用于存储对象和元数据。它坐落于 RADOS 之上，有自己的数据格式，并维护着自己的用户数据库、认证、和访问控制。RADOS 网关使用统一的名字空间，也就是说，你可以用 OpenStack Swift 兼容的 API 或者 Amazon S3 兼容的 API；例如，你可以用一个程序以 S3 兼容 API 写入数据、然后用另一个程序以 Swift 兼容 API 读出。

See RADOS Gateway for details.
详情参见 RADOS 网关。

8.7.5 cephfs

Todo cephfs, ceph-fuse
待完成：cephfs、ceph-fuse