

CEPH 官方文档中译

原 文: <http://ceph.com/docs/master/start/>

翻 译 人: 张绍文<gongfan193@gmail.com>

版 本: v2.0

许 可: 同原文

托 管 在: https://github.com/drunkard/docs_zh

1 开始.....	13
1.1 5 分钟快速入门.....	13
1.1.1 安装 Debian/Ubuntu.....	14
1.1.2 安装 ceph 软件包.....	14
1.1.3 写配置文件.....	14
1.1.4 部署配置.....	16
1.1.5 启动 ceph 集群.....	16
1.1.6 把密钥环拷贝到客户端.....	17
1.1.7 继续其他快速入门.....	17
1.2 块设备快速入门.....	17
1.3 CEPHFS 快速入门.....	18
1.3.1 内核空间驱动.....	18
1.3.2 用户空间(fuse).....	18
1.3.3 额外信息.....	18
1.4 对象存储快速入门.....	18
1.4.1 安装 Apache 和 FastCGI.....	19
1.4.2 安装 RADOS 网关.....	19
1.4.3 修改 ceph 配置文件.....	19
1.4.4 创建数据目录.....	20
1.4.5 创建网关配置文件.....	20
1.4.6 添加 FastCGI 脚本.....	21
1.4.7 生成密钥环和密钥.....	21
1.4.8 创建用户.....	22
1.4.9 启用 SSL.....	23
1.5 加入 ceph 社区.....	24
1.6 手动安装 ceph.....	24
2 安装.....	25
2.1 硬件推荐.....	25
2.1.1 CPU.....	25
2.1.2 内存.....	26
2.1.3 数据存储.....	26
2.1.3.1 硬盘.....	26
2.1.3.2 固态硬盘.....	27
2.1.3.3 控制器.....	28
2.1.3.4 其他注意事项.....	28
2.1.4 网络.....	28
2.1.5 故障域.....	29
2.1.6 最低硬件推荐.....	29
2.1.7 生产集群实例.....	30
2.2 推荐操作系统.....	30
2.2.1 CEPH 依赖.....	30
2.2.2 系统平台.....	31
2.3 Debian/Ubuntu 包的安装.....	32
2.3.1 安装发布密钥.....	32
2.3.2 添加软件源.....	32
2.3.2.1 稳定版——Bobtail.....	32

2.3.2.2 稳定版——Argonaut.....	32
2.3.2.3 开发版软件包.....	33
2.3.2.4 开发测试软件包.....	33
2.3.3 安装软件包.....	33
2.4 RPM 包的安装.....	34
2.4.1 安装发布密钥.....	34
2.4.2 添加发布包.....	34
2.4.2.1 稳定版——Bobtail.....	34
2.4.2.2 开发版包.....	34
2.4.3 安装软件包.....	35
2.5 升级 ceph.....	35
2.5.1 升级 OSD.....	36
2.5.2 升级监视器.....	36
2.5.3 升级元数据服务器.....	37
2.5.4 升级客户端.....	37
2.5.5 从 Argonaut 升级到 Bobtail.....	37
2.5.5.1 认证.....	38
2.5.5.2 监视器 on-wire 协议.....	38
2.5.5.3 RBD 映像.....	38
2.6 构建前提.....	39
2.6.1 Ubuntu.....	40
2.6.2 Debian.....	40
2.6.3 openSUSE 11.2 (及更高版)	40
2.7 下载 ceph 发布压缩包.....	40
2.8 安装 git 版.....	40
2.8.1 安装 git.....	41
2.8.2 生成 SSH 密钥对.....	41
2.8.3 添加密钥.....	41
2.9 克隆 ceph 源码仓库.....	41
2.9.1 克隆源码.....	42
2.9.2 选择分支.....	42
2.10 构建 ceph.....	42
2.11 安装 oprofile.....	43
2.11.1 安装.....	43
2.11.2 编译适合分析的 ceph.....	43
2.11.3 ceph 配置.....	43
2.12 构建 ceph 包.....	43
2.12.1 高级包管理器 (APT)	44
2.12.2 RPM 包管理器.....	44
2.13 贡献代码.....	44
3 RADOS 对象存储.....	46
3.1 配置.....	46
3.1.1 硬盘和文件系统.....	47
3.1.1.1 准备硬盘.....	47
3.1.1.2 文件系统.....	47
3.1.1.3 文件系统背景知识.....	48
3.1.2 ceph 的配置.....	48
3.1.2.1 配置文件 ceph.conf.....	48
3.1.2.2 配置段落.....	49
3.1.2.3 元变量.....	51
3.1.2.4 共有设置.....	51
3.1.2.5 网络.....	52
3.1.2.6 认证.....	52
3.1.2.7 监视器集群.....	52
3.1.2.8 OSDs.....	53

3.1.2.9 日志、调试.....	54
3.1.2.10 ceph.conf 实例.....	55
3.1.2.11 运行时更改.....	56
3.1.2.12 查看运行时配置.....	57
3.1.2.13 运行多个集群.....	57
3.1.3 网络配置.....	58
3.1.3.1 防火墙 iptables.....	59
3.1.3.1.1 监视器防火墙.....	59
3.1.3.1.2 MDS 防火墙.....	60
3.1.3.1.3 OSD 防火墙.....	60
3.1.3.2 ceph 网络.....	61
3.1.3.2.1 公公网.....	61
3.1.3.2.2 集群网.....	61
3.1.3.3 ceph 守护进程.....	62
3.1.3.4 网络配置选项.....	63
3.1.3.4.1 公公网.....	63
3.1.3.4.2 集群网.....	63
3.1.3.4.3 端口绑定.....	64
3.1.3.4.4 主机.....	64
3.1.3.4.5 TCP.....	65
3.1.4 认证配置 (cephx 配置)	65
3.1.4.1 启用/禁用认证.....	65
3.1.4.2 密钥.....	67
3.1.4.3 签名.....	68
3.1.4.4 生存期.....	69
3.1.5 监视器配置.....	69
3.1.5.1 背景.....	69
3.1.5.1.1 集群运行图.....	70
3.1.5.1.2 监视器法定人数.....	70
3.1.5.1.3 一致性.....	70
3.1.5.1.4 初始监视器.....	71
3.1.5.2 监视器的配置.....	71
3.1.5.2.1 最小配置.....	72
3.1.5.2.2 集群 ID.....	72
3.1.5.2.3 初始成员.....	73
3.1.5.2.4 数据.....	73
3.1.5.2.5 存储容量.....	74
3.1.5.2.6 心跳.....	75
3.1.5.2.7 监视器存储同步.....	75
3.1.5.2.8 Slurp (暴食?)	78
3.1.5.2.9 时钟.....	79
3.1.5.2.10 客户端.....	79
3.1.5.3 杂项.....	80
3.1.6 心跳配置 (监视器/OSD 交互的配置)	81
3.1.6.1 OSD 验证心跳.....	81
3.1.6.2 OSD 报告死亡 OSD.....	81
3.1.6.3 OSD 报告互联失败.....	82
3.1.6.4 OSD 报告自己的状态.....	82
3.1.6.5 配置选项.....	83
3.1.6.5.1 监视器选项.....	83
3.1.6.5.2 OSD 选项.....	84
3.1.7 OSD 配置.....	85
3.1.7.1 常规配置.....	86
3.1.7.2 日志选项.....	87
3.1.7.3 监视器和 OSD 的交互.....	88

3.1.7.4 数据归置.....	88
3.1.7.5 洗刷.....	88
3.1.7.6 操作数.....	89
3.1.7.7 回填.....	90
3.1.7.8 OSD 运行图.....	91
3.1.7.9 恢复.....	91
3.1.7.10 杂项.....	92
3.1.8 文件存储配置.....	93
3.1.8.1 扩展属性.....	94
3.1.8.2 同步间隔.....	94
3.1.8.3 flusher (回冲器?)	95
3.1.8.4 队列.....	95
3.1.8.5 超时选项.....	96
3.1.8.6 B-tree 文件系统.....	96
3.1.8.7 日志 (文件系统)	97
3.1.8.8 杂项.....	97
3.1.9 日志配置.....	98
3.1.10 存储池、归置组和 CRUSH 配置.....	100
3.1.11 消息.....	102
3.1.12 常规配置.....	103
3.2 ceph 部署.....	104
3.2.1 转移到 ceph-deploy.....	105
3.2.1.1 监视器密钥环.....	105
3.2.1.2 用默认路径.....	105
3.2.2 飞前检查.....	105
3.2.2.1 选装一个操作系统.....	106
3.2.2.2 安装 SSH 服务器.....	106
3.2.2.3 创建一用户.....	106
3.2.2.4 配置 SSH.....	106
3.2.2.5 安装 ceph-deploy.....	107
3.2.2.6 确认连通性.....	107
3.2.3 软件包管理.....	107
3.2.3.1 安装.....	107
3.2.3.2 卸载.....	108
3.2.4 创建一集群.....	108
3.2.4.1 用法.....	109
3.2.4.2 命名集群.....	109
3.2.5 增加/删除监视器.....	109
3.2.5.1 增加一监视器.....	110
3.2.5.2 删除一监视器.....	110
3.2.6 密钥管理.....	110
3.2.6.1 收集密钥.....	110
3.2.6.2 忘记密钥.....	111
3.2.7 增加/删除 OSD.....	111
3.2.7.1 列举磁盘.....	111
3.2.7.2 擦净磁盘.....	111
3.2.7.3 准备 OSD.....	112
3.2.7.4 激活 OSD.....	112
3.2.7.5 创建 OSD.....	112
3.2.7.6 拆除 OSD.....	113
3.2.8 增加/拆除元数据服务器.....	113
3.2.8.1 增加一元数据服务器.....	113
3.2.8.2 删除一元数据服务器.....	113
3.2.9 清除一主机.....	113
3.2.9.1 清除数据.....	113

3.2.9.2 Purge.....	114
3.2.10 管理任务.....	114
3.2.10.1 创建一管理主机.....	114
3.2.10.2 分发配置文件.....	114
3.2.10.3 检索配置文件.....	114
3.2.11 用 mkcephfs 部署 (已过时)	114
3.2.11.1 允许 root 登录集群主机.....	114
3.2.11.2 把配置文件复制到所有节点.....	115
3.2.11.3 创建默认目录.....	115
3.2.11.4 把硬盘挂载到数据目录.....	116
3.2.11.5 运行 mkcephfs.....	116
3.3 运维.....	117
3.3.1 操纵集群.....	118
3.3.1.1 用 upstart 控制 ceph.....	118
3.3.1.1.1 启动集群.....	118
3.3.1.1.2 关闭集群.....	118
3.3.1.1.3 启动一类进程.....	119
3.3.1.1.4 停止一类进程.....	119
3.3.1.1.5 启动一个进程.....	119
3.3.1.1.6 停止一个进程.....	119
3.3.1.2 以服务运行 ceph.....	119
3.3.1.2.1 启动集群.....	121
3.3.1.2.2 停止集群.....	121
3.3.2 监控集群.....	122
3.3.2.1 交互模式.....	122
3.3.2.2 检查集群健康状况.....	122
3.3.2.3 观察集群.....	122
3.3.2.4 检查集群状态.....	123
3.3.2.5 检查 OSD 状态.....	123
3.3.2.6 检查监视器状态.....	124
3.3.2.7 检查 MDS 状态.....	124
3.3.2.8 检查归置组状态.....	125
3.3.2.9 使用管理套接字.....	125
3.3.3 监控 OSD 和归置组.....	125
3.3.3.1 监控 OSD.....	125
3.3.3.2 归置组集.....	127
3.3.3.3 节点互联.....	128
3.3.3.4 归置组状态的监控.....	128
3.3.3.4.1 存储池在建中.....	131
3.3.3.4.2 互联建立中.....	131
3.3.3.4.3 活跃.....	132
3.3.3.4.4 整洁.....	132
3.3.3.4.5 已降级.....	132
3.3.3.4.6 恢复中.....	133
3.3.3.4.7 回填中.....	133
3.3.3.4.8 被重映射.....	134
3.3.3.4.9 发蔫 (stale)	134
3.3.3.5 找出故障归置组.....	134
3.3.3.6 定位对象.....	135
3.3.4 认证概览.....	136
3.3.4.1 ceph 认证 (cephx)	137
3.3.4.2 ceph 授权 (能力)	139
3.3.4.3 cephx 的局限性.....	140
3.3.5 cephx 认证.....	140
3.3.5.1 配置 cephx.....	141

3.3.5.1.1 client.admin 密钥.....	141
3.3.5.1.2 监视器密钥环.....	142
3.3.5.1.3 启用 cephx.....	142
3.3.5.1.4 禁用 cephx.....	143
3.3.5.1.5 守护进程密钥环.....	143
3.3.5.2 cephx 管理.....	144
3.3.5.2.1 增加密钥.....	144
3.3.5.2.2 删密钥.....	145
3.3.5.2.3 列出集群内的密钥.....	145
3.3.5.3 cephx 命令行选项.....	145
3.3.5.4 向后兼容性.....	146
3.3.6 数据归置概览.....	148
3.3.7 存储池.....	148
3.3.7.1 列出存储池.....	149
3.3.7.2 创建存储池.....	149
3.3.7.3 删除存储池.....	150
3.3.7.4 重命名存储池.....	150
3.3.7.5 查看存储池统计信息.....	150
3.3.7.6 拍下存储池快照.....	151
3.3.7.7 删除存储池快照.....	151
3.3.7.8 设置存储池键值.....	151
3.3.7.9 获取存储池键值.....	152
3.3.7.10 设置对象副本数.....	152
3.3.7.11 获取对象副本数.....	153
3.3.8 归置组.....	153
3.3.8.1 设置归置组数量.....	154
3.3.8.2 获取归置组数量.....	154
3.3.8.3 获取归置组统计信息.....	154
3.3.8.4 获取卡住的归置组统计信息.....	154
3.3.8.5 获取一归置组运行图.....	155
3.3.8.6 获取一 PG 的统计信息.....	155
3.3.8.7 洗刷归置组.....	155
3.3.8.8 恢复丢失的.....	155
3.3.8.8.1 归置组状态.....	156
3.3.8.8.2 归置组术语解释.....	157
3.3.9 CRUSH 图.....	158
3.3.9.1 编辑 CRUSH 图.....	159
3.3.9.1.1 获取 CRUSH 图.....	160
3.3.9.1.2 反编译 CRUSH 图.....	160
3.3.9.1.3 编译 CRUSH 图.....	160
3.3.9.1.4 设置 CRUSH 图.....	160
3.3.9.2 CRUSH 图参数.....	161
3.3.9.2.1 CRUSH 图之设备.....	161
3.3.9.2.2 CRUSH 图之桶类型.....	162
3.3.9.2.3 CRUSH 图之桶层次.....	162
3.3.9.2.4 CRUSH 图之规则.....	165
3.3.9.3 不同存储池置于不同 OSD.....	167
3.3.9.4 增加/移动 OSD.....	169
3.3.9.5 调整一 OSD 的 CRUSH 权重.....	170
3.3.9.6 删除 OSD.....	170
3.3.9.7 移动桶.....	171
3.3.9.8 可调选项.....	171
3.3.9.8.1 过时值的影响.....	172
3.3.9.8.2 CRUSH_TUNABLES.....	172
3.3.9.8.3 CRUSH_TUNABLES2.....	172

3.3.9.8.4 支持 CRUSH_TUNABLES 的客户端版本.....	172
3.3.9.8.5 支持 CRUSH_TUNABLES2 的客户端版本.....	173
3.3.9.8.6 一些要点.....	173
3.3.9.8.7 调整 CRUSH.....	173
3.3.9.8.8 调整 CRUSH——强硬方法.....	174
3.3.9.8.9 遗留值.....	174
3.3.10 增加/删除 OSD.....	174
3.3.10.1 增加 OSD.....	174
3.3.10.1.1 部署硬件.....	175
3.3.10.1.2 安装必要软件.....	175
3.3.10.1.3 增加 OSD (手动)	175
3.3.10.1.4 增加 OSD (chef)	177
3.3.10.1.5 启动 OSD.....	178
3.3.10.1.6 把 OSD 推进集群.....	178
3.3.10.1.7 观察数据迁移.....	178
3.3.10.2 删除 OSD (手动)	178
3.3.10.2.1 把 OSD 踢出集群.....	179
3.3.10.2.2 观察数据迁移.....	179
3.3.10.2.3 停止 OSD.....	179
3.3.10.2.4 删除 OSD.....	179
3.3.11 增加/删除监视器.....	180
3.3.11.1 增加监视器.....	180
3.3.11.1.1 部署硬件.....	181
3.3.11.1.2 安装必要软件.....	181
3.3.11.1.3 增加监视器 (手动)	181
3.3.11.2 删除监视器.....	182
3.3.11.2.1 删除监视器 (手动)	182
3.3.11.2.2 从不健康集群删除监视器.....	183
3.3.11.3 更改监视器 IP 地址.....	183
3.3.11.3.1 一致性要求.....	184
3.3.11.3.2 更改监视器 IP 地址 (正确方法)	184
3.3.11.3.3 更改监视器 IP 地址 (凌乱方法)	185
3.3.12 命令参考.....	186
3.3.12.1 监视器命令.....	186
3.3.12.2 系统命令.....	186
3.3.12.3 认证子系统.....	187
3.3.12.4 归置组子系统.....	187
3.3.12.5 OSD 子系统.....	188
3.3.12.6 MDS 子系统.....	191
3.3.12.7 监视器子系统.....	192
3.3.13 Ceph 社区.....	193
3.3.14 监视器故障恢复.....	194
3.3.14.1 客户端不能连接/挂载.....	194
3.3.14.2 挂掉监视器时的延时.....	194
3.3.15 OSD 故障排除.....	195
3.3.15.1 收集 OSD 数据.....	195
3.3.15.1.1 ceph 日志.....	195
3.3.15.1.2 管理套接字.....	195
3.3.15.1.3 显示剩余空间.....	196
3.3.15.1.4 I/O 统计信息.....	196
3.3.15.1.5 诊断消息.....	196
3.3.15.2 停止自动重均衡.....	196
3.3.15.3 OSD 没运行.....	197
3.3.15.3.1 OSD 起不来.....	197
3.3.15.3.2 OSD 失败.....	197

3.3.15.3.3 硬盘没剩余空间.....	198
3.3.15.4 OSD 龟速或无响应.....	199
3.3.15.4.1 网络问题.....	199
3.3.15.4.2 驱动器配置.....	199
3.3.15.4.3 坏扇区/碎片化硬盘.....	200
3.3.15.4.4 监视器和 OSD 蜗居.....	200
3.3.15.4.5 进程蜗居.....	200
3.3.15.4.6 日志级别.....	200
3.3.15.4.7 恢复节流.....	201
3.3.15.4.8 内核版本.....	201
3.3.15.4.9 内核与 syncfs 问题.....	201
3.3.15.4.10 文件系统问题.....	201
3.3.15.4.11 内存不足.....	201
3.3.15.4.12 old requests 或 slow requests.....	201
3.3.15.5 打摆子的 OSD.....	202
3.3.16 PG 错误排障.....	203
3.3.16.1 归置组总不整洁.....	203
3.3.16.2 卡住的归置组.....	204
3.3.16.3 归置组挂了——连接建立失败.....	204
3.3.16.4 未找到的对象.....	205
3.3.16.5 无根归置组.....	207
3.3.16.6 只有几个 OSD 接收数据.....	207
3.3.16.7 不能写入数据.....	208
3.3.17 日志记录和调试.....	208
3.3.17.1 运行时.....	208
3.3.17.2 启动时.....	209
3.3.17.3 加快日志滚动.....	209
3.3.17.4 Valgrind.....	210
3.3.17.5 子系统，日志和调试选项.....	210
3.3.17.5.1 ceph 子系统概览.....	210
3.3.17.5.2 日志记录选项.....	211
3.3.17.5.3 OSD.....	213
3.3.17.5.4 Filestore.....	214
3.3.17.5.5 MDS.....	214
3.3.17.5.6 RADOS 网关.....	215
3.3.18 CPU 剖析.....	216
3.3.18.1 初始化 oprofile.....	216
3.3.18.2 启动 oprofile.....	216
3.3.18.3 停止 oprofile.....	216
3.3.18.4 查看 oprofile 运行结果.....	216
3.3.18.5 重置 oprofile.....	217
3.3.19 内存剖析.....	217
3.3.19.1 启动剖析器.....	217
3.3.19.2 打印统计.....	217
3.3.19.3 转储堆栈信息.....	218
3.3.19.4 释放内存.....	218
3.3.19.5 停止剖析器.....	218
3.4 手册页<尚未整理>.....	218
3.5 故障排除.....	219
3.5.1 Ceph 社区.....	219
3.5.2 日志记录和调试.....	219
3.5.3 监视器故障恢复.....	219
3.5.4 OSD 故障排除.....	219
3.5.5 PG 故障排除.....	220
3.5.6 内存剖析.....	220

3.5.7 CPU 剖析.....	220
3.6 API 接口.....	220
3.6.1 librados (C).....	220
3.6.1.1 实例：连接并写入一个对象.....	220
3.6.1.2 异步 IO.....	221
3.6.1.3 API 调用.....	223
3.6.1.3.1 struct rados_pool_stat_t.....	223
3.6.1.3.2 struct rados_cluster_stat_t.....	224
3.6.1.3.3 定义.....	224
3.6.1.3.4 类.....	224
3.6.1.3.5 函数.....	226
3.6.2 libradospp (C++).....	249
4 CEPH 文件系统.....	250
4.1 增加/拆除元数据服务器.....	250
4.2 MDS 配置参考.....	251
4.3 Journaler.....	258
4.4 ceph-mds 在线手册.....	259
4.4.1 提纲.....	259
4.4.2 描述.....	259
4.4.3 选项.....	260
4.4.4 使用范围.....	260
4.4.5 参考.....	260
4.5 用内核驱动挂载 ceph 文件系统.....	260
4.6 用户空间挂载 ceph 文件系统.....	261
4.7 从 fstab 挂载.....	262
4.7.1 内核驱动.....	262
4.7.2 FUSE.....	262
4.8 cephfs 在线手册.....	262
4.8.1 提纲.....	262
4.8.2 描述.....	263
4.8.3 选项.....	263
4.8.3.1 查看选项：.....	263
4.8.3.2 设置选项.....	263
4.8.4 使用限制.....	264
4.8.5 使用范围.....	264
4.8.6 参考.....	264
4.9 ceph-fuse 在线手册.....	264
4.9.1 提纲.....	264
4.9.2 描述.....	264
4.9.3 选项.....	265
4.9.4 使用范围.....	265
4.9.5 参考.....	265
4.10 mount.ceph 在线手册.....	265
4.10.1 提纲.....	265
4.10.2 描述.....	265
4.10.3 选项.....	266
4.10.4 实例.....	268
4.10.5 使用范围.....	268
4.10.6 参考.....	268
4.11 让 hadoop 使用 cephfs.....	268
4.11.1 依赖关系.....	268
4.11.2 安装.....	269
4.11.2.1 CephFS Java Packages.....	269
4.11.3 hadoop 配置.....	269
4.11.3.1 对每文件定制复制的支持.....	269

4.12 libcephfs (javadoc).....	271
4.13 挂载故障排除.....	271
4.13.1 mount 5 Error.....	271
4.13.2 mount 12 Error.....	271
5 ceph 块设备.....	272
5.1 块设备命令.....	273
5.1.1 创建块设备映像.....	273
5.1.2 罗列块设备映像.....	273
5.1.3 检索映像信息.....	273
5.1.4 更改块设备映像尺寸.....	274
5.1.5 删除块设备映像.....	274
5.2 内核模块操作.....	274
5.2.1 加载 ceph RBD 模块.....	275
5.2.2 获取映像列表.....	275
5.2.3 映射块设备.....	275
5.2.4 查看已映射块设备.....	275
5.2.5 取消块设备映射.....	275
5.3 快照.....	276
5.3.1 cephx 注意事项.....	276
5.3.2 快照基础.....	276
5.3.2.1 创建快照.....	277
5.3.2.2 罗列快照.....	277
5.3.2.3 回滚快照.....	277
5.3.2.4 删除快照.....	277
5.3.2.5 清除快照.....	278
5.3.3 分层.....	278
5.3.3.1 分层入门.....	279
5.3.3.2 保护快照.....	280
5.3.3.3 克隆快照.....	280
5.3.3.4 取消快照保护.....	280
5.3.3.5 罗列一快照的子孙.....	281
5.3.3.6 拍平克隆品映像.....	281
5.4 QEMU 和块设备.....	281
5.4.1 安装 QEMU (12.04 及后续版本)	282
5.4.2 安装 QEMU (11.10 及更早版本)	282
5.4.3 用 QEMU 创建映像.....	282
5.4.4 用 QEMU 更改映像尺寸.....	282
5.4.5 用 QEMU 检索映像信息.....	283
5.4.6 通过 RBD 运行 QEMU.....	283
5.4.7 启用放弃或修剪功能.....	284
5.4.8 QEMU 缓存选项.....	284
5.5 Using libvirt with Ceph RBD.....	285
5.5.1 先决条件.....	285
5.5.2 在 Ubuntu 12.04 上安装 libvirt.....	285
5.5.3 在 Ubuntu 较老版本上安装 libvirt.....	286
5.5.4 ceph 用于虚拟机.....	286
5.5.4.1 配置 ceph.....	286
5.5.4.2 准备虚拟机管理器.....	287
5.5.4.3 新建虚拟机.....	287
5.5.4.4 配置 VM.....	288
5.5.4.5 总结.....	290
5.6 librbd 缓存配置.....	290
5.7 块设备和 OpenStack.....	292
5.7.1 创建存储池.....	293
5.7.2 配置 OpenStack 的 ceph 客户端.....	293

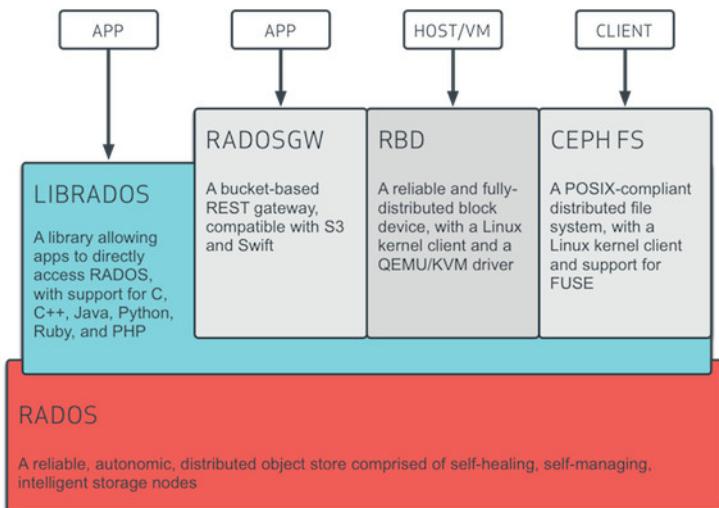
5.7.2.1 安装 ceph 客户端软件包.....	293
5.7.2.2 配置 ceph 客户端认证.....	294
5.7.3 让 OpenStack 使用 ceph.....	294
5.7.3.1 配置 Glance.....	294
5.7.3.2 配置 Cinder/nova-volume.....	295
5.7.4 重启 OpenStack.....	295
5.7.5 从块设备引导.....	296
5.8 块设备和 CloudStack.....	296
5.8.1 创建存储池.....	297
5.8.2 添加主存储.....	297
5.8.3 创建存储服务.....	298
5.8.4 限制.....	298
5.9 rbd 在线手册.....	299
5.9.1 提纲.....	299
5.9.2 描述.....	299
5.9.3 选项.....	299
5.9.4 参数.....	299
5.9.5 命令.....	301
5.9.6 映像名.....	303
5.9.7 条带化.....	304
5.9.8 实例.....	304
5.9.9 使用范围.....	306
5.9.10 参考.....	306
5.10 rbd-fuse 在线手册.....	306
5.10.1 提纲.....	306
5.10.2 描述.....	306
5.10.3 选项.....	306
5.10.4 使用范围.....	307
5.10.5 参考.....	307
5.11 ceph-rbdnamer 在线手册.....	307
5.11.1 提纲.....	307
5.11.2 描述.....	307
5.11.3 使用范围.....	307
5.11.4 参考.....	307
5.12 Librbd (Python).....	307
5.12.1 实例：创建并写入映像.....	307
5.12.2 API 参考.....	308
6 ceph 对象网关.....	314
7 API 文档.....	315
8 体系结构.....	315
8.1 ceph 存储集群.....	316
8.1.1 数据的存储.....	316
8.1.2 伸缩性和高可用性.....	317
8.1.2.1 CRUSH 简介.....	317
8.1.2.2 集群运行图.....	317
8.1.2.3 高可用监视器.....	318
8.1.2.4 高可用认证.....	318
8.1.2.5 智能程序支撑超大规模.....	319
8.1.3 动态集群管理.....	320
8.1.3.1 关于存储池.....	320
8.1.3.2 PG 映射到 OSD.....	321
8.1.3.3 计算 PG ID.....	322
8.1.3.4 互联和集合.....	322
8.1.3.5 重均衡.....	323
8.1.3.6 数据一致性.....	324

8.1.4 扩展 ceph.....	324
8.1.5 总结.....	325
8.2 ceph 协议.....	325
8.2.1 原生协议和 librados.....	325
8.2.2 对象关注/通知.....	326
8.2.3 数据裁剪.....	326
8.3 ceph 客户端.....	329
8.3.1 ceph 对象存储.....	330
8.3.2 ceph 块设备.....	330
8.3.3 ceph 文件系统.....	331

Welcome to Ceph

Ceph uniquely delivers **object, block, and file storage in one unified system**. Ceph is highly reliable, easy to manage, and free. The power of Ceph can transform your company's IT infrastructure and your ability to manage vast amounts of data. Ceph delivers extraordinary scalability—thousands of clients accessing petabytes to exabytes of data. Ceph leverages commodity hardware and intelligent daemons to accommodate large numbers of storage hosts, which communicate with each other to replicate data, and redistribute data dynamically. Ceph's cluster of monitors oversees the hosts in the Ceph storage cluster to ensure that the storage hosts are running smoothly.

ceph 独一无二地在一个统一的系统中同时提供了对象、块、和文件存储功能。它可靠性高、管理简单，并且是开源软件。ceph 的强大可以改变您公司的 IT 基础架构和海量数据管理能力。ceph 可提供极大的伸缩性——可供千用户访问 PB 乃至 EB 级的数据，它用普通硬件和智能守护进程来构成大量存储主机，靠相互通讯来复制数据、并动态地重分布数据。ceph 监视器集群监视着存储集群内的所有主机，确保它们运行正常。



关于 ceph

One Object Storage: The Ceph Object Store, called RADOS, is the object storage component for CephFS filesystems, Ceph RADOS Gateways, and Ceph Block Devices.

一个对象存储：ceph 的对象存储名为 RADOS，是 CephFS 文件系统、RADOS 网关、和 Ceph 块设备的存储部件。

Many Storage Interfaces: You can use CephFS, Ceph RADOS Gateway, or Ceph Block Devices in your deployment. You may also use all three interfaces with the same Ceph Object Store cluster! There's no reason to build three different storage clusters for three different types of storage interface!

多个存储接口：在你的部署中，可以使用 CephFS、Ceph RADOS 网关、或者 ceph 块设备，你也可以同时使用这三个接口，在同一个 ceph 对象存储集群中！无需为三种不同接口分别建设存储集群！

Use Commodity Hardware!: You can deploy Ceph with commodity hardware. You don't need to purchase proprietary storage or networking hardware commonly used in systems.

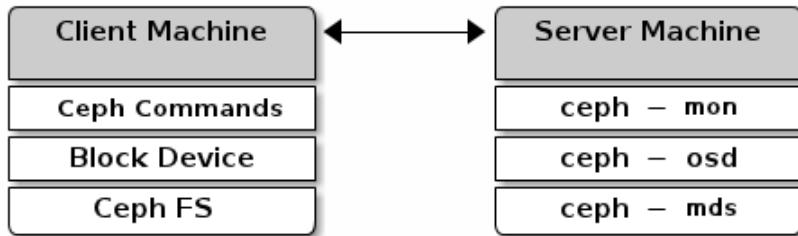
使用普通硬件！：你可以用普通硬件部署 ceph，而不需要购买昂贵的专用存储或网络硬件。

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`  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 [ceph 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格域名 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.

用服务器的全资格域名替换`{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-available 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": "QFAMEDSJJP5DEKJ00DDXY",  
      "secret_key": "iaSFLDVvDdQt61kNzHyW4fPLZugBAI1g17L00+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": "QFAMEDSJP5DEKJ00DDXY",
      "secret_key": "iaSFLDVvDdQt61kNzHyW4fPLZugBAI1g17L00+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": "QFAMEDSJP5DEKJ00DDXY",
          "secret_key": "iaSFLDVvDdQt61kNzHyW4fPLZugBAI1g17L00+87"}],
    "swift_keys": [
        { "user": "johndoe:swift",
          "secret_key": "E9T2rUZNu2gxUjcwUB08n\\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。	<ul style="list-style-type: none">• 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 by unsubscribing . A simple email is all it takes! If you would like to view the archives, go to Gmane . 您可以 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。	<ul style="list-style-type: none">• 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。	<ul style="list-style-type: none">• 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 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

Process	Criteria	Minimum Recommended
	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
	Processor	1x hex-core Opteron CPU
	RAM	16 GB
Dell PE R515	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 significantly 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	Oneiric 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	Oneiric 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 (`e16`), 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(`e16`)、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.e16.noarch.rpm'
```

For example, for CentOS 6 or other RHEL6 derivatives (`e16`):

例如，拿 CentOS 6 或其他 RHEL6 衍生物(`e16`)来说：

```
su -c 'rpm -Uvh http://ceph.com/rpm-bobtail/e16/x86_64/ceph-release-1-0.e16.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 exercise 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 perform 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 upgrading 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。

2.5.1 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](#)。

2.5.2 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。

2.5.3 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`
- `cdb`
- `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 cdb 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 cdb 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-perf-tools4).

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-perf-tools4）。

2.6.1 Ubuntu

- `uuid-dev`
- `libkeyutils-dev`
- `libgoogle-perf-tools-dev`
- `libatomic-ops-dev`
- `libaio-dev`
- `libgdata-common`
- `libgdata13`
- `libsnapy-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-perf-tools-dev libatomic-ops-dev libaio-dev  
libgdata-common libgdata13 libsnapy-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 **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 -j 4` 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 oprofile 剖析器。

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 [Ceph Logging and Debugging](#)。

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，必须克隆 ceph 仓库，你可以用最新代码构建安装包，Debian/Ubuntu 下用 `dpkg-buildpackage`、用 `rpmbuild` 构建 RPM 安装包。

Tip: When building on a multi-core CPU, use the `-j` and the number of cores * 2. For example, use `-j 4` 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

RADOS Object Store

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、RESTful 网关或 ceph 块设备的对象存储客户端时，ceph 从对象存储读取和写入数据。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；监视器集群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，最简系统至少需要两个 OSD 才能做到数据复制。



配置和部署

CONFIG AND DEPLOY

Once you have installed Ceph packages, you must configure. There are 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。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。

运维

Operations

Once you have a deployed Ceph cluster, you may begin operating your cluster.

部署后就可以开始操作 ceph 集群了。

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 部署都使用了 ceph 块设备、网关和/或 CephFS 文件系统。也可以开发程序直接和 ceph 对象存储对接。

3.1 配置

Configuration

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 作为集群时可以包含数千个对象存储设备（OSD）。最简系统至少需要二个 OSD 做数据复制。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) 必备
- ceph-mon (mandatory) 必备

- ceph-mds (mandatory for cephfs only) cephfs 必备

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).

各进程、守护进程或工具都会读取配置文件。一个进程可能需要不止一个守护进程例程的信息（如多个上下文）；一个守护进程或工具只有关于单个守护进程例程的信息（单上下文）。

Note: Ceph can run on a single host for evaluation purposes.

注意：试用时 Ceph 可运行在单机上。

3.1.1 硬盘和文件系统

3.1.1.1 准备硬盘

HARD DISK PREP

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 禁用硬盘的写缓冲功能。

```
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。

3.1.1.2 文件系统

FILE SYSTEMS

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；推荐用 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.

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。

You should always add the following line to the [osd] section of your `ceph.conf` file for ext4 filesystems; you can optionally use it for btrfs and XFS.:

使用 ext4 文件系统时，应该一直把下面的配置放于 [osd] 段下；用 btrfs 和 xfs 时可以选填。

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

3.1.1.3 文件系统背景知识

FS BACKGROUND INFO

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（很多在 ceph/src/common/config_opts.h 里面设置），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：

Authentication settings

Cluster membership

Host names

Host addresses

Paths to keyrings

Paths to journals

Paths to data

Other runtime options

- 认证配置
- 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：

1. \$CEPH_CONF (i.e., the path following the \$CEPH_CONF environment variable)
2. -c path/path (i.e., the -c command line argument)
3. /etc/ceph/ceph.conf
4. ~/.ceph/config
5. ./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(#echo)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 配置段落

Config Sections

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，或者某一类型的所有守护进程。要配置一系列守护进程，这些配置必须位于能收到配置的段落之下，比如：

[global]

Description: Settings under [global] affect all daemons in a Ceph cluster.

[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.

[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.

[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.

[mds] 下的配置影响集群里的所有 ceph-mds 进程。

Example: host = myserver01

[client]

Description: Settings under [client] affect all clients (e.g., mounted CephFS filesystems, mounted block devices, etc.)

[client] 下的配置影响所有客户端（如挂载的 CephFS 文件系统、挂载的块设备等等）。

Example: log file = /var/log/ceph/radosgw.log

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：

1. 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。
2. 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
#v 0.54 and earlier
auth supported = cephx

#v 0.55 and after
auth cluster required = cephx
auth service required = cephx
auth client required = 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.
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.
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.
展开为守护进程标识；对 osd.0 来说是 0，mds.a 是 a。

Example: /var/lib/ceph/\$type/\$cluster-\$id

\$host

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

\$name

Description: Expands to \$type.\$id.
展开为\$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 an 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：`host` 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，不能用于 `chef` 或 `ceph-deploy`。

3.1.2.5 网络

See the [Network Configuration Reference](#) for a detailed discussion about configuring a network for use with Ceph.

关于 ceph 网络配置的讨论见[网络配置参考](#)。

3.1.2.6 认证

Authentication

New in version Bobtail: 0.56

bobtail v0.56 新增。

For Bobtail (v 0.56) and beyond, you should expressly enable or disable authentication in the `[global]` section of your Ceph configuration file.

对于 v0.56 及后来版本，要在配置文件的`[global]` 中明确启用或禁用认证。

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

Additionally, you should enable message signing. See [Cephx Config Reference](#) and [Cephx Authentication](#) for details.

另外，你应该启用消息签名，详情见[认证配置（cephx 配置）](#) 和 [cephx 认证](#)。

Important: When upgrading, we recommend expressly disabling authentication first, then perform the upgrade. Once the upgrade is complete, re-enable authentication.

重要：我们建议，升级时先明确地关闭认证，再进行升级。等升级完成后重重新启用认证。

3.1.2.7 监视器集群

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（3 个）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.

注意：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:

你必须手动创建对应目录，前述的元变量必须先全部展开，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 选项更改此路径，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}
```

3.1.2.8 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
```

3.1.2.9 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.

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。

Note: See [Debugging and Logging](#) for details on log rotation.

注意：关于日志滚动见[日志记录和调试](#)。

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:

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
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

3.1.2.10 ceph.conf 实例

EXAMPLE CEPH.CONF

```
[global]
# For version 0.54 and earlier, you may enable
# authentication with the following setting.
# Specifying `cephx` enables authentication;
# and specifying `none` disables authentication.

#auth supported = cephx

# 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
    # uncomment the following line if you are mounting with ext4
    # 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 for you. Remove the comment `#` character for
    # the following settings and replace the values in parenthesis
    # 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"

[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
    #devs = {path-to-device}

[osd.1]
    host = myserver02
    #devs = {path-to-device}

[osd.2]
    host = myserver03
    #devs = {path-to-device}

[mds.a]
    host = myserver01
    #devs = {path-to-device}

```

3.1.2.11 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。

3.1.2.12 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:

各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
```

3.1.2.13 运行多个集群

Running Multiple Clusters

With Ceph, you can run multiple clusters on the same hardware. Running multiple clusters provides a higher level of isolation compared to using different pools on the same cluster with different CRUSH rulesets. A separate cluster will have separate monitor, OSD and metadata server processes. When running Ceph with default settings, the default cluster name is `ceph`, which means you would save your Ceph configuration file with the file name `ceph.conf` in the `/etc/ceph` default directory.

用 ceph 可以实现在同一套硬件上运行多个集群，运行多个集群和在同一个集群上使用 CRUSH 规则控制多个存储池相比提供了更高水平的隔离。独立的集群需要独立的监视器、OSD 和元数据服务器进程。默认配置下集群名字是 ceph，这意味着你得把配置文件保存为 /etc/ceph 下的 ceph.conf。

When you run multiple clusters, you must name your cluster and save the Ceph configuration file with the name of the cluster. For example, a cluster named `openstack` will have a Ceph configuration file with the file name `openstack.conf` in the `/etc/ceph` default directory.

运行多个集群时，你必须为集群命名并用这个名字保存配置文件。例如，名为 `openstack` 的集群其配置文件将是 /etc/ceph 下的 `openstack.conf`。

Important: Cluster names must consist of letters a-z and digits 0-9 only.

重要：集群名字里只能包含字母 a-z 和数字 0-9。

Separate clusters imply separate data disks and journals, which are not shared between clusters. Referring to [Metavariables](#), the `$cluster` metavariable evaluates to the cluster name (i.e., `openstack` in the foregoing example). Various settings use the `$cluster` metavariable, including:

独立的集群意味着独立数据盘和日志，它们不能在集群间共享。根据 [元变量](#)，`$cluster` 元变量对应集群名字（前例为 `openstack`）。多处设置都用到 `$cluster` 元变量，包括：

- `keyring`
- `admin socket`
- `log file`
- `pid file`
- `mon data`
- `mon cluster log file`
- `osd data`
- `osd journal`
- `mds data`

- `rgw data`

See [General Settings](#), [OSD Settings](#), [Monitor Settings](#), [MDS Settings](#), [RGW Settings](#) and [Log Settings](#) for relevant path defaults that use the `$cluster` metavariable.

和`$cluster`元变量相关的默认路径见[常规配置](#)、[OSD 选项](#)、[监视器选项](#)、[错误：引用源未找到](#)、[<6.3>](#)、[错误：引用源未找到](#)。

When deploying the Ceph configuration file, ensure that you use the cluster name in your command line syntax. For example:

分发配置文件时，确保在命令行下用的是集群名，例如：

```
ssh myserver01 sudo tee /etc/ceph/openstack.conf < /etc/ceph/openstack.conf
```

When creating default directories or files, you should also use the cluster name at the appropriate places in the path. For example:

创建默认目录和文件时，也要代入集群名。例如：

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

Important: When running monitors on the same host, you should use different ports. By default, monitors use port 6789. If you already have monitors using port 6789, use a different port for your other cluster(s).

重要：在一台主机上运行多个监视器时，你得指定不同端口。监视器默认使用 6789 端口，如果它已经被占，其它集群得另外指定端口。

To invoke a cluster other than the default `ceph` cluster, use the `--cluster=clustername` option with the `ceph` command. For example:

要调动一个非默认 `ceph` 的集群，要给 `ceph` 命令加`--cluster=clustername` 选项，例如：

```
ceph --cluster=openstack health
```

3.1.3 网络配置

Network Configuration Reference

Network configuration is critical for building a high performance Ceph cluster. The Ceph cluster does not perform request routing or dispatching on behalf of the client. Instead, Ceph clients (i.e., block device, CephFS, REST gateway) make requests directly to OSDs. Ceph OSDs perform data replication on behalf of clients, which means replication and other factors impose additional loads on Ceph cluster networks.

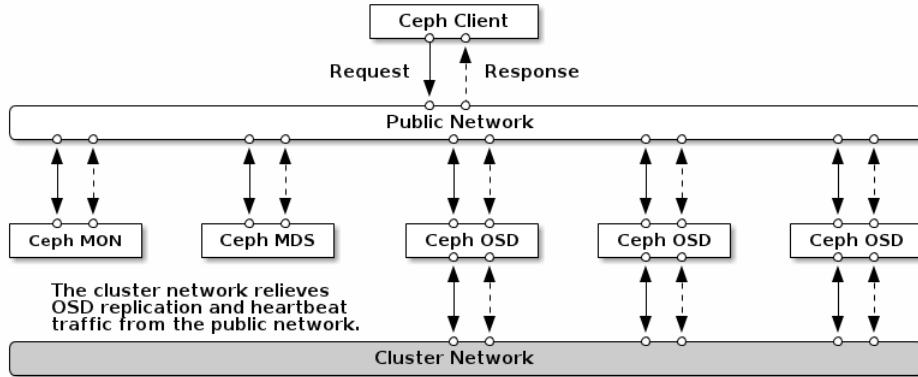
网络配置对构建高性能集群来说相当重要。`ceph` 集群不会代表客户端执行请求路由或调度，相反，`ceph` 客户端（如块设备、CephFS、REST 网关）直接向 OSD 请求，然后 OSD 为客户端执行数据复制，也就是说复制和其它因素会额外增加集群网的负载。

Our 5-minute Quick Start provides a trivial [Ceph configuration file](#) that sets monitor IP addresses and daemon host names only. The quick start configuration assumes a single “public” network. Ceph functions just fine with a public network only, but you may see significant performance improvement with a second “cluster” network in a large cluster.

我们的 5 分钟快速入门提供了一个简陋的 [ceph 配置文件](#)，其中只设置了监视器 IP 地址和守护进程所在的主机名，因为快速入门假设只有一个公共网。只用一个网可以运行 `ceph`，但是在大型集群里用单独的集群网可显著地提升性能。

We recommend running a Ceph cluster with two networks: a public (front-side) network and a cluster (back-side) network. To support two networks, your hosts need to have more than one NIC. See [Hardware Recommendations - Networks](#) for additional details.

我们推荐用两个网络运行 `ceph`: 一个公共网（前端）和一个集群网（后端）。为此，主机得配备多个网卡，见[网络](#)。



There are several reasons to consider operating two separate networks:

运营两个独立网络的考量主要有：

- Performance:** 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. Recovery and rebalancing can also introduce significant latency on the public network. See [How Ceph Scales](#) for additional details on how Ceph replicates data. See [Monitor / OSD Interaction](#) for details on heartbeat traffic.

性能：OSD 为客户端处理数据复制，复制多份时 OSD 间的网络负载势必会影响到客户端和 ceph 集群的通讯，包括延时增加、产生性能问题；恢复和重均衡也会显著增加公共网延时。关于 ceph 如何复制参见[错误：引用源未找到](#)；关于心跳流量参见[心跳配置（监视器/OSD 交互的配置）](#)。

- Security:** 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. Also, consider using [Message Signatures](#) to defeat spoofing attacks.

安全：大多数人都是良民，很少的一撮人喜欢折腾拒绝服务攻击（DoS）。当 OSD 间的流量瓦解时，归置组再也不能达到 `active+clean` 状态，这样用户就不能读写数据了。挫败此类攻击的一种好方法是维护一个完全独立的集群网，使之不能直连互联网；另外，请考虑用[签名](#)防止欺骗攻击。

3.1.3.1 防火墙 iptables

IP Tables

By default, daemons [bind](#) to ports within the `6800:7100` range. You may configure this range at your discretion. Before configuring your IP tables, check the default [iptables](#) configuration.

守护进程默认会**绑定**到 `6800:7100` 间的端口，你可以更改此范围。配置防火墙前先检查下 [iptables](#) 配置：

```
sudo iptables -L
```

Some Linux distributions include rules that reject all inbound requests except SSH from all network interfaces. For example:

一些 Linux 发行版的规则拒绝除 SSH 之外的所有入栈连接，例如：

```
REJECT all -- anywhere anywhere reject-with icmp-host-prohibited
```

You will need to delete these rules on both your public and cluster networks initially, and replace them with appropriate rules when you are ready to harden the ports on your cluster hosts.

你得先删掉公共网和集群网对应的这些规则，然后再增加安全保护规则。

3.1.3.1.1 监视器防火墙

Monitor IP Tables

Monitors listen on port 6789 by default. Additionally, monitors always operate on the public network. When you add the rule using the example below, make sure you replace `{iface}` with the public network interface (e.g., `eth0`, `eth1`, etc.), `{ip-address}` with the IP address of the public network and `{netmask}` with the netmask for the public network.

监视器默认监听 6789 端口，而且监视器总是运行在公共网。按下例增加规则时，要把{iface}替换为公共网接口（如 eth0、eth1 等等）、{ip-address}替换为公共网 IP、{netmask}替换为公共网掩码。

```
sudo iptables -A INPUT -i {iface} -p tcp -s {ip-address}/{netmask} --dport 6789 -j ACCEPT
```

3.1.3.1.2 MDS 防火墙

MDS IP Tables

Metadata servers listen on the first available port on the public network beginning at port 6800. Ensure that you open one port beginning at port 6800 for each metadata server that runs on the host. When you add the rule using the example below, make sure you replace {iface} with the public network interface (e.g., eth0, eth1, etc.), {ip-address} with the IP address of the public network and {netmask} with the netmask of the public network.

元数据服务器会监听公共网 6800 以上的第一个可用端口，确保打开元数据服务器上 6800 以上的端口。按下例增加规则时，要把{iface}替换为公共网接口（如 eth0、eth1 等等）、{ip-address}替换为公共网 IP、{netmask}替换为公共网掩码。

For example:

例如：

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

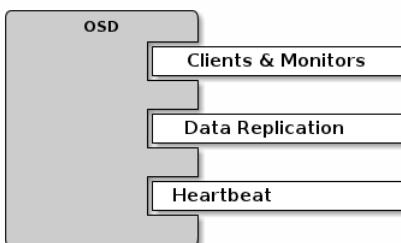
3.1.3.1.3 OSD 防火墙

OSD IP Tables

By default, OSDs bind to the first available ports on a host beginning at port 6800. Ensure that you open at least three ports beginning at port 6800 for each OSD that runs on the host. Each OSD on a host may use up to three ports:

OSD 默认绑定 6800 以上的第一个可用端口，要确保各 OSD 至少打开了 3 个 6800 以上的端口。一主机上的各个 OSD 最多会用到 3 个端口：

1. One for talking to clients and monitors.
一个用于和客户端、监视器通讯；
2. One for sending data to other OSDs.
一个用于发送数据到其他 OSD；
3. One for heartbeating.
一个用于心跳；



Ports are host-specific, so you don't need to open any more ports than the number of ports needed by Ceph daemons running on that host. 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.

每台主机的端口不尽相同，所以你不需要开放多于 ceph 所需的端口。但是应该多开放一点，因为有时候某个守护进程失败了，但是重启前并未全部释放，以致于重启后的进程监听了新端口。

If you set up separate public and cluster networks, you must add rules for both the public network and the cluster network, because clients will connect using the public network and other OSDs will connect using the cluster network. When you add the rule using the example below, make sure you replace {iface} with the network interface (e.g., eth0, eth1, etc.), {ip-address} with the IP address and {netmask} with the netmask of the public or cluster network. For example:

如果你要分开公共网和集群网，必须分别为之设置防火墙，因为客户端会通过公共网连接，而其他 OSD 会通过集群网连接。按下例增加规则时，要把{iface}替换为公共网接口（如 eth0、eth1 等等）、{ip-address}替换为公共网 IP、{netmask}替换为公共网掩码。例如：

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

Tip: If you run metadata servers on the same host as the OSDs, you can consolidate the public network configuration step. Ensure that you open the number of ports required for each daemon per host.

提示：如果你的元数据服务器和 OSD 在同一台主机上，可以合并公共网配置，要确保开放了足够用的端口。

3.1.3.2 ceph 网络

Ceph Networks

To configure Ceph networks, you must add a network configuration to the [global] section of the configuration file. Our 5-minute Quick Start provides a trivial [Ceph configuration file](#) that assumes one public network with client and server on the same network and subnet. Ceph functions just fine with a public network only. However, Ceph allows you to establish much more specific criteria, including multiple IP network and subnet masks for your public network. You can also establish a separate cluster network to handle OSD heartbeat, object replication and recovery traffic. Don't confuse the IP addresses you set in your configuration with the public-facing IP addresses network clients may use to access your service. Typical internal IP networks are often 192.168.0.0 or 10.0.0.0.

ceph 的网络配置要放到[global]段下。前述的 5 分钟快速入门提供了一个简陋的 [ceph 配置文件](#)，它假设服务器和客户端都位于同一个网段，ceph 可以很好地适应这种情形。然而 ceph 允许配置更精细的公共网，包括多 IP 和多掩码；也能用单独的网络处理 OSD 心跳、对象复制、和恢复流量。不要混淆你配置的 IP 地址和客户端用来访问存储服务的公共网地址。典型的内网常常是 192.168.0.0 和 10.0.0.0。

Tip: If you specify more than one IP address and subnet mask for either the public or the cluster network, the subnets within the network must be capable of routing to each other. Additionally, make sure you include each IP address/subnet in your IP tables and open ports for them as necessary.

提示：如果你给公共网或集群网配置了多个 IP 地址及子网掩码，这些子网必须能互通。另外要确保为各 IP/子网开放了必要的端口。

Note: Ceph uses [CIDR](#) notation for subnets (e.g., 10.0.0.0/24).

注意：ceph 用 CIDR 法表示子网，如 10.0.0.0/24。

When you've configured your networks, you may restart your cluster or restart each daemon. Ceph daemons bind dynamically, so you do not have to restart the entire cluster at once if you change your network configuration.

配置好几个网络后，可以重启集群或挨个重启守护进程。ceph 守护进程动态地绑定端口，所以更改网络配置后无需一次性重启整个集群。

3.1.3.2.1 公公网

Public Network

To configure a public network, add the following option to the [global] section of your Ceph configuration file.

要配置一个公共网，把下列选项加到配置文件的[global]段下。

```
[global]
...
public network = {public-network/netmask}
```

3.1.3.2.2 集群网

Cluster Network

If you declare a cluster network, OSDs will route heartbeat, object replication and recovery traffic over the cluster network. This may improve performance compared to using a single network. To configure a cluster

network, add the following option to the [global] section of your Ceph configuration file.

如果你声明了集群网，OSD 将把心跳、对象复制和恢复流量路由到集群网，与单个网络相比这会提升性能。要配置集群网，把下列选项加进配置文件的[global]段。

```
[global]
...
cluster network = {cluster-network/netmask}
```

We prefer that the cluster network is NOT reachable from the public network or the Internet for added security.

为安全起见，从公共网或互联网到集群网应该是不可达的。

3.1.3.3 ceph 守护进程

Ceph Daemons

Ceph has one network configuration requirement that applies to all daemons: the Ceph configuration file **MUST** specify the `host` for each daemon. Ceph also requires that a Ceph configuration file specify the monitor IP address and its port.

有一个网络配置是所有守护进程都要配的：各个守护进程都必须指定 `host`，ceph 也要求指定监视器 IP 地址及端口。

Important: Some deployment tools (e.g., `ceph-deploy`, Chef) may create a configuration file for you. **DO NOT** set these values if the deployment tool does it for you.

重要：一些部署工具（如 `ceph-deploy`、`chef`）会给你创建配置文件，如果它能胜任那就别设置这些值。

Tip: The `host` setting is the short name of the host (i.e., not an fqdn). It is **NOT** an IP address either. Enter `hostname -s` on the command line to retrieve the name of the host.

提示：`host` 选项是主机的短名，不是全资域名 FQDN，也不是 IP 地址。在命令行下输入 `hostname -s` 获取主机名。

```
[mon.a]
host = {hostname}
mon addr = {ip-address}:6789

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

You do not have to set the host IP address for a daemon. If you have a static IP configuration and both public and cluster networks running, the Ceph configuration file may specify the IP address of the host for each daemon. To set a static IP address for a daemon, the following option(s) should appear in the daemon instance sections of your `ceph.conf` file.

并非一定要给守护进程设置 IP 地址。如果你有一个静态配置，且分离了公共网和集群网，ceph 允许你在配置文件里指定主机的 IP 地址。要给守护进程设置静态 IP，可把下列选项加到 `ceph.conf`。

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

One NIC OSD in a Two Network Cluster

单网卡 OSD、双网络集群

Generally, we do not recommend deploying an OSD host with a single NIC in a cluster with two networks. However, you may accomplish this by forcing the OSD host to operate on the public network by adding a `public addr` entry to the [osd.n] section of the Ceph configuration file, where `n` refers to the number of the OSD with one NIC. Additionally, the public network and cluster network must be able to route traffic to each other, which we don't recommend for security reasons.

一般来说，我们不建议用单网卡 OSD 主机部署两个网络。然而这事可以实现，把 `public addr` 选项配在 [osd.n] 段下即可强制 OSD 主机运行在公共网，其中 `n` 是其 OSD 号。另外，公共网和集群网必须互通，考虑到安全因素我们不建议这样做。

3.1.3.4 网络配置选项

Network Config Settings

Network configuration settings are not required. Ceph assumes a public network with all hosts operating on it unless you specifically configure a cluster network.

网络配置选项不是必需的，ceph 假设所有主机都运行于公共网，除非你特意配置了一个集群网。

3.1.3.4.1 公共网

Public Network

The public network configuration allows you specifically define IP addresses and subnets for the public network. You may specifically assign static IP addresses or override `public network` settings using the `public addr` setting for a specific daemon.

公共网配置允许你明确地为公共网定义 IP 地址和子网。你可以明确分配静态 IP 或用 `public addr` 覆盖 `public network` 选项。

`public network`

Description: The IP address and netmask of the public (front-side) network (e.g., `192.168.0.0/24`). Set in `[global]`. You may specify comma-delimited subnets.

公共网（前端）的 IP 地址和掩码（如 `192.168.0.0/24`）。用于 `[global]` 下。多个子网用逗号分隔。

Type: `{ip-address}/{netmask} [, {ip-address}/{netmask}]`

Required: No

Default: N/A

`public addr`

Description: The IP address for the public (front-side) network. Set for each daemon.

用于公共网（前端）的 IP 地址。适用于各守护进程。

Type: IP Address

Required: No

Default: N/A

3.1.3.4.2 集群网

Cluster Network

The cluster network configuration allows you to declare a cluster network, and specifically define IP addresses and subnets for the cluster network. You may specifically assign static IP addresses or override `cluster network` settings using the `cluster addr` setting for specific OSD daemons.

集群网配置允许你声明一个集群网，并明确地定义 IP 地址和子网。你可以配置静态 IP 或为某 OSD 守护进程配置 `cluster addr` 以覆盖 `cluster network` 选项。

`cluster network`

Description: The IP address and netmask of the cluster (back-side) network (e.g., `10.0.0.0/24`). Set in `[global]`. You may specify comma-delimited subnets.

集群网（后端）的 IP 地址及掩码（如 `10.0.0.0/24`）。置于 `[global]` 下。多个子网用逗号分隔。

Type: `{ip-address}/{netmask} [, {ip-address}/{netmask}]`

Required: No

Default: N/A

`cluster addr`

Description: The IP address for the cluster (back-side) network. Set for each daemon.

集群网（后端）IP 地址。置于各守护进程下。

Type: Address

Required: No

Default: N/A

3.1.3.4.3 端口绑定

Bind

Bind settings set the default port ranges Ceph OSD and MDS daemons use. The default range is 6800:7100. Ensure that your [IP Tables](#) configuration allows you to use the configured port range.

绑定选项设置 OSD 和 MDS 使用的默认端口范围， 默认范围是 6800:7100。确保防火墙开放了对应端口范围。

You may also enable Ceph daemons to bind to IPv6 addresses.

你也可以让 ceph 守护进程绑定到 IPv6 地址。

ms bind port min

Description: The minimum port number to which an OSD or MDS daemon will bind.
OSD 或 MDS 可绑定的最小端口号。

Type: 32-bit Integer

Default: 6800

Required: No

ms bind port max

Description: The maximum port number to which an OSD or MDS daemon will bind.
OSD 或 MDS 可绑定的最大端口号。

Type: 32-bit Integer

Default: 7100

Required: No.

ms bind ipv6

Description: Enables Ceph daemons to bind to IPv6 addresses.
允许 ceph 绑定 IPv6 地址。

Type: Boolean

Default: false

Required: No

3.1.3.4.4 主机

Hosts

Ceph expects at least one monitor declared in the Ceph configuration file, with a `mon addr` setting under each declared monitor. Ceph expects a `host` setting under each declared monitor, metadata server and OSD in the Ceph configuration file.

ceph 需要至少一个监视器， 声明的每个监视器下都要加 `mon addr` 选项； 每个监视器、 元数据服务器和 OSD 下都要配 `host` 选项。

mon addr

Description: A list of `{hostname}:{port}` entries that clients can use to connect to a Ceph monitor. If not set, Ceph searches `[mon.*]` sections.

`{hostname}:{port}` 条目列表， 用以让客户端连接 ceph 监视器。如果未设置， ceph 查找 `[mon.*]` 段。

Type: String

Required: No

Default: N/A

host

Description: The hostname. Use this setting for specific daemon instances (e.g., `[osd.0]`).
主机名。此选项用于特定守护进程， 如 `[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).

提示：不要用 `localhost`。在命令行下执行 `hostname -s` 获取主机名（到第一个点，不是全资格域名），并用于配置文件。

Important: You should not specify any value for `host` when using a third party deployment system that retrieves the host name for you.

重要：用第三方部署工具时不要指定 `host` 选项，它会自行获取。

3.1.3.4.5 TCP

Ceph disables TCP buffering by default.

ceph 默认禁用 TCP 缓冲。

`tcp nodelay`

Description: Ceph enables `tcp nodelay` so that each request is sent immediately (no buffering). Disabling [Nagle's algorithm](#) increases network traffic, which can introduce latency. If you experience large numbers of small packets, you may try disabling `tcp nodelay`.

ceph 用 `tcp nodelay` 使系统尽快（不缓冲）发送每个请求。禁用 [Nagle](#) 算法可增加吞吐量，但会引进延时。如果你遇到大量小包，可以禁用 `tcp nodelay` 试试。

Type: Boolean

Required: No

Default: true

`tcp rbuf`

Description: The size of the socket buffer on the receiving end of a network connection. Disable by default.
网络套接字接收缓冲尺寸，默认禁用。

Type: 32-bit Integer

Required: No

Default: 0

`ms tcp read timeout`

Description: If a client or daemon makes a request to another Ceph daemon and does not drop an unused connection, the `tcp read timeout` defines the connection as idle after the specified number of seconds.

如果一客户端或守护进程请求连接到另一个 ceph 守护进程，且没有断开不再使用的连接，在 `tcp read timeout` 指定的秒数之后它将被标记为空闲。

Type: Unsigned 64-bit Integer

Required: No

Default: 900 15 minutes.

3.1.4 认证配置 (cephx 配置)

Cephx Config Reference

To protect against man-in-the-middle attacks, Ceph provides its `cephx` authentication system to authenticate users and daemons. See [Ceph Authentication & Authorization](#) for an introduction to `cephx` authentication. See the [Cephx Guide](#) for details on enabling/disabling, creating users and setting user capabilities.

为防止中间人攻击，ceph 用 `cephx` 认证系统来认证用户和守护进程。关于 `cephx` 认证的介绍见 [Ceph Authentication & Authorization](#)；关于如何启用/禁用、创建用户、设置用户能力见 [cephx 认证](#)。

3.1.4.1 启用/禁用认证

Enable/Disable Authentication

Depending on the version, Ceph either enables or disables authentication by default. Use the following settings to expressly enable or disable Ceph. See [Ceph Authentication](#) for additional details.

在不同版本中，ceph 默认启用或禁用了认证。下列选项可明确启用或禁用认证，详情见 [ceph 认证 \(cephx\)](#)。

Authentication Enablement Defaults

认证启用默认值

Ceph version 0.54 and earlier versions disable authentication by default. If you want to use Ceph authentication, you must specifically enable it for version 0.54 and earlier versions.

Ceph version 0.55 and later version enable authentication by default. If you do not want to use Ceph authentication, you must specifically disable it for versions 0.55 and later versions.

ceph 0.54 及其之前版本默认禁用认证，如果你想用认证，必须明确启用。

ceph 0.55 及其之后版本默认开启认证，如果你不想用认证，必须明确禁用。

Authentication Granularity

认证粒度

Ceph version 0.50 and earlier versions use `auth supported` to enable or disable authentication between the Ceph client and the cluster. Ceph authentication in earlier versions only authenticates users sending message traffic between the client and the cluster, so it does not have fine-grained control.

ceph 0.50 及之前版本用 `auth supported` 来打开或关闭 ceph 客户端和集群间的认证，而且早期版本只认证了客户端和集群间的部分，所以它不能精细控制。

Ceph version 0.51 and later versions use fine-grained control, which allows you to require authentication of the client by the cluster (`auth service required`), authentication of the cluster by the client (`auth client required`), and authentication of a daemon within the cluster by another daemon within the cluster (`auth cluster required`).

ceph 0.51 及其后续版本使用更精细的控制，它允许集群要求客户端提供认证信息 (`auth service required`)、客户端认证集群 (`auth client required`)、集群内的守护进程相互认证 (`auth cluster required`)。

`auth supported`

Deprecated since version 0.51.

从 0.51 开始废弃。

Description: Indicates whether to use authentication. If not specified, it defaults to `none`, which means it is disabled.

指示是否使用认证。如果未配置，就是 `none`，意思是禁用了。

Type: String

Required: No

Default: `none`

`auth cluster required`

New in version 0.51.

0.51 新增。

Description: If enabled, the cluster daemons (i.e., `ceph-mon`, `ceph-osd`, and `ceph-mds`) must authenticate with each other. Valid setting is `cephx` or `none`.

如果启用了，集群守护进程（如 `ceph-mon`、`ceph-osd` 和 `ceph-mds`）间必须相互认证。可用选项有 `cephx` 或 `none`。

Type: String

Required: No

Default: Version 0.54 and earlier `none`. Version 0.55 and later `cephx`.

0.54 及之前为 `none`，0.55 及之后为 `cephx`。

`auth service required`

New in version 0.51.

0.51 新增。

Description: If enabled, the cluster daemons require Ceph clients to authenticate with the cluster in order to access Ceph services. Valid setting is `cephx` or `none`.

如果启用，客户端要访问 ceph 服务的话，集群守护进程会要求它和集群认证。可用选项为 cephx 或 none。

Type: String
Required: No
Default: Version 0.54 and earlier none. Version 0.55 and later cephx.
0.54 及之前为 none，0.55 及之后为 cephx。

auth client required

New in version 0.51.

0.51 新增。

Description: If enabled, the client requires the Ceph cluster to authenticate with the client. Valid setting is cephx or none.

如果启用，客户端会要求 ceph 集群和它认证。可用选项为 cephx 或 none。

Type: String
Required: No
Default: Version 0.54 and earlier none. Version 0.55 and later cephx.
0.54 及之前为 none，0.55 及之后为 cephx。

3.1.4.2 密钥

Keys

When you run Ceph with authentication enabled, ceph administrative commands and Ceph clients require authentication keys to access the cluster.

如果你的集群启用了认证，ceph 管理命令和客户端得有密钥才能访问集群。

The most common way to provide these keys to the ceph administrative commands and clients is to include a Ceph keyring under the /etc/ceph directory. The filename is usually ceph.keyring (or \$cluster.keyring) or simply keyring. If you include the keyring under the /etc/ceph directory, you don't need to specify a keyring entry in your Ceph configuration file.

给 ceph 管理命令和客户端提供这些密钥的最常见方法就是把密钥环放到 /etc/ceph 下，文件名通常是 ceph.keyring (或 \$cluster.keyring) 或仅是 keyring。如果你的密钥环位于 /etc/ceph 下，就不需要在 ceph 配置文件里指定 keyring 选项了。

We recommend copying the cluster's keyring file to hosts where you'll run administrative commands, because it contains the client.admin key.

我们建议把集群的密钥环拷贝到你执行管理命令的主机，它包含 client.admin 密钥：

```
sudo scp {user}@{ceph-cluster-host}:/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：确保给客户端上的 ceph.keyring 设置合理的权限位（如 chmod 644）。

You may specify the key itself in the Ceph configuration file using the key setting (not recommended), or a path to a keyfile using the keyfile setting.

你可以用 key 选项把密钥写在配置文件里（别这样），或者用 keyfile 选项指定个路径。

keyring

Description: The path to the keyring file.
到密钥环文件的路径。
Type: String
Required: No
Default: /etc/ceph/\$cluster.\$name.keyring, /etc/ceph/\$cluster.keyring,
/etc/ceph/keyring, /etc/ceph/keyring.bin

keyfile

Description: The path to a key file (i.e., a file containing only the key).

到密钥文件的路径，如一个只包含密钥的文件。

Type: String
Required: No
Default: None

key

Description: The key (i.e., the text string of the key itself). Not recommended.
密钥（密钥文本），最好别这样做。

Type: String
Required: No
Default: None

3.1.4.3 签名

Signatures

In Ceph Bobtail and subsequent versions, we prefer that Ceph authenticate all ongoing messages between the entities using the session key set up for that initial authentication. However, Argonaut and earlier Ceph daemons do not know how to perform ongoing message authentication. To maintain backward compatibility (e.g., running both Bobtail and Argonaut daemons in the same cluster), message signing is off by default. If you are running Bobtail or later daemons exclusively, configure Ceph to require signatures.

在 bobtail 及后续版本，ceph 会用开始认证时生成的会话密钥认证所有在线实体。然而 Argonaut 及之前版本不知道如何认证在线消息，为保持向后兼容性（如在同一个集群里运行 bobtail 和 argonaut），消息签名默认是关闭的。如果你只运行 bobtail 和后续版本，可以让 ceph 要求签名。

Like other parts of Ceph authentication, Ceph provides fine-grained control so you can enable/disable signatures for service messages between the client and Ceph, and you can enable/disable signatures for messages between Ceph daemons.

像 ceph 认证的其他部分一样，客户端和集群间的消息签名也能做到细粒度控制；而且能启用或禁用 ceph 守护进程间的签名。

ceph require signatures

Description: If set to true, Ceph requires signatures on all message traffic between the client and the Ceph cluster, and between daemons within the cluster.
若设置为 true，ceph 集群会要求客户端签名所有消息，包括客户端和集群内其他守护进程间的。

Type: Boolean
Required: No
Default: false

cephx cluster require signatures

Description: If set to true, Ceph requires signatures on all message traffic between Ceph daemons within the cluster.
若设置为 true，ceph 要求集群内所有守护进程签名相互之间的消息。

Type: Boolean
Required: No
Default: false

cephx service require signatures

Description: If set to true, Ceph requires signatures on all message traffic between Ceph clients and the Ceph cluster.
若设置为 true，ceph 要求签名所有客户端和集群间的消息。

Type: Boolean
Required: No
Default: false

cephx sign messages

Description: If the Ceph version supports message signing, Ceph will sign all messages so they cannot be spoofed.
如果 ceph 版本支持消息签名，ceph 会签名所有消息以防欺骗。

Type: Boolean
Default: true

3.1.4.4 生存期

Time to Live

```
auth service ticket ttl
```

Description: When Ceph sends a client a ticket for authentication, the Ceph cluster assigns the ticket a time to live.

ceph 发给客户端一个用于认证的票据时分配给这个票据的生存期。

Type: Double

Default: 60*60

3.1.5 监视器配置

Monitor Config Reference

Understanding how to configure a Ceph monitor is an important part of building a reliable Ceph cluster. **All Ceph clusters have at least one monitor.** A monitor configuration usually remains fairly consistent, but you can add, remove or replace a monitor in a cluster. See [Adding/Removing a Monitor](#) for details.

理解如何配置 ceph 监视器是构建可靠集群的重要方面，任何 ceph 集群都需要至少一个监视器。一个监视器通常相当一致，但是你可以增加、删除、或替换集群中的监视器，详情见 [Adding/Removing a Monitor](#)。

3.1.5.1 背景

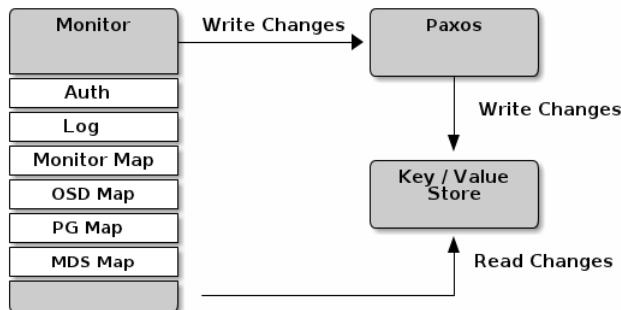
Background

Monitors maintain a “master copy” of the cluster map, which means a client can determine the location of all monitors, OSDs, and metadata servers just by connecting to one monitor and retrieving a current cluster map. Before Ceph clients can read from or write to OSDs or metadata servers, they must connect to a monitor first. With a current copy of the cluster map and the CRUSH algorithm, a client can compute the location for any object. The ability to compute object locations allows a client to talk directly to OSDs, which is a very important aspect of Ceph’s high scalability and performance.

监视器们维护着集群运行图的主副本，就是说客户端连到一个监视器并获取当前运行图就能确定所有监视器、OSD 和元数据服务器的位置。ceph 客户端读写 OSD 或元数据服务器前，必须先连到一个监视器，靠当前集群运行图的副本和 CRUSH 算法，客户端能计算出任何对象的位置，故此客户端有能力直接连到 OSD，这对 ceph 的高伸缩性、高性能来说非常重要。

The primary role of the monitor is to maintain a master copy of the cluster map. Monitors also provide authentication and logging services. Ceph monitors write all changes in the monitor services to a single Paxos instance, and Paxos writes the changes to a key/value store for strong consistency. Ceph monitors can query the most recent version of the cluster map during sync operations. Ceph monitors leverage the key/value store’s snapshots and iterators (using leveldb) to perform store-wide synchronization.

监视器的主要角色是维护集群运行图的主副本，它也提供认证和日志记录服务。ceph 监视器们把监视器服务的所有更改写入一个单独的 Paxos 例程，然后 Paxos 以键/值方式存储所有变更以实现高度一致性。同步期间，ceph 监视器能查询集群运行图的最新版本，它们通过操作键/值存储快照和迭代器（用 leveldb）来进行存储级同步。



Deprecated since version version: 0.58

从 0.58 开始废弃。

In Ceph versions 0.58 and earlier, Ceph monitors use a Paxos instance for each service and store the map as a file.

在 0.58 及更早版本中，ceph 监视器每个服务用一个 Paxos 例程，并把运行图存储为文件。

3.1.5.1.1 集群运行图

Cluster Maps

The cluster map is a composite of maps, including the monitor map, the OSD map, the placement group map and the metadata server map. The cluster map tracks a number of important things: which processes are `in` the cluster; which processes that are `in` the cluster are `up` and running or `down`; whether, the placement groups are `active` or `inactive`, and `clean` or in some other state; and, other details that reflect the current state of the cluster such as the total amount of storage space, and the amount of storage used.

集群运行图是多个图的组合，包括监视器图、OSD 图、归置组图和元数据服务器图。集群运行图追踪几个重要事件：哪些进程在集群里（`in`）；哪些进程在集群里（`in`）是 `up` 且在运行、或 `down`；归置组状态是 `active` 或 `inactive`、`clean` 或其他状态；和其他反映当前集群状态的信息，像总存储容量、和使用量。

When there is a significant change in the state of the cluster—e.g., an OSD goes down, a placement group falls into a degraded state, etc.—the cluster map gets updated to reflect the current state of the cluster. Additionally, the monitor also maintains a history of the prior states of the cluster. The monitor map, OSD map, placement group map and metadata server map each maintain a history of their map versions. We call each version an “epoch.”

当集群状态有明显变更时，如一 OSD 挂了、一归置组降级了等等，集群运行图会更新并反应集群当前状态。另外，监视器也维护着集群的主要状态历史。监视器图、OSD 图、归置组图和元数据服务器图各自维护着它们的运行图版本。我们把各版本称为一个 epoch。

When operating your cluster, keeping track of these states is an important part of your system administration duties. See [Monitoring a Cluster](#) and [Monitoring OSDs and PGs](#) for details.

运营集群时，跟踪这些状态是系统管理任务的重要部分。详情见 [Monitoring a Cluster](#) 和 [Monitoring OSDs and PGs](#)。

3.1.5.1.2 监视器法定人数

Monitor Quorum

Our 5-minute Quick Start provides a trivial [Ceph configuration file](#) that provides for one monitor in the test cluster. A cluster will run fine with a single monitor; however, **a single monitor is a single-point-of-failure**. To ensure high availability in a production cluster, you should run Ceph with multiple monitors so that the failure of a single monitor **WILL NOT** bring down your entire cluster.

前述的 5 分钟快速入门提供了一个简陋的 [ceph 配置文件](#)，它提供了一个监视器用于测试。只用一个监视器集群可以良好地运行，然而单监视器是一个单故障点，生产集群要实现高可用性的话得配置多个监视器，这样单个监视器的失效才不会影响整个集群。

When a cluster runs multiple monitors for high availability, Ceph monitors use [Paxos](#) to establish consensus about the master cluster map. A consensus requires a majority of monitors running to establish a quorum for consensus about the cluster map (e.g., 1; 2 out of 3; 3 out of 5; 4 out of 6; etc.).

集群用多个监视器实现高可用性时，多个监视器用 Paxos 算法对主集群运行图达成一致，这里的一致要求大多数监视器都在运行且够成法定人数（如 1 个、3 之 2 在运行、5 之 3、6 之 4 等等）。

3.1.5.1.3 一致性

Consistency

When you add monitor settings to your Ceph configuration file, you need to be aware of some of the architectural aspects of Ceph monitors. **Ceph imposes strict consistency requirements** for a Ceph monitor when discovering another Ceph monitor within the cluster. Whereas, Ceph clients and other Ceph daemons use the Ceph configuration file to discover monitors, monitors discover each other using the monitor map (`monmap`), not the Ceph configuration file.

你把监视器加进 ceph 配置文件时，得注意一些架构问题，ceph 发现集群内的其他监视器时对其有着严格的一致性要求。尽管如此，ceph 客户端和其他 ceph 守护进程用配置文件发现监视器，监视器却用监视器图（`monmap`）相互发现而非配置文件。

A monitor always refers to the local copy of the monmap when discovering other monitors in the cluster. Using the monmap instead of the Ceph configuration file 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**.

一个监视器发现集群内的其他监视器时总是参考 monmap 的本地副本，用 monmap 而非 ceph 配置文件避免了可能损坏集群的错误（如 `ceph.conf` 中指定地址或端口的拼写错误）。正因为监视器把 monmap 用于发现、并共享于客户端和其他 ceph 守护进程间，monmap 可严格地保证监视器的一致性是可靠的。

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. Maintaining a history enables a monitor that has an older version of the monmap to catch up with the current state of the cluster.

严格的一致性也适用于 monmap 的更新，因为关于监视器的任何更新、关于 monmap 的变更都是通过称为 Paxos 的分布式一致性算法传递的。监视器必须都同意 monmap 的每次更新，以确保法定人数里的每个监视器 monmap 版本相同，如增加、删除一个监视器。monmap 的更新是增量的，所以监视器们都有最近同意的版本，以及一系列之前版本。历史版本的存在允许一个落后的监视器跟上集群当前状态。

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 inadvertently use an older Ceph configuration 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.

如果监视器通过配置文件而非 monmap 相互发现，这会引进其他风险，因为 ceph 配置文件不是自动更新并分发的，监视器有可能不小心用了较老的配置文件，以致于不认识某监视器、放弃法定人数、或者产生一种 Paxos 不能确定当前系统状态的情形。

3.1.5.1.4 初始监视器

Bootstrapping Monitors

In most configuration and deployment cases, tools that deploy Ceph may help bootstrap the monitors by generating a monitor map for you (e.g., `mkcephfs`, `ceph-deploy`, etc). A monitor requires four explicit settings:

在大多数配置和部署案例中，部署 ceph 的工具可以帮你生成一个监视器图来初始化监视器（如 `mkcephfs`、`ceph-deploy` 等等），一个监视器需要 4 个选项：

- **Filesystem ID:** The `fsid` is the unique identifier for your object store. Since you can run multiple clusters on the same hardware, you must specify the unique ID of the object store when bootstrapping a monitor. Deployment tools usually do this for you (e.g., `mkcephfs` or `ceph-deploy` can call a tool like `uuidgen`), but you may specify the `fsid` manually too.

文件系统 ID：`fsid` 是对象存储的唯一标识符。你可以在一套硬件上运行多个集群，所以在初始化监视器时必须指定对象存储的唯一 ID。部署工具通常可替你完成（如 `mkcephfs` 或 `ceph-deploy` 会调用类似 `uuidgen` 的程序），但是你也可以手动指定 `fsid`。

- **Monitor ID:** A monitor ID is a unique ID assigned to each monitor within the cluster. It is an alphanumeric value, and by convention the identifier usually follows an alphabetical increment (e.g., `a`, `b`, etc.). This can be set in a Ceph configuration file (e.g., `[mon.a]`, `[mon.b]``, etc.), by a deployment tool, or using the `ceph` commandline.

监视器 ID：监视器 ID 是分配给集群内各监视器的唯一 ID，它是一个字母数字组合，为方便起见，标识符通常以字母顺序结尾（如 `a`、`b` 等等），可以设置于 ceph 配置文件（如 `[mon.a]`、`[mon.b]` 等等）、部署工具、或 `ceph` 命令行工具。

- **Keys:** The monitor must have secret keys. A deployment tool such as `mkcephfs` or `ceph-deploy` usually does this for you, but you may perform this step manually too. See [Monitor Keyrings](#) for details.

密钥：监视器必须有密钥。像 `mkcephfs` 或 `ceph-deploy` 这样的部署工具通常会自动生成，也可以手动完成。见 [Monitor Keyrings](#)。

For additional details on bootstrapping, see [Bootstrapping a Monitor](#).

关于初始化的其他信息见 [Bootstrapping a Monitor](#)。

3.1.5.2 监视器的配置

Configuring Monitors

To apply configuration settings to the entire cluster, enter the configuration settings under [global]. To apply configuration settings to all monitors in your cluster, enter the configuration settings under [mon]. To apply configuration settings to specific monitors, specify the monitor instance (e.g., [mon.a]). By convention, monitor instance names use alpha notation.

要把配置应用到整个集群，把它们放到[global]下；要用于所有监视器，置于[mon]下；要用于某监视器，指定监视器例程，如[mon.a]）。按惯例，监视器例程用字母命名。

```
[global]  
[mon]  
[mon.a]  
[mon.b]  
[mon.c]
```

3.1.5.2.1 最小配置

Minimum Configuration

The bare minimum monitor settings for a Ceph monitor via the Ceph configuration file include a hostname and a monitor address for each monitor. You can configure these under [mon] or under the entry for a specific monitor.

ceph 监视器的最简配置必须包括一个主机名和监视器地址，这些配置可置于[mon]下或某个监视器下。

```
[mon]  
    mon host = hostname1,hostname2,hostname3  
    mon addr = 10.0.0.10:6789,10.0.0.11:6789,10.0.0.12:6789  
[mon.a]  
    host = hostname1  
    mon addr = 10.0.0.10:6789
```

See the [Network Configuration Reference](#) for details.

详情参见 [Network Configuration Reference](#)。

Note: This minimum configuration for monitors assumes that a deployment tool generates the `fsid` and the `mon.` key for you.

注意：这里的监视器最简配置假设部署工具会自动给你生成 `fsid` 和 `mon.` 密钥。

Once you deploy a Ceph cluster, you **SHOULD NOT** change the IP address of the monitors. However, if you decide to change the monitor's IP address, you must follow a specific procedure. See [Changing a Monitor's IP Address](#) for details.

一旦部署了 ceph 集群，监视器 IP 不应该更改。然而，如果你决意要改，必须严格按照 [Changing a Monitor's IP Address](#) 来改。

3.1.5.2.2 集群 ID

Cluster ID

Each Ceph cluster has a unique identifier (`fsid`). If specified, it usually appears under the [global] section of the configuration file. Deployment tools usually generate the `fsid` and store it in the monitor map, so the value may not appear in a configuration file. The `fsid` makes it possible to run daemons for multiple clusters on the same hardware.

每个 ceph 集群都有一个唯一标识符（`fsid`）。如果要指定，它应该出现在配置文件的[global]段下。部署工具通常生成 `fsid` 并存于监视器图，所以不一定会出现在配置文件里，`fsid` 使得在一套硬件上运行多个集群成为可能。

`fsid`

Description: The cluster ID. One per cluster.
集群 ID，一集群一个。

Type: UUID
Required: Yes.

Default: N/A. May be generated by a deployment tool if not specified.
无。若未指定，部署工具会生成。

Note: Do not set this value if you use a deployment tool that does it for you.
注意：如果你用部署工具就没必要设置。

3.1.5.2.3 初始成员

Initial Members

We recommend running a production cluster with at least three monitors to ensure high availability. When you run multiple monitors, you may specify the initial monitors that must be members of the cluster in order to establish a quorum. This may reduce the time it takes for your cluster to come online.

我们建议在生产环境下最少部署 3 个监视器，以确保高可用性。运行多个监视器时，你可以指定为形成法定人数成员所需的初始监视器，这能减小集群上线时间。

```
[mon]
    mon initial members = a,b,c
```

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 (e.g., 3).
描述：

集群启动时初始监视器的 ID，若指定，ceph 需要奇数个监视器来确定最初法定人数（如 3）。

Type: String
Default: None

Note: A **majority** of monitors in your cluster must be able to reach each other in order to establish a quorum. You can decrease the initial number of monitors to establish a quorum with this setting.

注意：集群内的大多数监视器必须能互通以建立法定人数，你可以用此选项减小初始监视器数量来形成。

3.1.5.2.4 数据

Data

Ceph provides a default path where monitors store data. For optimal performance in a production cluster, we recommend running monitors on separate hosts and drives from OSDs. Monitors do lots of `fsync()`, which can interfere with OSD workloads.

ceph 监视器有存储数据的默认路径，生产集群为实现更高性能可把监视器部署到单独的主机和 OSD，因为监视器会频繁 `fsync()`，这可能影响 OSD。

In Ceph versions 0.58 and earlier, monitors store their data in files. This approach allows users to inspect monitor data with common tools like `ls` and `cat`. However, it doesn't provide strong consistency.

在 ceph 0.58 及更早版本中，监视器数据以文件保存，这样人们可以用 `ls` 和 `cat` 这些普通工具检查监视器数据，然而它不能提供健壮的一致性。

In Ceph versions 0.59 and later, monitors store their data as key/value pairs. Monitors require [ACID](#) transactions. Using a data store prevents recovering monitors from running corrupted versions through Paxos, and it enables multiple modification operations in one single atomic batch, among other advantages.

在 ceph 0.59 及后续版本中，监视器以键/值对存储数据。监视器需要 [ACID](#) 事务，数据存储的使用可防止监视器用损坏的版本进行恢复，除此之外，它允许在一个原子批量操作中进行多个修改操作。

Generally, we do not recommend changing the default data location. If you modify the default location, we recommend that you make it uniform across monitors by setting it in the [mon] section of the configuration file.

一般来说我们不建议更改默认数据位置，如果要改，我们建议所有监视器统一配置，加到配置文件的 [mon] 下。

mon data

Description: The monitor's data location.
监视器的数据位置。

Type: String

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

3.1.5.2.5 存储容量

Storage Capacity

When a Ceph cluster gets close to its maximum capacity (i.e., `mon osd full ratio`), Ceph prevents you from writing to or reading from OSDs as a safety measure to prevent data loss. Therefore, letting a production cluster approach its full ratio is not a good practice, because it sacrifices high availability. The default full ratio is `.95`, or 95% of capacity. This is a very aggressive setting for a test cluster with a small number of OSDs.

ceph 集群利用率接近最大容量时（如 `mon osd full ratio`），作为防止数据丢失的安全措施，它会阻止你读写 OSD。因此，让生产集群用满可不是好事，因为牺牲了高可用性。`full ratio` 默认值是 `.95` 或容量的 95%。对小型测试集群来说这是非常激进的设置。

Tip: When monitoring your cluster, be alert to warnings related to the `nearfull` ratio. This means that a failure of some OSDs could result in a temporary service disruption if one or more OSDs fails. Consider adding more OSDs to increase storage capacity.

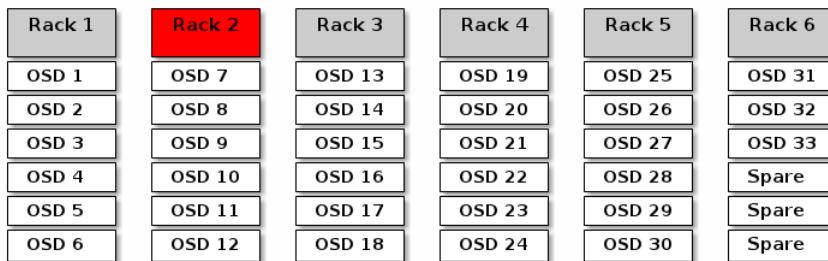
提示：监控集群时，要警惕和 `nearfull` 相关的警告。这意味着一些 OSD 的失败会导致临时服务中断，应该增加一些 OSD 来扩展存储容量。

A common scenario for test clusters involves a system administrator removing an OSD from the cluster to watch the cluster rebalance; then, removing another OSD, and so on until the cluster eventually reaches the full ratio and locks up. We recommend a bit of capacity planning even with a test cluster so that you can gauge how much spare capacity you will need to maintain for high availability. Ideally, you want to plan for a series of OSD failures where the cluster can recover to an `active + clean` state without replacing those OSDs immediately. You can run a cluster in an `active + degraded` state, but this is not ideal for normal operating conditions.

在测试集群的一个常见场景中，系统管理员从集群删除一个 OSD 接着观察重均衡；然后继续删除其他 OSD，直到集群达到占满率并锁死。我们建议，即使在测试集群里也要规划一点空闲容量用于保证高可用性。理想情况下，要做好这样的预案：一系列 OSD 失败后，短时间内未更换它们仍能恢复到 `active+clean` 状态。你也可以在 `active+degraded` 状态运行集群，但对正常使用来说并不好。

The following diagram depicts a simplistic Ceph cluster containing 33 hosts with one OSD per host, each OSD having a 3TB capacity. So this exemplary cluster has a maximum actual capacity of 99TB. With a `mon osd full ratio` of `0.95`, if the cluster falls to 5TB of remaining capacity, the cluster will not allow Ceph clients to read and write data. So its operating capacity is 95TB, not 99TB.

下图描述了一个简化的 ceph 集群，它包含 33 个主机、每主机一个 OSD、每 OSD 3TB 容量，所以这个小白鼠集群有 99TB 的实际容量，其 `mon osd full ratio` 为 `.95`。如果它只剩余 5TB 容量，集群就不允许客户端再读写数据，所以它的运行容量是 95TB，而非 99TB。



It is normal in such a cluster for one or two OSDs to fail. A less frequent but reasonable scenario involves a rack's router or power supply failing, which brings down multiple OSDs simultaneously (e.g., OSDs 7-12). In such a scenario, you should still strive for a cluster that can remain operational and achieve an `active + clean` state—even if that means adding a few hosts with additional OSDs in short order. If your capacity utilization is too high, you may not lose data, but you could still sacrifice data availability while resolving an outage within a failure domain if capacity utilization of the cluster exceeds the full ratio. For this reason, we recommend at least some rough capacity planning.

在这样的集群里，坏一或两个 OSD 很平常；一种罕见但可能发生的情景是一个机架的路由器或电源挂了，这会导致多个 OSD 同时离线（如 OSD 7-12），在这种情况下，你仍要力争保持集群可运行并达到 `active+clean` 状态，即使这意味着你得在短期内额外增加一些 OSD 及主机。如果集群利用率太高，在解决故障域期间也许不会丢数据，但很可能牺牲数据可用性，因为利用率超过了 `full ratio`。故此，我们建议至少要粗略地规划下容

量。

Identify two numbers for your cluster:

找出你集群的两个数字：

1. The number of OSDs.
OSD 数量。
2. The total capacity of the cluster
集群总容量。

If you divide the total capacity of your cluster by the number of OSDs in your cluster, you will find the mean average capacity of an OSD within your cluster. Consider multiplying that number by the number of OSDs you expect will fail simultaneously during normal operations (a relatively small number). Finally multiply the capacity of the cluster by the full ratio to arrive at a maximum operating capacity; then, subtract the number of amount of data from the OSDs you expect to fail to arrive at a reasonable full ratio. Repeat the foregoing process with a higher number of OSD failures (e.g., a rack of OSDs) to arrive at a reasonable number for a near full ratio.

用集群里 OSD 总数除以集群总容量，就能得到 OSD 平均容量；如果按预计的 OSD 数乘以这个值所得的结果计算（偏小），实际应用时将出错；最后再用集群容量乘以占满率能得到最大运行容量，然后扣除预估的 OSD 失败率；用较高的失败率（如整机架的 OSD）重复前述过程看是否接近占满率。

```
[global]
mon osd full ratio = .80
mon osd nearfull ratio = .70
```

mon osd full ratio

Description: The percentage of disk space used before an OSD is considered **full**.
OSD 硬盘使用率达到多少就认为它 **full**。

Type: Float
Default: .95

mon osd nearfull ratio

Description: The percentage of disk space used before an OSD is considered **nearfull**.
OSD 硬盘使用率达到多少就认为它 **nearfull**。

Type: Float
Default: .85

Tip: If some OSDs are nearfull, but others have plenty of capacity, you may have a problem with the CRUSH weight for the nearfull OSDs.

提示：如果一些 OSD 快满了，但其他的仍有足够空间，你可能配错 CRUSH 权重了。

3.1.5.2.6 心跳

Heartbeat

Ceph monitors know about the cluster by requiring reports from each OSD, and by receiving reports from OSDs about the status of their neighboring OSDs. Ceph provides reasonable default settings for monitor/OSD interaction; however, you may modify them as needed. See [Monitor/OSD Interaction](#) for details.

ceph 监视器要求各 OSD 向它报告、并接收 OSD 关于它们邻居的状态报告，以此来掌握集群。ceph 提供了监视器和 OSD 交互的合理默认值，然而你可以按需修改，详情见[监视器和 OSD 的交互](#)。

3.1.5.2.7 监视器存储同步

Monitor Store Synchronization

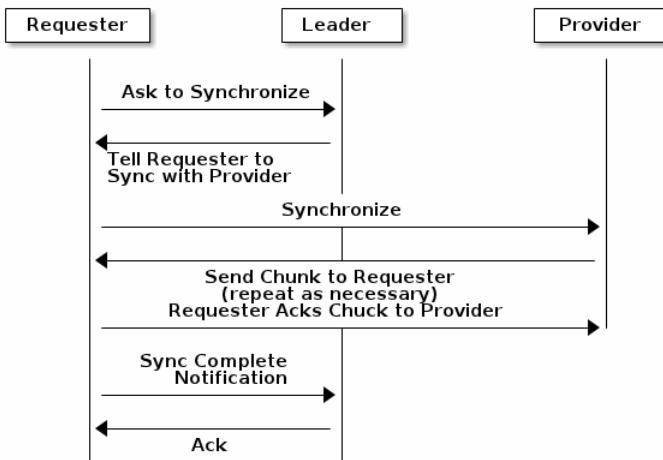
When you run a production cluster with multiple monitors (recommended), each monitor checks to see if a neighboring monitor has a more recent version of the cluster map (e.g., a map in a neighboring monitor with one or more epoch numbers higher than the most current epoch in the map of the instant monitor). Periodically, one monitor in the cluster may fall behind the other monitors to the point where it must leave the quorum, synchronize to retrieve the most current information about the cluster, and then rejoin the quorum. For the purposes of synchronization, monitors may assume one of three roles:

当你用多个监视器支撑一个生产集群时，各监视器都要检查邻居是否有集群运行图的最新版本（如，邻居监视器的图有一或多个 epoch 版本高于当前监视器的最高版 epoch），过一段时间，集群里的某个监视器可能落后于其它监视器太多而不得不离开法定人数，然后同步到集群当前状态，并重回法定人数。为了同步，监视器可能承担三种中的一种角色：

1. **Leader:** The *Leader* is the first monitor to achieve the most recent Paxos version of the cluster map.
leader: 它是实现最新 Paxos 版本的第一个监视器。
2. **Provider:** The *Provider* is a monitor that has the most recent version of the cluster map, but wasn't the first to achieve the most recent version.
provider: 有最新集群运行图的监视器，但不是第一个实现最新版。
3. **Requester:** A *Requester* is a monitor that has fallen behind the leader and must synchronize in order to retrieve the most recent information about the cluster before it can rejoin the quorum.
requester: 落后于 leader，重回法定人数前，必须同步以获取关于集群的最新信息。

These roles enable a leader to delegate synchronization duties to a provider, which prevents synchronization requests from overloading the leader—improving performance. In the following diagram, the requester has learned that it has fallen behind the other monitors. The requester asks the leader to synchronize, and the leader tells the requester to synchronize with a provider.

有了这些角色区分，leader 就可以给 provider 委派同步任务，这会避免同步请求压垮 leader、影响性能。在下面的图示中，requester 已经知道它落后于其它监视器，然后向 leader 请求同步，leader 让它去和 provider 同步。



Synchronization always occurs when a new monitor joins the cluster. During runtime operations, monitors may receive updates to the cluster map at different times. This means the leader and provider roles may migrate from one monitor to another. If this happens while synchronizing (e.g., a provider falls behind the leader), the provider can terminate synchronization with a requester.

新监视器加入集群时总要同步。在运行中，监视器会不定时收到集群运行图的更新，这就意味着 leader 和 provider 角色可能在监视器间漂移。如果这事发生在同步期间（如 provider 落后于 leader），provider 能终结和 requester 间的同步。

Once synchronization is complete, Ceph requires trimming across the cluster. Trimming requires that the placement groups are **active + clean**.

一旦同步完成，ceph 需要修复整个集群，使归置组回到 **active+clean** 状态。

`mon sync trim timeout`

Description:

Type: Double

Default: 30.0

`mon sync heartbeat timeout`

Description:

Type: Double

Default: 30.0

mon sync heartbeat interval

Description:

Type: Double

Default: 5.0

mon sync backoff timeout

Description:

Type: Double

Default: 30.0

mon sync timeout

Description:

Type: Double

Default: 30.0

mon sync max retries

Description:

Type: Integer

Default: 5

mon sync max payload size

Description: The maximum size for a sync payload.

同步载荷的最大尺寸。

Type: 32-bit Integer

Default: 1045676

mon accept timeout

Description: Number of seconds the Leader will wait for the Requester(s) to accept a Paxos update. It is also used during the Paxos recovery phase for similar purposes.

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。

Type: Float

Default: 10.0

paxos propose interval

Description: Gather updates for this time interval before proposing a map update.

提议更新之前收集本时间段的更新。

Type: Double

Default: 1.0

paxos min wait

Description: The minimum amount of time to gather updates after a period of inactivity.

经过一段不活跃时间后，收集更新的最小等待时间。

Type: Double

Default: 0.05

paxos trim tolerance

Description: The number of extra proposals tolerated before trimming.

修复前容忍的其他提议数量。

Type: Integer

Default: 30

paxos trim disabled max versions

Description: The maximum number of version allowed to pass without trimming.

允许不修复就通过的最大版本数

Type: Integer

Default: 100

mon lease

Description: The length (in seconds) of the lease on the monitor's versions.

监视器版本租期（秒）。

Type: Float
Default: 5

mon lease renew interval

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

Type: Float
Default: 3

mon lease ack timeout

Description: The number of seconds the Leader will wait for the Providers to acknowledge the lease extension.
leader 在等到 peons (随从) 确认延长租期前等待的时间

Type: Float
Default: 10.0

mon min osdmap epochs

Description: Minimum number of OSD map epochs to keep at all times.
一直保存的 OSD 图元素最小数量。

Type: 32-bit Integer
Default: 500

mon max pgmap epochs

Description: Maximum number of PG map epochs the monitor should keep.
监视器应该一直保存的 PG 图元素最大数量。

Type: 32-bit Integer
Default: 500

mon max log epochs

Description: Maximum number of Log epochs the monitor should keep.
监视器应该保留的最大日志数量。

Type: 32-bit Integer
Default: 500

3.1.5.2.8 Slurp (暴食?)

In Ceph version 0.58 and earlier, when a Paxos service drifts beyond a given number of versions, Ceph triggers the *slurp* mechanism, which establishes a connection with the quorum Leader and obtains every single version the Leader has for every service that has drifted. In Ceph versions 0.59 and later, slurp will not work, because there is a single Paxos instance for all services.

在 ceph 0.58 及之前版本中，当 Paxos 服务漂移出给定版本数时，就会触发 slurp 机制，它会和法定人数 leader 建立一个连接并获取 leader 拥有的每个版本，以同步每个漂移的服务。在 ceph 0.59 及后续版本，slurp 机制取消了，因为所有服务共享一个 Paxos 例程。

Deprecated since version 0.58.

从 0.58 过时。

paxos max join drift

Description: The maximum Paxos iterations before we must first sync the monitor data stores.
在我们首次同步监视器数据存储前，Paxos 迭代的最大数量。

Type: Integer
Default: 10

mon slurp timeout

Description: The number of seconds the monitor has to recover using slurp before the process is aborted and the monitor bootstraps.
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。

Type: Double
Default: 10.0

`mon slurp bytes`

Description: Limits the slurp messages to the specified number of bytes.
显式修复消息尺寸限制。

Type: 32-bit Integer

Default: 256 * 1024

3.1.5.2.9 时钟

Clock

`clock offset`

Description: How much to offset the system clock. See `Clock.cc` for details.
时钟可以漂移多少，详情见 `Clock.cc`。

Type: Double

Default: 0

Deprecated since version 0.58.

从 0.58 过时。

`mon tick interval`

Description: A monitor's tick interval in seconds.
监视器的心跳间隔，单位为秒。

Type: 32-bit Integer

Default: 5

`mon clock drift allowed`

Description: The clock drift in seconds allowed between monitors.
监视器间允许的时钟漂移量

Type: Float

Default: .050

`mon clock drift warn backoff`

Description: Exponential backoff for clock drift warnings
时钟偏移警告的退避指数

Type: Float

Default: 5

`mon timecheck interval`

Description: The time check interval (clock drift check) in seconds for the leader.
和 leader 的时间偏移检查（时钟漂移检查）。单位为秒。

Type: Float

Default: 300.0

3.1.5.2.10 客户端

Client

`mon client hung interval`

Description: The client will try a new monitor every N seconds until it establishes a connection.
客户端每 N 秒尝试一个新监视器，直到它建立连接。

Type: Double

Default: 3.0

`mon client ping interval`

Description: The client will ping the monitor every N seconds.
客户端每 N 秒 ping 一次监视器。

Type: Double

Default: 10.0

`mon client max log entries per message`

Description: The maximum number of log entries a monitor will generate per client message.
某监视器为每客户端生成的最大日志条数。

Type: Integer

Default: 1000

`mon client bytes`

Description: The amount of client message data allowed in memory (in bytes).
内存中允许存留的客户端消息数量。

Type: 64-bit Integer Unsigned

Default: 1000ul << 20

3.1.5.3 杂项

Miscellaneous

`mon max osd`

Description: The maximum number of OSDs allowed in the cluster.
集群允许的最大 OSD 数量。

Type: 32-bit Integer

Default: 10000

`mon globalid prealloc`

Description: The number of global IDs to pre-allocate for clients and daemons in the cluster.
为集群预分配的全局 ID 数量。

Type: 32-bit Integer

Default: 100

`mon sync fs threshold`

Description: Synchronize with the filesystem when writing the specified number of objects. Set it to 0 to disable it.
数量达到设定值时和文件系统同步，0 为禁用。

Type: 32-bit Integer

Default: 5

`mon subscribe interval`

Description: The refresh interval (in seconds) for subscriptions. The subscription mechanism enables obtaining the cluster maps and log information.
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。

Type: Double

Default: 300

`mon stat smooth intervals`

Description: Ceph will smooth statistics over the last N PG maps.
ceph 将平滑最后 N 个归置组图的统计信息。

Type: Integer

Default: 2

`mon probe timeout`

Description: Number of seconds the monitor will wait to find peers before bootstrapping.
监视器自举无效，搜寻节点前等待的时间。

Type: Double

Default: 2.0

`mon daemon bytes`

Description: The message memory cap for metadata server and OSD messages (in bytes).
给元数据服务器和 OSD 的消息使用的内存空间。

Type: 64-bit Integer Unsigned

Default: 4000ul << 20

mon max log entries per event

Description: The maximum number of log entries per event.
每个事件允许的最大日志条数。

Type: Integer

Default: 4096

3.1.6 心跳配置（监视器/OSD 交互的配置）

Configuring Monitor/OSD Interaction

After you have completed your initial Ceph configuration, you may deploy and run Ceph. When you execute a command such as `ceph health` or `ceph -s`, the monitor reports on the current state of the cluster. The monitor knows about the cluster by requiring reports from each OSD, and by receiving reports from OSDs about the status of their neighboring OSDs. If the monitor doesn't receive reports, or if it receives reports of changes in the cluster, the monitor updates the status of the cluster.

完成基本配置后就可以部署、运行 ceph 了。执行 `ceph health` 或 `ceph -s` 命令时，监视器会报告集群当前的状态。监视器通过让各 OSD 自己报告、并接收 OSD 关于邻居状态的报告来掌握集群动态。如果监视器没收到报告，或者它只收到集群的变更报告，那它就要更新集群状态。

Ceph provides reasonable default settings for monitor/OSD interaction. However, you may override the defaults. The following sections describe how Ceph monitors and OSDs interact for the purposes of monitoring.

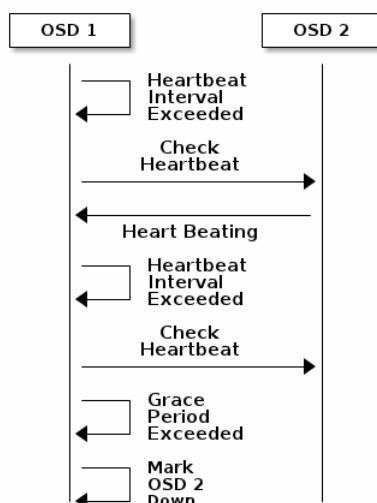
关于监视器/OSD 的交互 ceph 提供了合理的默认值，然而你可以覆盖它们。下面几段描述了 ceph 监视器如何与 OSD 交互。

3.1.6.1 OSD 验证心跳

OSDs Check Heartbeats

Each OSD checks the heartbeat of other OSDs every 6 seconds. You can change the heartbeat interval by adding an `osd heartbeat interval` setting under the `[osd]` section of your Ceph configuration file, or by setting the value at runtime. If an OSD doesn't show a heartbeat within a 20 second grace period, the cluster may consider the OSD down. You may change this grace period by adding an `osd heartbeat grace` setting under the `[osd]` section of your Ceph configuration file, or by setting the value at runtime.

各 OSD 每 6 秒会与其他 OSD 进行心跳检查，用 `[osd]` 下的 `osd heartbeat interval` 可更改此间隔、或运行时更改。如果一个 OSD 20 秒都没有心跳，集群就认为它 down 了，用 `[osd]` 下的 `osd heartbeat grace` 可更改宽限期、或者运行时更改。



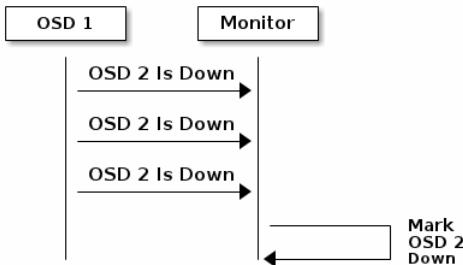
3.1.6.2 OSD 报告死亡 OSD

OSDs Report Down OSDs

By default, an OSD must report to the monitors that another OSD is down three times before the monitors acknowledge that the reported OSD is down. You can change the minimum number of `osd down` reports by

adding an `mon osd min down reports` setting (`osd min down reports` prior to v0.62) under the `[mon]` section of your Ceph configuration file, or by setting the value at runtime. By default, only one OSD is required to report another OSD down. You can change the number of OSDs required to report a monitor down by adding an `mon osd min down reporters` setting (`osd min down reporters` prior to v0.62) under the `[mon]` section of your Ceph configuration file, or by setting the value at runtime.

默认情况下，一个 OSD 必须向监视器报告三次另一个 OSD down 的消息，监视器才会认为那个被报告的 OSD down 了；配置文件里 `[mon]` 段下的 `mon osd min down reports` 选项（v0.62 之前是 `osd min down reports`）可更改这个最少 `osd down` 消息次数，或者运行时设置。默认情况下，只要有一个 OSD 报告另一个 OSD 挂的消息即可，配置文件里 `[mon]` 段下的 `mon osd min down reporters` 可用来更改必需 OSD 数（v0.62 之前的 `osd min down reporters`），或者运行时更改。

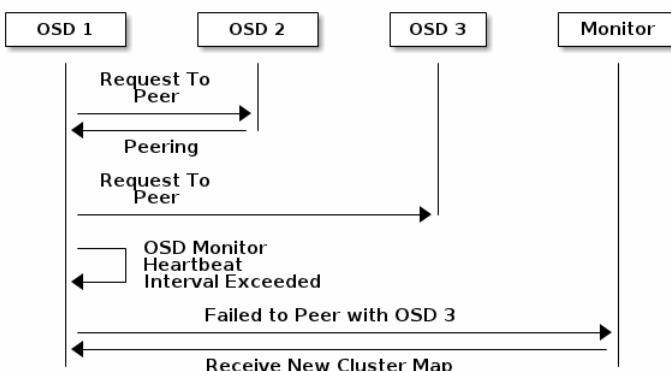


3.1.6.3 OSD 报告互联失败

OSDs Report Peering Failure

If an OSD cannot peer with any of the OSDs defined in its Ceph configuration file, it will ping the monitor for the most recent copy of the cluster map every 30 seconds. You can change the monitor heartbeat interval by adding an `osd mon heartbeat interval` setting under the `[osd]` section of your Ceph configuration file, or by setting the value at runtime.

如果一 OSD 不能和配置文件中定义的任何 OSD 建立连接，它会每 30 秒向监视器索要一次最新集群运行图，你可以在 `[osd]` 下设置 `osd mon heartbeat interval` 来更改这个心跳间隔，或者运行时更改。

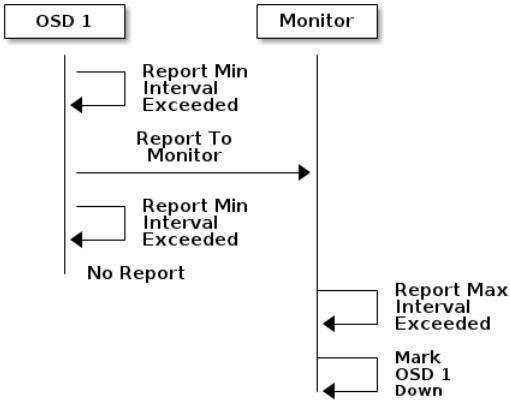


3.1.6.4 OSD 报告自己的状态

OSDs Report Their Status

If an OSD doesn't report to the monitor once at least every 120 seconds, the monitor will consider the OSD down. You can change the monitor report interval by adding an `osd mon report interval max` setting under the `[osd]` section of your Ceph configuration file, or by setting the value at runtime. The OSD attempts to report on its status every 30 seconds. You can change the OSD report interval by adding an `osd mon report interval min` setting under the `[osd]` section of your Ceph configuration file, or by setting the value at runtime.

如果一 OSD 至少 120 秒没向监视器报告过，监视器就认为它 down 了，你可以设置 `[osd]` 下的 `osd mon report interval max` 来更改此报告间隔，或者运行时更改。OSD 每 30 秒会报告它自己的状态，在 `[osd]` 段下设置 `osd mon report interval min` 可更改 OSD 报告间隔，或运行时更改。



3.1.6.5 配置选项

Configuration Settings

When modifying heartbeat settings, you should include them in the [global] section of your configuration file.

心跳选项应该置于配置文件的[global]段下。

3.1.6.5.1 监视器选项

Monitor Settings

`mon osd min up ratio`

Description: The minimum ratio of up OSDs before Ceph will mark OSDs down.

在把 OSD 标记为 `down` 前，保持处于 `up` 状态的 OSD 最小比例

Type: Double

Default: .3

`mon osd min in ratio`

Description: The minimum ratio of in OSDs before Ceph will mark OSDs out.

在把 OSD 标记为 `out` 前，保持处于 `in` 状态的 OSD 最小比例

Type: Double

Default: .3

`mon osd laggy halflife`

Description: The number of seconds laggy estimates will decay.

滞后量消退时间，秒。

Type: Integer

Default: 60*60

`mon osd laggy weight`

Description: The weight for new samples in laggy estimation decay.

滞后量消退时新样本的权重。

Type: Double

Default: 0.3

`mon osd adjust heartbeat grace`

Description: If set to `true`, Ceph will scale based on laggy estimations.

设置为 `true` 时，ceph 将根据滞后量伸缩。

Type: Boolean

Default: true

`mon osd adjust down out interval`

Description: If set to `true`, Ceph will scaled based on laggy estimations.

设置为 `true` 时，ceph 将根据滞后量伸缩。

Type: Boolean
Default: `true`

mon osd auto mark in

Description: Ceph will mark any booting OSDs as `in` in the cluster.
ceph 将把任何启动中的 OSD 标记为在集群中 (`in`)。

Type: Boolean
Default: `false`

mon osd auto mark auto out in

Description: Ceph will mark booting OSDs auto marked `out` of the cluster as `in` in the cluster.
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。

Type: Boolean
Default: `true`

mon osd auto mark new in

Description: Ceph will mark booting new OSDs as `in` in the cluster.
把 booting 状态的新 OSD 标记为 in。

Type: Boolean
Default: `true`

mon osd down out interval

Description: The number of seconds Ceph waits before marking an OSD `down` and `out` if it doesn't respond.
在 OSD 停止响应多少秒后把它标记为 down。

Type: 32-bit Integer
Default: `300`

mon osd downout subtree limit

Description: The largest CRUSH unit type that Ceph will automatically mark `out`.
ceph 可以自动标记为 out 的 CRUSH 单元。

Type: String
Default: `rack`

mon osd report timeout

Description: The grace period in seconds before declaring unresponsive OSDs `down`.
宣布无响应 OSD `down` 前的宽限期，秒。

Type: 32-bit Integer
Default: `900`

mon osd min down reporters

Description: The minimum number of OSDs required to report a `down` OSD.
确定一 OSD `down` 的最少报告来源 OSD 数。

Type: 32-bit Integer
Default: `1`

mon osd min down reports

Description: The minimum number of times an OSD must report that another is `down`.
一 OSD 必须报告另一个 `down` 的次数。

Type: 32-bit Integer
Default: `3`

3.1.6.5.2 OSD 选项

OSD Settings

`osd heartbeat address`

Description: An OSD's network address for heartbeats.

OSD 用于心跳的网络地址。

Type: Address

Default: The host address.

本主机地址。

`osd heartbeat interval`

Description: How often an OSD pings its peers (in seconds).

— OSD 探测邻居的频率，秒。

Type: 32-bit Integer

Default: 6

`osd heartbeat grace`

Description: The elapsed time when an OSD hasn't shown a heartbeat that the cluster considers it down.

OSD 多久没心跳就会被集群认为它挂 (down) 了。

Type: 32-bit Integer

Default: 20

`osd mon heartbeat interval`

Description: How often the OSD pings a monitor if it has no OSD peers.

OSD 没有邻居时多久探测一次监视器。

Type: 32-bit Integer

Default: 30

`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.

监视器允许 OSD 报告的最大间隔，超时将认为 OSD 挂了 (down)。

Type: 32-bit Integer

Default: 120

`osd mon report interval min`

Description: The minimum number of seconds for an OSD to report to a monitor to avoid the monitor considering the OSD down.

为免监视器把 OSD 标记为 down，进行报告的最小间隔。

Type: 32-bit Integer

Default: 5

Valid Range: Should be less than `osd mon report interval max`

要小于 `osd mon report interval max`

`osd mon ack timeout`

Description: The number of seconds to wait for a monitor to acknowledge a request for statistics.

OSD 等待监视器提供统计信息的时间，秒。

Type: 32-bit Integer

Default: 30

3.1.7 OSD 配置

OSD Config Reference

You can configure OSDs in the Ceph configuration file, but OSDs can use the default values and a very minimal configuration. A minimal OSD configuration sets `osd journal size` and `osd host`, and uses default values for nearly everything else.

你可以在配置文件里配置 OSD，但它可以用默认值和最小化配置。最简 OSD 配置需设置 `osd journal size` 和 `osd host`，其他几乎都能用默认值。

OSDs are numerically identified in incremental fashion, beginning with 0 using the following convention.

OSD 用递增数字标识，从 0 开始，按惯例这样表示：

```
osd.0  
osd.1  
osd.2
```

In a configuration file, you may specify settings for all OSDs in the cluster by adding configuration settings to the [osd] section of your configuration file. To add settings directly to a specific OSD (e.g., `osd host`), enter it in an OSD-specific section of your configuration file. For example:

在配置文件里，你可以在 [osd] 下配置所有 OSD；要添加针对特定 OSD 的选项（如 `osd host`），把它放到那个 OSD 段下即可，如：

```
[osd]  
    osd journal size = 1024  
  
[osd.0]  
    osd host = osd-host-a  
  
[osd.1]  
    osd host = osd-host-b
```

3.1.7.1 常规配置

General Settings

The following settings provide an OSD's ID, and determine paths to data and journals. Ceph deployment scripts typically generate the UUID automatically. We DO NOT recommend changing the default paths for data or journals, as it makes it more problematic to troubleshoot Ceph later.

下列选项给出一 OSD 的 ID，可以确定数据和日志的路径。ceph 部署脚本通常会自动生成 UUID。我们不建议更改数据和日志的默认路径，因为会使稍后的排障更麻烦。

The journal size should be at least twice the product of the expected drive speed multiplied by `filestore max sync interval`. However, the most common practice is to partition the journal drive (often an SSD), and mount it such that Ceph uses the entire partition for the journal.

日志尺寸最小应该是期望的驱动器速度和 `filestore max sync interval` 的乘积；最常见的方法是为日志驱动器（通常是 SSD）分区并挂载好，使得 ceph 可以用整个分区做日志。

osd uuid

Description: The universally unique identifier (UUID) for the OSD.

OSD 的全局唯一标识符（UUID）。

Type: UUID

Default: The UUID.

Note: The `osd uuid` applies to a single OSD. The `fsid` applies to the entire cluster.

`osd uuid` 适用于单个 OSD，`fsid` 适用于整个集群。

osd data

Description: The path to the OSDs data. You must create the directory when deploying Ceph. You should mount a drive for OSD data at this mount point. We do not recommend changing the default. 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。

Type: String

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

osd max write size

Description: The maximum size of a write in megabytes.

一次写入的最大尺寸，MB。

Type: 32-bit Integer

Default: 90

osd client message size cap

Description: The largest client data message allowed in memory.

内存里允许的最大客户端数据消息。

Type: 64-bit Integer Unsigned

Default: 500MB default. `500*1024L*1024L`

osd class dir

Description: The class path for RADOS class plug-ins.

RADOS 类插件的路径。

Type: String

Default: \$libdir/rados-classes

3.1.7.2 日志选项

Journal Settings

By default, Ceph expects that you will store an OSDs journal with the following path:

默认情况下，ceph 觉得你会把 OSD 日志存储于下列路径：

```
/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 会把日志存储在和 OSD 数据相同的硬盘上。追求高性能的 OSD 可用单独的硬盘存储日志数据，如固态硬盘能提供高性能日志。

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 max sync interval` and the expected throughput, and multiply the product by two (2):

`osd journal size` 默认值是 0，所以你得在 `ceph.conf` 里设置。此值应该是 `filestore max sync interval` 和期望吞吐量的乘积再乘以 2。

```
osd journal size = {2 * (expected throughput * filestore max 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:

期望的吞吐量应包含期望的硬盘吞吐量（如持续数据传输速率）、和网络吞吐量，例如一个 7200 转硬盘的速度大致是 100MB/s。硬盘和网络吞吐量中较小的一个是相对合理的吞吐量，有的用户则以 10GB 日志尺寸起步，例如：

```
osd journal size = 10000
```

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. We recommend using a drive separate from the `osd data` drive.

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。

Type: String

Default: /var/lib/ceph/osd/\$cluster-\$id/journal

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.

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。

Type: 32-bit Integer

Default: 5120

Recommended: Begin with 1GB. Should be at least twice the product of the expected speed multiplied by `filestore max sync interval`.

最少 1G，应该是期望的驱动器速度和 `filestore max sync interval` 的乘积。

See [Journal Config Reference](#) for additional details.

详情见 xxx

3.1.7.3 监视器和OSD 的交互

Monitor OSD Interaction

OSDs check each other's heartbeats and report to monitors periodically. Ceph can use default values in many cases. However, if your network has latency issues, you may need to adopt longer intervals. See [Configuring Monitor/OSD Interaction](#) for a detailed discussion of heartbeats.

OSD 周期性地相互检查心跳并报告给监视器。多数情况下，ceph 都可以用默认值，但是如果你的网络延时大，就得用较长间隔。关于心跳的讨论参见 [Configuring Monitor/OSD Interaction](#)。

3.1.7.4 数据归置

Data Placement

See [Pool & PG Config Reference](#) for details.

详情见 [Pool & PG Config Reference](#)。

3.1.7.5 洗刷

Scrubbing

In addition to making multiple copies of objects, Ceph insures data integrity by scrubbing placement groups. Ceph scrubbing is analogous to `fsck` on the object storage layer. For each placement group, Ceph generates a catalog of all objects and compares each primary object and its replicas to ensure that no objects are missing or mismatched. Light scrubbing (daily) checks the object size and attributes. Deep scrubbing (weekly) reads the data and uses checksums to ensure data integrity.

除了为对象保存多个副本外，ceph 还靠洗刷归置组来保证数据完整性。这种洗刷类似对象存储层的 `fsck`，对每个归置组，ceph 生成一个所有对象的目录，并比对每个主对象及其副本以确保没有对象丢失或错配。轻微洗刷（每天）检查对象尺寸和属性，深层洗刷（每周）会读出数据并用校验和保证数据完整性。

Scrubbing is important for maintaining data integrity, but it can reduce performance. You can adjust the following settings to increase or decrease scrubbing operations.

洗刷对维护数据完整性很重要，但会影响性能；你可以用下列选项来增加或减少洗刷操作。

`osd max scrubs`

Description: The maximum number of scrub operations for an OSD.

— OSD 的最大并发洗刷操作数。

Type: 32-bit Int

Default: 1

`osd scrub thread timeout`

Description: The maximum time in seconds before timing out a scrub thread.

洗刷线程最大死亡时值。

Type: 32-bit Integer

Default: 60

`osd scrub finalize thread timeout`

Description: The maximum time in seconds before timing out a scrub finalize thread.

洗刷终结线程最大超时值。

Type: 32-bit Integer

Default: 60*10

`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%.

最大 CPU 负载，当前 CPU 使用率高于此值时 ceph 不会洗刷。默认 50%。

Type: Float

Default: 0.5

`osd scrub min interval`

Description: The maximum interval in seconds for scrubbing the OSD when the cluster load is low.

集群负载低的时候，洗刷的最大间隔时间，秒。

Type: Float
Default: 5 minutes. 300

osd scrub max interval

Description: The maximum interval in seconds for scrubbing the OSD irrespective of cluster load.
不论集群负载如何，都要进行洗刷的时间间隔。

Type: Float
Default: Once per day. 60*60*24

osd deep scrub interval

Description: The interval for “deep” scrubbing (fully reading all data).
深层洗刷的间隔（完整地读所有数据）。

Type: Float
Default: Once per week. 60*60*24*7

osd deep scrub stride

Description: Read size when doing a deep scrub.
深层洗刷时的读取尺寸。

Type: 32-bit Int
Default: 512 KB. 524288

3.1.7.6 操作数

Operations

Operations settings allow you to configure the number of threads for servicing requests. If you set `osd op threads` to 0, it disables multi-threading. By default, Ceph uses two threads with a 30 second timeout and a 30 second complaint time if an operation doesn't complete within those time parameters. You can set operations priority weights between client operations and recovery operations to ensure optimal performance during recovery.

操作数选项允许你设置用于服务的线程数，如果把 `osd op threads` 设置为 0 就禁用了多线程。默认情况下，ceph 用两个参数 30 秒超时和 30 秒抗议时间来把握线程情况，如果一个操作在这些限定时间内没完成（ceph 会采取措施）。你可以调整客户端操作和恢复操作的优先程度来保证恢复期间仍有良好的性能。

osd op threads

Description: The number of threads to service OSD operations. Set to 0 to disable it. Increasing the number may increase the request processing rate.

OSD 操作线程数，0 为禁用。增大数量可以增加请求处理速度。

Type: 32-bit Integer
Default: 2

osd client op priority

Description: The priority set for client operations. It is relative to `osd recovery op priority`.
设置客户端操作优先级，它相对于 `osd recovery op priority`。

Type: 32-bit Integer
Default: 63

Valid Range: 1-63

osd recovery op priority

Description: The priority set for recovery operations. It is relative to `osd client op priority`.
设置恢复优先级，其值相对于 `osd client op priority`。

Type: 32-bit Integer
Default: 10

Valid Range: 1-63

osd op thread timeout

Description: The OSD operation thread timeout in seconds.
OSD 线程超时秒数。

Type: 32-bit Integer
Default: 30

`osd op complaint time`

Description: An operation becomes complaint worthy after the specified number of seconds have elapsed.
一个操作进行多久后开始抱怨。

Type: Float

Default: 30

`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.
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。

Type: 32-bit Integer

Default: 1

`osd op history size`

Description: The maximum number of completed operations to track.
要跟踪的最大已完成操作数量。

Type: 32-bit Unsigned Integer

Default: 20

`osd op history duration`

Description: The oldest completed operation to track.
要跟踪的最老已完成操作。

Type: 32-bit Unsigned Integer

Default: 600

`osd op log threshold`

Description: How many operations logs to display at once.
一次显示多少操作日志。

Type: 32-bit Integer

Default: 5

3.1.7.7 回填

Backfilling

When you add or remove OSDs to a cluster, the CRUSH algorithm will want to rebalance the cluster by moving placement groups to or from OSDs to restore the balance. The process of migrating placement groups and the objects they contain can reduce the cluster's operational performance considerably. To maintain operational performance, Ceph performs this migration with 'backfilling', which allows Ceph to set backfill operations to a lower priority than requests to read or write data.

当你增加或移除 OSD 时，CRUSH 算法将要求重新均衡集群，它会把一些归置组移出或移入多个 OSD 以回到均衡状态。归置组和对象的迁移过程会明显降低集群运营性能，为维持运营性能，ceph 用 backfilling 来执行此迁移，它可以使得 ceph 的回填操作优先级低于用户读写请求。

`osd max backfills`

Description: The maximum number of backfills allowed to or from a single OSD.
单个 OSD 允许的最大回填操作数。

Type: 64-bit Unsigned Integer

Default: 10

`osd backfill scan min`

Description: The scan interval in seconds for backfill operations when cluster load is low.
集群负载低时，回填操作时扫描间隔。

Type: 32-bit Integer

Default: 64

`osd backfill scan max`

Description: The maximum scan interval in seconds for backfill operations irrespective of cluster load.
回填操作时最大扫描间隔。

Type: 32-bit Integer

Default: 512

`osd backfill full ratio`

Description: Refuse to accept backfill requests when the OSD's full ratio is above this value.

OSD 的占满率达到多少时拒绝接受回填请求。

Type: Float

Default: 0.85

`osd backfill retry interval`

Description: The number of seconds to wait before retrying backfill requests.

重试回填请求前等待秒数。

Type: Double

Default: 10.0

3.1.7.8 OSD 运行图

OSD Map

OSD maps reflect the OSD daemons operating in the cluster. Over time, the number of map epochs increases. Ceph provides some settings to ensure that Ceph performs well as the OSD map grows larger.

OSD 运行图反映集群中运行的 OSD 守护进程，斗转星移，图元增加。ceph 用一些选项来确保 OSD 运行图增大时仍运行良好。

`osd map dedup`

Description: Enable removing duplicates in the OSD map.

允许删除 OSD 图里的重复项。

Type: Boolean

Default: true

`osd map cache size`

Description: The size of the OSD map cache in megabytes.

OSD 图缓存尺寸，MB。

Type: 32-bit Integer

Default: 500

`osd map cache bl size`

Description: The size of the in-memory OSD map cache in OSD daemons.

OSD 进程中，驻留内存的 OSD 图缓存尺寸。

Type: 32-bit Integer

Default: 50

`osd map cache bl inc size`

Description: The size of the in-memory OSD map cache incrementals in OSD daemons.

OSD 进程中，驻留内存的 OSD 图缓存增量尺寸。

Type: 32-bit Integer

Default: 100

`osd map message max`

Description: The maximum map entries allowed per MOSDMap message.

每个 MOSDMap 图消息允许的最大条目数量。

Type: 32-bit Integer

Default: 100

3.1.7.9 恢复

Recovery

When the cluster starts or when an OSD crashes and restarts, the OSD begins peering with other OSDs before writes can occur. See [Monitoring OSDs and PGs](#) for details.

当集群重启或某 OSD 崩溃后重启时，此 OSD 开始与其它 OSD 们建立连接，这样才能正常工作。详情见 [Monitoring OSDs and PGs](#)。

If an OSD crashed and comes back online, usually it will be out of sync with other OSDs containing more

recent versions of objects in the placement groups. When this happens, the OSD goes into recovery mode and seeks to get the latest copy of the data and bring its map back up to date. Depending upon how long the OSD was down, the OSD's objects and placement groups may be significantly out of date. Also, if a failure domain went down (e.g., a rack), more than one OSD may come back online at the same time. This can make the recovery process time consuming and resource intensive.

如果某 OSD 崩溃并重生，通常和其他 OSD 不同步，没有同归置组内最新版本的对象。这时，OSD 进入恢复模式并且搜索最新数据副本，并更新运行图。根据 OSD 挂的时间长短，OSD 的对象和归置组可能明显过期，另外，如果一个失效域挂了（如一个机架），多个 OSD 会同时重生，这样恢复时间更长、更耗资源。

To maintain operational performance, Ceph performs recovery with limitations on the number recovery requests, threads and object chunk sizes which allows Ceph perform well in a degraded state.

为保持运营性能，ceph 进行恢复时会限制恢复请求数、线程数、对象块尺寸，这样在降级状态下也能运行良好。

osd recovery delay start

Description: After peering completes, Ceph will delay for the specified number of seconds before starting to recover objects.

对等关系建立完毕后，ceph 开始对象恢复前等待的时间（秒）。

Type: Float

Default: 15

osd recovery max active

Description: The number of active recovery requests per OSD at one time. More requests will accelerate recovery, but the requests places an increased load on the cluster.

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。

Type: 32-bit Integer

Default: 5

osd recovery max chunk

Description: The maximum size of a recovered chunk of data to push.

一次推送的数据块的最大尺寸。

Type: 64-bit Integer Unsigned

Default: 1 << 20

osd recovery threads

Description: The number of threads for recovering data.

数据恢复时的线程数。

Type: 32-bit Integer

Default: 1

osd recovery thread timeout

Description: The maximum time in seconds before timing out a recovery thread.

恢复线程最大死亡时值。

Type: 32-bit Integer

Default: 30

osd recover clone overlap

Description: Preserves clone overlap during recovery. Should always be set to true.

数据恢复和迁移时保留的克隆重叠率。

Type: Boolean

Default: true

3.1.7.10 杂项

Miscellaneous

osd snap trim thread timeout

Description: The maximum time in seconds before timing out a snap trim thread.

快照修复线程最大死亡时值。

Type: 32-bit Integer

Default: 60*60*1

osd backlog thread timeout

Description: The maximum time in seconds before timing out a backlog thread.
积压线程最大死亡时值。

Type: 32-bit Integer

Default: 60*60*1

osd default notify timeout

Description: The OSD default notification timeout (in seconds).
OSD 默认通告超时。

Type: 32-bit Integer Unsigned

Default: 30

osd check for log corruption

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

Type: Boolean

Default: false

osd remove thread timeout

Description: The maximum time in seconds before timing out a remove OSD thread.
OSD 删除线程的最大死亡时值。

Type: 32-bit Integer

Default: 60*60

osd command thread timeout

Description: The maximum time in seconds before timing out a command thread.
命令线程最大超时值。

Type: 32-bit Integer

Default: 10*60

osd command max records

Description: Limits the number of lost objects to return.
限制返回的丢失对象数量。

Type: 32-bit Integer

Default: 256

osd auto upgrade tmap

Description: Uses tmap for omap on old objects.
在旧对象上给 omap 使用 tmap。

Type: Boolean

Default: true

osd tmapput sets users tmap

Description: Uses tmap for debugging only.
只在调试时使用 tmap。

Type: Boolean

Default: false

osd preserve trimmed log

Description: Preserves trimmed log files, but uses more disk space.
Type: Boolean

Default: false

3.1.8 文件存储配置

Filestore config reference

filestore debug omap check

Description: Debugging check on synchronization. Expensive. For debugging only.
同步检查。昂贵的调试选项。

Type: Boolean
Required: No
Default: 0

3.1.8.1 扩展属性

EXTENDED ATTRIBUTES

Extended Attributes (XATTRs) are an important 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) 是配置里的一个重要方面。一些文件系统对 XATTR 字节数有限制，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。

Ceph XATTRs are stored as `inline xattr`, using the XATTRs provided by the underlying file system, if it does not impose a size limit. If there is a size limit (4KB total on ext4, for instance), some Ceph XATTRs will be stored in an key/value database (aka `omap`) when the `filestore max inline xattr size` or `filestore max inline xattrs` threshold are reached.

ceph 扩展属性用底层文件系统（如果没有尺寸限制）的 XATTR 存储为 `inline xattr`。如果有限制，如 ext4 限制为 4KB，达到 `filestore max inline xattr size` 或 `filestore max inline xattrs` 阈值时一些 XATTR 将存储为键/值数据库（也叫 `omap`）。

`filestore xattr use omap`

Description: Use object map for XATTRS. Set to `true` for ext4 file systems.
为 XATTR 使用对象图，采用 ext4 文件系统时要设置为 `true`。

Type: Boolean
Required: No
Default: false

`filestore max inline xattr size`

Description: The maximum size of an XATTR stored in the filesystem (i.e., XFS, btrfs, ext4, etc.) per object. Should not be larger than the filesystem can handle.

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。

Type: Unsigned 32-bit Integer
Required: No
Default: 512

`filestore max inline xattrs`

Description: The maximum number of XATTRs stored in the filesystem per object.
每个对象存储在文件系统里的 XATTR 数量。

Type: 32-bit Integer
Required: No
Default: 2

3.1.8.2 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 to 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.

同步 filestore 的最大间隔秒数。

Type: Double
Required: No
Default: 5

filestore min sync interval

Description: The minimum interval in seconds for synchronizing the filestore.
同步 filestore 的最小间隔秒数。

Type: Double
Required: No
Default: .01

3.1.8.3 flusher (回冲器?)

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 `sync file range` 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.
启用 filestore flusher 功能。

Type: Boolean
Required: No
Default: false

filestore flusher max fds

Description: Sets the maximum number of file descriptors for the flusher.
设置回写器的最大文件描述符数量。

Type: Integer
Required: No
Default: 512

filestore sync flush

Description: Enables the synchronization flusher.
启用同步回写器。

Type: Boolean
Required: No
Default: false

filestore fsync flushes journal data

Description: Flush journal data during filesystem synchronization.
文件系统同步时也回写日志数据。

Type: Boolean
Required: No
Default: false

3.1.8.4 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.
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。

Type: Integer
Required: No. Minimal impact on performance.
无。对性能影响最小。
Default: 500

filestore queue max bytes

Description: The maximum number of bytes for an operation.
一个操作的最大字节数。

Type: Integer
Required: No
Default: 100 << 20

filestore queue committing max ops

Description: The maximum number of operations the filestore can commit.
文件存储能提交的最大操作数。

Type: Integer
Required: No
Default: 500

filestore queue committing max bytes

Description: The maximum number of bytes the filestore can commit.
文件存储器能提交的最大字节数。

Type: Integer
Required: No
Default: 100 << 20

3.1.8.5 超时选项

TIMEOUTS

filestore op threads

Description: The number of filesystem operation threads that execute in parallel.
最大并行文件系统操作线程数。

Type: Integer
Required: No
Default: 2

filestore op thread timeout

Description: The timeout for a filesystem operation thread (in seconds).
文件系统操作线程超时值，单位为秒。

Type: Integer
Required: No
Default: 60

filestore op thread suicide timeout

Description: The timeout for a commit operation before cancelling the commit (in seconds).
提交操作超时值（秒），超时后会取消。

Type: Integer
Required: No
Default: 180

3.1.8.6 B-tree 文件系统

B-TREE FILESYSTEM

filestore btrfs snap

Description: Enable snapshots for a btrfs filestore.
对 btrfs 文件存储器启用快照功能。

Type: Boolean
Required: No. Only used for btrfs.

无。仅适用于 btrfs。

Default: true

filestore btrfs clone range

Description: Enable cloning ranges for a btrfs filestore.

允许 btrfs 文件存储克隆动作排队。

Type: Boolean

Required: No. Only used for btrfs.

无。仅适用于 btrfs。

Default: true

3.1.8.7 日志 (文件系统)

JOURNAL

filestore journal parallel

Description: Enables parallel journaling, default for btrfs.

允许并发记日志，对 btrfs 默认开。

Type: Boolean

Required: No

Default: false

filestore journal writeahead

Description: Enables writeahead journaling, default for xfs.

允许预写日志，对 xfs 默认开。

Type: Boolean

Required: No

Default: false

filestore journal trailing

Description: Deprecated, never use.

过时了，从没用过。

Type: Boolean

Required: No

Default: false

3.1.8.8 杂项

MISC

filestore merge threshold

Description: Min number of files in a subdir before merging into parent

并入父目录前，子目录内的最小文件数。

Type: Integer

Required: No

Default: 10

filestore split multiple

Description: filestore_split_multiple*filestore_merge_threshold*16 is the max files in a subdir before splitting into child directories.

filestore_split_multiple*filestore_merge_threshold*16 是分割为子目录前某目录内的最大文件数。

Type: Integer

Required: No

Default: 2

filestore update to

Description: Limits filestore auto upgrade to specified version.

限制文件存储自动更新到某个指定版本。

Type: Integer

Required: No

Default: 1000

filestore blackhole

Description: Drop any new transactions on the floor.
丢弃任何讨论中的事务。

Type: Boolean

Required: No

Default: false

filestore dump file

Description: File onto which store transaction dumps?
存储事务转储目的文件。

Type: Boolean

Required: No

Default: false

filestore kill at

Description: inject a failure at the n'th opportunity
在第 N 次机会后注入一个失效。

Type: String

Required: No

Default: false

filestore fail eio

Description: Fail/Crash on eio.
在 IO 错误的时候失败或崩溃。

Type: Boolean

Required: No

Default: true

3.1.9 日志配置

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.

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。

Ceph OSDs support the following journal settings:

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`.

对日志启用径直 IO，需要 `journal block align` 设置为 `true`。

Type: Boolean

Required: Yes when using `aio`.

用 `aio` 时自动启用。

Default: `true`

`journal aio`

Description: Enables using `libaio` for asynchronous writes to the journal. Requires `journal dio` set to `true`.

异步写入日志时用 `libaio` 库，需要 `journal dio` 设为 `true`。

Type: Boolean

Required: No.

Default: `false`

`journal block align`

Description: Block aligns writes. Required for `dio` and `aio`.

块对齐写，`dio` 和 `aio` 需要。

Type: Boolean

Required: Yes when using `dio` and `aio`.

用 `dio` 和 `aio` 时自动启用。

Default: `true`

`journal max write bytes`

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

一次写入日志的最大尺寸。

Type: Integer

Required: No

Default: `10 << 20`

`journal max write entries`

Description: The maximum number of entries the journal will write at any one time.

一次写入日志的最大数量。

Type: Integer

Required: No

Default: `100`

`journal queue max ops`

Description: The maximum number of operations allowed in the queue at any one time.

队列里一次允许的最大操作数量。

Type: Integer

Required: No

Default: `500`

`journal queue max bytes`

Description: The maximum number of bytes allowed in the queue at any one time.

队列里一次允许的最大字节数。

Type: Integer

Required: No

Default: `10 << 20`

`journal align min size`

Description: Align data payloads greater than the specified minimum.

对齐大于指定最小值的数据有效载荷。

Type: Integer

Required: No

Default: `64 << 10`

`journal zero on create`

Description: Causes the file store to overwrite the entire journal with 0's during `mkfs`.
在创建文件系统期间用 0 填充整个日志。

Type: Boolean
Required: No
Default: `false`

3.1.10 存储池、归置组和CRUSH 配置

Pool, PG and CRUSH Config Reference

When you create pools and set the number of placement groups for the pool, Ceph uses default values when you don't specifically override the defaults. We recommend overriding some of the defaults. Specifically, we recommend setting a pool's replica size and overriding the default number of placement groups. You can specifically set these values when running `pool` commands. You can also override the defaults by adding new ones in the `[global]` section of your Ceph configuration file.

当你创建存储池并给它分配归置组数量时，如果你没指定 `ceph` 就用默认值。我们建议更改某些默认值，特别是存储池的副本数和默认归置组数量，可以在运行 `pool` 命令的时候设置这些值。你也可以把新默认值写入 `ceph` 配置文件的 `[global]` 段。

```
[global]

# By default, Ceph makes 2 replicas of objects. If you want to make three
# copies of an object the default value--a primary copy and two replica
# copies--reset the default values as shown in 'osd pool default size'.
# If you want to allow Ceph to write a lesser number of copies in a degraded
# state, set 'osd pool default min size' to a number less than the
# 'osd pool default size' value.

osd pool default size = 3 # Write an object 3 times.
osd pool default min size = 1 # Allow writing one copy in a degraded state.

# Ensure you have a realistic number of placement groups. We recommend
# approximately 100 per OSD. E.g., total number of OSDs multiplied by 100
# divided by the number of replicas (i.e., osd pool default size). So for
# 10 OSDs and osd pool default size = 3, we'd recommend approximately
# (100 * 10) / 3 = 333.

osd pool default pg num = 333
osd pool default ppg num = 333
```

mon max pool pg num

Description: The maximum number of placement groups per pool.
每个存储的最大归置组数量。

Type: Integer
Default: `65536`

mon pg create interval

Description: Number of seconds between PG creation in the same OSD.
在同一个 OSD 里创建 PG 的间隔秒数。

Type: Float
Default: `30.0`

mon pg stuck threshold

Description: Number of seconds after which PGs can be considered as being stuck.
多长时间无响应的 PG 才认为它卡住了。

Type: 32-bit Integer
Default: `300`

osd pg bits

Description: Placement group bits per OSD.
每个 OSD 的归置组位数。

Type: 32-bit Integer
Default: `6`

osd ppg bits

Description: The number of bits per OSD for PGPs.

每个 OSD 为 PGP 留的位数。

Type: 32-bit Integer

Default: 6

osd crush chooseleaf type

Description: The bucket type to use for `chooseleaf` in a CRUSH rule. Uses ordinal rank rather than name.

在一个 CRUSH 规则内用于 `chooseleaf` 的桶类型。用序列号而不是名字。

Type: 32-bit Integer

Default: 1. Typically a host containing one or more OSDs.

1, 通常一台主机包含一或多个 OSD。

osd min rep

Description: The minimum number of replicas for a ruleset.

一规则集的最小副本数。

Type: 32-bit Integer

Default: 1

osd max rep

Description: The maximum number of replicas for a ruleset.

一规则集的最大副本数。

Type: 32-bit Integer

Default: 10

osd pool default crush rule

Description: The default CRUSH ruleset to use when creating a pool.

创建一存储池时使用的默认 CRUSH 规则。

Type: 32-bit Integer

Default: 0

osd pool default size

Description: Sets the number of replicas for objects in the pool. The default value is the same as `ceph osd pool set {pool-name} size {size}`.

设置一存储池的对象副本数， 默认值等同于 `ceph osd pool set {pool-name} size {size}`

Type: 32-bit Integer

Default: 2

osd pool default min size

Description: Sets the minimum number of written replicas for objects in the pool in order to acknowledge a write operation to the client. If minimum is not met, Ceph will not acknowledge the write to the client. This setting ensures a minimum number of replicas when operating in `degraded` mode.

设置存储池中已写副本的最小数量， 以向客户端确认写操作。如果未达到最小值， ceph 就不会向客户端发送写确认。此选项确保了降级 (`degraded`) 模式下的最小副本数。

Type: 32-bit Integer

Default: 0, which means no particular minimum. If 0, minimum is `size - (size / 2)`.

0, 意思是没有最小值。如果为 0, 最小值是 `size - (size / 2)`

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`.

一个存储池的默认归置组数量， 默认值即是 `mkpool` 的 `pg_num` 参数。

Type: 32-bit Integer

Default: 8

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).

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。

Type: 32-bit Integer
Default: 8

osd pool default flags

Description: The default flags for new pools.
新存储池的默认标志。

Type: 32-bit Integer
Default: 0

osd max pgls

Description: The maximum number of placement groups to list. A client requesting a large number can tie up the OSD.
将列出的最大归置组数量，一客户端请求量大时会影响 OSD。

Type: Unsigned 64-bit Integer
Default: 1024
Note: Default should be fine.
默认值应该没问题。

osd min pg log entries

Description: The minimum number of placement group logs to maintain when trimming log files.
清理日志文件的时候保留的归置组日志量。

Type: 32-bit Int Unsigned
Default: 1000

osd default data pool replay window

Description: The time (in seconds) for an OSD to wait for a client to replay a request.
— OSD 等待客户端重播请求的时间，秒。

Type: 32-bit Integer
Default: 45

3.1.11 消息

Messaging

ms tcp nodelay

Description: Disables nagle's algorithm on messenger tcp sessions.
禁用 nagle 算法，是关于 TCP 会话传递的。

Type: Boolean
Required: No
Default: true

ms initial backoff

Description: The initial time to wait before reconnecting on a fault.
出错时重连的初始等待时间。

Type: Double
Required: No
Default: .2

ms max backoff

Description: The maximum time to wait before reconnecting on a fault.
出错重连时等待的最大时间。

Type: Double
Required: No
Default: 15.0

ms nocrc

Description: Disables crc on network messages. May increase performance if cpu limited.
禁用网络消息的 crc 校验，CPU 不足时可提升性能。

Type: Boolean
Required: No

Default: `false`

`ms die on bad msg`

Description: Debug option; do not configure.
调试选项，不要配置。

Type: Boolean

Required: No

Default: `false`

`ms dispatch throttle bytes`

Description: Throttles total size of messages waiting to be dispatched.
等着传送的消息尺寸阀值。

Type: 64-bit Unsigned Integer

Required: No

Default: `100 << 20`

`ms bind ipv6`

Description: Enable if you want your daemons to bind to IPv6 address instead of IPv4 ones. (Not required if you specify a daemon or cluster IP)
如果想让守护进程绑定到 IPv6 地址而非 IPv4 就得启用（如果你指定了守护进程或集群 IP 就不必要了）

Type: Boolean

Required: No

Default: `false`

`ms rwthread stack bytes`

Description: Debug option for stack size; do not configure.
堆栈尺寸调试选项，不要配置。

Type: 64-bit Unsigned Integer

Required: No

Default: `1024 << 10`

`ms tcp read timeout`

Description: Controls how long (in seconds) the messenger will wait before closing an idle connection.
控制信差关闭空闲连接前的等待秒数。

Type: 64-bit Unsigned Integer

Required: No

Default: `900`

`ms inject socket failures`

Description: Debug option; do not configure.
调试选项，别配置。

Type: 64-bit Unsigned Integer

Required: No

Default: `0`

3.1.12 常规配置

General config reference

`fsid`

Description: The filesystem ID. One per cluster.
文件系统 ID，每集群一个。

Type: UUID

Required: No.

Default: N/A. Generated if not specified.

`admin socket`

Description: The socket for executing administrative commands irrespective of whether Ceph monitors have established a quorum.

执行管理命令的套接字，不管 ceph 监视器团体是否已经建立。

Type: String

Required: No

Default: /var/run/ceph/\$cluster-\$name.asok

pid file

Description: Each running Ceph daemon has a running process identifier (PID) file.
每个运行的 ceph 进程都有一个 PID 文件。

Type: String

Required: No

Default: N/A. The default path is /var/run/\$cluster/\$name.pid. The PID file is generated upon start-up.

chdir

Description: The directory Ceph daemons change to once they are up and running. Default / directory recommended.
ceph 进程一旦启动、运行就进入这个目录，默认推荐/。

Type: String

Required: No

Default: /

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.
如果设置了，ceph 服务启动的时候会在操作系统级设置 max open fds（如最大数量的文件描述符），这有助于防止耗尽文件描述符。

Type: 64-bit Integer

Required: No

Default: 0

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
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。

Type: Boolean

Default: true

3.2 ceph 部署

Ceph deployment

The `ceph-deploy` tool is a way to deploy Ceph relying only upon SSH access to the servers, `sudo`, and some Python. It runs on your workstation, and does not require servers, databases, or any other tools. If you set up and tear down Ceph clusters a lot, and want minimal extra bureaucracy, `ceph-deploy` is an ideal tool. The `ceph-deploy` tool is not a generic deployment system. It was designed exclusively for Ceph users who want to get Ceph up and running quickly with sensible initial configuration settings without the overhead of installing Chef, Puppet or Juju. Users who want fine-control over security settings, partitions or directory locations should use a tool such as Juju, Puppet, [Chef](#) or Crowbar.

`ceph-deploy` 工具是一种部署 ceph 的方法，它只依赖到服务器的 SSH 访问、`sudo`、和 python。它可在你的工作站上运行，不需要服务器、数据库、或其它工具。如果你安装、拆卸过很多 ceph 集群，不想要额外的工具，那 `ceph-deploy` 是理想之选。它不是个通用部署系统，只为 ceph 用户设计，用它可以快速地设置并运行一个默认值较合理的集群，而无需头疼 chef、puppet 或 juju 的安装。如果您想良好地控制安全设置、分区、或目录位置，可以试试类似 juju、chef、crowbar 的工具。

ceph 部署

Ceph Deploy

With `ceph-deploy`, you can install Ceph packages on remote hosts, create a cluster, add monitors, gather (or forget) keys, add metadata servers and OSDs, configure admin hosts, and tear down the clusters. With a

single tool, you can develop scripts to create, deploy and tear down clusters quickly and easily.

用 `ceph-deploy` 可以远程安装 ceph 软件包、创建集群、增加监视器、收集（或丢弃）密钥、增加元数据服务器和 OSD、配置管理主机、拆除集群。只用这个工具你就可以快速、轻易地开发脚本来创建、部署和拆除集群。

mkcephfs (已过时)

The `mkcephfs` utility generates an `fsid` and keys for your cluster, as defined by the Ceph configuration file. It does not create directories for you and relies on use of the `root` password. As of Ceph v0.60, it is deprecated in favor of `ceph-deploy`.

`mkcephfs` 工具可为你生成一个 `fsid` 和密钥，正如配置文件里所定义。它没给你创建目录，且依赖 `root` 密码，从 v0.60 起它被 `ceph-deploy` 取代了。

3.2.1 转移到 ceph-deploy

Transitioning to ceph-deploy

If you have an existing cluster that you deployed with `mkcephfs`, you will need to make a few changes to your configuration to ensure that your cluster will work with `ceph-deploy`.

如果你在用的集群是用 `mkcephfs` 部署的，那得稍微改动下配置文件才能和 `ceph-deploy` 配合。

3.2.1.1 监视器密钥环

Monitor Keyring

You will need to add `caps mon = "allow *"` to your monitor keyring if it is not already in the keyring. By default, the monitor keyring is located under `/var/lib/ceph/mon/ceph-$id/keyring`. When you have added the `caps` setting, your monitor keyring should look something like this:

如果你的监视器密钥环里没有 `caps mon = "allow *"`，那就把它加上。监视器的密钥环默认位于 `/var/lib/ceph/mon/ceph-$id/keyring`，加上 `caps` 选项后你的监视器密钥环看起来大致如此：

```
[mon.]  
key = AQBJIHhRuHCwDRAAZjBTSJcIBIOGpdOR9ToiyQ==  
caps mon = "allow *"
```

Adding `caps mon = "allow *"` will ease the transition from `mkcephfs` to `ceph-deploy` by allowing `ceph-create-keys` to use the `mon.` keyring file in `$mon_data` and get the caps it needs.

增加 `caps mon = "allow *"` 允许了 `ceph-create-keys` 使用位于 `$mon_data` 下的 `mon.` 密钥环文件，并获取需要的能力，因此简化了 `mkcephfs` 到 `ceph-deploy` 的转移。

3.2.1.2 用默认路径

Use Default Paths

Under the `/var/lib/ceph` directory, the `mon` and `osd` directories need to use the default paths.

在 `/var/lib/ceph` 下，`mon` 和 `osd` 目录要使用默认路径。

- **OSDs:** The path should be `/var/lib/ceph/osd/ceph-$id`

OSDs: 路径应该是 `/var/lib/ceph/osd/ceph-$id`

- **MON:** The path should be `/var/lib/ceph/mon/ceph-$id`

MON: 路径应该是 `/var/lib/ceph/mon/ceph-$id`

Under those directories, the keyring should be in a file named `keyring`.

在这些目录下，密钥环应该位于名为 `keyring` 的文件内。

3.2.2 飞前检查

Preflight Checklist

New in version 0.60.

0.60 版新增。

This **Preflight Checklist** will help you prepare an admin node for use with `ceph-deploy`, and server nodes for use with passwordless `ssh` and `sudo`.

Before you can deploy Ceph using `ceph-deploy`, you need to ensure that you have a few things set up first on your admin node and on nodes running Ceph daemons.

这篇飞前检查帮你准备一个运行 `ceph-deploy` 的管理节点，以及多个无密码 `ssh` 和 `sudo` 登录的服务器节点。

在能用 `ceph-deploy` 部署 ceph 前，你得先确认位于管理节点和 ceph 守护进程运行节点上的一些配置。

3.2.2.1 选装一个操作系统

Install an Operating System

Install a recent release of Debian or Ubuntu (e.g., 12.04, 12.10) on your nodes. For additional details on operating systems or to use other operating systems other than Debian or Ubuntu, see [OS Recommendations](#).

安装一个最新版的 Debian 或 Ubuntu (如 12.04、12.10)，关于 Debian 或 Ubuntu 以外的操作系统详见 [OS Recommendations](#)。

3.2.2.2 安装 SSH 服务器

Install an SSH Server

The `ceph-deploy` utility requires `ssh`, so your server node(s) require an SSH server.

`ceph-deploy` 工具需要 `ssh`，所以你的服务器节点需要 `ssh` 服务。

```
sudo apt-get install openssh-server
```

3.2.2.3 创建一用户

Create a User

Create a user on nodes running Ceph daemons.

在运行 ceph 守护进程的节点上创建一个普通用户。

Tip: We recommend a username that brute force attackers won't guess easily (e.g., something other than `root`, `ceph`, etc).

提示：我们建议选一个暴力攻击者不可能轻易猜到的用户名，如 `root`、`ceph` 之外的名字。

```
ssh user@ceph-server
sudo useradd -d /home/ceph -m ceph
sudo passwd ceph
```

`ceph-deploy` installs packages onto your nodes. This means that the user you create requires passwordless `sudo` privileges.

`ceph-deploy` 会在节点安装软件包，所以你创建的用户需要无密码 `sudo` 权限。

Note: We **DO NOT** recommend enabling the `root` password for security reasons.

注意：安全起见，我们不推荐启用 `root` 密码。

To provide full privileges to the user, add the following to `/etc/sudoers.d/ceph`.

为赋予用户所有权限，把下列加入 `/etc/sudoers.d/ceph`。

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

3.2.2.4 配置 SSH

Configure SSH

Configure your admin machine with password-less SSH access to each node running Ceph daemons (leave

the passphrase empty).

配置你的管理主机，使之可通过 SSH 无密码访问各节点，口令留空。

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

Copy the key to each node running Ceph daemons:

把公钥拷贝到各节点：

```
ssh-copy-id ceph@ceph-server
```

Modify your `~/.ssh/config` file of your admin node so that it defaults to logging in as the user you created when no username is specified.

修改你管理节点上的`~/.ssh/config`，以使未指定用户名时默认用你刚刚创建的用户名。

```
Host ceph-server
  Hostname ceph-server.fqdn-or-ip-address.com
  User ceph
```

3.2.2.5 安装 ceph-deploy

Install ceph-deploy

To install `ceph-deploy`, execute the following:

执行下列命令安装 `ceph-deploy`:

```
wget -q -O 'https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.asc' | sudo apt-key add -
echo deb http://ceph.com/debian-cuttlefish/ $(lsb_release -sc) main | sudo tee
/etc/apt/sources.list.d/ceph.list
sudo apt-get update
sudo apt-get install ceph-deploy
```

3.2.2.6 确认连通性

Ensure Connectivity

Ensure that your Admin node has connectivity to the network and to your Server node (e.g., ensure `iptables`, `ufw` or other tools that may prevent connections, traffic forwarding, etc. to allow what you need).

确保你的管理节点网络连接正常、且能连接服务器节点（如确认 `iptables`、`ufw` 或其它可阻断连接的工具、流量转发等等这些组件配置正常）。

Tip: The `ceph-deploy` tool is new and you may encounter some issues without effective error messages.

提示：`ceph-deploy` 工具是全新的，所以你可能碰到一些问题，却没有用的提示。

Once you have completed this pre-flight checklist, you are ready to begin using `ceph-deploy`.

完成飞前检查后，你就可以开始用 `ceph-deploy` 部署了。

3.2.3 软件包管理

Package Management

3.2.3.1 安装

Install

To install Ceph packages on your cluster hosts, open a command line on your client machine and type the following:

要在集群主机上安装 `ceph` 软件包，在管理主机上打开命令行并执行下列命令：

```
ceph-deploy install {hostname [hostname] ...}
```

Without additional arguments, `ceph-deploy` will install the most recent stable Ceph package to the cluster host(s). To specify a particular package, you may select from the following:

没提供额外选项的话 `ceph-deploy` 默认会把最新稳定版安装到集群主机，要指定某个软件包可以用下列参数：

- `--stable <code-name>`
- `--testing`
- `--dev <branch-or-tag>`

For example:

例如：

```
ceph-deploy install --stable cuttlefish hostname1  
ceph-deploy install --testing hostname2  
ceph-deploy install --dev wip-some-branch hostname{1,2,3,4,5}
```

For additional details, see [Installing Debian/Ubuntu Packages](#). For additional usage, execute:

其他详情见 [Installing Debian/Ubuntu Packages](#)。其他用法见：

```
ceph-deploy install -h
```

3.2.3.2 卸载

Uninstall

To uninstall Ceph packages from your cluster hosts, open a terminal on your admin host and type the following:

要从集群主机卸载 ceph 软件包，在管理主机的终端下执行：

```
ceph-deploy uninstall {hostname [hostname] ...}
```

On a Debian or Ubuntu system, you may also:

在 Debian 或 Ubuntu 系统上你也可以：

```
ceph-deploy purge {hostname [hostname] ...}
```

The tool will unininstall ceph packages from the specified hosts. Purge additionally removes configuration files.

此工具会从指定主机上卸载 ceph 软件包，另外 `purge` 会删除配置文件。

3.2.4 创建一集群

Create a Cluster

The first step in using Ceph with `ceph-deploy` is to create a new Ceph cluster. A new Ceph cluster has:

使用 `ceph-deploy` 的第一步就是新建一个集群，新集群具备：

- A Ceph configuration file, and
一个 ceph 配置文件，以及
- A monitor keyring.
一个监视器密钥环。

The Ceph configuration file consists of at least:

ceph 配置文件至少要包含：

- Its own filesystem ID (`fsid`)
它自己的文件系统 ID (`fsid`)
- The initial monitor(s) hostname(s), and
最初的监视器（们）及其主机名（们），以及
- The initial monitor(s) and IP address(es).

最初的监视器及其 IP 地址。

For additional details, see the [Monitor Configuration Reference](#).

The `ceph-deploy` tool also creates a monitor keyring and populates it with a `[mon.]` key. For additional details, see the [Cephx Guide](#).

详情见 [Monitor Configuration Reference](#)。

`ceph-deploy` 工具也创建了一个监视器密钥环并置于`[mon.]`内，详情见 [Cephx Guide](#)。

3.2.4.1 用法

Usage

To create a cluster with `ceph-deploy`, use the `new` command and specify the host(s) that will be initial members of the monitor quorum.

要用 `ceph-deploy` 创建集群，用 `new` 命令、并指定几个主机作为初始监视器法定人数。

```
ceph-deploy new {host [host], ...}
```

For example:

例如：

```
ceph-deploy new mon1.foo.com
ceph-deploy new mon{1,2,3}
```

The `ceph-deploy` utility will use DNS to resolve hostnames to IP addresses. The monitors will be named using the first component of the name (e.g., `mon1` above). It will add the specified host names to the Ceph configuration file. For additional details, execute:

`ceph-deploy` 工具会用 DNS 把主机名解析为 IP 地址。监视器将被命名为域名的第一段（如前述的 `mon1`），它会把指定主机名加入 `ceph` 配置文件。其他用法见：

```
ceph-deploy new -h
```

3.2.4.2 命名集群

Naming a Cluster

By default, Ceph clusters have a cluster name of `ceph`. You can specify a cluster name if you want to run multiple clusters on the same hardware. For example, if you want to optimize a cluster for use with block devices, and another for use with the gateway, you can run two different clusters on the same hardware if they have a different `fsid` and cluster name.

默认，`ceph` 集群的名字为 `ceph`，如果你想在同一套硬件上运行多套集群可以指定其他名字。比如，如果优化一个集群用于块设备，另一个用作网关，你可以在同一套硬件上运行两个不同的集群，但是它们要配置不同的 `fsid` 和集群名。

```
ceph-deploy --cluster {cluster-name} new {host [host], ...}
```

For example:

例如：

```
ceph-deploy --cluster rbd-cluster new ceph-mon1
ceph-deploy --cluster rbd-cluster new ceph-mon{1,2,3}
```

Note: If you run multiple clusters, ensure you adjust the default port settings and open ports for your additional cluster(s) so that the networks of the two different clusters don't conflict with each other.

注意：如果你运行多个集群，必须修改默认端口选项并为其打开端口，这样两个不同的集群网才不会相互冲突。

3.2.5 增加/删除监视器

Add/Remove Monitors

With `ceph-deploy`, adding and removing monitors is a simple task. You just add or remove one or more

monitors on the command line with one command. Before `ceph-deploy`, the process of [adding and removing monitors](#) involved numerous manual steps. Using `ceph-deploy` imposes a restriction: **you may only install one monitor per host.**

用 `ceph-deploy` 增加和删除监视器很简单，只要一个命令就可以增加或删除一或多个监视器。以前，增加和删除监视器涉及数个手动步骤。用 `ceph-deploy` 也预示着一个局限性，你只能在一主机上安装一个监视器。

Note: We do not recommend comingling monitors and OSDs on the same host.

注意：我们不建议把监视器和 OSD 置于同一主机上。

For high availability, you should run a production Ceph cluster with **AT LEAST** three monitors. Ceph uses the Paxos algorithm, which requires a consensus among the majority of monitors in a quorum. You can establish a monitor quorum with only one monitor; however, you can not determine a majority with two monitors. A majority of monitors must be counted as such: 1:1, 2:3, 3:4, 3:5, 4:6, etc.

考虑到高可用性，生产集群应该至少有 3 个监视器。`ceph` 用 Paxos 算法，要求法定人数里的大多数达成一致。你可以只用一个监视器形成法定人数，然而你不能用两个监视器确定大多数。大多数监视器的比例必须像：1:1、2:3、3:4、3:5、4:6 等等。

See [Monitor Config Reference](#) for details on configuring monitors.

关于监视器的配置见 [Monitor Config Reference](#)。

3.2.5.1 增加一监视器

Add a Monitor

Once you create a cluster and install Ceph packages to the monitor host(s), you may deploy the monitor(s) to the monitor host(s). When using `ceph-deploy`, the tool enforces a single monitor per host.

创建集群并在监视器主机上安装 `ceph` 软件包后，接着部署监视器。用 `ceph-deploy` 时，它限制一主机只能装一个监视器。

```
ceph-deploy mon create {host-name [host-name]...}
```

Note: Ensure that you add monitors such that they may arrive at a consensus among a majority of monitors.

注意：确保你增加的监视器能在大多数成员中达成一致。

3.2.5.2 删除一监视器

Remove a Monitor

If you have a monitor in your cluster that you'd like to remove, you may use the `destroy` option.

如果你想删除集群中的某个监视器，可以用 `destroy` 选项。

```
ceph-deploy mon destroy {host-name [host-name]...}
```

Note: Ensure that if you remove a monitor, the remaining monitors will be able to establish a consensus. If that is not possible, consider adding a monitor before removing the monitor you would like to take offline.

注意：确保你删除一监视器后，其余监视器仍能达成一致。如果不可能，删除它之前先加一个。

3.2.6 密钥管理

Keys Management

3.2.6.1 收集密钥

Gather Keys

Before you can provision a host to run OSDs or metadata servers, you must gather monitor keys and the OSD and MDS bootstrap keyrings. To gather keys, enter the following:

在准备一台主机作为 OSD 或元数据服务器时，你得收集监视器、OSD、和 MDS 的初始密钥环，可用下列命令：

```
ceph-deploy gatherkeys {monitor-host}
```

Note: To retrieve the keys, you specify a host that has a Ceph monitor.

注意：检索密钥时，指定一个包含 ceph 监视器的主机。

3.2.6.2 忘记密钥

Forget Keys

When you are no longer using `ceph-deploy` (or if you are recreating a cluster), you should delete the keys in the local directory of your admin host. To delete keys, enter the following:

不再使用 `ceph-deploy` (或另建一集群) 时，你应该删除管理主机上、本地目录中的密钥。可用下列命令：

```
ceph-deploy forgetkeys
```

3.2.7 增加/删除 OSD

Add/Remove OSDs

Adding and removing Ceph OSD Daemons to your cluster may involve a few more steps when compared to adding and removing other Ceph daemons. Ceph OSD Daemons write data to the disk and to journals. So you need to provide a disk for the OSD and a path to the journal partition (i.e., this is the most common configuration, but you may configure your system to your own needs).

新增和拆除 ceph 的 OSD 进程相比另二种要多几步。OSD 守护进程要向磁盘写数据和日志，所以你得相应地提供一硬盘和日志分区（这是最常见的配置，但你可以按需配置）。

By default, `ceph-deploy` will create an OSD with the XFS filesystem. You may override the filesystem type by providing a `--fs-type FS_TYPE` argument, where `FS_TYPE` is an alternate filesystem such as `ext4` or `btrfs`.

默认情况下，`ceph-deploy` 会创建基于 XFS 文件系统的 OSD，但你可以用 `--fs-type FS_TYPE` 覆盖默认值，其中 `FS_TYPE` 是别种文件系统，如 `ext4` 或 `btrfs`。

In Ceph v0.60 and later releases, Ceph supports `dm-crypt` on disk encryption. You may specify the `--dm-crypt` argument when preparing an OSD to tell `ceph-deploy` that you want to use encryption. You may also specify the `--dmcrypt-key-dir` argument to specify the location of `dm-crypt` encryption keys.

在 ceph-v0.60 及其后续发布，`ceph` 可用 `dm-crypt` 加密，在准备 OSD 时你可以用 `--dm-crypt` 参数告诉 `ceph-deploy` 你想用加密功能。也可以用 `--dmcrypt-key-dir` 参数指定 `dm-crypt` 加密密钥的位置。

You should test various drive configurations to gauge their throughput before building out a large cluster. See [Data Storage](#) for additional details.

在构建一个大型集群前，你应该测试各种驱动器配置来衡量其吞吐量。详情见 [Data Storage](#)。

3.2.7.1 列举磁盘

List Disks

To list the disks on a node, execute the following command:

执行下列命令列举一节点上的磁盘：

```
ceph-deploy disk list {node-name} [node-name]...
```

3.2.7.2 擦净磁盘

Zap Disks

To zap a disk (delete its partition table) in preparation for use with Ceph, execute the following:

用下列命令擦净（删除分区表）磁盘，以用于 ceph：

```
ceph-deploy disk zap {osd-server-name}:{disk-name}
ceph-deploy disk zap osdserver1:sdb
```

Important: This will delete all data.

重要：这会删除所有数据。

3.2.7.3 准备 OSD

Prepare OSDs

Once you create a cluster, install Ceph packages, and gather keys, you may prepare the OSDs and deploy them to the OSD node(s). If you need to identify a disk or zap it prior to preparing it for use as an OSD, see [List Disks](#) and [Zap Disks](#).

创建集群、安装 ceph 软件包、收集密钥完成后你就可以准备 OSD 并把它们部署到 OSD 节点了。如果你想确认某磁盘或擦净它，参见 [List Disks](#) 和 [Zap Disks](#)。

```
ceph-deploy osd prepare {node-name}:{disk}[:{path/to/journal}]  
ceph-deploy osd prepare osdserver1:sdb:/dev/ssd1
```

The `prepare` command only prepares the OSD. It does not activate it. To activate a prepared OSD, use the `activate` command. See [Activate OSDs](#) for details.

`prepare` 命令只准备 OSD，不会激活它。要激活一个已准备好的 OSD，用 `activate` 命令，见 [Activate OSDs](#)。

The foregoing example assumes a disk dedicated to one Ceph OSD Daemon, and a path to an SSD journal partition. We recommend storing the journal on a separate drive to maximize throughput. You may dedicate a single drive for the journal too (which may be expensive) or place the journal on the same disk as the OSD (not recommended as it impairs performance). In the foregoing example we store the journal on a partitioned solid state drive.

前例假定一个硬盘只会用于一个 OSD 守护进程，以及一个到 SSD 日志分区的路径。我们建议把日志存储于另外的驱动器以最优化性能；你也可以指定一单独的驱动器用于日志（也许比较昂贵）、或者把日志放到 OSD 数据盘（不建议，因为它损伤性能）。前例中我们把日志存储于已分区的固态硬盘。

Note: When running multiple Ceph OSD daemons on a single node, and sharing a partitioned journal with each OSD daemon, you should consider the entire node the minimum failure domain for CRUSH purposes, because if the SSD drive fails, all of the Ceph OSD daemons that journal to it will fail too.

注意：在一个节点运行多个 OSD 守护进程、且多个 OSD 守护进程共享一个日志分区时，你应该考虑整个节点的最小 CRUSH 故障域，因为如果这个 SSD 坏了，所有用其做日志的 OSD 守护进程也会失效。

3.2.7.4 激活 OSD

Activate OSDs

Once you prepare an OSD you may activate it with the following command.

准备好 OSD 后，可以用下列命令激活它。

```
ceph-deploy osd activate {node-name}:{path/to/disk}[:{path/to/journal}]  
ceph-deploy osd activate osdserver1:/dev/sdb1:/dev/ssd1
```

The `activate` command will cause your OSD to come up and be placed in the cluster. The `activate` command uses the path to the partition created when running the `prepare` command.

`activate` 命令会让 OSD 进入 `up` 且 `in` 状态，此命令所用路径和 `prepare` 相同。

3.2.7.5 创建 OSD

Create OSDs

You may prepare OSDs, deploy them to the OSD node(s) and activate them in one step with the `create` command. The `create` command is a convenience method for executing the `prepare` and `activate` command sequentially.

你可以用 `create` 命令一次完成准备 OSD、部署到 OSD 节点、并激活它。`create` 命令是依次执行 `prepare` 和 `activate` 命令的捷径。

```
ceph-deploy osd create {node-name}:{disk}[:{path/to/journal}]  
ceph-deploy osd create osdserver1:sdb:/dev/ssd1
```

3.2.7.6 拆除 OSD

Destroy OSDs

Note: Coming soon. See [Remove OSDs](#) for manual procedures.

注意：稍后完成。手动过程见 [Remove OSDs](#)。

3.2.8 增加/拆除元数据服务器

Add/Remove Metadata Server

With `ceph-deploy`, adding and removing metadata servers is a simple task. You just add or remove one or more metadata servers on the command line with one command.

用 `ceph-deploy` 增加和拆除元数据服务器很简单，只要一个命令就可以增加或拆除一或多个元数据服务器。

Note: CephFS is in production using 1 metadata server per cluster. You **MUST** deploy at least one metadata server to use CephFS.

注意：生产环境下，每个 CephFS 集群只用一个元数据服务器，但你必须至少有一个才能用 CephFS。

See [MDS Config Reference](#) for details on configuring metadata servers.

元数据服务器配置见 [MDS Config Reference](#)。

3.2.8.1 增加一元数据服务器

Add a Metadata Server

Once you deploy monitors and OSDs you may deploy the metadata server(s).

部署完监视器和 OSD 后，还可以部署元数据服务器。

```
ceph-deploy mds create {host-name}[:{daemon-name}] [{host-name}[:{daemon-name}]] ...
```

You may specify a daemon instance a name (optional) if you would like to run multiple daemons on a single server.

如果你想在同一主机上运行多个守护进程，可以为每个进程指定名字（可选）。

3.2.8.2 删 除一元数据服务器

Remove a Metadata Server

Coming soon...

3.2.9 清除一主机

Purge a Host

When you remove Ceph daemons and uninstall Ceph, there may still be extraneous data from the cluster on your server. The `purge` and `purgedata` commands provide a convenient means of cleaning up a host.

拆除 `ceph` 守护进程并卸载软件包后，此主机上可能还有集群中的数据，`purge` 和 `purgedata` 命令提供了一种清理主机的简便方法。

3.2.9.1 清除数据

Purge Data

To remove all data from `/var/lib/ceph` (but leave Ceph packages intact), execute the `purgedata` command.

要清理掉 `/var/lib/ceph` 下的所有数据（但保留 `ceph` 软件包），执行 `purgedata` 命令：

```
ceph-deploy purgedata {hostname} [{hostname} ...]
```

3.2.9.2 Purge

To remove all data from `/var/lib/ceph` and uninstall Ceph packages, execute the `purge` command.

要清理掉`/var/lib/ceph`下的所有数据、并卸载ceph软件包，用`purge`命令。

```
ceph-deploy purge {hostname} [{hostname} ...]
```

3.2.10 管理任务

Admin Tasks

Once you have set up a cluster with `ceph-deploy`, you may provide the client admin key and the Ceph configuration file to another host so that a user on the host may use the `ceph` command line as an administrative user.

用`ceph-deploy`建立一个集群后，你可以把客户端管理密钥和`ceph`配置文件发给其他管理员，以便让他用`ceph`命令管理集群。

3.2.10.1 创建一管理主机

Create an Admin Host

To enable a host to execute `ceph` commands with administrator privileges, use the `admin` command.

要允许一主机以管理员权限执行`ceph`命令，用`admin`命令：

```
ceph-deploy admin {host-name [host-name]...}
```

3.2.10.2 分发配置文件

Deploy Config File

To send an updated copy of the Ceph configuration file to hosts in your cluster, use the `config push` command.

要把改过的配置文件分发给集群内各主机，可用`config push`命令。

```
ceph-deploy config push {host-name [host-name]...}
```

Tip: With a base name and increment host-naming convention, it is easy to deploy configuration files via simple scripts (e.g., `ceph-deploy config hostname{1,2,3,4,5}`).

提示：用相同前缀和递增数字命名的话，部署可以通过一个命令轻易完成，如`ceph-deploy config hostname{1,2,3,4,5}`

3.2.10.3 检索配置文件

Retrieve Config File

To retrieve a copy of the Ceph configuration file from a host in your cluster, use the `config pull` command.

要从集群内的一主机检索配置文件，用`config pull`命令。

```
ceph-deploy config pull {host-name [host-name]...}
```

3.2.11 用mkcephfs部署（已过时）

Deploying with mkcephfs

The `mkcephfs` tool is the old method of deploying new Ceph clusters. It is now deprecated in favor of `ceph-deploy`, which has better support for modifying an existing cluster.

`mkcephfs`是以前的`ceph`集群部署方法，现在淘汰了，转移到了`ceph-deploy`，它能更好地修改现有集群。

3.2.11.1 允许root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。

```
sudo -i  
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.
```

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
```

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
```

3.2.11.2 把配置文件复制到所有节点

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，所以你得手动把配置文件（如下例中的 `mycluster.conf`）复制到 OSD 主机的 `/etc/ceph/ceph.conf`。

```
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
```

3.2.11.3 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
```

3.2.11.4 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). When mounting disks in this manner, there is no need for an entry in `/etc/fstab`.

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。用这种方式挂载硬盘时，`/etc/fstab` 里的条目无所谓。

New in version 0.56.

For Bobtail (v 0.56) and beyond, you may specify the file system type, filesystem options, and mount options. Add the following to the `[global]` section of your Ceph configuration file, and replace the values in braces with appropriate values:

bobtail (0.56) 版之后，你可以指定文件系统类型、创建选项、和挂载选项。把下列这段加到配置文件的 `[global]` 段下，并替换掉花括号里的内容：

```
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:

例如：

```
osd mkfs type = btrfs
osd mkfs options btrfs = -m raid0
osd mount options btrfs = rw, noatime
```

For each `[osd.n]` section of your configuration file, specify the storage device. For example:

配置文件里的每个 `[osd.n]` 段下给指定一个存储设备，例如：

```
[osd.1]
  devs = /dev/sda
[osd.2]
  devs = /dev/sdb
```

3.2.11.5 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。

To run `mkcephfs`, execute the following:

`mkcephfs` 命令的用法如下：

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

Note: For `mkcephfs` to use the `mkfs` configuration options, you MUST specify a `devs` entry for each OSD.

注意：要让 `mkcephfs` 使用 `mkfs` 配置选项，必须给每个 OSD 指定 `devs`。

The script adds an admin key to the `ceph.keyring`, which is analogous to a root password. See Authentication when running with `cephx` enabled. To start the cluster, execute the following:

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。执行下列命令启动集群：

```
service ceph -a start
```

See [Operating a Cluster](#) for details. Also see [man mkcephfs](#).

详情参见 [man mkcephfs](#)。

3.3 运维

Cluster Operations

高级操作

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.

高级集群操作主要包括用 `ceph` 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。

- [Operating a Cluster](#)
- [Monitoring a Cluster](#)
- [Monitoring OSDs and PGs](#)
- [Authentication Overview](#)
- [Cephx Authentication](#)

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.

你的集群开始运行后，就可以尝试数据归置了。Ceph 是 PB 级数据存储集群，它用 CRUSH 算法、靠存储池和归置组在集群内分布数据。

- [Data Placement Overview](#)
- [Pools](#)
- [Placement Groups](#)
- [CRUSH Maps](#)

低级运维

Low-level Operations

Low-level cluster operations consist of starting, stopping, and restarting a particular daemon within a cluster; changing the settings of a particular daemon or subsystem; and, adding a daemon to the cluster or removing a daemon from the cluster. The most common use cases for low-level operations include growing or shrinking the Ceph cluster and replacing legacy or failed hardware with new hardware.

低级集群运维包括启动、停止、重启集群内的某个具体守护进程；更改某守护进程或子系统配置；增加或拆除守护进程。低级运维的一个常见情形包括扩展、缩减 ceph 集群，以及更换损坏硬件。

- [Adding/Removing OSDs](#)
- [Adding/Removing Monitors](#)
- [Command Reference](#)

故障排除

Troubleshooting

Ceph is still on the leading edge, so you may encounter situations that require you to evaluate your Ceph configuration and modify your logging and debugging settings to identify and remedy issues you are encountering with your cluster.

ceph 仍在积极开发中，所以你可能碰到一些问题，需要评估 ceph 配置文件、并修改日志和调试选项来纠正它。

- [The Ceph Community](#)
- [Recovering from Monitor Failures](#)
- [Troubleshooting OSDs](#)
- [Troubleshooting PGs](#)
- [Logging and Debugging](#)
- [CPU Profiling](#)
- [Memory Profiling](#)

3.3.1 操纵集群

Operating a cluster

3.3.1.1 用 upstart 控制 ceph

Running Ceph with Upstart

When deploying Ceph Cuttlefish and beyond with `ceph-deploy`, you may start and stop Ceph daemons or the entire cluster using the event-based [Upstart](#). Upstart does not require you to define daemon instances in the Ceph configuration file (although, they are still required for `sysvinit` should you choose to use it).

用 `ceph-deploy` 部署完 ceph Cuttlefish 及更高版本后，你可以用基于事件的 [Upstart](#) 来启动、关闭整个集群。`upstart` 不要求你在配置文件里定义守护进程例程，然而 `sysvinit` 仍需要。

To list the Ceph Upstart jobs and instances, execute:

用下列命令列出 ceph 作业和例程：

```
sudo initctl list | grep ceph
```

See [initctl](#) for additional details.

详情参见 [initctl](#)。

3.3.1.1.1 启动集群

Starting a Cluster

To start the cluster, execute the following:

要启动集群，用下列命令：

```
sudo start ceph-all
```

3.3.1.1.2 关闭集群

Stopping a Cluster

To stop the cluster, execute the following:

要停止集群，用下列命令：

```
sudo stop ceph-all
```

3.3.1.1.3 启动一类进程

Starting all Daemons by Type

To start all daemons of a particular type, execute one of the following:

要启动某一类守护进程，用下列命令：

```
sudo start ceph-osd-all  
sudo start ceph-mon-all  
sudo start ceph-mds-all
```

3.3.1.1.4 停止一类进程

Stopping all Daemons by Type

To stop all daemons of a particular type, execute one of the following:

要停止某一类守护进程，用下列命令：

```
sudo stop ceph-osd-all  
sudo stop ceph-mon-all  
sudo stop ceph-mds-all
```

3.3.1.1.5 启动一个进程

Starting a Daemon

To start a specific daemon instance, execute one of the following:

要启动一指定守护进程例程，用下列命令之一：

```
sudo start ceph-osd id={id}  
sudo start ceph-mon id={hostname}  
sudo start ceph-mds id={hostname}
```

For example:

例如：

```
sudo start ceph-osd id=1  
sudo start ceph-mon id=ceph-server  
sudo start ceph-mds id=ceph-server
```

3.3.1.1.6 停止一个进程

Stopping a Daemon

To stop a specific daemon instance, execute one of the following:

要停止一指定守护进程例程，用下列命令之一：

```
sudo stop ceph-osd id={id}  
sudo stop ceph-mon id={hostname}  
sudo stop ceph-mds id={hostname}
```

For example:

例如：

```
sudo stop ceph-osd id=1  
sudo start ceph-mon id=ceph-server  
sudo start ceph-mds id=ceph-server
```

3.3.1.2 以服务运行ceph

Running Ceph as a Service

When you deploy Ceph Argonaut or Bobtail with `mkcephfs`, use the service or traditional sysvinit.

你用 `mkcephfs` 部署了 ceph 的 Argonaut 或 Bobtail 版时，用服务或传统的 sysvinit（控制 ceph）。

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 集群的功能，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
<code>--verbose</code>	<code>-v</code>	Use verbose logging. 详细的日志。
<code>--valgrind</code>	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。
<code>--allhosts</code>	<code>-a</code>	Execute on all hosts in <code>ceph.conf</code> . Otherwise, it only executes on <code>localhost</code> . 在 <code>ceph.conf</code> 里配置的所有主机上执行，否则它只在本机执行。
<code>--restart</code>	N/A	Automatically restart daemon if it core dumps. 核心转储后自动重启。
<code>--norestart</code>	N/A	Don't restart a daemon if it core dumps. 核心转储后不自动重启。
<code>--conf</code>	<code>-c</code>	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
<code>start</code>	Start the daemon(s).
<code>stop</code>	Stop the daemon(s).
<code>forcestop</code>	Force the daemon(s) to stop. Same as <code>kill -9</code>
<code>killall</code>	Kill all daemons of a particular type.
<code>cleanlogs</code>	Cleans out the log directory.
<code>cleanalllogs</code>	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：

To start a Ceph daemon on the local **Ceph Node**, use the following syntax:

要启动本地 ceph 节点上的守护进程，用下列语法：

```
sudo /etc/init.d/ceph start osd.0
```

To start a Ceph daemon on another node, use the following syntax:

要启动另外一节点上的 ceph 守护进程，依下列语法：

```
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。

3.3.1.2.1 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 (i.e., execute on all nodes), Ceph should begin operating. You may also specify a particular daemon instance to constrain the command to a single instance. To start a Ceph daemon on the local Ceph Node, use the following syntax:

使用 -a (在所有节点上执行) 选项可操作整个集群，你也可以指定一个具体例程把操作限定到某一个例程，例如要启动本地节点的一个例程，按下列语法：

```
sudo /etc/init.d/ceph start osd.0
```

To start a Ceph daemon on another node, use the following syntax:

要启动其他节点上的一个例程，按下列语法：

```
sudo /etc/init.d/ceph -a start osd.0
```

3.3.1.2.2 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 ceph -a 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
```

Once you execute with -a (i.e., execute on all nodes), Ceph should shut down. You may also specify a particular daemon instance to constrain the command to a single instance. To stop a Ceph daemon on the local Ceph Node, use the following syntax:

执行命令时一旦加了 -a (意为在所有节点执行)，整个 ceph 集群都会关闭。你也可以加选项把操作限定到某个具体例程。要停止本地节点上的一个守护进程，按下列语法：

```
sudo /etc/init.d/ceph stop osd.0
```

To stop a Ceph daemon on another node, use the following syntax:

要停止别处守护进程，按下列语法：

```
sudo /etc/init.d/ceph -a stop osd.0
```

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

Interactive Mode

To run the ceph tool in interactive mode, type ceph at the command line with no arguments. For example:

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

Watching a Cluster

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 检查集群状态

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

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 状态为 `up` 且 `in`：

```
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		mainrack		
-2		3		host	osd-host	
0		1			osd.0	up 1
1		1			osd.1	up 1
2		1			osd.2	up 1

For a detailed discussion, refer to [Monitoring OSDs and Placement Groups](#).

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 active|inactive，执行下面的命令来确保元数据服务器 up 且 active：

```
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 [Monitoring OSDs and Placement Groups](#).

归置组把对象映射到 OSD，归置组的状态应该是 **active** 且 **clean**，其它的 PG 状态请参见 [Monitoring OSDs and Placement Groups](#)。

3.3.2.9 使用管理套接字

Using the Admin Socket

The Ceph admin socket allows you to query a daemon via a socket interface. By default, Ceph sockets reside under `/var/run/ceph`. To access a socket, use the following command:

`ceph` 管理套接字允许你通过 socket 接口查询守护进程。它们默认存在于 `/var/run/ceph` 下，用下列命令访问套接字：

```
ceph --admin-daemon /var/run/ceph/{socket-name}
```

To view the available admin socket commands, execute the following command:

用下列命令查看可用的管理套接字命令：

```
ceph --admin-daemon /var/run/ceph/{socket-name} help
```

The admin socket command enables you to show and set your configuration at runtime. See [Viewing a Configuration at Runtime](#) for details.

管理套接字命令允许你在运行时查看和修改配置，见 [Viewing a Configuration at Runtime](#)。

Additionally, you can set configuration values at runtime directly (i.e., the admin socket bypasses the monitor, unlike `ceph {daemon-type} tell {id} injectargs`, which relies on the monitor but doesn't require you to login directly to the host in question).

另外，你可以在运行时直接修改配置选项。管理套接字会绕过监视器，不要求你直接登录主机，不像 `ceph {daemon-type} tell {id} injectargs` 依赖监视器。

3.3.3 监控 OSD 和归置组

Monitoring OSDs and PGs

High availability and high reliability require a fault-tolerant approach to managing hardware and software issues. Ceph has no single point-of-failure, and can service requests for data in a “degraded” mode. Ceph’s [data placement](#) introduces a layer of indirection to ensure that data doesn’t bind directly to particular OSD addresses. This means that tracking down system faults requires finding the [placement group](#) and the underlying OSDs at root of the problem.

高可用性和高可靠性要求容错方法来管理软硬件。`ceph` 没有单故障点，并且能在“降级”模式下继续提供服务。其[数据归置](#)引进了一个间接层，它可保证数据不会直接绑死到某一个特定 OSD 地址，这也意味着追踪系统错误的根源得深入[归置组](#)及底层的 OSD。

Tip: A fault in one part of the cluster may prevent you from accessing a particular object, but that doesn’t mean that you can’t access other objects. When you run into a fault, don’t panic. Just follow the steps for monitoring your OSDs and placement groups. Then, begin troubleshooting.

提示：集群某一部分失效可能导致不能访问某个对象，但不会牵连其他对象。碰到这种问题时无需恐慌，只需按步骤检查 OSD 和归置组，然后排除故障。

Ceph is generally self-repairing. However, when problems persist, monitoring OSDs and placement groups will help you identify the problem.

`ceph` 通常能自己康复，然而如果故障持续存在，监控 OSD 和归置组有助于找出问题所在。

3.3.3.1 监控 OSD

Monitoring OSDs

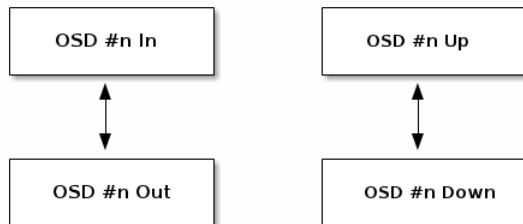
An OSD’s status is either in the cluster (`in`) or out of the cluster (`out`); and, it is either up and running (`up`), or it is down and not running (`down`). If an OSD is `up`, it may be either `in` the cluster (you can read and

write data) or it is **out** of the cluster. If it was **in** the cluster and recently moved **out** of the cluster, Ceph will migrate placement groups to other OSDs. If an OSD is **out** of the cluster, CRUSH will not assign placement groups to the OSD. If an OSD is **down**, it should also be **out**.

某 OSD 的状态可以是在集群内(**in**)或集群外(**out**)、也可以是活着且在运行(**up**)或挂了且不在运行(**down**)。如果一个 OSD 活着，它也可以是 **in** (你可以读写数据) 或者 **out** 集群。如果它以前是 **in** 但最近 **out** 了，ceph 会把其归置组迁移到其他 OSD。如果一 OSD **out** 了，CRUSH 就不会再分配归置组给它。如果它挂了 (**down**) 其状态也应该是 **out**。

Note: If an OSD is **down** and **in**, there is a problem and the cluster will not be in a healthy state.

注意：如果一 OSD 状态为 **down** 且 **in**，必定有问题，而且集群处于非健康状态。



If you execute a command such as `ceph health`, `ceph -s` or `ceph -w`, you may notice that the cluster does not always echo back **HEALTH OK**. Don't panic. With respect to OSDs, you should expect that the cluster will **NOT** echo **HEALTH OK** in a few expected circumstances:

如果你执行过这些命令，如 `ceph health`、`ceph -s`、或 `ceph -w`，也许注意到了，集群并非一直返回 **HEALTH OK**，别紧张。就 OSD 而言你应该明确，在一些情况下集群不会返回 **HEALTH OK**。

1. You haven't started the cluster yet (it won't respond).

你还没启动集群（它不会响应的）。

2. You have just started or restarted the cluster and it's not ready yet, because the placement groups are getting created and the OSDs are in the process of peering.

你刚刚启动或重启完集群，而且它还没准备好，因为归置组正被创建、OSD 们正在相互建立连接。

3. You just added or removed an OSD.

你刚刚增加或拆除一个 OSD。

4. You just have modified your cluster map.

你刚刚修改完集群运行图。

An important aspect of monitoring OSDs is to ensure that when the cluster is up and running that all OSDs that are **in** the cluster are **up** and running, too. To see if all OSDs are running, execute:

OSD 监控的一个重要事情是，当集群启动并运行时，所有 OSD 也应该是启动(**up**)并在集群内(**in**)运行。用下列命令查看：

```
ceph osd stat
```

The result should tell you the map epoch (eNNNN), the total number of OSDs (x), how many are **up** (y) and how many are **in** (z).

其结果会告诉你运行图版本(eNNNN)、OSD 总数(x)、y 个是 **up** 的、z 个是 **in** 的。

```
eNNNN: x osds: y up, z in
```

If the number of OSDs that are **in** the cluster is more than the number of OSDs that are **up**, execute the following command to identify the `ceph-osd` daemons that aren't running:

如果处于 **in** 状态的 OSD 多于 **up** 的，用下列命令看看哪些 `ceph-osd` 守护进程没在运行：

```
ceph osd tree
```

```
dumped osdmap tree epoch 1
```

#	id	weight	type	name	up/down	reweight
-1	2	2	pool	openstack		
-3	2		rack	dell-2950-rack-A		
-2	2		host	dell-2950-A1		
0	1			osd.0	up	1
1	1			osd.1	down	1

Tip: The ability to search through a well-designed CRUSH hierarchy may help you troubleshoot your cluster by identifying the physical locations faster.

提示：精心设计的CRUSH分级结构可以帮你更快的定位到物理位置、加快故障排除。

If an OSD is down, start it:

若一个OSD是down的，启动它：

```
sudo /etc/init.d/ceph -a start osd.1
```

See [OSD Not Running](#) for problems associated with OSDs that stopped, or won't restart.

和OSD没运行或不启动相关的问题请看[OSD Not Running](#)。

3.3.3.2 归置组集

PG Sets

When CRUSH assigns placement groups to OSDs, it looks at the number of replicas for the pool and assigns the placement group to OSDs such that each replica of the placement group gets assigned to a different OSD. For example, if the pool requires three replicas of a placement group, CRUSH may assign them to `osd.1`, `osd.2` and `osd.3` respectively. CRUSH actually seeks a pseudo-random placement that will take into account failure domains you set in your [CRUSH map](#), so you will rarely see placement groups assigned to nearest neighbor OSDs in a large cluster. We refer to the set of OSDs that should contain the replicas of a particular placement group as the **Acting Set**. In some cases, an OSD in the Acting Set is `down` or otherwise not able to service requests for objects in the placement group. When these situations arise, don't panic. Common examples include:

CRUSH要把归置组分配到OSD时，它先查询这个存储池的副本数设置，再把归置组分配到OSD，这样就把各副本分配到了不同OSD。比如，如果存储池要求归置组有3个副本，CRUSH可能把它们分别分配到 `osd.1`、`osd.2`、`osd.3`。考虑到你设置于CRUSH运行图中的失败域，实际上CRUSH找出的是伪随机位置，所以在大型集群中，你很少能看到归置组被分配到了相邻的OSD。我们把涉及某个特定归置组副本的一组OSD称为**acting set**。在一些情况下，位于acting set中的一个OSD `down`了或者不能为归置组内的对象提供服务，这些情形发生时无需惊慌，常见原因如下：

- You added or removed an OSD. Then, CRUSH reassigned the placement group to other OSDs—thereby changing the composition of the Acting Set and spawning the migration of data with a “backfill” process.

你增加或拆除了一OSD。然后CRUSH把那个归置组分配到了其他OSD，因此改变了Acting Set的构成、并且用backfill进程启动了数据迁移。

- An OSD was `down`, was restarted, and is now `recovering`.
 - OSD `down`了、重启了、而现在正恢复(`recovering`)。
- An OSD in the Acting Set is `down` or unable to service requests, and another OSD has temporarily assumed its duties.

acting set中的一个OSD挂了，不能提供服务，另一个OSD临时接替其工作。

Ceph processes a client request using the **Up Set**, which is the set of OSDs that will actually handle the requests. In most cases, the Up Set and the Acting Set are virtually identical. When they are not, it may indicate that Ceph is migrating data, an OSD is recovering, or that there is a problem (i.e., Ceph usually echoes a “HEALTH WARN” state with a “stuck stale” message in such scenarios).

ceph靠up set处理客户端请求，它们是最终处理请求的一系列OSD。大多数情况下up set和acting set本质上相同，如果不同，说明可能ceph在迁移数据、某OSD在恢复、或者哪里有问题。这种情况下，ceph通常表现为HEALTH WARN状态，还有“stuck stale”消息。

To retrieve a list of placement groups, execute:

用下列命令获取归置组列表：

```
ceph pg dump
```

To view which OSDs are within the Acting Set or the Up Set for a given placement group, execute:

要根据指定归置组号查看哪些 OSD 位于 Acting Set 或 Up Set 里，执行：

```
ceph pg map {pg-num}
```

The result should tell you the osdmap epoch (eNNN), the placement group number ({pg-num}), the OSDs in the Up Set (up[]), and the OSDs in the acting set (acting[]).

其结果会告诉你 osdmap 版本 (eNNN)、归置组号 ({pg-num})、Up Set 内的 OSD (up[])、和 Acting Set 内的 OSD (acting[])。

```
osdmap eNNN pg {pg-num} -> up [0,1,2] acting [0,1,2]
```

Note: If the Up Set and Acting Set do not match, this may be an indicator that the cluster rebalancing itself or of a potential problem with the cluster.

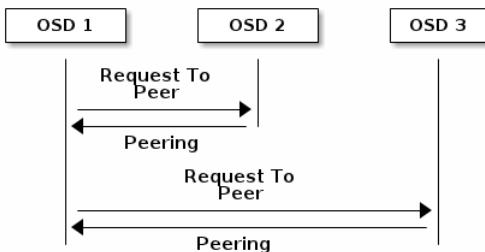
注意：如果 Up Set 和 Acting Set 不一致，这可能表明集群内部在重均衡或者有潜在问题。

3.3.3.3 节点互联

Peering

Before you can write data to a placement group, it must be in an **active** state, and it **should** be in a **clean** state. For Ceph to determine the current state of a placement group, the primary OSD of the placement group (i.e., the first OSD in the acting set), peers with the secondary and tertiary OSDs to establish agreement on the current state of the placement group (assuming a pool with 3 replicas of the PG).

写入数据前，归置组必须处于 **active**、而且应该是 **clean** 状态。假设一存储池的归置组有 3 个副本，为让 ceph 确定归置组的当前状态，一归置组的主 OSD（如 acting set 内的第一个 OSD）会与第二和第三 OSD 建立连接、并就归置组的当前状态达成一致意见。



The OSDs also report their status to the monitor. See [Configuring Monitor/OSD Interaction](#) for details. To troubleshoot peering issues, see [Peering Failure](#).

OSD 们也向监视器报告自己的状态，详情见 [Configuring Monitor/OSD Interaction](#)。要排除连接建立问题，参见 [Peering Failure](#)。

3.3.3.4 归置组状态的监控

Monitoring Placement Group States

If you execute a command such as `ceph health`, `ceph -s` or `ceph -w`, you may notice that the cluster does not always echo back **HEALTH OK**. After you check to see if the OSDs are running, you should also check placement group states. You should expect that the cluster will **NOT** echo **HEALTH OK** in a number of placement group peering-related circumstances:

如果你执行过 `ceph health`、`ceph -s`、或 `ceph -w` 命令，你也许注意到了集群并非总返回 **HEALTH OK**。检查完 OSD 是否在运行后，你还应该检查归置组状态。你应该明白，在归置组建立连接时集群不会返回 **HEALTH OK**：

1. You have just created a pool and placement groups haven't peered yet.
2. The placement groups are recovering.
3. You have just added an OSD to or removed an OSD from the cluster.
4. You have just modified your CRUSH map and your placement groups are migrating.
5. There is inconsistent data in different replicas of a placement group.

6. Ceph is scrubbing a placement group's replicas.

1. 刚刚创建了一个存储池，归置组还没建立连接；
2. 归置组正在恢复；
3. 刚刚增加或删除了一个 OSD；
4. 刚刚修改了 CRUSH 图，并且归置组正在迁移；
5. 某一归置组的副本间的数据不一致；
6. ceph 正在洗刷一个归置组的副本；

If one of the foregoing circumstances causes Ceph to echo **HEALTH_WARN**, don't panic. In many cases, the cluster will recover on its own. In some cases, you may need to take action. An important aspect of monitoring placement groups is to ensure that when the cluster is up and running that all placement groups are **active**, and preferably in the **clean** state. To see the status of all placement groups, execute:

如果是前述原因之一导致了 ceph 返回 **HEALTH_WARN**, 无需紧张。很多情况下，集群会自行恢复；有些时候你得采取些措施。归置组监控的一件重要事情是保证集群起来并运行着，所有归置组都处于 **active** 状态、并且最好是 **clean** 状态。用下列命令查看所有归置组状态：

```
ceph pg stat
```

The result should tell you the placement group map version (vNNNNNNN), the total number of placement groups (x), and how many placement groups are in a particular state such as **active+clean** (y).

其结果会告诉你归置组运行图的版本号 (vNNNNNNN)、归置组总数 x、有多少归置组处于某种特定状态，如 **active+clean** (y)。

```
vNNNNNNN: x pgs: y active+clean; z bytes data, aa MB used, bb GB / cc GB avail
```

Note: It is common for Ceph to report multiple states for placement groups.

注意：ceph 报告多种状态是常见的。

In addition to the placement group states, Ceph will also echo back the amount of data used (aa), the amount of storage capacity remaining (bb), and the total storage capacity for the placement group. These numbers can be important in a few cases:

除了归置组状态之外，ceph 也会报告数据占据的空间 (aa)、剩余空间 (bb) 和归置组总容量。这些数字在某些情况下是很重要的：

- You are reaching your **near full ratio** or **full ratio**.
快达到 **near full ratio** 或 **full ratio** 时。
- Your data isn't getting distributed across the cluster due to an error in your CRUSH configuration.
由于 CRUSH 配置错误致使你的数据没能在集群内分布。

归置组 ID

Placement Group IDs

Placement group IDs consist of the pool number (not pool name) followed by a period (.) and the placement group ID—a hexadecimal number. You can view pool numbers and their names from the output of **ceph osd lspools**. The default pool names **data**, **metadata** and **rbd** correspond to pool numbers 0, 1 and 2 respectively. A fully qualified placement group ID has the following form:

归置组 ID 包含存储池号（不是存储池名字），后面跟一个点（.），然后是归置组 ID 一个十六进制数字。用 **ceph osd lspools** 可查看存储池号及其名字，默认存储池名字 **data**、**metadata**、和 **rbd** 对应的存储池号分别是 0、1、2。完整的归置组 ID 格式如下：

```
{pool-num}.{pg-id}
```

And it typically looks like this:

典型长相：

```
0.1f
```

To retrieve a list of placement groups, execute the following:

用下列命令获取归置组列表：

```
ceph pg dump
```

You can also format the output in JSON format and save it to a file:

你也可以让它输出到 JSON 格式，并保存到文件：

```
ceph pg dump -o {filename} --format=json
```

To query a particular placement group, execute the following:

要查询某个归置组，用下列命令：

```
ceph pg {poolnum}.{pg-id} query
```

Ceph will output the query in JSON format.

ceph 会输出成 JSON 格式。

```
{  
    "state": "active+clean",  
    "up": [  
        1,  
        0  
    ],  
    "acting": [  
        1,  
        0  
    ],  
    "info": {  
        "pgid": "1.e",  
        "last_update": "4'1",  
        "last_complete": "4'1",  
        "log_tail": "0'0",  
        "last_backfill": "MAX",  
        "purged_snaps": "[]",  
        "history": {  
            "epoch_created": 1,  
            "last_epoch_started": 537,  
            "last_epoch_clean": 537,  
            "last_epoch_split": 534,  
            "same_up_since": 536,  
            "same_interval_since": 536,  
            "same_primary_since": 536,  
            "last_scrub": "4'1",  
            "last_scrub_stamp": "2013-01-25 10:12:23.828174"  
        },  
        "stats": {  
            "version": "4'1",  
            "reported": "536'782",  
            "state": "active+clean",  
            "last_fresh": "2013-01-25 10:12:23.828271",  
            "last_change": "2013-01-25 10:12:23.828271",  
            "last_active": "2013-01-25 10:12:23.828271",  
            "last_clean": "2013-01-25 10:12:23.828271",  
            "last_unstale": "2013-01-25 10:12:23.828271",  
            "mapping_epoch": 535,  
            "log_start": "0'0",  
            "ondisk_log_start": "0'0",  
            "created": 1,  
            "last_epoch_clean": 1,  
            "parent": "0.0",  
            "parent_split_bits": 0,  
            "last_scrub": "4'1",  
            "last_scrub_stamp": "2013-01-25 10:12:23.828174",  
            "log_size": 128,  
            "ondisk_log_size": 128,  
            "stat_sum": {  
                "num_bytes": 205,  
                "num_objects": 1,  
                "num_object_clones": 0,  
                "num_object_copies": 0,  
                "num_objects_missing_on_primary": 0,  
                "num_objects_degraded": 0,  
            }  
        }  
    }  
}
```

```

    "num_objects_unfound": 0,
    "num_read": 1,
    "num_read_kb": 0,
    "num_write": 3,
    "num_write_kb": 1
  },
  "stat_cat_sum": {

  },
  "up": [
    1,
    0
  ],
  "acting": [
    1,
    0
  ]
},
"empty": 0,
"dne": 0,
"incomplete": 0
},
"recovery_state": [
{
  "name": "Started\\Primary\\Active",
  "enter_time": "2013-01-23 09:35:37.594691",
  "might_have_unfound": [
    ],
  "scrub": {
    "scrub_epoch_start": "536",
    "scrub_active": 0,
    "scrub_block_writes": 0,
    "finalizing_scrub": 0,
    "scrub_waiting_on": 0,
    "scrub_waiting_on_whom": [
      ]
    }
  },
  {
    "name": "Started",
    "enter_time": "2013-01-23 09:35:31.581160"
  }
]
}

```

The following subsections describe common states in greater detail.

后续子章节详述了常见状态。

3.3.3.4.1 存储池在建中

Creating

When you create a pool, it will create the number of placement groups you specified. Ceph will echo **creating** when it is creating one or more placement groups. Once they are created, the OSDs that are part of a placement group's Acting Set will peer. Once peering is complete, the placement group status should be **active+clean**, which means a Ceph client can begin writing to the placement group.

创建存储池时，它会创建指定数量的归置组。ceph 在创建一或多个归置组时会显示 **creating**；创建完后，在其归置组的 Acting Set 里的 OSD 将建立互联；一旦互联完成，归置组状态应该变为 **active+clean**，意思是 ceph 客户端可以向归置组写入数据了。



3.3.3.4.2 互联建立中

Peering

When Ceph is Peering a placement group, Ceph is bringing the OSDs that store the replicas of the placement group into **agreement about the state** of the objects and metadata in the placement group. When Ceph completes peering, this means that the OSDs that store the placement group agree about the current state of the placement group. However, completion of the peering process does **NOT** mean that each replica has the latest contents.

ceph 为归置组建立互联时，会让存储归置组副本的 OSD 之间就其中的对象和元数据状态达成一致。ceph 完成了互联，也就意味着存储着归置组的 OSD 就其当前状态达成了一致。然而，互联过程的完成并不能表明各副本都有了数据的最新版本。

权威历史

Authoritative History

Ceph will **NOT** acknowledge a write operation to a client, until all OSDs of the acting set persist the write operation. This practice ensures that at least one member of the acting set will have a record of every acknowledged write operation since the last successful peering operation.

With an accurate record of each acknowledged write operation, Ceph can construct and disseminate a new authoritative history of the placement group—a complete, and fully ordered set of operations that, if performed, would bring an OSD's copy of a placement group up to date.

ceph 不会向客户端确认写操作，直到 acting set 里的所有 OSD 都完成了写操作。这样处理保证了从上次成功互联起，acting set 中至少有一个成员确认了每个写操作。

有了各个已确认写操作的精确记录，ceph 就可以构建和散布一个新的归置组权威历史——一个完整、完全有序的操作集，如果被采用，就能把一个 OSD 的归置组副本更新到最新。

3.3.3.4.3 活跃

Active

Once Ceph completes the peering process, a placement group may become **active**. The **active** state means that the data in the placement group is generally available in the primary placement group and the replicas for read and write operations.

ceph 完成互联进程后，一归置组就可变为 active。active 状态通常意味着在主归置组和副本中的数据都可以读写。

3.3.3.4.4 整洁

Clean

When a placement group is in the **clean** state, the primary OSD and the replica OSDs have successfully peered and there are no stray replicas for the placement group. Ceph replicated all objects in the placement group the correct number of times.

某一归置组处于 clean 状态时，主 OSD 和副本 OSD 已成功互联，并且没有偏离的归置组。ceph 已把归置组中的对象复制了规定次数。

3.3.3.4.5 已降级

Degraded

When a client writes an object to the primary OSD, the primary OSD is responsible for writing the replicas to the replica OSDs. After the primary OSD writes the object to storage, the placement group will remain in a **degraded** state until the primary OSD has received an acknowledgement from the replica OSDs that Ceph created the replica objects successfully.

当客户端向主 OSD 写入数据时，由主 OSD 负责把副本写入其余复制 OSD。主 OSD 把对象写入复制 OSD 后，在没收到成功完成的确认前，主 OSD 会一直停留在 degraded 状态。

The reason a placement group can be **active+degraded** is that an OSD may be **active** even though it doesn't hold all of the objects yet. If an OSD goes down, Ceph marks each placement group assigned to the OSD as **degraded**. The OSDs must peer again when the OSD comes back online. However, a client can still write a new object to a **degraded** placement group if it is **active**.

归置组状态可以是 **active+degraded** 状态，原因在于一 OSD 即使没所有对象也可以处于 **active** 状态。如果一 OSD 挂了，ceph 会把相关的归置组都标记为 **degraded**；那个 OSD 重生后，它们必须重新互联。然而，如果归置组仍处于 **active** 状态，即便它处于 **degraded** 状态，客户端还可以向其写入新对象。

If an OSD is down and the degraded condition persists, Ceph may mark the down OSD as `out` of the cluster and remap the data from the down OSD to another OSD. The time between being marked `down` and being marked `out` is controlled by `mon osd down out interval`, which is set to 300 seconds by default.

如果一 OSD 挂了，且 `degraded` 状态持续，ceph 会把 `down` 的 OSD 标记为在集群外（`out`）并把那些 `down` 掉的 OSD 上的数据重映射到其它 OSD。从标记为 `down` 到 `out` 的时间间隔由 `mon osd down out interval` 控制，默认是 300 秒。

A placement group can also be `degraded`, because Ceph cannot find one or more objects that Ceph thinks should be in the placement group. While you cannot read or write to unfound objects, you can still access all of the other objects in the `degraded` placement group.

归置组也会被降级（`degraded`），因为归置组找不到本应存在于归置组中的一或多个对象，这时，你不能读或写找不到的对象，但仍能访问其它位于降级归置组中的对象。

3.3.3.4.6 恢复中

Recovering

Ceph was designed for fault-tolerance at a scale where hardware and software problems are ongoing. When an OSD goes `down`, its contents may fall behind the current state of other replicas in the placement groups. When the OSD is back up, the contents of the placement groups must be updated to reflect the current state. During that time period, the OSD may reflect a `recovering` state.

ceph 被设计为可容错，可抵御一定规模的软、硬件问题。当某 OSD 挂了（`down`）时，其内容版本会落后于归置组内的其它副本；它重生（`up`）时，归置组内容必须更新，以反映当前状态；在此期间，OSD 在 `recovering` 状态。

Recovery isn't always trivial, because a hardware failure might cause a cascading failure of multiple OSDs. For example, a network switch for a rack or cabinet may fail, which can cause the OSDs of a number of host machines to fall behind the current state of the cluster. Each one of the OSDs must recover once the fault is resolved.

恢复并非总是这些小事，因为一次硬件失败可能牵连多个 OSD。比如一个机柜的网络交换机失败了，这会导致多个主机落后于集群的当前状态，问题解决后每一个 OSD 都必须恢复。

Ceph provides a number of settings to balance the resource contention between new service requests and the need to recover data objects and restore the placement groups to the current state. The `osd recovery delay start` setting allows an OSD to restart, re-peer and even process some replay requests before starting the recovery process. The `osd recovery threads` setting limits the number of threads for the recovery process (1 thread by default). The `osd recovery thread timeout` sets a thread timeout, because multiple OSDs may fail, restart and re-peer at staggered rates. The `osd recovery max active` setting limits the number of recovery requests an OSD will entertain simultaneously to prevent the OSD from failing to serve. The `osd recovery max chunk` setting limits the size of the recovered data chunks to prevent network congestion.

ceph 提供了很多选项来均衡资源竞争，如新服务请求、恢复数据对象和恢复归置组到当前状态。`osd recovery delay start` 选项允许一 OSD 在开始恢复进程前，先重启、重建互联、甚至处理一些重放请求；`osd recovery threads` 选项限制恢复进程的线程数，默认为 1 线程；`osd recovery thread timeout` 设置线程超时，因为多个 OSD 可能交替失败、重启和重建互联；`osd recovery max active` 选项限制一 OSD 最多同时接受多少请求，以防它压力过大而不能正常服务；`osd recovery max chunk` 选项限制恢复数据块尺寸，以防网络拥塞。

3.3.3.4.7 回填中

Back Filling

When a new OSD joins the cluster, CRUSH will reassign placement groups from OSDs in the cluster to the newly added OSD. Forcing the new OSD to accept the reassigned placement groups immediately can put excessive load on the new OSD. Back filling the OSD with the placement groups allows this process to begin in the background. Once backfilling is complete, the new OSD will begin serving requests when it is ready.

有新 OSD 加入集群时，CRUSH 会把现有集群内的归置组重分配给它。强制新 OSD 立即接受重分配的归置组会使之过载，用归置组回填可使这个过程在后台开始。回填完成后，新 OSD 准备好时就可以对外服务了。

During the backfill operations, you may see one of several states: `backfill_wait` indicates that a backfill operation is pending, but isn't underway yet; `backfill` indicates that a backfill operation is underway; and, `backfill_too_full` indicates that a backfill operation was requested, but couldn't be completed due to insufficient storage capacity.

在回填操作期间，你可能见到几种状态之一：`backfill_wait` 表明一回填操作挂起了，还没开始；`backfill` 表明一回填操作正在进行；`backfill_too_full` 表明请求回填了，但是因存储空间不足而不能完成。

Ceph provides a number of settings to manage the load spike associated with reassigning placement groups to an OSD (especially a new OSD). By default, `osd_max_backfills` sets the maximum number of concurrent backfills to or from an OSD to 10. The `osd backfill full ratio` enables an OSD to refuse a backfill request if the OSD is approaching its full ratio (85%, by default). If an OSD refuses a backfill request, the `osd backfill retry interval` enables an OSD to retry the request (after 10 seconds, by default). OSDs can also set `osd backfill scan min` and `osd backfill scan max` to manage scan intervals (64 and 512, by default).

ceph 提供了多个选项来解决重分配归置组时相关的负载问题。默认，`osd_max_backfill` 把双向的回填并发量都设置为 10；`osd backfill full ratio` 可让一 OSD 在接近占满率（默认 85%）时拒绝回填请求，如果一 OSD 拒绝了回填请求，在 `osd backfill retry interval` 间隔之后将重试（默认 10 秒）；OSD 也能用 `osd backfill scan min` 和 `osd backfill scan max` 来管理扫描间隔（默认 64 和 512）。

3.3.3.4.8 被重映射

Remapped

When the Acting Set that services a placement group changes, the data migrates from the old acting set to the new acting set. It may take some time for a new primary OSD to service requests. So it may ask the old primary to continue to service requests until the placement group migration is complete. Once data migration completes, the mapping uses the primary OSD of the new acting set.

某一归置组的 Acting Set 变更时，数据要从旧集合迁移到新的。主 OSD 要花费一些时间才能提供服务，所以它可以让老的主 OSD 持续服务、直到归置组迁移完。数据迁移完后，主 OSD 会映射到新 acting set。

3.3.3.4.9 发蔫 (stale)

Stale

While Ceph uses heartbeats to ensure that hosts and daemons are running, the `ceph-osd` daemons may also get into a `stuck` state where they aren't reporting statistics in a timely manner (e.g., a temporary network fault). By default, OSD daemons report their placement group, up thru, boot and failure statistics every half second (i.e., 0.5), which is more frequent than the heartbeat thresholds. If the **Primary OSD** of a placement group's acting set fails to report to the monitor or if other OSDs have reported the primary OSD down, the monitors will mark the placement group `stale`.

虽然 ceph 用心跳来保证主机和守护进程在运行，但是 `ceph-osd` 仍有可能进入 `stuck` 状态，它们没有按时报告其状态（如网络瞬断）。默认，OSD 守护进程每半秒（0.5）会一次报告其归置组、出流量、引导和失败统计状态，此频率高于心跳阈值。如果一归置组的主 OSD 所在的 acting set 没能向监视器报告、或者其它监视器已经报告了那个主 OSD 已 `down`，监视器们就会把此归置组标记为 `stale`。

When you start your cluster, it is common to see the `stale` state until the peering process completes. After your cluster has been running for awhile, seeing placement groups in the `stale` state indicates that the primary OSD for those placement groups is `down` or not reporting placement group statistics to the monitor.

启动集群时，会经常看到 `stale` 状态，直到互联完成。集群运行一阵后，如果还能看到有归置组位于 `stale` 状态，就说明那些归置组的主 OSD 挂了（`down`）、或没在向监视器报告统计信息。

3.3.3.5 找出故障归置组

Identifying Troubled PGs

As previously noted, a placement group isn't necessarily problematic just because its state isn't `active+clean`. Generally, Ceph's ability to self repair may not be working when placement groups get stuck. The stuck states include:

如前所述，一个归置组状态是 `active+clean` 时未必有问题。一般来说，归置组卡住时 ceph 的自修复功能往往无能为力，卡住的状态细分为：

- **Unclean:** Placement groups contain objects that are not replicated the desired number of times. They should be recovering.
不干净：归置组里有些对象的复制数未达到期望次数，它们应该在恢复中。
- **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 up.

不活跃：归置组不能处理读写，因为它们在等着一个持有最新数据的 OSD 再次进入 up 状态。

- **Stale:** Placement groups are in an unknown state, because the OSDs that host them have not reported to the monitor cluster in a while (configured by `mon osd report timeout`).

发蔫：归置组们处于一种未知状态，因为存储它们的 OSD 有一阵子没向监视器报告了（由 `mon osd report timeout` 配置）。

To identify stuck placement groups, execute the following:

为找出卡住的归置组，执行：

```
ceph pg dump_stuck [unclean|inactive|stale]
```

See [Placement Group Subsystem](#) for additional details. To troubleshoot stuck placement groups, see [Troubleshooting PG Errors](#).

详情见 [Placement Group Subsystem](#)，排除卡住的归置组见 [Troubleshooting PG Errors](#)。

3.3.3.6 定位对象

Finding an Object Location

To store object data in the Ceph Object Store, a Ceph client must:

要把对象数据存入 Ceph 对象存储，一 ceph 客户端必须：

1. Set an object name

设置对象名

2. Specify a [pool](#)

指定一存储池

The Ceph client retrieves the latest cluster map and the CRUSH algorithm calculates how to map the object to a [placement group](#), and then calculates how to assign the placement group to an OSD dynamically. To find the object location, all you need is the object name and the pool name. For example:

ceph 客户端索取最新集群运行图、并用 CRUSH 算法计算对象到[归置组](#)的映射，然后计算如何动态地把归置组映射到 OSD。要定位对象，只需要知道对象名和存储池名字，例如：

```
ceph osd map {poolname} {object-name}
```

练习：定位一个对象

Excercise: Locate an Object

As an exercise, lets create an object. Specify an object name, a path to a test file containing some object data and a pool name using the `rados put` command on the command line. For example:

反正是练习，我们先创建一个对象。给 `rados put` 命令指定一对象名、一个包含数据的测试文件路径、和一个存储池名字，例如：

```
rados put {object-name} {file-path} --pool=data  
rados put test-object-1 testfile.txt --pool=data
```

To verify that the Ceph Object Store stored the object, execute the following:

用下列命令确认 ceph 对象存储已经包含此对象：

```
rados -p data ls
```

Now, identify the object location:

现在可以定位对象了：

```
ceph osd map {pool-name} {object-name}  
ceph osd map data test-object-1
```

Ceph should output the object's location. For example:

ceph 应该输出对象的位置，例如：

```
osdmap e537 pool 'data' (0) object 'test-object-1' -> pg 0.d1743484 (0.4) -> up [1,0] acting [1,0]
```

To remove the test object, simply delete it using the `rados rm` command. For example:

要删除测试对象，用 `rados rm` 即可，如：

```
rados rm test-object-1 --pool=data
```

As the cluster evolves, the object location may change dynamically. One benefit of Ceph's dynamic rebalancing is that Ceph relieves you from having to perform the migration manually. See the [Architecture](#) section for details.

随着集群的运转，对象位置会动态改变。ceph 动态重均衡的优点之一，就是把你从手动迁移中解救了，详情见 [Architecture](#)。

3.3.4 认证概览

ceph 认证及授权

Ceph authentication & authorization

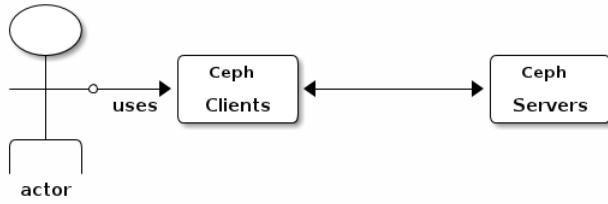
Ceph is a distributed storage system where a typical deployment involves a relatively small quorum of monitors, scores of metadata servers (MDSs) and many thousands of OSD daemons operating across many hosts/nodes—representing the server portion of the Ceph object store. Ceph clients such as CephFS, 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` 库与 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。

3.3.4.1 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.

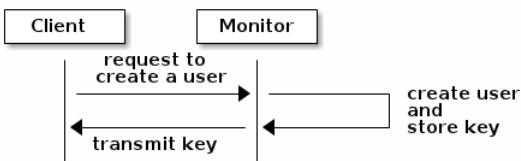
用户/参与者通过调用 `ceph` 客户端来联系监视器，不像 Kerberos，每个监视器都能认证用户、发布密钥，所以使用 `cephx` 时不会有单点故障或瓶颈。监视器返回一个类似 Kerberos 票据的认证数据结构，它包含一个可用于获取 `ceph` 服务的会话密钥，会话密钥是用户的永久私钥自加密过的，只有此用户能从 `ceph` 监视器请求服务。客户端用会话密钥向监视器请求需要的服务，然后监视器给客户端一个凭证用以向实际持有数据的 OSD 认证。`ceph` 的监视器和 OSD 共享相同的密钥，所以集群内任何 OSD 或元数据服务器都认可客户端从监视器获取的凭证，像 Kerberos 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。

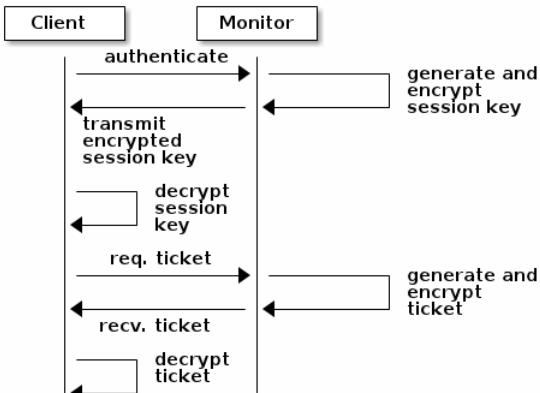
Note: The `client.admin` user must provide the user ID and secret key to the user in a secure manner.

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。



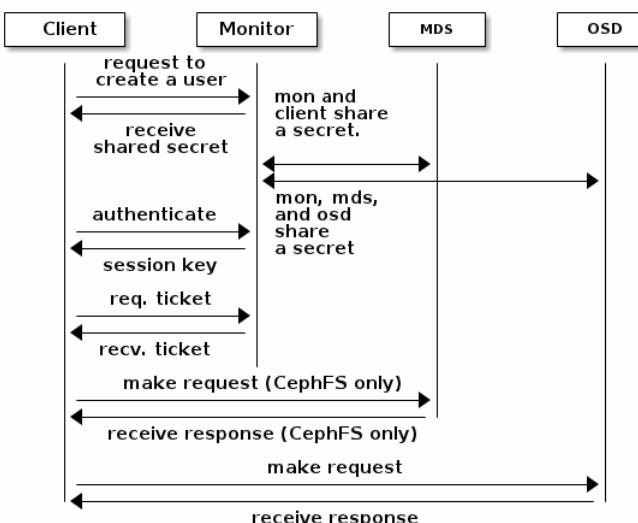
To authenticate with the monitor, the client passes in the user name to the monitor, and the monitor generates a session key and encrypts it with the secret key associated to the user name. Then, the monitor transmits the encrypted ticket back to the client. The client then decrypts the payload with 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。

3.3.4.2 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.
在守护进程的访问设置之前，仅对 MDS 隐含 `rw`。

Example: `ceph-authtool -n client.foo --cap mds 'allow'`

r

Description: Gives the user read access. Required with monitors to retrieve the CRUSH map.
授予用户读权限，监视器需要它才能搜刮 CRUSH 图。

Example: `ceph-authtool -n client.foo --cap mon 'allow r'`

w

Description: Gives the user write access to objects.
授予用户写对象的权限。

Example: `ceph-authtool -n client.foo --cap osd 'allow w'`

x

Description: Gives the user the capability to call class methods (i.e., both read and write).
授予用户调用类方法的能力，如同时有读和写。

Example: `ceph-authtool -n client.foo --cap osd 'allow x'`

class-read

Descriptions: Gives the user the capability to call class read methods. Subset of x.
授予用户调用类读取方法的能力，`x`的子集。

Example: `ceph-authtool -n client.foo --cap osd 'allow class-read'`

class-write

Description: Gives the user the capability to call class write methods. Subset of x.
授予用户调用类写入方法的能力，`x`的子集。

Example: `ceph-authtool -n client.foo --cap osd 'allow class-write'`

*

Description: Gives the user read, write and execute permissions for a particular daemon/pool, and the ability to execute admin commands.

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。

Example: `ceph-authtool -n client.foo --cap osd 'allow *'`

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'
```

3.3.4.3 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。

3.3.5 cephx 认证

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.
none: 任何用户无需认证就能访问数据；
- Cephx: Ceph requires user authentication in a manner similar to Kerberos.
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。

Important: The cephx protocol does not address data encryption in transport (e.g., SSL/TLS) or encryption at rest.

重要：cephx 协议不解决传输加密（如 SSL/TLS）或存储加密问题。

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。

3.3.5.1 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.

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。

1. First, you must generate a secret key for the default client.admin user so the administrator can execute Ceph commands.
首先，必须给默认的 `client.admin` 生成密钥，以便管理员执行 ceph 命令；
2. Second, you must generate a monitor secret key and distribute it to all monitors in the cluster.
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；
3. Finally, you can follow the remaining steps in Enabling Cephx to enable authentication.
最后，你要完成启用 cephx 里的剩余步骤。

3.3.5.1.1 `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。

Important: To run Ceph commands on the command line with cephx enabled, you need to create a key for the `client.admin` user, and create a secret file under /etc/ceph.

重要：在启用了 cephx 的 ceph 上执行命令时，你得给 `client.admin` 用户创建一个密钥、并在 /etc/ceph 下创建一个密钥文件。

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](#)。

3.3.5.1.2 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。

3.3.5.1.3 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。

1. 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!
```

2. 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
```

3. 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 data](#) 目录下：

```
cp /tmp/monitor-key /var/lib/ceph/mon/ceph-a/keyring
```

4. 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
```

5. 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
```

6. 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
```

7. 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。

1. Start or restart the Ceph cluster.

启动或重启 ceph 集群：

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

3.3.5.1.4 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.

下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。

1. 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
```

2. 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
```

3. Start or restart the Ceph cluster.

启动或重启 ceph 集群：

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

3.3.5.1.5 守护进程密钥环

DAEMON KEYRINGS

With the exception of the monitors, Ceph generates daemon keyrings in the same way that it generates user keyrings. 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 `auth` 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。

3.3.5.2 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。

3.3.5.2.1 增加密钥

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访问监视器、元数据服务器和集群。能力用字符串定义，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，所有能力都可对地表示为 `{type}` 和 `{capability}`。

```
sudo ceph auth get-or-create 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 `client.foo`，他对 `osd` 有 `rw` 权限、对 `mon` 有 `r` 权限：

```
sudo ceph auth get-or-create client.foo osd 'allow rw' mon 'allow r' > keyring.foo
```

Note: User names are associated to user types, which include `client`, `osd`, `mon`, and `mds`. In most cases, you will be creating keys for `client` users.

注意：用户名和用户类型相关联，包括 `client` 和 `osd`、`mon`、和 `mds`，在大多数情况下，你要给 `client` 用户创建密钥。

After you add a key to the cluster keyring, go to the relevant client(s) and copy the keyring from the cluster host to the client(s).

把密钥加入集群密钥环后，把集群密钥环从集群主机拷贝到相关客户端。

```
sudo scp {user}@{ceph-cluster-host}:/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：确保你客户端主机上的 `ceph.keyring` 文件设置了合适的权限位（如 `chmod 644`）。

3.3.5.2.2 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 `client`、`osd`、`mon`、`mds` 中一个，`{ID|username}` 是守护进程或用户名的 ID。

After you delete a key from the cluster keyring, go to the relevant client(s) and copy the keyring from the cluster host to the client(s).

从集群密钥环删除密钥后，把集群密钥环从集群主机拷贝到相关客户端。

```
sudo scp {user}@{ceph-cluster-host}:/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：确保你客户端主机上的 `ceph.keyring` 文件设置了合适的权限位（如 `chmod 644`）。

3.3.5.2.3 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
```

3.3.5.3 `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-or-create 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 获得密钥并保存在本地，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。

3.3.5.4 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 Ceph with the default scenario, Ceph still authenticates the initial communication, but you lose desirable session security.

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。

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 in the [global] section of your [Ceph configuration](#) file directly below the line that specifies the use of Cephx for authentication:

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 ; everywhere possible
```

You can also selectively require signatures for cluster internal communications only, separate from client-facing service:

你也可以选择只对集群内部通讯要求签名，它和面向客户端的服务是分离的：

```
cephx cluster require signatures = true ; for cluster-internal communication
cephx service require signatures = true ; for client-facing service
```

An option to make a client require signatures from the cluster is not yet implemented.

客户端向集群要求签名的选项还没实现。

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。

3.3.6 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 ls pools
```

The default pools include:

默认存储池有：

- `data`
- `metadata`
- `rbd`

3.3.7.2 创建存储池

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.
存储池名称，必须唯一。

Type: String

Required: Yes

`{pg-num}`

Description: The total number of placement groups for the pool. See [Placement Groups](#) for details on calculating a suitable number. The default value 8 is NOT suitable for most systems.

存储池拥有的归置组总数。关于如何计算合适的数值，请参见 [Placement Groups](#)。默认值 8 对大多数系统都不合适。

Type: Integer

Required: Yes

Default: 8

{pgp-num}

Description: The total number of placement groups for placement purposes. This **should be equal to the total number of placement groups**, except for placement group splitting scenarios.

用于归置的归置组总数。此值应该等于归置组总数，归置组分割的情况下除外。

Type: Integer
Required: Yes

Default: 8

When you create a pool, set the number of placement groups to a reasonable value (e.g., 100). Consider the total number of placement groups per OSD too. Placement groups are computationally expensive, so performance will degrade when you have many pools with many placement groups (e.g., 50 pools with 100 placement groups each). The point of diminishing returns depends upon the power of the OSD host.

创建存储池时，要设置一个合理的归置组数量，如 100。也要考虑到每 OSD 的归置组总数，因为归置组很耗计算资源，所以很多存储池和很多归置组（如 50 个存储池，各包含 100 归置组）会导致性能下降。收益递减点取决于 OSD 主机的强大。

Important: Increasing the number of placement groups in a pool after you create the pool is still an experimental feature in Bobtail (v 0.56). We recommend defining a reasonable number of placement groups and maintaining that number until Ceph's placement group splitting and merging functionality matures.

重要: 增加一现有存储池中的归置组数量仍然是 v0.56 新增的一个实验功能。我们建议创建时就指定一个合适的归置组数量并维持不变，直到 ceph 的归置组拆分、合并功能成熟。

See Placement Groups for details on calculating an appropriate number of placement groups for your pool.

如何为存储池计算合适的归置组数量请参见 [归置组](#)。

3.3.7.3 删除存储池

DELETE A POOL

To delete a pool, execute:

要删除一存储池，执行：

```
ceph osd pool delete {pool-name} [{pool-name} --yes-i-really-really-mean-it]
```

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）。

Note: Version 0.48 Argonaut and above.

注意：适用 0.48 及以上。

3.3.7.5 查看存储池统计信息

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}
```

Note: Version 0.48 Argonaut and above.

注意：适用 0.48 及以上。

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}
```

Note: Version 0.48 Argonaut and above.

注意：适用 0.48 及以上。

3.3.7.8 设置存储池键值

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.

设置存储池中对象的副本数，详情参见[设置对象副本数](#)。

Type: Integer

min_size

Description: Sets the minimum number of replicas required for io. See [Set the Number of Object Replicas](#) for further details

设置 IO 需要的最小副本数，详情参见[设置对象副本数](#)。

Type: Integer

Note: Version 0.54 and above

注意：0.54 版及以上。

crash_replay_interval

Description: The number of seconds to allow clients to replay acknowledged, but uncommitted requests.

允许客户端重放确认而未提交请求的秒数。

Type: Integer

pgp_num

Description: The effective number of placement groups to use when calculating data placement.
计算数据归置时使用的有效归置组数量。

Type: Integer

Valid Range: Equal to or less than pg_num.

crush_ruleset

Description: The ruleset to use for mapping object placement in the cluster.
集群内映射对象归置时使用的规则集。

Type: Integer

Note: Version 0.48 Argonaut and above.

注意：0.48 版及以上。

3.3.7.9 获取存储池键值

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.
存储池的归置组数量。

Type: Integer

pgp_num

Description: The effective number of placement groups to use when calculating data placement.
计算数据归置时使用的归置组有效数量。

Type: Integer

Valid Range: Equal to or less than pg_num.
小于等于 pg_num。

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}
```

Important: The {num-replicas} includes the object itself. If you want the object and two copies of the object for a total of three instances of the object, specify 3.

重要：{num-replicas}包括对象自身，如果你想要对象自身及其两份拷贝共计三份，指定 3。

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 获得对象副本数

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).

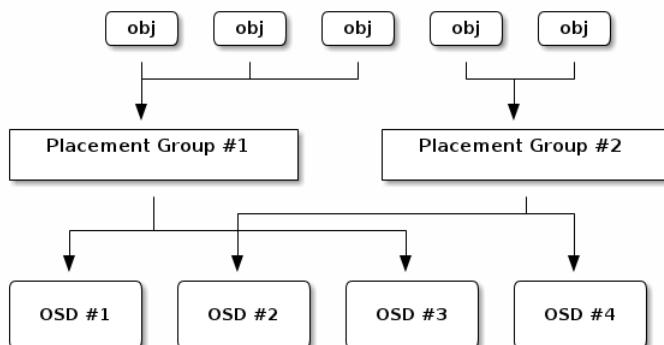
ceph 会列出存储池，且高亮 rep size 属性，它创建了一个对象的两个副本（两份拷贝）。

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.
直接地：每个 PG 需要一些内存和 CPU；
- Indirectly: The total number of PGs increases the peering count.
间接地：PG 总量增加了连接建立数量。

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:

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：

$$\text{Total PGs} = \frac{(\text{OSDs} * 100)}{\text{Replicas}}$$

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 `plain` (默认) 和 `json`。阈值定义的是，归置组被认为卡住前等待的最长时间（默认 300 秒）。

3.3.8.5 获取一归置组运行图

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。

3.3.8.1 归置组状态

Placement Group States

When checking a cluster’s status (e.g., running `ceph -w` or `ceph -s`), Ceph will report on the status of the placement groups. A placement group has one or more states. The optimum state for placement groups in the placement group map is `active + clean`.

检查集群状态时（如运行 `ceph -s` 或 `ceph -w`），ceph 会报告归置组状态。一个归置组有一到多种状态，其最优状态为 `active+clean`。

Creating

Ceph is still creating the placement group.
ceph 仍在创建归置组。

Active

Ceph will process requests to the placement group.
ceph 可处理到归置组的请求。

Clean

Ceph replicated all objects in the placement group the correct number of times.
ceph 把归置组内的对象复制了规定次数。

Down

A replica with necessary data is down, so the placement group is offline.
包含必备数据的副本挂了，所以归置组离线。

Replay

The placement group is waiting for clients to replay operations after an OSD crashed.
某 OSD 崩溃后，归置组在等待客户端重放操作。

Splitting

Ceph is splitting the placement group into multiple placement groups. (functional?)
ceph 正在把一归置组分割为多个。（实现了？）

Scrubbing

Ceph is checking the placement group for inconsistencies.
ceph 正在检查归置组的一致性。

Degraded

Ceph has not replicated some objects in the placement group the correct number of times yet.
归置组内的对象还没复制到规定次数。

Inconsistent

Ceph detects inconsistencies in the one or more replicas of an object in the placement group (e.g. objects are the wrong size, objects are missing from one replica after recovery finished, etc.).
ceph 检测到了归置组内一或多个副本间不一致（如各对象大小不一、恢复后对象还没复制到副本那里、等等）。

Peering

The placement group is undergoing the peering process
归置组正在互联。

Repair

Ceph is checking the placement group and repairing any inconsistencies it finds (if possible).
ceph 正在检查归置组、并试图修复发现的不一致（如果可能的话）。

Recovering

Ceph is migrating/synchronizing objects and their replicas.
ceph 正在迁移/同步对象及其副本。

Backfill

Ceph is scanning and synchronizing the entire contents of a placement group instead of inferring what contents need to be synchronized from the logs of recent operations. **Backfill** is a special case of recovery.

ceph 正在扫描并同步整个归置组的内容，而不是根据日志推算哪些最新操作需要同步。Backfill 是 recovery 的一种特殊情况。

Wait-backfill

The placement group is waiting in line to start backfill.

归置组正在排队，等候回填。

Incomplete

Ceph detects that a placement group is missing a necessary period of history from its log. If you see this state, report a bug, and try to start any failed OSDs that may contain the needed information.

ceph 根据一归置组的日志检测到它丢失了一段历史。如果你看到这种状态，请报告一个 bug，并试着重启所有包含相关信息的 OSD。

Stale

The placement group is in an unknown state - the monitors have not received an update for it since the placement group mapping changed.

归置组处于一种未知状态——从归置组运行图变更起就没再收到它的更新。

Remapped

The placement group is temporarily mapped to a different set of OSDs from what CRUSH specified.

归置组被临时映射到了另外一组 OSD，它们不是 CRUSH 算法指定的。

3.3.8.8.2 归置组术语解释

Placement Group Concepts

When you execute commands like `ceph -w`, `ceph osd dump`, and other commands related to placement groups, Ceph may return values using some of the following terms:

当你执行诸如 `ceph -w`、`ceph osd dump`、及其他和归置组相关的命令时，ceph 会返回下列术语及其值：

Peering

The process of bringing all of the OSDs that store a Placement Group (PG) into agreement about the state of all of the objects (and their metadata) in that PG. Note that agreeing on the state does not mean that they all have the latest contents.

是一种过程，它使得存储着同一归置组的所有 OSD 对归置组内的所有对象及其元数据统一意见。需要注意的是，达成一致不意味着它们都有最新内容。

Acting Set

The ordered list of OSDs who are (or were as of some epoch) responsible for a particular placement group.

一列有序 OSD，它们为某一特定归置组（或其中一些元版本）负责。

Up Set

The ordered list of OSDs responsible for a particular placement group for a particular epoch according to CRUSH. Normally this is the same as the **Acting Set**, except when the **Acting Set** has been explicitly overridden via `pg_temp` in the OSD Map.

[译者：此处原文没看懂……瞎译如下]

一列有序 OSD，它们为某一特定元版本所在的特定归置组负责。它通常和 Acting Set 相同，除非 Acting Set 被 OSD 运行图里的 `pg_temp` 显式地覆盖掉了。

Current Interval or Past Interval

A sequence of OSD map epochs during which the **Acting Set** and **Up Set** for particular placement group do not change.

某一特定归置组所在 Acting Set 和 Up Set 未更改时的一系列 OSD 运行图元版本。

Primary

The member (and by convention first) of the **Acting Set**, that is responsible for coordination peering, and is the only OSD that will accept client-initiated writes to objects in a placement group.

Acting Set 的成员（按惯例第一个），它负责协调互联，并且是归置组内唯一接受客户端初始写入的 OSD。

Replica

A non-primary OSD in the **Acting Set** for a placement group (and who has been recognized as such and activated by the primary).

一归置组的 Acting Set 内不是主 OSD 的其它 OSD，它们被同等对待、并由主 OSD 激活。

Stray

An OSD that is not a member of the current **Acting Set**, but has not yet been told that it can delete its copies of a particular placement group.

某一 OSD，它不再是当前 Acting Set 的成员，但还没被告知它可以删除那个归置组拷贝。

Recovery

Ensuring that copies of all of the objects in a placement group are on all of the OSDs in the **Acting Set**. Once **Peering** has been performed, the **Primary** can start accepting write operations, and **Recovery** can proceed in the background.

确保 Acting Set 内、一归置组中的所有对象的副本都存在于所有 OSD 上。一旦互联完成，主 OSD 就以接受写操作，且恢复进程可在后台进行。

PG Info

Basic metadata about the placement group's creation epoch, the version for the most recent write to the

placement group, **last epoch started**, **last epoch clean**, and the beginning of the **current interval**. Any inter-OSD communication about placement groups includes the **PG Info**, such that any OSD that knows a placement group exists (or once existed) also has a lower bound on **last epoch clean** or **last epoch started**.

[译者：此处原文没看透……瞎译如下]

基本元数据，关于归置组创建元版本、向归置组的最新写版本、最近的开始元版本 (*last epoch started*)、最近的干净元版本 (*last epoch clean*)、和当前间隔 (*current interval*) 的起点。OSD 间关于归置组的任何通讯都包含 PG Info，这样任何知道一归置组存在（或曾经存在）的 OSD 也必定有 *last epoch clean* 或 *last epoch started* 的下限。

PG Log

A list of recent updates made to objects in a placement group. Note that these logs can be truncated after all OSDs in the **Acting Set** have acknowledged up to a certain point.

一归置组内对象的一系列最近更新。注意，这些日志在 Acting Set 内的所有 OSD 确认更新到某点后可以删除。

Missing Set

Each OSD notes update log entries and if they imply updates to the contents of an object, adds that object to a list of needed updates. This list is called the **Missing Set** for that <OSD, PG>.

各 OSD 都记录更新日志，而且如果它们包含对象内容的更新，会把那个对象加入一个待更新列表，这个列表叫做那个<OSD, PG>的 Missing Set。

Authoritative History

A complete, and fully ordered set of operations that, if performed, would bring an OSD's copy of a placement group up to date.

一个完整、完全有序的操作集合，如果应用，可把一 OSD 上的归置组副本带到最新。

Epoch

A (monotonically increasing) OSD map version number

单递增 OSD 运行图版本号。

Last Epoch Start

The last epoch at which all nodes in the **Acting Set** for a particular placement group agreed on an **Authoritative History**. At this point, **Peering** is deemed to have been successful.

一最新元版本，在这点上，一归置组所对应 Acting Set 内的所有节点都对权威历史达成了一致、并且互联被认为成功了。

up_thru

Before a **Primary** can successfully complete the **Peering** process, it must inform a monitor that is alive through the current osd map **Epoch** by having the monitor set its **up thru** in the osd map. This helps Peering ignore previous **Acting Sets** for which Peering never completed after certain sequences of failures, such as the second interval below:

主 OSD 要想成功完成互联，它必须通过当前 OSD 运行图 epoch 通知一个监视器，并在 osd 运行图中设置其 up_thru。这会使互联进程忽略之前的 Acting Set，因为它经历特定顺序的失败后一直不能互联，比如像下面的第二周期：

- **acting set = [A,B]**
- **acting set = [A]**
- **acting set = [] very shortly after (e.g., simultaneous failure, but staggered detection)**
acting set = [] 之后很短时间（例如同时失败、但探测是交叉的）
- **acting set = [B] (B restarts, A does not)**

Last Epoch Clean

The last **Epoch** at which all nodes in the **Acting set** for a particular placement group were completely up to date (both placement group logs and object contents). At this point, **recovery** is deemed to have been completed.

最近的 Epoch，这时某一特定归置组所在 Acting Set 内的所有节点都全部更新了（包括归置组日志和对象内容）。在这点上，恢复被认为已完成。

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 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：

1. [Get the CRUSH Map](#).
2. [Decompile](#) the CRUSH Map.
3. Edit at least one of [Devices](#), [Buckets](#) and [Rules](#).
4. [Recompile](#) the CRUSH Map.
5. [Set the CRUSH Map](#).

1. 获取 CRUSH 图；
2. 反编译 CRUSH 图；
3. 至少编辑一个设备、桶、规则；
4. 重编译 CRUSH 图；
5. 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[设置存储池键值](#)。

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 将把你指定的已编译的 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 four main sections to a CRUSH Map.

CRUSH 图主要有 4 个主要段落。

1. **Devices:** Devices consist of any object storage device—i.e., the storage drive corresponding to a `ceph-osd` daemon. You should have a device for each OSD daemon in your Ceph configuration file.

devices 由任意对象存储设备组成，如对应一个 `ceph-osd` 进程的存储器。ceph 配置文件里的每个 OSD 都应该有一个设备。

2. **Bucket Types:** Bucket types define the types of buckets used in your CRUSH hierarchy. Buckets consist of a hierarchical aggregation of storage locations (e.g., rows, racks, hosts, etc.) and their assigned weights.

bucket types（桶类型）：定义了 CRUSH 分级结构里要用的桶类型，桶由逐级汇聚的存储位置（如行、机柜、主机等等）及其权重组成。

3. **Bucket Instances:** Once you define bucket types, you must declare bucket instances for your hosts, and any other failure domain partitioning you choose.

Bucket Instances: 定义了桶类型后，还必须声明主机的桶类型、以及规划的其它故障域。

4. **Rules:** Rules consist of the manner of selecting buckets.

rules: 由选择桶的方法组成。

If you launched Ceph using one of our Quick Start guides, you'll notice that you didn't need to create a CRUSH map. Ceph's deployment tools generate a default CRUSH map that lists devices from the OSDs you defined in your Ceph configuration file, and it declares a bucket for each host you specified in the [osd] sections of your Ceph configuration file. You should create your own CRUSH maps with buckets that reflect your cluster's failure domains to better ensure data safety and availability.

如果你用我们的某个 Quick Start Guide 配起了 ceph，应该注意到了，你并不需要创建 CRUSH 运行图。ceph 部署工具生成了默认 CRUSH 运行图，它列出了你定义在 ceph 配置文件中的 OSD 设备、并把配置文件 [osd] 段下定义的各 OSD 主机声明为桶。为保证数据安全和可用，你应该创建自己的 CRUSH 图，以反映出自己集群的故障域。

Note: The generated CRUSH map doesn't take your larger grained failure domains into account. So you should modify your CRUSH map to account for larger grained failure domains such as racks, rows, data centers, etc.

注意：生成的 CRUSH 图没考虑大粒度故障域，所以你修改 CRUSH 图时要考虑上，像机柜、行、数据中心。

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. To declare a device in the CRUSH map, create a new line under your list of devices, enter `device` followed by a unique numeric ID, followed by the corresponding `ceph-osd` daemon instance.

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。要在 CRUSH 图里声明一个设备，在设备列表后面新建一行，输入 `device`、之后是唯一的数字 ID、之后是相应的 `ceph-osd` 守护进程例程名字。

```
#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 图之桶类型

CRUSH Map Bucket Types

The second list in the CRUSH map defines ‘bucket’ types. Buckets facilitate a hierarchy of nodes and leaves. Node (or non-leaf) buckets typically represent physical locations in a hierarchy. Nodes aggregate other nodes or leaves. Leaf buckets represent ceph-osd daemons and their corresponding storage media.

CRUSH 图里的第二个列表定义了 bucket (桶) 类型，桶简化了节点和叶子层次。节点 (或非叶) 桶在分级结构里一般表示物理位置，节点汇聚了其它节点或叶子，叶桶表示 ceph-osd 守护进程及其对应的存储媒体。

Tip: The term “bucket” used in the context of CRUSH means a node in the hierarchy, i.e. a location or a piece of physical hardware. It is a different concept from the term “bucket” when used in the context of RADOS Gateway APIs.

提示：CRUSH 中用到的 bucket 意思是分级结构中的一个节点，比如一个位置或一部分硬件。但是在 RADOS 网关接口的术语中，它又是不同的概念。

To add a bucket type to the CRUSH map, create a new line under your list of bucket types. Enter `type` followed by a unique numeric ID and a bucket name. By convention, there is one leaf bucket and it is `type 0`; however, you may give it any name you like (e.g., `osd`, `disk`, `drive`, `storage`, etc.):

要往 CRUSH 图中增加一种 bucket 类型，在现有桶类型列表下方新增一行，输入 `type`、之后是惟一数字 ID 和一个桶名。按惯例，会有一个叶子桶为 `type 0`；然而你可以指定任何名字（如 `osd`、`disk`、`drive`、`storage` 等等）：

```
#types  
type {num} {bucket-name}
```

For example:

例如：

```
# types  
type 0 osd  
type 1 host  
type 2 rack
```

3.3.9.2.3 CRUSH 图之桶层次

CRUSH Map Bucket Hierarchy

The CRUSH algorithm distributes data objects among storage devices according to a per-device weight value, approximating a uniform probability distribution. CRUSH distributes objects and their replicas according to the hierarchical cluster map you define. Your CRUSH map represents the available storage devices and the logical elements that contain them.

CRUSH 算法根据各设备的权重、大致统一的概率把数据对象分布到存储设备中。CRUSH 根据你定义的集群运行图分布对象及其副本，CRUSH 图表达了可用存储设备以及包含它们的逻辑单元。

To map placement groups to OSDs across failure domains, a CRUSH map defines a hierarchical list of bucket types (i.e., under `#types` in the generated CRUSH map). The purpose of creating a bucket hierarchy is to segregate the leaf nodes by their failure domains, such as hosts, racks, rows, rooms, and data centers. With the exception of the leaf nodes representing OSDs, the rest of the hierarchy is arbitrary, and you may define it according to your own needs.

要把归置组映射到跨故障域的 OSD，一个 CRUSH 图需定义一系列分级桶类型（如在现有 CRUSH 图的 `#type` 下）。创建桶分级结构的目的是按故障域隔离叶节点，像主机、机柜、行、房间、和数据中心。除了表示叶节点的 OSD，其它分级结构都是任意的，你可以按需定义。

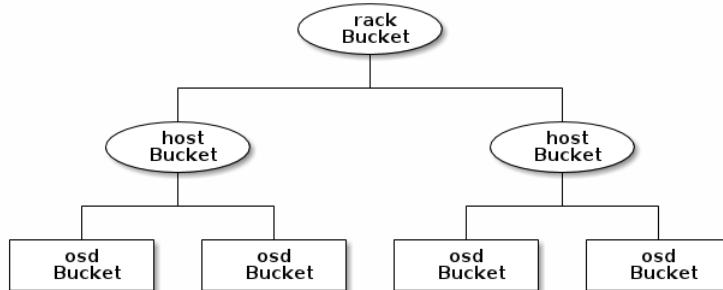
We recommend adapting your CRUSH map to your firm's hardware naming conventions and using instances names that reflect the physical hardware. Your naming practice can make it easier to administer the cluster and troubleshoot problems when an OSD and/or other hardware malfunctions and the administrator need access to physical hardware.

我们建议 CRUSH 图内的命名符合贵公司的硬件命名规则，并且采用反映物理硬件的例程名。良好的命名可简化集群管理和故障排除，当 OSD 和/或其它硬件出问题时，管理员可轻易找到对应物理硬件。

In the following example, the bucket hierarchy has a leaf bucket named `osd`, and two node buckets named

host 和 rack 分别表示机架和主机。

在下例中，桶分级结构有一个分支名为 osd、和两个节点分别名为 host 和 rack。



Note: The higher numbered rack bucket type aggregates the lower numbered host bucket type.

注意：编号较高的 rack 桶类型汇聚编号较低的 host 桶类型。

Since leaf nodes reflect storage devices declared under the #devices list at the beginning of the CRUSH map, you do not need to declare them as bucket instances. The second lowest bucket type in your hierarchy usually aggregates the devices (i.e., it's usually the computer containing the storage media, and uses whatever term you prefer to describe it, such as "node", "computer", "server," "host", "machine", etc.).

位于 CRUSH 图起始部分的 #devices 列表表示叶节点的存储设备，没必要声明为桶例程。位于分级结构第二低层的桶一般用于汇聚设备（如它通常是包含存储媒体的计算机，你可以用自己喜欢的名字描述，如节点、计算机、服务器、主机、机器等等）。

When declaring a bucket instance, you must specify its type, give it a unique name (string), assign it a unique ID expressed as a negative integer (optional), specify a weight relative to the total capacity/capability of its item(s), specify the bucket algorithm (usually straw), and the hash (usually 0, reflecting hash algorithm rjenkins1). A bucket may have one or more items. The items may consist of node buckets or leaves. Items may have a weight that reflects the relative weight of the item.

声明一个桶例程时，你必须指定其类型、惟一名称（字符串）、惟一负整数 ID（可选）、指定各条目总容量/能力相关的权重、指定桶算法（通常是 straw）、和哈希（通常为 0，表示哈希算法 rjenkins1）。一个桶可以包含一到多条，这些条目可以由节点桶或叶子组成，它们可以有个权重用来反映条目的相对权重。

You may declare a node bucket with the following syntax:

你可以按下列语法声明一个节点桶：

```
[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]  
}
```

For example, using the diagram above, we would define two host buckets and one rack bucket. The OSDs are declared as items within the host buckets:

例如，用上面的图表，我们可以定义两个主机桶和一个机架桶，OSD 被声明为主机桶内的条目：

```
host node1 {  
    id -1  
    alg straw  
    hash 0  
    item osd.0 weight 1.00  
    item osd.1 weight 1.00  
}  
  
host node2 {  
    id -2  
    alg straw  
    hash 0  
    item osd.2 weight 1.00  
    item osd.3 weight 1.00  
}
```

```

rack rack1 {
    id -3
    alg straw
    hash 0
    item node1 weight 2.00
    item node2 weight 2.00
}

```

Note: In the foregoing example, note that the rack bucket does not contain any OSDs. Rather it contains lower level host buckets, and includes the sum total of their weight in the item entry.

注意：在前述示例中，机柜桶不包含任何 OSD，它只包含低一级的主机桶、以及其内条目的权重之和。

桶类型

Bucket Types

Ceph supports four bucket types, each representing a tradeoff between performance and reorganization efficiency. If you are unsure of which bucket type to use, we recommend using a `straw` bucket. For a detailed discussion of bucket types, refer to [CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data](#), and more specifically to [Section 3.4](#). The bucket types are:

`ceph` 支持四种桶，每种都是性能和组织简易间的折衷。如果你不确定用哪种桶，我们建议 `straw`，关于各类桶的详细讨论见 [CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data](#)，特别是 [Section 3.4](#)。支持的桶类型有：

1. **Uniform:** Uniform buckets aggregate devices with **exactly** the same weight. For example, when firms commission or decommission hardware, they typically do so with many machines that have exactly the same physical configuration (e.g., bulk purchases). When storage devices have exactly the same weight, you may use the `uniform` bucket type, which allows CRUSH to map replicas into uniform buckets in constant time. With non-uniform weights, you should use another bucket algorithm.

Uniform: 这种桶用完全相同的权重汇聚设备。例如，公司采购或淘汰硬件时，一般都有相同的物理配置（如批发）。当存储设备权重都相同时，你可以用 `uniform` 桶类型，它允许 CRUSH 按常数把副本映射到 `uniform` 桶。权重不统一时，你应该采用其它算法。

2. **List:** List buckets aggregate their content as linked lists. Based on the `p` algorithm, a list is a natural and intuitive choice for an **expanding cluster**: either an object is relocated to the newest device with some appropriate probability, or it remains on the older devices as before. The result is optimal data migration when items are added to the bucket. Items removed from the middle or tail of the list, however, can result in a significant amount of unnecessary movement, making list buckets most suitable for circumstances in which they **never (or very rarely) shrink**.

list: 这种桶把它们的内容汇聚为链表。它基于 `p` 算法，一个列表就是一个自然、直观的**扩张集群**：对象会按一定概率被重定位到最新的设备、或者像从前一样仍保留在较老的设备上。结果是优化了新条目加入桶时的数据迁移。然而，如果从链表的中间或末尾删除了一些条目，将会导致大量没必要的挪动。所以这种桶适合**永不或极少缩减**的场景。

3. **Tree:** Tree buckets use a binary search tree. They are more efficient than list buckets when a bucket contains a larger set of items. Based on the `R` algorithm, tree buckets reduce the placement time to $O(\log n)$, making them suitable for managing much larger sets of devices or nested buckets.

tree: 它用一种二进制搜索树，在桶包含大量条目时比 `list` 桶更高效。它基于 `R` 算法，`tree` 桶把归置时间减少到了 $O(\log n)$ ，这使得它们更适合管理更大规模的设备或嵌套桶。

4. **Straw:** List and Tree buckets use a divide and conquer strategy in a way that either gives certain items precedence (e.g., those at the beginning of a list) or obviates the need to consider entire subtrees of items at all. That improves the performance of the replica placement process, but can also introduce suboptimal reorganization behavior when the contents of a bucket change due an addition, removal, or re-weighting of an item. The straw bucket type allows all items to fairly “compete” against each other for replica placement through a process analogous to a draw of straws.

straw: `list` 和 `tree` 桶用分而治之策略，给特定条目一定优先级（如位于链表开头的条目）、或避开对整个子树上所有条目的考虑。这样提升了副本归置进程的性能，但是也导致了重新组织时的次优结果，如增加、拆除、或重设某条目的权重。`straw` 桶类型允许所有条目模拟拉稻草的过程公平地相互“竞争”副本归置。

Hash

Each bucket uses a hash algorithm. Currently, Ceph supports `rjenkins1`. Enter `0` as your hash setting to select `rjenkins1`.

各个桶都用了一种哈希算法，当前 ceph 仅支持 `rjenkins1`，输入 `0` 表示哈希算法设置为 `rjenkins1`。

调整桶的权重

Weighting Bucket Items

Ceph expresses bucket weights as double integers, which allows for fine weighting. A weight is the relative difference between device capacities. We recommend using `1.00` as the relative weight for a 1TB storage device. In such a scenario, a weight of `0.5` would represent approximately 500GB, and a weight of `3.00` would represent approximately 3TB. Higher level buckets have a weight that is the sum total of the leaf items aggregated by the bucket.

ceph 用双整形表示桶权重。权重和设备容量不同，我们建议用 `1.00` 作为 1TB 存储设备的相对权重，这样 `0.5` 的权重大概代表 500GB、`3.00` 大概代表 3TB。较高级桶的权重是所有枝叶桶的权重之和。

A bucket item weight is one dimensional, but you may also calculate your item weights to reflect the performance of the storage drive. For example, if you have many 1TB drives where some have relatively low data transfer rate and the others have a relatively high data transfer rate, you may weight them differently, even though they have the same capacity (e.g., a weight of `0.80` for the first set of drives with lower total throughput, and `1.20` for the second set of drives with higher total throughput).

一个桶的权重是一维的，你也可以计算条目权重来反映存储设备性能。例如，如果你有很多 1TB 的硬盘，其中一些数据传输速率相对低、其他的数据传输率相对高，即使它们容量相同，也应该设置不同的权重（如给吞吐量较低的硬盘设置权重 `0.8`，较高的设置 `1.2`）。

3.3.9.2.4 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`。

CRUSH rules defines placement and replication strategies or distribution policies that allow you to specify exactly how CRUSH places object replicas. For example, you might create a rule selecting a pair of targets for 2-way mirroring, another rule for selecting three targets in two different data centers for 3-way mirroring, and yet another rule for RAID-4 over six storage devices. For a detailed discussion of CRUSH rules, refer to [CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data](#), and more specifically to [Section 3.2](#).

CRUSH 规则定义了归置和复制策略、或分布策略，用它可以规定 CRUSH 如何放置对象副本。例如，你也许想创建一条规则用以选择一对目的地做双路复制；另一条规则用以选择位于两个数据中心的三个目的地做三路镜像；又一条规则用 6 个设备做 RAID-4。关于 CRUSH 规则的详细研究见 [CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data](#)，主要是 Section 3.2。

A rule takes the following form:

规则格式如下：

```

rule <rulename> {

    ruleset <ruleset>
    type [ replicated | raid4 ]
    min_size <min-size>
    max_size <max-size>
    step take <bucket-type>
    step [choose|chooseleaf] [firstn|indep] <N> <bucket-type>
    step emit
}

```

ruleset

Description: A means of classifying a rule as belonging to a set of rules. Activated by [setting the ruleset in a pool](#).

区分一条规则属于某个规则集的手段。在存储池里设置 ruleset 后激活。

Purpose: A component of the rule mask.
规则掩码的一个组件。

Type: Integer
Required: Yes
Default: 0

type

Description: Describes a rule for either a storage drive (replicated) or a RAID.
为硬盘（复制的）或 RAID 写一条规则。

Purpose: A component of the rule mask.
Type: String
Required: Yes
Default: replicated

Valid Values: Currently only replicated

min_size

Description: If a pool makes fewer replicas than this number, CRUSH will NOT select this rule.
如果一个归置组副本数小于此数，CRUSH 将不应用此规则。

Type: Integer
Purpose: A component of the rule mask.
Required: Yes
Default: 1

max_size

Description: If a pool makes more replicas than this number, CRUSH will NOT select this rule.
如果一个归置组副本数大于此数，CRUSH 将不应用此规则。

Type: Integer
Purpose: A component of the rule mask.
Required: Yes
Default: 10

step take <bucket-type>

Description: Takes a bucket name, and begins iterating down the tree.
选取桶名并类推到树底。

Purpose: A component of the rule.
Required: Yes
Example: step take data

step choose firstn {num} type {bucket-type}

Description: Selects the number of buckets of the given type. The number is usually the number of replicas in the pool (i.e., pool size).

- If {num} == 0, choose pool-num-replicas buckets (all available).
- If {num} > 0 && < pool-num-replicas, choose that many buckets.
- If {num} < 0, it means pool-num-replicas - {num}.

选取指定类型桶的数量，这个数字通常是存储池的副本数（如 pool size）。

- 如果{num}==0 选择 pool-num-replicas 个桶（所有可用的）。

- 如果`{num}>0 && <pool-num-replicas`就选择那么多的桶;
- 如果`{num}<0`它意为`pool-num-replicas-{num}`;

Purpose: A component of the rule.

Prerequisite: Follows step take or step choose.

Example: `step choose firstn 1 type row`

step chooseleaf firstn {num} type {bucket-type}

Description: Selects a set of buckets of `{bucket-type}` and chooses a leaf node from the subtree of each bucket in the set of buckets. The number of buckets in the set is usually the number of replicas in the pool (i.e., pool size).

选择`{bucket-type}`类型的一堆桶，并从各桶的子树里选择一个叶子节点。集合内桶的数量通常是存储池的副本数（如 pool size）。

- If `{num} == 0`, choose `pool-num-replicas` buckets (all available).
- If `{num} > 0 && < pool-num-replicas`, choose that many buckets.
- If `{num} < 0`, it means `pool-num-replicas - {num}`.

Purpose: A component of the rule. Usage removes the need to select a device using two steps.

规则的一组件。它的使用避免了通过两步来选择一设备。

Prerequisite: Follows step take or step choose.

Example: `step chooseleaf firstn 0 type row`

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 pick from different trees in the same rule.

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。

Purpose: A component of the rule.

规则组件。

Prerequisite: Follows step choose.

Example: `step emit`

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 不同存储池置于不同 OSD

Placing Different Pools on Different OSDs:

Suppose you want to have most pools default to OSDs backed by large hard drives, but have some pools mapped to OSDs backed by fast solid-state drives (SSDs). It's possible to have multiple independent CRUSH hierarchies within the same CRUSH map. Define two hierachies with two different root nodes—one for hard disks (e.g., "root platter") and one for SSDs (e.g., "root ssd") as shown below:

假设你想让大多数存储池坐落于使用大硬盘的 OSD 上，但是其中一些存储池映射到使用高速 SSD 的 OSD 上。在同一个 CRUSH 图内有多个独立的 CRUSH 树是可能的，定义两棵树、分别有自己的根节点——一个用于硬盘（如 root platter）、一个用于 SSD（如 root ssd），如：

```
device 0 osd.0
device 1 osd.1
device 2 osd.2
device 3 osd.3
device 4 osd.4
device 5 osd.5
device 6 osd.6
device 7 osd.7

host ceph-osd(ssd-server-1 {
    id -1
    alg straw
    hash 0
    item osd.0 weight 1.00
    item osd.1 weight 1.00
}
```

```

host ceph-osd(ssd)-server-2 {
    id -2
    alg straw
    hash 0
    item osd.2 weight 1.00
    item osd.3 weight 1.00
}

host ceph-osd(platter)-server-1 {
    id -3
    alg straw
    hash 0
    item osd.4 weight 1.00
    item osd.5 weight 1.00
}

host ceph-osd(platter)-server-2 {
    id -4
    alg straw
    hash 0
    item osd.6 weight 1.00
    item osd.7 weight 1.00
}

root platter {
    id -5
    alg straw
    hash 0
    item ceph-osd-platter-server-1 weight 2.00
    item ceph-osd-platter-server-2 weight 2.00
}

root ssd {
    id -6
    alg straw
    hash 0
    item ceph-osd(ssd)-server-1 weight 2.00
    item ceph-osd(ssd)-server-2 weight 2.00
}

rule data {
    ruleset 0
    type replicated
    min_size 2
    max_size 2
    step take platter
    step chooseleaf firstn 0 type host
    step emit
}

rule metadata {
    ruleset 1
    type replicated
    min_size 0
    max_size 10
    step take platter
    step chooseleaf firstn 0 type host
    step emit
}

rule rbd {
    ruleset 2
    type replicated
    min_size 0
    max_size 10
    step take platter
    step chooseleaf firstn 0 type host
    step emit
}

rule platter {
    ruleset 3
    type replicated
    min_size 0
}

```

```

        max_size 10
        step take platter
        step chooseleaf firstn 0 type host
        step emit
    }

    rule ssd {
        ruleset 4
        type replicated
        min_size 0
        max_size 10
        step take ssd
        step chooseleaf firstn 0 type host
        step emit
    }

    rule ssd-primary {
        ruleset 4
        type replicated
        min_size 0
        max_size 10
        step take ssd
        step chooseleaf firstn 1 type host
        step emit
        step take platter
        step chooseleaf firstn -1 type host
        step emit
    }
}

```

You can then set a pool to use the SSD rule by:

然后你可以设置一个存储池，让它使用 SSD 规则：

```
ceph osd pool set <poolname> crush_ruleset 4
```

Similarly, using the `ssd-primary` rule will cause each placement group in the pool to be placed with an SSD as the primary and platters as the replicas.

同样，用 `ssd-primary` 规则将使存储池内的各归置组用 SSD 作主 OSD，普通硬盘作副本。

3.3.9.4 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 `ceph osd crush set`. For Argonaut (v 0.48), execute the following:

要增加或删除在线集群里 OSD 所对应的 CRUSH 图条目，执行 `ceph osd crush set` 命令。对于 v0.48 版，执行下列：

```
ceph osd crush set {id} {name} {weight} pool={pool-name} [{bucket-type}={bucket-name} ...]
```

For Bobtail (v 0.56), execute the following:

bobtail (v0.56) 可执行下列：

```
ceph osd crush set {id-or-name} {weight} root={pool-name} [{bucket-type}={bucket-name} ...]
```

Where:

其中：

`id`

Description: The numeric ID of the OSD.
OSD 的数字标识符。

Type: Integer

Required: Yes

Example: 0

`name`

Description: The full name of the OSD.
OSD 的全名。

Type: String
Required: Yes
Example: osd.0

weight

Description: The CRUSH weight for the OSD.
OSD 的 CRUSH 权重。

Type: Double
Required: Yes
Example: 2.0

root

Description: The root of the tree in which the OSD resides.
OSD 所在树的根。

Type: Key/value pair.
Required: Yes
Example: root=default

bucket-type

Description: You may specify the OSD's location in the CRUSH hierarchy.
定义 OSD 在 CRUSH 分级结构中的位置。

Type: Key/value pairs.
Required: No
Example: datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1

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 osd.0 1.0 root=default datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1
```

3.3.9.5 调整一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:

要调整在线集群中一OSD 的CRUSH 权重，执行命令：

```
ceph osd crush reweight {name} {weight}
```

Where:

其中：

name

Description: The full name of the OSD.
OSD 的全名。

Type: String
Required: Yes
Example: osd.0

weight

Description: The CRUSH weight for the OSD.
OSD 的 CRUSH 权重。

Type: Double
Required: Yes
Example: 2.0

3.3.9.6 删除OSD

REMOVE AN OSD

To remove an OSD from the CRUSH map of a running cluster, execute the following:

要在在线集群里把一 OSD 踢出 CRUSH 图，执行命令：

```
ceph osd crush remove {name}
```

Where:

其中：

name

Description: The full name of the OSD.

OSD 全名。

Type: String

Required: Yes

Example: osd.0

3.3.9.7 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:

其中：

bucket-name

Description: The name of the bucket to move/reposition.

要移动或复位的桶名。

Type: String

Required: Yes

Example: foo-bar-1

bucket-type

Description: You may specify the bucket's location in the CRUSH hierarchy.

你可以指定桶在 CRUSH 分级结构里的位置。

Type: Key/value pairs.

Required: No

Example: datacenter=dc1 room=room1 row=foo rack=bar host=foo-bar-1

3.3.9.8 可调选项

TUNABLES

New in version 0.48.

0.48 版加入。

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) and `CRUSH_TUNABLES2` to reflect support for tunables.

最近发布的 ceph 允许在 CRUSH 图里使用可调值，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 `CRUSH_TUNABLES` (值 0x40000) 和 `CRUSH_TUNABLES2` 来反映支持可调值。

If the OSDMap currently used by the `ceph-mon` or `ceph-osd` daemon has non-legacy values, it will require the `CRUSH_TUNABLES` or `CRUSH_TUNABLES2` 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` 或 `CRUSH_TUNABLES2` 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。

3.3.9.8.1 过时值的影响

IMPACT OF LEGACY VALUES

The legacy values result in several misbehaviors:

遗留值导致几个不当行为：

- For hierarchies 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 hierarchies 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).

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。
- When some OSDs are marked out, the data tends to get redistributed to nearby OSDs instead of across the entire hierarchy.

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。

3.3.9.8.2 CRUSH_TUNABLES

- `choose_local_tries`: Number of local retries. Legacy value is 2, optimal value is 0.

`choose_local_tries`: 本地重试次数。以前是 2，最优值是 0。
- `choose_local_fallback_tries`: Legacy value is 5, optimal value is 0.

`choose_local_fallback_tries`: 以前 5，现在是 0。
- `choose_total_tries`: Total number of attempts to choose an item. Legacy value was 19, subsequent testing indicates that a value of 50 is more appropriate for typical clusters. For extremely large clusters, a larger value might be necessary.

`choose_total_tries`: 选择一个条目的最大尝试次数。以前 19，后来的测试表明，对典型的集群来说 50 更合适。最相当大的集群来说，更大的值也许必要。

3.3.9.8.3 CRUSH_TUNABLES2

- `chooseleaf_descend_once`: Whether a recursive chooseleaf attempt will retry, or only try once and allow the original placement to retry. Legacy default is 0, optimal value is 1.

`chooseleaf_descend_once`: 是否重递归叶子选择，或只试一次、并允许最初归置组重试。以前默认 0，最优为 1。

3.3.9.8.4 支持 CRUSH_TUNABLES 的客户端版本

Which client versions support CRUSH_TUNABLES

哪些客户端版本支持 CRUSH_TUNABLES。

- argonaut series, v0.48.1 or later
- v0.49 or later
- Linux kernel version v3.6 or later (for the file system and RBD kernel clients)

3.3.9.8.5 支持 CRUSH_TUNABLES2 的客户端版本

Which client versions support CRUSH_TUNABLES2

哪些客户端版本支持 CRUSH_TUNABLES2。

- v0.55 or later, including bobtail series (v0.56.x)
- Linux kernel version v3.9 or later (for the file system and RBD kernel clients)

3.3.9.8.6 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.
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，做好移动一部分数据的准备。
- The `ceph-osd` and `ceph-mon` daemons will start requiring the feature bits 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.
一旦更新运行图，`ceph-osd` 和 `ceph-mon` 就会开始向新建连接要求功能位，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 default 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 previously 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。

3.3.9.8.7 调整 CRUSH

TUNING CRUSH

The simplest way to adjust the crush tunables is by changing to a known profile. Those are:

更改 crush 可调值的最简方法就是改到一个已知配置，它们有：

- `legacy`: the legacy behavior from argonaut and earlier.
- `argonaut`: the legacy values supported by the original argonaut release
- `bobtail`: the values supported by the bobtail release
- `optimal`: the current best values
- `default`: the current default values for a new cluster

Currently, `legacy`, `default`, and `argonaut` are the same, and `bobtail` and `optimal` include CRUSH_TUNABLES and CRUSH_TUNABLES2.

当前，`legacy`、`default`、和 `argonaut` 相同；`bobtail` 和 `optimal` 包含 CRUSH_TUNABLES 和 CRUSH_TUNABLES2。

You can select a profile on a running cluster with the command:

你可以在运行着的集群上选择一个配置：

```
ceph osd crush tunables {PROFILE}
```

Note that this may result in some data movement.

要注意，这可能产生一些数据迁移。

3.3.9.8.8 调整CRUSH——强硬方法

Tuning CRUSH, the hard way

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
```

3.3.9.9 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 增加/删除OSD

Adding/Removing OSDs

When you have a cluster up and running, you may add OSDs or remove OSDs from the cluster at runtime.

如果您的集群已经在运行，你可以在运行时添加或删除OSD。

3.3.10.1 增加OSD

Adding OSDs

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 drive within a host machine. If your host has multiple storage drives, you may map one `ceph-osd` daemon for each drive.

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。

3.3.10.1.1 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 drives. 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](#)。

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。

3.3.10.1.2 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.

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 chef 用户、配置 SSH 密钥对、安装 Ruby、安装 chef 客户端，详情参见 [Installing Chef](#)。

3.3.10.1.3 增加 OSD (手动)

ADDING AN OSD (MANUAL)

This procedure sets up an `ceph-osd` daemon, configures it to use one drive, and configures the cluster to distribute data to the OSD. If your host has multiple drives, you may add an OSD for each drive 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 drive 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 drive capacity grows 40% per year, so newer OSD hosts may have larger hard drive 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)。

1. Create the OSD. If no UUID is given, it will be set automatically when the OSD starts up. The following command will output the OSD number, which you will need for subsequent steps.

创建 OSD。如果未指定 UUID，OSD 启动时会自动生成一个。下列命令会输出 OSD 号，后续步骤你会

用到。

```
ceph osd create [{uuid}]
```

2. 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}
```

3. If the OSD is for a drive other than the OS drive, prepare it for use with Ceph, and mount it to the directory you just created:

如果准备用于 OSD 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}
```

4. 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
```

5. Add the new OSD to your ceph.conf file.

把新 OSD 添加到 ceph.conf 文件里：

```
[osd.1]
host = {hostname}
```

6. 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
```

7. 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。

8. 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
```

9. 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](#)。

For Argonaut (v 0.48), execute the following:

若用的是 v0.48 版，执行下列命令：

```
ceph osd crush set {id} {name} {weight} pool={pool-name} [{bucket-type}={bucket-name} ...]
```

For Bobtail (v 0.56), execute the following:

若用的是 v0.56 版，执行下列命令：

```
ceph osd crush set {id-or-name} {weight} root={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。

3.3.10.1.4 增加 OSD (chef)

ADDING AN OSD (CHEF)

This procedure configures your OSD using `chef-client`. If your host has multiple drives, you may need to execute the procedure for preparing an OSD drive for each data drive on your host.

此步骤用 `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 drive capacity grows 40% per year, so newer OSD hosts may have larger hard drive 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）。

1. Execute `chef-client` to register it with Chef as a Chef node.

执行 `chef-client` 来注册为 chef 节点。

2. 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 [Configure Nodes](#)。进入 chef 环境，把 "`role[ceph-osd]`" 添加到运行列表。

3. Execute [Prepare OSD Drives](#) for each drive.

在每个硬盘上执行 Prepare OSD Disks。

4. 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 {name} {weight} [{bucket-type}={bucket-name} ...]
```

3.3.10.1.5 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**。你必须先启动 OSD 它才能收数据。可以用管理主机上的 `service ceph`、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。

3.3.10.1.6 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}
```

3.3.10.1.7 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 **degraded**（降级）的对象、且迁移完成后回到 **active+clean** 状态。（Ctrl-c 退出）

3.3.10.2 删除 OSD (手动)

Removing OSDs (Manual)

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 drive within a host machine. If your host has multiple storage drives, you may need to remove one `ceph-osd` daemon for each drive. 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 `full ratio` 值，删除 OSD 可能导致集群达到或超过 `full ratio` 值。

3.3.10.2.1 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}
```

3.3.10.2.2 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 (`out`) 集群，ceph 就会开始重新均衡集群、把归置组迁出将删除的 OSD。你可以用 ceph 工具观察此过程。

```
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 (`degraded`) 对象、迁移完成后最终回到 `active+clean` 状态。 (Ctrl-c 中止)

3.3.10.2.3 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`。

3.3.10.2.4 删除 OSD

Removing the OSD

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 drives, you may need to remove an OSD for each drive 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。

1. 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}
```

2. 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。

3. Remove the OSD.

删除 OSD。

```
ceph osd rm {osd-num}
#for example
ceph osd rm 1
```

4. 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
```

5. Remove the OSD entry from your `ceph.conf` file.

从 `ceph.conf` 配置文件里删除对应条目。

```
[osd.1]
host = {hostname}
```

6. 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
```

3.3.11 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。

3.3.11.1 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。

3.3.11.1.1 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。

3.3.11.1.2 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.

对于 chef 部署的集群，先创建一个 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 [Installing Chef](#)。

3.3.11.1.3 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` 达到法定人数。

At this point you should define your monitor's id. Traditionally, monitors have been named with single letters (`a`, `b`, `c`, ...), but you are free to define the id as you see fit. For the purpose of this document, please take into account that `{mon-id}` should be the id you chose, without the `mon.` prefix (i.e., `{mon-id}` should be the `a` on `mon.a`).

现在该指定监视器的标识号了。传统上，监视器曾用单个字母（`a`、`b`、`c`……）命名，但你可以指定任何形式。在本文档里，要记住`{mon-id}`应该是你所选的标识号，不包含`mon.`前缀，如在`mon.a`中，其`{mon-id}`是`a`。

1. 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}
```

2. 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`{tmp}`，用以保存此过程中用到的文件。此目录要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}
```

3. 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}
```

4. 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}
```

5. 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}
```

6. 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
```

7. Add the new monitor to the list of monitors for your 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>]
```

8. 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}
```

3.3.11.2 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。必须有足够的监视器才能对集群运行图达成共识。

3.3.11.2.1 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，你得增加或删除一个监视器，直到凑足法定人数所必需的 `ceph-mon` 数。

1. Stop the monitor.

停止监视器。

```
service ceph -a stop mon.{mon-id}
```

2. Remove the monitor from the cluster.

从集群删除监视器。

```
ceph mon remove {mon-id}
```

3. Remove the monitor entry from ceph.conf.

删除 `ceph.conf` 对应条目。

3.3.11.2.2 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 unhealthy 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`。

1. Identify a surviving monitor and log in to that host.

找出活着的监视器，并登录其所在主机。

```
ceph mon dump  
ssh {mon-host}
```

2. Stop the ``ceph-mon'' daemon and extract a copy of the monap file.

停止 `ceph-mon` 守护进程并提取 `monmap` 副本。

```
service ceph stop mon || stop ceph-mon-all  
ceph-mon -i {mon-id} --extract-monmap {map-path}  
# for example,  
ceph-mon -i a --extract-monmap /tmp/monmap
```

3. 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:

删除非存活监视器。例如，如果你有 3 个监视器 `mon.a`、`mon.b` 和 `mon.c`，其中仅保留 `mon.a`，按如下步骤：

```
monmaptool {map-path} --rm {mon-id}  
# for example,  
monmaptool /tmp/monmap --rm b  
monmaptool /tmp/monmap --rm c
```

4. 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:

把去除了死监视器的残留运行图注入存活的监视器。比如，用下列命令把一张运行图注入 `mon.a` 监视器：

```
ceph-mon -i {mon-id} --inject-monmap {map-path}  
# for example,  
ceph-mon -i a --inject-monmap /tmp/monmap
```

3.3.11.3 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。

3.3.11.3.1 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 Paxos 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 inadvertently 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。

3.3.11.3.2 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。

3.3.11.3.3 更改监视器 IP 地址 (凌乱方法)

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。

3.3.12 命令参考

Control commands

3.3.12.1 监视器命令

MONITOR COMMANDS

Monitor commands are issued using the ceph utility:

监视器命令用 ceph 工具发出：

```
ceph [-m monhost] {command}
```

The command is usually (though not always) of the form:

命令格式通常是（但不总是）：

```
ceph {subsystem} {command}
```

3.3.12.2 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
```

3.3.12.3 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
```

3.3.12.4 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 plain (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“卡住了”（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
```

3.3.12.5 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 UUID is given, it will be set automatically when the OSD starts up.

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 [{uuid}]
```

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} [{pool-name} --yes-i-really-really-mean-it]  
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 uncommitted 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]
```

3.3.12.6 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。

```
ceph mds fail 0
```

Marks the active MDS as failed, triggering failover to a standby if present.

把活跃 MDS 标记为失败，如果有候补此命令会触发故障转移。

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。

3.3.12.7 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"}]}'
```

```
        "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
```

3.3.13 Ceph 社区

The Ceph Community

The Ceph community is an excellent source of information and help. For operational issues with Ceph releases we recommend you [subscribe to the ceph-users email list](#). When you no longer want to receive emails, you can [unsubscribe from the ceph-users email list](#).

ceph 社区是极好的信息和帮助来源。想了解与 ceph 发布相关的运维问题，建议您[订阅 ceph-users 邮件列表](#)，不再想收邮件时可以[取消订阅](#)。

If you have read through this guide and you have contacted [ceph-users](#), but you haven't resolved your issue, you may contact [Inktank](#) for support.

如果你已经读过本手册、也联系过 ceph-users，但是还没有解决问题，你可以联系 [Inktank](#) 寻求支持。

You may also [subscribe to the ceph-devel email list](#). You should do so if your issue is:

你也可以[订阅 ceph-devel 邮件列表](#)，遇到以下问题时你应该这样做：

- Likely related to a bug
可能和某个缺陷有关
- Related to a development release package
和某开发版有关
- Related to a development testing package
和某开发测试版有关
- Related to your own builds
和你自己的构建有关

If you no longer want to receive emails from the [ceph-devel](#) email list, you may [unsubscribe from the ceph-devel email list](#).

如果你不再想收 ceph-devel 的邮件，你可以[取消订阅](#)。

Tip: The Ceph community is growing rapidly, and community members can help you if you provide them with detailed information about your problem. You can attach your ceph configuration file, log files, CRUSH map, and other details (e.g., `ceph osd tree`) to help people understand your issues.

提示：ceph 社区在快速成长，如果你提供了问题的详情社区成员就可以帮你解决。你可以附加你的 ceph 配置文件、日志文件、CRUSH 图以及其它细节（如 `ceph osd tree`），以便人们了解你的问题。

3.3.14 监视器故障恢复

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。

3.3.14.1 客户端不能连接/挂载

Client Can't Connect/Mount

Check your IP tables. Some OS install utilities add a REJECT rule to `iptables`. The rule rejects all clients trying to connect to the host except for `ssh`. If your monitor host's IP tables have such a REJECT rule in place, clients connecting from a separate node will fail to mount with a timeout error. You need to address `iptables` rules that reject clients trying to connect to Ceph daemons. For example, you would need to address rules that look like this appropriately:

检查防火墙配置。有些系统安装工具把 REJECT 规则加入了 `iptables`，它会拒绝除 `ssh` 以外的所有入栈连接。如果你的监视器主机有这样的 REJECT 规则，别的客户端进来的连接将遇到超时错误而不能挂载。得先找到这条拒绝客户端连入的 `iptables` 规则，例如，你要找到形似以下的规则：

```
REJECT all -- anywhere anywhere reject-with icmp-host-prohibited
```

You may also need to add rules to IP tables on your Ceph hosts to ensure that clients can access the ports associated with your Ceph monitors (i.e., port 6789 by default) and Ceph OSDs (i.e., 6800 et. seq. by default). For example:

你也许还要在 `ceph` 主机上增加 `iptables` 规则来放通 `ceph` 监视器端口（如默认的 6789 端口）、和 OSD 端口（如默认从 6800 开始的一系列端口）。例如：

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

3.3.14.2 挂掉监视器时的延时

Latency with Down Monitors

When you have a monitor that is down, you may experience some latency as clients will try to connect to a monitor in the configuration even though it is down. If the client fails to connect to the monitor within a timeout window, the client will try another monitor in the cluster.

当某个监视器挂了时，你可能感觉到延时增大了，因为客户端仍然会试图连接配置文件里的监视器，即使它挂掉了。如果在超时窗口内客户端没能连到监视器，客户端会尝试集群内的另一监视器。

You can also specify the `-m` option to point to a monitor that is up and in the quorum to avoid latency.

你也能用 `-m` 选项指定活着且健康的监视器，以避免延时。

3.3.15 OSD 故障排除

Troubleshooting OSDs

Before troubleshooting your OSDs, check your monitors and network first. If you execute `ceph health` or `ceph -s` on the command line and Ceph returns a health status, the return of a status means that the monitors have a quorum. If you don't have a monitor quorum or if there are errors with the monitor status, address the monitor issues first. Check your networks to ensure they are running properly, because networks may have a significant impact on OSD operation and performance.

进行 OSD 排障前，首先检查监视器和网络。如果 `ceph health` 或 `ceph -s` 返回的状态是 `health`，这意味着监视器形成了法定人数。如果你还没监视器法定人数、或者监视器状态错误，要先解决监视器问题。核实下你的网络，确保它在正常运行，因为网络对 OSD 运行和性能有显著影响。

3.3.15.1 收集 OSD 数据

Obtaining Data About OSDs

A good first step in troubleshooting your OSDs is to obtain information in addition to the information you collected while [monitoring your OSDs](#) (e.g., `ceph osd tree`).

开始 OSD 排障的第一步最好先获取些额外监控信息，如 `ceph osd tree`。

3.3.15.1.1 ceph 日志

Ceph Logs

If you haven't changed the default path, you can find Ceph log files at `/var/log/ceph`:

如果你没改默认路径，可以在 `/var/log/ceph` 下找到日志：

```
ls /var/log/ceph
```

If you don't get enough log detail, you can change your logging level. See [Ceph Logging and Debugging](#) and [Logging and Debugging Config Reference](#) in the Ceph Configuration documentation for details. Also, see [Debugging and Logging](#) in the Ceph Operations documentation to ensure that Ceph performs adequately under high logging volume.

如果你没有足够的细节日志，可以增大日志级别。详情见 `ceph` 配置文档里的 [Ceph Logging and Debugging](#) 和 [Logging and Debugging Config Reference](#)，确保 `ceph` 在大日志量情况下仍能平稳运行。

3.3.15.1.2 管理套接字

Admin Socket

Use the admin socket tool to retrieve runtime information. For details, list the sockets for your Ceph processes:

用管理套接字检索运行时信息。所有 `ceph` 套接字列表：

```
ls /var/run/ceph
```

Then, execute the following, replacing `{socket-name}` with an actual socket name to show the list of available options:

然后，执行下例命令显示可用选项，用实际的套接字名字取代 `{socket-name}`：

```
ceph --admin-daemon /var/run/ceph/{socket-name} help
```

The admin socket, among other things, allows you to:

和其它手段相比，管理接口允许你：

- List your configuration at runtime
- Dump historic operations
- Dump the operation priority queue state

- Dump operations in flight
- Dump perfcounters

- 在运行时列出配置
- 列出历史操作
- 列出操作的优先队列状态
- 列出正在进行的操作
- 列出性能计数器

3.3.15.1.3 显示剩余空间

Display Freespace

Filesystem issues may arise. To display your filesystem's free space, execute `df`.

可能是文件系统问题，用 `df` 显示文件系统剩余空间。

```
df -h
```

Execute `df --help` for additional usage.

其它用法见 `df --help`。

3.3.15.1.4 I/O 统计信息

I/O Statistics

Use `iostat` to identify I/O-related issues.

用 `iostat` 定位 I/O 相关问题。

```
iostat -x
```

3.3.15.1.5 诊断消息

Diagnostic Messages

To retrieve diagnostic messages, use `dmesg` with `less`, `more`, `grep` or `tail`. For example:

要查看诊断信息，配合 `less`、`more`、`grep` 或 `tail` 使用 `dmesg`，例如：

```
dmesg | grep scsi
```

3.3.15.2 停止自动重均衡

Stopping w/out Rebalancing

Periodically, you may need to perform maintenance on a subset of your cluster, or resolve a problem that affects a failure domain (e.g., a rack). If you do not want CRUSH to automatically rebalance the cluster as you stop OSDs for maintenance, set the cluster to `noout` first:

你得周期性地维护集群的子系统、或解决某个失败域的问题（如一机架）。如果你不想在停机维护 OSD 时让 CRUSH 自动重均衡，提前设置 `noout`：

```
ceph osd set noout
```

Once the cluster is set to `noout`, you can begin stopping the OSDs within the failure domain that requires maintenance work.

在集群上设置 `noout` 后，你就可以停机维护失败域内的 OSD 了。

```
ceph osd stop osd.{num}
```

Note: Placement groups within the OSDs you stop will become degraded while you are addressing issues within the failure domain.

注意：在定位同一故障域内的问题时，停机 OSD 内的归置组其状态会变为 `degraded`。

Once you have completed your maintenance, restart the OSDs.

维护结束后，重启 OSD。

```
ceph osd start osd.{num}
```

Finally, you must unset the cluster from `noout`.

最后，解除 `noout` 标志。

```
ceph osd unset noout
```

3.3.15.3 OSD 没运行

OSD Not Running

Under normal circumstances, simply restarting the `ceph-osd` daemon will allow it to rejoin the cluster and recover.

通常情况下，简单地重启 `ceph-osd` 进程就可以重回集群并恢复。

3.3.15.3.1 OSD 起不来

An OSD Won't Start

If you start your cluster and an OSD won't start, check the following:

如果你重启了集群，但其中一个 OSD 起不来，依次检查：

- **Configuration File:** If you were not able to get OSDs running from a new installation, check your configuration file to ensure it conforms (e.g., `host` not `hostname`, etc.).

配置文件：如果你新装的 OSD 不能启动，检查下配置文件，确保它合义性（比如 `host` 而非 `hostname`，等等）；

- **Check Paths:** Check the paths in your configuration, and the actual paths themselves for data and journals. If you separate the OSD data from the journal data and there are errors in your configuration file or in the actual mounts, you may have trouble starting OSDs. If you want to store the journal on a block device, you should partition your journal disk and assign one partition per OSD.

检查路径：检查你配置的路径，以及它们自己的数据和日志路径。如果你分离了 OSD 数据和日志、而配置文件和实际挂载点存在出入，启动 OSD 时就可能遇挫。如果你想把日志存储于一个块设备，应该为日志硬盘分区并为各 OSD 分别分配一分区。

- **Kernel Version:** Identify the kernel version and distribution you are using. Ceph uses some third party tools by default, which may be buggy or may conflict with certain distributions and/or kernel versions (e.g., Google perfetto). Check the [OS recommendations](#) to ensure you have addressed any issues related to your kernel.

内核版本：确认你在用的内核版本以及所用的发布版。`ceph` 默认依赖一些第三方工具，这些工具可能有缺陷或者与特定发布版和/或内核版本冲突（如 Google perfetto）。检查下 [OS recommendations](#) 确保你已经解决了内核相关的问题。

- **Segment Fault:** If there is a segment fault, turn your logging up (if it isn't already), and try again. If it segment faults again, contact the `ceph-devel` email list and provide your Ceph configuration file, your monitor output and the contents of your log file(s).

段错误：如果有了段错误，提高日志级别（如果还没提高），再试试。如果重现了，联系 `ceph-devel` 并提供你的配置文件、显示器输出和日志文件内容。

If you cannot resolve the issue and the email list isn't helpful, you may contact [Inktank](#) for support.

如果问题仍未解决，email 也无用，你可以联系 Inktank 寻求帮助。

3.3.15.3.2 OSD 失败

An OSD Failed

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
```

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`.

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 [ceph-devel](#) email 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 `ceph-devel` 邮件列表。

3.3.15.3.3 硬盘没剩余空间

No Free Drive Space

Ceph prevents you from writing to a full OSD so that you don't lose data. In an operational cluster, you should receive a warning when your cluster is getting near its full ratio. The `mon osd full ratio` defaults to `0.95`, or 95% of capacity before it stops clients from writing data. The `mon osd nearfull ratio` defaults to `0.85`, or 85% of capacity when it generates a health warning.

`ceph` 不允许你向满的 OSD 写入数据，以免丢失数据。在运营着的集群中，你应该能收到集群空间将满的警告。`mon osd full ratio` 默认为 `0.95`、或达到 95% 时它将阻止客户端写入数据。`mon osd nearfull ratio` 默认为 `0.85`、也就是说达到容量的 85% 时它会产生健康警告。

Full cluster issues usually arise when testing how Ceph handles an OSD failure on a small cluster. When one node has a high percentage of the cluster's data, the cluster can easily eclipse its nearfull and full ratio immediately. If you are testing how Ceph reacts to OSD failures on a small cluster, you should leave ample free disk space and consider temporarily lowering the `mon osd full ratio` and `mon osd nearfull ratio`.

满载集群问题一般产生于测试 `ceph` 在小型集群上如何处理 OSD 失败时。当某一节点利用率较高时，集群能够很快掩盖将满和占满率。如果你在测试小型集群上的 `ceph` 如何应对 OSD 失败，应该保留足够的空间，然后试着临时降低 `mon osd full ratio` 和 `mon osd nearfull ratio` 值。

Full `ceph-osds` will be reported by `ceph health`:

`ceph health` 会显示将满的 `ceph-osds`：

```
ceph health
HEALTH_WARN 1 nearfull osds
osd.2 is near full at 85%
```

Or:

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。

If you cannot start an OSD because it is full, you may delete some data by deleting some placement group directories in the full OSD.

如果因满载而导致 OSD 不能启动，你可以试着删除那个 OSD 上的一些归置组数据目录。

Important: If you choose to delete a placement group directory on a full OSD, **DO NOT** delete the same placement group directory on another full OSD, or **YOU MAY LOSE DATA**. You **MUST** maintain at least one copy of your data on at least one OSD.

重要: 如果你准备从满载 OSD 中删除某个归置组，注意不要删除另一个 OSD 上的同名归置组，否则**你会丢数据**。必须在多个 OSD 上保留至少一份数据副本。

See [Monitor Config Reference](#) for additional details.

额外细节参见 [Monitor Config Reference](#)。

3.3.15.4 OSD 龟速或无响应

OSDs are Slow/Unresponsive

A commonly recurring issue involves slow or unresponsive OSDs. Ensure that you have eliminated other troubleshooting possibilities before delving into OSD performance issues. For example, ensure that your network(s) is working properly and your OSDs are running. Check to see if OSDs are throttling recovery traffic.

一个反复出现的问题是龟速或无响应。在深入性能问题前，你应该先确保不是其他故障。例如，确保你的网络运行正常、且 OSD 在运行，还要检查 OSD 是否被恢复流量拖住了。

Tip: Newer versions of Ceph provide better recovery handling by preventing recovering OSDs from using up system resources so that `up` and `in` OSDs aren't available or are otherwise slow.

提示: 较新版本的 ceph 能更好地处理恢复，可防止恢复进程耗尽系统资源而导致 `up` 且 `in` 的 OSD 不可用或响应慢。

3.3.15.4.1 网络问题

Networking Issues

Ceph is a distributed storage system, so it depends upon networks to peer with OSDs, replicate objects, recover from faults and check heartbeats. Networking issues can cause OSD latency and flapping OSDs. See [Flapping OSDs](#) for details.

ceph 是一个分布式存储系统，所以它依赖于网络来互联 OSD 们、复制对象、恢复错误、和检查心跳。网络问题会导致 OSD 延时和打摆子，详情参见 [Flapping OSDs](#)。

Ensure that Ceph processes and Ceph-dependent processes are connected and/or listening.

确保 ceph 进程和 ceph 依赖的进程连接了、和/或在监听。

```
netstat -a | grep ceph  
netstat -l | grep ceph  
sudo netstat -p | grep ceph
```

Check network statistics.

检查网络状况。

```
netstat -s
```

3.3.15.4.2 驱动器配置

Drive Configuration

A storage drive should only support one OSD. Sequential read and sequential write throughput can bottleneck if other processes share the drive, including journals, operating systems, monitors, other OSDs and non-Ceph processes.

一个存储驱动器应该只用于一个 OSD。如果有其它进程共享驱动器，顺序读和顺序写吞吐量会成为瓶颈，包

括日志记录、操作系统、监视器、其它 OSD 和非 ceph 进程。

Ceph acknowledges writes **after** journaling, so fast SSDs are an attractive option to accelerate the response time—particularly when using the **ext4** or **XFS** filesystems. By contrast, the **btrfs** filesystem can write and journal simultaneously.

ceph 在日志记录完成之后才会确认写操作，所以使用 **ext4** 或 **xfs** 文件系统时高速的 SSD 对降低响应延时很有吸引力。相反，**btrfs** 文件系统可以同时读写。

Note: Partitioning a drive does not change its total throughput or sequential read/write limits. Running a journal in a separate partition may help, but you should prefer a separate physical drive.

注意：给驱动器分区并不能改变总吞吐量或顺序读写限制。把日志分离到单独的分区可能有帮助，但最好是另外一块硬盘的分区。

3.3.15.4.3 坏扇区/碎片化硬盘

Bad Sectors / Fragmented Disk

Check your disks for bad sectors and fragmentation. This can cause total throughput to drop substantially.

检修下硬盘是否有坏扇区和碎片。这会导致总吞吐量极大下降。

3.3.15.4.4 监视器和 OSD 蜗居

Co-resident Monitors/OSDs

Monitors are generally light-weight processes, but they do lots of **fsync()**, which can interfere with other workloads, particularly if monitors run on the same drive as your OSDs. Additionally, if you run monitors on the same host as the OSDs, you may incur performance issues related to:

监视器是普通的轻量级进程，但它们会频繁调用 **fsync()**，这会妨碍其它工作量，特别是监视器和 OSD 共享驱动器时。另外，如果你在 OSD 主机上同时运行监视器，遭遇的性能问题可能和这些相关：

- Running an older kernel (pre-3.0)
运行较老的内核（3.0 之前）
- Running Argonaut with an old **glibc**
v0.48 版运行在老的 **glibc** 之上
- Running a kernel with no syncfs(2) syscall.
运行的内核没有 syncfs(2) 系统调用

In these cases, multiple OSDs running on the same host can drag each other down by doing lots of commits. That often leads to the bursty writes.

在这些情况下，同一主机上的多个 OSD 会相互拖垮对方。它们经常导致爆炸式写入。

3.3.15.4.5 进程蜗居

Co-resident Processes

Spinning up co-resident processes such as a cloud-based solution, virtual machines and other applications that write data to Ceph while operating on the same hardware as OSDs can introduce significant OSD latency. Generally, we recommend optimizing a host for use with Ceph and using other hosts for other processes. The practice of separating Ceph operations from other applications may help improve performance and may streamline troubleshooting and maintenance.

共存于同一套硬件、并向 ceph 写入数据的进程（像基于云的解决方案、虚拟机和其他应用程序）会导致 OSD 延时大增。一般来说，我们建议用一主机跑 ceph、其它主机跑其它进程，实践证明把 ceph 和其他应用程序分开可提高性能、并简化故障排除和维护。

3.3.15.4.6 日志级别

Logging Levels

If you turned logging levels up to track an issue and then forgot to turn logging levels back down, the OSD may be putting a lot of logs onto the disk. If you intend to keep logging levels high, you may consider

mounting a drive to the default path for logging (i.e., `/var/log/ceph/$cluster-$name.log`).

如果你为追踪某问题提高过日志级别、但结束后忘了调回去，这个 OSD 将向硬盘写入大量日志。如果你想始终保持高日志级别，可以考虑给默认日志路径挂载个硬盘，如`/var/log/ceph/$cluster-$name.log`。

3.3.15.4.7 恢复节流

Recovery Throttling

Depending upon your configuration, Ceph may reduce recovery rates to maintain performance or it may increase recovery rates to the point that recovery impacts OSD performance. Check to see if the OSD is recovering.

根据你的配置，ceph 可以降低恢复速度来维持性能，否则它会不顾 OSD 性能而加快恢复速度。检查下 OSD 是否正在恢复。

3.3.15.4.8 内核版本

Kernel Version

Check the kernel version you are running. Older kernels may not receive new backports that Ceph depends upon for better performance.

检查下你在用的内核版本。较老的内核也许没有移植能提高 ceph 性能的功能。

3.3.15.4.9 内核与 syncfs 问题

Kernel Issues with SyncFS

Try running one OSD per host to see if performance improves. Old kernels might not have a recent enough version of glibc to support syncfs(2).

试试在一主机上只运行一个 OSD，看看能否提升性能。老内核未必支持有 syncfs(2) 系统调用的 glibc。

3.3.15.4.10 文件系统问题

Filesystem Issues

Currently, we recommend deploying clusters with XFS or ext4. The btrfs filesystem has many attractive features, but bugs in the filesystem may lead to performance issues.

当前，我们推荐基于 xfs 或 ext4 部署集群。btrfs 有很多诱人的功能，但文件系统内的 bug 可能导致性能问题。

3.3.15.4.11 内存不足

Insufficient RAM

We recommend 1GB of RAM per OSD daemon. You may notice that during normal operations, the OSD only uses a fraction of that amount (e.g., 100-200MB). Unused RAM makes it tempting to use the excess RAM for co-resident applications, VMs and so forth. However, when OSDs go into recovery mode, their memory utilization spikes. If there is no RAM available, the OSD performance will slow considerably.

我们建议为每 OSD 进程规划 1GB 内存。你也许注意到了，通常情况下 OSD 仅会用一小部分（如 100-200MB）。你也许想用这些空闲内存跑一些其他应用，如虚拟机等等，然而当 OSD 进入恢复状态时，其内存利用率激增，如果没有可用内存，此 OSD 的性能将差的多。

3.3.15.4.12 old requests 或 slow requests

Old Requests or Slow Requests

If a ceph-osd daemon 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.

如果某 ceph-osd 守护进程对一请求响应很慢，它会生成日志消息来抱怨请求耗费的时间过长。默认警告阀值是 30 秒，用 `osd op complaint time` 选项来配置。这种情况发生时，集群日志系统会收到这些消息。

Legacy versions of Ceph complain about ‘old requests’:

很老的版本抱怨“old requests”：

```
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
```

New versions of Ceph complain about ‘slow requests’:

较新版本的 ceph 抱怨“slow requests”：

```
{date} {osd.num} [WRN] 1 slow requests, 1 included below; oldest blocked for > 30.005692 secs  
{date} {osd.num} [WRN] slow request 30.005692 seconds old, received at {date-time}:  
osd_op(client.4240.0:8 benchmark_data_ceph-1_39426_object7 [write 0~4194304] 0.69848840) v4 currently  
waiting for subops from [610]
```

Possible causes include:

可能起因有：

- A bad drive (check `dmesg` output)
坏驱动器（查看 dmesg 输出）；
- A bug in the kernel file system bug (check `dmesg` output)
内核文件系统缺陷（查看 dmesg 输出）；
- An overloaded cluster (check system load, iostat, etc.)
集群过载（检查系统负载、iostat 等等）；
- A bug in the `ceph-osd` daemon.
`ceph-osd` 守护进程缺陷。

Possible solutions

可能的解决方法：

- Remove VMs Cloud Solutions from Ceph Hosts
从 ceph 主机去除 VM 云解决方案；
- Upgrade Kernel
升级内核；
- Upgrade Ceph
升级 ceph；
- Restart OSDs
重启 OSD。

3.3.15.5 打摆子的 OSD

Flapping OSDs

We recommend using both a public (front-end) network and a cluster (back-end) network so that you can better meet the capacity requirements of object replication. Another advantage is that you can run a cluster network such that it isn’t connected to the internet, thereby preventing some denial of service attacks. When OSDs peer and check heartbeats, they use the cluster (back-end) network when it’s available. See [Monitor/OSD Interaction](#) for details.

我们建议同时部署公网（前端）和集群网（后端），这样能更好地满足对象复制的容量需求。另一个优点是你可以运营一个不连接互联网的集群，以此避免拒绝估计。OSD 们互联和检查心跳时会优选集群网（后端），详情见 [Monitor/OSD Interaction](#)。

However, if the cluster (back-end) network fails or develops significant latency while the public (front-end) network operates optimally, OSDs currently do not handle this situation well. What happens is that OSDs mark each other `down` on the monitor, while marking themselves `up`. We call this scenario ‘flapping’.

然而，如果集群网（后端）失败、或出现了明显的延时，同时公网（前端）却运行良好，OSD 现在不能很好地处理这种情况。这时 OSD 们会向监视器报告邻居 `down` 了、同时报告自己是 `up` 的，我们把这种情形称为打摆子（flapping）。

If something is causing OSDs to ‘flap’ (repeatedly getting marked down and then up again), you can force the monitors to stop the flapping with:

如果有东西导致 OSD 打摆子（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 nounp      # 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 nounp
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`，它们分别可防止 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: `noup`, `noout`, and `nodown` 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 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。

3.3.16 PG 错误排障

Troubleshooting PGs

3.3.16.1 归置组总不整洁

There are a few cases where Ceph placement groups never get clean:

在某些情况下，ceph 归置组一直不能整洁：

1. **One OSD:** If you deviate from the quick start and use only one OSD, you will likely run into problems. OSDs report other OSDs to the monitor, and also interact with other OSDs when replicating data. If you have only one OSD, a second OSD cannot check its heartbeat. Also, if you remove an OSD and have only one OSD remaining, you may encounter problems. A secondary or tertiary OSD expects another OSD to tell it which placement groups it should have. The lack of another OSD prevents this from occurring. So a placement group can remain stuck “stale” forever.

单个 OSD：如果你违背了快速入门，只用了一个 OSD，很可能碰到问题。OSD 向监视器报告其它 OSD 的动态，要复制数据时要与其它 OSD 交互，如果你只有一个 OSD、没有第二个 OSD 检查心跳，或者说你删除了一个 OSD、只剩下了一个 OSD 了，这时就会出现问题。第二或第三个 OSD 期望别的 OSD 告诉它应该持有哪些归置组，可是再没别的 OSD 了，所以它只能永远卡在“过时”状态。

2. **Pool Size = 1:** If you have only one copy of an object, no other OSD will tell the OSD which objects it should have. For each placement group mapped to the remaining OSD (see `ceph pg dump`), you can force the OSD to notice the placement groups it needs by running:

pool size = 1: 如果你只有某对象的一个副本，那就没其它 OSD 告诉它应该持有哪些对象。对于每个映射到活跃 OSD 的归置组（见 `ceph pg dump`），你都可以强制 OSD 通知它需要的归置组，运行：

```
ceph pg force_create_pg <pgid>
```

3. **CRUSH Rules:** Another candidate for placement groups remaining unclean involves errors in your CRUSH map.

CRUSH 规则：另一个使得归置组总停留在不洁状态的原因是 CRUSH 图中的错误。

As a general rule, you should run your cluster with more than one OSD and a pool size greater than 1 object replica.

作为常识，你应该用不止一个 OSD 和大于 1 的对象副本数构建集群。

3.3.16.2 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:

有失败时归置组会进入“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。我们具体检查：

- **inactive** - The placement group has not been **active** for too long (i.e., it hasn't been able to service read/write requests).
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** - The placement group has not been **clean** for too long (i.e., it hasn't been able to completely recover from a previous failure).
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** - The placement group status has not been updated by a **ceph-osd**, indicating that all nodes storing this placement group may be **down**.
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:

你可以摆出卡住的归置组：

```
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](#))；

卡在 **stale** 状态的归置组通过修复 **ceph-osd** 进程通常可以修复；卡在 **inactive** 状态的归置组通常是连接建立问题 (参见 [Placement Group Down - Peering Failure](#))；卡在 **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 [Unfound Objects](#))。

3.3.16.3 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（`lost`）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。

3.3.16.4 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` 抱怨有找不到（`unfound`）的对象：

```
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.

- 1 挂了；
- 2 独自处理一些写动作；
- 1 起来了；
- 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。

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.

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（`down`）。所有可能的状态有：

* 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。

3.3.16.5 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 **stale**、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）。

3.3.16.6 只有几个OSD 接收数据

Only a Few OSDs Receive Data

If you have many nodes in your cluster and only a few of them receive data, [check](#) the number of placement groups in your pool. Since placement groups get mapped to OSDs, a small number of placement groups will not distribute across your cluster. Try creating a pool with a placement group count that is a multiple of the number of OSDs. See [Placement Groups](#) for details. The default placement group count for pools isn't useful, but you can change it [here](#).

如果你的集群有很多节点，但只有其中几个接收数据，检查下存储池里的归置组数量。因为归置组是映射到多个OSD的，这样少量的归置组将不能分布于整个集群。试着创建个新存储池，其归置组数量是OSD数量的若干倍。详情见[Placement Groups](#)，存储池的默认归置组数量没多大用，你可以更改它。

3.3.16.7 不能写入数据

Can't Write Data

If your cluster is up, but some OSDs are down and you cannot write data, check to ensure that you have the minimum number of OSDs running for the placement group. If you don't have the minimum number of OSDs running, Ceph will not allow you to write data because there is no guarantee that Ceph can replicate your data. See `osd pool default min size` in the [Pool, PG and CRUSH Config Reference](#) for details.

如果你的集群已启动，但一些OSD没起来，导致不能写入数据，确认下运行的OSD数量满足归置组要求的最低OSD数。如果不能满足，ceph就不会允许你写入数据，因为ceph不能保证复制能如愿进行。详情参见[Pool, PG and CRUSH Config Reference](#)里的`osd pool default min size`。

3.3.17 日志记录和调试

Logging and Debugging

Typically, when you add debugging to your Ceph configuration, you do so at runtime. You can also add Ceph debug logging to your Ceph configuration file if you are encountering issues when starting your cluster. You may view Ceph log files under `/var/log/ceph` (the default location).

一般来说，你应该在运行时增加调试选项来调试问题；也可以把调试选项添加到ceph配置文件里来调试启动问题，然后查看`/var/log/ceph`（默认位置）下的日志文件。

Tip: When debug output slows down your system, the latency can hide race conditions.

提示：调试输出会拖慢系统，这种延时有可能掩盖竞争环境。

Logging is resource intensive. If you are encountering a problem in a specific area of your cluster, enable logging for that area of the cluster. For example, if your OSDs are running fine, but your metadata servers are not, you should start by enabling debug logging for the specific metadata server instance(s) giving you trouble. Enable logging for each subsystem as needed.

日志记录是资源密集任务。如果你碰到的问题在集群的某个特定区域，只启用那个区域对应的日志功能即可。例如，你的OSD运行良好、元数据服务器却不行，这时应该先打开那个可疑元数据服务器例程的调试日志；如果不另行打开各子系统的日志。

Important: Verbose logging can generate over 1GB of data per hour. If your OS disk reaches its capacity, the node will stop working.

重要：详尽的日志每小时可能超过1GB，如果你的系统盘满了，这个节点就会停止工作。

If you enable or increase the rate of Ceph logging, ensure that you have sufficient disk space on your OS disk. See [Accelerating Log Rotation](#) for details on rotating log files. When your system is running well, 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.

如果你要打开或增加ceph日志级别，确保系统盘空间足够。滚动日志文件的方法见[Accelerating Log Rotation](#)。集群稳定运行后，可以关闭不必要的调试选项以更好地运行。在运营中记录调试输出会拖慢系统、且浪费资源。

See [Subsystem, Log and Debug Settings](#) for details on available settings.

可用选项参见[Subsystem, Log and Debug Settings](#)。

3.3.17.1 运行时

Runtime

If you would like to see the configuration settings at runtime, you must log in to a host with a running daemon and execute the following:

如果你想查看一进程的运行时配置，必须先登录对应主机，然后执行命令：

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

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

To activate Ceph's debugging output (i.e., `dout()`) at runtime, use the `ceph tell` command to inject arguments into the runtime configuration:

要在运行时激活 ceph 的调试输出 (如 `dout()`)，用 `ceph tell` 命令把参数注入运行时配置：

```
ceph {daemon-type} tell {daemon 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` 或 `mds` 替代 `{daemon-type}`。你可以用星号 (*) 把配置应用到同类型的所有守护进程，或者指定具体守护进程的标识号 (即其名字或字母)。例如，要给名为 `osd.0` 的 `ceph-osd` 守护进程提高调试级别，用下列命令：

```
ceph osd tell 0 injectargs '--debug_osd 0/5'
```

The `ceph tell` command goes through the monitors. If you cannot bind to the monitor, you can still make the change by logging into the host of the daemon whose configuration you'd like to change using `ceph --admin-daemon`. For example:

`ceph tell` 命令会贯穿所有监视器。如果你不能绑定监视器，还可以登录你要改的那台主机用 `ceph --admin-daemon` 来更改。例如：

```
sudo ceph --admin-daemon /var/run/ceph/ceph-osd.0.asok config set debug_osd 0/5
```

See [Subsystem, Log and Debug Settings](#) for details on available settings.

可用选项参见 [Subsystem, Log and Debug Settings](#)。

3.3.17.2 启动时

Boot Time

To activate Ceph's debugging output (i.e., `dout()`) at boot time, you must add settings to your Ceph configuration file. 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:

要在启动时激活调试输出 (如 `dout()`)，你得把选项加入配置文件。各进程共有配置可写在配置文件的 `[global]` 下，某类进程的配置可写在守护进程段下 (如 `[mon]`、`[osd]`、`[mds]`)。例如：

```
[global]
    debug ms = 1/5

[mon]
    debug mon = 20
    debug paxos = 1/5
    debug auth = 2

[osd]
    debug osd = 1/5
    debug filestore = 1/5
    debug journal = 1
    debug monc = 5/20

[mds]
    debug mds = 1
    debug mds balancer = 1
    debug mds log = 1
    debug mds migrator = 1
```

See [Subsystem, Log and Debug Settings](#) for details.

详情见 [Subsystem, Log and Debug Settings](#)。

3.3.17.3 加快日志滚动

Accelerating Log Rotation

If your OS disk is relatively full, you can accelerate log rotation by modifying the Ceph log rotation file at `/etc/logrotate.d/ceph`. Add a size setting after the rotation frequency to accelerate log rotation (via cronjob) if your logs exceed the size setting. For example, the default setting looks like this:

如果你的系统盘比较满，可以修改`/etc/logrotate.d/ceph`内的日志滚动配置以加快滚动。在滚动频率后增加一个尺寸选项（达到此尺寸就滚动）来加快滚动（通过 cronjob）。例如默认配置大致如此：

```
rotate 7
weekly
compress
sharedscripts
```

Modify it by adding a `size` setting.

增加一个 `size` 选项：

```
rotate 7
weekly
size 500M
compress
sharedscripts
```

Then, start the crontab editor for your user space.

然后，打开 `crontab` 编辑器。

```
crontab -e
```

Finally, add an entry to check the `etc/logrotate.d/ceph` file.

最后，增加一条用以检查`/etc/logrotate.d/ceph`文件。

```
30 * * * * /usr/sbin/logrotate /etc/logrotate.d/ceph >/dev/null 2>&1
```

The preceding example checks the `etc/logrotate.d/ceph` file every 30 minutes.

本例中每 30 分钟检查一次`/etc/logrotate.d/ceph`文件。

3.3.17.4 Valgrind

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`.

你也许还得追踪内存和线程问题，可以在 Valgrind 中运行一个守护进程、一类进程、或整个集群。valgrind 是计算密集型程序，应该只用于开发或调试，否则会拖慢系统。其消息记录到 `stderr`。

3.3.17.5 子系统, 日志和调试选项

Subsystem, Log and Debug Settings

In most cases, you will enable debug logging output via subsystems.

大多数情况下你可以通过子系统打开调试。

3.3.17.5.1 ceph 子系统概览

Ceph Subsystems

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. Ceph's logging levels operate on a scale of 1 to 20, where 1 is terse and 20 is verbose.

各子系统都有日志级别用于分别控制其输出日志、和暂存日志，你可以分别为这些子系统设置不同的记录级别。ceph 的日志级别从 1 到 20，1 是简洁、20 是详尽。

A debug logging setting can take a single value for the log level and the memory level, which sets them both as the same value. For example, if you specify `debug ms = 5`, Ceph will treat it as a log level and a memory level of 5. You may also specify them separately. The first setting is the log level, and the second setting is the memory level. You must separate them with a forward slash (/). For example, if you want to set the `ms` subsystem's debug logging level to 1 and its memory level to 5, you would specify it as `debug ms = 1/5`. For example:

调试选项允许用单个数字同时设置日志级别和内存级别，会设置为一样。比如，如果你指定 `debug ms = 5`，ceph 会把日志级别和内存级别都设置为 5。也可以分别设置，第一个选项是日志级别、后一个是内存级别，二者必须用斜线 (/) 分隔。假如你想把 `ms` 子系统的调试日志级别设为 1、内存级别设为 5，可以写为 `debug ms = 1/5`，如下：

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

The following table provides a list of Ceph subsystems and their default log and memory levels. Once you complete your logging efforts, restore the subsystems to their default level or to a level suitable for normal operations.

下表列出了 ceph 子系统及其默认日志和内存级别。一旦你完成调试，应该恢复默认值、或一个适合平常运营的级别。

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
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

3.3.17.5.2 日志记录选项

Logging Settings

Logging and debugging settings are not required in a Ceph configuration file, but you may override default settings as needed. Ceph supports the following settings:

日志和调试选项不是必需配置，但你可以按需覆盖默认值。ceph 支持如下配置：

log file

Description: The location of the logging file for your cluster.
集群日志文件的位置。

Type: String
Required: No
Default: /var/log/ceph/\$cluster-\$name.log

log max new

Description: The maximum number of new log files.
新日志文件的最大数量。

Type: Integer
Required: No
Default: 1000

log max recent

Description: The maximum number of recent events to include in a log file.
一个日志文件包含的最新事件的最大数量。

Type: Integer
Required: No
Default: 1000000

log to stderr

Description: Determines if logging messages should appear in stderr.
设置日志消息是否输出到标准错误。

Type: Boolean
Required: No
Default: true

err to stderr

Description: Determines if error messages should appear in stderr.
设置错误消息是否输出到标准错误。

Type: Boolean
Required: No
Default: true

log to syslog

Description: Determines if logging messages should appear in syslog.
设置日志消息是否输出到 syslog。

Type: Boolean
Required: No
Default: false

err to syslog

Description: Determines if error messages should appear in syslog.
设置错误消息是否输出到 syslog。

Type: Boolean
Required: No
Default: false

log flush on exit

Description: Determines if Ceph should flush the log files after exit.
设置 ceph 退出后是否回写日志文件。

Type: Boolean
Required: No
Default: true

clog to monitors

Description: Determines if clog messages should be sent to monitors.

设置是否把 clog 消息发送给监视器。

Type: Boolean

Required: No

Default: true

clog to syslog

Description: Determines if clog messages should be sent to syslog.

设置是否把 clog 输出到 syslog。

Type: Boolean

Required: No

Default: false

mon cluster log to syslog

Description: Determines if the cluster log should be output to the syslog.

设置集群日志是否输出到 syslog。

Type: Boolean

Required: No

Default: false

mon cluster log file

Description: The location of the cluster's log file.

集群日志位置。

Type: String

Required: No

Default: /var/log/ceph/\$cluster.log

3.3.17.5.3 OSD

osd debug drop ping probability

Description: ?

Type: Double

Required: No

Default: 0

osd debug drop ping duration

Description:

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: Preserves trimmed logs after trimming.

裁减后保留剩余日志。

Type: Boolean

Required: No

Default: false

```
osd tmapput sets uses tmap
```

Description: Uses tmap. For debug only.

使用 tmap, 仅用于调试。

Type: Boolean

Required: No

Default: false

```
osd min pg log entries
```

Description: The minimum number of log entries for placement groups.

归置组日志最小条数。

Type: 32-bit Unsigned Integer

Required: No

Default: 1000

```
osd op log threshold
```

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

一次发送多少操作日志消息。

Type: Integer

Required: No

Default: 5

3.3.17.5.4 Filestore

```
filestore debug omap check
```

Description: Debugging check on synchronization. This is an expensive operation.

检查 omap, 这是昂贵的操作。

Type: Boolean

Required: No

Default: 0

3.3.17.5.5 MDS

```
mds debug scatterstat
```

Description: Ceph will assert that various recursive stat invariants are true (for developers only).

ceph 将把各种回归状态常量设置为真（谨为开发者）。

Type: Boolean

Required: No

Default: false

```
mds debug frag
```

Description: Ceph will verify directory fragmentation invariants when convenient (developers only).

ceph 将在方便时校验目录碎片（谨为开发者）。

Type: Boolean

Required: No

Default: false

```
mds debug auth pins
```

Description: The debug auth pin invariants (for developers only).

debug auth pin 恒量（谨为开发者）。

Type: Boolean

Required: No

Default: false

```
mds debug subtrees
```

Description: The debug subtree invariants (for developers only).

debug subtree 恒量（谨为开发者）。

Type: Boolean

Required: No

Default: false

3.3.17.5.6 RADOS 网关

RADOS Gateway

`rgw log nonexistent bucket`

Description: Should we log a non-existent buckets?
记录不存在的桶?

Type: Boolean
Required: No
Default: `false`

`rgw log object name`

Description: Should an object's name be logged. // man date to see codes (a subset are supported)
是否记录对象名称。

Type: String
Required: No
Default: `%Y-%m-%d-%H-%i-%n`

`rgw log object name utc`

Description: Object log name contains UTC?
对象日志名称包含 UTC?

Type: Boolean
Required: No
Default: `false`

`rgw enable ops log`

Description: Enables logging of every RGW operation.
允许记录 RGW 的每一个操作。

Type: Boolean
Required: No
Default: `true`

`rgw enable usage log`

Description: Enable logging of RGW's bandwidth usage.
允许记录 RGW 的带宽使用。

Type: Boolean
Required: No
Default: `true`

`rgw usage log flush threshold`

Description: Threshold to flush pending log data.
回写未决的日志数据阀值。

Type: Integer
Required: No
Default: `1024`

`rgw usage log tick interval`

Description: Flush pending log data every s seconds.
未决日志回写间隔。

Type: Integer
Required: No
Default: `30`

`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.
日志对象名字里包含 UTC 时间戳。

Type: Boolean
Required: No
Default: `false`

3.3.18 CPU 剖析

CPU Profiling

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.

如果你从源码编译时启用了 oprofile，那就可以剖析 ceph 的 CPU 使用情况，详情见 [Installing Oprofile](#)。

3.3.18.1 初始化 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` 时要初始化，找到对应于当前运行内核的 `vmlinux` 映像：

```
ls /boot
sudo opcontrol --init
sudo opcontrol --setup --vmlinux={path-to-image} --separate=library --callgraph=6
```

3.3.18.2 启动 oprofile

STARTING OPROFILE

To start `oprofile` execute the following command:

执行下面的命令启动 oprofile：

```
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：

3.3.18.3 停止 oprofile

STOPPING OPROFILE

To stop `oprofile` execute the following command:

执行下面的命令停止 oprofile：

```
opcontrol --stop
```

3.3.18.4 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。

3.3.18.5 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。

3.3.19 内存剖析

Memory Profiling

Ceph OSD and metadata server daemons can generate heap profiles using `tcmalloc`. To generate heap profiles, ensure you have `google-perf-tools` installed:

ceph 的 OSD 和元数据服务器可利用 `tcmalloc` 生成堆栈剖析，此功能依赖 `google-perf-tools`:

```
sudo apt-get google-perf-tools
```

The profiler dumps output to your `log file` directory (i.e., `/var/log/ceph`). See [Logging and Debugging](#) for details. To view the profiler logs with Google's performance tools, execute the following:

剖析器会把输出保存到 `log file` 目录（如`/var/log/ceph`），详情见 [Logging and Debugging](#)。剖析器日志可用 Google 性能工具来查看，执行如下命令：

```
google-pprof -gv {log-path/filename}
```

Refer to [Google Heap Profiler](#) for additional details.

额外信息见 [Google Heap Profiler](#)。

Once you have the heap profiler installed, start your cluster and begin using the heap profiler. You may enable or disable the heap profiler at runtime, or ensure that it runs continuously. For the following commandline usage, replace `{daemon-type}` with `osd` or `mds`, and replace `daemon-id` with the OSD number or metadata server letter.

安装堆栈剖析器后，启动集群就可以开始使用了。你可以在运行时启用、或禁用堆栈剖析器，或确保它在持续运行。在随后的几个命令行用法中，用 `osd` 或 `mds` 替换掉 `{daemon-type}`、用 OSD 号或元数据服务器字母替换掉 `daemon-id`。

3.3.19.1 启动剖析器

Starting the Profiler

To start the heap profiler, execute the following:

要启动堆栈剖析器，执行如下命令：

```
ceph {daemon-type} tell {daemon-id} heap start_profiler
```

For example:

例如：

```
ceph osd tell 1 heap start_profiler
```

3.3.19.2 打印统计

Printing Stats

To print out statistics, execute the following:

用下列命令打印统计状态：

```
ceph {daemon-type} tell {daemon-id} heap stats
```

For example:

例如：

```
ceph osd tell 0 heap stats
```

Note: Printing stats does not require the profiler to be running and does not dump the heap allocation information to a file.

注意：打印状态不要求剖析器在运行，也不会把堆栈分配信息转储到文件。

3.3.19.3 转储堆栈信息

Dumping Heap Information

To dump heap information, execute the following:

用下列命令转储堆栈信息：

```
ceph {daemon-type} tell {daemon-id} heap dump
```

For example:

例如：

```
ceph mds tell a heap dump
```

Note: Dumping heap information only works when the profiler is running.

注意：只能在剖析器运行的时候转储堆栈信息。

3.3.19.4 释放内存

Releasing Memory

To release memory that `tcmalloc` has allocated but which is not being used by the Ceph daemon itself, execute the following:

要释放由 `tcmalloc` 分配、但不是被 `ceph` 守护进程使用的内存，用下列命令：

```
ceph {daemon-type} tell {daemon-id} heap release
```

For example:

例如：

```
ceph osd tell 2 heap release
```

3.3.19.5 停止剖析器

Stopping the Profiler

To stop the heap profiler, execute the following:

要停止堆栈剖析器，执行下列命令：

```
ceph {daemon-type} tell {daemon-id} heap stop_profiler
```

For example:

例如：

```
ceph {daemon-type} tell {daemon-id} heap stop_profiler
```

3.4 手册页<尚未整理>

Object Store Manpages

- [ceph – ceph file system control utility](#)
- [ceph-rest-api – ceph RESTlike administration server](#)
- [ceph-authtool – ceph keyring manipulation tool](#)
- [ceph-clsinfo – show class object information](#)
- [ceph-conf – ceph conf file tool](#)
- [ceph-debugpack – ceph debug packer utility](#)
- [ceph-dencoder – ceph encoder/decoder utility](#)
- [ceph-mon – ceph monitor daemon](#)
- [ceph-osd – ceph object storage daemon](#)
- [ceph-run – restart daemon on core dump](#)
- [ceph-syn – ceph synthetic workload generator](#)
- [crushtool – CRUSH map manipulation tool](#)
- [librados-config – display information about librados](#)
- [monmaptool – ceph monitor cluster map manipulation tool](#)
- [osdmaptool – ceph osd cluster map manipulation tool](#)
- [rados – rados object storage utility](#)

3.5 故障排除

Troubleshooting

Ceph is still on the leading edge, so you may encounter situations that require you to examine your configuration, modify your logging output, troubleshoot monitors and OSDs, profile memory and CPU usage, and reach out to the Ceph community for help.

ceph 仍在积极开发中，所以你可能碰到一些问题，需要检查 ceph 配置文件、修改日志、排除监视器和 OSD 故障、剖析内存和 CPU 使用情况、并到 ceph 社区寻求帮助。

- [The Ceph Community](#)
- [Logging and Debugging](#)
- [Recovering from Monitor Failures](#)
- [Troubleshooting OSDs](#)
- [Troubleshooting PGs](#)
- [Memory Profiling](#)
- [CPU Profiling](#)

3.5.1 Ceph 社区

见 [Ceph 社区](#)。

3.5.2 日志记录和调试

见 [日志记录和调试](#)。

3.5.3 监视器故障恢复

见 [监视器故障恢复](#)。

3.5.4 OSD 故障排除

见 [OSD 故障排除](#)。

3.5.5 PG 故障排除

3.5.6 内存剖析

见[内存剖析](#)。

3.5.7 CPU 剖析

见[CPU 剖析](#)。

3.6 API 接口

APIs

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。

3.6.1 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。

3.6.1.1 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);
```

3.6.1.2 异步IO

Asynchronous 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。

3.6.1.3 API 调用

API calls

3.6.1.3.1 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
```

3.6.1.3.2 struct rados_cluster_stat_t

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
```

3.6.1.3.3 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  
LIBRADOS_SNAP_HEAD  
LIBRADOS_SNAP_DIR
```

3.6.1.3.4 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

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\(\)](#))
存储池——上下文创建时就设置了（见 [rados_ioctx_create\(\)](#)）
- snapshot context for writes (see [rados_ioctx_selfmanaged_snap_set_write_ctx\(\)](#))
用于写的快照上下文（见 [rados_ioctx_selfmanaged_snap_set_write_ctx\(\)](#)）
- snapshot id to read from (see [rados_ioctx_snap_set_read\(\)](#))
要读的快照 ID（见 [rados_ioctx_snap_set_read\(\)](#)）
- object locator for all single-object operations (see [rados_ioctx_locator_set_key\(\)](#))
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(), and rados_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\(\)](#)、[and rados_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 attributes on an object.

Used with rados_getxattrs(), rados_getxattrs_next(), and rados_getxattrs_end().

列出一个对象扩展属性的迭代器。

和 [rados_getxattrs\(\)](#)、[rados_getxattrs_next\(\)](#)、[and rados_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 asynchronous operations, take two parameters:

异步操作回调，需 2 个参数：

- cb the completion that has finished

- cb 刚完成的操作；
- arg application defined data made available to the callback function
- 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
opcode 未定义；
- ver version of the watched object
ver 被监视对象的版本；
- arg application-specific data
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。

rados_log_callback_t

3.6.1.3.5 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 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, __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 pool
- 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

```
void rados_ioctx_set_namespace(rados_ioctx_t io, const char *nspace)
```

Set the namespace for objects within an io context.

在一个 IO 上下文内设置命名空间。

The namespace specification further refines a pool into different domains. The mapping of objects to pgs is also based on this value.

命名空间把一个存储池重定义为不同的域。对象到归置组的映射也是基于这个值的。

Parameters:

- **io** – the io context to change

io - 要更改的 IO 上下文。

- **nspace** – the name to use as the namespace, or NULL use the default namespace

nspace - 要用作命名空间的名字，NULL 是使用默认命名空间。

```
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 snapshot 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 snapshots 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 precede 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 – iocxt
- 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_iocxt_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 – iocxt
- o – object name
- buf – buffer
- buflen – buffer length in bytes

Returns:

- 0 on success, negative error code on failure

```
int rados_tmap_get(rados_iocxt_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 – iocxt
- 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_iocxt_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 journaling 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 replicas
- 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。

这意味着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.

这意味着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 asynchronous 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，失败时为负数错误代码。

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)
```

Asynchronously 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_aio_flush_async(rados_ioctx_t io, rados_completion_t completion)**

Schedule a callback for when all currently pending aio writes are safe.

This is a non-blocking version of [rados_aio_flush\(\)](#).

Parameters:

- **io** – the context to flush
- **completion** – what to do when the writes are safe

Returns:

- 0 on success, negative error code on failure

int **rados_aio_stat(rados_ioctx_t io, const char *o, rados_completion_t completion, uint64_t *psize, time_t *pmtime)**

Asynchronously get object stats (size/mtime)

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_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)
```

Synchronously 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

```
int rados_lock_exclusive(rados\_ioctx\_t io, const char *o, const char *name, const char *cookie, const char *desc, struct timeval *duration, uint8_t flags)
```

Take an exclusive lock on an object.

在一个对象上设置独占锁。

Parameters:

- **io** – the context to operate in
- **oid** – the name of the object
- **name** – the name of the lock
- **cookie** – user-defined identifier for this instance of the lock
- **desc** – user-defined lock description
- **duration** – the duration of the lock. Set to NULL for infinite duration.
- **flags** – lock flags

Returns: 0 on success, negative error code on failure

-EBUSY if the lock is already held by another (client, cookie) pair

-EEXIST if the lock is already held by the same (client, cookie) pair

```
int rados_lock_shared(rados\_ioctx\_t io, const char *o, const char *name, const char *cookie, const char *tag, const char *desc, struct timeval *duration, uint8_t flags)
```

Take a shared lock on an object.

在一对象上设置共享锁。

Parameters:

- **io** – the context to operate in
- **o** – the name of the object
- **name** – the name of the lock
- **cookie** – user-defined identifier for this instance of the lock
- **tag** – The tag of the lock
- **desc** – user-defined lock description
- **duration** – the duration of the lock. Set to NULL for infinite duration.
- **flags** – lock flags

Returns: 0 on success, negative error code on failure

-EBUSY if the lock is already held by another (client, cookie) pair

-EEXIST if the lock is already held by the same (client, cookie) pair

```
int rados_unlock(rados\_ioctx\_t io, const char *o, const char *name, const char *cookie)
```

Release a shared or exclusive lock on an object.

释放共享或独占锁。

Parameters:

- **io** – the context to operate in

- **o** – the name of the object
- **name** – the name of the lock
- **cookie** – user-defined identifier for the instance of the lock

Returns: 0 on success, negative error code on failure

-ENOENT if the lock is not held by the specified (client, cookie) pair

`ssize_t rados_list_lockers(rados_ioctx_t io, const char *o, const char *name, int *exclusive, char *tag, size_t *tag_len, char *clients, size_t *clients_len, char *cookies, size_t *cookies_len, char *addrs, size_t *addrs_len)`

List clients that have locked the named object lock and information about the lock.

列出锁住指定对象的客户端、以及锁信息。

The number of bytes required in each buffer is put in the corresponding size out parameter. If any of the provided buffers are too short, -ERANGE is returned after these sizes are filled in.

各缓冲需要的字节数放置在参数之外的 size 中。如果某一个缓冲太小，这些区域填充满后会返回-ERANGE。

Parameters:

- **io** – the context to operate in
- **o** – the name of the object
- **name** – the name of the lock
- **exclusive** – where to store whether the lock is exclusive (1) or shared (0)
- **tag** – where to store the tag associated with the object lock
- **tag_len** – number of bytes in tag buffer
- **clients** – buffer in which locker clients are stored, separated by '\0'
- **clients_len** – number of bytes in the clients buffer
- **cookies** – buffer in which locker cookies are stored, separated by '\0'
- **cookies_len** – number of bytes in the cookies buffer
- **addrs** – buffer in which locker addresses are stored, separated by '\0'
- **addrs_len** – number of bytes in the clients buffer

Returns: number of lockers on success, negative error code on failure

-ERANGE if any of the buffers are too short

`int rados_break_lock(rados_ioctx_t io, const char *o, const char *name, const char *client, const char *cookie)`

Releases a shared or exclusive lock on an object, which was taken by the specified client.

释放一对象上的共享或独占锁，由特定客户端调用。

Parameters:

- **io** – the context to operate in
- **o** – the name of the object
- **name** – the name of the lock
- **client** – the client currently holding the lock
- **cookie** – user-defined identifier for the instance of the lock

Returns: 0 on success, negative error code on failure

-ENOENT if the lock is not held by the specified (client, cookie) pair

-EINVAL if the client cannot be parsed

`int rados_mon_command(rados_t cluster, const char **cmd, size_t cmdlen, const char *inbuf, size_t inbuflen, char **outbuf, size_t *outbuflen, char **outs, size_t *outslen)`

Send monitor command.

向监视器发送命令。

Note: Takes command string in carefully-formatted JSON; must match defined commands, types, etc.

注意：从格式化完好的 JSON 中读取命令字符串，必须匹配定义的命令、类型等等。

The result buffers are allocated on the heap; the caller is expected to release that memory with [rados_buffer_free\(\)](#). The buffer and length pointers can all be NULL, in which case they are not filled in.

结果缓冲区被分配为堆栈，其句柄应该用 [rados_buffer_free\(\)](#) 释放内存。缓冲和长度指针可都为空，这时它们都未填充。

Parameters:

- **cluster** – cluster handle
- **cmd** – an array of char *'s representing the command
- **cmdlen** – count of valid entries in cmd
- **inbuf** – any bulk input data (crush map, etc.)
- **outbuf** – double pointer to output buffer
- **outbuflen** – pointer to output buffer length
- **outs** – double pointer to status string
- **outslen** – pointer to status string length

Returns: 0 on success, negative error code on failure

```
int rados_mon_command_target(rados\_t cluster, const char *name, const char **cmd, size_t cmdlen, const char *inbuf, size_t inbuflen, char **outbuf, size_t *outbuflen, char **outs, size_t *outslen)
```

Send monitor command to a specific monitor.

把监视器命令发给指定监视器。

Note: Takes command string in carefully-formatted JSON; must match defined commands, types, etc.

注意：命令字符串应该用 JSON 格式化好，已定义命令、类型必须匹配。

The result buffers are allocated on the heap; the caller is expected to release that memory with [rados_buffer_free\(\)](#). The buffer and length pointers can all be NULL, in which case they are not filled in.

结果缓冲区被分配为堆栈，其句柄应该用 [rados_buffer_free\(\)](#) 释放内存。缓冲和长度指针可都为空，这时它们都未填充。

Parameters:

- **cluster** – cluster handle
- **name** – target monitor's name
- **cmd** – an array of char *'s representing the command
- **cmdlen** – count of valid entries in cmd
- **inbuf** – any bulk input data (crush map, etc.)
- **outbuf** – double pointer to output buffer
- **outbuflen** – pointer to output buffer length
- **outs** – double pointer to status string
- **outslen** – pointer to status string length

Returns: 0 on success, negative error code on failure

```
void rados_buffer_free(char *buf)
```

free a rados-allocated buffer

释放一 rados 分配的缓冲区。

Release memory allocated by librados calls like [rados_mon_command\(\)](#).

释放由 librados 调用（像 [rados_mon_command\(\)](#)）分配的内存，

Parameters:

- **buf** – buffer pointer

```
int rados_osd_command(rados\_t cluster, int osdid, const char **cmd, size_t cmdlen, const char *inbuf, size_t
```

```
inbuflen, char **outbuf, size_t *outbuflen, char **outs, size_t *outslen)
```

```
int rados_pg_command(rados\_t cluster, const char *pgstr, const char **cmd, size_t cmdlen, const char *inbuf,  
size_t inbuflen, char **outbuf, size_t *outbuflen, char **outs, size_t *outslen)
```

```
int rados_monitor_log(rados\_t cluster, const char *level, rados\_log\_callback\_t cb, void *arg)
```

3.6.2 libradospp (C++)

Todo write me!

待完成。

4 CEPH 文件系统

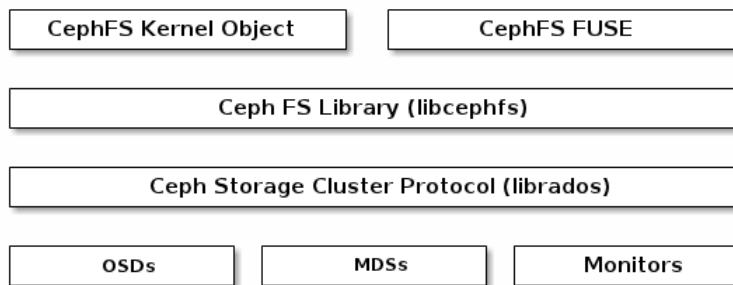
CEPH FS

The [Ceph Filesystem](#) (Ceph FS) is a POSIX-compliant filesystem that uses a Ceph Storage Cluster to store its data. The Ceph filesystem uses the same Ceph Storage Cluster system as Ceph Block Devices, Ceph Object Storage with its S3 and Swift APIs, or native bindings (librados).

Ceph 文件系统是个 POSIX 兼容的文件系统，它使用 ceph 存储集群来存储数据。Ceph FS 使用的 ceph 存储集群系统和 ceph 块设备及 ceph 对象存储相同，就像 ceph 对象存储也有 S3 和 Swift API、或本地库 (librados) 一样。

Important: Ceph FS is currently not recommended for production data.

重要：现在还不建议 Ceph 文件系统用于生产环境数据。



Using the Ceph Filesystem requires at least one [Ceph Metadata Server](#) in your Ceph Storage Cluster.
ceph 文件系统要求 ceph 存储集群内至少有一个 ceph 元数据服务器。

Step 1: Metadata Server

To run the Ceph Filesystem, you must have a running Ceph Storage Cluster with at least one [Ceph Metadata Server](#) running.

- [Add/Remove MDS](#)
- [MDS Configuration](#)
- [Journaler Configuration](#)
- [Manpage ceph-mds](#)

Step 2: Mount Ceph FS

Once you have a healthy Ceph Storage Cluster with at least one Ceph Metadata Server, you may mount your Ceph Filesystem. Ensure that your client has network connectivity and the proper authentication keyring.

- [Mount Ceph FS](#)
- [Mount Ceph FS as FUSE](#)
- [Mount Ceph FS in fstab](#)
- [Manpage cephfs](#)
- [Manpage ceph-fuse](#)
- [Manpage mount.ceph](#)

Additional Details

- [Using Ceph with Hadoop](#)
- [libcephfs](#)
- [Troubleshooting](#)

4.1 增加/拆除元数据服务器

见[增加/拆除元数据服务器](#)。

4.2 MDS 配置参考

mds config reference

`mon force standby active`

Description: If `true` monitors force standby-replay to be active. Set under `[mon]` or `[global]`.
如设为 `true`, 监视器会把 standby-replay 状态强设为 active。设置于 `[mon]` 或 `[global]` 下。

Type: Boolean

Default: `true`

`max mds`

Description: The number of active MDS daemons during cluster creation. Set under `[mon]` or `[global]`.
集群创建时的 MDS 守护进程数量, 设置于 `[mon]` 或 `[global]` 下。

Type: 32-bit Integer

Default: `1`

`mds max file size`

Description: The maximum allowed file size to set when creating a new file system.
创建一新文件系统时允许的最大文件尺寸。

Type: 64-bit Integer Unsigned

Default: `1ULL << 40`

`mds cache size`

Description: The number of inodes to cache.
缓存的索引节点数。

Type: 32-bit Integer

Default: `100000`

`mds cache mid`

Description: The insertion point for new items in the cache LRU (from the top).
把新条目插入缓存 LRU 时的插入点 (从顶端)。

Type: Float

Default: `0.7`

`mds dir commit ratio`

Description: The fraction of directory that is dirty before Ceph commits using a full update (instead of partial update).
目录的脏到什么程度时做完全更新 (而不是部分更新)。

Type: Float

Default: `0.5`

`mds dir max commit size`

Description: The maximum size of a directory update before Ceph breaks it into smaller transactions) (MB).
一个目录更新超过多大时要拆分为较小的事物 (MB)。

Type: 32-bit Integer

Default: `90`

`mds decay halflife`

Description: The half-life of MDS cache temperature.
MDS 缓存热度的半衰期。

Type: Float

Default: `5`

`mds beacon interval`

Description: The frequency (in seconds) of beacon messages sent to the monitor.
标识消息发送给监视器的频率, 秒。

Type: Float

Default: 4

mds beacon grace

Description: The interval without beacons before Ceph declares an MDS laggy (and possibly replace it).
多久没收到标识消息就认为 MDS 落后了（并可能替换它）。

Type: Float

Default: 15

mds blacklist interval

Description: The blacklist duration for failed MDSs in the OSD map.

OSD 运行图里的 MDS 失败后，把它留在黑名单里的时间。

Type: Float

Default: 24.0*60.0

mds session timeout

Description: The interval (in seconds) of client inactivity before Ceph times out capabilities and leases.

客户端多久不活跃时，ceph 就清除其能力和租期。

Type: Float

Default: 60

mds session autoclose

Description: The interval (in seconds) before Ceph closes a laggy client's session.

自动关闭空闲会话前的等待时间，秒。

Type: Float

Default: 300

mds reconnect timeout

Description: The interval (in seconds) to wait for clients to reconnect during MDS restart.

mds 重启期间客户端等待时间，秒。

Type: Float

Default: 45

mds tick interval

Description: How frequently the MDS performs internal periodic tasks.

MDS 执行内部周期性任务的间隔。

Type: Float

Default: 5

mds dirstat min interval

Description: The minimum interval (in seconds) to try to avoid propagating recursive stats up the tree.

在目录树中递归地查找目录信息的最小间隔，秒。

Type: Float

Default: 1

mds scatter nudge interval

Description: How quickly dirstat changes propagate up.

dirstat 变更传播多快。

Type: Float

Default: 5

mds client prealloc inos

Description: The number of inode numbers to preallocate per client session.

为每个客户端会话预分配的索引节点数。

Type: 32-bit Integer

Default: 1000

mds early reply

Description: Determines whether the MDS should allow clients to see request results before they commit to the journal.

请求结果成功提交到日志前是否应该先展示给客户端。

Type: Boolean

Default: true

mds use tmap

Description: Use trivialmap for directory updates.

把 trivialmap 用于目录更新。

Type: Boolean

Default: true

mds default dir hash

Description: The function to use for hashing files across directory fragments.

用于哈希文件在目录中分布情况的函数。

Type: 32-bit Integer

Default: 2 (i.e., rjenkins)

mds log

Description: Set to true if the MDS should journal metadata updates (disabled for benchmarking only).

默认为 true， MDS 是否要记录元数据更新（只可在性能评测时禁用）。

Type: Boolean

Default: true

mds log skip corrupt events

Description: Determines whether the MDS should try to skip corrupt journal events during journal replay.

在日志重放时， MDS 是否应跳过损坏的日志事件。

Type: Boolean

Default: false

mds log max events

Description: The maximum events in the journal before we initiate trimming. Set to -1 to disable limits.

日志里的事件达到多少时开始裁剪，设为 -1 取消限制。

Type: 32-bit Integer

Default: -1

mds log max segments

Description: The maximum number of segments (objects) in the journal before we initiate trimming. Set to -1 to disable limits.

日志里的片段（对象）达到多少时开始裁剪，设为 -1 取消限制。

Type: 32-bit Integer

Default: 30

mds log max expiring

Description: The maximum number of segments to expire in parallel.

可同时过期的片段数。

Type: 32-bit Integer

Default: 20

mds log eopen size

Description: The maximum number of inodes in an EOpen event.

在一个 EOpen 事件中最大索引节点数。

Type: 32-bit Integer

Default: 100

mds bal sample interval

Description: Determines how frequently to sample directory temperature (for fragmentation decisions).

对目录热度取样的频率（碎片粒度）。

Type: Float

Default: 3

mds bal replicate threshold

Description: The maximum temperature before Ceph attempts to replicate metadata to other nodes.
达到多大热度时 ceph 就把元数据复制到其它节点。

Type: Float

Default: 8000

mds bal unreplicate threshold

Description: The minimum temperature before Ceph stops replicating metadata to other nodes.
热度低到多少时 ceph 就不再把元数据复制到其它节点。

Type: Float

Default: 0

mds bal frag

Description: Determines whether the MDS will fragment directories.
MDS 是否应该把目录破为碎片。

Type: Boolean

Default: false

mds bal split size

Description: The maximum directory size before the MDS will split a directory fragment into smaller bits.
目录尺寸大到多少时 MDS 就把片段拆分成更小的片段。

Type: 32-bit Integer

Default: 10000

mds bal split rd

Description: The maximum directory read temperature before Ceph splits a directory fragment.
目录的最大读取热度达到多大时 ceph 将拆分此片段。

Type: Float

Default: 25000

mds bal split wr

Description: The maximum directory write temperature before Ceph splits a directory fragment.
目录的最大写热度达到多大时 ceph 将拆分此片段。

Type: Float

Default: 10000

mds bal split bits

Description: The number of bits by which to split a directory fragment.
把一个目录片段再分割成多大。

Type: 32-bit Integer

Default: 3

mds bal merge size

Description: The minimum directory size before Ceph tries to merge adjacent directory fragments.
目录尺寸小到多少时 ceph 就把它合并到邻近目录片段。

Type: 32-bit Integer

Default: 50

mds bal merge rd

Description: The minimum read temperature before Ceph merges adjacent directory fragments.
读热度低于此值时 ceph 将合并邻近目录片段。

Type: Float

Default: 1000

mds bal merge wr

Description: The minimum write temperature before Ceph merges adjacent directory fragments.
写热度低于此值时 ceph 就合并邻近目录片段。

Type: Float
Default: 1000

mds bal interval

Description: The frequency (in seconds) of workload exchanges between MDSs.
MDS 服务器负荷交换频率，秒。

Type: 32-bit Integer
Default: 10

mds bal fragment interval

Description: The frequency (in seconds) of adjusting directory fragmentation.
邻近目录分片频率，秒。

Type: 32-bit Integer
Default: 5

mds bal idle threshold

Description: The minimum temperature before Ceph migrates a subtree back to its parent.
热度低于此值时 ceph 把子树迁移回父节点。

Type: Float
Default: 0

mds bal max

Description: The number of iterations to run balancer before Ceph stops. (used for testing purposes only)
均衡器迭代到此数量时 ceph 就停止（仅适用于测试）。

Type: 32-bit Integer
Default: -1

mds bal max until

Description: The number of seconds to run balancer before Ceph stops. (used for testing purposes only)
均衡器运行多久就停止（仅适用于测试）。

Type: 32-bit Integer
Default: -1

mds bal mode

Description: The method for calculating MDS load.

计算 MDS 负载的方法。

- 1 = Hybrid.
- 2 = Request rate and latency.
- 3 = CPU load.

Type: 32-bit Integer

Default: 0

mds bal min rebalance

Description: The minimum subtree temperature before Ceph migrates.
子树热度最小多少时开始迁移。

Type: Float
Default: 0.1

mds bal min start

Description: The minimum subtree temperature before Ceph searches a subtree.
子树热度最小多少时 ceph 才去搜索。

Type: Float
Default: 0.2

mds bal need min

Description: The minimum fraction of target subtree size to accept.

接受的最小目标子树片段。

Type: Float

Default: 0.8

mds bal need max

Description: The maximum fraction of target subtree size to accept.
目标子树片段的最大尺寸。

Type: Float

Default: 1.2

mds bal midchunk

Description: Ceph will migrate any subtree that is larger than this fraction of the target subtree size.
尺寸大于目标子树片段的子树，ceph 将迁移它。

Type: Float

Default: 0.3

mds bal minchunk

Description: Ceph will ignore any subtree that is smaller than this fraction of the target subtree size.
尺寸小于目标子树片段的子树，ceph 将忽略它。

Type: Float

Default: 0.001

mds bal target removal min

Description: The minimum number of balancer iterations before Ceph removes an old MDS target from the MDS map.
ceph 从 MDS 运行图中剔除旧数据前，均衡器至少递归多少次。

Type: 32-bit Integer

Default: 5

mds bal target removal max

Description: The maximum number of balancer iteration before Ceph removes an old MDS target from the MDS map.
ceph 从 MDS 运行图中剔除旧数据前，均衡器最多递归多少次。

Type: 32-bit Integer

Default: 10

mds replay interval

Description: The journal poll interval when in standby-replay mode. (“hot standby”)
MDS 处于 standby-replay (热备) 模式下时的日志滚动间隔。

Type: Float

Default: 1

mds shutdown check

Description: The interval for polling the cache during MDS shutdown.
MDS 关闭期间缓存更新间隔。

Type: 32-bit Integer

Default: 0

mds thrash exports

Description: Ceph will randomly export subtrees between nodes (testing only).
ceph 会把子树随机地在节点间迁移。

Type: 32-bit Integer

Default: 0

mds thrash fragments

Description: Ceph will randomly fragment or merge directories.
ceph 会随机地分片或合并目录。

Type: 32-bit Integer

Default: 0

`mds dump cache on map`

Description: Ceph will dump the MDS cache contents to a file on each MDSMap.
ceph 会把各 MDSMap 的 MDS 缓存内容转储到一文件。

Type: Boolean

Default: `false`

`mds dump cache after rejoin`

Description: Ceph will dump MDS cache contents to a file after rejoining the cache (during recovery).
ceph 重新加入缓存（恢复期间）后会把 MDS 缓存内容转储到一文件。

Type: Boolean

Default: `false`

`mds verify scatter`

Description: Ceph will assert that various scatter/gather invariants are `true` (developers only).
ceph 将把各种传播/聚集常量声明为 `true`（仅适合开发者）。

Type: Boolean

Default: `false`

`mds debug scatterstat`

Description: Ceph will assert that various recursive stat invariants are `true` (for developers only).
ceph 将把各种递归统计常量声明为 `true`（仅适合开发者）。

Type: Boolean

Default: `false`

`mds debug frag`

Description: Ceph will verify directory fragmentation invariants when convenient (developers only).
ceph 将在方便时校验目录分段（仅适合开发者）。

Type: Boolean

Default: `false`

`mds debug auth pins`

Description: The debug auth pin invariants (for developers only).
常量 debug auth pin（仅适合开发者）。

Type: Boolean

Default: `false`

`mds debug subtrees`

Description: The debug subtree invariants (for developers only).
常量 debug subtree（仅适合开发者）。

Type: Boolean

Default: `false`

`mds kill mdstable at`

Description: Ceph will inject MDS failure in MDSTable code (for developers only).
ceph 将向 MDSTable 代码注入 MDS 失败事件（仅适合开发者）。

Type: 32-bit Integer

Default: `0`

`mds kill export at`

Description: Ceph will inject MDS failure in the subtree export code (for developers only).
ceph 将向子树出口代码注入 MDS 失败事件（仅适合开发者）。

Type: 32-bit Integer

Default: `0`

`mds kill import at`

Description: Ceph will inject MDS failure in the subtree import code (for developers only).
ceph 将向子树入口代码注入 MDS 失败事件（仅适合开发者）。

Type: 32-bit Integer

Default: `0`

`mds kill link at`

Description: Ceph will inject MDS failure in hard link code (for developers only).

ceph 将向硬链接代码注入 MDS 失败事件（仅适合开发者）。

Type: 32-bit Integer

Default: 0

`mds kill rename at`

Description: Ceph will inject MDS failure in the rename code (for developers only).

ceph 将向重命名代码注入 MDS 失败事件（仅适合开发者）。

Type: 32-bit Integer

Default: 0

`mds wipe sessions`

Description: Ceph will delete all client sessions on startup (for testing only).

ceph 将在启动时删除所有客户端会话（仅适合测试）。

Type: Boolean

Default: 0

`mds wipe ino prealloc`

Description: Ceph will delete ino preallocation metadata on startup (for testing only).

ceph 将在启动时删除索引节点号预分配元数据（仅适合测试）。

Type: Boolean

Default: 0

`mds skip ino`

Description: The number of inode numbers to skip on startup (for testing only).

启动时要跳过的索引节点号数量（仅适合测试）。

Type: 32-bit Integer

Default: 0

`mds standby for name`

Description: An MDS daemon will standby for another MDS daemon of the name specified in this setting.

指定一 MDS 守护进程的名字，此进程将作为它的候补。

Type: String

Default: N/A

`mds standby for rank`

Description: An MDS daemon will standby for an MDS daemon of this rank.

此 MDS 将作为本机架上 MDS 守护进程的候补。

Type: 32-bit Integer

Default: -1

`mds standby replay`

Description: Determines whether a `ceph-mds` daemon should poll and replay the log of an active MDS (hot standby).

决定一 `ceph-mds` 守护进程是否应该滚动并重放活跃 MDS 的日志（热备）。

Type: Boolean

Default: false

4.3 Journaler

`journaler allow split entries`

Description: Allow an entry to span a stripe boundary

允许一条目跨越条带边界。

Type: Boolean

Required: No

Default: true

`journaler write head interval`

Description: How frequently to update the journal head object
多频繁地更新日志头部对象?

Type: Integer
Required: No
Default: 15

journaler prefetch periods

Description: How many stripe periods to read-ahead on journal replay
重放日志时预读多少条带。

Type: Integer
Required: No
Default: 10

journal prezero periods

Description: How many stripe periods to zero ahead of write position
把写位置前的多少条带清零。

Type: Integer
Required: No
Default: 10

journaler batch interval

Description: Maximum additional latency in seconds we incur artificially.
人为导致的最大额外延时。

Type: Double
Required: No
Default: .001

journaler batch max

Description: Maximum bytes we'll delay flushing.
我们会延迟回写的最大字节数。

Type: 64-bit Unsigned Integer
Required: No
Default: 0

4.4 ceph-mds 在线手册

ceph-mds——ceph 元数据服务器守护进程

ceph-mds – ceph metadata server daemon

4.4.1 提纲

Synopsis

ceph-mds -i name [[-hot-standby [rank]] | [-journal_check rank]]

4.4.2 描述

Description

ceph-mds is the metadata server daemon for the Ceph distributed file system. One or more instances of ceph-mds collectively manage the file system namespace, coordinating access to the shared OSD cluster.

ceph-mds 是 ceph 分布式文件系统的元数据服务器守护进程。一或多个 ceph-mds 例程协作着管理文件系统的命名空间、协调到共享 OSD 集群的访问。

Each ceph-mds daemon instance should have a unique name. The name is used to identify daemon instances in the ceph.conf.

各 ceph-mds 守护进程例程都应该有惟一的名字，此名用于在 ceph.conf 里标识例程。

Once the daemon has started, the monitor cluster will normally assign it a logical rank, or put it in a standby pool to take over for another daemon that crashes. Some of the specified options can cause other behaviors.

一旦守护进程启动，监视器集群会给它分配一个逻辑机架，或把它放入一个候补存储池用于接替其它崩溃的进程。一些特定选项会导致其它行为。

If you specify hot-standby or journal-check, you must either specify the rank on the command line, or specify one of the mds_standby_for_[rank|name] parameters in the config. The command line specification overrides the config, and specifying the rank overrides specifying the name.

如果你指定了热备或日志检查选项，还必须在命令行加上机架参数、或在配置文件里指定 mds_standby_for_[rank|name]之一。命令行中指定的会覆盖配置文件，指定机架会覆盖指定的名字。

4.4.3 选项

Options

-f, --foreground

Foreground: do not daemonize after startup (run in foreground). Do not generate a pid file. Useful when run via [ceph-run\(8\)](#).

前台：启动后不要进入后台（在前台运行），不要生成 pid 文件。通过 ceph-run(8) 调用时有用。

-d

Debug mode: like -f, but also send all log output to stderr.

调试模式：像 -f，但它也把所有输出日志发到了标准错误。

-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.

在启动时用 ceph.conf 而非默认的 /etc/ceph/ceph.conf 来确定监视器地址。

-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 查找）。

4.4.4 使用范围

Availability

`ceph-mon` is part of the Ceph distributed file system. Please refer to the Ceph documentation at <http://ceph.com/docs> for more information.

`ceph-mon` 是 ceph 分布式文件系统的一部分，更多内容请参考 <http://ceph.com/docs>。

4.4.5 参考

See also

[ceph\(8\)](#), [ceph-mon\(8\)](#), [ceph-osd\(8\)](#)

4.5 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、密钥。

```
sudo mount -t ceph 192.168.0.1:6789:/ /mnt/mycephfs -o  
name=admin,secret=AQATSKdNGBnwLhAAAnNDKnH65FmVKpXZJ VasUeQ==
```

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[认证概览](#)。

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 `umount` 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`。

4.6 用户空间挂载 ceph 文件系统

Mount ceph FS as a fuse

For Ceph version 0.55 and later, cephx authentication is on by default. Before mounting a Ceph File System in User Space (FUSE), ensure that the client host has a copy of the Ceph configuration file and a keyring with CAPS for the Ceph metadata server.

ceph-0.55 及后续版本默认开启了 cephx 认证。从用户空间 (FUSE) 挂载一 ceph 文件系统前，确保客户端主机有一份 ceph 配置副本、和具备 ceph 元数据服务能力的密钥环。

1. From your client host, copy the Ceph configuration file from the monitor host to the `/etc/ceph` directory.

在客户端主机上，把监视器主机上的 ceph 配置文件拷贝到 `/etc/ceph/` 目录下。

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

2. From your client host, copy the Ceph keyring from the monitor host to to the `/etc/ceph` directory.

在客户端主机上，把监视器主机上的 ceph 密钥环拷贝到 `/etc/ceph` 目录下。

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

3. Ensure that the Ceph configuration file and the keyring have appropriate permissions set on your client machine (e.g., `chmod 644`).

确保客户端机器上的 ceph 配置文件和密钥环都有合适的权限位，如 `chmod 644`。

For additional details on cephx configuration, see [CEPHX Config Reference](#).

cephx 配置的详细信息见 [CEPHX Config Reference](#)。

To mount the Ceph file system as a 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，可以用 `ceph-fuse` 命令，例如：

```
sudo mkdir /home/username/cephfs  
sudo ceph-fuse -m 192.168.0.1:6789 /home/username/cephfs
```

See [ceph-fuse](#) for additional details.

详情参见 [ceph-fuse 在线手册](#)。

4.7 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.

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。

4.7.1 内核驱动

Kernel Driver

To mount Ceph FS in your file systems table as a kernel driver, add the following to /etc/fstab:

要用内核驱动挂载 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,noatime      0  
2
```

Important The name and secret or secretfile options are mandatory when you have Ceph authentication running.

重要：启用了认证时，`name` 及 `secret` 或 `secretfile` 选项是强制的。

See [Authentication](#) for details.

详情参见[认证概览](#)。

4.7.2 FUSE

To mount Ceph FS in your file systems table as a filesystem in user space, add the following to /etc/fstab:

要在用户空间挂载 ceph 文件系统，按如下加入 /etc/fstab：

```
#DEVICE          PATH        TYPE        OPTIONS  
id={user-ID}[,conf={path/to/conf.conf}] /mount/path  fuse.ceph defaults 0 0
```

For example:

例如：

```
id=admin  /mnt/ceph  fuse.ceph defaults 0 0  
id=myuser,conf=/etc/ceph/cluster.conf  /mnt/ceph2  fuse.ceph defaults 0 0
```

The DEVICE field is a comma-delimited list of options to pass to the command line. Ensure you use the ID (e.g., `admin`, not `client.admin`) . You can pass any valid `ceph-fuse` option to the command line this way.

DEVICE 字段是逗号分隔的一系列选项，确保填上了 ID (如 `admin`，不是 `client.admin`) 。你可以用此方法把任何 `ceph-fuse` 接受的选项加上。

See [Authentication](#) for details.

详情见[Authentication](#)。

4.8 cephfs 在线手册

cephfs – ceph file system options utility

cephfs——ceph 文件系统选项工具

4.8.1 提纲

Synopsis

cephfs [path command options]

4.8.2 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，用于查看和修改 ceph 分布式文件系统内的文件布局和定位数据。

Choose one of the following three commands:

支持下列三个子命令：

- **show_layout** View the layout information on a file or directory
show layout 查看一文件或目录的布局信息
- **set_layout** Set the layout information on a file or directory
set layout 设置一文件或目录的布局信息
- **show_location** View the location information on a file
show location 查看一文件的定位信息

4.8.3 选项

Options

Your applicable options differ depending on whether you are setting or viewing layout/location.

使用设置或查看布局/位置命令时，可用的可选参数不同。

4.8.3.1 查看选项：

Viewing options:

-l --offset

Specify an offset for which to retrieve location data

检索（指定文件的）定位数据时指定个偏移量。

4.8.3.2 设置选项

Setting options:

-u --stripe_unit

Set the size of each stripe

设置各条带的尺寸。

-c --stripe_count

Set the number of objects to stripe across

指定条带跨越的对象数

-s --object_size

Set the size of the objects to stripe across

指定条带跨越的对象尺寸

-p --pool

Set the pool (by numeric value, not name!) to use

指定目标存储池（数字，而非名字）

-o --osd

Set the preferred OSD to use as the primary

指定这些 OSD 优先作为主 OSD

4.8.4 使用限制

Limitations

When setting layout data, the specified object size must evenly divide by the specified stripe unit. Any parameters you don't set explicitly are left at the system defaults.

设置布局数据时，指定的对象尺寸必须被指定的条带单位平均分割，未指定的参数都按系统默认值。

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.

显式地设置一文件和目录布局的含义不同。设置一文件的布局指示如何放置这一个文件，而且必须在写入数据前就设置好。删减文件也不允许你更改其布局。

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.

设置一目录的布局设置了“默认布局”，它将用于设置此目录内后续新建文件（或子目录）的文件布局。先前已存在文件的布局不会变。

You'll notice that the layout information allows you to specify a preferred OSD for placement. This feature is unsupported and ignored in modern versions of the Ceph servers; do not use it.

也许你注意到了布局信息允许你指定优先使用的 OSD。此功能已废弃，较新版本的 ceph 服务器会忽略它，别用。

4.8.5 使用范围

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，更多信息参见 <http://ceph.com/docs>。

4.8.6 参考

See also

[ceph\(8\)](#)

4.9 ceph-fuse 在线手册

ceph-fuse – FUSE-based client for ceph

ceph-fuse 是基于 FUSE 的 ceph 客户端。

4.9.1 提纲

Synopsis

ceph-fuse [-m monaddr:port] mountpoint [fuse options]

4.9.2 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 ceph 分布式文件系统的 FUSE（用户空间文件系统）客户端，它会把 ceph 文件系统（用 -m 选项或 ceph.conf 指定）挂载到指定挂载点。

The file system can be unmounted with:

文件系统可这样卸载：

```
fusermount -u mountpoint
```

or by sending SIGINT to the ceph-fuse process.

或向 ceph-fuse 进程发送 SIGINT 信号。

4.9.3 选项

Options

Any options not recognized by ceph-fuse will be passed on to libfuse.

ceph-fuse 识别不了的选项将传递给 libfuse。

-d

Detach from console and daemonize after startup.

启动后将脱离终端、进入守护状态。

-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.

用指定的 ceph.conf 而非默认的 /etc/ceph/ceph.conf 来查找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，而不是从 ceph.conf 里找。

-r root_directory

Use root_directory as the mounted root, rather than the full Ceph tree.

把文件系统内的 root_directory 作为根挂载，而不是整个 ceph 文件系统树。

4.9.4 使用范围

Availability

ceph-fuse 是 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>。

4.9.5 参考

See also

fusermount(8), [ceph\(8\)](#)

4.10 mount.ceph 在线手册

mount.ceph – mount a ceph file system

mount.ceph——用于挂载 ceph 文件系统

4.10.1 提纲

Synopsis

mount.ceph monaddr1[,monaddr2,...]:/[subdir] dir [-o options]

4.10.2 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` 是在 Linux 主机上挂载 ceph 文件系统的简单助手。它只负责把监视器主机名解析为 IP 地址、从硬盘读取认证密钥，大多数实际工作由 Linux 内核客户端组件完成。事实上，无需认证的 ceph 文件系统无需 `mount.ceph` 也能挂载，只要指定监视器 IP 地址即可：

```
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.

各监视器地址 monaddr 的格式为 host[:port]，如果端口未指定，就用默认的 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。要成功地挂载只需一个监视器即可，客户端将从某个能响应的监视器获知其它监视器。然而最好指定多个监视器，以免挂载时正好赶上那个监视器挂了。

A subdirectory subdir may be specified if a subset of the file system is to be mounted.

如果要挂载文件系统的一子集，可指定一个子目录 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` 助手程序的惯例是前两个选项分别为要挂载的设备和目标路径，其它选项必须位于这些固定参数之后。

4.10.3 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

整数。默认：5

mount_timeout

int (seconds), Default: 60

整数（秒）。默认：60

osd_idle_ttl

int (seconds), Default: 60

整数（秒）。默认：60

caps_wanted_delay_min

int, cap release delay, Default: 5

整数，能力释放延迟时间。默认：5

caps_wanted_delay_max

int, cap release delay, Default: 60

整数，能力释放延迟时间。默认：60

cap_release_safety

int, Default: calculated

整数。默认：自行计算

readdir_max_entries

int, Default: 1024

整数。默认：1024

readdir_max_bytes

int, Default: 524288 (512*1024)

整数。默认：524288 (512*1024)

write_congestion_kb

int (kb), max writeback in flight. scale with available memory. Default: calculated from available memory

整数 (kb) , 运行中的最大回写量，随可用内存变化。默认：根据可用内存计算

snapdirname

string, set the name of the hidden snapdir. Default: .snap

字符串，为快照的隐藏目录设置个名字。默认：.snap

name

RADOS user to authenticate as when using cephx. Default: guest

使用 cephx 认证时的 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 的密钥。这个选项不安全，因为它把密钥暴露在了命令行，用 secretfile 选项避免此问题。

secretfile

path to file containing the secret key to use with cephx

用于 cephx 的密钥文件路径。

ip

my ip

本机 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，而不是和挂载同一集群的例程共享资源。

dirstat

funky cat dirname for stats, Default: off

用 cat dirname 读取文件信息。默认：关闭

nodirstat

no funky cat dirname for stats

不用 cat dirname 读取文件信息

rbytes

Report the recursive size of the directory contents for st_size on directories. Default: on

目录的 st_size 报告产生于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.

目录的 st_size 无需通过递归目录内容来获取。

nocrc

no data crc on writes

写入时不做 ctc 校验

noasync readdir

no dcache readdir

读目录时不经过 dcache

4.10.4 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:

如果有多个监视器：

```
mount.ceph monhost1,monhost2,monhost3:/ /mnt/foo
```

If [ceph-mon](#)(8) is running on a non-standard port:

如果 ceph-mon(8) 运行于非默认端口：

```
mount.ceph monhost1:7000,monhost2:7000,monhost3:7000:/ /mnt/foo
```

To mount only part of the namespace:

只挂载文件系统命名空间的一部分：

```
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) 安装正确，mount(8) 应该能自动调用它：

```
mount -t ceph monhost:/ /mnt/foo
```

4.10.5 使用范围

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.

mount.ceph 是 ceph 分布式文件系统的一部分，更多信息参见 <http://ceph.com/docs>。

4.10.6 参考

See also

[ceph-fuse](#)(8), [ceph](#)(8)

4.11 让 hadoop 使用 cephfs

Using hadoop with cephfs

The Ceph file system can be used as a drop-in replacement for the Hadoop File System (HDFS). This page describes the installation and configuration process of using Ceph with Hadoop.

ceph 文件系统可作为 Hadoop 文件系统 (HDFS) 的落地式替代品，本章描述了 ceph 用于 hadoop 存储的安装和配置过程。

4.11.1 依赖关系

Dependencies

- CephFS Java Interface

- Hadoop CephFS Plugin

Important: Currently requires Hadoop 1.1.X stable series

重要: 当前要求 Hadoop 1.1.x 稳定版系列。

4.11.2 安装

Installation

There are three requirements for using CephFS with Hadoop. First, a running Ceph installation is required. The details of setting up a Ceph cluster and the file system are beyond the scope of this document. Please refer to the Ceph documentation for installing Ceph.

ceph 用于 hadoop 有三个必要条件。首先，必须有一个运行的 ceph。建设一个 ceph 集群和文件系统不是本章的讨论范围，请参考 ceph 安装文档。

The remaining two requirements are a Hadoop installation, and the Ceph file system Java packages, including the Java CephFS Hadoop plugin. The high-level steps are to add the dependencies to the Hadoop installation CLASSPATH, and configure Hadoop to use the Ceph file system.

另外两个必要条件是 Hadoop 安装和 ceph 文件系统 Java 软件包，包括 Java 的 CephFS Hadoop 插件。后续步骤分别是把依赖添加到 Hadoop 的环境变量 CLASSPATH、更改 Hadoop 配置让它使用 Ceph 文件系统。

4.11.2.1 CephFS Java Packages

- CephFS Hadoop plugin ([hadoop-cephfs.jar](#))

Adding these dependencies to a Hadoop installation will depend on your particular deployment. In general the dependencies must be present on each node in the system that will be part of the Hadoop cluster, and must be in the CLASSPATH searched for by Hadoop. Typically approaches are to place the additional jar files into the `hadoop/lib` directory, or to edit the `HADOOP_CLASSPATH` variable in `hadoop-env.sh`.

安装 Hadoop 时的依赖取决于你怎么部署的，一般来说，这些依赖必须安装到 Hadoop 集群的各节点、且必须能从 CLASSPATH 里找到。典型做法是把另加的 jar 文件放入 `hadoop/lib` 目录、或编辑 `hadoop-env.sh` 中的 `HADOOP_CLASSPATH` 变量。

The native Ceph file system client must be installed on each participating node in the Hadoop cluster.

Hadoop 集群内的各参与节点都必须安装原生 ceph 文件系统客户端。

4.11.3 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
<code>fs.default.name</code>	Ceph URI	<code>ceph://[monaddr:port]/</code>
<code>ceph.conf.file</code>	Local path to ceph.conf	<code>/etc/ceph/ceph.conf</code>
<code>ceph.conf.options</code>	Comma separated list of Ceph configuration key/value pairs	<code>opt1=val1,opt2=val2</code>
<code>ceph.root.dir</code>	Mount root directory	Default value: /
<code>ceph.mon.address</code>	Monitor address	host:port
<code>ceph.auth.id</code>	Ceph user id	Example: admin
<code>ceph.auth.keyfile</code>	Ceph key file	
<code>ceph.auth.keyring</code>	Ceph keyring file	
<code>ceph.object.size</code>	Default file object size in bytes	Default value (64MB): 67108864
<code>ceph.data.pools</code>	List of Ceph data pools for storing file.	Default value: default Ceph pool.
<code>ceph.localize.reads</code>	Allow reading from file replica objects	Default value: true

4.11.3.1 对每文件定制复制的支持

Support For Per-file Custom Replication

The Hadoop file system interface allows users to specify a custom replication factor (e.g. 3 copies of each block) when creating a file. However, object replication factors in the Ceph file system are controlled on a per-pool basis, and by default a Ceph file system will contain only a single pre-configured pool. Thus, in order to support per-file replication with Hadoop over Ceph, additional storage pools with non-default replications factors must be created, and Hadoop must be configured to choose from these additional pools.

Hadoop 文件系统接口允许用户在创建文件时定制复制因子（例如每块 3 个副本）。然而对象复制因子在 ceph 里是以每存储池为单位进行控制的，并且 ceph 文件系统默认会包含一个预配置的存储池。因此，在 ceph 之上运行 hadoop 时，为支持每文件复制策略，必须创建多个有非默认复制因子的存储池，且必须配置 hadoop 让它自动选择合适存储池。

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 pools `pool1`, `pool2`, and `pool5` when selecting a target pool to store a file.

额外的数据存储池用 `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存储池。例如，下面的配置会使 hadoop 在为文件选择存储池时考虑 `pool1`、`pool2`、`pool5`：

```
<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。要创建一个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` 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 is 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，必须把新存储池可用于数据存储的消息告知元数据服务，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 by examining the output of 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>
```

存储池选择语义

Pool Selection Rules

The following rules describe how Hadoop chooses a pool given a desired replication factor and the set of pools specified using the `ceph.data.pools` configuration option.

下面的规则描述了 Hadoop 如何根据期望复制因子从 `ceph.data.pools` 配置的一堆存储池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。

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:rep1=<name>:<value> wanted=<value>
```

4.12 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](#)。

4.13 挂载故障排除

Troubleshooting

4.13.1 mount 5 Error

A mount 5 error typically occurs if a MDS server is laggy or if it crashed. Ensure at least one MDS is up and running, and the cluster is `active + healthy`.

mount 5 错误通常是 MDS 服务器滞后或崩溃导致的。要确保至少有一个 MDS 是启动且运行的，集群也要处于 `active+healthy` 状态。

4.13.2 mount 12 Error

A mount 12 error with `cannot allocate memory` usually occurs if you have a version mismatch between the [Ceph Client](#) version and the [Ceph Storage Cluster](#) version. Check the versions using:

mount 12 错误显示 `cannot allocate memory`，常见于客户端和 ceph 存储集群版本不匹配。用以下命令检查版本：

```
ceph -v
```

If the Ceph Client is behind the Ceph cluster, try to upgrade it:

如果 ceph 客户端版本落后于集群，试着升级它：

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

You may need to uninstall, autoclean and autoremove `ceph-common` and then reinstall it so that you have the latest version.

你也许得卸载、清理和删除 `ceph-common`，然后再重新安装，以确保安装的是最新版。

5 ceph 块设备

Ceph Block Device

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 capabilities such as snapshotting, replication and consistency. Ceph's 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 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 Ceph Block Devices, you must have access to 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 rbd-fuse](#)
- [Manpage ceph-rbdnamer](#)
- [librbd](#)

5.1 块设备命令

Block Device Commands

The `rbd` command enables you to create, list, introspect and remove block device images. You can also use it to clone images, create snapshots, rollback an image to a snapshot, view a snapshot, etc. For details on using the `rbd` command, see [RBD – Manage RADOS Block Device \(RBD\) Images](#) for details.

`rbd` 命令可用于创建、罗列、内省和删除块设备映像，也可克隆映像、创建快照、回滚快照、查看快照等等。`rbd` 命令用法详情见 [RBD – Manage RADOS Block Device \(RBD\) Images](#)。

Important: To use Ceph Block Device commands, you must have access to a running Ceph cluster.

重要：要使用 ceph 块设备命令，你必须有对应集群的访问权限。

5.1.1 创建块设备映像

Creating a Block Device Image

Before you can add a block device to a node, you must create an image for it in the [Ceph Storage Cluster](#) first. To create a block device image, execute the following:

要想把块设备加入某节点，你得先在 ceph 存储集群中创建一个映像，用下列命令：

```
rbd create {image-name} --size {megabytes} --pool {pool-name}
```

For example, to create a 1GB image named `foo` that stores information in a pool named `swimmingpool`, execute the following:

例如，要在 `swimmingpool` 这个存储池中创建一个名为 `foo`、大小为 1GB 的映像，执行下列命令：

```
rbd create foo --size 1024
rbd create bar --size 1024 --pool swimmingpool
```

Note: You must create a pool first before you can specify it as a source. See [Storage Pools](#) for details.

注意：指定此存储池前必须先创建它，详情见 [Storage Pools](#)。

5.1.2 罗列块设备映像

Listing Block Device Images

To list block devices in the `rbd` pool, execute the following (i.e., `rbd` is the default pool name):

要罗列 `rbd` 存储池中的块设备，用下列命令（如 `rbd` 是默认存储池名字）：

```
rbd ls
```

To list block devices in a particular pool, execute the following, but replace `{poolname}` with the name of the pool:

用下列命令罗列某个特定存储池中的块设备，用存储池名字替换掉 `{poolname}`：

```
rbd ls {poolname}
```

For example:

例如：

```
rbd ls swimmingpool
```

5.1.3 检索映像信息

Retrieving Image Information

To retrieve information from a particular image, execute the following, but replace {image-name} with the name for the image:

用下列命令检索某特定映像的信息，用{image-name}替换映像名字：

```
rbd --image {image-name} info
```

For example:

例如：

```
rbd --image foo info
```

To retrieve information from an image within a pool, execute the following, but replace {image-name} with the name of the image and replace {pool-name} with the name of the pool:

用下列命令检索某存储池内一映像的信息，用{image-name}替换掉映像名字、用{pool-name}替换掉存储池名字：

```
rbd --image {image-name} -p {pool-name} info
```

For example:

例如：

```
rbd --image bar -p swimmingpool info
```

5.1.4 更改块设备映像尺寸

Resizing a Block Device Image

[Ceph Block Device](#) images are thin provisioned. They don't actually use any physical storage until you begin saving data to them. However, they do have a maximum capacity that you set with the `--size` option. If you want to increase (or decrease) the maximum size of a Ceph Block Device image, execute the following:

ceph 块设备映像是瘦设备，只有在你开始写入数据时它们才会占用物理空间。然而，它们都有最大容量，就是你设置的`--size`选项。如果你想增加（或减小）ceph 块设备映像的最大尺寸，用下列命令：

```
rbd resize --image foo --size 2048
```

5.1.5 删除块设备映像

Removing a Block Device Image

To remove a block device, execute the following, but replace {image-name} with the name of the image you want to remove:

用下列命令删除块设备，用{image-name}替换映像名字：

```
rbd rm {image-name}
```

For example:

例如：

```
rbd rm foo
```

To remove a block device from a pool, execute the following, but replace {image-name} with the name of the image to remove and replace {pool-name} with the name of the pool:

用下列命令从某存储池中删除一个块设备，用{image-name}替换要删除的映像名、用{pool-name}替换存储池名字：

```
rbd rm {image-name} -p {pool-name}
```

For example:

例如：

```
rbd rm bar -p swimmingpool
```

5.2 内核模块操作

Kernel Module Operations

Important: To use kernel module operations, you must have a running Ceph cluster.

重要: 要用内核模块操作，必须有一个运行着的 ceph 集群。

5.2.1 加载 ceph RBD 模块

Load the Ceph RBD Module

To map a block device image to a kernel module, first load the Ceph RBD module:

要把一个块设备映像映射为内核模块，先加载 RBD 模块：

```
modprobe rbd
```

5.2.2 获取映像列表

Get a List of Images

To mount a block device image, first return a list of the images.

要挂载映像设备，先罗列映像。

```
rbd list
```

5.2.3 映射块设备

Map a Block Device

Use **rbd** to map an image name to a kernel module. You must specify the image name, the pool name, and the user name.

用 **rbd** 把映像名映射为内核模块。必须指定映像名、存储池名、和用户名。

```
sudo rbd map {image-name} --pool {pool-name} --id {user-name}
```

For example:

例如：

```
sudo rbd map foo --pool rbd myimage --id admin
```

If you use [cephx](#) authentication, you must also specify a secret. It may come from a keyring or a file containing the secret.

如果你启用了 cephx 认证，你必须提供密钥，可以从密钥环或文件获取。

```
sudo rbd map --pool rbd myimage --id admin --keyring /path/to/keyring
sudo rbd map --pool rbd myimage --id admin --keyfile /path/to/file
```

5.2.4 查看已映射块设备

Show Mapped Block Devices

To show block device images mapped to kernel modules with the **rbd** command, specify the **showmapped** option.

可以用 **rbd** 命令的 **showmapped** 选项查看映射为内核模块的块设备映像。

```
sudo rbd showmapped
```

5.2.5 取消块设备映射

Unmapping a Block Device

To unmap a block device image with the **rbd** command, specify the **rm** option and the device name (i.e., by convention the same as the block device image name).

要取消块设备映射，用 **rbd** 命令、指定 **rm** 选项和设备名（即为方便起见使用的同名块设备映像）。

```
sudo rbd unmap /dev/rbd/{poolname}/{imagename}
```

For example:

例如：

```
sudo rbd unmap /dev/rbd/rbd/foo
```

5.3 快照

Snapshots

A snapshot is a read-only copy of the state of an image at a particular point in time. One of the advanced features of Ceph block devices is that you can create snapshots of the images to retain a history of an image's state. Ceph also supports snapshot layering, which allows you to clone images (e.g., a VM image) quickly and easily. Ceph supports block device snapshots using the `rbd` command and many higher level interfaces, including [QEMU](#), [libvirt](#), [OpenStack](#) and [CloudStack](#).

一份快照是某映像在一个特定时间点的一份只读副本。`ceph` 块设备的一个高级功能就是你可以为映像创建快照来保留其历史；`ceph` 还支持分层快照，让你快速、容易地克隆映像（如 VM 映像）。`ceph` 的快照功能支持 `rbd` 命令和多种高级接口，包括 [QEMU](#), [libvirt](#), [OpenStack](#) 和 [CloudStack](#)。

Important: To use RBD snapshots, you must have a running Ceph cluster.

重要：要使用 RBD 快照功能，你必须有个运行着的集群。

Note: STOP I/O BEFORE snapshotting an image. If the image contains a filesystem, the filesystem must be in a consistent state BEFORE snapshotting.

注意：先停止 I/O 操作再拍映像的快照，如果映像内包含文件系统，拍快照前确保文件系统是一致的。



5.3.1 cephx 注意事项

Cephx Notes

When [cephx](#) is enabled, you must specify a user and a secret file on the command line, or use the `CEPH_ARGS` environment variable to avoid re-entry of the following parameters.

启用了 `cephx` 时，你必须在命令行指定用户名和密钥文件，或者用 `CEPH_ARGS` 环境变量避免重复输入下列参数。

```
rbd --id {user-name} --keyring=/path/to/secret [commands]
```

For example:

例如：

```
rbd --id client.admin --keyring=/etc/ceph/ceph.keyring [commands]
```

Tip: Add the user and secret to the `CEPH_ARGS` environment variable so that you don't need to enter them each time.

提示：把用户名和密钥写入 `CEPH_ARGS` 环境变量，省得每次手动输入。

5.3.2 快照基础

Snapshot Basics

The following procedures demonstrate how to create, list, and remove snapshots using the `rbd` command on the command line.

下列过程演示了如何用 `rbd` 命令创建、罗列、和删除快照。

5.3.2.1 创建快照

Create Snapshot

To create a snapshot with rbd, specify the snap create option, the pool name and the image name.

用 rbd 命令创建快照，要指定 snap create 选项、外加存储池名、映像名。

```
rbd --pool {pool-name} snap create --snap {snap-name} {image-name}
rbd snap create {pool-name}/{image-name}@{snap-name}
```

For example:

例如：

```
rbd --pool rbd snap create --snap snapname foo
rbd snap create rbd/foo@snapname
```

5.3.2.2 罗列快照

List Snapshots

To list snapshots of an image, specify the pool name and the image name.

要列出一映像的快照，指定存储池名和映像名。

```
rbd --pool {pool-name} snap ls {image-name}
rbd snap ls {pool-name}/{image-name}
```

For example:

例如：

```
rbd --pool rbd snap ls foo
rbd snap ls rbd/foo
```

5.3.2.3 回滚快照

Rollback Snapshot

To rollback to a snapshot with rbd, specify the snap rollback option, the pool name, the image name and the snap name.

要用 rbd 回滚到一映像，指定 snap rollback 选项、存储池名、映像名和快照名。

```
rbd --pool {pool-name} snap rollback --snap {snap-name} {image-name}
rbd snap rollback {pool-name}/{image-name}@{snap-name}
```

For example:

例如：

```
rbd --pool rbd snap rollback --snap snapname foo
rbd snap rollback rbd/foo@snapname
```

For the rollback section, you could mention that rollback means overwriting the current version with data from a snapshot, and takes longer with larger images. So cloning is preferable for fast recovery.

关于回滚，你可以认为回滚意味着用快照中的数据覆盖当前版本，而且映像较大时耗时较长。所以要想快速恢复的话用克隆比较好。

Note: Rolling back an image to a snapshot means overwriting the current version of the image with data from a snapshot. The time it takes to execute a rollback increases with the size of the image. It is **faster to clone** from a snapshot **than to rollback** an image to a snapshot, and it is the preferred method of returning to a pre-existing state.

注意：把映像回滚到一快照的意思是，用快照中的数据覆盖映像的当前版本，此过程花费的时间随映像尺寸增长。从快照克隆要快于把映像回滚到此快照，这也是回到先前状态的首选方法。

5.3.2.4 删除快照

Delete a Snapshot

To delete a snapshot with rbd, specify the snap rm option, the pool name, the image name and the

username.

要用 rbd 删除一快照，指定 snap rm 选项、存储池名、映像名和用户名。

```
rbd --pool {pool-name} snap rm --snap {snap-name} {image-name}
rbd snap rm {pool-name}/{image-name}@{snap-name}
```

For example:

例如：

```
rbd --pool rbd snap rm --snap snapname foo
rbd snap rm rbd/foo@snapname
```

Note: Ceph OSDs delete data asynchronously, so deleting a snapshot doesn't free up the disk space immediately.

注意：ceph 的 OSD 异步地删除数据，所以删除快照后不会立即释放磁盘空间。

5.3.2.5 清除快照

Purge Snapshots

To delete all snapshots for an image with rbd, specify the snap purge option and the image name.

要用 rbd 删除一映像的所有快照，指定 snap purge 选项和映像名。

```
rbd --pool {pool-name} snap purge {image-name}
rbd snap purge {pool-name}/{image-name}
```

For example:

例如：

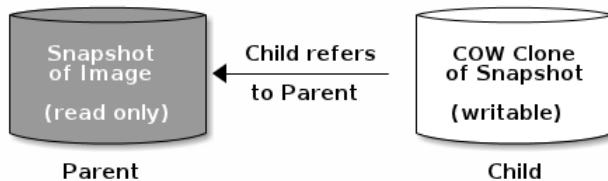
```
rbd --pool rbd snap purge foo
rbd snap purge rbd/foo
```

5.3.3 分层

Layering

Ceph supports the ability to create many copy-on-write (COW) clones of a block device snapshot. Snapshot layering enables Ceph block device clients to create images very quickly. For example, you might create a block device image with a Linux VM written to it; then, snapshot the image, protect the snapshot, and create as many copy-on-write clones as you like. A snapshot is read-only, so cloning a snapshot simplifies semantics—making it possible to create clones rapidly.

ceph 支持创建某一设备快照的很多写时复制（COW）克隆。分层快照使得 ceph 块设备客户端可以很快地创建映像，例如，你可以创建一个块设备映像，其中有 Linux VM；然后拍快照、保护快照，再创建任意多写时复制克隆。快照是只读的，所以克隆快照的语义简单的多，克隆它也可以很迅速。



Note: The terms “parent” and “child” mean a Ceph block device snapshot (parent), and the corresponding image cloned from the snapshot (child). These terms are important for the command line usage below.

注意：这里的术语“父”和“子”意思是一个 ceph 块设备快照（父），和从此快照克隆出来的对应映像（子）。这些术语对下列的命令行用法来说很重要。

Each cloned image (child) stores a reference to its parent image, which enables the cloned image to open the parent snapshot and read it.

各个克隆出来的映像（子）都存储着对父映像的引用，这使得克隆出来的映像可以打开父映像并读取它。

A COW clone of a snapshot behaves exactly like any other Ceph block device image. You can read to, write from, clone, and resize cloned images. There are no special restrictions with cloned images. However, the

copy-on-write 克隆指的是快照，所以你 **MUST** 在克隆前保护快照。以下的图描述了此过程。

一个快照的 COW 克隆和其它任何 ceph 块设备映像的行为完全一样。克隆出的映像没有特别的限制，你可以读出、写入、克隆、调整其大小，然而快照的写时复制克隆引用了快照，所以你克隆前必须保护它。下图描述了此过程。

Note: Ceph only supports cloning for `format 2` images (i.e., created with `rbd create --format 2`), and is not yet supported by the kernel `rbd` module. So you **MUST** use QEMU/KVM or `librbd` directly to access clones in the current release.

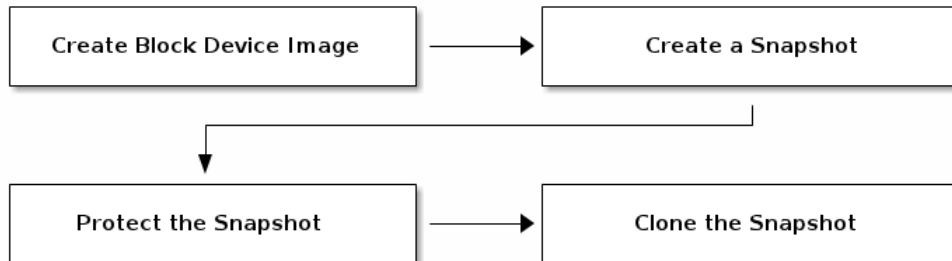
注意：ceph 仅支持克隆 `format 2` 映像（即用 `rbd create --format 2` 创建的），而且内核 `rbd` 模块还不支持。所以现在你只能用 QEMU/KVM 或 `librbd` 直接访问克隆品。

5.3.3.1 分层入门

Getting Started with Layering

Ceph block device layering is a simple process. You must have an image. You must create a snapshot of the image. You must protect the snapshot. Once you have performed these steps, you can begin cloning the snapshot.

ceph 块设备的分层是个简单的过程。你必须有个映像、必须为它创建快照、必须保护快照，执行过这些步骤后，你才能克隆快照。



The cloned image has a reference to the parent snapshot, and includes the pool ID, image ID and snapshot ID. The inclusion of the pool ID means that you may clone snapshots from one pool to images in another pool.

克隆出的映像包含到父快照的引用、也包含存储池 ID、映像 ID 和快照 ID。包含存储池 ID 意味着你可以把一存储池内的快照克隆到别的存储池。

1. **Image Template:** A common use case for block device layering is to create a master image and a snapshot that serves as a template for clones. For example, a user may create an image for a Linux distribution (e.g., Ubuntu 12.04), and create a snapshot for it. Periodically, the user may update the image and create a new snapshot (e.g., `sudo apt-get update`, `sudo apt-get upgrade`, `sudo apt-get dist-upgrade` followed by ```rbd snap create```). As the image matures, the user can clone any one of the snapshots.

映像模板：块设备分层的一个常见用法是创建一个主映像及其快照，并作为模板以供克隆。例如，一用户创建一 Linux 发行版（如 Ubuntu 12.04）的映像、并为其拍快照；此用户可能会周期性地更新映像、并创建新的快照（如在 `rbd snap create` 之后执行 `sudo apt-get update`、`sudo apt-get upgrade`、`sudo apt-get dist-upgrade`），当映像成熟时，用户可以克隆任意快照。

2. **Extended Template:** A more advanced use case includes extending a template image that provides more information than a base image. For example, a user may clone an image (e.g., a VM template) and install other software (e.g., a database, a content management system, an analytics system, etc.) and then snapshot the extended image, which itself may be updated just like the base image.

扩展模板：更高级的用法包括扩展映像模板，让它包含比基础映像更多的信息。例如，用户可以克隆一个映像（如 VM 模板）、然后安装其它软件（如数据库、内容管理系统、分析系统等等）、然后为此扩展映像拍快照，拍下的快照可以像基础映像一样更新。

3. **Template Pool:** One way to use block device layering is to create a pool that contains master images that act as templates, and snapshots of those templates. You may then extend read-only privileges to users so that they may clone the snapshots without the ability to write or execute within the pool.

模板存储池：块设备分层的一种用法是创建一存储池，其中包含作为模板的主映像、和那些模板的快

照。然后把只读权限分给用户，这样他们就可以克隆快照了，而无需分配此存储池内的写和执行权限。

4. **Image Migration/Recovery:** One way to use block device layering is to migrate or recover data from one pool into another pool.

映像迁移/恢复：块设备分层的一种用法是把一存储池内的数据迁移或恢复到另一存储池。

5.3.3.2 保护快照

Protecting a Snapshot

Clones access the parent snapshots. All clones would break if a user inadvertently deleted the parent snapshot. To prevent data loss, you **MUST** protect the snapshot before you can clone it.

克隆品要访问父快照。如果哪个用户不小心删除了父快照，所有克隆品都会损坏。为防止数据丢失，必须先保护、然后再克隆快照。

```
rbd --pool {pool-name} snap protect --image {image-name} --snap {snapshot-name}  
rbd snap protect {pool-name}/{image-name}@{snapshot-name}
```

For example:

例如：

```
rbd --pool rbd snap protect --image my-image --snap my-snapshot  
rbd snap protect rbd/my-image@my-snapshot
```

Note: You cannot delete a protected snapshot.

注意：你删除不了受保护的快照。

5.3.3.3 克隆快照

Cloning a Snapshot

To clone a snapshot, specify you need to specify the parent pool, image and snapshot; and, the child pool and image name. You must protect the snapshot before you can clone it.

要克隆快照，你得指定父存储池、映像、和快照，还有子存储池和映像名。克隆前必须先保护它。

```
rbd --pool {pool-name} --image {parent-image} --snap {snap-name} --dest-pool {pool-name} --dest  
{child-image}  
rbd clone {pool-name}/{parent-image}@{snap-name} {pool-name}/{child-image-name}
```

For example:

例如：

```
rbd clone rbd/my-image@my-snapshot rbd/new-image
```

Note: You may clone a snapshot from one pool to an image in another pool. For example, you may maintain read-only images and snapshots as templates in one pool, and writeable clones in another pool.

注意：你可以把一存储池中映像的快照克隆到另一存储池。例如，你可以把一存储池中的只读映像及其快照当模板维护、却把可写克隆置于另一存储池。

5.3.3.4 取消快照保护

Unprotecting a Snapshot

Before you can delete a snapshot, you must unprotect it first. Additionally, you may **NOT** delete snapshots that have references from clones. You must flatten each clone of a snapshot, before you can delete the snapshot.

删除快照前，必须先取消保护。另外，你不能删除被克隆品引用的快照，所以删除快照前必须先拍平此快照的各个克隆。

```
rbd --pool {pool-name} snap unprotect --image {image-name} --snap {snapshot-name}  
rbd snap unprotect {pool-name}/{image-name}@{snapshot-name}
```

For example:

例如：

```
rbd --pool rbd snap unprotect --image my-image --snap my-snapshot  
rbd snap unprotect rbd/my-image@my-snapshot
```

5.3.3.5 罗列一快照的子孙

Listing Children of a Snapshot

To list the children of a snapshot, execute the following:

用下列命令罗列一快照的子孙：

```
rbd --pool {pool-name} children --image {image-name} --snap {snap-name}  
rbd children {pool-name}/{image-name}@{snapshot-name}
```

For example:

例如：

```
rbd --pool rbd children --image my-image --snap my-snapshot  
rbd children rbd/my-image@my-snapshot
```

5.3.3.6 拍平克隆品映像

Flattening a Cloned Image

Cloned images retain a reference to the parent snapshot. When you remove the reference from the child clone to the parent snapshot, you effectively “flatten” the image by copying the information from the snapshot to the clone. The time it takes to flatten a clone increases with the size of the snapshot. To delete a snapshot, you must flatten the child images first.

克隆来的映像仍保留了父快照的引用。要从子克隆删除这些到父快照的引用，你可以把快照的信息复制给子克隆，也就是“拍平”它。拍平克隆品的时间因快照尺寸而不同。要删除快照，必须先拍平子映像。

```
rbd --pool {pool-name} flatten --image {image-name}  
rbd flatten {pool-name}/{image-name}
```

For example:

例如：

```
rbd --pool rbd flatten --image my-image  
rbd flatten rbd/my-image
```

Note: Since a flattened image contains all the information from the snapshot, a flattened image will take up more storage space than a layered clone.

注意：因为拍平的映像包含了快照的所有信息，所以拍平的映像占用的存储空间会比分层克隆品大。

5.4 QEMU 和块设备

QEMU and Block Devices

The most frequent Ceph Block Device use case involves providing block device images to virtual machines. For example, a user may create a “golden” image with an OS and any relevant software in an ideal configuration. Then, the user takes a snapshot of the image. Finally, the user clones the snapshot (usually many times). See [S snapshots](#) for details. The ability to make copy-on-write clones of a snapshot means that Ceph can provision block device images to virtual machines quickly, because the client doesn’t have to download an entire image each time it spins up a new virtual machine.

ceph 块设备最常见的用法之一是作为虚拟机的块设备映像。例如，一用户可创建名为 **golden** 的映像，并安装、配置好了操作系统和相关软件，然后为此映像拍下快照，最后再克隆此快照（通常很多次）。详情参见 [S snapshots](#)，能制作快照的写时复制克隆品意味着 ceph 可以快速地为虚拟机提供块设备映像，因为客户端每次启动一个新虚拟机时不必下载整个映像。



Ceph Block Devices can integrate with the QEMU virtual machine. For details on QEMU, see [QEMU Open Source Processor Emulator](#). For QEMU documentation, see [QEMU Manual](#).

ceph 块设备可以和 QEMU 虚拟机集成到一起，关于 QEMU 请参考 [QEMU Open Source Processor Emulator](#), 其文档：[QEMU Manual](#)。

Important: To use Ceph Block Devices with QEMU, you must have access to a running Ceph cluster.

重要：要让 QEMU 使用 ceph 块设备，你必须有个运行着的 ceph 集群。

5.4.1 安装 QEMU (12.04 及后续版本)

Installing QEMU (12.04 Precise and later)

QEMU packages are incorporated into Ubuntu 12.04 Precise Pangolin and later versions. To install QEMU, execute the following:

QEMU 软件包已经集成进了 Ubuntu 12.04 及后续版本，用下列命令安装：

```
sudo apt-get install qemu
```

5.4.2 安装 QEMU (11.10 及更早版本)

Installing QEMU (11.10 Oneric and earlier)

For Ubuntu distributions 11.10 Oneiric and earlier, you must install the 0.15 version of QEMU or later. To build QEMU from source, use the following procedure:

对于 Ubuntu 发行版的 11.10 及更早版本，必须安装 0.15 版及之后的 QEMU。根据下列步骤从源码编译：

```
cd {your-development-directory}
git clone git://git.qemu.org/qemu.git
cd qemu
./configure --enable-rbd
make; make install
```

5.4.3 用 QEMU 创建映像

Creating Images with QEMU

You can create a block device image from QEMU. You must specify `rbd`, the pool name, and the name of the image you wish to create. You must also specify the size of the image.

你可以用 QEMU 创建块设备映像。必须指定 `rbd`、存储池名、要创建的映像名以及映像尺寸。

```
qemu-img create -f rbd rbd:{pool-name}/{image-name} {size}
```

For example:

例如：

```
qemu-img create -f rbd rbd:data/foo 10G
```

5.4.4 用 QEMU 更改映像尺寸

Resizing Images with QEMU

You can resize a block device image from QEMU. You must specify `rbd`, the pool name, and the name of the image you wish to resize. You must also specify the size of the image.

你可以通过 QEMU 调整块设备尺寸。必须指定 `rbd`、存储池名、要调整的映像名，还有映像尺寸。

```
qemu-img resize -f rbd rbd:{pool-name}/{image-name} {size}
```

For example:

例如：

```
qemu-img resize -f rbd rbd:data/foo 10G
```

5.4.5 用 QEMU 检索映像信息

Retrieving Image Info with QEMU

You can retrieve block device image information from QEMU. You must specify `rbd`, the pool name, and the name of the image.

你可以用 QEMU 检索块设备映像信息。必须指定 `rbd`、存储池名、和映像名。

```
qemu-img info -f rbd rbd:{pool-name}/{image-name}
```

For example:

例如：

```
qemu-img info -f rbd rbd:data/foo
```

5.4.6 通过 RBD 运行 QEMU

Running QEMU with RBD

QEMU can pass a block device from the host on to a guest, but since QEMU 0.15, there's no need to map an image as a block device on the host. Instead, QEMU can access an image as a virtual block device directly via `librbd`. This performs better because it avoids an additional context switch, and can take advantage of [RBD caching](#).

QEMU 能把一主机上的块设备传递给访客，但从 QEMU 0.15 起，不需要在主机上把映像映射为块设备了。QEMU 现在能直接用 `librbd` 把映像当虚拟块设备访问了，这样性能更好，因为它避免了额外的上下文切换，而且能利用 [RBD 缓存](#)。

You can use `qemu-img` to convert existing virtual machine images to Ceph block device images. For example, if you have a qcow2 image, you could run:

你可以用 `qemu-img` 把已有的虚拟机映像转换为 ceph 块设备映像，比如你有一个 qcow2 映像，可以这样转：

```
qemu-img convert -f qcow2 -O rbd debian_squeeze.qcow2 rbd:data/squeeze
```

To run a virtual machine booting from that image, you could run:

要从那个映像引导虚拟机，执行：

```
qemu -m 1024 -drive format=raw,file=rbd:data/squeeze
```

[RBD caching](#) can significantly improve performance. Since QEMU 1.2, QEMU's cache options control `librbd` caching:

[RBD 缓存](#) 可显著提升性能。从 QEMU 1.2 起，缓存选项可控制 `librbd` 缓存：

```
qemu -m 1024 -drive format=rbd,file=rbd:data/squeeze,cache=writeback
```

If you have an older version of QEMU, you can set the `librbd` cache configuration (like any Ceph configuration option) as part of the 'file' parameter:

如果你的 QEMU 版本较老，你可以用 `file` 参数更改 `librbd` 缓存配置（就像其它 `ceph` 配置选项一样）：

```
qemu -m 1024 -drive format=raw,file=rbd:data/squeeze:rbd_cache=true,cache=writeback
```

Important: If you set `rbd_cache=true`, you must set `cache=writeback` or risk data loss. Without `cache=writeback`, QEMU will not send flush requests to `librbd`. If QEMU exits uncleanly in this configuration, filesystems on top of `rbd` can be corrupted.

重要：如果你设置了 `rbd_cache=true`，那就必须设置 `cache=writeback` 否则有可能丢数据。没有 `cache=writeback`，QEMU 就不会向 `librbd` 发送回写请求，如果 QEMU 退出时未清理干净，`rbd` 之上的文件系统就有可能被篡改。

5.4.7 启用放弃或修剪功能

Enabling Discard/TRIM

Since Ceph version 0.46 and QEMU version 1.1, Ceph Block Devices support the discard operation. This means that a guest can send TRIM requests to let a Ceph block device reclaim unused space. This can be enabled in the guest by mounting `ext4` or XFS with the `discard` option.

从 ceph 0.46 和 QEMU 1.1 起，ceph 块设备支持放弃操作，这意味着访客可以发送 TRIM 请求来让 ceph 块设备回收未使用的空间。此功能可在访客上挂载 `ext4` 或 `xfs` 时用 `discard` 选项启用。

For this to be available to the guest, it must be explicitly enabled for the block device. To do this, you must specify a `discard_granularity` associated with the drive:

要使此功能对访客可用，必须对块设备显式启用。为此，你必须指定在驱动器上指定 `discard_granularity`。

```
qemu -m 1024 -drive format=raw,file=rbd:data/squeeze,id=drive1,if=none \
      -device ide-hd,drive=drive1,discard_granularity=512
```

Note that this uses the IDE driver. The virtio driver does not support discard.

注意这个使用 IDE 驱动，virtio 驱动不支持 discard。

If using libvirt, edit your libvirt domain's configuration file using `virsh edit` to include the `xmlns:qemu` value. Then, add a `qemu:commandline` block as a child of that domain. The following example shows how to set two devices with `qemu id=` to different `discard_granularity` values.

如果用的是 libvirt，得用 `virsh edit` 编辑配置文件，加上 `xmlns:qemu` 值。然后加一个 `qemu:commandline` 块作为那个域的子域。下例展示了如何用 `qemu id=` 为两个设备设置不同的 `discard_granularity` 值。

```
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
  <qemu:commandline>
    <qemu:arg value='-set' />
    <qemu:arg value='block.scsi0-0-0.discard_granularity=4096' />
    <qemu:arg value='-set' />
    <qemu:arg value='block.scsi0-0-1.discard_granularity=65536' />
  </qemu:commandline>
</domain>
```

5.4.8 QEMU 缓存选项

QEMU Cache Options

QEMU's cache options correspond to the following Ceph [RBD Cache](#) settings.

QEMU 的缓存选项对应下列的 ceph [RBD 缓存](#) 选项。

Writeback:

回写：

```
rbd_cache = true
```

Writethrough:

写透：

```
rbd_cache = true
rbd_cache_max_dirty = 0
```

None:

无：

```
rbd_cache = false
```

QEMU's cache settings override Ceph's default settings (i.e., settings that are not explicitly set in the Ceph configuration file). If you explicitly set [RBD Cache](#) settings in your Ceph configuration file, your Ceph settings override the QEMU cache settings. If you set cache settings on the QEMU command line, the QEMU command line settings override the Ceph configuration file settings.

QEMU 的缓存选项会覆盖 ceph 的默认选项（就是那些 ceph 配置文件里没有的选项）；如果你在 ceph 配置文件内设置了 RBD 缓存选项，那么它们会覆盖 QEMU 缓存选项。如果你在 QEMU 命令行上设置缓存选项，它们会覆盖 ceph 配置文件里的选项。

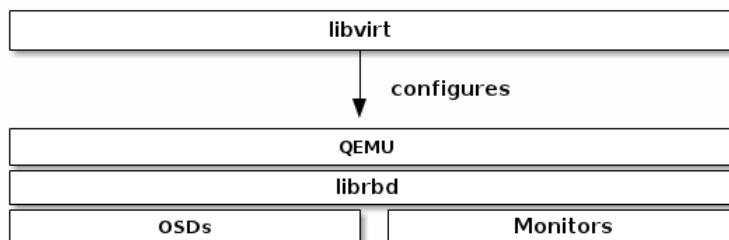
5.5 Using libvirt with Ceph RBD

The `libvirt` library creates a virtual machine abstraction layer between hypervisor interfaces and the software applications that use them. With `libvirt`, developers and system administrators can focus on a common management framework, common API, and common shell interface (i.e., `virsh`) to many different hypervisors, including:

- QEMU/KVM
- XEN
- LXC
- VirtualBox
- etc.

Ceph block devices support QEMU/KVM. You can use Ceph block devices with software that interfaces with `libvirt`. The following stack diagram illustrates how `libvirt` and QEMU use Ceph block devices via `librbd`.

ceph 块设备支持 QEMU/KVM，所以你可以在 ceph 块设备之上运行能与 `libvirt` 交互的软件。下面的堆栈图解释了 `libvirt` 和 QEMU 如何通过 `librbd` 使用 ceph 块设备。



The most common `libvirt` use case involves providing Ceph block devices to cloud solutions like OpenStack or CloudStack. The cloud solution uses `libvirt` to interact with QEMU/KVM, and QEMU/KVM interacts with Ceph block devices via `librbd`. See [Block Devices and OpenStack](#) and [Block Devices and CloudStack](#) for details.

`libvirt` 常见于为云解决方案提供 ceph 块设备，像 OpenStack、CloudStack，它们用 `libvirt` 和 QEMU/KVM 交互、QEMU/KVM 再与 ceph 块设备通过 `librbd` 交互。详情见 [Block Devices and OpenStack](#) 和 [Block Devices and CloudStack](#)。

You can also use Ceph block devices with `libvirt`, `virsh` and the `libvirt` API. See [libvirt Virtualization API](#) for details.

你也可以通过 `libvirt`、`virsh` 和 `libvirt` API 使用 ceph 块设备，详情见 [libvirt Virtualization API](#)。

5.5.1 先决条件

Prerequisites

- [Install](#) and [configure](#) a Ceph cluster
- [Install and configure](#) QEMU/KVM

5.5.2 在 Ubuntu 12.04 上安装 libvirt

Installing `libvirt` on Ubuntu 12.04 Precise

`libvirt` packages are incorporated into the Ubuntu 12.04 precise distribution. To install `libvirt` on precise, execute the following:

`libvirt` 软件包已进入 Ubuntu 12.04 发行版的仓库，可执行下列命令安装：

```
sudo apt-get update && sudo apt-get install libvirt-bin
```

5.5.3 在 Ubuntu 较老版本上安装 libvirt

Installing libvirt on Earlier Versions of Ubuntu

For Ubuntu distributions 11.10 oneiric and earlier, you must build libvirt from source. Clone the libvirt repository, and use [AutoGen](#) to generate the build. Then, execute `make` and `make install` to complete the installation. For example:

对于 Ubuntu 11.10 及更早版本，你得从源码编译 libvirt。先克隆 libvirt 源码库，然后用 autogen 生成可构建源码，然后执行 make && make install 完成安装。例如：

```
git clone git://libvirt.org/libvirt.git
cd libvirt
./autogen.sh
make
sudo make install
```

See [libvirt Installation](#) for details.

详情见 [libvirt Installation](#)。

5.5.4 ceph 用于虚拟机

Using Ceph with Virtual Machines

To create VMs that use Ceph block devices, use the procedures in the following sections. In the exemplary embodiment, we've used `libvirt-pool` for the pool name, `client.libvirt` for the user name, and `new-libvirt-image` for the image name. You may use any value you like, but ensure you replace those values when executing commands in the subsequent procedures.

要创建使用 ceph 块设备的虚拟机，请看后面几段。具体应用时，我们用 `libvirt-pool` 作为存储池名、`client.libvirt` 作为用户名、`new-libvirt-image` 作为映像名，你可以任意命名，确保在后续过程中用自己的名字替换掉对应名字即可。

5.5.4.1 配置 ceph

Configuring Ceph

To configure Ceph for use with libvirt, perform the following steps:

要把 ceph 用于 libvirt，执行下列步骤：

1. [Create a pool](#) (or use the default). The following example uses the pool name `libvirt-pool` with 128 placement groups.

创建一存储池（或者用默认的）。本例用 `libvirt-pool` 作存储池名，配备了 128 个归置组。

```
ceph osd pool create libvirt-pool 128 128
```

Verify the pool exists.

验证存储池是否存在。

```
ceph osd ls pools
```

2. [Create a Ceph Name](#) (or use `client.admin` for version 0.9.7 and earlier). The following example uses the Ceph name `client.libvirt` and references `libvirt-pool`.

新建一 ceph 用户名（0.9.7 版之前的话用 `client.admin`），本例用 `client.libvirt`、且权限限制到 `libvirt-pool`。

```
ceph auth get-or-create client.libvirt mon 'allow r' osd 'allow class-read object_prefix
rbd_children, allow rwx pool=libvirt-pool'
```

Verify the name exists.

验证名字是否存在。

```
ceph auth list
```

NOTE: libvirt will access Ceph using the ID `libvirt`, not the Ceph name `client.libvirt`. See [Cephx Commandline](#) for detailed explanation of the difference between ID and name.

注意：libvirt 访问 ceph 时将用 libvirt 作为 ID，而不是 client.libvirt。关于 ID 和名字不同的详细解释见于 [Cephx Commandline](#)。

3. Use QEMU to [create an image](#) in your RBD pool. The following example uses the image name `new-libvirt-image` and references `libvirt-pool`.

用 QEMU 在 RBD 存储池中创建一映像。本例中映像名为 `new-libvirt-image`、存储池为 `libvirt-pool`。

```
qemu-img create -f rbd rbd:libvirt-pool/new-libvirt-image 2G
```

Verify the image exists.

验证映像是否存在。

```
rbd -p libvirt-pool ls
```

NOTE: You can also use [rbd create](#) to create an image, but we recommend ensuring that QEMU is working properly.

注意：你也可以用 `rbd create` 创建映像，但我们建议顺便确认下 QEMU 可正常运行。

5.5.4.2 准备虚拟机管理器

Preparing the VM Manager

You may use `libvirt` without a VM manager, but you may find it simpler to create your first domain with `virt-manager`.

即使没 VM 管理器你也可以用 `libvirt`，但是用 `virt-manager` 创建域更简单。

1. Install a virtual machine manager. See [KVM/VirtManager](#) for details.

安装个虚拟机管理器，详情见 [KVM/VirtManager](#)。

```
sudo apt-get install virt-manager
```

2. Download an OS image (if necessary).

下载一 OS 映像。

3. Launch the virtual machine manager.

启动虚拟机管理器。

```
sudo virt-manager
```

5.5.4.3 新建虚拟机

Creating a VM

To create a VM with `virt-manager`, perform the following steps:

要用 `virt-manager` 创建 VM，按下列步骤：

1. Press the **Create New Virtual Machine** button.

点击 Create New Virtual Machine 按钮。

2. Name the new virtual machine domain. In the exemplary embodiment, we use the name `libvirt-virtual-machine`. You may use any name you wish, but ensure you replace `libvirt-virtual-machine` with the name you choose in subsequent commandline and configuration examples.

为新虚拟机命名，本例中我们用 `libvirt-virtual-machine`，你可以任意命名，在后续命令行和配置实例中替换掉 `libvirt-virtual-machine` 即可。

```
libvirt-virtual-machine
```

3. Import the image.

导入映像。

```
/path/to/image/recent-linux.img
```

NOTE: Import a recent image. Some older images may not rescan for virtual devices properly.

注意：导入一个近期映像，一些较老的映像未必能正确地重扫描虚拟设备。

4. Configure and start the VM.

配置并启动 VM。

5. You may use `virsh list` to verify the VM domain exists.

用 `virsh list` 验证 VM 域存在。

```
sudo virsh list
```

6. Login to the VM (root/root)

登入 VM (root/root)

7. Stop the VM before configuring it for use with Ceph.

配置它使用 ceph 前停止 VM。

5.5.4.4 配置 VM

Configuring the VM

When configuring the VM for use with Ceph, it is important to use `virsh` where appropriate. Additionally, `virsh` commands often require root privileges (i.e., `sudo`) and will not return appropriate results or notify you that that root privileges are required. For a reference of `virsh` commands, refer to [Virsh Command Reference](#).

配置 VM 使用 ceph 时，切记尽量用 `virsh`。另外，`virsh` 命令通常需要 root 权限（如 `sudo`），否则不会返回正确结果或提示你需要 root 权限，`virsh` 命令参考见 [Virsh Command Reference](#)。

1. Open the configuration file with `virsh edit`.

用 `virsh edit` 打开配置文件。

```
sudo virsh edit {vm-domain-name}
```

Under `<devices>` there should be a `<disk>` entry.

`<devices>` 下应该是 `<disk>` 条目。

```
<devices>
    <emulator>/usr/bin/kvm</emulator>
    <disk type='file' device='disk'>
        <driver name='qemu' type='raw'/>
        <source file='/path/to/image/recent-linux.img' />
        <target dev='vda' bus='virtio' />
        <address type='drive' controller='0' bus='0' unit='0' />
    </disk>
```

Replace `/path/to/image/recent-linux.img` with the path to the OS image. The minimum kernel for using the faster `virtio` bus is 2.6.25. See [Virtio](#) for details.

IMPORTANT: Use `sudo virsh edit` instead of a text editor. If you edit the configuration file under `/etc/libvirt/qemu` with a text editor, `libvirt` may not recognize the change. If there is a discrepancy between the contents of the XML file under `/etc/libvirt/qemu` and the result of `sudo virsh dumpxml {vm-domain-name}`, then your VM may not work properly.

用你的 OS 映像路径取代 `/path/to/image/recent-linux.img`，可利用较快的 `virtio` 总线的最低内核版本是 2.6.25，参见 [Virtio](#)。

重要：要用 `sudo virsh edit` 而非文本编辑器，如果你用文本编辑器编辑了 `/etc/libvirt/qemu` 下的配置文件，`libvirt` 未必能感知你做的更改。如果 `/etc/libvirt/qemu` 下的 XML 文件和 `sudo virsh dumpxml {vm-domain-name}` 输出结果内容不同，VM 可能会运行异常。

2. Add the Ceph RBD image you created as a `<disk>` entry.

把你创建的 ceph RBD 映像创建为 `<disk>` 条目。

```
<disk type='network' device='disk'>
    <source protocol='rbd' name='libvirt-pool/new-libvirt-image'>
        <host name='{monitor-host}' port='6789' />
    </source>
```

```
<target dev='hdb' bus='ide'/>
</disk>
```

Replace `{monitor-host}` with the name of your host, and replace the pool and/or image name as necessary. You may add multiple `<host>` entries for your Ceph monitors. The `dev` attribute is the logical device name that will appear under the `/dev` directory of your VM. The optional `bus` attribute indicates the type of disk device to emulate. The valid settings are driver specific (e.g., “ide”, “scsi”, “virtio”, “xen”, “usb” or “sata”).

See [Disks](#) for details of the `<disk>` element, and its child elements and attributes.

用你的主机名替换`{monitor-host}`、可能还有存储池、映像名。你可以为 ceph 监视器添加多条 `<host>`, `dev` 属性是将出现在 VM 之 `/dev` 目录下的逻辑设备名, 可选的 `bus` 属性是要模拟的磁盘类型。可用和驱动相关, 如 ide、scsi、virtio、xen、usb 或 sata。

关于 `<disk>` 标签及其子标签和属性, 详见 [Disks](#)。

3. Save the file.

保存文件。

4. If you are using [Ceph Authentication](#), you must generate a secret.

如果你在用 [Ceph Authentication](#), 必须生成密钥。

```
cat > secret.xml <<EOF
<secret ephemeral='no' private='no'>
    <usage type='ceph'>
        <name>client.libvirt secret</name>
    </usage>
</secret>
EOF
```

5. Define the secret.

定义密钥。

```
sudo virsh secret-define --file secret.xml
<uuid of secret is output here>
```

6. Get the `client.libvirt` key and save the key string to a file.

获取 `client.libvirt` 密钥并把字符串保存于文件。

```
sudo ceph auth list
vim client.libvirt.key
```

7. Set the UUID of the secret.

设置密钥的 UUID。

```
sudo virsh secret-set-value --secret {uuid of secret} --base64 $(cat client.libvirt.key) && rm
client.libvirt.key secret.xml
```

You must also set the secret manually by adding the following `<auth>` entry to the `<disk>` element you entered earlier (replacing the `uuid` value with the result from the command line example above).

还必须手动设置密钥, 把下面的 `<auth>` 条目添加到前面的 `<disk>` 标签内 (用上一命令的输出结果替换掉 `uuid` 值)。

```
sudo virsh edit {vm-domain-name}
```

Then, add `<auth></auth>` element to the domain configuration file:

然后, 把 `<auth></auth>` 标签加进域配置文件:

```
...
</source>
<auth username='libvirt'>
    <secret type='ceph' uuid='9ec59067-fdbc-a6c0-03ff-df165c0587b8' />
</auth>
<target ...>
```

NOTE: The exemplary ID is `libvirt`, not the Ceph name `client.libvirt` as generated at step 2 of [Configuring Ceph](#). Ensure you use the ID component of the Ceph name you generated. If for some reason you need to regenerate the secret, you will have to execute `sudo virsh secret-`

`undefine {uuid}` before executing `sudo virsh secret-set-value` again.

注意：示例 ID 是 `libvirt`，不是 [Configuring Ceph](#) 生成的 ceph 名 `client.libvirt`，确保你用的是 ceph 名的 ID 部分。如果出于某些原因你需要更换密钥，必须先执行 `sudo virsh secret-undefine {uuid}`，然后再执行 `sudo virsh secret-set-value`。

5.5.4.5 总结

Summary

Once you have configured the VM for use with Ceph, you can start the VM. To verify that the VM and Ceph are communicating, you may perform the following procedures.

完成上面的配置后你就可以启动 VM 了，为确认 VM 和 ceph 在通讯，你可以执行如下过程。

1. Check to see if Ceph is running:

检查 ceph 在运行。

```
ceph health
```

2. Check to see if the VM is running.

检查 VM 在运行。

```
sudo virsh list
```

3. Check to see if the VM is communicating with Ceph. Replace `{vm-domain-name}` with the name of your VM domain:

检查 VM 是否在和 ceph 通讯，用你的 VM 域名字替换 `{vm-domain-name}`:

```
sudo virsh qemu-monitor-command --hmp {vm-domain-name} 'info block'
```

4. Check to see if the device from `<target dev='hdb' bus='ide'>` appears under `/dev` or under `proc/partitions`.

检查一下 `<target dev='hdb' bus='ide'>` 定义的设备是否出现在 `/dev` 或 `/proc/partitions` 里。

```
ls dev  
cat proc/partitions
```

If everything looks okay, you may begin using the Ceph block device within your VM.

如果看起来一切正常，你就可以在 VM 内使用 ceph 块设备了。

5.6 librbd 缓存配置

librbd Cache Settings

See [Block Device](#) for additional details.

详见 [Block Device](#)。

Kernel Caching

The kernel driver for Ceph block devices can use the Linux page cache to improve performance.

内核缓存

ceph 块设备的内核驱动可利用 Linux 页缓存来提升性能。

The user space implementation of the Ceph block device (i.e., `librbd`) cannot take advantage of the Linux page cache, so it includes its own in-memory caching, called “RBD caching.” RBD caching behaves just like well-behaved hard disk caching. When the OS sends a barrier or a flush request, all dirty data is written to the OSDs. This means that using write-back caching is just as safe as using a well-behaved physical hard disk with a VM that properly sends flushes (i.e. Linux kernel $\geq 2.6.32$). The cache uses a Least Recently Used (LRU) algorithm, and in write-back mode it can coalesce contiguous requests for better throughput.

ceph 块设备的用户空间实现（即 `librbd`）不能利用 Linux 页缓存，所以它自己实现了内存缓存，名为“RBD 缓存”。RBD 缓存行为就像硬盘缓存一样端正，当 OS 发送了 barrier 或 flush 请求时，所有脏数据都会写入 OSD，这意味着只要 VM 会正确地发送回写命令（即内核版本大于 2.6.32），使用回写缓存和常见物理硬盘一样安全。此缓存用最近最少使用（Least Recently Used, LRU）算法，而且在回写模式下它能合并相邻请求以提高吞吐量。

New in version 0.46.

0.46 新增。

Ceph supports write-back caching for RBD. To enable it, add `rbd cache = true` to the `[client]` section of your `ceph.conf` file. By default `librbd` does not perform any caching. Writes and reads go directly to the storage cluster, and writes return only when the data is on disk on all replicas. With caching enabled, writes return immediately, unless there are more than `rbd cache max dirty` unflushed bytes. In this case, the write triggers writeback and blocks until enough bytes are flushed.

ceph 支持为 RBD 做写回缓存，要启用此功能，在 ceph.conf 配置文件的 [client] 段下添加 `rbd cache = true`。librbd 默认不会进行任何缓存，写和读都直接到达存储集群，而且所有数据都完成复制后写动作才会返回；启用缓存后，写动作会立即返回，除非未回写的字节数大于 `rbd cache max dirty`，这种情况下，写动作会触发回写机制并一直阻塞着，直到回写完了足够多的字节数。

New in version 0.47.

0.47 新增。

Ceph supports write-through caching for RBD. You can set the size of the cache, and you can set targets and limits to switch from write-back caching to write through caching. To enable write-through mode, set `rbd cache max dirty` to 0. This means writes return only when the data is on disk on all replicas, but reads may come from the cache. The cache is in memory on the client, and each RBD image has its own. Since the cache is local to the client, there's no coherency if there are others accessing the image. Running GFS or OCFS on top of RBD will not work with caching enabled.

ceph 支持为 RBD 写透做缓存。你可以设置缓存尺寸、还能设置从写回缓存切换到写透缓存的目标和临界点。要启用写透模式，把 `rbd cache max dirty` 设为 0，这意味着数据的所有复制都完成时写才会返回，但是读可以来自缓存。在客户端，缓存位于内存中，且个 RBD 映像有自己的缓存。对客户端来说正因为缓存位于本地，所以对映像的访问没有相干性。打开缓存时，在 RBD 之上不能运行 GFS 或 OCFS。

The `ceph.conf` file settings for RBD should be set in the `[client]` section of your configuration file. The settings include:

RBD 选项应该位于 ceph.conf 配置文件的 [client] 段下，可用选项有：

`rbd cache`

Description: Enable caching for RADOS Block Device (RBD).
允许为 RADOS 块设备提供缓存。

Type: Boolean
Required: No
Default: `false`

`rbd cache size`

Description: The RBD cache size in bytes.
RBD 缓存尺寸，字节。

Type: 64-bit Integer
Required: No
Default: `32 MiB`

`rbd cache max dirty`

Description: The `dirty` limit in bytes at which the cache triggers write-back. If 0, uses write-through caching.
使缓存触发写回的 `dirty` 临界点，若为 0，直接使用写透缓存。

Type: 64-bit Integer
Required: No
Constraint: Must be less than `rbd cache size`.
必须小于 `rbd cache size`。
Default: `24 MiB`

`rbd cache target dirty`

Description: The `dirty target` before the cache begins writing data to the data storage. Does not block writes to the cache.
缓存开始写回数据的目的地 `dirty target`，不会阻塞到缓存的写动作。

Type: 64-bit Integer

Required: No

Constraint: Must be less than rbd cache max dirty.

必须小于 rbd cache max dirty。

Default: 16 MiB

rbd cache max dirty age

Description: The number of seconds dirty data is in the cache before writeback starts.

写回开始前，脏数据在缓存中的暂存时间。

Type: Float

Required: No

Default: 1.0

New in version 0.60.

0.60 新增。

rbd cache writethrough until flush

Description: Start out in write-through mode, and switch to write-back after the first flush request is received. Enabling this is a conservative but safe setting in case VMs running on rbd are too old to send flushes, like the virtio driver in Linux before 2.6.32.

开始进入写透模式，并且在首个 flush 请求收到后切回写回模式。启用它保守但安全，以防 rbd 之上的虚拟机内核太老、不能发送 flush，像 2.6.32 之前的 virtio 驱动。

Type: Boolean

Required: No

Default: false

5.7 块设备和 OpenStack

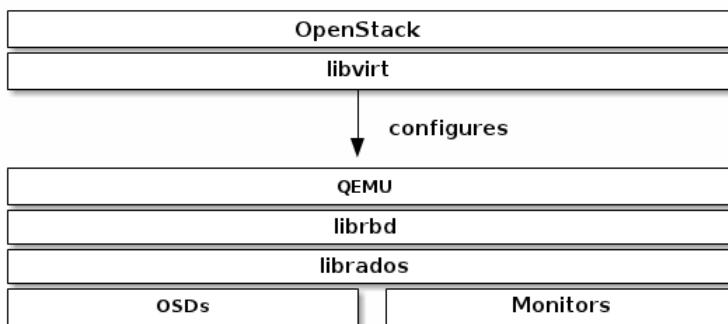
Block Devices and OpenStack

You may use Ceph Block Device images with OpenStack through `libvirt`, which configures the QEMU interface to `librbd`. Ceph stripes block device images as objects across the cluster, which means that large Ceph Block Device images have better performance than a standalone server!

通过 libvirt 你可以把 ceph 块设备用于 OpenStack，它配置了 QEMU 到 librbd 的接口。ceph 把块设备分块为对象并分布到集群中，这意味着大个的 ceph 块设备映像其性能会比独立服务器更好。

To use Ceph Block Devices with OpenStack, you must install QEMU, `libvirt`, and OpenStack first. We recommend using a separate physical host for your OpenStack installation. OpenStack recommends a minimum of 8GB of RAM and a quad-core processor. The following diagram depicts the OpenStack/Ceph technology stack.

要把 ceph 块设备用于 OpenStack，必须先安装 QEMU、libvirt 和 OpenStack。我们建议用一台独立的物理主机安装 OpenStack，此主机最少需 8GB 内存和一个 4 核 CPU。下面的图表描述了 OpenStack/Ceph 技术栈。



Important: To use Ceph Block Devices with OpenStack, you must have access to a running Ceph Storage Cluster.

重要：要让 OpenStack 使用 ceph 块设备，你必须有相应的 ceph 集群访问权限。

Two parts of OpenStack integrate with Ceph's block devices:

OpenStack 里有两个地方要和 ceph 块设备结合：

- **Images:** OpenStack Glance manages images for VMs. Images are immutable. OpenStack treats images as binary blobs and downloads them accordingly.
映像：OpenStack 的 Glance 管理着 VM 的映像。映像相对恒定，OpenStack 把它们当作二进制大数数据块、并按需下载。
- **Volumes:** Volumes are block devices. OpenStack uses volumes to boot VMs, or to attach volumes to running VMs. OpenStack manages volumes using `nova-volume` prior to the Folsom release. OpenStack manages volumes using Cinder services beginning with the Folsom release.

卷宗：卷宗是块设备，OpenStack 用它们引导虚拟机、或贴在卷上运行 VM。在 OpenStack 的 Folsom 之前的发布中用 `nova-volume` 管理卷宗、从 Folsom 开始用 Cinder 服务管理卷宗。

Beginning with OpenStack Folsom and Ceph 0.52, you can use OpenStack Glance to store images in a Ceph Block Device, and you can use Cinder or `nova-volume` to boot a VM using a copy-on-write clone of an image.

从 OpenStack Folsom 和 Ceph 0.52 开始，你可以用 OpenStack Glance 把映像存储到 ceph 块设备中，还可以用 Cinder 或 `nova-volume` 来引导用映像的写时复制克隆制成的虚拟机。

The instructions below detail the setup for Glance and Nova/Cinder, although they do not have to be used together. You may store images in Ceph block devices while running VMs using a local disk, or vice versa.

后面的指令详细描述 Glance 和 Nova/Cinder 的安装设置，虽然它们不一定一起用。你可以在本地硬盘上运行 VM、却把映像存储于 ceph 块设备，反之亦然。

5.7.1 创建存储池

Create a Pool

By default, Ceph block devices use the `rbd` pool. You may use any available pool. We recommend creating a pool for Nova/Cinder and a pool for Glance. Ensure your Ceph cluster is running, then create the pools.

默认情况下，ceph 块设备使用 `rbd` 存储池，你可以用任何可用存储池。我们建议分别为 Nova/Cinder 和 Glance 创建存储池。确保 ceph 集群在运行，然后创建存储池。

```
ceph osd pool create volumes 128
ceph osd pool create images 128
```

See [Create a Pool](#) for detail on specifying the number of placement groups for your pools, and [Placement Groups](#) for details on the number of placement groups you should set for your pools.

参考 [Create a Pool](#) 为存储池指定归置组数量，参考 [Placement Groups](#) 确定应该为存储池分配多少归置组。

5.7.2 配置 OpenStack 的 ceph 客户端

Configure OpenStack Ceph Clients

The hosts running `glance-api`, `nova-compute`, and `nova-volume` or `cinder-volume` act as Ceph clients. Each requires the `ceph.conf` file:

运行着 `glance-api`、`nova-compute`、`nova-volume` 或 `cinder-volume` 的主机被当作 ceph 客户端，它们都需要 `ceph.conf` 文件。

```
ssh {your-openstack-server} sudo tee /etc/ceph/ceph.conf </etc/ceph/ceph.conf
```

5.7.2.1 安装 ceph 客户端软件包

Install Ceph client packages

On the `glance-api` host, you'll need the Python bindings for `librbd`:

在运行 `glance-api` 的主机上你需要 `librbd` 的 python 接口：

```
sudo apt-get install python-ceph
```

On the `nova-volume` or `cinder-volume` host, use the client command line tools:

运行 `nova-volume` 或 `cinder-volume` 的主机需要客户端命令行工具：

```
sudo apt-get install ceph-common
```

5.7.2.2 配置ceph客户端认证

Setup Ceph Client Authentication

If you have [cephx authentication](#) enabled, create a new user for Nova/Cinder and Glance.

For Ceph version 0.53 or lower, execute the following:

如果你启用了 cephx 认证，需要分别为 Nova/Cinder 和 Glance 创建新用户。在 Ceph 0.53 或更早版本上执行如下：

```
ceph auth get-or-create client.volumes mon 'allow r' osd 'allow x, allow rwx pool=volumes, allow rx pool=images'  
ceph auth get-or-create client.images mon 'allow r' osd 'allow x, allow rwx pool=images'
```

In Ceph version 0.54, more specific permissions were added, so the users can be restricted further. For Ceph version 0.54 or later, execute the following:

在 Ceph 0.54 上，有了更细致的权限划分，所以能更好地限制用户。可执行此命令：

```
ceph auth get-or-create client.volumes mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=volumes, allow rx pool=images'  
ceph auth get-or-create client.images mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=images'
```

Add the keyrings for `client.volumes` and `client.images` to the appropriate hosts and change their ownership:

分别部署 `client.volumes` 和 `client.images` 的密钥环、并设置合适的所有者：

```
ceph auth get-or-create client.images | ssh {your-glance-api-server} sudo tee /etc/ceph/ceph.client.images.keyring  
ssh {your-glance-api-server} sudo chown glance:glance /etc/ceph/ceph.client.images.keyring  
ceph auth get-or-create client.volumes | ssh {your-volume-server} sudo tee /etc/ceph/ceph.client.volumes.keyring  
ssh {your-volume-server} sudo chown cinder:cinder /etc/ceph/ceph.client.volumes.keyring
```

Hosts running `nova-compute` do not need the keyring. Instead, they store the secret key in libvirt. Create a temporary copy of the secret key on the hosts running `nova-compute`:

运行 `nova-compute` 的主机不需要密钥环，相反它们把密钥环存储于 `libvirt`。在此主机上放置一个密钥的临时副本：

```
ssh {your-compute-host} client.volumes.key <`ceph auth get-key client.volumes`
```

Then, on the compute hosts, add the secret key to libvirt and remove the temporary copy of the key:

然后，在计算主机上把密钥加进 libvirt、然后删除临时副本：

```
cat > secret.xml <<EOF  
<secret ephemeral='no' private='no'>  
  <usage type='ceph'>  
    <name>client.volumes secret</name>  
  </usage>  
</secret>  
EOF  
sudo virsh secret-define --file secret.xml  
<uuid of secret is output here>  
sudo virsh secret-set-value --secret {uuid of secret} --base64 $(cat client.volumes.key) && rm client.volumes.key secret.xml
```

Save the uuid of the secret for configuring `nova-compute` later.

保留密钥的 uuid，稍后配置 nova-compute 要用。

5.7.3 让 OpenStack 使用 ceph

Configure OpenStack to use Ceph

5.7.3.1 配置Glance

Configuring Glance

Glance can use multiple back ends to store images. To use Ceph block devices by default, edit `/etc/glance/glance-api.conf` and add:

Glance 可使用多种后端存储映像，要让它默认使用 ceph 块设备，编辑 /etc/glance/glance-api.conf、添加：

```
default_store=rbd
rbd_store_user=images
rbd_store_pool=images
```

If you're using Folsom and want to enable copy-on-write cloning of images into volumes, also add:

如果你在用 Folsom，并且想在卷宗内允许映像的写时复制克隆，还得加上：

```
show_image_direct_url=True
```

Note that this exposes the back end location via Glance's API, so the endpoint with this option enabled should not be publicly accessible.

注意，这里通过 Glance 的 API 展示了后端位置，所以此选项启用时的终结点不能公开访问。

5.7.3.2 配置 Cinder/nova-volume

Configuring Cinder/nova-volume

OpenStack 要求一个驱动来与 Ceph 块设备交互。你必须指定块设备所在的存储池名字。编辑 OpenStack 主机上的 /etc/cinder/cinder.conf、对于 Folsom 及更早版本添加如下：

```
volume_driver=cinder.volume.driver.RBDDriver
rbd_pool=volumes
```

For Grizzly, use:

对于 Grizzly，加这个：

```
volume_driver=cinder.volume.drivers.rbd.RBDDriver
rbd_pool=volumes
glance_api_version=2
```

If you're not using Cinder, replace Cinder with Nova in the previous section.

如果你没用 Cinder，前述段落中要用 Nova 替换 Cinder。

If you're using [cephx authentication](#), also configure the user and uuid of the secret you added to libvirt earlier:

如果你在用 cephx 认证，还需要配置用户及其密钥（前面保存在了 libvirt 里）的 uuid：

```
rbd_user=volumes
rbd_secret_uuid={uuid of secret}
```

Finally, on each host running cinder-volume or nova-volume, add CEPH_ARGS="--id volumes" to the init/upstart script that starts it.

最后，在各运行 cinder-volume 或 nova-volume 的主机上，把 CEPH_ARGS="--id volumes" 添加到启动它的 init/upstart 脚本里。

For example, on Ubuntu, add env CEPH_ARGS="--id volumes" to the top of /etc/init/cinder-volume.conf.

例如，在 Ubuntu 上把 env CEPH_ARGS="--id volumes" 添加到 /etc/init/cinder-volume.conf 的顶部。

5.7.4 重启 OpenStack

Restart OpenStack

To activate the Ceph block device driver and load the block device pool name into the configuration, you must restart OpenStack. Navigate the directory where you installed OpenStack, and execute the following:

要激活 ceph 块设备驱动、并把块设备存储池名载入配置，必须重启 OpenStack。进入 OpenStack 安装目录并执行：

```
./rejoin-stack.sh
```

If you have OpenStack configured as a service, you can also execute these commands on the appropriate hosts:

如果你把 OpenStack 配置成了服务，也可以在对应主机上执行这些命令：

```
sudo service glance-api restart  
sudo service nova-compute restart  
sudo service cinder-volume restart
```

Once OpenStack is up and running, you should be able to create a volume with OpenStack on a Ceph block device.

一旦 OpenStack 启动并运行，你就可以用 OpenStack 在 ceph 块设备上创建卷宗了。

5.7.5 从块设备引导

Booting from a Block Device

If you're using OpenStack Folsom or later, you can create a volume from an image using the Cinder command line tool:

如果你在运行 OpenStack Folsom 或更新版本，你可以用 Cinder 命令行从一映像创建卷宗：

```
cinder create --image-id {id of image} --display-name {name of volume} {size of volume}
```

Note that image must be raw format. You can use [qemu-img](#) to convert from one format to another, i.e.:

注意映像必须是 raw 格式，你可以用 qemu-img 转换格式，如：

```
qemu-img convert -f qcow2 -O raw precise-cloudimg.img precise-cloudimg.raw
```

Before Ceph 0.52 the image will be a full copy of the data. With Ceph 0.52 and later when Glance and Cinder are both using Ceph block devices, the image is a copy-on-write clone, so volume creation is very fast.

在 ceph 0.52 之前，映像是数据的完整拷贝；但是之后的版本中 Glance 和 Cinder 都使用 ceph 块设备，映像是写时复制克隆，所以创建卷宗非常快。

In the OpenStack dashboard you can then boot from that volume by launching a new instance, choosing the image that you created the volume from, and selecting 'boot from volume' and the volume you created.

在 OpenStack 仪表板上，你可以这样从卷宗启动：启动一新例程、选择从哪个卷宗创建的映像、选择“boot from volume”和你创建的卷宗。

5.8 块设备和 CloudStack

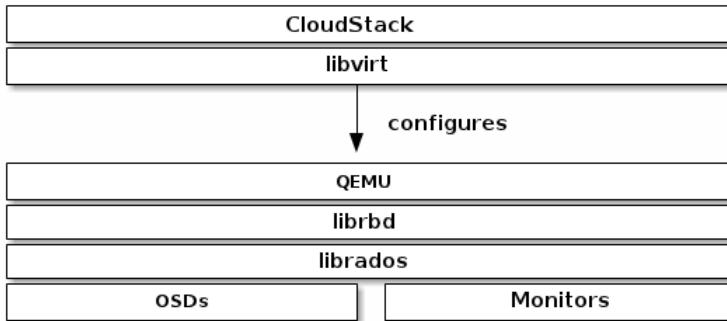
Block Devices and CloudStack

You may use Ceph Block Device images with CloudStack 4.0 and higher through `libvirt`, which configures the QEMU interface to `librbd`. Ceph stripes block device images as objects across the cluster, which means that large Ceph Block Device images have better performance than a standalone server!

CloudStack 4.0 及以上版本可以通过 libvirt 使用 ceph 块设备，libvirt 会配置 QEMU 和 librbd 交互。ceph 会把块设备映像分块为对象并分布到整个集群，这意味着大个的 ceph 块设备性能会优于单体服务器。

To use Ceph Block Devices with CloudStack 4.0 and higher, you must install QEMU, `libvirt`, and CloudStack first. We recommend using a separate physical host for your CloudStack installation. CloudStack recommends a minimum of 4GB of RAM and a dual-core processor, but more CPU and RAM will perform better. The following diagram depicts the CloudStack/Ceph technology stack.

要让 CloudStack 4.0 及更高版使用 ceph 块设备，你得先安装 QEMU、libvirt、和 CloudStack。我们建议在另外一台物理服务器上安装 CloudStack，此软件最低需要 4GB 内存和一个双核 CPU，但是资源越多越好。下图描述了 CloudStack/ceph 技术栈。



Important: To use Ceph Block Devices with CloudStack, you must have access to a running Ceph Storage Cluster.

重要：要让 CloudStack 使用 ceph 块设备，你必须有 ceph 存储集群的访问权限。

CloudStack integrates with Ceph's block devices to provide CloudStack with a back end for CloudStack's Primary Storage. The instructions below detail the setup for CloudStack Primary Storage.

CloudStack 集成了 ceph 的块设备作为它的主要存储（Primary Storage），下列指令详述了 CloudStack 的安装。

Note: We recommend installing with Ubuntu 13.04 or later so that you can use package installation instead of having to compile libvirt from source.

注意：我们建议您安装 Ubuntu 13.04 或更高版本，这样就不用手动编译 libvirt 了。

Installing and configuring QEMU for use with CloudStack doesn't require any special handling. Ensure that you have a running Ceph Storage Cluster. Install QEMU and configure it for use with Ceph; then, install libvirt version 0.9.13 or higher (you may need to compile from source) and ensure it is running with Ceph.

安装、配置 QEMU 用于 CloudStack 不需要任何特殊处理。确保你的 ceph 存储集群在运行，配置好 QEMU 即可；然后安装 libvirt 0.9.13 或更高版本（也许得手动编译）并确保它与 ceph 磨合正常。

1. [Install and Configure QEMU](#).
2. [Install and Configure libvirt](#) version 0.9.13 or higher.
3. Also see [KVM Hypervisor Host Installation](#).

Note: Raring Ringtail (13.04) will have libvirt version 0.9.13 or higher with RBD storage pool support enabled by default.

注意：Ubuntu 13.04 版搭载了 libvirt 0.9.13 或更高版本，而且默认启用了 RBD 存储池。

index:: pools; CloudStack

5.8.1 创建存储池

Create a Pool

By default, Ceph block devices use the rbd pool. Create a pool for CloudStack NFS Primary Storage. Ensure your Ceph cluster is running, then create the pool.

默认情况下，ceph 块设备使用 rbd 存储池，建议为 CloudStack NFS 主存储新建一存储池。确保 ceph 集群在运行，再创建存储池：

```
ceph osd pool create cloudstack
```

See [Create a Pool](#) for details on specifying the number of placement groups for your pools, and [Placement Groups](#) for details on the number of placement groups you should set for your pools.

参考 [Create a Pool](#) 为存储池指定归置组数量，参考 [Placement Groups](#) 确定应该为存储池分配多少归置组。

5.8.2 添加主存储

Add Primary Storage

To add primary storage, refer to [Add Primary Storage \(4.0.0\)](#) or [Add Primary Storage \(4.0.1\)](#). To add a Ceph block device, the steps include:

添加主存储方法见 [Add Primary Storage \(4.0.0\)](#) 或 [Add Primary Storage \(4.0.1\)](#)。添加一 ceph 块设备步骤如下：

1. Log in to the CloudStack UI.
 2. Click **Infrastructure** on the left side navigation bar.
 3. Select the Zone you want to use for Primary Storage.
 4. Click the **Compute** tab.
 5. Select **View All** on the *Primary Storage* node in the diagram.
 6. Click **Add Primary Storage**.
 7. Follow the CloudStack instructions.
 - For **Protocol**, select RBD.
 - Add cluster information (cephx is supported).
 - Add **rbd** as a tag.
-
1. 登入 CloudStack 界面；
 2. 点击左侧导航条的 Infrastructure；
 3. 选择要用于主存储的域；
 4. 点击 Compute 标签；
 5. 选择图表中 Primary Storage 节点上的 View All
 6. 点击 Add Primary Storage；
 7. 依次按提示执行：
 - Protocol 那里选择 RBD；
 - 添加集群信息（支持 cephx）；
 - 把 rbd 加为标签。

5.8.3 创建存储服务

Create a Disk Offering

To create a new disk offering, refer to [Create a New Disk Offering \(4.0.0\)](#) or [Create a New Disk Offering \(4.0.1\)](#). Create a disk offering so that it matches the **rbd** tag. The **StoragePoolAllocator** will choose the **rbd** pool when searching for a suitable storage pool. If the disk offering doesn't match the **rbd** tag, the **StoragePoolAllocator** may select the pool you created (e.g., `cloudstack`).

要新建硬盘存储服务，参考 [Create a New Disk Offering \(4.0.0\)](#) 或 [Create a New Disk Offering \(4.0.1\)](#)。创建一存储服务以与 **rbd** 相配，这样 **StoragePoolAllocator** 查找合适存储池时就会选择 **rbd** 存储池；如果存储服务没有与 **rbd** 标签相配，**StoragePoolAllocator** 就会选用你创建的存储池（即 `cloudstack`）。

5.8.4 限制

Limitations

- CloudStack will only bind to one monitor (You can however create a Round Robin DNS record over multiple monitors)
CloudStack 只能绑定一个监视器（但你可以创建一个轮询域名来滚动多个监视器）
- CloudStack does not support cloning snapshots.
CloudStack 不支持克隆快照
- You still need a (small) NFS based Primary Storage for the SystemVMs
你仍需要一个较小、基于 NFS 的主存储用于虚拟机系统本身

- You may need to compile **libvirt** to use version 0.9.13 with Ubuntu.

在 Ubuntu 下你也许得手动编译 libvirt 0.9.13。

5.9 rbd 在线手册

rbd – manage rados block device (RBD) images

rbd——管理 RADOS 块设备映像

5.9.1 提纲

Synopsis

rbd [-c **ceph.conf**] [-m **monaddr**] [-p | --pool **pool**] [--size **size**] [--order **bits**] [**command** ...]

5.9.2 描述

Description

rbd is a utility for manipulating rados block device (RBD) images, used by the Linux rbd driver and the rbd storage driver for Qemu/KVM. RBD images are simple block devices that are striped over objects and stored in a RADOS object store. The size of the objects the image is striped over must be a power of two.

rbd 是个修改 RBD 映像的工具，QEMU/KVM 通过 Linux 内核驱动和 **rbd** 存储驱动使用 RBD。RBD 块设备是很简单的映像，它被分块后存储于 RADOS 对象存储集群，对象的尺寸必须是 2 的幂。

5.9.3 选项

Options

-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.

指定 **ceph.conf** 配置文件，而不是用默认的 /etc/ceph/ceph.conf 来确定启动时需要的监视器。

-m monaddress[:port]

Connect to specified monitor (instead of looking through **ceph.conf**).

连接到指定监视器，无需通过 **ceph.conf** 查找。

-p pool, --pool pool

Interact with the given pool. Required by most commands.

在指定存储池下操作，大多数命令都得指定。

--no-progress

Do not output progress information (goes to standard error by default for some commands).

不显示进度（有些命令会默认输出到标准输出）。

5.9.4 参数

Parameters

--image-format format

Specifies which object layout to use. The default is 1.

选择用哪个对象布局，默认为 1。

- format 1 - Use the original format for a new rbd image. This format is understood by all versions of librbd and the kernel rbd module, but does not support newer features like cloning.

format 1——新建 **rbd** 映像时使用最初的格式。此格式兼容所有版本的 **librbd** 和内核模块，但是不支持较新的功能，像克隆。

- format 2 - Use the second rbd format, which is supported by librbd (but not the kernel rbd

module) at this time. This adds support for cloning and is more easily extensible to allow more features in the future.

format 2——使用第二版 rbd 格式， librbd 已支持但内核模块尚未。此格式增加了克隆支持，使得扩展更容易，还允许以后增加新功能。

--size size-in-mb

Specifies the size (in megabytes) of the new rbd image.

指定新 rbd 映像的尺寸， MB。

--order bits

Specifies the object size expressed as a number of bits, such that the object size is $1 \ll \text{order}$. The default is 22 (4 MB).

指定对象尺寸。用位数表示，即对象大小为 $1 \ll \text{order}$ ，默认为 22 (4MB)。

--stripe-unit size-in-bytes

Specifies the stripe unit size in bytes. See striping section (below) for more details.

指定条带单元尺寸，字节数。详情见下面的条带化一段。

--stripe-count num

Specifies the number of objects to stripe over before looping back to the first object. See striping section (below) for more details.

条带化要至少跨越多少对象才能转回第一个。详情见条带化一节。

--snap snap

Specifies the snapshot name for the specific operation.

某些操作需要指定快照名。

--id username

Specifies the username (without the `client.` prefix) to use with the map command.

指定 map 命令要用到的用户名（不含 client. 前缀）。

--keyfile filename

Specifies a file containing the secret to use with the map command. If not specified, `client.admin` will be used by default.

为 map 命令指定一个包含密钥的文件。如果没指定， 默认使用 client.admin。

--keyring filename

Specifies a keyring file containing a secret for the specified user to use with the map command. If not specified, the default keyring locations will be searched.

因 map 命令所需，指定一个用户及其密钥文件。如果未指定，从默认密钥环里找。

--shared tag

Option for `lock add` that allows multiple clients to lock the same image if they use the same tag. The tag is an arbitrary string. This is useful for situations where an image must be open from more than one client at once, like during live migration of a virtual machine, or for use underneath a clustered filesystem.

`lock add` 命令的选项，它允许使用同一标签的多个客户端同时锁住同一映像。标签是任意字符串。当某映像必须从多个客户端同时打开时，此选项很有用，像迁移活动虚拟机时、或者在集群文件系统下使用时。

--format format

Specifies output formatting (default: plain, json, xml)

指定输出格式， 默认： plain、 json、 xml。

--pretty-format

Make json or xml formatted output more human-readable.

使 json 或 xml 格式的输出更易读。

5.9.5 命令

Commands

ls [-l | -long] [pool-name]

Will list all rbd images listed in the rbd_directory object. With -l, also show snapshots, and use longer-format output including size, parent (if clone), format, etc.

列出 rbd_directory 对象中的所有 rbd 映像。加-l 选项后也显示快照，并用长格式输出，包括大小、父映像（若是克隆品）、格式等等。

info [image-name]

Will dump information (such as size and order) about a specific rbd image. If image is a clone, information about its parent is also displayed. If a snapshot is specified, whether it is protected is shown as well.

显示指定 rbd 映像的信息（如大小和顺序）。若映像是克隆品，会显示相关父快照；若指定了快照，会显示是否被保护。

create [image-name]

Will create a new rbd image. You must also specify the size via --size. The --stripe-unit and --stripe-count arguments are optional, but must be used together.

如要新建 rbd 映像，必须用--size 指定尺寸。--strip-unit 和--strip-count 参数是可选项，但必须一起用。

clone [parent-snapname] [image-name]

Will create a clone (copy-on-write child) of the parent snapshot. Object order will be identical to that of the parent image unless specified. Size will be the same as the parent snapshot.

The parent snapshot must be protected (see *rbd snap protect*). This requires format 2.

创建一个父快照的克隆品（写时复制子映像）。对象顺序将与父映像完全一样，除非另外指定过。尺寸和父快照一样。

flatten [image-name]

If image is a clone, copy all shared blocks from the parent snapshot and make the child independent of the parent, severing the link between parent snap and child. The parent snapshot can be unprotected and deleted if it has no further dependent clones.

This requires format 2.

如果映像是个克隆品，从父快照拷贝所有共享块，并使子快照独立于父快照、切断父子快照间的链接。如果没有克隆品引用此父快照了，就可以取消保护并删除。

只适用于 format 2。

children [image-name]

List the clones of the image at the given snapshot. This checks every pool, and outputs the resulting poolname/imagename.

This requires format 2.

列出此映像指定快照的克隆品。它会检查各存储池、并输出存储池名/映像名。

只适用于 format 2。

resize [image-name] [-allow-shrink]

Resizes rbd image. The size parameter also needs to be specified. The --allow-shrink option lets the size be reduced.

rbd 大小调整。尺寸参数必须指定；--allow-shrink 选项允许缩小。

rm [image-name]

Deletes an rbd image (including all data blocks). If the image has snapshots, this fails and nothing is deleted.

删除一 rbd 映像，包括所有数据块。如果映像有快照，此命令会失效。

export [image-name] [dest-path]

Exports image to dest path (use - for stdout).

把映像导出到目的路径，用-（短线）输出到标准输出。

import [path] [dest-image]

Creates a new image and imports its data from path (use - for stdin). The import operation will try to create sparse rbd images if possible. For import from stdin, the sparsification unit is the data block size of the destination image ($1 << \text{order}$).

创建一映像，并从目的路径导入数据，用-（短线）从标准输入导入。如果可能的话，导入操作会试着创建稀疏映像。如果从标准输入导入，稀疏化单位将是目标映像的数据块尺寸（即 $1 << \text{order}$ ）。

export-diff [image-name] [dest-path] [-from-snap snapname]

Exports an incremental diff for an image to dest path (use - for stdout). If an initial snapshot is specified, only changes since that snapshot are included; otherwise, any regions of the image that contain data are included. The end snapshot is specified using the standard --snap option or @snap syntax (see below). The image diff format includes metadata about image size changes, and the start and end snapshots. It efficiently represents discarded or ‘zero’ regions of the image.

导出一映像的增量差异，用-导出到标准输出。若给了起始快照，就只包含与此快照的差异部分；否则包含映像的所有数据部分；结束快照用--snap 选项或@snap（见下文）指定。此映像的差异格式包含了映像尺寸变更的元数据、起始和结束快照，它高效地表达了被忽略或映像内的全 0 区域。

import-diff [src-path] [image-name]

Imports an incremental diff of an image and applies it to the current image. If the diff was generated relative to a start snapshot, we verify that snapshot already exists before continuing. If there was an end snapshot we verify it does not already exist before applying the changes, and create the snapshot when we are done.

导入一映像的增量差异并应用到当前映像。如果此差异是在起始快照基础上生成的，我们会先校验那个已存在快照再继续；如果指定了结束快照，我们先检查它是否存在、再应用变更，结束后再创建结束快照。

diff [image-name] [-from-snap snapname]

Dump a list of byte extents in the image that have changed since the specified start snapshot, or since the image was created. Each output line includes the starting offset (in bytes), the length of the region (in bytes), and either ‘zero’ or ‘data’ to indicate whether the region is known to be zeros or may contain other data.

打印出从指定快照点起、或从映像创建点起，映像内的变动区域。输出的各行都包含起始偏移量（按字节）、数据块长度（按字节）、还有 zero 或 data，用来指示此范围以前是 0 还是其它数据。

cp [src-image] [dest-image]

Copies the content of a src-image into the newly created dest-image. dest-image will have the same size, order, and format as src-image.

把源映像内容复制进新建的目标映像，目标映像和源映像将有相同的尺寸、顺序和格式。

mv [src-image] [dest-image]

Renames an image. Note: rename across pools is not supported.

映像改名。注意：源、目的映像不能跨存储池。

snap ls [image-name]

Dumps the list of snapshots inside a specific image.

列出一映像内的快照。

snap create [image-name]

Creates a new snapshot. Requires the snapshot name parameter specified.

新建一快照。需指定快照名。

snap rollback [image-name]

Rollback image content to snapshot. This will iterate through the entire blocks array and update the data head content to the snapshot version.

把指定映像回滚到快照。此动作会递归整个块阵列，并把数据头内容更新到快照版本。

snap rm [image-name]

Removes the specified snapshot.

删除指定快照。

snap purge [image-name]

Removes all snapshots from an image.

删除一映像的所有快照。

snap protect [image-name]

Protect a snapshot from deletion, so that clones can be made of it (see `rbd clone`). Snapshots must be protected before clones are made; protection implies that there exist dependent cloned children that refer to this snapshot. `rbd clone` will fail on a nonprotected snapshot.

This requires format 2.

保护快照，防删除，这样才能从它克隆（见 `rbd clone`）。做克隆前必须先保护快照，保护意味着克隆出的子快照依赖于此快照。`rbd clone` 不能在未保护的快照上操作。

只适用于 format 2。

snap unprotect [image-name]

Unprotect a snapshot from deletion (undo `snap protect`). If cloned children remain, `snap unprotect` fails. (Note that clones may exist in different pools than the parent snapshot.)

This requires format 2.

取消对快照的保护（撤销 `snap protect`）。如果还有克隆出的子快照尚在，此命令会失效。（注意克隆品可能位于不同于父快照的存储池。）

只适用于 format 2。

map [image-name]

Maps the specified image to a block device via the rbd kernel module.

通过内核 rbd 模块把指定映像映射到某一块设备。

unmap [device-path]

Unmaps the block device that was mapped via the rbd kernel module.

取消通过内核 rbd 模块的映射。

showmapped

Show the rbd images that are mapped via the rbd kernel module.

显示通过内核 rbd 模块映射过的 rbd 映像。

lock list [image-name]

Show locks held on the image. The first column is the locker to use with the `lock remove` command.

显示锁着映像的锁，第一列是 `lock remove` 可以使用的锁名。

lock add [image-name] [lock-id]

Lock an image. The lock-id is an arbitrary name for the user's convenience. By default, this is an exclusive lock, meaning it will fail if the image is already locked. The `--shared` option changes this behavior. Note that locking does not affect any operation other than adding a lock. It does not protect an image from being deleted.

为映像加锁，锁标识是用户一己所好的任意名字。默认加的是互斥锁，也就是说如果已经加过锁的话此命令会失败；`--shared` 选项会改变此行为。注意，加锁操作本身不影响除加锁之外的任何操作，也不会保护对象、防止它被删除。

lock remove [image-name] [lock-id] [locker]

Release a lock on an image. The lock id and locker are as output by `lock ls`.

释放映像上的锁。锁标识和其持有者来自 `lock ls`。

bench-write [image-name] [-io-size [io-size-in-bytes]] [-io-threads [num-ios-in-flight]] [-io-total [total-bytes-to-write]]

Generate a series of sequential writes to the image and measure the write throughput and latency. Defaults are: `-io-size` 4096, `-io-threads` 16, `-io-total` 1GB

生成一系列顺序写来衡量写吞吐量和延时。默认参数为 `-io-size` 4096、`-io-threads` 16、`-io-total` 1GB。

5.9.6 映像名

Image name

In addition to using the `--pool` and the `--snap` options, the image name can include both the pool name and the snapshot name. The image name format is as follows:

除了`--pool`和`--snap`选项之外，映像名还能包含存储池名和快照名，其格式如下：

```
[pool/]image-name[@snap]
```

Thus an image name that contains a slash character ('/') requires specifying the pool name explicitly.

因此包含斜杠 (/) 的映像名显式地指定了存储池名。

5.9.7 条带化

Striping

RBD images are striped over many objects, which are then stored by the Ceph distributed object store (RADOS). As a result, read and write requests for the image are distributed across many nodes in the cluster, generally preventing any single node from becoming a bottleneck when individual images get large or busy.

RBD 映像被条带化为很多对象，然后存储到 ceph 分布式对象存储 (RADOS) 集群中。因此，到此映像的读和写请求会被分布到集群内的很多节点，也因此避免了映像巨大或繁忙时可能出现的单节点瓶颈。

The striping is controlled by three parameters:

条带化由 3 个参数控制：

order

The size of objects we stripe over is a power of two, specifically $2^{[*order*]}$ bytes. The default is 22, or 4 MB.

stripe_unit

Each $[*stripe_unit*]$ contiguous bytes are stored adjacently in the same object, before we move on to the next object.

stripe_count

After we write $[*stripe_unit*]$ bytes to $[*stripe_count*]$ objects, we loop back to the initial object and write another stripe, until the object reaches its maximum size (as specified by $[*order*]$). At that point, we move on to the next $[*stripe_count*]$ objects.

order

条带化产生的对象尺寸是 2 的幂，即 $2^{[*order*]}$ 字节。默认为 22，或 4MB。

stripe_unit

各条带单位是连续的字节，相邻地存储于同一对象，用满再去下一个对象。

stripe_count

我们把 `stripe_unit` 个字节写够 `stripe_count` 个对象后，再转回到第一个对象写另一轮条带，直到达到对象的最大尺寸（由 `order` 影响）。此时，我们再用下一轮 `stripe_count` 个对象。

By default, `[stripe_unit]` is the same as the object size and `[stripe_count]` is 1. Specifying a different `[stripe_unit]` requires that the STRIPINGV2 feature be supported (added in Ceph v0.53) and format 2 images be used.

默认情况下，`[stripe_unit]` 和对象尺寸相同、且`[stripe_count]` 为 1；另外指定`[stripe_unit]` 需 STRIPINGV2 功能 (ceph 0.53 起加入) 并使用 format 2 格式的映像。

5.9.8 实例

Examples

To create a new rbd image that is 100 GB:

要新建一 100GB 的 rbd 映像：

```
rbd -p mypool create myimage --size 102400
```

or alternatively:

或者这样：

```
rbd create mypool/myimage --size 102400
```

To use a non-default object size (8 MB):

用个非默认对象尺寸，8MB：

```
rbd create mypool/myimage --size 102400 --order 23
```

To delete an rbd image (be careful!):

删除一rbd映像（小心！）：

```
rbd rm mypool/myimage
```

To create a new snapshot:

新建快照：

```
rbd snap create mypool/myimage@mysnap
```

To create a copy-on-write clone of a protected snapshot:

创建已保护快照的写时复制克隆：

```
rbd clone mypool/myimage@mysnap otherpool/cloneimage
```

To see which clones of a snapshot exist:

查看快照有哪些克隆品：

```
rbd children mypool/myimage@mysnap
```

To delete a snapshot:

删除快照：

```
rbd snap rm mypool/myimage@mysnap
```

To map an image via the kernel with cephx enabled:

启用cephx时通过内核映射一映像：

```
rbd map mypool/myimage --id admin --keyfile secretfile
```

To unmap an image:

取消映像映射：

```
rbd unmap /dev/rbd0
```

To create an image and a clone from it:

创建一映像及其克隆品：

```
rbd import --format 2 image mypool/parent
rbd snap create --snap snapname mypool/parent
rbd snap protect mypool/parent@snap
rbd clone mypool/parent@snap otherpool/child
```

To create an image with a smaller stripe_unit (to better distribute small writes in some workloads):

新建一stripe_unit较小的映像（在某些情况下可更好地分布少量写）：

```
rbd -p mypool create myimage --size 102400 --stripe-unit 65536 --stripe-count 16
```

To change an image from one format to another, export it and then import it as the desired format:

更改一映像的格式，先导出、再导入为期望格式：

```
rbd export mypool/myimage@snap /tmp/img
rbd import --format 2 /tmp/img mypool/myimage2
```

To lock an image for exclusive use:

互斥地锁住一映像：

```
rbd lock add mypool/myimage mylockid
```

To release a lock:

释放锁：

```
rbd lock remove mypool/myimage mylockid client.2485
```

5.9.9 使用范围

Availability

rbd is part of the Ceph distributed file system. Please refer to the Ceph documentation at <http://ceph.com/docs> for more information.

rbd 是 ceph 分布式文件系统的一部分，更多信息参见 <http://ceph.com/docs>。

5.9.10 参考

See also

[ceph\(8\)](#), [rados\(8\)](#)

5.10 rbd-fuse 在线手册

rbd-fuse – expose rbd images as files

rbd-fuse——把 rbd 映像展现为文件

5.10.1 提纲

Synopsis

rbd-fuse [-p pool] [-c conffile] mountpoint [fuse options]

5.10.2 描述

Description

rbd-fuse is a FUSE (File system in USErspace) client for RADOS block device (rbd) images. Given a pool containing rbd images, it will mount a userspace filesystem allowing access to those images as regular files at mountpoint.

rbd-fuse 是个 rbd 映像的用户空间文件系统 (FUSE) 客户端。给一个包含 rbd 映像的存储池，它就可以在用户空间把那些映像挂载到 mountpoint 下，并显示为普通文件。

The file system can be unmounted with:

用下列命令卸载：

```
fusermount -u mountpoint
```

or by sending SIGINT to the rbd-fuse process.

或者向 rbd-fuse 进程发送 SIGINT 信号。

5.10.3 选项

Options

Any options not recognized by rbd-fuse will be passed on to libfuse.

rbd-fuse 不认识的选项将传递给 libfuse。

-c ceph.conf

Use **ceph.conf** configuration file instead of the default /etc/ceph/ceph.conf to determine monitor addresses during startup.

用指定的 ceph.conf 配置文件、而非默认的 /etc/ceph/ceph.conf 来确定监视器地址，启动时要用。

-p pool

Use **pool** as the pool to search for rbd images. Default is rbd.

在 pool 存储池中搜索 rbd 映像，默认为 rbd。

5.10.4 使用范围

Availability

rbd-fuse is part of the Ceph distributed file system. Please refer to the Ceph documentation at <http://ceph.com/docs> for more information.

rbd-fuse 是 ceph 分布式文件系统的一部分，更多信息参见 <http://ceph.com/docs>。

5.10.5 参考

See also

fusermount(8), [**rbd\(8\)**](#)

5.11 ceph-rbdnamer 在线手册

ceph-rbdnamer – udev helper to name RBD devices

ceph-rbdnamer——用于命名 RBD 设备的 udev 辅助程序

5.11.1 提纲

Synopsis

ceph-rbdnamer num

5.11.2 描述

Description

ceph-rbdnamer prints the pool and image name for the given RBD devices to stdout. It is used by *udev* (using a rule like the one below) to set up a device symlink.

ceph-rbdnamer 把指定 RBD 设备所属的存储池和映像名打印到标准输出，以便 *udev*（使用类似如下的规则）设置设备的符号链接：

```
KERNEL=="rbd[0-9]*", PROGRAM="/usr/bin/ceph-rbdnamer %n", SYMLINK+="rbd/%c{1}/%c{2}"
```

5.11.3 使用范围

Availability

ceph-rbdnamer is part of the Ceph distributed file system. Please refer to the Ceph documentation at <http://ceph.com/docs> for more information.

ceph-rbdnamer 是 ceph 分布式文件系统的一部分，更多信息参见 <http://ceph.com/docs>。

5.11.4 参考

See also

[**rbd\(8\)**](#), [**ceph\(8\)**](#)

5.12 Librbd (Python)

The *rbd* python module provides file-like access to RBD images.

rbd 的 python 模块为 RBD 映像提供了类似文件的访问方法。

5.12.1 实例：创建并写入映像

Example: Creating and writing to an image

To use *rbd*, you must first connect to RADOS and open an IO context:

要使用 *rbd*，必须先连接 RADOS 并打开 IO 上下文：

```
cluster = rados.Rados(conffile='my_ceph.conf')
cluster.connect()
ioctx = cluster.open_ioctx('mypool')
```

Then you instantiate an :class:rbd.RBD object, which you use to create the image:

然后实例化: class:rbd.RBD 对象，用它来创建映像：

```
rbd_inst = rbd.RBD()
size = 4 * 1024**3 # 4 GiB
rbd_inst.create(ioctx, 'myimage', size)
```

To perform I/O on the image, you instantiate an :class:rbd.Image object:

要在映像上进行 I/O 操作，需实例化: class:rbd.Image 对象：

```
image = rbd.Image(ioctx, 'myimage')
data = 'foo' * 200
image.write(data, 0)
```

This writes 'foo' to the first 600 bytes of the image. Note that data cannot be :type:unicode - Librbd does not know how to deal with characters wider than a :c:type:char.

In the end, you'll want to close the image, the IO context and the connection to RADOS:

上面的代码向映像前面写入了 600 字节。注意数据不能是: type:unicode， librbd 不能如何处理大于:c:type:char 的字符串。

最后，关闭映像、IO 上下文、和到 RADOS 的连接。

```
image.close()
ioctx.close()
cluster.shutdown()
```

To be safe, each of these calls would need to be in a separate :finally block:

安全起见，每个调用都应该封装到单独的: finally 块内。

```
cluster = rados.Rados(conffile='my_ceph_conf')
try:
    ioctx = cluster.open_ioctx('my_pool')
    try:
        rbd_inst = rbd.RBD()
        size = 4 * 1024**3 # 4 GiB
        rbd_inst.create(ioctx, 'myimage', size)
        image = rbd.Image(ioctx, 'myimage')
        try:
            data = 'foo' * 200
            image.write(data, 0)
        finally:
            image.close()
    finally:
        ioctx.close()
finally:
    cluster.shutdown()
```

This can be cumbersome, so the Rados, Ioctx, and Image classes can be used as context managers that close/shutdown automatically (see [PEP 343](#)). Using them as context managers, the above example becomes:

这样做有些繁琐，所以 Rados、Ioctx、Image 类可以当上下文管理器来用，它能自动关闭（见 PEP 343）。当上下文管理器用时，上面的实例可以写成：

```
with rados.Rados(conffile='my_ceph.conf') as cluster:
    with cluster.open_ioctx('mypool') as ioctx:
        rbd_inst = rbd.RBD()
        size = 4 * 1024**3 # 4 GiB
        rbd_inst.create(ioctx, 'myimage', size)
        with rbd.Image(ioctx, 'myimage') as image:
            data = 'foo' * 200
            image.write(data, 0)
```

5.12.2 API 参考

API Reference

This module is a thin wrapper around librbd.

此模块是 librbd 的瘦包装。

It currently provides all the synchronous methods of librbd that do not use callbacks.

当前它提供了 librbd 的所有未使用回调的同步方法。

Error codes from librbd are turned into exceptions that subclass `Error`. Almost all methods may raise `Error` (the base class of all rbd exceptions), `PermissionError` and `IOError`, in addition to those documented for the method.

librbd 中的错误代码转换成了 `Error` 子类中的异常。几乎所有方法都能产生 `Error` (所有 rbd 异常的基础类)、`PermissionError` 和 `IOError` 等等。

A number of methods have string arguments, which must not be unicode to interact correctly with librbd. If unicode is passed to these methods, a `TypeError` will be raised.

很多方法都有字符串参数，它们都不能是 unicode，否则不能正确地和 librbd 交互。如果向这些方法传递 unicode，将抛出 `TypeError`。

`class rbd.RBD`

This class wraps librbd CRUD functions.

`clone(p_ioctx, p_name, p_snapname, c_ioctx, c_name, features=0, order=None)`

Clone a parent rbd snapshot into a COW sparse child.

Parameters:

- `p_ioctx` – the parent context that represents the parent snap
- `p_name` – the parent image name
- `p_snapname` – the parent image snapshot name
- `c_ioctx` – the child context that represents the new clone
- `c_name` – the clone (child) name
- `features (int)` – bitmask of features to enable; if set, must include layering
- `order (int)` – the image is split into ($2^{**\text{order}}$) byte objects

Raises : `TypeError`

Raises : `InvalidArgumentException`

Raises : `ImageExists`

Raises : `FunctionNotSupported`

Raises : `ArgumentOutOfRangeException`

`create(ioctx, name, size, order=None, old_format=True, features=0, stripe_unit=0, stripe_count=0)`

Create an rbd image.

Parameters:

- `ioctx (rados.Ioctx)` – the context in which to create the image
- `name (str)` – what the image is called
- `size (int)` – how big the image is in bytes
- `order (int)` – the image is split into ($2^{**\text{order}}$) byte objects
- `old_format (bool)` – whether to create an old-style image that is accessible by old clients, but can't use more advanced features like layering.
- `features (int)` – bitmask of features to enable
- `stripe_unit (int)` – stripe unit in bytes (default 0 for object size)
- `stripe_count (int)` – objects to stripe over before looping

Raises : `ImageExists`

Raises : TypeError

Raises : InvalidArgument

Raises : FunctionNotSupported

list(ioctx)

List image names.

Parameters: ioctx (rados.Ioctx) – determines which RADOS pool is read

Returns: list – a list of image names

remove(ioctx, name)

Delete an RBD image. This may take a long time, since it does not return until every object that comprises the image has been deleted. Note that all snapshots must be deleted before the image can be removed. If there are snapshots left, ImageHasSnapshots is raised. If the image is still open, or the watch from a crashed client has not expired, ImageBusy is raised.

Parameters:

- ioctx (rados.Ioctx) – determines which RADOS pool the image is in
- name (str) – the name of the image to remove

Raises : ImageNotFound, ImageBusy, ImageHasSnapshots

rename(ioctx, src, dest)

Rename an RBD image.

Parameters:

- ioctx (rados.Ioctx) – determines which RADOS pool the image is in
- src (str) – the current name of the image
- dest (str) – the new name of the image

Raises : ImageNotFound, ImageExists

version()

Get the version number of the librbd C library.

Returns: a tuple of (major, minor, extra) components of the librbd version

class rbd.Image(ioctx, name, snapshot=None, read_only=False)

This class represents an RBD image. It is used to perform I/O on the image and interact with snapshots.

Note: Any method of this class may raise ImageNotFound if the image has been deleted.

break_lock(client, cookie)

Release a lock held by another rados client.

close()

Release the resources used by this image object.

After this is called, this object should not be used.

`copy(dest_ioctx, dest_name)`

Copy the image to another location.

- Parameters:**
- `dest_ioctx` (`rados.Ioctx`) – determines which pool to copy into
 - `dest_name` (`str`) – the name of the copy

Raises : `ImageExists`

`create_snap(name)`

Create a snapshot of the image.

- Parameters:** `name` (`str`) – the name of the snapshot

Raises : `ImageExists`

`diff_iterate(offset, length, from_snapshot, iterate_cb)`

Iterate over the changed extents of an image.

This will call `iterate_cb` with three arguments:

`(offset, length, exists)`

where the changed extent starts at `offset` bytes, continues for `length` bytes, and is full of data (if `exists` is `True`) or zeroes (if `exists` is `False`).

If `from_snapshot` is `None`, it is interpreted as the beginning of time and this generates all allocated extents.

The end version is whatever is currently selected (via `set_snap`) for the image.

Raises `InvalidArgumentException` if `from_snapshot` is after the currently set snapshot.

Raises `ImageNotFound` if `from_snapshot` is not the name of a snapshot of the image.

- Parameters:**
- `offset` (`int`) – start offset in bytes
 - `length` (`int`) – size of region to report on, in bytes
 - `from_snapshot` (`str or None`) – starting snapshot name, or `None`
 - `iterate_cb` (`function acceptance arguments for offset, length, and exists`) – function to call for each extent

Raises : `InvalidArgumentException`, `IOError`, `ImageNotFound`

`discard(offset, length)`

Trim the range from the image. It will be logically filled with zeroes.

`flatten()`

Flatten clone image (copy all blocks from parent to child)

`flush()`

Block until all writes are fully flushed if caching is enabled.

`is_protected_snap(name)`

Find out whether a snapshot is protected from deletion.

Parameters: name (str) – the snapshot to check

Returns: bool - whether the snapshot is protected

Raises : IOError, ImageNotFound

list_children()

List children of the currently set snapshot (set via set_snap()).

Returns: list - a list of (pool name, image name) tuples

list_lockers()

List clients that have locked the image and information about the lock.

Returns: dict - contains the following keys:

- **tag** - the tag associated with the lock (every additional locker must use the same tag)
- **exclusive** - boolean indicating whether the lock is exclusive or shared
- **lockers** - a list of (client, cookie, address) tuples

list_snaps()

Iterate over the snapshots of an image.

Returns: [SnapIterator](#)

lock_exclusive(cookie)

Take an exclusive lock on the image.

Raises : ImageBusy if a different client or cookie locked it ImageExists if the same client and cookie locked it

lock_shared(cookie, tag)

Take a shared lock on the image. The tag must match that of the existing lockers, if any.

Raises : ImageBusy if a different client or cookie locked it ImageExists if the same client and cookie locked it

protect_snap(name)

Mark a snapshot as protected. This means it can't be deleted until it is unprotected.

Parameters: name (str) – the snapshot to protect

Raises : IOError, ImageNotFound

read(offset, length)

Read data from the image. Raises InvalidArgument if part of the range specified is outside the image.

Parameters: • offset (int) – the offset to start reading at

• length (int) – how many bytes to read

Returns: str - the data read

Raises : InvalidArgument, IOError

remove_snap(name)

Delete a snapshot of the image.

Parameters: name (str) – the name of the snapshot

Raises : IOError, ImageBusy

resize(size)

Change the size of the image.

Parameters: size (int) – the new size of the image

rollback_to_snap(name)

Revert the image to its contents at a snapshot. This is a potentially expensive operation, since it rolls back each object individually.

Parameters: name (str) – the snapshot to rollback to

Raises : IOError

set_snap(name)

Set the snapshot to read from. Writes will raise ReadOnlyImage while a snapshot is set. Pass None to unset the snapshot (reads come from the current image), and allow writing again.

Parameters: name (str or None) – the snapshot to read from, or None to unset the snapshot
size()

Get the size of the image. If open to a snapshot, returns the size of that snapshot.

Returns: the size of the image in bytes

stat()

Get information about the image. Currently parent pool and parent name are always -1 and ".

Returns: dict - contains the following keys:

- size (int) - the size of the image in bytes
- obj_size (int) - the size of each object that comprises the image
- num_objs (int) - the number of objects in the image
- order (int) - $\log_2(\text{object_size})$
- block_name_prefix (str) - the prefix of the RADOS objects used to store the image
- parent_pool (int) - deprecated
- parent_name (str) - deprecated

See also `format()` and `features()`.

stripe_count()

Returns the stripe count used for the image.

stripe_unit()

Returns the stripe unit used for the image.

unlock(cookie)

Release a lock on the image that was locked by this rados client.

unprotect_snap(name)

Mark a snapshot unprotected. This allows it to be deleted if it was protected.

Parameters: name (str) – the snapshot to unprotect

```
Raises : IOError, ImageNotFound  
write(data, offset)
```

Write data to the image. Raises `InvalidArgumentException` if part of the write would fall outside the image.

Parameters:

- `data (str)` – the data to be written
- `offset (int)` – where to start writing data

Returns: int - the number of bytes written

Raises : IncompleteWriteError, LogicError, InvalidArgumentException, IOError

```
class rbd.SnapIterator(image)  
Iterator over snapshot info for an image.
```

Yields a dictionary containing information about a snapshot.

Keys are:

- `id (int)` - numeric identifier of the snapshot
- `size (int)` - size of the image at the time of snapshot (in bytes)
- `name (str)` - name of the snapshot

6 ceph 对象网关

Ceph Object Gateway

[Ceph Object Gateway](#) is an object storage interface built on top of `librgw` and `librados` to provide applications with a RESTful gateway to Ceph Storage Clusters. [Ceph Object Storage](#) supports two interfaces:

ceph 对象网关是个对象存储接口，构建于 `librgw` 和 `librados` 之上，用 RESTful 作为 ceph 存储集群的网关提供给应用程序。ceph 对象存储提供了 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 Gafeway 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.

Ceph Object Storage uses the Ceph Object Gateway daemon (`radosgw`), which is a FastCGI module for interacting with `librgw` and `librados`. Since it provides interfaces compatible with OpenStack Swift and Amazon S3, the Ceph Object Gateway has its own user management. Ceph Object Gateway can store data in the same Ceph Storage Cluster used to store data from Ceph Filesystem clients or Ceph Block Device clients. The S3 and Swift APIs share a common namespace, so you may write data with one API and retrieve it with the other.

ceph 对象存储使用 ceph 对象网关守护进程 (`radosgw`)，它是个与 `librgw` 和 `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。ceph 对象网关可与 Ceph FS 客户端或 ceph 块设备客户端共用一个存储集群。S3 和 Swift API 共用一个通用命名空间，所以你可以用一个 API 写、然后用另一个检索。



Note: Ceph Object Storage does **NOT** use the Ceph Metadata Server.

注意：ceph 对象存储不使用 Ceph 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

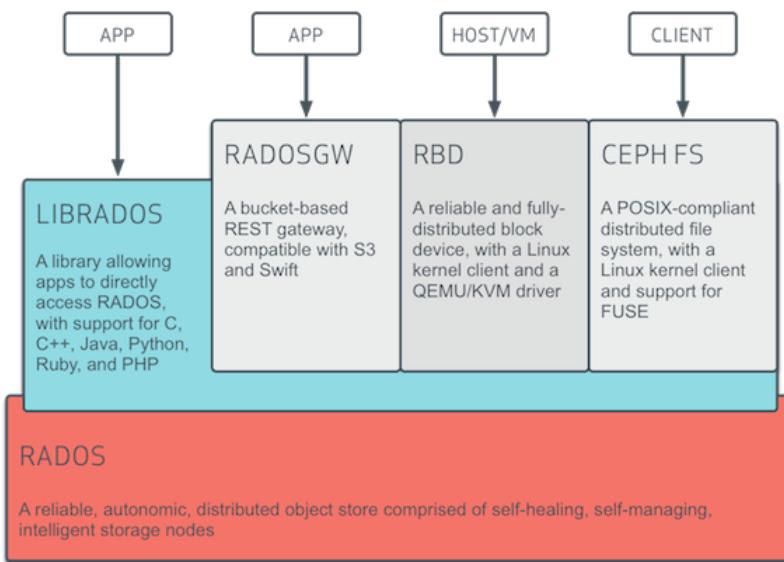
后续将补。。

8 体系结构

Architecture

Ceph uniquely delivers **object, block, and file storage** in one unified system. Ceph is highly reliable, easy to manage, and free. The power of Ceph can transform your company's IT infrastructure and your ability to manage vast amounts of data. Ceph delivers extraordinary scalability—thousands of clients accessing petabytes to exabytes of data. A **Ceph Node** leverages commodity hardware and intelligent daemons, and a **Ceph Storage Cluster** accommodates large numbers of nodes, which communicate with each other to replicate and redistribute data dynamically. A **Ceph Monitor** can also be placed into a cluster of Ceph monitors to oversee the Ceph nodes in the Ceph Storage Cluster (a monitor cluster ensures high availability).

Ceph 独一无二地用统一的系统提供了对象、块、和文件系统功能，它可靠性高、管理简便、并且是自由软件。ceph 的强大足以改变贵公司的 IT 基础架构、和管理海量数据。ceph 具有极大的伸缩性——支持数千客户端访问 EB 级数据；每个 ceph 节点都揉合了普通硬件和智能守护进程，而一个 ceph 存储集群容纳了大量节点，它们相互通信、复制数据、动态地重分布数据；ceph 监视器也能组成集群来监控整个 ceph 存储集群内的 ceph 节点，同时，监视器集群也提供了高可用性。



8.1 ceph 存储集群

The Ceph Storage Cluster

Ceph provides an infinitely scalable [Ceph Storage Cluster](#) based upon , which you can read about in [RADOS - A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters](#). Storage cluster clients and each [Ceph OSD Daemon](#) use the CRUSH algorithm to efficiently compute information about data location, instead of having to depend on a central lookup table. Ceph's high-level features include providing a native interface to the Ceph Storage Cluster via `librados`, and a number of service interfaces built on top of `librados`.

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](#)。存储集群的客户端和各个 OSD 守护进程使用 CRUSH 算法高效地计算数据位置，而不是查询某个表。它的高级功能包括：基于 `librados` 的原生存储接口、和多种基于 `librados` 的服务接口。

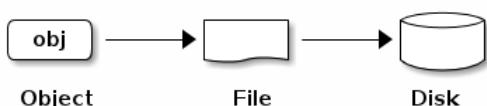


8.1.1 数据的存储

Storing Data

The Ceph Storage Cluster receives data from [Ceph Clients](#)—whether it comes through a [Ceph Block Device](#), [Ceph Object Storage](#), the [Ceph Filesystem](#) or a custom implementation you create using `librados`—and it stores the data as objects. Each object corresponds to a file in a filesystem, which is stored on an [Object Storage Device](#). Ceph OSD Daemons handle the read/write operations on the storage disks.

ceph 存储集群从客户端接收数据——不管是来自 ceph 块设备、对象存储、ceph 文件系统、还是基于 librados 的自定义实现——并存储为对象。对象是相应文件系统中的文件，但它是作为对象存储设备使用的，这些读/写操作是 OSD 守护进程处理的。



Ceph OSD Daemons 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 [Ceph Clients](#). For example, CephFS uses metadata to store file attributes such as the file owner, created date, last modified date, and so forth.

OSD 在扁平的命名空间内把所有数据存储为对象（也就是没有目录层次）。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。

ID	Binary Data	Metadata
1234	01010101010100110101010010 0101100001010100110101010010 0101100001010100110101010010	name1 value1 name2 value2 nameN valueN

Note: An object ID is unique across the entire cluster, not just the local filesystem.

注意：一个对象 ID 不止在本地唯一，它在整个集群内都是唯一的。

8.1.2 伸缩性和高可用性

Scalability and High Availability

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、前端等等），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（就是说如果中央化组件挂了，整个系统就挂了）。

Ceph eliminates the centralized gateway to enable clients to interact with Ceph OSD Daemons directly. Ceph OSD Daemons create object replicas on other Ceph Nodes to ensure data safety and high availability. Ceph also uses a cluster of monitors to ensure high availability. To eliminate centralization, Ceph uses an algorithm called CRUSH.

ceph 消除了集中网关，允许客户端直接和 OSD 守护进程通讯。OSD 守护进程自动在其它 ceph 节点上创建对象复制品来确保数据安全和高可用性；为保证高可用性，监视器也实现了集群化。为消除中央集权制，ceph 使用了 CRUSH 算法。

8.1.2.1 CRUSH 简介

CRUSH Introduction

Ceph Clients and Ceph OSD Daemons both use the algorithm to efficiently compute information about data containers on demand, instead of having to depend on a central lookup table. CRUSH provides a better data management mechanism compared to older approaches, and enables massive scale by cleanly distributing the work to all the clients and OSD daemons in the cluster. CRUSH uses intelligent data replication to ensure resiliency, which is better suited to hyper-scale storage. The following sections provide additional details on how CRUSH works. For a detailed discussion of CRUSH, see [CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data](#).

ceph 客户端和 OSD 都用此算法来按需计算与数据相关的信息，而不是查询某个集中的表。和以往方法相比，CRUSH 的数据管理机制更好，它很干脆地把某些工作丢给集群内的所有客户端和 OSD 来处理，因此具有极大的伸缩性。CRUSH 用智能数据复制确保弹性，更能适应超大规模存储。下列几段描述了 CRUSH 如何工作，更详细的机制请参阅论文：[CRUSH - Controlled, Scalable, Decentralized Placement of Replicated Data](#)。

8.1.2.2 集群运行图

Cluster Map

Ceph depends upon Ceph Clients and Ceph OSD Daemons having knowledge of the cluster topology, which is inclusive of 5 maps collectively referred to as the “Cluster Map”:

ceph 依赖于 ceph 客户端和 OSD，因为它们知道集群的拓扑，这个拓扑由 5 张图共同描述，统称为“集群运行图”：

1. **The Monitor Map:** Contains the cluster `fsid`, the position, name address and port of each monitor. It also indicates the current epoch, when the map was created, and the last time it changed. To view a monitor map, execute `ceph mon dump`.

监视器图：包含集群内各监视器的 `fsid`、位置、名字、地址和端口，也包括当前时间结、此图何时创建、最近修改时间。要查看监视器图，用 `ceph mon dump` 命令。

2. **The OSD Map:** Contains the cluster `fsid`, when the map was created and last modified, a list of pools, replica sizes, PG numbers, a list of OSDs and their status (e.g., `up`, `in`). To view an OSD map,

execute `ceph osd dump`.

OSD 图: 包含集群 `fsid`、此图何时创建、最近修改时间、存储池列表、副本数量、归置组数量、OSD 列表及其状态（如 `up`、`in`）。要查看 OSD 运行图，用 `ceph osd dump` 命令。

3. **The PG Map:** Contains the PG version, its time stamp, the last OSD map epoch, the full ratios, and details on each placement group such as the PG ID, the *Up Set*, the *Acting Set*, the state of the PG (e.g., `active + clean`), and data usage statistics for each pool.

归置组图: 包含归置组版本、其时间戳、最新的 OSD 图时间结、占满率、以及各归置组详情，像归置组 ID、up set、acting set、PG 状态（如 `active+clean`），和各存储池的数据使用情况统计。

4. **The CRUSH Map:** Contains a list of storage devices, the failure domain hierarchy (e.g., device, host, rack, row, room, etc.), and rules for traversing the hierarchy when storing data. To view a CRUSH map, execute `ceph osd getcrushmap -o {filename}`; then, decompile it by executing `crushtool -d {comp-crushmap-filename} -o {decomp-crushmap-filename}`. You can view the decompiled map in a text editor or with `cat`.

CRUSH 图: 包含存储设备列表、失败域树状结构（如设备、主机、机架、行、房间、等等）、和存储数据时如何利用此树状结构的规则。要查看 CRUSH 规则，执行 `ceph osd getcrushmap -o {filename}` 命令；然后用 `crushtool -d {comp-crushmap-filename} -o {decomp-crushmap-filename}` 反编译；然后就可以用 `cat` 或编辑器查看了。

5. **The MDS Map:** Contains the current MDS map epoch, when the map was created, and the last time it changed. It also contains the pool for storing metadata, a list of metadata servers, and which metadata servers are `up` and `in`. To view an MDS map, execute `ceph mds dump`.

MDS 图: 包含当前 MDS 图的时间结、此图创建于何时、最近修改时间，还包含了存储元数据的存储池、元数据服务器列表、还有哪些元数据服务器是 `up` 且 `in` 的。要查看 MDS 图，执行 `ceph mds dump`。

Each map maintains an iterative history of its operating state changes. Ceph Monitors maintain a master copy of the cluster map including the cluster members, state, changes, and the overall health of the Ceph Storage Cluster.

各运行图维护着各自运营状态的变更，`ceph` 监视器维护着一份集群运行图的主拷贝，包括集群成员、状态、变更、以及 `ceph` 存储集群的整体健康状况。

8.1.2.3 高可用监视器

High Availability Monitors

Before Ceph Clients can read or write data, they must contact a Ceph Monitor to obtain the most recent copy of the cluster map. A Ceph Storage Cluster can operate with a single monitor; however, this introduces a single point of failure (i.e., if the monitor goes down, Ceph Clients cannot read or write data).

`ceph` 客户端读或写数据前必须先连接到某个 `ceph` 监视器、获得最新的集群运行图副本。一个 `ceph` 存储集群只需要单个监视器就能运行，但它就成了单故障点（即如果此监视器当机，`ceph` 客户端就不能读写数据了）。

For added reliability and fault tolerance, Ceph supports a cluster of monitors. In a cluster of monitors, latency and other faults can cause one or more monitors to fall behind the current state of the cluster. For this reason, Ceph must have agreement among various monitor instances regarding the state of the cluster. Ceph always uses a majority of monitors (e.g., 1, 2:3, 3:5, 4:6, etc.) and the [Paxos](#) algorithm to establish a consensus among the monitors about the current state of the cluster.

为增强可靠性和容错能力，`ceph` 支持监视器集群；在一个监视器集群内，延时以及其它错误会导致一到多个监视器滞后于集群的当前状态，因此，`ceph` 的各监视器例程必须就集群的当前状态达成一致。为此，`ceph` 总是使用大多数监视器（如：1、2:3、3:5、4:6 等等）和 Paxos 算法就集群的当前状态达成一致。

For details on configuring monitors, see the [Monitor Config Reference](#).

关于配置监视器的详情，见 [Monitor Config Reference](#)。

8.1.2.4 高可用认证

High Availability Authentication

Ceph clients can authenticate users with Ceph Monitors, Ceph OSD Daemons and Ceph Metadata Servers, using Ceph's Kerberos-like `cephx` protocol. Authenticated users gain authorization to read, write and execute Ceph commands. The Cephx authentication system avoids a single point of failure to ensure scalability and high availability. For details on Cephx and how it differs from Kerberos, see [Ceph Authentication and Authorization](#).

通过类似 Kerberos 的 cephx 认证协议，ceph 客户端能通过 ceph 监视器、OSD 和元数据服务器认证其用户，通过认证的用户可获得读、写和执行 ceph 命令的授权。cephx 认证系统消除了单故障点，进而保证了可扩展性和高可用性。关于 cephx 的细节和与 Kerberos 的不同之处，参见[认证概览](#)。

8.1.2.5 智能程序支撑超大规模

Smart Daemons Enable Hyperscale

In many clustered architectures, the primary purpose of cluster membership is so that a centralized interface knows which nodes it can access. Then the centralized interface provides services to the client through a double dispatch—which is a **huge** bottleneck at the petabyte-to-exabyte scale.

在很多集群化体系结构中，集群成员的主要目的都相似，集中式接口知道它能访问哪些节点，然后此中央接口通过一个两级调度把服务调给客户端，在 PB 到 EB 级系统中这个调度系统必将成为瓶颈。

Ceph eliminates the bottleneck: Ceph's OSD Daemons AND Ceph Clients are cluster aware. Like Ceph clients, each Ceph OSD Daemon knows about other Ceph OSD Daemons in the cluster. This enables Ceph OSD Daemons to interact directly with other Ceph OSD Daemons and Ceph monitors. Additionally, it enables Ceph Clients to interact directly with Ceph OSD Daemons.

ceph 消除了此瓶颈：其 OSD 守护进程和客户端都能感知集群，比如 ceph 客户端、各 OSD 守护进程都知道集群内有哪些节点，这样 OSD 就能直接和其它 OSD 守护进程和监视器们通讯。另外，ceph 客户端也能直接和 OSD 守护进程交互。

The ability of Ceph Clients, Ceph Monitors and Ceph OSD Daemons to interact with each other means that Ceph OSD Daemons can utilize the CPU and RAM of the Ceph nodes to easily perform tasks that would bog down a centralized server. The ability to leverage this computing power leads to several major benefits:

ceph 客户端、监视器和 OSD 守护进程可以相互直接交互，这意味着 OSD 可以利用本地节点的 CPU 和 RAM 执行那些有可能拖垮中央服务器的任务。这种设计均衡了计算资源，带来几个好处：

1. **OSDs Service Clients Directly:** 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 Ceph Clients to contact Ceph OSD Daemons 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 Ceph OSD Daemon instead of a centralized server.

OSD 直接服务客户端：由于任何网络设备都有最大并发连接上限，规模巨大时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 OSD 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，而不是一中央服务器。

2. **OSD Membership and Status:** Ceph OSD Daemons join a cluster and report on their status. At the lowest level, the Ceph OSD Daemon status is `up` or `down` reflecting whether or not it is running and able to service Ceph Client requests. If a Ceph OSD Daemon is `down` and `in` the Ceph Storage Cluster, this status may indicate the failure of the Ceph OSD Daemon. If a Ceph OSD Daemon is not running (e.g., it crashes), the Ceph OSD Daemon cannot notify the Ceph Monitor that it is `down`. The Ceph Monitor can ping a Ceph OSD Daemon periodically to ensure that it is running. However, Ceph also empowers Ceph OSD Daemons to determine if a neighboring OSD is `down`, to update the cluster map and to report it to the Ceph monitor(s). This means that Ceph monitors can remain light weight processes. See [Monitoring OSDs](#) and [Heartbeats](#) for additional details.

OSD 成员和状态：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`，反映它是否在运行、能否提供服务。如果一 OSD 状态为 `down` 且 `in`，表明 OSD 守护进程可能失败了；如果一 OSD 守护进程没在运行（比如崩溃了），它就不能亲自向监视器报告自己是 `down` 的。ceph 监视器能周期性地 ping OSD 守护进程，以确保它们在运行，然而它也能授权 OSD 进程去确认邻居 OSD 是否 `down` 了，并更新集群运行图、报告给监视器。这种机制意味着监视器还是轻量级进程。详情见 [Monitoring OSDs](#) 和 [Heartbeats](#)。

3. **Data Scrubbing:** As part of maintaining data consistency and cleanliness, Ceph OSD Daemons can scrub objects within placement groups. That is, Ceph OSD Daemons can compare object metadata in one placement group with its replicas in placement groups stored on other OSDs. Scrubbing (usually performed daily) catches bugs or filesystem errors. Ceph OSD Daemons also perform deeper scrubbing by comparing data in objects bit-for-bit. Deep scrubbing (usually performed weekly) finds bad sectors on a drive that weren't apparent in a light scrub. See [Data Scrubbing](#) for details on configuring scrubbing.

数据洗刷：作为维护数据一致性和清洁度的一部分，OSD 能洗刷归置组内的数据。就是说，Ceph OSD 能比较对象元数据位于不同 OSD 上的几个副本的元数据，以捕捉 OSD 缺陷或文件系统错误（每天）。OSD 也能做深度洗刷（每周），即按位比较对象中的数据，以找出轻度洗刷时未发现的硬盘坏扇区。

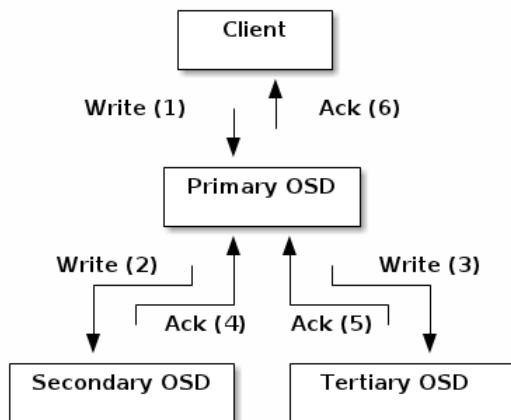
4. **Replication:** Like Ceph Clients, Ceph OSD Daemons use the CRUSH algorithm, but the Ceph OSD

Daemon uses it to compute where replicas of objects should be stored (and for rebalancing). In a typical write scenario, a client uses the CRUSH algorithm to compute where to store an object, maps the object to a pool and placement group, then looks at the CRUSH map to identify the primary OSD for the placement group.

复制：和 ceph 客户端一样，OSD 也用 CRUSH 算法，但用于计算副本存到哪里（也用于重均衡）。一个典型的情形是，一客户端用 CRUSH 算法算出对象应存到哪里，并把对象映射到存储池和归置组，然后查找 CRUSH 图来确定此归置组的主 OSD。

The client writes the object 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 object 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 object was stored successfully.

客户端把对象写入目标归置组的主 OSD，然后这个主 OSD 再用它的 CRUSH 图副本找出用于放对象副本的第二、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。



With the ability to perform data replication, Ceph OSD Daemons relieve Ceph clients from that duty, while ensuring high data availability and data safety.

有了做副本的能力，OSD 守护进程就可以减轻客户端的复制压力，同时保证了数据的高可靠性和安全性。

8.1.3 动态集群管理

Dynamic Cluster Management

In the [Scalability and High Availability](#) section, we explained how Ceph uses CRUSH, cluster awareness and intelligent daemons to scale and maintain high availability. Key to Ceph's design is the autonomous, self-healing, and intelligent Ceph OSD Daemon. Let's take a deeper look at how CRUSH works to enable modern cloud storage infrastructures to place data, rebalance the cluster and recover from faults dynamically.

在 [Scalability and High Availability](#) 一节，我们解释了 ceph 如何用 CRUSH、集群感知性和智能 OSD 守护进程来扩展和维护高可靠性。ceph 的关键设计是自治，自修复、智能的 OSD 守护进程。让我们深入了解下 CRUSH 如何运作，现代云存储基础设施如何动态地放置数据、重均衡、从错误中恢复。

8.1.3.1 关于存储池

About Pools

The Ceph storage system supports the notion of ‘Pools’, which are logical partitions for storing objects. Pools set the following parameters:

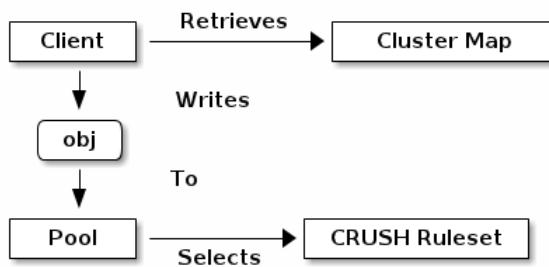
Ceph 存储系统支持“池”概念，它是存储对象的逻辑分区。存储池设置了如下参数：

- Ownership/Access to Objects
对象的所有权/访问权限；
- The Number of Object Replicas
对象副本数；

- The Number of Placement Groups, and
归置组数量；
- The CRUSH Ruleset to Use.
使用的 CRUSH 规则集。

Ceph Clients retrieve a [Cluster Map](#) from a Ceph Monitor, and write objects to pools. The pool's size or number of replicas, the CRUSH ruleset and the number of placement groups determine how Ceph will place the data.

ceph 客户端从监视器获取一张集群运行图，并把对象写入存储池。存储池的 size 或副本数、CRUSH 规则集和归置组数量决定着 ceph 如何放置数据。



8.1.3.2 PG 映射到 OSD

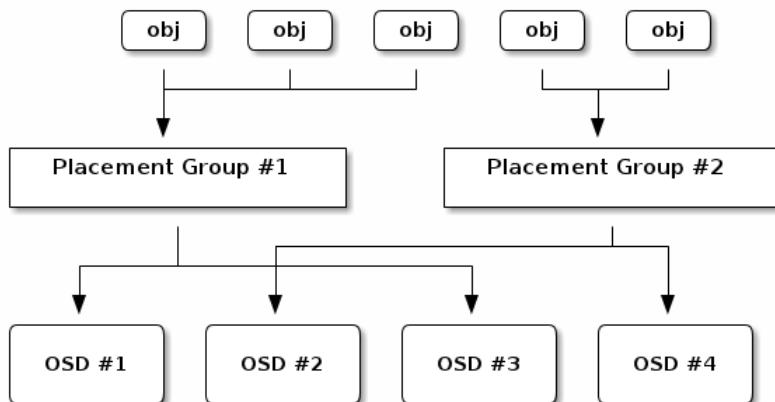
Mapping PGs to OSDs

Each pool has a number of placement groups. CRUSH maps PGs to OSDs dynamically. When a Ceph Client stores objects, CRUSH will map each object to a placement group.

各归置组都有很多归置组，CRUSH 动态的把它们映射到 OSD。ceph 客户端要存对象时，CRUSH 将把各对象映射到某个归置组。

Mapping objects to placement groups creates a layer of indirection between the Ceph OSD Daemon and the Ceph Client. The Ceph Storage Cluster must be able to grow (or shrink) and rebalance where it stores objects dynamically. If the Ceph Client “knew” which Ceph OSD Daemon had which object, that would create a tight coupling between the Ceph Client and the Ceph OSD Daemon. Instead, the CRUSH algorithm maps each object to a placement group and then maps each placement group to one or more Ceph OSD Daemons. This layer of indirection allows Ceph to rebalance dynamically when new Ceph OSD Daemons and the underlying OSD devices come online. The following diagram depicts how CRUSH maps objects to placement groups, and placement groups to OSDs.

把对象映射到归置组在 OSD 和客户端间创建了一个间接层。由于 ceph 集群必须能增大或缩小、并动态地重均衡。如果要让客户端“知道”哪个 OSD 有哪个对象，就会导致客户端和 OSD 密耦合；相反，CRUSH 算法把一堆对象映射到一归置组、然后再把各归置组映射到一或多个 OSD，这一间接层可以让 ceph 在 OSD 守护进程和底层设备上线时动态地重均衡。下列图表描述了如何用 CRUSH 把对象映射到归置组、再把归置组映射到 OSD。



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 object.

有了集群运行图副本和 CRUSH 算法，客户端就能精确地计算出到哪个 OSD 读、写某特定对象。

8.1.3.3 计算 PG ID

Calculating PG IDs

When a Ceph Client binds to a Ceph Monitor, it retrieves the latest copy of the [Cluster Map](#). With the cluster map, the client knows about all of the monitors, OSDs, and metadata servers in the cluster. **However, it doesn't know anything about object locations.**

ceph 客户端绑定到某监视器时，会索取最新的集群运行图副本，有了此图，客户端就能知道集群内的所有监视器、OSD、和元数据服务器。然而它对对象的位置一点也不了解。

Object locations get computed.

对象位置是计算出来的。

The only input required by the client is the object ID and the pool. It's simple: Ceph stores data in named pools (e.g., "liverpool"). When a client wants to store a named object (e.g., "john," "paul," "george," "ringo", etc.) it calculates a placement group using the object name, a hash code, the number of OSDs in the cluster and the pool name. Ceph clients use the following steps to compute PG IDs.

客户端只需输入对象 ID 和存储池，此事简单：ceph 把数据存在某存储池（如 liverpool）中。当客户端想要存命名对象（如 john、paul、george、ringo 等等）时，它用对象名计算归置组（一个哈希值）、OSD 号、存储池。ceph 按下列步骤计算 PG ID。

1. The client inputs the pool ID and the object ID. (e.g., pool = "liverpool" and object-id = "john")
 2. CRUSH takes the object ID and hashes it.
 3. CRUSH calculates the hash modulo the number of OSDs. (e.g., $0x58$) to get a PG ID.
 4. CRUSH gets the pool ID given the pool name (e.g., "liverpool" = 4)
 5. CRUSH prepends the pool ID to the pool ID to the PG ID (e.g., 4. $0x58$).
-
1. 客户端输入存储池 ID 和对象 ID（如 pool="liverpool" 和 object-id="john"）；
 2. CRUSH 拿到对象 ID 并哈希它；
 3. CRUSH 用 OSD 数（如 0x58）对哈希值取模，这就是归置组 ID；
 4. CRUSH 根据存储池名取得存储池 ID（如 liverpool=4）；
 5. CRUSH 把存储池 ID 加到 PG ID 之前。

Computing object locations is much faster than performing object location query over a chatty session. The algorithm allows a client to compute where objects **should** be stored, and enables the client to contact the primary OSD to store or retrieve the objects.

计算对象位置远快于查询定位，此算法允许客户端计算对象应该存到哪里，并允许客户端连接存储此主 OSD 来存储或检索对象。

8.1.3.4 互联和集合

Peering and Sets

In previous sections, we noted that Ceph OSD Daemons check each other's heartbeats and report back to the Ceph Monitor. Another thing Ceph OSD daemons do is called 'peering', which is the process of bringing all of the OSDs that store a Placement Group (PG) into agreement about the state of all of the objects (and their metadata) in that PG. In fact, Ceph OSD Daemons [Report Peering Failure](#) to the Ceph Monitors. Peering issues usually resolve themselves; however, if the problem persists, you may need to refer to the [Troubleshooting Peering Failure](#) section.

在前面的章节中，我们注意到 OSD 守护进程相互检查心跳并回馈给监视器；它们的另一行为叫“互联”，这是一种把一归置组内所有对象（及其元数据）所在的 OSD 带到一致状态的过程。事实上，OSD 守护进程会向监视器报告互联失败，互联问题一般会自行恢复，然而如果问题一直持续，你也许得参照 [Troubleshooting Peering Failure](#) 解决。

Note: Agreeing on the state does not mean that the PGs have the latest contents.

注意：对状态达成一致并不意味着 PG 持有最新内容。

The Ceph Storage Cluster was designed to store at least two copies of an object (i.e., `size = 2`), which is the minimum requirement for data safety. For high availability, a Ceph Storage Cluster should store more than two copies of an object (e.g., `size = 3` and `min size = 2`) so that it can continue to run in a **degraded** state while maintaining data safety.

ceph 存储集群被设计为至少存储两份（即 `size=2`），这是保证数据安全的最小需求。为保证高可靠性，ceph 存储集群应该至少保存一对象的两个副本（如 `size = 3` 且 `min size = 2`），这样才能在维持数据安全的同时、仍保持在 **degraded** 状态。

Referring back to the diagram in [Smart Daemons Enable Hyperscale](#), we do not name the Ceph OSD Daemons specifically (e.g., `osd.0`, `osd.1`, etc.), but rather refer to them as **Primary**, **Secondary**, and so forth. By convention, the **Primary** is the first OSD in the **Acting Set**, and is responsible for coordinating the peering process for each placement group where it acts as the **Primary**, and is the **ONLY** OSD that will accept client-initiated writes to objects for a given placement group where it acts as the **Primary**.

回想前面 [Smart Daemons Enable Hyperscale](#) 中的图表，我们没明确地提 OSD 守护进程的名字（如 `osd.0`、`osd.1` 等等），而是称之为 **主**、**次**、以此类推。按惯例，**主 OSD** 是 **acting set** 中的第一个 OSD，而且它负责协调各归置组的互联进程，所以称之为 **主 OSD**；也只有它会接受客户端到某归置组最初的写入请求。

When a series of OSDs are responsible for a placement group, that series of OSDs, we refer to them as an **Acting Set**. An **Acting Set** may refer to the Ceph OSD Daemons that are currently responsible for the placement group, or the Ceph OSD Daemons that were responsible for a particular placement group as of some epoch.

当一系列 OSD 负责一归置组时，这一系列的 OSD 就成为一个 **acting set**。一个 **acting set** 可对应当前负责此归置组的一些 OSD，或者说一些 OSD 在一些时间结上负责某个特定归置组。

The Ceph OSD daemons that are part of an **Acting Set** may not always be **up**. When an OSD in the **Acting Set** is **up**, it is part of the **Up Set**. The **Up Set** is an important distinction, because Ceph can remap PGs to other Ceph OSD Daemons when an OSD fails.

OSD 守护进程作为 **acting set** 的一部分，不一定总在 **up** 状态。当一 OSD 在 **acting set** 中是 **up** 状态时，它就是 **up set** 的一部分。**up set** 是个重要特征，因为某 OSD 失败时 ceph 会把 PG 映射到其他 OSD。

Note: In an **Acting Set** for a PG containing `osd.25`, `osd.32` and `osd.61`, the first OSD, `osd.25`, is the **Primary**. If that OSD fails, the Secondary, `osd.32`, becomes the **Primary**, and `osd.25` will be removed from the **Up Set**.

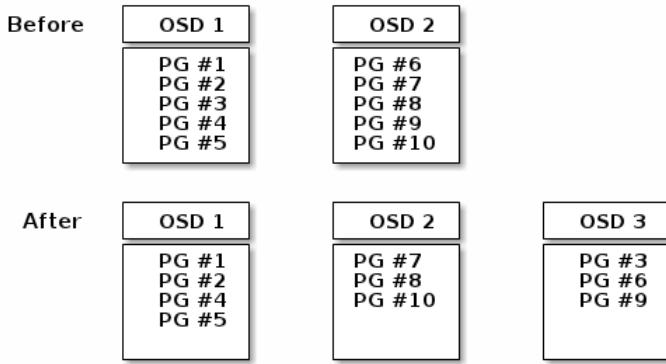
注意：在某 PG 的 **acting set** 中包含了 `osd.25`、`osd.32` 和 `osd.61`，第一个 `osd.25` 是 **主 OSD**，如果它失败了，第二个 `osd.32` 就成为 **主 OSD**，`osd.25` 会被移出 **up set**。

8.1.3.5 重均衡

Rebalancing

When you add a Ceph OSD Daemon to a Ceph Storage Cluster, the cluster map gets updated with the new OSD. Referring back to [Calculating PG IDs](#), this changes the cluster map. Consequently, it changes object placement, because it changes an input for the calculations. The following diagram depicts the rebalancing process (albeit rather crudely, since it is substantially less impactful with large clusters) where some, but not all of the PGs migrate from existing OSDs (OSD 1, and OSD 2) to the new OSD (OSD 3). Even when rebalancing, CRUSH is stable. Many of the placement groups remain in their original configuration, and each OSD gets some added capacity, so there are no load spikes on the new OSD after rebalancing is complete.

你向 ceph 存储集群新增一 OSD 守护进程时，集群运行图就要用新增的 OSD 更新。回想 [计算 PG ID](#)，这个动作会更改集群运行图，因此也改变了对象位置，因为计算时的输入条件变了。下面的图描述了重均衡过程（此图很粗略，因为在大型集群里变动幅度小的多），是其中的一些而不是所有 PG 都从已有 OSD (OSD 1 和 2) 迁移到新 OSD (OSD 3)。即使在重均衡中，CRUSH 都是稳定的，很多归置组仍维持最初的配置，且各 OSD 都腾出了些空间，所以重均衡完成后新 OSD 上不会有突增负载。



8.1.3.6 数据一致性

Data Consistency

As part of maintaining data consistency and cleanliness, Ceph OSDs can also scrub objects within placement groups. That is, Ceph OSDs can compare object metadata in one placement group with its replicas in placement groups stored in other OSDs. Scrubbing (usually performed daily) catches OSD bugs or filesystem errors. OSDs can also perform deeper scrubbing by comparing data in objects bit-for-bit. Deep scrubbing (usually performed weekly) finds bad sectors on a disk that weren't apparent in a light scrub.

作为维护数据一致和清洁的一部分，OSD 也能洗刷归置组内的对象，也就是说，OSD 会比较归置组内位于不同 OSD 的各对象副本的元数据。洗刷（通常每天执行）是为捕获 OSD 缺陷和文件系统错误，OSD 也能执行深度洗刷：按位比较对象内的数据；深度洗刷（通常每周执行）是为捕捉磁盘上的坏扇区，在轻度洗刷时不可能发现此问题。

See [Data Scrubbing](#) for details on configuring scrubbing.

关于数据洗刷的配置见 [Data Scrubbing](#)。

8.1.4 扩展 ceph

Extending Ceph

You can extend Ceph by creating shared object classes called ‘Ceph Classes’. Ceph loads .so classes stored in the `osd class dir` directory dynamically (i.e., `$libdir/rados-classes` by default). When you implement a class, you can create new object methods that have the ability to call the native methods in the Ceph Object Store, or other class methods you incorporate via libraries or create yourself.

你可以用“Ceph Classes”共享对象类来扩展 ceph 功能，ceph 会动态地载入位于 osd class dir 目录下的.so 类文件（即默认的\$libdir/rados-classes）。如果你实现了一个类，就可以创建新的对象方法去调用 ceph 对象存储内的原生方法、或者公用库或自建库里的其它类方法。

On writes, Ceph Classes can call native or class methods, perform any series of operations on the inbound data and generate a resulting write transaction that Ceph will apply atomically.

写入时，ceph 类能调用原生或类方法，对入栈数据执行任意操作、生成最终写事务，并由 ceph 原子地应用。

On reads, Ceph Classes can call native or class methods, perform any series of operations on the outbound data and return the data to the client.

读出时，ceph 类能调用原生或类方法，对出栈数据执行任意操作、把数据返回给客户端。

Ceph Class Example

A Ceph class for a content management system that presents pictures of a particular size and aspect ratio could take an inbound bitmap image, crop it to a particular aspect ratio, resize it and embed an invisible copyright or watermark to help protect the intellectual property; then, save the resulting bitmap image to the object store.

ceph 类实例

一个为内容管理系统写的类可能要实现如下功能，它要展示特定尺寸和长宽比的位图，所以入栈图片要裁剪为特定长宽比、缩放它、并嵌入个不可见的版权或水印用于保护知识产权；然后把生成的位图保存为对象。

See `src/objclass/objclass.h`, `src/fooclass.cc` and `src/barclass` for exemplary

implementations.

实现模型见 `src/objclass/objclass.h`、`src/fooclass.cc`、和 `src/barclass`。

8.1.5 总结

Summary

Ceph Storage Clusters are dynamic—like a living organism. Whereas, many storage appliances do not fully utilize the CPU and RAM of a typical commodity server, Ceph does. From heartbeats, to peering, to rebalancing the cluster or recovering from faults, Ceph offloads work from clients (and from a centralized gateway which doesn't exist in the Ceph architecture) and uses the computing power of the OSDs to perform the work. When referring to [Hardware Recommendations](#) and the [Network Config Reference](#), be cognizant of the foregoing concepts to understand how Ceph utilizes computing resources.

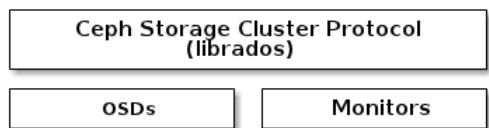
ceph 存储集群是动态的——像个生物体。尽管很多存储应用不能完全利用一台普通服务器上的 CPU 和 RAM 资源，但是 ceph 能。从心跳到互联、到重均衡、再到错误恢复，ceph 都把客户端（和中央网关，但在 ceph 架构中不存在）解脱了，用 OSD 的计算资源完成此工作。参考前面的 [Hardware Recommendations](#) 和 [Network Config Reference](#) 理解前述概念，就不难理解 ceph 如何利用计算资源了。

8.2 ceph 协议

Ceph Protocol

Ceph Clients use the native protocol for interacting with the Ceph Storage Cluster. Ceph packages this functionality into the `librados` library so that you can create your own custom Ceph Clients. The following diagram depicts the basic architecture.

ceph 客户端用原生协议和存储集群交互，ceph 把此功能封装进了 librados 库，这样你就能创建自己的定制客户端了，下图描述了基本架构。



8.2.1 原生协议和 librados

Native Protocol and librados

Modern applications need a simple object storage interface with asynchronous communication capability. The Ceph Storage Cluster provides a simple object storage interface with asynchronous communication capability. The interface provides direct, parallel access to objects throughout the cluster.

现代程序都需要可异步通讯的简单对象存储接口。ceph 存储集群提供了一个有异步通讯能力的简单对象存储接口，此接口提供了直接写入、并行访问集群的功能。

- Pool Operations
- Snapshots and Copy-on-write Cloning
- Read/Write Objects - Create or Remove - Entire Object or Byte Range - Append or Truncate
- Create/Set/Get/Remove XATTRs
- Create/Set/Get/Remove Key/Value Pairs
- Compound operations and dual-ack semantics
- Object Classes

- 存储池操作；
- 快照和写时复制克隆；
- 读/写对象——创建或删除整个对象或字节范围、可追加或截断；

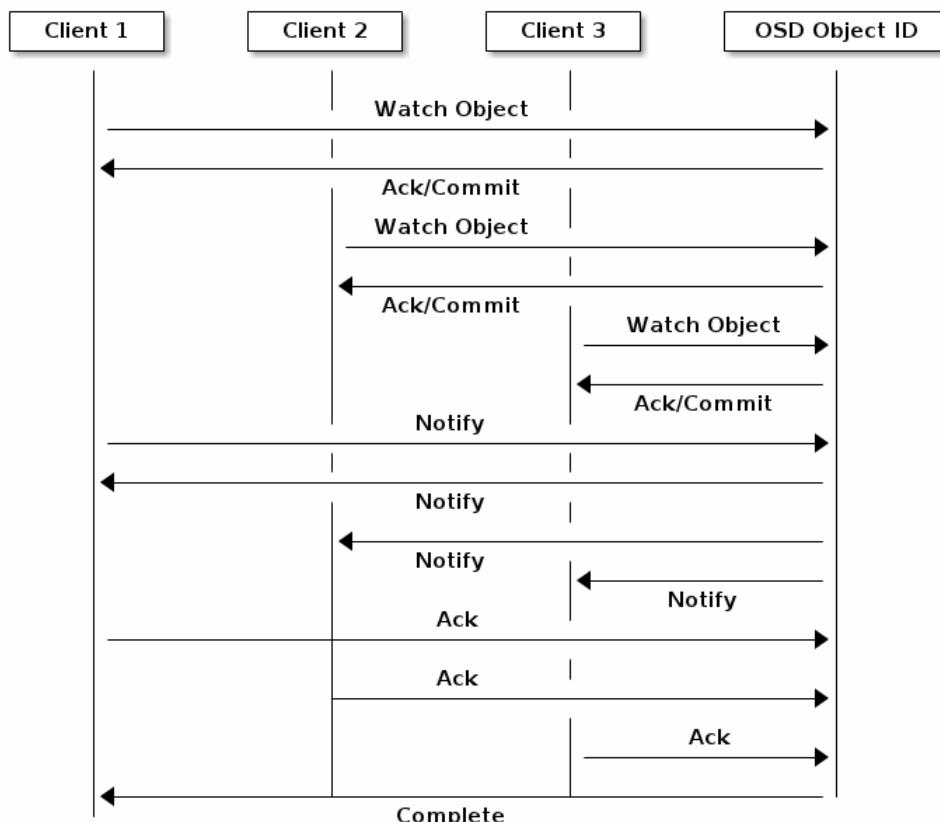
- 创建/设置/获取/删除扩展属性；
- 创建/设置/获取/删除键/值对；
- 混合操作和双重确认；
- 对象类。

8.2.2 对象关注/通知

Object Watch/Notify

A client can register a persistent interest with an object and keep a session to the primary OSD open. The client can send a notification message and payload to all watchers and receive notification when the watchers receive the notification. This enables a client to use any object a synchronization/communication channel.

客户端可以注册对某个对象的持续兴趣，并使到主 OSD 的会话保持活跃。客户端可以发送一通知消息和载荷给所有关注者、并可收集关注者的接收通知。这个功能使得客户端可把任意对象用作同步/通讯通道。



8.2.3 数据裁剪

Data Striping

Storage devices have throughput limitations, which impact performance and scalability. So storage systems often support [striping](#)—storing sequential pieces of information across multiple storage devices—to increase throughput and performance. The most common form of data striping comes from [RAID](#). The RAID type most similar to Ceph's striping is [RAID 0](#), or a 'striped volume.' Ceph's striping offers the throughput of RAID 0 striping, the reliability of n-way RAID mirroring and faster recovery.

存储设备都有吞吐量限制，它会影响性能和伸缩性，所以存储系统一般都支持条带化（把连续的信息分段存储于多个设备）以增加吞吐量和性能。数据条带化最常见于 RAID 中，RAID 中最接近 ceph 条带化方式的是 RAID 0、或者条带化的卷，ceph 的条带化提供了像 RAID 0 一样的吞吐量、像 N 路 RAID 镜像一样的可靠性、和更快的恢复。

Ceph provides three types of clients: Ceph Block Device, Ceph Filesystem, and Ceph Object Storage. A Ceph Client converts its data from the representation format it provides to its users (a block device image, RESTful objects, CephFS filesystem directories) into objects for storage in the Ceph Storage Cluster.

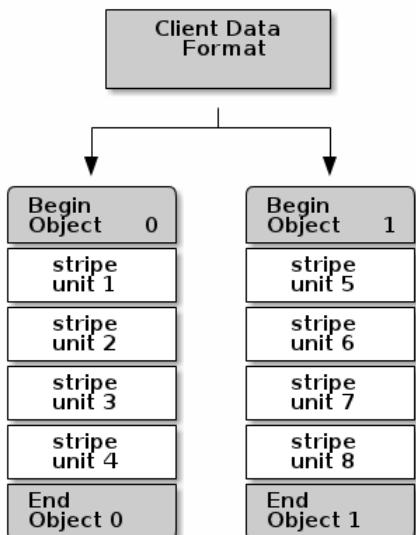
ceph 提供了三种类型的客户端：块设备、文件系统和对象存储。一个 ceph 客户端把展现给用户的数据格式（一块设备映像、RESTful 对象、CephFS 文件系统目录）转换为可存储于 ceph 存储集群的对象。

Tip: The objects Ceph stores in the Ceph Storage Cluster are not striped. Ceph Object Storage, Ceph Block Device, and the Ceph Filesystem stripe their data over multiple Ceph Storage Cluster objects. Ceph Clients that write directly to the Ceph Storage Cluster via librados must perform the the striping (and parallel I/O) for themselves to obtain these benefits.

提示：在 ceph 存储集群内存储的那些对象是没条带化的。ceph 对象存储、ceph 块设备、和 ceph 文件系统把他们的数据条带化为 ceph 存储集群内的对象，客户端通过 librados 直接写入 ceph 存储集群前必须先自己条带化（和并行 I/O）才能享用这些优势。

The simplest Ceph striping format involves a stripe count of 1 object. Ceph Clients write stripe units to a Ceph Storage Cluster object until the object is at its maximum capacity, and then create another object for additional stripes of data. The simplest form of striping may be sufficient for small block device images, S3 or Swift objects and CephFS files. However, this simple form doesn't take maximum advantage of Ceph's ability to distribute data across placement groups, and consequently doesn't improve performance very much. The following diagram depicts the simplest form of striping:

最简单的 ceph 条带化格式就是拆分为一个对象。ceph 客户端分散地把条带单元写入 ceph 存储集群的对象，直到对象容量达到上限，才会再创建另一个对象存储未完的数据。这种最简单的条带化对小个儿的块设备映像、S3、Swift 对象或 CephFS 文件来说也许足够了；然而这种简单的形式不能最大化 ceph 在归置组间分布数据的能力，也不能最大化性能。下图描述了条带化的最简形式：



If you anticipate large images sizes, large S3 or Swift objects (e.g., video), or large CephFS directories, you may see considerable read/write performance improvements by striping client data over multiple objects within an object set. Significant write performance occurs when the client writes the stripe units to their corresponding objects in parallel. Since objects get mapped to different placement groups and further mapped to different OSDs, each write occurs in parallel at the maximum write speed. A write to a single disk would be limited by the head movement (e.g. 6ms per seek) and bandwidth of that one device (e.g. 100MB/s). By spreading that write over multiple objects (which map to different placement groups and OSDs) Ceph can reduce the number of seeks per drive and combine the throughput of multiple drives to achieve much faster write (or read) speeds.

如果要处理大尺寸图像、大个 S3 或 Swift 对象（如视频）、或大个的 CephFS 目录，你就能看到条带化到多个对象能带来显著的读/写性能提升。当客户端能把条带单元并行地写入相应用对象时，才会有明显的写性能，因为对象映射到了不同的归置组、并对应不同 OSD，可以分别以最大速度写入。到磁盘的写入受限于磁头移动（即 6ms 寻道时间）、存储设备带宽，ceph 把写入分布到多个对象（它们映射到了不同归置组和 OSD），这样可减少每设备寻道次数、联合多个驱动器的吞吐量，以达到更高的写（或读）速度。

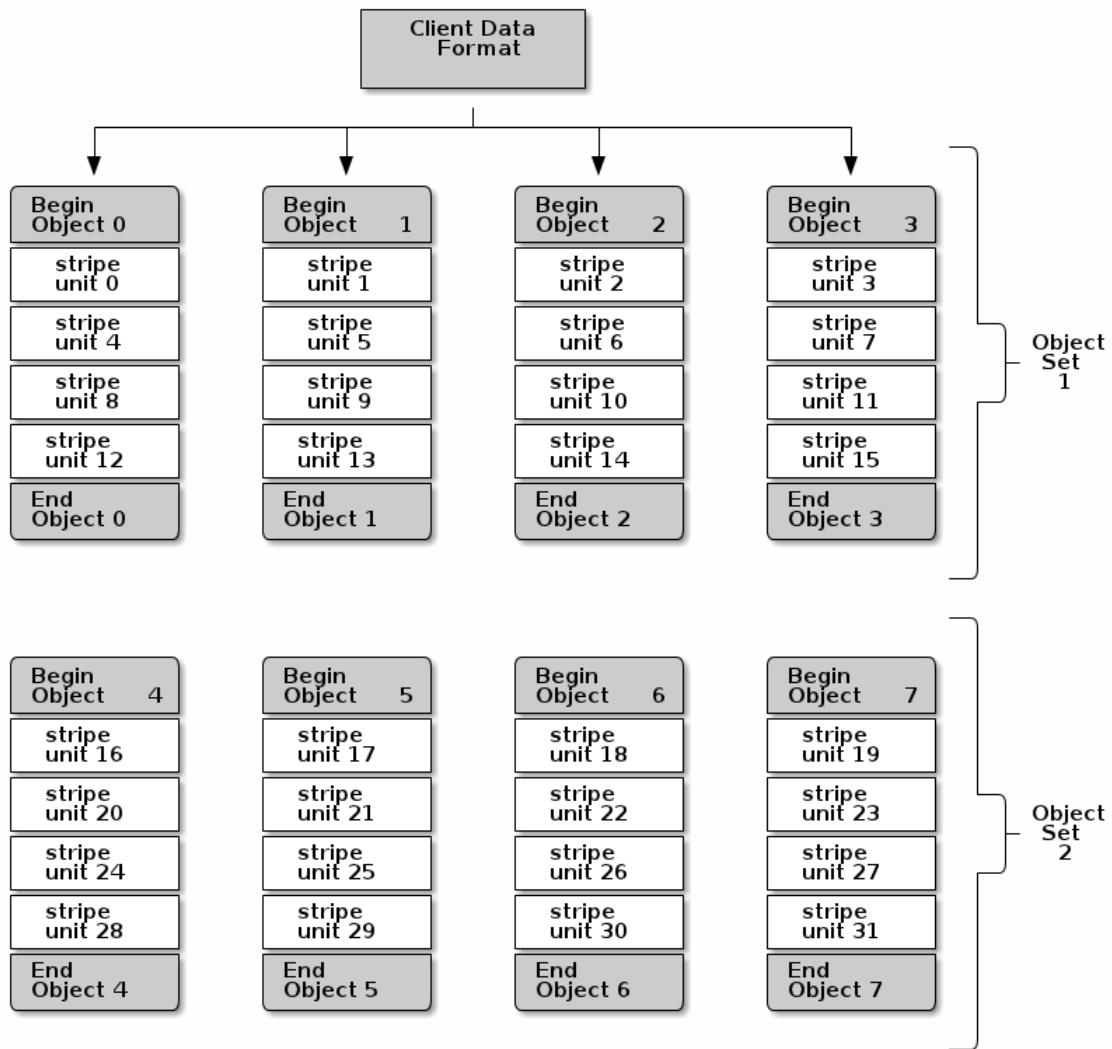
Note: Striping is independent of object replicas. Since CRUSH replicates objects across OSDs, stripes get replicated automatically.

注意：条带化独立于对象复制。因为 CRUSH 会在 OSD 间复制对象，数据条带是自动被复制的。

In the following diagram, client data gets striped across an object set (object set 1 in the following diagram) consisting of 4 objects, where the first stripe unit is stripe unit 0 in object 0, and the fourth

stripe unit is stripe unit 3 in object 3. After writing the fourth stripe, the client determines if the object set is full. If the object set is not full, the client begins writing a stripe to the first object again (object 0 in the following diagram). If the object set is full, the client creates a new object set (object set 2 in the following diagram), and begins writing to the first stripe (stripe unit 16) in the first object in the new object set (object 4 in the diagram below).

在下图中，客户端数据条带化到一个对象集（下图中的 object set 1），它包含 4 个对象，其中，第一个条带单元是 object 0 的 stripe unit 0、第四个条带是 object 3 的 stripe unit 3，写完第四个条带，客户端要确认对象集是否满了。如果对象集没满，客户端再从第一个对象起写入条带（下图中的 object 0）；如果对象集满了，客户端就得创建新对象集（下图的 object set 2），然后从新对象集中的第一个对象（下图中的 object 4）起开始写入第一个条带（stripe unit 16）。



Three important variables determine how Ceph stripes data:

三个重要变量决定着 ceph 如何条带化数据：

- **Object Size:** Objects in the Ceph Storage Cluster have a maximum configurable size (e.g., 2MB, 4MB, etc.). The object size should be large enough to accommodate many stripe units, and should be a multiple of the stripe unit.

对象尺寸：ceph 存储集群里的对象有最大可配置尺寸（如 2MB、4MB 等等），对象尺寸必须足够大才能容纳很多条带单元、而且应该是条带单元的整数倍。

- **Stripe Width:** Stripes have a configurable unit size (e.g., 64kb). The Ceph Client divides the data it will write to objects into equally sized stripe units, except for the last stripe unit. A stripe width, should be a fraction of the Object Size so that an object may contain many stripe units.

条带宽度：条带都有可配置的单位尺寸（如 64KB）。ceph 客户端把数据等分成适合写入对象的条带单元，除了最后一个。条带宽度应该是对象尺寸的分数片段，这样对象才能包含很多条带单元。

- **Stripe Count:** The Ceph Client writes a sequence of stripe units over a series of objects determined by the stripe count. The series of objects is called an object set. After the Ceph Client writes to the last object in the object set, it returns to the first object in the object set.

条带数量：ceph 客户端把一系列条带单元写入由条带数量所确定的一系列对象，这一系列的对象称为一个对象集。客户端写到对象集内的最后一个对象时，再返回到第一个。

Important: Test the performance of your striping configuration before putting your cluster into production. You CANNOT change these striping parameters after you stripe the data and write it to objects.

重要：把集群投入生产环境前要先测试条带化配置，因为把数据条带化到对象中之后这些参数就不可更改了。

Once the Ceph Client has striped data to stripe units and mapped the stripe units to objects, Ceph's CRUSH algorithm maps the objects to placement groups, and the placement groups to Ceph OSD Daemons before the objects are stored as files on a storage disk.

ceph 客户端把数据等分为条带单元并映射到对象后，用 CRUSH 算法把对象映射到归置组、归置组映射到 OSD，然后才能以文件形式存储到硬盘上。

Note: Since a client writes to a single pool, all data striped into objects get mapped to placement groups in the same pool. So they use the same CRUSH map and the same access controls.

注意：因为客户端写入单个存储池，条带为对象的所有数据也被映射到同一存储池内的归置组，所以它们要共享相同的 CRUSH 图和相同的访问权限。

8.3 ceph 客户端

Ceph Clients

Ceph Clients include a number of service interfaces. These include:

ceph 客户端包括数种服务接口，有：

- **Block Devices:** The [Ceph Block Device](#) (a.k.a., 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：ceph 块设备（也叫 RBD）服务提供了大小可调、精炼、支持快照和克隆。为提供高性能，ceph 把块设备条带化到整个集群。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对象过载。

- **Object Storage:** The [Ceph Object Storage](#) (a.k.a., RGW) service provides RESTful APIs with interfaces that are compatible with Amazon S3 and OpenStack Swift.

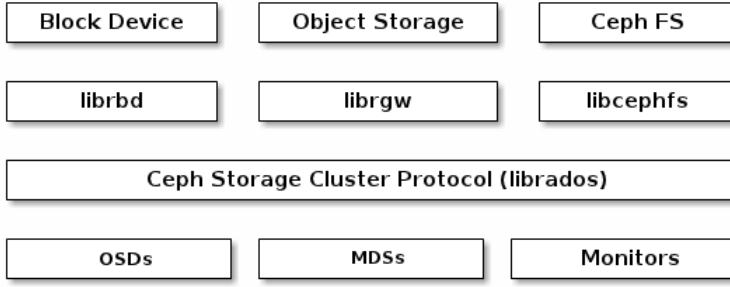
对象存储：ceph 对象存储（也叫 RGW）服务提供了 RESTful API，它有与 Amazon S3 和 OpenStack Swift 兼容的接口。

- **Filesystem:** The [Ceph Filesystem](#) (CephFS) service provides a POSIX compliant filesystem usable with `mount` or as a filesystem in user space (FUSE).

文件系统：Ceph 文件系统（CephFS）服务提供了兼容 POSIX 的文件系统，可以直接 `mount` 或挂载为用户空间文件系统（FUSE）。

Ceph can run additional instances of OSDs, MDSs, and monitors for scalability and high availability. The following diagram depicts the high-level architecture.

ceph 能额外运行多个 OSD、MDS、和监视器来保证伸缩性和高可靠性，下图描述了高级架构。



8.3.1 ceph 对象存储

Ceph Object Storage

The Ceph Object Storage daemon, `radosgw`, is a FastCGI service that provides a [RESTful](#) HTTP API to store objects and metadata. It layers on top of the Ceph Storage Cluster 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.

`ceph` 对象存储守护进程是 `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 `ceph` 存储集群之上, 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 网关使用统一的命名空间, 也就是说, 你可以用 OpenStack Swift 兼容的 API 或者 Amazon S3 兼容的 API; 例如, 你可以用一个程序通过 S3 兼容 API 写入数据、然后用另一个程序通过 Swift 兼容 API 读出。

S3/Swift Objects and Store Cluster Objects Compared

Ceph's Object Storage uses the term **object** to describe the data it stores. S3 and Swift objects are not the same as the objects that Ceph writes to the Ceph Storage Cluster. Ceph Object Storage objects are mapped to Ceph Storage Cluster objects. The S3 and Swift objects do not necessarily correspond in a 1:1 manner with an object stored in the storage cluster. It is possible for an S3 or Swift object to map to multiple Ceph objects.

S3/Swift 对象和存储集群对象比较

`ceph` 对象存储用对象这个术语来描述它存储的数据。S3 和 Swift 对象不同于 `ceph` 写入存储集群的对象, `ceph` 对象存储系统内的对象可以映射到 `ceph` 存储集群内的对象; S3 和 Swift 对象却不一定 1:1 地映射到存储集群内的对象, 它有可能映射到了多个 `ceph` 对象。

See [Ceph Object Storage](#) for details.

详情见 [Ceph Object Storage](#)。

8.3.2 ceph 块设备

Ceph Block Device

A Ceph Block Device stripes a block device image over multiple objects in the Ceph Storage 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.

Ceph 块设备把一个设备映像条带化到集群内的多个对象, 其中各对象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。

Thin-provisioned snapshottable Ceph Block Devices are an attractive option for virtualization and cloud computing. In virtual machine scenarios, people typically deploy a Ceph Block Device 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 thin-provisioned Ceph Block Devices with Qemu and `libvirt` to support OpenStack and CloudStack among other solutions.

简配、可快照的 `ceph` 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 网络存储驱动部署 ceph 块设备，其中宿主机用 librbd 向访客提供块设备服务；很多云计算堆栈用 libvirt 和管理程序集成。你可以用简配的 ceph 块设备搭配 Qemu 和 libvirt 来支持 OpenStack 和 CloudStack，一起构成完整的方案。

While we do not provide librbd support with other hypervisors at this time, you may also use Ceph Block Device kernel objects to provide a block device to a client. Other virtualization technologies such as Xen can access the Ceph Block Device kernel object(s). This is done with the command-line tool rbd.

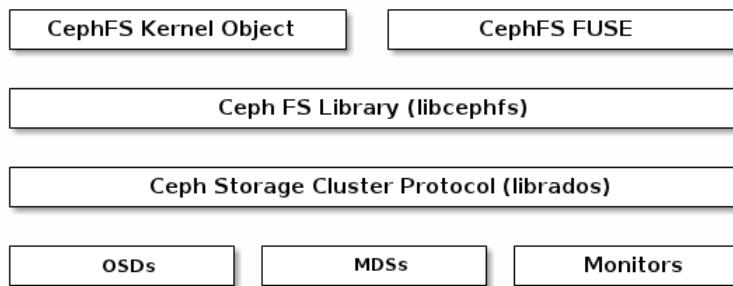
现在还没其它管理程序支持 librbd，你可以用 ceph 块设备内核对象向客户端提供块设备。其它虚拟化技术，像 Xen 能访问 ceph 块设备内核对象，用命令行工具 rbd 实现。

8.3.3 ceph 文件系统

Ceph Filesystem

The Ceph Filesystem (Ceph FS) provides a POSIX-compliant filesystem as a service that is layered on top of the object-based Ceph Storage Cluster. Ceph FS files get mapped to objects that Ceph stores in the Ceph Storage Cluster. Ceph Clients mount a CephFS filesystem as a kernel object or as a Filesystem in User Space (FUSE).

ceph 文件系统（Ceph FS）是与 POSIX 兼容的文件系统服务，坐落于基于对象的 ceph 存储集群之上，其内的文件被映射到 ceph 存储集群内的对象。客户端可以把此文件系统挂载为内核对象或用户空间文件系统（FUSE）。



The Ceph Filesystem service includes the Ceph Metadata Server (MDS) deployed with the Ceph Storage cluster. The purpose of the MDS is to store all the filesystem metadata (directories, file ownership, access modes, etc) in high-availability Ceph Metadata Servers where the metadata resides in memory. The reason for the MDS (a daemon called ceph-mds) is that simple filesystem operations like listing a directory or changing a directory (ls, cd) would tax the Ceph OSD Daemons unnecessarily. So separating the metadata from the data means that the Ceph Filesystem can provide high performance services without taxing the Ceph Storage Cluster.

ceph 文件系统服务包含随 ceph 存储集群部署的元数据服务器（MDS）。MDS 的作用是把所有文件系统元数据（目录、文件所有者、访问模式等等）永久存储在相当可靠的元数据服务器中，元数据驻留在内存中。

MDS（名为 ceph-mds 的守护进程）存在的原因是，简单的文件系统操作像列出目录（ls）、或进入目录（cd），这些操作本无需扰动 OSD。所以把元数据从数据里分出来意味着 ceph 文件系统能提供高性能服务，又没额外增加存储集群负载。

Ceph FS separates the metadata from the data, storing the metadata in the MDS, and storing the file data in one or more objects in the Ceph Storage Cluster. 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 FS 从数据中分离出了元数据，并存储于 MDS，文件数据存储于存储集群中的一或多个对象。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.

伸缩性：多个 ceph-mds 例程可以同时处于 **active** 状态，它们会把目录树拆分为子树（和单个热点目录的碎片），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* instance for high availability.

待命和活跃 MDS 可组合，例如，运行 3 个处于 **active** 状态的 ceph-mds 例程以实现扩展、和 1 个 *standby* 例程以实现高可用性。