部署指南

SUSE Enterprise Storage 5

部署指南

SUSE Enterprise Storage 5

作者：Tomáš Bažant、Jana Haláčková、Sven Seeberg

出版日期：2019-09-02

SUSE LLC
10 Canal Park Drive
Suite 200
Cambridge MA 02141
USA
https://www.suse.com/documentation

目录

关于本指南 ix

I SUSE ENTERPRISE STORAGE 1

1 SUSE Enterprise Storage 和 Ceph 2

1.1 Ceph 的特性 2

1.2 核心组件 3

RADOS 3 • CRUSH 4 • Ceph 节点和守护进程 4

1.3 存储结构 6

存储池 6 • 归置组 6 • 示例 6

1.4 BlueStore 8

1.5 其他信息 9

2 硬件要求和建议 10

2.1 对象存储节点 10

最低要求 10 • 最小磁盘大小 11 • BlueStore 的 WAL 和 DB 设备的建议大小 11 • 使用 SSD 存储 OSD 日记 12 • 磁盘的最大建议数量 12

2.2 监视器节点 13

2.3 对象网关节点 13

2.4 元数据服务器节点 14

2.5 Salt Master 14

2.6 iSCSI 节点 14

2.7 网络建议 14
将专用网添加到正在运行的集群 15 • 不同子网中的监视器节点 16
2.8 命名限制 16
2.9 共享一台服务器的 OSD 和监视器 17
2.10 最低集群配置 17
2.11 建议的生产集群配置 18
2.12 SUSE Enterprise Storage 以及其他 SUSE 产品 19
SUSE Manager 19
3 Ceph 管理节点 HA 设置 20
3.1 Ceph 管理节点的 HA 集群概述 20
3.2 构建用于 Ceph 管理节点的 HA 集群 21
II 集群部署和升级 23
4 使用 DeepSea/Salt 部署 24
4.1 阅读发行说明 24
4.2 DeepSea 简介 25
组织和重要位置 26 • 定位 Minion 27
4.3 集群部署 29
4.4 DeepSea CLI 37
DeepSea CLI：监视模式 37 • DeepSea CLI：独立模式 37
4.5 配置和自定义 39
policy.cfg 文件 39 • 自定义 ceph.conf 文件 46
5 从旧版本升级 47
5.1 阅读发行说明 47

5.2 一般升级过程 47

5.3 升级期间加密 OSD 49

5.4 从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5 51

将 OSD 迁移到 BlueStore 57

5.5 从 SUSE Enterprise Storage 4（`ceph-deploy` 部署）升级到版本 5 60

5.6 从 SUSE Enterprise Storage 4（Crowbar 部署）升级到版本 5 66

5.7 从 SUSE Enterprise Storage 3 升级到版本 5 72

6 备份集群配置 73

6.1 备份 Salt 配置 73

6.2 备份 DeepSea 配置 73

7 自定义默认配置 74

7.1 使用自定义的配置文件 74

禁用部署步骤 74 • 替换部署步骤 75 • 修改部署步骤 77 • 修改部署阶段 78 • 在阶段 0 期间禁用更新和重引导 79

7.2 修改已发现的配置 80

III 安装其他服务 83

8 安装用于访问数据的服务 84

9 Ceph Object Gateway 85

9.1 手动安装对象网关 85

对象网关配置 86

10 安装 iSCSI 网关 92

10.1 iSCSI 块存储 92

Linux 内核 iSCSI 目标 93 • iSCSI 发起程序 93

10.2 有关 lrbd 的一般信息 94

10.3 部署考虑事项 95

10.4 安装和配置 96

将 iSCSI 网关部署到 Ceph 集群 96 • 创建 RBD 映像 96 • 通过 iSCSI 导出 RBD 映像 97 • 可选设置 100 • 高级设置 101

10.5 使用 tcmu-runner 导出 RADOS 块设备映像 105

安装 106 • 配置和部署 106 • 用法 108

11 安装 CephFS 109

11.1 支持的 CephFS 方案和指导 109

11.2 Ceph 元数据服务器 110

添加元数据服务器 110 • 配置元数据服务器 110

11.3 CephFS 111

创建 CephFS 111 • MDS 集群大小 112 • MDS 集群和更新 113

12 安装 NFS Ganesha 114

12.1 准备 114

一般信息 114 • 要求摘要 114

12.2 示例安装 115

12.3 高可用性主动-被动配置 116

基本安装 116 • 清理资源 119 • 设置 Ping 资源 119 • NFS Ganesha HA 和 DeepSea 120

12.4 更多信息 121

13 通过 Samba 导出 CephFS 122

13.1 示例安装 122

13.2 高可用性配置 123

A 文档更新 128

A.1 2018 年 10 月（SUSE Enterprise Storage 5.5 发布） 128

A.2 2017 年 11 月（维护性更新） 131

A.3 2017 年 10 月（SUSE Enterprise Storage 5 发布） 132

关于本指南

SUSE Enterprise Storage 是 SUSE Linux Enterprise 的扩展。它融合了 Ceph (http://ceph.com/) 存储项目的功能与 SUSE 的企业工程和支持。SUSE Enterprise Storage 为 IT 组织提供了部署分布式存储体系结构的能力，该体系结构可支持使用市售硬件平台的许多用例。

本指南可帮助您了解 SUSE Enterprise Storage 的概念，并重点介绍如何管理 Ceph 基础架构。它还说明了如何将 Ceph 与其他相关解决方案（例如 OpenStack 或 KVM）搭配使用。

本手册中的许多章节中都包含指向附加文档资源的链接。其中包括系统上提供的附加文档以及因特网上提供的文档。

有关该产品可用文档的概述和最新文档更新，请参见 http://www.suse.com/documentation。

1 可用文档

针对本产品提供的手册如下：

《管理指南》

该指南说明了安装后通常需要执行的各项管理任务。该指南还介绍了将 Ceph 与 `libvirt`、Xen 或 KVM 等虚拟化解决方案集成的步骤，以及通过 iSCSI 和 RADOS 网关访问集群中存储的对象的方法。

部署指南

引导您完成 Ceph 集群及 Ceph 所有相关服务的安装步骤。该指南还阐述了基本 Ceph 集群结构，并提供了相关的术语。

在已安装系统的 `/usr/share/doc/manual` 下可以找到产品手册的 HTML 版本。在 http://www.suse.com/documentation 上可以找到最新的文档更新，并可从中下载多种格式的产品文档。

2 反馈

提供了多种反馈渠道：

错误和增强请求

有关产品可用的服务和支持选项，请参见 http://www.suse.com/support/ 。

要报告产品组件的错误，请从 http://www.suse.com/support/ 登录 Novell Customer Center，然后选择 My Support（我的支持）› Service Request（服务请求）。

用户意见

我们希望收到您对本手册和本产品中包含的其他文档的意见和建议。请使用联机文档每页底部的"用户注释"功能或转到 http://www.suse.com/documentation/feedback.html 并在此处输入注释。

邮件

如有对本产品文档的反馈，也可以发送邮件至 `doc-team@suse.de` 。请确保反馈中含有文档标题、产品版本和文档发布日期。要报告错误或给出增强建议，请提供问题的简要说明并指出相应章节编号和页码（或 URL）。

3 文档约定

以下是本手册中使用的版式约定：

- `/etc/passwd`：目录名称和文件名
- `placeholder`：将 `placeholder` 替换为实际值
- `PATH`：环境变量 PATH
- `ls`、`--help`：命令、选项和参数
- `user`：用户和组
- Alt、Alt-F1：按键或组合键；这些键以大写形式显示，如在键盘上一样
- 文件、文件 › 另存为：菜单项，按钮
- 跳舞的企鹅（企鹅一章，↑其他手册）：此内容参见自其他手册中的一章。

4 关于本手册的制作

本书用 GeekoDoc（DocBook 的子集，请参见 http://www.docbook.org ）编写。XML 源文件采用 `xmllint` 校验、经过 `xsltproc` 处理，并使用 Norman Walsh 的样式表的自定义版本转换为 XSL-FO。最终的 PDF 可通过 Apache 开发的 FOP 或 RenderX 开发的 XEP 编排格式。包 `daps` 中提供了用于制作此手册的创作和发布工具。DocBook Authoring and Publishing Suite (DAPS) 以开源软件的形式开发。有关详细信息，请参见 http://daps.sf.net/ 。

5 Ceph Contributors

The Ceph project and its documentation is a result of hundreds of contributors and organizations. See https://ceph.com/contributors/ for more details.

I SUSE Enterprise Storage

1 SUSE Enterprise Storage 和 Ceph 2

2 硬件要求和建议 10

3 Ceph 管理节点 HA 设置 20

1 SUSE Enterprise Storage 和 Ceph

SUSE Enterprise Storage 是基于 Ceph 技术的分布式存储系统，旨在提高可伸缩性、可靠性和性能。Ceph 集群可在常见网络（例如以太网）中的市售服务器上运行。该集群能够正常扩展到数千台服务器（后面称为“节点”）以及 PB 量级的处理能力。与使用分配表来存储和提取数据的传统系统相反，Ceph 使用确定性算法来为数据分配存储空间，并且不采用任何集中式信息结构。Ceph 假设在存储集群中，硬件的添加或删除属于惯例，而不是例外情况。Ceph 集群可将数据分布和重新分布、数据复制、故障检测和恢复等管理任务自动化。Ceph 既可自我修复，又可自我管理，因此可以降低管理开销和预算开销。

本章提供 SUSE Enterprise Storage 的综合概述，并简要介绍一些最重要的组件。

提示

从 SUSE Enterprise Storage 5 开始，DeepSea 是唯一的集群部署方法。有关部署过程的详细信息，请参见第 4 章 “使用 DeepSea/Salt 部署”。

1.1 Ceph 的特性

Ceph 环境具有以下特性：

可伸缩性

Ceph 可扩展到数千个节点，并可管理 PB 量级的存储。

市售硬件

无需特殊的硬件即可运行 Ceph 集群。有关详细信息，请参见第 2 章 “硬件要求和建议”

自我管理

Ceph 集群可自我管理。添加、删除节点或节点发生故障时，集群可自动重新分布数据。此外，它还能识别过载的磁盘。

无单一故障点

重要的信息不会单独存储在集群中的某个节点上。可以配置冗余数量。

开放源代码软件

Ceph 是一套开源软件解决方案，独立于特定的硬件或供应商。

1.2 核心组件

要充分利用 Ceph 的强大功能，需要了解一些基本的组件和概念。本节介绍经常在其他章节中提到的 Ceph 的某些组成部分。

1.2.1 RADOS

Ceph 的基本组件称为 RADOS（可靠自主分布式对象存储）。该组件负责管理集群中存储的数据。Ceph 中的数据通常以对象的形式存储。每个对象由标识符和数据组成。

RADOS 提供以下方法来访问涉及许多用例的存储对象：

对象网关
: 对象网关是 RADOS 对象存储区的 HTTP REST 网关。使用该网关可以直接访问 Ceph 集群中存储的对象。

RADOS 块设备
: 可以像访问任何其他块设备一样访问 RADOS 块设备 (RBD)。例如，可将这些设备与 `libvirt` 结合使用，以实现虚拟化。

CephFS
: Ceph 文件系统是符合 POSIX 标准的文件系统。

`librados`
: `librados` 是一个库，可与许多编程语言结合使用，以创建能够直接与存储集群交互的应用。

`librados` 由对象网关和 RBD 使用，而 CephFS 直接与 RADOS 连接。请参见图 1.1 “与 Ceph 对象存储区连接”。

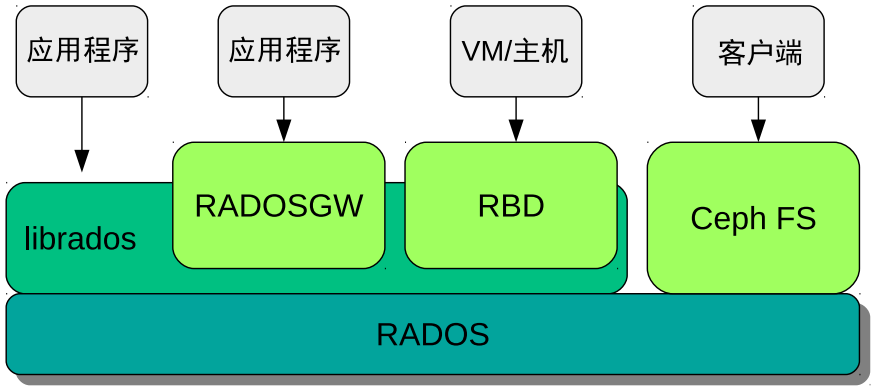

图 1.1： 与 CEPH 对象存储区连接

1.2.2 CRUSH

Ceph 集群的核心是 CRUSH 算法。CRUSH 是 Controlled Replication Under Scalable Hashing（基于可缩放哈希的受控复制）的缩写。CRUSH 是处理存储分配的函数，所需的参数相对较少。这意味着，只需提供少量的信息就能计算对象的存储位置。参数是集群的现行对照，包括运行状况、管理员定义的某些放置规则，以及需要存储或检索的对象名称。提供这些信息后，Ceph 集群中的所有节点即可计算对象及其副本的存储位置。这使数据写入或读取变得非常有效。CRUSH 会尝试在集群中的所有节点上均匀分布数据。

CRUSH 地图包含所有存储节点，以及管理员定义的有关在集群中存储对象的放置规则。该地图定义了通常对应于集群物理结构的分层结构。例如，包含数据的磁盘位于主机中，主机位于机柜中，机柜位于设备排中，而设备排位于数据中心中。此结构可用于定义故障域。然后，Ceph 可确保将复制项存储在特定故障域的不同分支中。

如果将故障域设置为机柜，则在不同的机柜中分布对象复制项。这样就可以缓解机柜中的交换机发生故障所造成的服务中断。如果某个电源分配单元为一排机柜供电，则可将故障域设置为设备排。当电源分配单元发生故障时，其他设备排中仍可提供复制的数据。

1.2.3 Ceph 节点和守护进程

在 Ceph 中，节点是为集群工作的服务器。它们可以运行多种不同类型的守护进程。建议在每个节点上只运行一种类型的守护进程，但 MGR 守护进程除外，此类守护进程可与 MON 共置。每个集群至少需要有 MON、MGR 和 OSD 守护进程：

Ceph Monitor

Ceph Monitor（往往缩写为 MON）节点负责维护有关集群运行状况、所有节点的地图和数据分布规则的信息（请参见第 1.2.2 节 “CRUSH”）。

如果发生故障或冲突，集群中的 Ceph Monitor 节点会根据多数派原则确定哪些信息是正确的。为了构成有效的多数派，建议设置奇数数量的 Ceph Monitor 节点，并至少设置三个这样的节点。

如果使用多个站点，应在奇数个站点之间分布 Ceph Monitor 节点。每个站点的 Ceph Monitor 节点数应满足以下要求：当一个站点发生故障时，50% 以上的 Ceph Monitor 节点可保持正常运行。

Ceph Manager

Ceph Manager (MGR) 会从整个集群收集状态信息。Ceph Manager 守护进程与监视器守护进程一同运行。它提供附加的监视功能，并与外部监视和管理系统连接。

Ceph Manager 不需要额外的配置，只需确保它在运行即可。您可以使用 DeepSea 将它部署为独立的角色。

Ceph OSD

Ceph OSD 是一个守护进程，负责处理属于物理或逻辑存储单元（硬盘或分区）的对象存储设备。对象存储设备可以是物理磁盘/分区，也可以是逻辑卷。此外，该守护进程会处理数据复制，并在添加或删除节点后进行重新平衡。

Ceph OSD 守护进程与监视器守护进程通讯，并为其提供其他 OSD 守护进程的状态。

要使用 CephFS、对象网关、NFS Ganesha 或 iSCSI 网关，还需要其他节点：

元数据服务器 (MDS)

元数据服务器存储 CephFS 的元数据。使用 MDS 可以执行 `ls` 等基本文件系统命令，而不会让集群过载。

对象网关

对象网关提供的 Ceph Object Gateway 是 RADOS 对象存储区的 HTTP REST 网关。它与 OpenStack Swift 和 Amazon S3 兼容，具有自己的用户管理功能。

NFS Ganesha

使用 NFS Ganesha 可通过 NFS 访问对象网关或 CephFS。该组件在用户空间而不是内核空间中运行，直接与对象网关或 CephFS 交互。

iSCSI 网关

iSCSI 是一种存储网络协议，可让客户端将 SCSI 命令发送到远程服务器上的 SCSI 存储设备（目标）。

1.3 存储结构

1.3.1 存储池

Ceph 集群中存储的对象放置在存储池中。对外部环境而言，存储池代表集群的逻辑分区。对于每个存储池，可以定义一组规则，例如，每个对象必须有多少个复制项。存储池的标准配置称为副本池。

存储池通常包含多个对象，但也可以将其配置为充当类似 RAID 5 的作用。在此配置中，对象连同其他编码块一起存储在块中。编码块包含冗余信息。管理员可以定义数据块和编码块的数量。在此配置中，存储池称为纠删码池。

1.3.2 归置组

归置组 (PG) 用于在存储池中分布数据。创建存储池时，会设置特定数目的归置组。归置组在内部用于将对象分组，是影响 Ceph 集群性能的重要因素。对象的 PG 根据该对象的名称确定。

1.3.3 示例

本节提供有关 Ceph 如何管理数据的简化示例（请参见图 1.2 “小规模 Ceph 示例”）。此示例并不代表 Ceph 集群的建议配置。硬件设置由三个存储节点或 Ceph OSD（`主机 1`、`主机 2` 和 `主机 3`）组成。每个节点包含三个用作 OSD（`osd.0` 到 `osd.9`）的硬盘。此示例中忽略了 Ceph Monitor 节点。

注意：Ceph OSD 与 OSD 之间的差别

尽管 Ceph OSD 或 Ceph OSD 守护进程是指节点上运行的守护进程，但 OSD 一词指的是与守护进程交互的逻辑磁盘。

集群包含两个存储池：存储池 A 和存储池 B。尽管存储池 A 仅复制对象两次，但存储池 B 的恢复能力更重要，该存储池中的每个对象都有三个复制项。

当应用将某个对象放入存储池中时（例如，通过 REST API），将会根据存储池和对象名称选择归置组（PG1 到 PG4）。然后，CRUSH 算法会根据包含对象的归置组，计算要将对象存储到哪些 OSD。

在此示例中，故障域设置为主机。这可确保将对象的复制项存储在不同的主机上。根据针对存储池设置的复制级别，对象将存储在归置组使用的两个或三个 OSD 上。

写入对象的应用只与一个 Ceph OSD（主要 Ceph OSD）交互。主要 Ceph OSD 处理复制过程，并在所有其他 OSD 存储对象后确认写入过程完成。

如果 osd.5 发生故障，osd.1 上仍会提供 PG1 中的所有对象。一旦集群发现某个 OSD 发生故障，另一个 OSD 会立即接管工作。在此示例中，osd.4 用于取代 osd.5。然后，osd.1 上存储的对象会复制到 osd.4，以恢复复制级别。

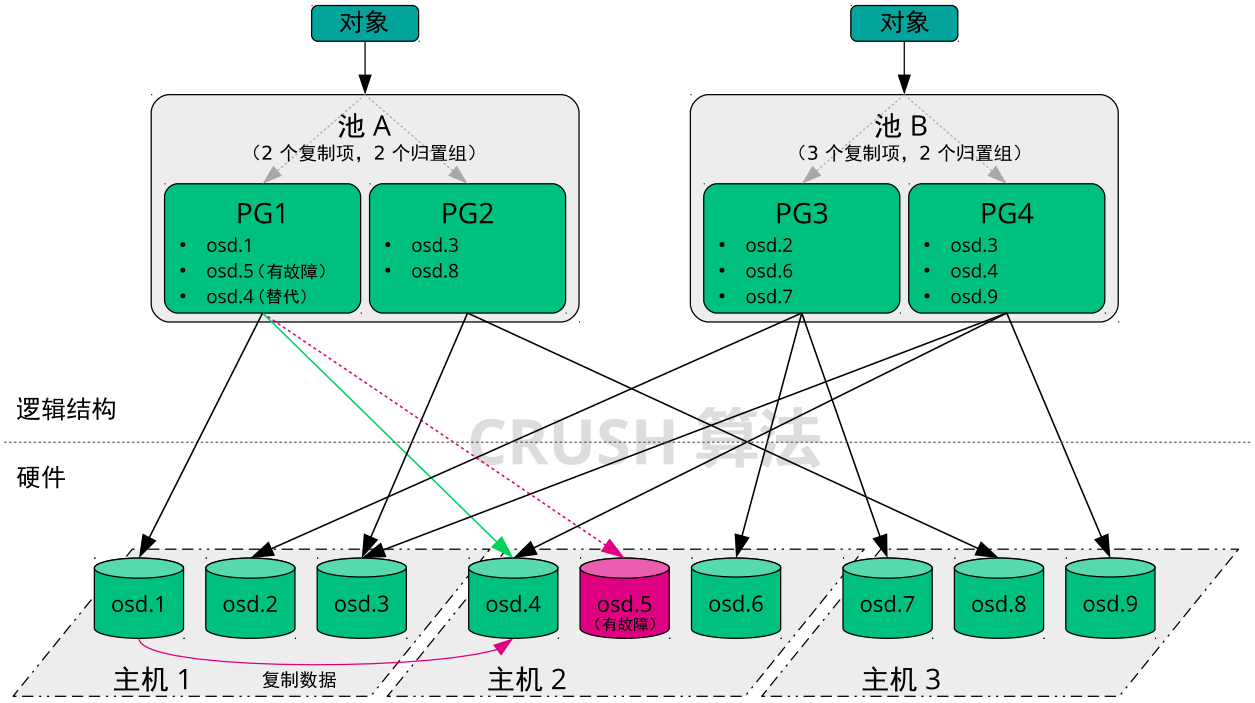

图 1.2： 小规模 CEPH 示例

如果将包含新 OSD 的新节点添加到集群，集群地图将会更改。然后，CRUSH 函数将返回对象的不同位置。接收新位置的对象将被重新定位。此过程将使得所有 OSD 被平衡使用。

1.4 BlueStore

从 SUSE Enterprise Storage 5 开始，使用 BlueStore 作为 Ceph 的新默认存储后端。BlueStore 的性能优于 FileStore，具有全数据校验和及内置压缩功能。

它可管理一至三个存储设备。在最简单的情况下，BlueStore 会使用一个主存储设备。该设备通常被分割成两个分区：

1. 一个名为 BlueFS 的较小分区，用于实施 RocksDB 所需的类似文件系统的功能。
2. 其余部分通常是一个较大的分区，占用了设备的剩余空间。它直接由 BlueStore 管理，包含所有实际数据。此主设备通常通过数据目录中的块符号链接来标识。

还可跨两个额外的设备来部署 BlueStore：

WAL 设备可用于存储 BlueStore 的内部日记或预写日志。它通过数据目录中的 `block.wal` 符号链接来标识。只有 WAL 设备速度比主设备或 DB 设备快时，使用单独的 WAL 设备才比较实用，例如，下列情况就很适合：

- WAL 设备是 NVMe，DB 设备是 SSD，数据设备是 SSD 或 HDD。
- WAL 和 DB 设备都是单独的 SSD，数据设备是 SSD 或 HDD。

DB 设备可用于存储 BlueStore 的内部元数据。BlueStore（更确切地说，是嵌入式 RocksDB）会将尽可能多的元数据存放于 DB 设备，以提升性能。再次重申，只有共享 DB 设备速度比主设备快时，供应共享 DB 设备才有所助益。

提示：规划 DB 大小

规划时考虑充分，以便为 DB 设备分配足够的大小。如果 DB 设备填满，元数据将溢出到主设备，这会严重降低 OSD 的性能。

您可以使用 `ceph daemon osd.ID perf dump` 命令来检查 WAL/DB 分区是否即将填满及溢出。`slow_used_bytes` 值显示即将溢出的数据量：

```
root@minion > ceph daemon osd.ID perf dump | jq '.bluefs'
"db_total_bytes": 1073741824,
"db_used_bytes": 33554432,
"wal_total_bytes": 0,
"wal_used_bytes": 0,
```

```
"slow_total_bytes": 554432,
"slow_used_bytes": 554432,
```

1.5 其他信息

- 作为一个社区项目，Ceph 自身具有丰富的联机文档。对于本手册中未介绍的主题，请参见 http://docs.ceph.com/docs/master/ 。
- 由 S.A. Weil、S.A. Brandt、E.L. Miller 和 C. Maltzahn 撰写的原始文献 CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data（CRUSH：复制数据的受控、可缩放、分布式放置）提供了有关 Ceph 内部工作原理的有用见解。特别是在部署大规模集群时，推荐您阅读此文章。可在 http://www.ssrc.ucsc.edu/papers/weil-sc06.pdf 上找到该文献。

2 硬件要求和建议

Ceph 的硬件要求在很大程度上取决于 IO 工作负载。在着手进行详细规划时，应考虑以下硬件要求和建议。

一般情况下，本节所述的建议是按进程提出的。如果同一台计算机上有多个进程，则需要提高 CPU、RAM、磁盘和网络要求。

2.1 对象存储节点

2.1.1 最低要求

- 至少需要 4 个 OSD 节点，每个节点包含 8 个 OSD 磁盘。
- 对于不使用 BlueStore 的 OSD 而言，在每个 OSD 存储节点上，至少需要为每 TB 的原始 OSD 容量提供 1 GB RAM。建议为每 TB 原始 OSD 容量提供 1.5 GB RAM。在恢复期间，为每 TB 原始 OSD 容量提供 2 GB RAM 可能会达到最佳效果。
  对于使用 BlueStore 的 OSD 而言，首先要计算出针对不使用 BlueStore 的 OSD 建议的 RAM 大小，然后计算 2 GB 加上每个 OSD 进程建议的 RAM 的 BlueStore 快速缓存大小，并在这两个结果中选择 RAM 值较大的那一个。请注意，对于 HDD，默认的 BlueStore 快速缓存为 1 GB；而对于 SSD 设备，默认则为 3 GB。总之，请从下面两个值中选择较大的那个：

  ```
  [1GB * OSD count * OSD size]
  ```

  或者

  ```
  [(2 + BS cache) * OSD count]
  ```

- 对于每个 OSD 守护进程进程，至少需要为每个 OSD 提供 1.5 GHz 的逻辑 CPU 内核。建议为每个 OSD 守护进程进程提供 2 GHz 内核。请注意，Ceph 为每个存储磁盘运行一个 OSD 守护进程进程；请不要计入专门用作 OSD 日记、WAL 日记、omap 元数据或此三者任意组合形式的保留磁盘空间。

- 10 Gb 以太网（与多个交换机绑定的两个网络接口）。
- 采用 JBOD 配置的 OSD 磁盘。
- OSD 磁盘应该专门由 SUSE Enterprise Storage 使用。
- 操作系统专用的磁盘/SSD，最好采用 RAID 1 配置。
- 如果此 OSD 主机将要托管用于快速缓存分层的一部分快速缓存池，请至少额外分配 4 GB RAM。
- 出于磁盘性能原因，OSD 节点应是未虚拟化的裸机。

2.1.2 最小磁盘大小

需要在 OSD 上运行以下两种类型的磁盘空间：磁盘日记（针对 FileStore）或 WAL/DB 设备（针对 BlueStore）的空间以及存储数据的主空间。日记/WAL/DB 的最小（默认）值为 6 GB。数据的最小空间为 5 GB，因为系统会自动为小于 5 GB 的分区指定权重 0。

因此，尽管 OSD 的最小磁盘空间为 11 GB，但不建议使用小于 20 GB 的磁盘，即使在测试中也是如此。

2.1.3 BlueStore 的 WAL 和 DB 设备的建议大小

提示：更多信息

有关 BlueStore 的详细信息，请参见第 1.4 节 “BlueStore”。

下面是针对 WAL/DB 设备大小调整的几条规则。使用 DeepSea 部署 OSD 和 BlueStore 时，它会自动应用建议的规则，并向管理员告知实情。

- 每 TB 的 OSD 容量为 10GB 的 DB 设备（OSD 的 1/100）。
- WAL 设备介于 500MB 到 2GB 之间。WAL 大小取决于数据流量和工作负载，而不是取决于 OSD 大小。如果您知道 OSD 实际上能在吞吐量极高时处理小型写入和覆盖，则 WAL 最好较大而不是较小。1GB WAL 设备是一个较好的折衷方案，可满足大多数部署要求。

- 如果您打算将 WAL 和 DB 设备置于同一磁盘，建议您为这两个设备使用一个分区，而不是为每个设备使用单独的分区。这样，Ceph 便可以使用 DB 设备来执行 WAL 操作。这对于磁盘空间的管理也会更有效，因为 Ceph 只会在需要时才会为 WAL 使用 DB 分区。另一个好处是 WAL 分区填满的可能性很小，当该分区未完全占用时，转而用于 DB 操作不会浪费空间。

  要与 WAL 共享 DB 设备，请不要指定 WAL 设备，而是仅指定 DB 设备：

```
bluestore_block_db_path = "/path/to/db/device"
bluestore_block_db_size = 10737418240
bluestore_block_wal_path = ""
bluestore_block_wal_size = 0
```

- 您也可以将 WAL 置于其自己的独立设备上。在这种情况下，建议针对 WAL 操作使用最快的设备。

2.1.4 使用 SSD 存储 OSD 日记

固态硬盘 (SSD) 不包含移动部件。这可以减少随机访问时间和读取延迟，同时加快数据吞吐量。由于 SSD 的每 MB 价格大大高于旋转型硬盘，SSD 只适用于较小规模的存储。

如果将日记存储在 SSD 上，并将对象数据存储在独立的硬盘上，OSD 的性能会得到大幅提高。

提示：在一个 SSD 中共享多个日记

由于日记数据占用的空间相对较小，因此您可以将多个日记目录装入单个 SSD 磁盘。请注意，每共享一个日记，SSD 磁盘的性能就会有所下降。不建议在同一个 SSD 磁盘中共享 6 个以上的日记，或者在 NVMe 磁盘中共享 12 个以上的日记。

2.1.5 磁盘的最大建议数量

您可以在一台服务器上使用所允许的任意数量的磁盘。规划每台服务器的磁盘数量时，需要考虑以下几点：

- 网络带宽：在一台服务器中使用的磁盘越多，执行磁盘写入操作时必须通过网卡传输的数据就越多。
- 内存：为获得最佳性能，请每安装 1 TB 磁盘空间至少保留 2 GB RAM。
- 容错：整台服务器发生故障时，该服务器包含的磁盘越多，则集群暂时丢失的 OSD 就越多。此外，为了确保复制规则的运行，需要将有故障服务器中的所有数据复制到集群中的其他节点。

2.2 监视器节点

- 至少需要三个 Ceph Monitor 节点。监视器数量应始终为奇数 (1+2n)。
- 4 GB RAM。
- 有四个逻辑内核的处理器。
- 强烈建议对监视器使用 SSD 或其他速度足够快的存储类型，特别是针对每个监视器节点上的 `/var/lib/ceph` 路径，因为仲裁可能不稳定且磁盘延迟较高。建议提供两个采用 RAID 1 配置的磁盘来实现冗余。建议对监视器进程使用独立的磁盘，或者至少是独立的磁盘分区，以防止日志文件缓增等问题导致监视器的可用磁盘空间不足。
- 每个节点只能有一个监视器进程。
- 仅当有足够的硬件资源可用时，才支持混用 OSD、监视器或对象网关节点。这意味着，对于所有服务需要提高相应要求。
- 与多个交换机绑定的两个网络接口。

2.3 对象网关节点

对象网关节点应有 6 到 8 个 CPU 内核和 32 GB RAM（建议 64 GB）。如果将其他进程共置在同一台计算机上，则需要提高资源的要求。

2.4 元数据服务器节点

元数据服务器节点的适当大小取决于特定用例。一般而言，元数据服务器需要处理的打开文件越多，所需要的 CPU 和 RAM 就越多。以下是最低要求：

- 每个元数据服务器守护进程需要 3G RAM。
- 绑定网络接口。
- 2.5 GHz CPU，至少有两个内核。

2.5 Salt Master

至少需要 4 GB RAM 和四核 CPU。这包括在 Salt Master 上运行 openATTIC 的要求。对于包含数百个节点的大型集群，建议提供 6 GB RAM。

2.6 iSCSI 节点

iSCSI 节点应有 6 到 8 个 CPU 内核和 16 GB RAM。

2.7 网络建议

要运行 Ceph 的网络环境最好是至少包含两个网络接口的绑定组合，该组合使用 VLAN 逻辑分割为公共部分和可信的内部部分。如果可能，建议采用 802.3ad 绑定模式，以提供最高的带宽和恢复能力。

公共 VLAN 用于向客户提供服务，而内部部分则用于提供经身份验证的 Ceph 网络通讯。建议采用此模式的主要原因在于，尽管 Ceph 可提供身份验证并在创建机密密钥后防范攻击，但用于配置这些密钥的讯息可能会公开传输，因而容易受到攻击。

提示：通过 DHCP 配置的节点

如果存储节点是通过 DHCP 配置的，则默认超时可能会不够长，无法保证在各个 Ceph 守护进程启动前正确配置网络。如果发生此问题，Ceph MON 和 OSD 将不会正常启动（运行 `systemctl status ceph\*` 会导致“无法绑定”错误）。为避免此问题发生，建议在存储集群的每个节点上，将 DHCP 客户端超时增加到至少 30 秒。为此，可在每个节点上更改以下设置：

在 `/etc/sysconfig/network/dhcp` 中，设置

```
DHCLIENT_WAIT_AT_BOOT="30"
```

在 `/etc/sysconfig/network/config` 中，设置

```
WAIT_FOR_INTERFACES="60"
```

2.7.1 将专用网添加到正在运行的集群

如果您在部署 Ceph 期间未指定集群网络，则系统假设使用的是单个公共网络环境。尽管 Ceph 可在公共网络中正常运行，但如果您设置了另一个专用集群网络，Ceph 的性能和安全性将会得到提升。要支持两个网络，每个 Ceph 节点上至少需有两个网卡。

需要对每个 Ceph 节点应用以下更改。对小型集群执行此操作的速度相对较快，但如果集群包含数百甚至数千个节点，则此过程可能十分耗时。

1. 在每个集群节点上停止 Ceph 相关的服务。
   在 `/etc/ceph/ceph.conf` 中添加一行以定义集群网络，例如：

   ```
   cluster network = 10.0.0.0/24
   ```

   如果需要指定具体的静态 IP 地址或覆盖 `cluster network` 设置，可以使用可选的 `cluster addr` 实现此目的。
2. 检查专用集群网络是否在 OS 级别按预期工作。
3. 在每个集群节点上启动 Ceph 相关的服务。

```
sudo systemctl start ceph.target
```

2.7.2 不同子网中的监视器节点

如果监视器节点位于多个子网中，例如，位于不同的机房并由不同的交换机提供服务，则您需要相应地调整 `ceph.conf` 文件。例如，如果节点的 IP 地址为 192.168.123.12、1.2.3.4 和 242.12.33.12，请将以下几行添加到 global 段落：

```
[global]
[...]
mon host = 192.168.123.12, 1.2.3.4, 242.12.33.12
mon initial members = MON1, MON2, MON3
[...]
```

此外，如果您需要指定每个监视器的公用地址或网络，则需要为每个监视器添加 `[mon.X]` 段落：

```
[mon.MON1]
public network = 192.168.123.0/24

[mon.MON2]
public network = 1.2.3.0/24

[mon.MON3]
public network = 242.12.33.12/0
```

2.8 命名限制

一般情况下，Ceph 不支持在配置文件、存储池名称、用户名等内容中使用非 ASCII 字符。配置 Ceph 集群时，建议在所有 Ceph 对象/配置名称中仅使用简单的字母数字字符（A-Z、a-z、0-9）和最少量的标点符号（“.”、“-”、“_”）。

2.9 共享一台服务器的 OSD 和监视器

尽管从技术上讲可在测试环境中的同一台服务器上运行 Ceph OSD 和监视器，但强烈建议在生产环境中为每个监视器节点使用独立的服务器。主要原因在于性能 — 集群包含的 OSD 越多，监视器节点需要执行的 I/O 操作就越多。另外，在监视器节点与 OSD 之间共享一台服务器时，OSD I/O 操作将会成为监视器节点的限制因素。

另一个考虑要点是，是否要在 OSD、监视器节点与服务器上的操作系统之间共享磁盘。答案非常简单：如果可能，请将一个独立的磁盘专门用于 OSD，并将一台独立的服务器用于监视器节点。

尽管 Ceph 支持基于目录的 OSD，但 OSD 应始终包含一个专用磁盘，而不能与操作系统共享一个磁盘。

提示

如果确实有必要在同一台服务器上运行 OSD 和监视器节点，请将一个独立磁盘装入 `/var/lib/ceph/mon` 目录以在该磁盘上运行监视器，这样可以稍微改善性能。

2.10 最低集群配置

- 四个对象存储节点
  - 10 Gb 以太网（与多个交换机绑定的两个网络）
  - 每个存储集群有 32 个 OSD
  - OSD 日记可以驻留在 OSD 磁盘上
  - 每个对象存储节点有专用的 OS 磁盘
  - 在每个对象存储节点上，为每 TB 的原始 OSD 容量提供 1 GB RAM
  - 为每个对象存储节点上的每个 OSD 提供 1.5 GHz
  - Ceph Monitor、网关和元数据服务器可以驻留在对象存储节点上

- 三个 Ceph Monitor 节点（需要使用 SSD 作为专用 OS 驱动器）
- Ceph Monitor、对象网关和元数据服务器节点需要冗余部署
- iSCSI 网关、对象网关和元数据服务器需要额外的 4 GB RAM 和四个内核

- 具有 4 GB RAM、四个内核和 1 TB 容量的独立管理节点

2.11 建议的生产集群配置

- 七个对象存储节点
  - 单个节点不超过总容量的 15% 左右
  - 10 Gb 以太网（与多个交换机绑定的四个物理网络）
  - 每个存储集群有 56 个以上的 OSD
  - 每个 OSD 存储节点包含 RAID 1 OS 磁盘
  - 根据 6:1 的 SSD 日记与 OSD 的比率为日记提供 SSD
  - 在每个对象存储节点上，为每 TB 的原始 OSD 容量提供 1.5 GB RAM
  - 为每个对象存储节点上的每个 OSD 提供 2 GHz
- 专用的物理基础架构节点
  - 三个 Ceph Monitor 节点：4 GB RAM，四核处理器，RAID 1 SSD 磁盘
  - 一个 SES 管理节点：4 GB RAM，四核处理器，RAID 1 SSD 磁盘
  - 网关或元数据服务器节点的冗余物理部署：
    - 对象网关节点：32 GB RAM，八核处理器，RAID 1 SSD 磁盘
    - iSCSI 网关节点：16 GB RAM，四核处理器，RAID 1 SSD 磁盘
    - 元数据服务器节点（一个主动/一个热待机）：32 GB RAM，八核处理器，RAID 1 SSD 磁盘

2.12 SUSE Enterprise Storage 以及其他 SUSE 产品

本节包含有关将 SUSE Enterprise Storage 与其他 SUSE 产品集成的重要信息。

2.12.1 SUSE Manager

SUSE Manager 与 SUSE Enterprise Storage 未集成，因此 SUSE Manager 当前无法管理 SUSE Enterprise Storage 集群。

3 Ceph 管理节点 HA 设置

Ceph 管理节点是一种其上运行有 Salt Master 服务的 Ceph 集群节点。管理节点是 Ceph 集群的中心点，因为它通过查询其他群集节点的 Salt Minion 服务并提供指示，来管理其余的集群节点。它通常也会包含其他服务，例如 openATTIC Web UI，由 Prometheus 监视工具包对 Grafana 仪表盘提供支持。

如果 Ceph 管理节点发生故障，通常需要为该节点提供新的工作硬件，并通过最近的备份恢复完整的集群配置堆栈。此类方法很费时，并会导致集群故障。

为防止出现由于管理节点故障导致的 Ceph 集群性能下降，建议您为 Ceph 管理节点使用高可用性 (HA) 集群。

3.1 Ceph 管理节点的 HA 集群概述

HA 集群的原理是当其中一个集群节点发生故障时，由另一个节点自动接管其角色，包括虚拟化 Ceph 管理节点。使用此方法时，其他 Ceph 集群节点不会注意到 Ceph 管理节点发生故障。

Ceph 管理节点的最低 HA 解决方案需要以下硬件：

- 能够运行具有高可用性扩展的 SUSE Linux Enterprise 以及虚拟化 Ceph 管理节点的两台裸机服务器。
- 两个或多个冗余网络通讯路径，例如通过网络设备绑定。
- 用于托管 Ceph 管理节点虚拟机的磁盘映像的共享存储。需要可通过这两台服务器访问的共享存储。例如，该共享存储可以是 NFS 导出项、Samba 共享或 iSCSI 目标。

有关集群要求的更多详细信息，请访问 https://www.suse.com/documentation/sle-ha-12/install-quick/data/install-quick.html#sec_ha_inst_quick_req 。

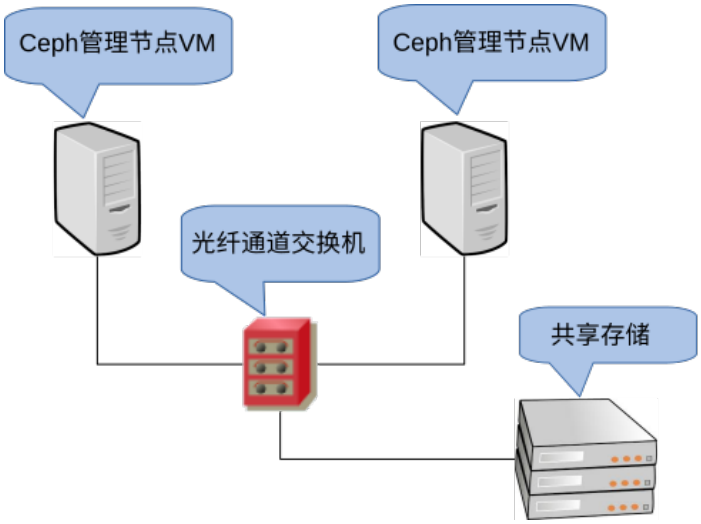

图 3.1： CEPH 管理节点的双节点 HA 集群

3.2 构建用于 Ceph 管理节点的 HA 集群

以下过程汇总了为虚拟化 Ceph 管理节点而构建 HA 集群的几个最重要步骤。有关详细信息，请参见指定链接。

1. 设置一个具有共享存储的基本双节点 HA 集群，如 https://www.suse.com/documentation/sle-ha-12/install-quick/data/install-quick.html 中所述。
2. 在两个集群节点上，安装运行 KVM 超级管理程序和 `libvirt` 工具包所需的所有程序包，如 https://www.suse.com/documentation/sles-12/book_virt/data/sec_vt_installation_kvm.html 中所述。
3. 在第一个集群节点上，使用 `libvirt` 创建新的 KVM 虚拟机 (VM)，如 https://www.suse.com/documentation/sles-12/book_virt/data/sec_libvirt_inst_vmm.html 中所述。使用预配置的共享存储来存储 VM 的磁盘映像。
4. VM 设置完成后，将其配置导出到共享存储上的 XML 文件。使用以下语法：

```
root # virsh dumpxml VM_NAME > /path/to/shared/vm_name.xml
```

5. 为 Ceph 管理节点 VM 创建资源。有关创建 HA 资源的一般信息，请参见 https://www.suse.com/documentation/sle-ha-12/book_sleha/data/cha_conf_hawk2.html 。http://www.linux-ha.org/wiki/VirtualDomain_%28resource_agent%29 中提供了有关为 KVM 虚拟机创建资源的详细信息。

6. 在新创建的 VM guest 中，部署 Ceph 管理节点，包括您需要在其上使用的其他服务。遵循第 4.3 节 “集群部署”中的相关步骤。同时，在非 HA 集群服务器上部署其余 Ceph 集群节点。

II 集群部署和升级

4 使用 DeepSea/Salt 部署 24

5 从旧版本升级 47

6 备份集群配置 73

7 自定义默认配置 74

4 使用 DeepSea/Salt 部署

注意：SUSE Enterprise Storage 5 中已删除 `ceph-deploy`

SUSE Enterprise Storage 4 中已弃用 `ceph-deploy` 集群部署工具。从 SUSE Enterprise Storage 5 开始，随着 DeepSea 的推出，此工具已完全删除。

Salt 连同 DeepSea 是一个组件堆栈，可帮助您部署和管理服务器基础架构。Salt 具有很高的可缩放性，速度快，且相对容易运行。在开始使用 Salt 部署集群之前，请阅读以下注意事项：

- Salt Minion 是由一个称为 Salt Master 的专用节点控制的节点。Salt Minion 具有角色，例如 Ceph OSD、Ceph Monitor、Ceph Manager、对象网关、iSCSI 网关或 NFS Ganesha。
- Salt Master 运行自己的 Salt Minion。运行特权任务（例如，创建、授权密钥以及将密钥复制到 Minion）需要 Salt Master，这样，远程 Minion 就永远不需要运行特权任务。

  

  #### 提示：每台服务器共享多个角色

  如果将每个角色都部署在一个独立节点上，则 Ceph 集群的性能是最佳的。但实际部署有时会要求多个角色共享一个节点。为避免性能欠佳以及升级过程出现问题，请勿向 Salt Master 部署 Ceph OSD、元数据服务器或 Ceph Monitor 角色。

- Salt Minion 需要能通过网络正确解析 Salt Master 的主机名。默认情况下，Minion 会查找 `salt` 主机名，但您可以在 `/etc/salt/minion` 文件中指定可通过网络访问的其他任何主机名，具体请参见第 4.3 节 “集群部署”。

4.1 阅读发行说明

在发行说明中，可以找到有关自 SUSE Enterprise Storage 的上一个版本发行后所进行的更改的其他信息。检查发行说明以了解：

- 您的硬件是否有特殊的注意事项。
- 所用的任何软件包是否已发生重大更改。
- 是否需要对您的安装实施特殊预防措施。

发行说明还提供未能及时编入手册中的信息。它们还包含有关已知问题的说明。

安装包 `release-notes-ses` 之后，可在本地的 `/usr/share/doc/release-notes` 目录中或 https://www.suse.com/releasenotes/ 网页上找到发行说明。

4.2 DeepSea 简介

DeepSea 旨在节省管理员的时间，让他们自信地对 Ceph 集群执行复杂操作。

Ceph 是一款高度可配置的软件解决方案。它提高了系统管理员的自由度和职责履行能力。

最低的 Ceph 设置能够很好地满足演示目的，但无法展示 Ceph 在处理大量节点时可体现的卓越功能。

DeepSea 会收集并存储有关单台服务器的相关数据，例如地址和设备名称。对于诸如 Ceph 的分布式存储系统，可能需要收集并存储数百个这样的项目。收集信息并手动将数据输入到配置管理工具的过程非常耗费精力，并且容易出错。

准备服务器、收集配置信息以及配置和部署 Ceph 所需执行的步骤大致相同。但是，这种做法无法解决管理独立功能的需求。在日常操作中，必须做到不厌其烦地将硬件添加到给定的功能，以及适当地删除硬件。

DeepSea 通过以下策略解决了这些需求：DeepSea 可将管理员的多项决策合并到单个文件中。这些决策包括集群指定、角色指定和配置指定。此外，DeepSea 还会收集各组任务以组成一个简单的目标。每个目标就是一个阶段：

DEEPSEA 阶段说明

- 阶段 0 — 准备：在此阶段，将应用全部所需的更新，并且可能会重引导您的系统。

  重要：重引导 Salt Master 后重新运行阶段 0

  如果在执行阶段 0 期间，Salt Master 重引导以加载新内核版本，则您需要再次运行阶段 0，否则将无法定位 Minion。

- 阶段 1 — 发现：在此阶段，将检测集群中的所有硬件，并收集 Ceph 配置所需的信息。有关配置的详细信息，请参见第 4.5 节 “配置和自定义”。

- 阶段 2 — 配置：您需要以特定的格式准备配置数据。

- 阶段 3 — 部署：创建包含必要 Ceph 服务的基本 Ceph 集群。有关必要服务的列表，请参见第 1.2.3 节 “Ceph 节点和守护进程”。
- 阶段 4 — 服务：可在此阶段安装 Ceph 的其他功能，例如 iSCSI、对象网关和 CephFS。其中每个功能都是可选的。
- 阶段 5 — 删除阶段：此阶段不是必需的，在初始设置期间，通常不需要此阶段。在此阶段，将会删除 Minion 的角色以及集群配置。如果您需要从集群中删除某个存储节点，则需要运行此阶段。有关详细信息，请参见《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.3 节 “删除和重新安装集群节点”。

有关 DeepSea 的更详细介绍，请访问 https://github.com/suse/deepsea/wiki 。

4.2.1 组织和重要位置

Salt 在 Master 节点上使用多个标准位置和多个命名约定：

`/srv/pillar`

该目录存储集群 Minion 的配置数据。Pillar 是向所有集群 Minion 提供全局配置值的接口。

`/srv/salt/`

该目录存储 Salt 状态文件（也称为 sls 文件）。状态文件是集群应该所处的状态的带格式说明。有关详细信息，请参见 Salt 文档 (https://docs.saltstack.com/en/latest/topics/tutorials/starting_states.html) 。

`/srv/module/runners`

该目录存储称作运行程序的 Python 脚本。运行程序在 Master 节点上执行。

`/srv/salt/_modules`

该目录存储称作模块的 Python 脚本。这些模块将应用到集群中的所有 Minion。

`/srv/pillar/ceph`

该目录由 DeepSea 使用。收集的配置数据存储在此处。

`/srv/salt/ceph`

DeepSea 使用的目录。其中存储了可采用不同格式的 sls 文件，但 sls 文件包含在各子目录中。每个子目录仅包含一种类型的 sls 文件。例如，`/srv/salt/ceph/stage` 包含 `salt-run state.orchestrate` 执行的编制文件。

4.2.2 定位 Minion

DeepSea 命令通过 Salt 基础架构执行。使用 `salt` 命令时，您需要指定一组将受到该命令影响的 Salt Minion。我们将该组 Minion 描述为 `salt` 命令的目标。以下各节说明定位 Minion 的可行方法。

4.2.2.1 匹配 Minion 名称

您可以通过名称匹配来定位一个或一组 Minion。Minion 的名称通常为运行该 Minion 的节点的短主机名。这是一种常规的 Salt 定位方法，与 DeepSea 无关。您可以使用通配、正则表达式或列表来限制 Minion 名称的范围。遵循的一般语法如下：

```
root@master # salt target example.module
```

提示：仅 Ceph 集群

如果您环境中的所有 Salt Minion 均属于 Ceph 集群，则可以安全地使用 `'*'` 替换 `target`，以包含所有注册 Minion。

匹配 example.net 域中的所有 Minion（假设 Minion 名称与其“完整”的主机名相同）：

```
root@master # salt '*.example.net' test.ping
```

将“web1”Minion 与“web5”Minion 匹配：

```
root@master # salt 'web[1-5]' test.ping
```

使用正则表达式匹配“web1-prod”和“web1-devel”Minion：

```
root@master # salt -E 'web1-(prod|devel)' test.ping
```

匹配简单的 Minion 列表：

```
root@master # salt -L 'web1,web2,web3' test.ping
```

匹配集群中的所有 Minion：

```
root@master # salt '*' test.ping
```

4.2.2.2 使用“deepsea”Grain 进行定位

在一个异构受 Salt 管理环境（部分节点上部署了 SUSE Enterprise Storage 以及其他集群解决方案）中，建议将“deepsea”Grain 应用到相关 Minion 来加以“标记”。这样您便可以在无法通过 Minion 名称匹配的环境中轻松定位 DeepSea Minion。

要将“deepse”Grain 应用到一组 Minion，请运行以下命令：

```
root@master # salt target grains.append deepsea default
```

要从一组 Minion 删除“deepse”Grain，请运行以下命令：

```
root@master # salt target grains.delval deepsea destructive=True
```

将“deepsea”Grain 应用到相关 Minion 后，您可以执行以下命令来进行定位：

```
root@master # salt -G 'deepsea:*' test.ping
```

或执行以下等效命令：

```
root@master # salt -C 'G@deepsea:*' test.ping
```

4.2.2.3 设置 deepsea_minions 选项

设置 `deepsea_minions` 选项的目标是 DeepSea 部署所需。在阶段执行期间，DeepSea 会使用该选项指示 Minion（有关详细信息，请参见 DeepSea 阶段说明）。

要设置或更改 `deepsea_minions` 选项，请编辑 Salt Master 上的 `/srv/pillar/ceph/deepsea_minions.sls` 文件，添加或替换下行：

```
deepsea_minions: target
```

提示：deepsea_minions 目标

对于 `deepsea_minions` 选项的 `target`，您可以使用以下任何定位方法：匹配 Minion 名称和使用“deepsea”Grain 进行定位。

匹配集群中的所有 Salt Minion：

```
deepsea_minions: '*'
```

使用“deepsea”Grain 匹配所有 Minion：

```
deepsea_minions: 'G@deepsea:*'
```

4.2.2.4 更多信息

您可以使用 Salt 基础架构以更高级的方式来定位 Minion。有关所有定位技术的说明，请参见 https://docs.saltstack.com/en/latest/topics/targeting/ 。

此外，还可从“deepsea-minions”手册页了解有关 DeepSea 定位的更多详细信息（`man 7 deepsea_minions`）。

4.3 集群部署

集群部署过程包括多个阶段。首先，需要通过配置 Salt 来准备集群的所有节点，然后部署并配置 Ceph。

提示：在未定义 OSD 配置的情况下部署监视器节点

如果您需要跳过定义 OSD 配置，而先部署监视器节点，可以通过设置 `DEV_ENV` 变量来实现。该设置允许您在没有 `profile/` 目录的情况下部署监视器，以及部署至少包含一个存储、监视器和管理器节点的集群。

要设置环境变量，请将其全局启用，方法是在 `/srv/pillar/ceph/stack/global.yml` 文件中进行设置，或者仅针对当前的外壳会话设置：

```
root@master # export DEV_ENV=true
```

下面详细说明了集群准备过程。

1. 在集群的每个节点上安装并注册 SUSE Linux Enterprise Server 12 SP3 以及 SUSE Enterprise Storage 扩展。
2. 列出现有的软件储存库，确认是否已安装并注册正确的产品。该列表与以下输出类似：

```
 root@minion > zypper lr -E
#  | Alias   | Name                                 | Enabled | GPG Check |
 Refresh
---+---------+--------------------------------------+---------+-----------
+--------
 4 | [...]   | SUSE-Enterprise-Storage-5-Pool       | Yes     | (r ) Yes  | No
 6 | [...]   | SUSE-Enterprise-Storage-5-Updates    | Yes     | (r ) Yes  |
 Yes
 9 | [...]   | SLES12-SP3-Pool                      | Yes     | (r ) Yes  | No
11 | [...]   | SLES12-SP3-Updates                   | Yes     | (r ) Yes  |
 Yes
```

3. 在每个节点上配置网络设置，包括正确的 DNS 名称解析。Salt Master 和所有 Salt Minion 需要根据各自的主机名相互解析。有关配置网络的详细信息，请参见 https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_basicnet_yast.html。有关配置 DNS 服务器的详细信息，请参见 https://www.suse.com/documentation/sles-12/book_sle_admin/data/cha_dns.html。
4. 配置、启用并启动 NTP 时间同步服务器：

```
root@master # systemctl enable ntpd.service
root@master # systemctl start ntpd.service
```

在 https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_netz_xntp_yast.html 中可以找到有关设置 NTP 的详细信息。

5. 检查 AppArmor 服务是否正在运行，并在每个集群节点上禁用该服务。启动 YaST AppArmor 模块，选择设置，然后取消选择启用 Apparmor 复选框。点击完成进行确认。请注意，在启用 AppArmor 的情况下，SUSE Enterprise Storage 将无法正常工作。

6. 在 Salt Master 节点上安装 `salt-master` 和 `salt-minion` 包：

```
root@master # zypper in salt-master salt-minion
```

检查 `salt-master` 服务是否已启用并启动，并根据需要将它启用并启动：

```
root@master # systemctl enable salt-master.service
root@master # systemctl start salt-master.service
```

7. 如果要使用防火墙，请确认 Salt Master 节点是否为所有 Salt Minion 节点打开了端口 4505 和 4506。如果这些端口处于关闭状态，可以使用 `yast2 firewall` 命令并通过允许 SaltStack 服务来打开这些端口。

警告：使用防火墙时 DeepSea 阶段失败

当防火墙处于活动状态（甚至只是配置了防火墙）时，DeepSea 部署阶段会失败。要正确通过该阶段，需要运行以下命令关闭防火墙

```
root@master # systemctl stop SuSEfirewall2.service
```

或在 `/srv/pillar/ceph/stack/global.yml` 中将 `FAIL_ON_WARNING` 选项设为“False”：

```
FAIL_ON_WARNING: False
```

8. 在所有 Minion 节点上安装 `salt-minion` 包。

```
root@minion > zypper in salt-minion
```

请确保所有其他节点都可将每个节点的完全限定的域名解析为公共网络 IP 地址。

9. 将所有 Minion（包括 Master Minion）配置为连接到 Master。如果无法通过主机名 `salt` 访问 Salt Master，请编辑文件 `/etc/salt/minion`，或创建包含以下内容的新文件 `/etc/salt/minion.d/master.conf`：

```
master: host_name_of_salt_master
```

如果对上述配置文件执行了任何更改，请在所有 Salt Minion 上重启动 Salt 服务：

```
root@minion > systemctl restart salt-minion.service
```

10. 检查所有节点上是否已启用并启动 `salt-minion` 服务。根据需要启用并启动该服务：

```
root@minion > systemctl enable salt-minion.service
root@minion > systemctl start salt-minion.service
```

11. 校验每个 Salt Minion 的指纹，如果指纹匹配，则接受 Salt Master 上的所有 Salt 密钥。查看每个 Minion 的指纹：

```
root@minion > salt-call --local key.finger
local:
3f:a3:2f:3f:b4:d3:d9:24:49:ca:6b:2c:e1:6c:3f:c3:83:37:f0:aa:87:42:e8:ff...
```

收集到所有 Salt Minion 的指纹后，将列出 Salt Master 上所有未接受 Minion 密钥的指纹：

```
root@master # salt-key -F
[...]
Unaccepted Keys:
minion1:
3f:a3:2f:3f:b4:d3:d9:24:49:ca:6b:2c:e1:6c:3f:c3:83:37:f0:aa:87:42:e8:ff...
```

如果 Minion 的指纹匹配，则接受这些密钥：

```
root@master # salt-key --accept-all
```

12. 校验是否已接受密钥：

```
root@master # salt-key --list-all
```

13. 在部署 SUSE Enterprise Storage 之前，请确保以前的集群用作 OSD 的所有磁盘均为空且不包含分区。为确保这一点，您需要手动擦除所有磁盘。请记得将“X”替换为正确的盘符：

    a. 停止使用特定磁盘的所有进程。

    b. 确认磁盘上是否装入任何分区，并视需要进行卸载。

    c. 如果磁盘由 LVM 管理，请停用整个 LVM 基础架构并将其删除。有关更多详细信息，请参见https://www.suse.com/documentation/sles-12/stor_admin/data/cha_lvm.html。

    d. 如果磁盘是 MD RAID 的一部分，请停用 RAID。有关更多详细信息，请参见https://www.suse.com/documentation/sles-12/stor_admin/data/part_software_raid.html。

    e. 

    提示：重引导服务器

    如果您在执行以下步骤时收到诸如“分区正在使用”或“无法使用新的分区表更新内核”之类的错误讯息，请重引导服务器。

    擦除每个分区的开头部分：

    ```
    for partition in /dev/sdX[0-9]*
    do
      dd if=/dev/zero of=$partition bs=4096 count=1 oflag=direct
    done
    ```

    f. 擦除分区表：

    ```
    sgdisk -Z --clear -g /dev/sdX
    ```

    g. 擦除备份分区表：

    ```
    size=`blockdev --getsz /dev/sdX`
    position=$((size/4096 - 33))
    dd if=/dev/zero of=/dev/sdX bs=4M count=33 seek=$position oflag=direct
    ```

14. 在 Salt Master 节点上安装 DeepSea：

```
root@master # zypper in deepsea
```

15. 检查 Salt Master 上的文件 `/srv/pillar/ceph/master_minion.sls` 是否指向您的 Salt Master。如果可以通过其他主机名访问您的 Salt Master，请使用存储集群适用的主机名。如果在 ses 域中使用了 Salt Master 的默认主机名 salt，则该文件如下所示：

```
master_minion: salt.ses
```

现在部署并配置 Ceph。除非另有说明，否则必须执行所有步骤。

注意：Salt 命令约定

可通过两种方式运行 `salt-run state.orch`，一种方式是使用 `stage.<stage number>`，另一种方式是使用阶段的名称。这两种表示法会产生相同的效果，至于使用哪个命令，完全取决于您的偏好。

过程 4.1： 运行部署阶段

1. 包含属于当前正在部署的 Ceph 集群的 Salt Minion。有关定位 Minion 的详细信息，请参见第 4.2.2.1 节 “匹配 Minion 名称”。

2. 准备集群。有关更多详细信息，请参见DeepSea 阶段说明。

```
root@master # salt-run state.orch ceph.stage.0
```

或者

```
root@master # salt-run state.orch ceph.stage.prep
```

注意：使用 DeepSea CLI 运行或监视阶段

使用 DeepSea CLI，可通过在监视模式下运行 DeepSea CLI，或者直接通过 DeepSea CLI 运行阶段，来实时跟踪阶段执行进度。有关详细信息，请参见第 4.4 节 “DeepSea CLI”。

3. 可选：为 `/var/lib/ceph/` 创建 Btrfs 子卷。只能在执行 DeepSea 的后续阶段之前执行此步骤。要迁移现有目录或了解更多详细信息，请参见《管理指南》, 第 18 章 “技巧与提示”, 第 18.5 节 “/var/lib/ceph 的 Btrfs 子卷”。

```
root@master # salt-run state.orch ceph.migrate.subvolume
```

4. 发现阶段会从所有 Minion 收集数据并创建配置分段，这些分段存储在 `/srv/pillar/ceph/proposals` 目录中。数据以 YAML 格式存储在 *.sls 或 *.yml 文件中。

```
root@master # salt-run state.orch ceph.stage.1
```

或者

```
root@master # salt-run state.orch ceph.stage.discovery
```

5. 成功完成上述命令后，请在 `/srv/pillar/ceph/proposals` 中创建 `policy.cfg` 文件。有关详细信息，请参见第 4.5.1 节 “policy.cfg 文件”。

提示

如果需要更改集群的网络设置，请编辑 `/srv/pillar/ceph/stack/ceph/cluster.yml`，调整以 `cluster_network:` 和 `public_network:` 开头的行。

6. 配置阶段将会分析 `policy.cfg` 文件，并将包含的文件合并成其最终形式。集群和角色相关的内容放置在 `/srv/pillar/ceph/cluster` 中，而 Ceph 特定的内容放置在 `/srv/pillar/ceph/stack/default` 中。
运行以下命令以触发配置阶段：

```
root@master # salt-run state.orch ceph.stage.2
```

或者

```
root@master # salt-run state.orch ceph.stage.configure
```

配置步骤可能需要花几秒时间。命令完成后，您可以通过运行以下命令，查看指定 Minion（例如，名为 `ceph_minion1`、`ceph_minion2` 等的 Minion）的pillar 数据：

```
root@master # salt 'ceph_minion*' pillar.items
```

注意：重写默认值

一旦命令完成，您便可查看默认配置并根据需要进行更改。有关详细信息，请参见第 7 章 “自定义默认配置”。

7. 现在运行部署阶段。在此阶段，将会验证 pillar，并在存储节点上启动监视器和 OSD 守护进程。运行以下命令以启动该阶段：

```
root@master # salt-run state.orch ceph.stage.3
```

或者

```
root@master # salt-run state.orch ceph.stage.deploy
```

该命令可能需要花几分钟时间。如果该命令失败，则您需要解决问题，然后再次运行前面的阶段。该命令成功后，请运行以下命令来检查状态：

```
root@master # ceph -s
```

8. Ceph 集群部署过程的最后一个步骤是服务阶段。在此阶段，您要实例化当前支持的所有服务：iSCSI 网关、CephFS、对象网关、openATTIC 和 NFS Ganesha。此阶段将创建所需的存储池、授权密钥环和启动服务。要启动该阶段，请运行以下命令：

```
root@master # salt-run state.orch ceph.stage.4
```

或者

```
root@master # salt-run state.orch ceph.stage.services
```

根据具体的设置，该命令可能会运行几分钟时间。

4.4 DeepSea CLI

DeepSea 还提供了一个 CLI 工具，供用户监视或运行阶段，同时实时将执行进度可视化。

支持使用以下两种模式来可视化阶段的执行进度：

DEEPSEA CLI 模式

- 监视模式：可视化另一个终端会话中发出的 `salt-run` 命令所触发 DeepSea 阶段的执行进度。
- 独立模式：运行 DeepSea 阶段，并在该阶段的构成步骤执行时提供相应的实时可视化效果。

重要：DeepSea CLI 命令

只能使用 `root` 特权在 Salt Master 节点中运行 DeepSea CLI 命令。

4.4.1 DeepSea CLI：监视模式

进度监视器提供详细的实时可视化效果，显示在其他终端会话中使用 `salt-run state.orch` 命令执行阶段期间发生的情况。

在运行任何 `salt-run state.orch` 命令之前，需要启动监视器，使其可以检测到阶段的执行已开始。

如果在发出 `salt-run state.orch` 命令之后再启动监视器，将不会显示执行进度。

可运行以下命令来启动监视模式：

```
root@master # deepsea monitor
```

有关 `deepsea monitor` 命令的可用命令行选项的详细信息，请查看该命令的手册页：

```
root@master # man deepsea-monitor
```

4.4.2 DeepSea CLI：独立模式

在独立模式下，可以使用 DeepSea CLI 来运行 DeepSea 阶段，并实时显示其执行进度。

从 DeepSea CLI 中运行 DeepSea 阶段的命令的格式如下：

```
root@master # deepsea stage run stage-name
```

其中，`stage-name` 对应于 Salt 编制状态文件的引用方式。例如，对应于 /srv/salt/ceph/stage/deploy 中目录的 部署 阶段以 ceph.stage.deploy 的形式引用。

此命令可代替用于运行 DeepSea 阶段（或任何 DeepSea 编制状态文件）的基于 Salt 的命令。

命令 `deepsea stage run ceph.stage.0` 与 `salt-run state.orch ceph.stage.0` 的效果相同。

有关 `deepsea stage run` 命令接受的可用命令行选项的详细信息，请查看该命令的手册页：

```
root@master # man deepsea-stage run
```

下图显示了运行阶段 2 时，DeepSea CLI 的输出示例：

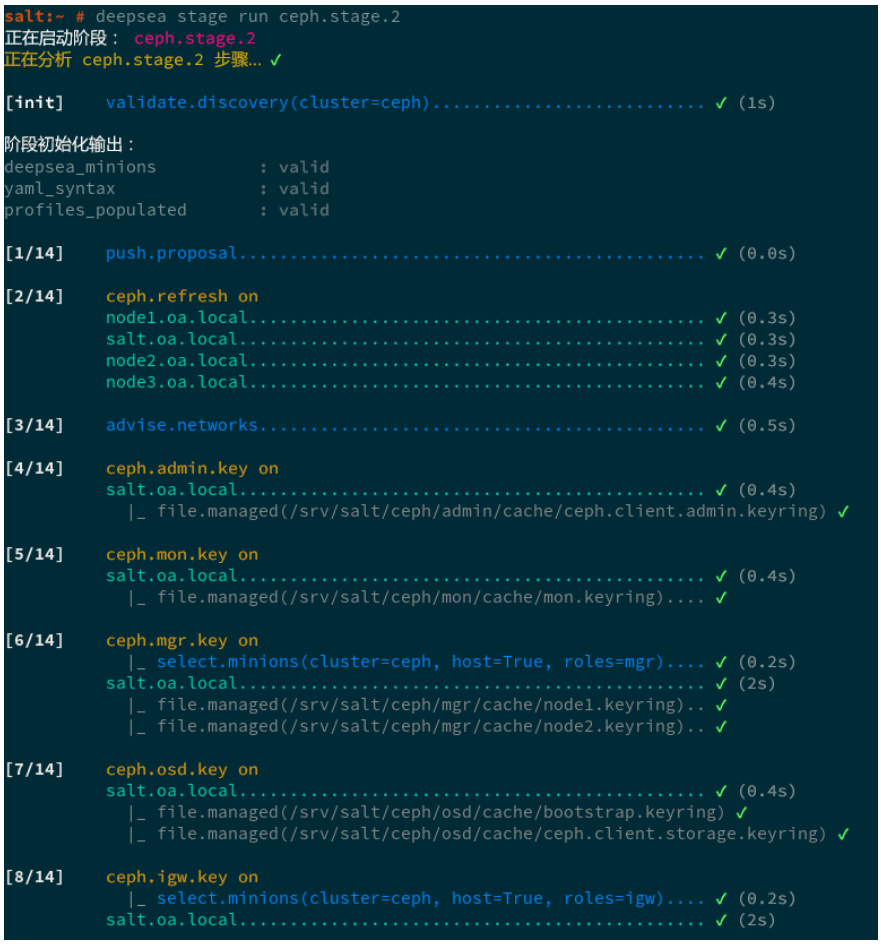

图 4.1： DEEPSEA CLI 阶段执行进度输出

4.4.2.1 DeepSea CLI `stage run` 别名

针对 Salt 的高级用户，我们还支持使用别名来运行 DeepSea 阶段，采用运行阶段所用的 Salt 命令（例如 `salt-run state.orch stage-name`）作为 DeepSea CLI 的命令。

示例：

```
root@master # deepsea salt-run state.orch stage-name
```

4.5 配置和自定义

4.5.1 `policy.cfg` 文件

`/srv/pillar/ceph/proposals/policy.cfg` 配置文件用于确定单个集群节点的角色。例如，哪个节点充当 OSD，或哪个节点充当监视器节点。请编辑 `policy.cfg`，以反映所需的集群设置。段落采用任意顺序，但所包含行的内容将覆盖前面行的内容中匹配的密钥。

提示：`policy.cfg` 的示例

可以在 `/usr/share/doc/packages/deepsea/examples/` 目录中找到完整策略文件的多个示例。

4.5.1.1 集群指定

在 `cluster` 段落选择集群的 Minion。可以选择所有 Minion，或者将 Minion 加入黑名单或白名单。下面显示了名为 `ceph` 的集群的示例。

要包含所有 Minion，请添加以下几行：

```
cluster-ceph/cluster/*.sls
```

要将特定的 Minion 加入白名单，请运行以下命令：

```
cluster-ceph/cluster/abc.domain.sls
```

要将一组 Minion 加入白名单，可以使用外壳通配符：

```
cluster-ceph/cluster/mon*.sls
```

要将 Minion 加入黑名单，可将其设置为 `unassigned`：

```
cluster-unassigned/cluster/client*.sls
```

4.5.1.2 角色指定

本节详细介绍了如何为您的集群节点指定“角色”。在此上下文中，“角色”是指您需要在节点上运行的服务，例如 Ceph Monitor、对象网关、iSCSI 网关或 openATTIC。不会自动指定任何角色，只会部署已添加到 `policy.cfg` 中的角色。

指定遵循以下模式：

```
role-ROLE_NAME/PATH/FILES_TO_INCLUDE
```

其中的项目具有以下含义和值：

- `ROLE_NAME` 为下列任何一项：“master”、“admin”、“mon”、“mgr”、“mds”、“igw”、“rgw”、“ganesha”或“openattic”。
- `PATH` 是 .sls 或 .yml 文件的相对目录路径。对于 .sls 文件，该路径通常是 `cluster`；而 .yml 文件则位于 `stack/default/ceph/minions`。
- `FILES_TO_INCLUDE` 是 Salt 状态文件或 YAML 配置文件。该文件通常包含 Salt Minion 主机名，例如 `ses5min2.yml`。可以使用外壳通配进行更具体的匹配。

每个角色的示例如下：

- master - 该节点具有所有 Ceph 集群的管理密钥环。目前仅支持一个 Ceph 集群。由于 master 角色是必需的，因此，请始终添加类似下方所示的行：

  ```
  role-master/cluster/master*.sls
  ```

- admin - 该 Minion 将获得管理密钥环。可按如下方式定义角色：

  ```
  role-admin/cluster/abc*.sls
  ```

- mon - 该 Minion 将向 Ceph 集群提供监视服务。此角色需要已指定 Minion 的地址。从 SUSE Enterprise Storage 5 开始，将以动态方式计算公用地址，并且 Salt Pillar 中不再需要该地址。

  ```
  role-mon/cluster/mon*.sls
  ```

  该示例将监视角色指定给一组 Minion。

- mgr - 从整个集群收集所有状态信息的 Ceph Manager 守护进程。请将它部署在您打算在其上部署 Ceph Monitor 角色的所有 Minion 上。

  ```
  role-mgr/cluster/mgr*.sls
  ```

- mds - 该 Minion 将提供元数据服务来支持 CephFS。

  ```
  role-mds/cluster/mds*.sls
  ```

- igw - 该 Minion 将充当 iSCSI 网关。此角色需要所指定 Minion 的地址，因此，您还需要包含 `stack` 目录中的文件：

  ```
  role-igw/stack/default/ceph/minions/xyz.domain.yml
  role-igw/cluster/*.sls
  ```

- rgw - 该 Minion 将充当对象网关：

  ```
  role-rgw/cluster/rgw*.sls
  ```

- openattic - 该 Minion 将充当 openATTIC 服务器：

  ```
  role-openattic/cluster/openattic*.sls
  ```

  有关详细信息，请参见《管理指南》, 第 15 章 “openATTIC”。

- ganesha - 该 Minion 将充当 NFS Ganesha 服务器。“ganesha”角色需要集群中的“rgw”或“mds”角色，否则，验证将于阶段 3 中失败。
  要成功安装 NFS Ganesha，需要进行额外的配置。如果您要使用 NFS Ganesha，请在执行阶段 2 和 4 之前阅读第 12 章 “安装 NFS Ganesha”。但是，可以稍后再安装 NFS Ganesha。

在某些情况下，为 NFS Ganesha 节点定义自定义角色可能很有用。有关详细信息，请参见《管理指南》, 第 14 章 “NFS Ganesha：通过 NFS 导出 Ceph 数据”, 第 14.3 节 “自定义 NFS Ganesha 角色”。

注意：集群节点的多个角色

可将多个角色指定到一个节点。例如，可将 mds 角色指定到监视器节点：

```
role-mds/cluster/mon[1,2]*.sls
```

4.5.1.3 通用配置

通用配置段落包括发现（阶段 1）期间生成的配置文件。这些配置文件存储 `fsid` 或 `public_network` 等参数。要包含所需的 Ceph 通用配置，请添加以下几行：

```
config/stack/default/global.yml
config/stack/default/ceph/cluster.yml
```

4.5.1.4 配置指定

在 Ceph 中，单个存储角色并不足以描述同一硬件中可用的许多磁盘配置。DeepSea 阶段 1 将生成默认的存储配置建议。默认情况下，此建议是一个 `bluestore` 配置，它会尝试针对给定的硬件设置建议性能最高的配置。例如，外部日记优先于包含对象和元数据的单个磁盘。固态存储优先于旋转型磁盘。与角色类似，配置在 `policy.cfg` 中指定。

可在 profile-default 目录树中找到默认建议。要包含这种指定，请将以下两行添加到 `policy.cfg`。

```
profile-default/cluster/*.sls
profile-default/stack/default/ceph/minions/*.yml
```

您也可以使用建议运行程序，根据喜好创建自定义的存储配置。此运行程序提供三个方法：help、peek 和 populate。

`salt-run proposal.help` 列显此运行程序接受的各个自变量的相关帮助文本。

`salt-run proposal.peek` 根据传递的自变量显示生成的建议。

`salt-run proposal.populate` 将建议写入 `/srv/pillar/ceph/proposals` 子目录。传递 `name=myprofile` 可为存储配置命名。这会生成 profile-myprofile 子目录。

对于所有的其他自变量，请参见 `salt-run proposal.help` 的输出。

4.5.1.5 部署加密的 OSD

从 SUSE Enterprise Storage 5 开始，默认会使用 BlueStore 而非 FileStore 来部署 OSD。虽然 BlueStore 支持加密，但默认以非加密模式部署 Ceph OSD。我们假设部署 OSD 时使用的数据和 WAL/DB 磁盘都是干净的，且没有分区。如果磁盘曾经使用，请执行步骤 13中所述的过程进行擦除。

要在您的新部署中使用加密的 OSD，请结合 `encryption=dmcrypt` 自变量使用 `proposal.populate` 运行程序：

```
root@master # salt-run proposal.populate encryption=dmcrypt
```

4.5.1.6 项目过滤

有时，使用 *.sls 通配法无法包含给定目录中的所有文件。`policy.cfg` 文件分析器可识别以下过滤器：

警告：高级方法

本节介绍供高级用户使用的过滤方法。如果使用不当，过滤可能会导致问题发生，例如，节点编号改变。

slice=[start:end]

使用 slice 过滤器可以仅包含从 start 到 end-1 的项目。请注意，给定目录中的项目将按字母数字顺序排序。下行包含 `role-mon/cluster/` 子目录中的第三到第五个文件：

```
role-mon/cluster/*.sls slice[3:6]
```

re=regexp

使用正则表达式过滤器可以仅包含与给定表达式匹配的项目。例如：

```
role-mon/cluster/mon*.sls re=.*1[135]\.subdomainX\.sls$
```

4.5.1.7 policy.cfg 文件示例

下面是一个基本 `policy.cfg` 文件的示例：

```
## Cluster Assignment
cluster-ceph/cluster/*.sls (1)

## Roles
# ADMIN
role-master/cluster/examplesesadmin.sls (2)
role-admin/cluster/sesclient*.sls (3)

# MON
role-mon/cluster/ses-example-[123].sls (4)

# MGR
role-mgr/cluster/ses-example-[123].sls (5)

# MDS
role-mds/cluster/ses-example-4.sls (6)

# IGW
role-igw/stack/default/ceph/minions/ses-example-4.yml (7)
role-igw/cluster/ses-example-4.sls (8)

# RGW
role-rgw/cluster/ses-example-4.sls (9)

# openATTIC
role-openattic/cluster/openattic*.sls (10)

# COMMON
config/stack/default/global.yml (11)
config/stack/default/ceph/cluster.yml (12)
```

```
## Profiles
profile-default/cluster/*.sls 13
profile-default/stack/default/ceph/minions/*.yml 14
```

1 指示在 Ceph 集群中包含所有 Minion。如果您不想在 Ceph 集群中包含某些 Minion，请使用：

```
cluster-unassigned/cluster/*.sls
cluster-ceph/cluster/ses-example-*.sls
```

第一行将所有 Minion 标记为未指定。第二行覆盖与“ses-example-*.sls”匹配的 Minion，并将其指定到 Ceph 集群。

2 名为“examplesesadmin”的 Minion 具有“master”角色。顺便指出，这表示该 Minion 将获得集群的管理密钥。

3 与“sesclient*”匹配的所有 Minion 也将获得管理密钥。

4 与“ses-example-[123]”匹配的所有 Minion（应该是 ses-example-1、ses-example-2 和 ses-example-3 这三个 Minion）将设置为 MON 节点。

5 与“ses-example-[123]”匹配的所有 Minion（示例中的所有 MON 节点）将设置为 MGR 节点。

6 Minion“ses-example-4”将具有 MDS 角色。

7 确保 DeepSea 知道 IGW 节点的 IP 地址。

8 Minion“ses-example-4”将具有 IGW 角色。

9 Minion“ses-example-4”将具有 RGW 角色。

10 指定要部署 openATTIC 用户界面来管理 Ceph 集群。有关详细信息，请参见《管理指南》，第 15 章 “openATTIC”。

11 表示我们接受 `fsid` 和 `public_network` 等通用配置参数的默认值。

12 表示我们接受 `fsid` 和 `public_network` 等通用配置参数的默认值。

13 告知 DeepSea 要为每个 Minion 使用默认的硬件配置。选择默认的硬件配置意味着我们要将所有附加磁盘（根磁盘除外）用作 OSD。

14 告知 DeepSea 要为每个 Minion 使用默认的硬件配置。选择默认的硬件配置意味着我们要将所有附加磁盘（根磁盘除外）用作 OSD。

4.5.2 自定义 ceph.conf 文件

如果需要将自定义设置放入 `ceph.conf` 配置文件，请参见《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.11 节 “自定义 ceph.conf 文件”了解更多详细信息。

5 从旧版本升级

本章介绍将 SUSE Enterprise Storage 从旧版本升级到最新版本的步骤。

5.1 阅读发行说明

在发行说明中，可以找到有关自 SUSE Enterprise Storage 的上一个版本发行后所进行的更改的其他信息。检查发行说明以了解：

- 您的硬件是否有特殊的注意事项。
- 所用的任何软件包是否已发生重大更改。
- 是否需要对您的安装实施特殊预防措施。

发行说明还提供未能及时编入手册中的信息。它们还包含有关已知问题的说明。

安装包 `release-notes-ses` 之后，可在本地的 `/usr/share/doc/release-notes` 目录中或 https://www.suse.com/releasenotes/ 网页上找到发行说明。

5.2 一般升级过程

开始升级过程之前，请考虑以下几项：

升级顺序

在升级 Ceph 集群之前，需要针对 SCC 或 SMT 正确注册底层 SUSE Linux Enterprise Server 和 SUSE Enterprise Storage。当集群已联机并正在运行时，可以升级集群中的守护进程。某些类型的守护进程依赖于其他守护进程。例如，Ceph Object Gateway 依赖于 Ceph Monitor 和 Ceph OSD 守护进程。建议按以下顺序升级：

1. Ceph Monitor
2. Ceph Manager
3. Ceph OSD
4. 元数据服务器

5. 对象网关
6. iSCSI 网关
7. NFS Ganesha

删除不需要的操作系统快照

删除节点操作系统分区上不需要的文件系统快照。如此可确保升级期间有足够的可用磁盘空间。

检查集群运行状况

建议在开始升级过程之前先检查集群运行状况。

逐个升级

建议逐个升级特定类型的所有守护进程（例如，所有监视器守护进程或所有 OSD 守护进程），以确保它们全部采用相同的版本。另外，建议在尝试体验某个版本的新功能之前，升级集群中的所有守护进程。

升级特定类型的所有守护进程之后，请检查守护进程的状态。

确保在升级所有监视器之后，每个监视器已重新加入仲裁：

```
root # ceph mon stat
```

确保在升级所有 OSD 之后，每个 Ceph OSD 守护进程已重新加入集群：

```
root # ceph osd stat
```

设置 `require-osd-release luminous` 标识

将最后一个 OSD 升级到 SUSE Enterprise Storage 5 之后，监视器节点会检测所有 OSD 是否正在运行“luminous”版本的 Ceph，并可能会指出未设置 `require-osd-release luminous` osdmap 标识。在这种情况下，您需要手动设置此标识，以确认由于集群已升级到“luminous”，无法将它降级回 Ceph“jewel”。运行以下命令来设置该标识：

```
root@minion > sudo ceph osd require-osd-release luminous
```

该命令完成后，警告将会消失。

在全新安装的 SUSE Enterprise Storage 5 上，当 Ceph Monitor 创建初始 osdmap 时，会自动设置此标识，因此最终用户无需执行任何操作。

5.3 升级期间加密 OSD

从 SUSE Enterprise Storage 5 开始，默认会使用 BlueStore 而非 FileStore 来部署 OSD。虽然 BlueStore 支持加密，但默认以非加密模式部署 Ceph OSD。下面的过程介绍了在升级过程中加密 OSD 的步骤。我们假设部署 OSD 时使用的数据和 WAL/DB 磁盘都是干净的，且没有分区。如果磁盘曾经使用，请执行步骤 13中所述的过程进行擦除。

重要：一次一个 OSD

您需要逐个部署加密的 OSD，不能同时部署。这是因为 OSD 的数据会用完，集群将数次重复经历重新平衡的过程。

1. 为您的部署确定 `bluestore block db size` 和 `bluestore block wal size` 的值，并在 Salt Master 上的 `/srv/salt/ceph/configuration/files/ceph.conf.d/global.conf` 文件中添加这些值。需要以字节指定这些值。

   ```
   [global]
   bluestore block db size = 48318382080
   bluestore block wal size = 2147483648
   ```

   有关自定义 `ceph.conf` 文件的详细信息，请参见《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.11 节 “自定义 ceph.conf 文件”。

2. 运行 DeepSea 阶段 3 来分发更改：

   ```
   root@master # salt-run state.orch ceph.stage.3
   ```

3. 确认相关 OSD 节点上是否已更新 `ceph.conf` 文件：

   ```
   root@minion > cat /etc/ceph/ceph.conf
   ```

4. 对 `/srv/pillar/ceph/proposals/profile-default/stack/default/ceph/minions` 目录下与您要加密的 OSD 相关的 *.yml 文件进行编辑。检查它们的路径是否与 `/srv/pillar/ceph/proposals/policy.cfg` 文件中定义的路径一致，以确保您修改的是正确的 *.yml 文件。

重要：长磁盘标识符

在 `/srv/pillar/ceph/proposals/profile-default/stack/default/ceph/minions/*.yml` 文件中标识 OSD 磁盘时，请使用长磁盘标识符。

下面显示了一个 OSD 配置示例。请注意，由于我们需要加密，因此删除了 `db_size` 和 `wal_size` 选项：

```
ceph:
 storage:
   osds:
     /dev/disk/by-id/scsi-
SDELL_PERC_H730_Mini_007027b1065faa972100d34d7aa06d86:
       format: bluestore
       encryption: dmcrypt
       db: /dev/disk/by-id/nvme-
INTEL_SSDPEDMD020T4D_HHHL_NVMe_2000GB_PHFT642400HV2P0EGN
       wal: /dev/disk/by-id/nvme-
INTEL_SSDPEDMD020T4D_HHHL_NVMe_2000GB_PHFT642400HV2P0EGN
     /dev/disk/by-id/scsi-
SDELL_PERC_H730_Mini_00d146b1065faa972100d34d7aa06d86:
       format: bluestore
       encryption: dmcrypt
       db: /dev/disk/by-id/nvme-
INTEL_SSDPEDMD020T4D_HHHL_NVMe_2000GB_PHFT642400HV2P0EGN
       wal: /dev/disk/by-id/nvme-
INTEL_SSDPEDMD020T4D_HHHL_NVMe_2000GB_PHFT642400HV2P0EGN
```

5. 运行 DeepSea 阶段 2 和 3 以加密模式部署新的块存储 OSD：

```
root@master # salt-run state.orch ceph.stage.2
root@master # salt-run state.orch ceph.stage.3
```

您可以运行 `ceph -s` 或 `ceph osd tree` 来监控进度。在下一个 OSD 节点上重复该过程前，请务必使集群重新平衡。

5.4 从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5

重要：软件要求

您需要在要升级的所有 Ceph 节点上安装以下软件并将其更新到最新的包版本，然后才能开始升级过程：

- SUSE Linux Enterprise Server 12 SP2
- SUSE Enterprise Storage 4

此外，在开始升级之前，需要通过运行 `zypper migration`（或偏好的升级方式）将 Salt Master 节点升级到 SUSE Linux Enterprise Server 12 SP3 和 SUSE Enterprise Storage 5。

警告：升级前需考虑的要点

- 检查 AppArmor 服务是否正在运行，并在每个集群节点上禁用该服务。启动 YaST AppArmor 模块，选择设置，然后取消选择启用 Apparmor 复选框。点击完成进行确认。
  请注意，在启用 AppArmor 的情况下，SUSE Enterprise Storage 将无法正常工作。
- 尽管集群在升级期间可完全正常运行，但 DeepSea 会设置“noout”标识，用于阻止 Ceph 在停机期间重新平衡数据，从而避免不必要的数据传输。
- 为了优化升级过程，DeepSea 会根据节点的指定角色并遵循 Ceph 上游的建议，按以下顺序升级节点：MON、MGR、OSD、MDS、RGW、IGW 和 NFS Ganesha。
  请注意，如果节点运行多个服务，DeepSea 将无法防止未按上述顺序执行升级。
- 尽管 Ceph 集群在升级期间可正常运行，但节点可能需要重引导才能应用新内核版本等设置。为了减少等待中的 I/O 操作，建议在升级过程中拒绝传入请求。

- 集群升级可能要花费很长时间 — 所需时间大约为升级一台计算机的时间乘以集群节点数。
- 从 Ceph Luminous 开始，不再支持 `osd crush location` 配置选项。请在升级前更新您的 DeepSea 配置文件，以使用 `crush location`。

要将 SUSE Enterprise Storage 4 集群升级到版本 5，请执行以下步骤：

1. 运行以下命令设置新的内部对象排序顺序：

```
root # ceph osd set sortbitwise
```

提示

要校验命令是否成功，我们建议运行以下命令

```
root # ceph osd dump --format json-pretty | grep sortbitwise
 "flags": "sortbitwise,recovery_deletes,purged_snapdirs",
```

2. 使用 `rpm -q deepsea` 校验 Salt Master 节点上的 DeepSea 包版本是否至少以 `0.7` 开头。例如：

```
root # rpm -q deepsea
deepsea-0.7.27+git.0.274c55d-5.1
```

如果 DeepSea 包版本号以 0.6 开头，请再次确认是否已成功将 Salt Master 节点迁移到 SUSE Linux Enterprise Server 12 SP3 和 SUSE Enterprise Storage 5（请参见本节开头的重要：软件要求）。在开始升级过程之前，必须满足此先决条件。

3. a. 如果已将系统注册到 SUSEConnect 并使用 SCC/SMT，则不需要执行进一步的操作。继续 步骤 4。

b. 如果您使用的不是 SCC/SMT 而是媒体 ISO 或其他包来源，请手动添加以下储存库：SLE12-SP3 基础、SLE12-SP3 更新、SES5 基础和 SES5 更新。您可以使用 `zypper` 命令来执行此操作。首先，删除所有现有的软件储存库，然后添加所需的新储存库，最后刷新储存库来源：

```
root # zypper sd {0..99}
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Products/Storage/5/x86_64/product/
 SES5-POOL
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Updates/Storage/5/x86_64/update/
 SES5-UPDATES
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Products/SLE-SERVER/12-SP3/x86_64/
product/ SLES12-SP3-POOL
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Updates/SLE-SERVER/12-SP3/x86_64/
update/ SLES12-SP3-UPDATES
root # zypper ref
```

之后，更改您的 Pillar 数据以使用另一个策略。编辑

```
/srv/pillar/ceph/stack/name_of_cluster/cluster.yml
```

并添加下行：

```
upgrade_init: zypper-dup
```

提示

`zypper-dup` 策略要求您手动添加最新的软件储存库，而默认的 `zypper-migration` 则依赖于 SCC/SMT 提供的储存库。

4. 更新 Pillar：

```
root@master # salt target saltutil.sync_all
```

有关 Salt Minion 定位的详细信息，请参见第 4.2.2 节 “定位 Minion”。

5. 校验是否已成功写入 Pillar：

```
root@master # salt target pillar.get upgrade_init
```

该命令的输出应会镜像您添加的项。

6. 升级 Salt Minion：

```
root@master # salt target state.apply ceph.updates.salt
```

7. 校验是否已升级所有 Salt Minion：

```
root@master # salt target test.version
```

8. 包含集群的 Salt Minion。有关更多详细信息，请参见过程 4.1 “运行部署阶段”的第 4.2.2 节 “定位 Minion”。

9. 开始升级 SUSE Linux Enterprise Server 和 Ceph：

```
root@master # salt-run state.orch ceph.maintenance.upgrade
```

提示：重引导后重新运行

如果升级过程导致 Salt Master 重引导，请重新运行该命令，以再次启动 Salt Minion 的升级过程。

10. 升级后，检查所有节点上的 AppArmor 是否已禁用并已停止：

```
root # systemctl disable apparmor.service
systemctl stop apparmor.service
```

11. 升级后，Ceph Manager 暂时还未安装。要让集群保持正常状态，请执行以下操作：

    a. 运行阶段 0 以启用 Salt REST API：

```
root@master # salt-run state.orch ceph.stage.0
```

b. 运行阶段 1 以创建 `role-mgr/` 子目录：

```
root@master # salt-run state.orch ceph.stage.1
```

c. 根据第 4.5.1 节 “policy.cfg 文件”中所述编辑 policy.cfg，并将一个 Ceph Manager 角色添加到部署了 Ceph Monitor 的节点。此外，请为集群的某个节点添加 openATTIC 角色。有关更多详细信息，请参见《管理指南》, 第 15 章 “openATTIC”。

d. 运行阶段 2 以更新 Pillar：

```
root@master # salt-run state.orch ceph.stage.2
```

e. DeepSea 现在用另一种方法来生成 `ceph.conf` 配置文件，有关更多详细信息，请参见《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.11 节 “自定义 `ceph.conf` 文件”。

f. 运行阶段 3 以部署 Ceph Manager：

```
root@master # salt-run state.orch ceph.stage.3
```

g. 运行阶段 4 以正确配置 openATTIC：

```
root@master # salt-run state.orch ceph.stage.4
```

注意：Ceph 密钥功能不匹配

如果 `ceph.stage.3` 失败并出现“错误 EINVAL：实体 client.bootstrap-osd 存在，但功能不匹配”，则表示现有集群的 `client.bootstrap.osd` 密钥的密钥功能 (caps) 与 DeepSea 尝试设置的功能不匹配。在红色错误讯息的上方，可以看到失败命令 `ceph auth` 的转储。请查看此命令，检查所用的密钥 ID 和文件。对于 `client.bootstrap-osd`，该命令是

```
root # ceph auth add client.bootstrap-osd \
 -i /srv/salt/ceph/osd/cache/bootstrap.keyring
```

要解决密钥功能不匹配问题，请检查 DeepSea 正在尝试部署的密钥环文件的内容，例如：

```
cephadm > cat /srv/salt/ceph/osd/cache/bootstrap.keyring
[client.bootstrap-osd]
     key = AQD6BpVZgqVwHBAAQerW3atANeQhia8m5xaigw==
     caps mgr = "allow r"
     caps mon = "allow profile bootstrap-osd"
```

将此内容与 `ceph auth get client.bootstrap-osd` 的输出进行比较：

```
root # ceph auth get client.bootstrap-osd
exported keyring for client.bootstrap-osd
[client.bootstrap-osd]
     key = AQD6BpVZgqVwHBAAQerW3atANeQhia8m5xaigw==
     caps mon = "allow profile bootstrap-osd"
```

可以看到，后一个密钥缺少 `caps mgr = "allow r"`。要解决此问题，请运行：

```
root # ceph auth caps client.bootstrap-osd mgr \
 "allow r" mon "allow profile bootstrap-osd"
```

现在，运行 `ceph.stage.3` 应该会成功。

运行 `ceph.stage.4` 时，元数据服务器和对象网关密钥环可能会出现相同的问题。可以运用上述相同的过程：检查失败的命令、要部署的密钥环文件，以及现有密钥的功能。然后运行 `ceph auth caps` 更新现有的密钥功能，使之与 DeepSea 正在部署的功能匹配。

重要：升级失败

如果集群处于“HEALTH_ERR”状态的持续时间超过 300 秒，或者每个指定角色的服务之一处于停机状态的持续时间超过 900 秒，则表示升级失败。在这种情况下，请尝试找出并解决问题，然后重新运行升级过程。请注意，在虚拟化环境中，超时会更短。

重要：重引导 OSD

升级到 SUSE Enterprise Storage 5 之后，FileStore OSD 启动所需的时间大约会多五分钟，因为 OSD 将对其磁盘中的文件执行一次性转换。

提示：检查集群组件/节点的版本

如果您需要确定单个集群组件和节点的版本（例如，确定所有节点在升级后是否真正处于相同的增补程序级别），可以运行

```
root@master # salt-run status.report
```

该命令将会遍历所有连接的 Salt Minion，并扫描 Ceph、Salt 和 SUSE Linux Enterprise Server 的版本号，最后提供一份报告，其中会列出大多数节点的版本，并显示其版本与大多数节点不相同的节点。

5.4.1 将 OSD 迁移到 BlueStore

OSD BlueStore 是 OSD 守护进程的新后端。从 SUSE Enterprise Storage 5 开始，它是默认的选项。与以文件形式将对象存储在 XFS 文件系统中的 FileStore 相比，BlueStore 可提供更高的性能，因为它直接将对象存储在底层块设备中。BlueStore 还能实现 FileStore 所不能提供的其他功能，例如内置压缩和 EC 覆盖。

具体而言，在 BlueStore 中，OSD 包含一个“wal”（预写式日志）设备和一个“db”（RocksDB 数据库）设备。RocksDB 数据库会保存 BlueStore OSD 的元数据。默认情况下，这两个设备驻留在 OSD 所在的同一台设备上，但也可以将它们放置在更快/不同的媒体上。

在 SES5 中，FileStore 和 BlueStore 均受支持，FileStore 和 BlueStore OSD 可在单个集群中共存。在 SUSE Enterprise Storage 升级过程中，FileStore OSD 不会自动转换到 BlueStore。请注意，在尚未迁移到 BlueStore 的 OSD 上，将无法使用特定于 BlueStore 的功能。

在转换到 BlueStore 之前，OSD 需要运行 SUSE Enterprise Storage 5。转换是个缓慢的过程，因为所有数据需要重新写入两次。尽管迁移过程可能需要很长时间才能完成，但在此期间，集群的工作不会中断，所有客户端都可继续访问集群。但是，迁移期间的性能势必会下降，原因是需要重新平衡和回填集群数据。

请执行以下过程将 FileStore OSD 迁移到 BlueStore：

提示：关闭安全措施

运行迁移所需的 Salt 命令会被安全措施阻止。要关闭这些预防措施，请运行下面的命令：

```
root@master # salt-run disengage.safety
```

1. 迁移硬件配置：

   ```
   root@master # salt-run state.orch ceph.migrate.policy
   ```

   此运行程序会迁移 `policy.cfg` 文件当前正在使用的所有硬件配置。它会处理 `policy.cfg`，查找使用原始数据结构的任何硬件配置，并将其转换为新的数据结构。这样会产生名为“migrated-`original_name`”的新硬件配置。`policy.cfg` 也会更新。如果原始配置包含单独的日记，BlueStore 配置将对该 OSD 的“wal”和“db”使用相同的设备。

2. DeepSea 通过将 OSD 的权重设置为 0（“抽取”数据，直到 OSD 已空）来迁移 OSD。您可以逐个迁移 OSD，也可以一次性迁移所有 OSD。在任一情况下，当 OSD 已空时，编制进程会删除该 OSD，然后使用新配置重新创建 OSD。

提示：建议的方法

如果您有大量的物理存储节点，或几乎没有任何数据，请使用 `ceph.migrate.nodes`。如果一个节点占用的容量小于总容量的 10%，则使用 `ceph.migrate.nodes` 同时移动这些 OSD 中的所有数据的速度可能略微快一点。

如果您不确定要使用哪种方法，或者站点包含的存储节点较少（例如，每个节点包含的数据大于集群数据的 10%），请选择 `ceph.migrate.osds`。

a. 要逐个迁移 OSD，请运行：

```
root@master # salt-run state.orch ceph.migrate.osds
```

b. 要同时迁移每个节点上的所有 OSD，请运行：

```
root@master # salt-run state.orch ceph.migrate.nodes
```

提示

编制进程不提供有关迁移进度的反馈，您可以使用

```
root # ceph osd tree
```

定期查看哪些 OSD 的权重为 0。

迁移到 BlueStore 之后，对象计数将保持不变，磁盘使用率也几乎相同。

5.5 从 SUSE Enterprise Storage 4（ceph-deploy 部署）升级到版本 5

重要：软件要求

您需要在要升级的所有 Ceph 节点上安装以下软件并将其更新到最新的包版本，然后才能开始升级过程：

- SUSE Linux Enterprise Server 12 SP2
- SUSE Enterprise Storage 4

为集群选择 Salt Master。如果集群部署了 Calamari，则 Calamari 节点已经是 Salt Master。否则，您用来运行 `ceph-deploy` 命令的管理节点将成为 Salt Master。

在开始以下过程之前，您需要运行 `zypper migration`（或偏好的升级方式）将 Salt Master 节点升级到 SUSE Linux Enterprise Server 12 SP3 和 SUSE Enterprise Storage 5。

要将使用 `ceph-deploy` 部署的 SUSE Enterprise Storage 4 集群升级到版本 5，请执行以下步骤：

过程 5.1： 要对所有集群节点应用的步骤（包括 CALAMARI 节点）

1. 安装 SLE-12-SP2/SES4 中的 `salt` 包：

```
root # zypper install salt
```

2. 安装 SLE-12-SP2/SES4 中的 `salt-minion` 包，然后启用并启动相关的服务：

```
root # zypper install salt-minion
root # systemctl enable salt-minion
root # systemctl start salt-minion
```

3. 确保主机名“salt”可解析成 Salt Master 节点的 IP 地址。如果无法通过主机名 `salt` 访问 Salt Master，请编辑文件 `/etc/salt/minion`，或创建包含以下内容的新文件 `/etc/salt/minion.d/master.conf`：

```
master: host_name_of_salt_master
```

提示

现有 Salt Minion 的 `/etc/salt/minion.d/calamari.conf` 中已设置 `master:` 选项。使用什么配置文件名无关紧要，但 `/etc/salt/minion.d/` 目录非常重要。

如果对上述配置文件执行了任何更改，请在所有 Salt Minion 上重启动 Salt 服务：

```
root@minion > systemctl restart salt-minion.service
```

4. a. 如果已将系统注册到 SUSEConnect 并使用 SCC/SMT，则不需要执行进一步的操作。

   b. 如果您使用的不是 SCC/SMT 而是媒体 ISO 或其他包来源，请手动添加以下储存库：SLE12-SP3 基础、SLE12-SP3 更新、SES5 基础和 SES5 更新。您可以使用 `zypper` 命令来执行此操作。首先，删除所有现有的软件储存库，然后添加所需的新储存库，最后刷新储存库来源：

```
root # zypper sd {0..99}
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Products/Storage/5/x86_64/product/
 SES5-POOL
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Updates/Storage/5/x86_64/update/
 SES5-UPDATES
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Products/SLE-SERVER/12-SP3/x86_64/
product/ SLES12-SP3-POOL
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Updates/SLE-SERVER/12-SP3/x86_64/
update/ SLES12-SP3-UPDATES
root # zypper ref
```

过程 5.2： 要对 SALT MASTER 节点应用的步骤

1. 运行以下命令设置新的内部对象排序顺序：

```
root@master # ceph osd set sortbitwise
```

提示

要校验命令是否成功，我们建议运行以下命令

```
root@master # ceph osd dump --format json-pretty | grep sortbitwise
 "flags": "sortbitwise,recovery_deletes,purged_snapdirs",
```

2. 将 Salt Master 节点升级到 SUSE Linux Enterprise Server 12 SP3 和 SUSE Enterprise Storage 5。对于在 SCC 中注册的系统，请使用 `zypper migration`。如果您是手动提供所需的软件储存库，请使用 `zypper dup`。升级后，请先确保 Salt Master 节点上只有 SUSE Linux Enterprise Server 12 SP3 和 SUSE Enterprise Storage 5 的储存库处于活动状态（且已刷新），然后再继续。

3. 如果该 Master 节点尚不存在，请安装 `salt-master` 包，然后启用并启动相关的服务：

```
root@master # zypper install salt-master
root@master # systemctl enable salt-master
root@master # systemctl start salt-master
```

4. 通过列出所有 Salt Minion 的密钥来校验这些 Minion 是否存在：

```
root@master # salt-key -L
```

5. 将所有 Salt Minion 密钥添加到 Salt Master（包括 Minion Master）：

```
root@master # salt-key -A -y
```

6. 确保所有 Salt Minion 的密钥已被接受：

```
root@master # salt-key -L
```

7. 确保 Salt Master 节点上的软件是最新的：

```
root@master # zypper migration
```

8. 安装 `deepsea` 包：

```
root@master # zypper install deepsea
```

9. 包含集群的 Salt Minion。有关更多详细信息，请参见过程 4.1 “运行部署阶段”的第 4.2.2 节 “定位 Minion”。

10. 导入 `ceph-deploy` 安装的现有集群：

```
root@master # salt-run populate.engulf_existing_cluster
```

该命令将执行以下操作：

- 将全部所需的 Salt 和 DeepSea 模块分发到所有 Salt Minion。
- 检查运行中的 Ceph 集群，并在 `/srv/pillar/ceph/proposals` 中填充集群的布局。
  将使用与所有检测到的运行中 Ceph 服务匹配的角色创建 `/srv/pillar/ceph/proposals/policy.cfg`。查看此文件以校验每个现有的 MON、OSD、RGW 和 MDS 节点是否具有适当的角色。OSD 节点将导入到 `profile-import/` 子目录，因此您可以检查 `/srv/pillar/ceph/proposals/profile-import/cluster/` 和 `/srv/pillar/ceph/proposals/profile-import/stack/default/ceph/minions/` 中的文件，以确认是否正确选取了 OSD。

  

  注意

  对于 Salt Master 节点，生成的 `policy.cfg` 只会应用检测到的 Ceph 服务的角色“role-mon”、“role-mgr”、“role-mds”、“role-rgw”、“role-admin”和“role-master”。如果需要其他角色，则需手动将其添加到该文件（请参见第 4.5.1.2 节 “角色指定”）。

- 现有集群的 `ceph.conf` 将保存到 `/srv/salt/ceph/configuration/files/ceph.conf.import`。

- `/srv/pillar/ceph/proposals/config/stack/default/ceph/cluster.yml` 将包含集群的 fsid、集群网络和公共网络，并指定 `configuration_init: default-import` 选项，如此 DeepSea 将使用前面所述的 `ceph.conf.import` 配置文件，而不是使用 DeepSea 的默认模板 `/srv/salt/ceph/configuration/files/ceph.conf.j2`。

  

  ### 注意：自定义 ceph.conf

  如果您需要将 `ceph.conf` 文件与自定义更改相集成，请等待导入/升级过程成功完成。然后，编辑 `/srv/pillar/ceph/proposals/config/stack/default/ceph/cluster.yml` 文件，注释掉下面一行：

  ```
  configuration_init: default-import
  ```

  保存文件，然后按《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.11 节 “自定义 ceph.conf 文件”中的信息操作。

- 集群的各个密钥环将保存到以下目录：

  ```
  /srv/salt/ceph/admin/cache/
  /srv/salt/ceph/mon/cache/
  /srv/salt/ceph/osd/cache/
  /srv/salt/ceph/mds/cache/
  /srv/salt/ceph/rgw/cache/
  ```

  确认这些密钥环文件都存在，并且下面的目录中没有密钥环文件（低于 SUSE Enterprise Storage 5 的版本中不存在 Ceph Manager）：

  ```
  /srv/salt/ceph/mgr/cache/
  ```

11. `salt-run populate.engulf_existing_cluster` 命令不会处理 openATTIC 配置的导入过程。您需要手动编辑 `policy.cfg` 文件，并添加 `role-openattic` 一行。有关更多详细信息，请参见第 4.5.1 节 “policy.cfg 文件”。

12. `salt-run populate.engulf_existing_cluster` 命令不会处理 iSCSI 网关配置的导入过程。如果您的集群包含 iSCSI 网关，请手动导入其配置：

a. 在其中一个 iSCSI 网关节点上，导出当前的 `lrbd.conf`，然后将其复制到 Salt Master 节点：

```
root@minion > lrbd -o >/tmp/lrbd.conf
root@minion > scp /tmp/lrbd.conf admin:/srv/salt/ceph/igw/cache/
lrbd.conf
```

b. 在 Salt Master 节点上，将默认 iSCSI 网关配置添加到 DeepSea 设置中：

```
root@master # mkdir -p /srv/pillar/ceph/stack/ceph/
root@master # echo 'igw_config: default-ui' >> /srv/pillar/ceph/stack/
ceph/cluster.yml
root@master # chown salt:salt /srv/pillar/ceph/stack/ceph/cluster.yml
```

c. 在 `policy.cfg` 中添加 iSCSI 网关角色，然后保存文件：

```
role-igw/stack/default/ceph/minions/ses-1.ses.suse.yml
role-igw/cluster/ses-1.ses.suse.sls
[...]
```

13. 运行阶段 1 以创建所有可能的角色：

```
root@master # salt-run state.orch ceph.stage.1
```

14. 在 `/srv/pillar/ceph/stack` 下生成所需的子目录：

```
root@master # salt-run push.proposal
```

15. 确认有正常运行的受 DeepSea 管理集群且为其正确指定了角色：

```
root@master # salt target pillar.get roles
```

将输出与集群的实际布局进行比较。

16. Calamari 会持续运行一个安排的 Salt 作业来检查集群状态，请删除该作业：

```
root@minion > salt target schedule.delete ceph.heartbeat
```

17. 然后，执行第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中所述的过程。

5.6 从 SUSE Enterprise Storage 4（Crowbar 部署）升级到版本 5

重要：软件要求

您需要在要升级的所有 Ceph 节点上安装以下软件并将其更新到最新的包版本，然后才能开始升级过程：

- SUSE Linux Enterprise Server 12 SP2
- SUSE Enterprise Storage 4

要将使用 Crowbar 部署的 SUSE Enterprise Storage 4 升级到版本 5，请执行以下步骤：

1. 对每个 Ceph 节点（包括 Calamari 节点）停止并禁用与 Crowbar 相关的所有服务：

```
root@minion > sudo systemctl stop chef-client
root@minion > sudo systemctl disable chef-client
root@minion > sudo systemctl disable crowbar_join
root@minion > sudo systemctl disable crowbar_notify_shutdown
```

2. 对每个 Ceph 节点（包括 Calamari 节点），确认软件储存库指向 SUSE Enterprise Storage 5 和 SUSE Linux Enterprise Server 12 SP3 产品。如果仍然存在指向较旧产品版本的储存库，请将其禁用。
3. 对每个 Ceph 节点（包括 Calamari 节点），确认 `salt-minion` 已安装。如果未安装，请予以安装：

```
root@minion > sudo zypper in salt salt-minion
```

4. 对于未安装 `salt-minion` 包的 Ceph 节点，请创建 `/etc/salt/minion.d/master.conf` 文件，并在其中将 `master` 选项指向 Calamari 节点的完整主机名：

```
master: full_calamari_hostname
```

提示

现有 Salt Minion 的 `/etc/salt/minion.d/calamari.conf` 中已设置 `master:` 选项。使用什么配置文件名无关紧要，但 `/etc/salt/minion.d/` 目录非常重要。

启用并启动 `salt-minion` 服务：

```
root@minion > sudo systemctl enable salt-minion
root@minion > sudo systemctl start salt-minion
```

5. 在 Calamari 节点上接受其他所有 Salt Minion 密钥：

```
root@master # salt-key -L
[...]
Unaccepted Keys:
d52-54-00-16-45-0a.example.com
d52-54-00-70-ac-30.example.com
[...]

root@master # salt-key -A
The following keys are going to be accepted:
Unaccepted Keys:
d52-54-00-16-45-0a.example.com
d52-54-00-70-ac-30.example.com
Proceed? [n/Y] y
Key for minion d52-54-00-16-45-0a.example.com accepted.
Key for minion d52-54-00-70-ac-30.example.com accepted.
```

6. 如果 Ceph 部署在公共网络上，且没有 VLAN 接口，请在 Crowbar 公共网络为 Calamari 节点添加一个 VLAN 接口。

7. 使用 `zypper migration` 或您偏好的方法将 Calamari 节点升级到 SUSE Linux Enterprise Server 12 SP3 和 SUSE Enterprise Storage 5。此后，Calamari 节点便成为 Salt Master。升级后，重引导 Salt Master。

8. 在 Salt Master 上安装 DeepSea：

```
root@master # zypper in deepsea
```

9. 指定 `deepsea_minions` 选项，以将正确的 Salt Minion 组添加到部署阶段。有关更多详细信息，请参见第 4.2.2.3 节 “设置 deepsea_minions 选项”。

10. DeepSea 期望所有 Ceph 节点都有相同的 `/etc/ceph/ceph.conf`。Crowbar 会为每个节点部署一个稍有差别的 `ceph.conf`，因此您需要使它们保持一致：

    - 删除 `osd crush location hook` 选项，它是由 Calamari 所添加。
    - 从 `[mon]` 段落中删除 `public addr` 选项。
    - 从 `mon host` 选项中删除端口号。

11. 之前，如果您运行的是对象网关，Crowbar 会部署一个单独的 `/etc/ceph/ceph.conf.radosgw` 文件，以便将 Keystone 机密与常规 `ceph.conf` 文件区分开。Crowbar 还会添加一个自定义 `/etc/systemd/system/ceph-radosgw@.service` 文件。由于 DeepSea 不支持此文件，您需要将其删除：

    - 将所有 `[client.rgw....]` 段落（`ceph.conf.radosgw` 文件中）追加到所有节点上的 `/etc/ceph/ceph.conf` 中。
    - 在对象网关节点上运行下面的命令：

```
root@minion > rm /etc/systemd/system/ceph-radosgw@.service
systemctl reenable ceph-radosgw@rgw.public.$hostname
```

12. 复查从 Salt Master 运行时 `ceph status` 是否起作用：

```
root@master # ceph status
cluster a705580c-a7ae-4fae-815c-5cb9c1ded6c2
health HEALTH_OK
[...]
```

13. 导入现有集群：

```
root@master # salt-run populate.engulf_existing_cluster
root@master # salt-run state.orch ceph.stage.1
root@master # salt-run push.proposal
```

14. `salt-run populate.engulf_existing_cluster` 命令不会处理 iSCSI 网关配置的导入过程。如果您的集群包含 iSCSI 网关，请手动导入其配置：

    a. 在其中一个 iSCSI 网关节点上，导出当前的 `lrbd.conf`，然后将其复制到 Salt Master 节点：

    ```
    root@minion > lrbd -o > /tmp/lrbd.conf
    root@minion > scp /tmp/lrbd.conf admin:/srv/salt/ceph/igw/cache/
    lrbd.conf
    ```

    b. 在 Salt Master 节点上，将默认 iSCSI 网关配置添加到 DeepSea 设置中：

    ```
    root@master # mkdir -p /srv/pillar/ceph/stack/ceph/
    root@master # echo 'igw_config: default-ui' >> /srv/pillar/ceph/stack/
    ceph/cluster.yml
    root@master # chown salt:salt /srv/pillar/ceph/stack/ceph/cluster.yml
    ```

    c. 在 `policy.cfg` 中添加 iSCSI 网关角色，然后保存文件：

    ```
    role-igw/stack/default/ceph/minions/ses-1.ses.suse.yml
    role-igw/cluster/ses-1.ses.suse.sls
    [...]
    ```

15. a. 如果已将系统注册到 SUSEConnect 并使用 SCC/SMT，则不需要执行进一步的操作。

    b. 如果您使用的不是 SCC/SMT 而是媒体 ISO 或其他包来源，请手动添加以下储存库：SLE12-SP3 基础、SLE12-SP3 更新、SES5 基础和 SES5 更新。您可以使用 `zypper` 命令来执行此操作。首先，删除所有现有的软件储存库，然后添加所需的新储存库，最后刷新储存库来源：

    ```
    root # zypper sd {0..99}
    root # zypper ar \
    ```

```
 http://172.17.2.210:82/repo/SUSE/Products/Storage/5/x86_64/product/
 SES5-POOL
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Updates/Storage/5/x86_64/update/
 SES5-UPDATES
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Products/SLE-SERVER/12-SP3/x86_64/
product/ SLES12-SP3-POOL
root # zypper ar \
 http://172.17.2.210:82/repo/SUSE/Updates/SLE-SERVER/12-SP3/x86_64/
update/ SLES12-SP3-UPDATES
root # zypper ref
```

之后，更改您的 Pillar 数据以使用另一个策略。编辑

```
/srv/pillar/ceph/stack/name_of_cluster/cluster.yml
```

并添加下行：

```
upgrade_init: zypper-dup
```

提示

`zypper-dup` 策略要求您手动添加最新的软件储存库，而默认的 `zypper-migration` 则依赖于 SCC/SMT 提供的储存库。

16. 修复主机 Grain，以便让 DeepSea 在公共网络上为 Ceph 守护进程实例 ID 使用短主机名。对于每个节点，您需要使用新的（短）主机名运行 `grains.set`。运行 `grains.set` 前，请通过运行 `ceph status` 校验当前的监视器实例。下面是设置短主机名前后的示例：

```
root@master # salt target grains.get host
d52-54-00-16-45-0a.example.com:
    d52-54-00-16-45-0a
d52-54-00-49-17-2a.example.com:
    d52-54-00-49-17-2a
d52-54-00-76-21-bc.example.com:
```

```
    d52-54-00-76-21-bc
d52-54-00-70-ac-30.example.com:
    d52-54-00-70-ac-30
```

```
root@master # salt d52-54-00-16-45-0a.example.com grains.set \
 host public.d52-54-00-16-45-0a
root@master # salt d52-54-00-49-17-2a.example.com grains.set \
 host public.d52-54-00-49-17-2a
root@master # salt d52-54-00-76-21-bc.example.com grains.set \
 host public.d52-54-00-76-21-bc
root@master # salt d52-54-00-70-ac-30.example.com grains.set \
 host public.d52-54-00-70-ac-30
```

```
root@master # salt target grains.get host
d52-54-00-76-21-bc.example.com:
    public.d52-54-00-76-21-bc
d52-54-00-16-45-0a.example.com:
    public.d52-54-00-16-45-0a
d52-54-00-49-17-2a.example.com:
    public.d52-54-00-49-17-2a
d52-54-00-70-ac-30.example.com:
    public.d52-54-00-70-ac-30
```

17. 运行升级：

```
root@master # salt target state.apply ceph.updates
root@master # salt target test.version
root@master # salt-run state.orch ceph.maintenance.upgrade
```

每个节点都将重引导。集群将再次启动，报告没有活动的 Ceph Manager 实例。这是正常的。此时，Calamari 应未安装/不再运行。

18. 运行所有必要的部署阶段，以将集群置于健康状态：

```
root@master # salt-run state.orch ceph.stage.0
root@master # salt-run state.orch ceph.stage.1
root@master # salt-run state.orch ceph.stage.2
root@master # salt-run state.orch ceph.stage.3
```

19. 要部署 openATTIC（请参见《管理指南》，第 15 章 “openATTIC”），请在 `/srv/pillar/ceph/proposals/policy.cfg` 中添加合适的 `role-openattic` 行（请参见第 4.5.1.2 节 “角色指定”），然后运行：

```
root@master # salt-run state.orch ceph.stage.2
root@master # salt-run state.orch ceph.stage.4
```

20. 升级期间，您可能会收到“错误 EINVAL：项目 [...] 存在，但功能不匹配”错误。要修正这些错误，请参见第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”。

21. 执行剩余的清理操作：

    - Crowbar 会在每个 OSD 的 `/etc/fstab` 中创建一些条目。这些条目并不需要，因此请将其删除。
    - Calamari 会持续运行一个安排的 Salt 作业来检查集群状态，请删除该作业：

      ```
      root@master # salt target schedule.delete ceph.heartbeat
      ```

    - 同时还安装了一些不需要的包，大多为 ruby gem 以及 chef 相关的包。虽然删除它们并不是必需的，但您可能希望通过运行 `zypper rm pkg_name` 来删除。

5.7 从 SUSE Enterprise Storage 3 升级到版本 5

重要：软件要求

您需要在要升级的所有 Ceph 节点上安装以下软件并将其更新到最新的包版本，然后才能开始升级过程：

- SUSE Linux Enterprise Server 12 SP1
- SUSE Enterprise Storage 3

要将 SUSE Enterprise Storage 3 集群升级到版本 5，请依次执行过程 5.1 “要对所有集群节点应用的步骤（包括 Calamari 节点）”和过程 5.2 “要对 Salt Master 节点应用的步骤”中所述的步骤。

6 备份集群配置

本章说明应备份管理节点上的哪些文件。完成集群部署或迁移之后，应立即创建这些目录的备份。

6.1 备份 Salt 配置

可以备份 `/etc/salt/` 目录。该目录包含 Salt 配置文件，例如 Salt Master 密钥和已接受的客户端密钥。

从严格意义上讲，Salt 文件并不是备份管理节点所必需的，但这些文件能够简化 Salt 集群的重新部署。如果不备份这些文件，需要在新管理节点上重新注册 Salt Minion。

注意：Salt Master 私用密钥的安全性

务必将 Salt Master 私用密钥的备份存储在安全位置。Salt Master 密钥可用于操纵所有集群节点。

从备份恢复 `/etc/salt` 目录后，请重启动 Salt 服务：

```
root@master # systemctl restart salt-master
root@master # systemctl restart salt-minion
```

6.2 备份 DeepSea 配置

DeepSea 所需的所有文件都存储在 `/srv/pillar/`、`/srv/salt/` 和 `/etc/salt/master.d` 中。

如果需要重新部署管理节点，请在新节点上安装 DeepSea 包，并将备份的数据移回这些目录中。然后，无需做出更多更改，即可再次使用 DeepSea。在再次使用 DeepSea 之前，请确保已在管理节点上正确注册所有 Salt Minion。

7 自定义默认配置

可以更改在阶段 2 生成的默认集群配置（请参见DeepSea 阶段说明）。例如，您可能需要更改网络设置，或默认安装在 Salt Master 上的软件。要执行前一项操作，可以修改在阶段 2 之后更新的 pillar；对于后一项操作，通常可以通过创建自定义 `sls` 文件并将其添加到 pillar 来完成。以下各节介绍了详细信息。

7.1 使用自定义的配置文件

本节列出了需要添加/更改您自己的 `sls` 文件的多个任务。当您需要更改默认部署流程时，通常会使用这样的过程。

提示：为自定义 .sls 文件添加前缀

您的自定义 .sls 文件与 DeepSea 的 .sls 文件属于同一个子目录。为了防止从 DeepSea 包中新添加的文件重写您的 .sls 文件，请在您的文件名前面加上 `custom-` 字符串前缀。

7.1.1 禁用部署步骤

如果您遇到一个不属于 DeepSea 部署过程的特定任务，因而需要跳过此任务，请按照下面的示例创建一个“no-operation”文件：

过程 7.1： 禁用时间同步

1. 创建并保存包含以下内容的 `/srv/salt/ceph/time/disabled.sls`：

```
disable time setting:
test.nop
```

2. 编辑 `/srv/pillar/ceph/stack/global.yml`，添加下行，并保存该文件：

```
time_init: disabled
```

3. 通过刷新 pillar 并运行以下步骤进行校验：

```
root@master # salt target saltutil.pillar_refresh
root@master # salt 'admin.ceph' state.apply ceph.time
admin.ceph:
  Name: disable time setting - Function: test.nop - Result: Clean

Summary for admin.ceph
------------
Succeeded: 1
Failed:    0
------------
Total states run:     1
```

注意：唯一 ID

任务 ID“disable time setting”可以是在 `sls` 文件中唯一的任何讯息。可通过指定唯一的说明来防止 ID 冲突。

7.1.2 替换部署步骤

如果需要将特定步骤的默认行为替换为自定义行为，请创建包含替换内容的自定义 `sls` 文件。默认情况下，`/srv/salt/ceph/pool/default.sls` 会创建名为“demo”的 rbd 映像。在本示例中，我们不想创建此映像，而是需要以下两个映像：“archive1”和“archive2”。

过程 7.2： 将 DEMO RBD 映像替换为两个自定义 RBD 映像

1. 创建并保存包含以下内容的 `/srv/salt/ceph/pool/custom.sls`：

```
wait:
  module.run:
    - name: wait.out
    - kwargs:
        'status': "HEALTH_ERR" (1)
    - fire_event: True

archive1:
  cmd.run:
```

```
    - name: "rbd -p rbd create archive1 --size=1024"
    - unless: "rbd -p rbd ls | grep -q archive1$"
    - fire_event: True

archive2:
  cmd.run:
    - name: "rbd -p rbd create archive2 --size=768"
    - unless: "rbd -p rbd ls | grep -q archive2$"
    - fire_event: True
```

1. wait 模块将会暂停，直到 Ceph 集群状态不是 HEALTH_ERR 为止。在全新安装中，当有足够数量的 OSD 可用且存储池创建完成之前，Ceph 集群可能会一直保持此状态。
2. rbd 命令不是幂等的。如果在提供映像之后重新运行同一个创建命令，则 Salt 状态将失败。unless 语句可防止这种情况。

2. 要调用新建的自定义文件而不是默认文件，需要编辑 /srv/pillar/ceph/stack/ceph/cluster.yml，添加下行，然后保存该文件：

```
pool_init: custom
```

3. 通过刷新 pillar 并运行以下步骤进行校验：

```
root@master # salt target saltutil.pillar_refresh
root@master # salt 'admin.ceph' state.apply ceph.pool
```

注意：授权

创建存储池或映像需要有足够的授权。admin.ceph Minion 具有管理密钥环。

提示：替代方法

另一种做法是更改 /srv/pillar/ceph/stack/ceph/roles/master.yml 中的变量。使用此文件可缓解其他 Minion 的 pillar 数据的混乱情况。

7.1.3 修改部署步骤

有时，您可能需要通过特定的步骤来执行某些附加任务。不建议修改相关的状态文件，因为这可能导致将来的升级变得复杂。应创建一个单独的文件来执行第 7.1.2 节 “替换部署步骤”中所述的相同附加任务。

为新的 `sls` 文件提供一个描述性名称。例如，如果除了 demo 映像以外，您还需要创建两个 rbd 映像，则可将文件命名为 `archive.sls`。

过程 7.3： 创建两个附加 RBD 映像

1. 创建并保存包含以下内容的 `/srv/salt/ceph/pool/custom.sls`：

```
include:
 - .archive
 - .default
```

提示：Include 优先顺序

在本示例中，Salt 将会依次创建 archive 映像和 demo 映像。本示例中的顺序并不重要。要更改顺序，请颠倒 `include:` 指令后面的行。

可将 include 行直接添加到 `archive.sls`，这样也会创建所有映像。但是，不管将 include 行放在哪个位置，Salt 都会先处理所包含文件中的步骤。尽管可以使用 requires 和 order 语句覆盖此行为，但使用包含其他语句的单独文件可保证顺序正确，并减少出现混淆的可能性。

2. 编辑 `/srv/pillar/ceph/stack/ceph/cluster.yml`，添加下行，并保存该文件：

```
pool_init: custom
```

3. 通过刷新 pillar 并运行以下步骤进行校验：

```
root@master # salt target saltutil.pillar_refresh
root@master # salt 'admin.ceph' state.apply ceph.pool
```

7.1.4 修改部署阶段

如果需要添加完全独立的部署步骤，请创建 3 个新文件：用于执行命令的 `sls` 文件、编制文件，以及使新步骤与原始部署步骤保持一致的自定义文件。

例如，如果在执行准备阶段的过程中需要在所有 Minion 上运行 `logrotate`：

首先创建一个 `sls` 文件并包含 `logrotate` 命令。

过程 7.4： 在所有 SALT MINION 上运行 `logrotate`

1. 创建一个目录，例如 `/srv/salt/ceph/logrotate`。
2. 创建并保存包含以下内容的 `/srv/salt/ceph/logrotate/init.sls`：

```
rotate logs:
  cmd.run:
    - name: "/usr/sbin/logrotate /etc/logrotate.conf"
```

3. 校验该命令是否在 Minion 上正常工作：

```
root@master # salt 'admin.ceph' state.apply ceph.logrotate
```

由于编制文件需要在其他所有准备步骤之前运行，因此请将其添加到准备阶段 0：

1. 创建并保存包含以下内容的 `/srv/salt/ceph/stage/prep/logrotate.sls`：

```
logrotate:
  salt.state:
    - tgt: '*'
    - sls: ceph.logrotate
```

2. 校验编制文件是否正常工作：

```
root@master # salt-run state.orch ceph.stage.prep.logrotate
```

最后一个文件是自定义文件，其中包含附加步骤和原始步骤：

1. 创建并保存包含以下内容的 `/srv/salt/ceph/stage/prep/custom.sls`：

```
include:
```

```
  - .logrotate
  - .master
  - .minion
```

2. 覆盖默认行为。编辑 `/srv/pillar/ceph/stack/global.yml`，添加下行，并保存该文件：

```
stage_prep: custom
```

3. 校验阶段 0 是否正常工作：

```
root@master # salt-run state.orch ceph.stage.0
```

注意：为何使用 global.yml？

之所以选择 `global.yml` 文件而不选择 `cluster.yml`，是因为在准备阶段，所有 Minion 都不属于 Ceph 集群，并且无权访问 `cluster.yml` 中的任何设置。

7.1.5 在阶段 0 期间禁用更新和重引导

阶段 0 期间（有关 DeepSea 阶段的详细信息，请参见DeepSea 阶段说明）。Salt Master 和 Salt Minion 可能会因新更新的包（例如 `kernel`）要求重引导系统而进行重引导。

为防止在阶段 0 期间更新或重引导集群节点，请编辑 `/srv/pillar/ceph/stack/ceph/cluster.yml` 并添加 `stage_prep_master` 或 `stage_prep_minion` 选项，具体取决于您是否需要修改 Salt Master、所有 Salt Minion 或所有节点的行为。

这两个选项接受以下值：

default-no-update-no-reboot
: 阻止节点更新其包和重引导。

default-no-update-reboot
: 阻止节点更新其包，但允许重引导。

default-update-no-reboot
: 阻止节点重引导，但允许更新其包。

default

允许更新节点包和重引导。

7.2 修改已发现的配置

完成阶段 2 之后，您可能需要更改已发现的配置。要查看当前设置，请运行：

```
root@master # salt target pillar.items
```

单个 Minion 的默认配置的输出通常类似下方所示：

```
----------
    available_roles:
        - admin
        - mon
        - storage
        - mds
        - igw
        - rgw
        - client-cephfs
        - client-radosgw
        - client-iscsi
        - mds-nfs
        - rgw-nfs
        - master
    cluster:
        ceph
    cluster_network:
        172.16.22.0/24
    fsid:
        e08ec63c-8268-3f04-bcdb-614921e94342
    master_minion:
        admin.ceph
    mon_host:
        - 172.16.21.13
        - 172.16.21.11
        - 172.16.21.12
```

```
    mon_initial_members:
        - mon3
        - mon1
        - mon2
    public_address:
        172.16.21.11
    public_network:
        172.16.21.0/24
    roles:
        - admin
        - mon
        - mds
    time_server:
        admin.ceph
    time_service:
        ntp
```

将在多个配置文件之间分布上述设置。包含这些文件的目录结构在 `/srv/pillar/ceph/stack/stack.cfg` 目录中定义。以下文件通常用于描述您的集群：

- `/srv/pillar/ceph/stack/global.yml` - 该文件会影响 Salt 集群中的所有 Minion。
- `/srv/pillar/ceph/stack/ceph/cluster.yml` - 该文件会影响名为 `ceph` 的 Ceph 集群中的所有 Minion。
- `/srv/pillar/ceph/stack/ceph/roles/role.yml` - 会影响 `ceph` 集群中指定了特定角色的所有 Minion。
- `/srv/pillar/ceph/stack/cephminions/minion ID/yml` - 会影响单个 Minion。

注意：重写包含默认值的目录

`/srv/pillar/ceph/stack/default` 中存在存储了默认配置设置的平行目录树。请不要更改此处的值，否则会将其重写。

更改所收集的配置的典型过程如下：

1. 找到需要更改的配置项目的所在位置。例如，如果需要更改集群网络等集群相关设置，请编辑文件 `/srv/pillar/ceph/stack/ceph/cluster.yml` 。
2. 保存文件。
3. 运行以下命令来校验所做的更改：

   ```
   root@master # salt target saltutil.pillar_refresh
   ```

   ，再输入

   ```
   root@master # salt target pillar.items
   ```

III 安装其他服务

8 安装用于访问数据的服务 84

9 Ceph Object Gateway 85

10 安装 iSCSI 网关 92

11 安装 CephFS 109

12 安装 NFS Ganesha 114

13 通过 Samba 导出 CephFS 122

8 安装用于访问数据的服务

部署 SUSE Enterprise Storage 集群后，可能需要安装对象网关或 iSCSI 网关等附加软件以便访问您的数据，或者，也可以在 Ceph 集群之上部署一个集群文件系统。本章重点介绍手动安装。如果您的集群是使用 Salt 部署的，请参见第 4 章 “使用 DeepSea/Salt 部署”了解有关安装特定网关或 CephFS 的过程。

9 Ceph Object Gateway

Ceph Object Gateway 是构建在 `librgw` 之上的对象存储接口，为应用提供用于访问 Ceph 集群的 RESTful 网关。该网关支持两种接口：

- S3 兼容：通过与 Amazon S3 RESTful API 的某个大型子集兼容的接口提供对象存储功能。
- Swift 兼容：通过与 OpenStack Swift API 的某个大型子集兼容的接口提供对象存储功能。

对象网关守护进程使用嵌入式 HTTP 服务器 (CivetWeb) 来与 Ceph 集群交互。由于对象网关提供与 OpenStack Swift 和 Amazon S3 兼容的接口，因此具有自己的用户管理功能。对象网关可将数据存储在用于存储来自 CephFS 客户端或 RADOS 块设备客户端的数据的同一个集群中。S3 和 Swift API 共享一个公用的名称空间，因此，您可以使用其中一个 API 写入数据，使用另一个 API 检索数据。

重要：DeepSea 部署的对象网关

从 SUSE Enterprise Storage 5 开始，对象网关作为 DeepSea 角色安装，因此，您不需要手动安装。

要在集群部署期间安装对象网关，请参见第 4.3 节 “集群部署”。

要将包含对象网关的新节点添加到集群，请参见《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.2 节 “为节点添加新的角色”。

9.1 手动安装对象网关

1. 在未使用端口 80 的节点上安装对象网关。例如，运行 openATTIC 的节点已在使用端口 80。以下命令会安装所有必需的组件：

```
cephadm > sudo zypper ref && sudo zypper in ceph-radosgw
```

2. 如果之前的对象网关实例中的 Apache 服务器正在运行，请停止该服务器并禁用相关的服务：

```
cephadm > sudo systemctl stop disable apache2.service
```

3. 编辑 `/etc/ceph/ceph.conf`，添加以下几行：

```
[client.rgw.gateway_host]
 rgw frontends = "civetweb port=80"
```

提示

如果您想将对象网关/CivetWeb 配置为用于 SSL 加密，请相应地修改该行：

```
rgw frontends = civetweb port=7480s
 ssl_certificate=path_to_certificate.pem
```

4. 重启动对象网关服务。

```
cephadm > sudo systemctl restart ceph-radosgw@rgw.gateway_host
```

9.1.1 对象网关配置

配置对象网关需要执行多个步骤。

9.1.1.1 基本配置

配置 Ceph Object Gateway 需要一个正常运行的 Ceph 存储集群。Ceph Object Gateway 是 Ceph 存储集群的客户端。作为 Ceph 存储集群客户端，它需要：

- 网关实例的主机名，例如 `gateway`。
- 具有相应权限和密钥环的存储集群用户名。
- 用于存储网关数据的存储池。
- 网关实例的数据目录。
- Ceph 配置文件中的实例项。

每个实例都必须具有用户名和密钥才能与 Ceph 存储集群通讯。在以下步骤中，我们将使用监视器节点创建引导密钥环，然后基于引导密钥环创建对象网关实例用户密钥环。随后创建客户端用户名和密钥。接下来，我们将密钥添加到 Ceph 存储集群。最后，将密钥环分发到包含网关实例的节点。

1. 为网关创建密钥环：

```
cephadm > sudo ceph-authtool --create-keyring /etc/ceph/
ceph.client.rgw.keyring
cephadm > sudo chmod +r /etc/ceph/ceph.client.rgw.keyring
```

2. 为每个实例生成 Ceph Object Gateway 用户名和密钥。例如，我们将在 `client.radosgw` 后面使用名称 `gateway`：

```
cephadm > sudo ceph-authtool /etc/ceph/ceph.client.rgw.keyring \
  -n client.rgw.gateway --gen-key
```

3. 为密钥添加功能：

```
cephadm > sudo ceph-authtool -n client.rgw.gateway --cap osd 'allow rwx' \
  --cap mon 'allow rwx' /etc/ceph/ceph.client.rgw.keyring
```

4. 创建可让 Ceph Object Gateway 访问 Ceph 存储集群的密钥环和密钥之后，请将密钥添加到 Ceph 存储集群。例如：

```
cephadm > sudo ceph -k /etc/ceph/ceph.client.admin.keyring auth add
 client.rgw.gateway \
  -i /etc/ceph/ceph.client.rgw.keyring
```

5. 将密钥环分发到包含网关实例的节点：

```
cephadm > sudo scp /etc/ceph/ceph.client.rgw.keyring  ceph@hostname:/home/
ceph
cephadm > ssh hostname
cephadm > sudo mv ceph.client.rgw.keyring /etc/ceph/ceph.client.rgw.keyring
```

提示：使用引导密钥环

一种替代方法是创建对象网关引导密钥环，然后基于该密钥环创建对象网关密钥环：

1. 在某个监视器节点上创建对象网关引导密钥环：

```
cephadm > sudo ceph \
 auth get-or-create client.bootstrap-rgw mon 'allow profile
 bootstrap-rgw' \
 --connect-timeout=25 \
 --cluster=ceph \
 --name mon. \
 --keyring=/var/lib/ceph/mon/ceph-node_host/keyring \
 -o /var/lib/ceph/bootstrap-rgw/keyring
```

2. 创建 /var/lib/ceph/radosgw/ceph-rgw_name 目录，用于存储引导密钥环：

```
cephadm > sudo mkdir \
/var/lib/ceph/radosgw/ceph-rgw_name
```

3. 基于新建的引导密钥环创建对象网关密钥环：

```
cephadm > sudo ceph \
 auth get-or-create client.rgw.rgw_name osd 'allow rwx' mon 'allow
 rw' \
 --connect-timeout=25 \
 --cluster=ceph \
 --name client.bootstrap-rgw \
 --keyring=/var/lib/ceph/bootstrap-rgw/keyring \
 -o /var/lib/ceph/radosgw/ceph-rgw_name/keyring
```

4. 将对象网关密钥环复制到对象网关主机：

```
cephadm > sudo scp \
/var/lib/ceph/radosgw/ceph-rgw_name/keyring \
rgw_host:/var/lib/ceph/radosgw/ceph-rgw_name/keyring
```

9.1.1.2 创建存储池（可选）

Ceph Object Gateway 需要使用 Ceph 存储集群池来存储特定的网关数据。如果创建的用户具有适当的权限，则网关会自动创建存储池。但是，请务必在 Ceph 配置文件中为每个存储池设置适当的默认归置组数量。

存储池名称遵循 `ZONE_NAME.POOL_NAME` 语法。使用默认区域配置网关时，本示例中的默认区域名称为“default”：

```
.rgw.root
default.rgw.control
default.rgw.meta
default.rgw.log
default.rgw.buckets.index
default.rgw.buckets.data
```

要手动创建存储池，请参见《管理指南》, 第 7 章 “管理存储池”, 第 7.2.2 节 “创建存储池”。

重要：对象网关和纠删码池

只能对 `default.rgw.buckets.data` 存储池执行纠删码操作。需要复制所有其他存储池，否则将无法访问网关。

9.1.1.3 将网关配置添加到 Ceph

将 Ceph Object Gateway 配置添加到 Ceph 配置文件。Ceph Object Gateway 配置要求您标识 Ceph Object Gateway 实例。然后，指定安装了 Ceph Object Gateway 守护进程的主机名、密钥环（用于 cephx）和日志文件（可选）。例如：

```
[client.rgw.instance-name]
host = hostname
keyring = /etc/ceph/ceph.client.rgw.keyring
```

提示：对象网关日志文件

要覆盖默认的对象网关日志文件，请包含以下命令：

```
log file = /var/log/radosgw/client.rgw.instance-name.log
```

当配置客户端类型为 Ceph Object Gateway (radosgw) 的 Ceph 存储集群客户端时，网关实例的 `[client.rgw.*]` 部分会识别 Ceph 配置文件的此部分。后接实例名称。例如：

```
[client.rgw.gateway]
host = ceph-gateway
keyring = /etc/ceph/ceph.client.rgw.keyring
```

注意

`host` 必须是您的计算机主机名，不包括域名。

然后关闭 `print continue`。如果将其设置为 true，执行 PUT 操作时可能会遇到问题：

```
rgw print continue = false
```

要通过子域 S3 调用使用 Ceph Object Gateway（例如 `http://bucketname.hostname`），必须在 Ceph 配置文件的 `[client.rgw.gateway]` 段落下面添加 Ceph Object Gateway DNS 名称：

```
[client.rgw.gateway]
...
rgw dns name = hostname
```

此外，在使用 `http://bucketname.hostname` 语法时，应考虑在客户端计算机上安装 Dnsmasq 这样的 DNS 服务器。`dnsmasq.conf` 文件应包含以下设置：

```
address=/hostname/host-ip-address
listen-address=client-loopback-ip
```

然后，在客户端计算机上添加 `client-loopback-ip` IP 地址作为第一台 DNS 服务器。

9.1.1.4 创建数据目录

部署脚本可能无法创建默认的 Ceph Object Gateway 数据目录。如果尚未为 radosgw 守护进程的每个实例创建数据目录，请加以创建。Ceph 配置文件中的 `host` 变量确定哪个主机运行 radosgw 守护进程的每个实例。通常的格式会指定 radosgw 守护进程、集群名称和守护进程 ID。

```
cephadm > sudo mkdir -p /var/lib/ceph/radosgw/cluster-id
```

使用上面的示例 `ceph.conf` 设置执行以下命令：

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

9.1.1.5 重启动服务并启动网关

为了确保所有组件重新加载其配置，建议重启动 Ceph 存储集群服务。然后启动 `radosgw` 服务。有关详细信息，请参见《管理指南》, 第 2 章 “简介”和《管理指南》, 第 11 章 “Ceph Object Gateway”, 第 11.3 节 “操作对象网关服务”。

服务启动并运行后，您可以发出匿名的 GET 请求，以检查网关是否返回响应。向域名发出简单的 HTTP 请求应会返回以下响应：

```
<ListAllMyBucketsResult>
      <Owner>
              <ID>anonymous</ID>
              <DisplayName/>
      </Owner>
      <Buckets/>
</ListAllMyBucketsResult>
```

10 安装 iSCSI 网关

iSCSI 是一种存储区域网络 (SAN) 协议，可让客户端（称作发起程序）将 SCSI 命令发送到远程服务器上的 SCSI 存储设备（目标）。SUSE Enterprise Storage 包含一个可通过 iSCSI 协议向异构客户端（例如 Microsoft Windows* 和 VMware* vSphere）开放 Ceph 存储管理的工具。多路径 iSCSI 访问可让这些客户端实现可用性与可伸缩性，标准化 iSCSI 协议还在客户端与 SUSE Enterprise Storage 集群之间额外提供了一层安全隔离。该配置工具名为 `lrbd`。使用 `lrbd`，Ceph 存储管理员可以定义精简配置且复制的高可用性卷，用于支持只读快照、读写克隆资源，以及 Ceph RADOS 块设备 (RBD) 的自动大小调整。然后，管理员可以通过单个 `lrbd` 网关主机或支持多路径故障转移的多个网关主机来导出卷。Linux、Microsoft Windows 和 VMware 主机可以使用 iSCSI 协议连接到卷，因此可像任何其他 SCSI 块设备一样供您使用。这意味着，SUSE Enterprise Storage 客户可在 Ceph 上有效运行完整的块存储基础架构子系统，享用传统 SAN 的所有功能和优势，促成将来的发展。

本章详细介绍如何设置 Ceph 集群基础架构和 iSCSI 网关，使客户端主机能够通过 iSCSI 协议，像在本地存储设备上一样使用远程存储的数据。

10.1 iSCSI 块存储

iSCSI 是 RFC 3720 中指定的、使用因特网协议 (IP) 的小型计算机系统接口 (SCSI) 命令集的一种实施。iSCSI 以服务形式实施，其中，客户端（发起程序）在 TCP 端口 3260 上通过会话来与服务器（目标）通讯。iSCSI 目标的 IP 地址和端口称为 iSCSI 门户，其中，一个目标可通过一个或多个门户公开。一个目标与一个或多个门户的组合称为目标门户组 (TPG)。

iSCSI 的底层数据链路层协议通常为以太网。更具体地说，现代 iSCSI 基础架构使用 10 Gb 以太网或更快的网络实现最佳吞吐量。强烈建议在 iSCSI 网关与后端 Ceph 集群之间建立 10 Gb 以太网连接。

10.1.1 Linux 内核 iSCSI 目标

Linux 内核 iSCSI 目标最初称作 linux-iscsi.org 的 LIO，它是项目的原始域和网站。在过去一段时间，适用于 Linux 平台的 iSCSI 目标实施竞争产品不少于四个，但 LIO 作为单一 iSCSI 参照目标最终获得了压倒性优势。LIO 的主流内核代码使用简单但有点含糊的名称“目标”，旨在区分“目标核心”与各种前端和后端目标模块。

可以说，最常用的前端模块就是 iSCSI。但是，LIO 也支持光纤通道 (FC)、基于以太网的光纤通道 (FCoE) 和其他多种前端协议。目前，SUSE Enterprise Storage 仅支持 iSCSI 协议。

最常用的目标后端模块是能够方便地在目标主机上重新导出任何可用块设备的模块。此模块名为 iblock。但是，LIO 还有一个 RBD 特定的后端模块，该模块支持对 RBD 映像进行并行化多路径 I/O 访问。

10.1.2 iSCSI 发起程序

本节简要介绍 Linux、Microsoft Windows 和 VMware 平台上使用的 iSCSI 发起程序。

10.1.2.1 Linux

Linux 平台的标准发起程序是 `open-iscsi`。`open-iscsi` 会起动守护进程 `iscsid`，然后，用户可以使用该守护进程来发现任何给定门户上的 iSCSI 目标、登录到目标，以及映射 iSCSI 卷。`iscsid` 会与 SCSI 中间层通讯以创建内核中块设备，然后，内核便可像对待系统中任何其他的 SCSI 块设备一样来处理这些设备。可以结合设备映射程序多路径 (`dm-multipath`) 工具一起部署 `open-iscsi` 发起程序，以提供高度可用的 iSCSI 块设备。

10.1.2.2 Microsoft Windows 和 Hyper-V

Microsoft Windows 操作系统的默认 iSCSI 发起程序是 Microsoft iSCSI 发起程序。iSCSI 服务可通过图形用户界面 (GUI) 进行配置，并支持使用多路径 I/O 实现高可用性。

10.1.2.3 VMware

VMware vSphere 和 ESX 的默认 iSCSI 发起程序是 VMware ESX 软件 iSCSI 发起程序 `vmkiscsi`。启用该发起程序后，可通过 vSphere 客户端或使用 `vmkiscsi-tool` 命令对其进行配置。然后，可以使用 VMFS 来格式化通过 vSphere iSCSI 存储适配器连接的存储卷，并像使用任何其他 VM 存储设备一样使用它们。VMware 发起程序也支持使用多路径 I/O 实现高可用性。

10.2 有关 lrbd 的一般信息

`lrbd` 兼具 RADOS 块设备的优势与 iSCSI 无所不包的多样性。在 iSCSI 目标主机（称为 `lrbd` 网关）上采用 `lrbd` 后，需要利用块存储的任何应用都可受益于 Ceph，即使不运行 Ceph 客户端协议也是如此。而用户可以使用 iSCSI 或任何其他目标前端协议连接到 LIO 目标，从而可以转换针对 RBD 存储的所有目标 I/O。

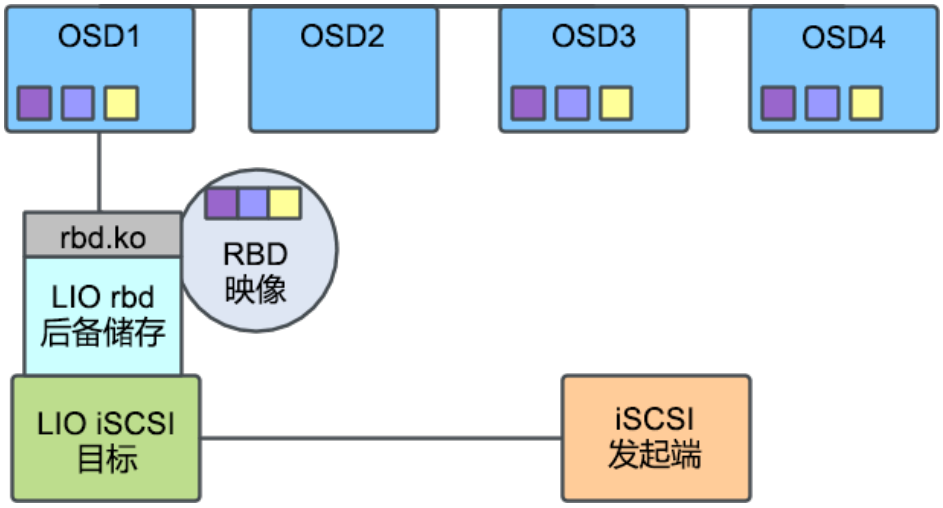

图 10.1： 包含单个 ISCSI 网关的 CEPH 集群

`lrbd` 本来就具有高可用性，并支持多路径操作。因此，下游发起程序主机可以使用多个 iSCSI 网关实现高可用性和可伸缩性。与包含多个网关的 iSCSI 配置通讯时，发起程序可在多个网关之间实现 iSCSI 请求的负载平衡。如果某个网关发生故障（暂时不可访问，或因为维护已被禁用），将通过另一个网关以透明方式继续处理 I/O。

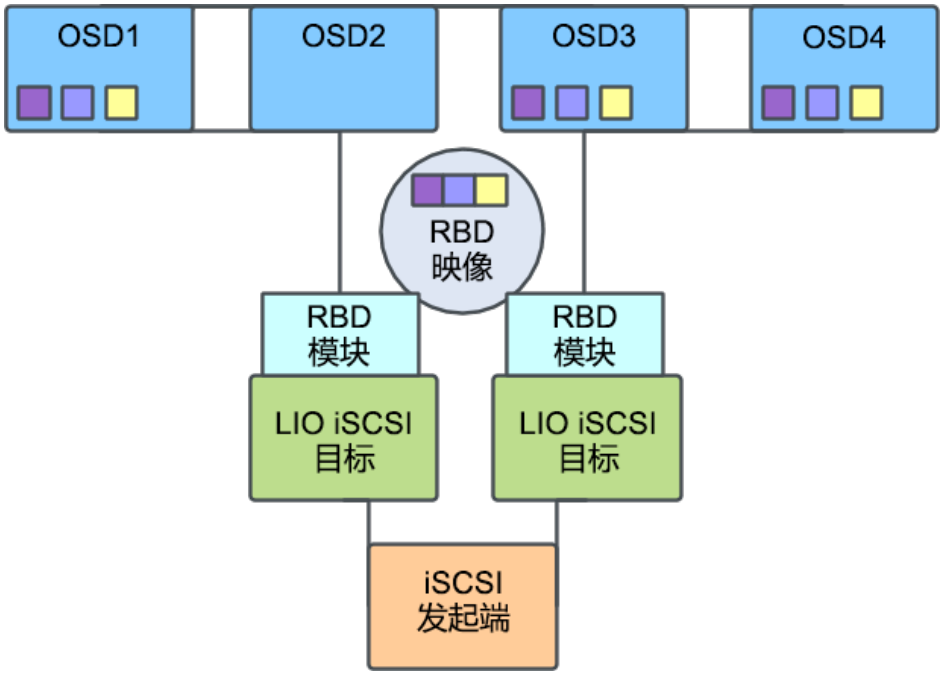

图 10.2： 包含多个 ISCSI 网关的 CEPH 集群

10.3 部署考虑事项

包含 `lrbd` 的最低 SUSE Enterprise Storage 配置包括以下组件：

- 一个 Ceph 存储集群。该 Ceph 集群至少包括四台物理服务器，其中每台服务器至少托管八个对象存储守护进程 (OSD)。在此类配置中，有三个 OSD 节点额外充当监视器 (MON) 主机。
- 一台通过 `lrbd` 配置且运行 LIO iSCSI 目标的 iSCSI 目标服务器。
- 一台 iSCSI 发起程序主机，它运行 `open-iscsi` (Linux)、Microsoft iSCSI 发起程序 (Microsoft Windows) 或任何其他兼容的 iSCSI 发起程序实施。

使用 `lrbd` 的建议 SUSE Enterprise Storage 生产配置包括：

- 一个 Ceph 存储集群。一个 Ceph 生产集群，它由任意数量（通常是 10 个以上）的 OSD 节点组成，其中每个节点通常运行 10-12 个对象存储守护进程 (OSD)，以及至少三台专用 MON 主机。
- 多台通过 `lrbd` 配置且运行 LIO iSCSI 目标的 iSCSI 目标服务器。为实现 iSCSI 故障转移和负载平衡，这些服务器必须运行支持 `target_core_rbd` 模块的内核。可通过 SUSE Linux Enterprise Server 维护渠道获取更新包。
- 任意数量的 iSCSI 发起程序主机，这些主机运行 `open-iscsi` (Linux)、Microsoft iSCSI 发起程序 (Microsoft Windows) 或任何其他兼容的 iSCSI 发起程序实施。

10.4 安装和配置

本节介绍在 SUSE Enterprise Storage 的基础上安装和配置 iSCSI 网关的步骤。

10.4.1 将 iSCSI 网关部署到 Ceph 集群

您可以在 Ceph 集群部署期间部署 iSCSI 网关，或者使用 DeepSea 将其添加到现有集群。

要在集群部署期间加入 iSCSI 网关，请参见第 4.5.1.2 节 “角色指定”。

要将 iSCSI 网关添加到现有集群，请参见《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.2 节 “为节点添加新的角色”。

10.4.2 创建 RBD 映像

RBD 映像创建于 Ceph 存储区中，随后会导出到 iSCSI。建议为此使用专用的 RADOS 存储池。您可以在能使用 Ceph `rbd` 命令行实用程序连接到存储集群的任何主机上创建卷。这需要客户端至少有一个精简的 ceph.conf 配置文件，以及相应的 CephX 身份验证身份凭证。

要通过 iSCSI 创建一个随后可供导出的新卷，请使用 `rbd create` 命令并指定卷大小（以 MB 为单位）。例如，要在名为 `iscsi` 的存储池中创建名为 `testvol` 的 100 GB 卷，请运行：

```
root # rbd --pool iscsi create --size=102400 testvol
```

上述命令将以默认格式 2 创建一个 RBD 卷。

注意

从 SUSE Enterprise Storage 3 开始，默认卷格式为 2，格式 1 已弃用。但是，您仍可以使用 `--image-format 1` 选项创建采用已弃用格式 1 的卷。

10.4.3 通过 iSCSI 导出 RBD 映像

要通过 iSCSI 导出 RBD 映像，请使用 `lrbd` 实用程序。`lrbd` 可用于创建、查看和修改采用 JSON 格式的 iSCSI 目标配置。

提示：将更改导入到 openATTIC

DeepSea 和 openATTIC 中看不到使用 `lrbd` 命令对 iSCSI 网关配置所做的任何更改。要导入您的手动更改，需要将 iSCSI 网关配置导出到一个文件：

```
root@minion > lrbd -o /tmp/lrbd.conf
```

然后将其复制到 Salt Master 以便 DeepSea 和 openATTIC 可以看到该文件：

```
root@minion > scp /tmp/lrbd.conf ses5master:/srv/salt/ceph/igw/cache/
lrbd.conf
```

最后，编辑 `/srv/pillar/ceph/stack/global.yml` 并做如下设置：

```
igw_config: default-ui
```

要编辑配置，请使用 `lrbd -e` 或 `lrbd --edit`。此命令将调用 `EDITOR` 环境变量定义的默认编辑器。可以通过同时设置 `-E` 选项和 `-e` 来覆盖此行为。

下面的示例配置针对以下情境：

- 有两个分别名为 `iscsi1.example.com` 和 `iscsi2.example.com` 的 iSCSI 网关主机，
- 使用 `iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol` 的 iSCSI 限定名称 (IQN) 定义单个 iSCSI 目标，

- 包含单个 iSCSI 逻辑单元 (LU)，
- 以 RADOS 存储池 `rbd` 中名为 `testvol` 的 RBD 映像为基础，
- 通过名为“east”和“west”的两个门户导出目标：

```
{
    "auth": [
        {
            "target": "iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol",
            "authentication": "none"
        }
    ],
    "targets": [
        {
            "target": "iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol",
            "hosts": [
                {
                    "host": "iscsi1.example.com",
                    "portal": "east"
                },
                {
                    "host": "iscsi2.example.com",
                    "portal": "west"
                }
            ]
        }
    ],
    "portals": [
        {
            "name": "east",
            "addresses": [
                "192.168.124.104"
            ]
        },
        {
            "name": "west",
            "addresses": [
                "192.168.124.105"
```

```
            ]
        }
    ],
    "pools": [
        {
            "pool": "rbd",
            "gateways": [
                {
                    "target": "iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol",
                    "tpg": [
                        {
                            "image": "testvol"
                        }
                    ]
                }
            ]
        }
    ]
    }
```

请注意，每当您在配置中引用某个主机名时，此主机名必须与 iSCSI 网关的 `uname -n` 命令输出相匹配。

编辑的 JSON 存储在每个存储池的单个 RADOS 对象的扩展属性 (xattrs) 中。此对象适用于在其中编辑了 JSON 的网关主机，以及与同一 Ceph 集群连接的所有网关主机。不会将任何配置信息存储在 `lrbd` 网关本地。

要激活该配置，请将其存储在 Ceph 集群中，并以 `root` 身份执行以下其中一项操作：

- 从命令行运行 `lrbd` 命令（不带附加选项），

或者

- 使用 `service lrbd restart` 重启动 `lrbd` 服务。

`lrbd` “服务”不会运行任何后台守护进程，只是调用 `lrbd` 命令。此类服务称为“一次性”服务。

您还应该启用在系统启动时自动配置 `lrbd` 的功能。为此，请运行 `systemctl enable lrbd` 命令。

上述配置反映了一种简单的单一网关设置。`lrbd` 配置可能要复杂且强大得多。`lrbd` RPM 包随附了大量的配置示例，安装后，您可以通过查看 `/usr/share/doc/packages/lrbd/samples` 目录的内容来参考这些示例。也可以从 https://github.com/SUSE/lrbd/tree/master/samples 获取这些示例。

10.4.4 可选设置

以下设置可能对某些环境有用。用于映像的属性有 `uuid`、`lun`、`retries`、`sleep` 和 `retry_errors`。使用前两个属性（`uuid` 和 `lun`）可以硬编码特定映像的“uuid”或“lun”。您可为映像指定这两者中的任一属性。`retries`、`sleep` 和 `retry_errors` 会影响映射 rbd 映像的尝试。

```
"pools": [
    {
        "pool": "rbd",
        "gateways": [
        {
        "host": "igw1",
        "tpg": [
                        {
                            "image": "archive",
                            "uuid": "12345678-abcd-9012-efab-345678901234",
                            "lun": "2",
                            "retries": "3",
                            "sleep": "4",
                            "retry_errors": [ 95 ],
                            [...]
                        }
                    ]
                }
        ]
    }
]
```

10.4.5 高级设置

可以为 `lrbd` 配置随后将传递给 LIO I/O 目标的高级参数。这些参数会划分为 iSCSI 和后备存储组件，然后可分别在 `lrbd` 配置的“targets”和“tpg”段落中指定。

警告

不建议更改这些参数的默认设置。

```
"targets": [
    {
        [...]
        "tpg_default_cmdsn_depth": "64",
        "tpg_default_erl": "0",
        "tpg_login_timeout": "10",
        "tpg_netif_timeout": "2",
        "tpg_prod_mode_write_protect": "0",
    }
]
```

选项说明如下：

tpg_default_cmdsn_depth

默认的 CmdSN（命令顺序号）深度。限制 iSCSI 发起程序在任意时刻可拥有的未处理请求数量。

tpg_default_erl

默认的错误恢复级别。

tpg_login_timeout

登录超时值（以秒为单位）。

tpg_netif_timeout

NIC 故障超时（以秒为单位）。

tpg_prod_mode_write_protect

如果设置为 1，则阻止写入到 LUN。

```
"pools": [
```

```
    {
        "pool": "rbd",
        "gateways": [
        {
        "host": "igw1",
        "tpg": [
                    {
                        "image": "archive",
                        "backstore_block_size": "512",
                        "backstore_emulate_3pc": "1",
                        "backstore_emulate_caw": "1",
                        "backstore_emulate_dpo": "0",
                        "backstore_emulate_fua_read": "0",
                        "backstore_emulate_fua_write": "1",
                        "backstore_emulate_model_alias": "0",
                        "backstore_emulate_rest_reord": "0",
                        "backstore_emulate_tas": "1",
                        "backstore_emulate_tpu": "0",
                        "backstore_emulate_tpws": "0",
                        "backstore_emulate_ua_intlck_ctrl": "0",
                        "backstore_emulate_write_cache": "0",
                        "backstore_enforce_pr_isids": "1",
                        "backstore_fabric_max_sectors": "8192",
                        "backstore_hw_block_size": "512",
                        "backstore_hw_max_sectors": "8192",
                        "backstore_hw_pi_prot_type": "0",
                        "backstore_hw_queue_depth": "128",
                        "backstore_is_nonrot": "1",
                        "backstore_max_unmap_block_desc_count": "1",
                        "backstore_max_unmap_lba_count": "8192",
                        "backstore_max_write_same_len": "65535",
                        "backstore_optimal_sectors": "8192",
                        "backstore_pi_prot_format": "0",
                        "backstore_pi_prot_type": "0",
                        "backstore_queue_depth": "128",
                        "backstore_unmap_granularity": "8192",
                        "backstore_unmap_granularity_alignment": "4194304"
                    }
```

```
                ]
            }
        ]
    }
]
```

选项说明如下：

backstore_block_size

底层设备的块大小。

backstore_emulate_3pc

如果设置为 1，则启用“第三方复制”。

backstore_emulate_caw

如果设置为 1，则启用“比较并写入”。

backstore_emulate_dpo

如果设置为 1，则打开“禁用页面写出”。

backstore_emulate_fua_read

如果设置为 1，则启用“强制单元读取访问”。

backstore_emulate_fua_write

如果设置为 1，则启用“强制单元写入访问”。

backstore_emulate_model_alias

如果设置为 1，则使用后端设备名称作为模型别名。

backstore_emulate_rest_reord

如果设置为 0，则队列算法修饰符的重新排序受限。

backstore_emulate_tas

如果设置为 1，则启用“任务已中止状态”。

backstore_emulate_tpu

如果设置为 1，则启用“精简配置 - 取消映射”。

backstore_emulate_tpws

如果设置为 1，则启用“精简配置 - 写入相同内容”。

backstore_emulate_ua_intlck_ctrl

如果设置为 1，则启用“单元警告联锁”。

backstore_emulate_write_cache

如果设置为 1，则打开“启用写入快速缓存”。

backstore_enforce_pr_isids

如果设置为 1，则强制永久性预留 ISID。

backstore_fabric_max_sectors

结构一次可以传输的最大扇区数。

backstore_hw_block_size

硬件块大小（以字节为单位）。

backstore_hw_max_sectors

硬件一次可以传输的最大扇区数。

backstore_hw_pi_prot_type

如果值为非零，则在底层硬件上启用 DIF 保护。

backstore_hw_queue_depth

硬件队列深度。

backstore_is_nonrot

如果设置为 1，则后备存储为非旋转设备。

backstore_max_unmap_block_desc_count

UNMAP 的最大块描述符数。

backstore_max_unmap_lba_count:

UNMAP 的最大 LBA 数。

backstore_max_write_same_len

WRITE_SAME 的最大长度。

backstore_optimal_sectors

扇区中的最佳请求大小。

backstore_pi_prot_format

DIF 保护格式。

backstore_pi_prot_type
: DIF 保护类型。

backstore_queue_depth
: 队列深度。

backstore_unmap_granularity
: UNMAP 粒度。

backstore_unmap_granularity_alignment
: UNMAP 粒度对齐。

对于目标，可以使用 `tpg` 属性优化内核参数。使用此选项时要特别小心。

```
"targets": [
{
    "host": "igw1",
    "target": "iqn.2003-01.org.linux-iscsi.generic.x86:sn.abcdefghijk",
    "tpg_default_cmdsn_depth": "64",
    "tpg_default_erl": "0",
    "tpg_login_timeout": "10",
    "tpg_netif_timeout": "2",
    "tpg_prod_mode_write_protect": "0",
    "tpg_t10_pi": "0"
}
```

提示

如果站点需要静态指定的 LUN，请对每个 LUN 指定编号。

10.5 使用 tcmu-runner 导出 RADOS 块设备映像

从版本 5 开始，SUSE Enterprise Storage 针对 `tcmu-runner` 随附了一个用户空间 RBD 后端（有关详细信息，请参见 `man 8 tcmu-runner`）。

警告：技术预览

基于 `tcmu-runner` 的 iSCSI 网关部署目前以技术预览的方式提供。有关使用 `lrbd` 进行基于内核的 iSCSI 网关部署的指导，请参见第 10 章 “安装 iSCSI 网关”。

与基于内核的 `lrbd` iSCSI 网关部署不同，基于 `tcmu-runner` 的 iSCSI 网关不支持多路径 I/O 或 SCSI 永久性预留。

由于 DeepSea 和 openATTIC 目前不支持 `tcmu-runner` 部署，因此您需要手动管理安装、部署和监视。

10.5.1 安装

在 iSCSI 网关节点上，安装 SUSE Enterprise Storage 5 媒体中的 `tcmu-runner-handler-rbd` 包，以及 `libtcmu1` 和 `tcmu-runner` 包依赖项。安装用于配置的 `targetcli-fb` 包。请注意，`targetcli-fb` 包与“non-fb”版本的 `targetcli` 包不兼容。

确认 `tcmu-runner` `systemd` 服务正在运行：

```
root # systemctl enable tcmu-runner
tcmu-gw:~ # systemctl status tcmu-runner
● tcmu-runner.service - LIO Userspace-passthrough daemon
  Loaded: loaded (/usr/lib/systemd/system/tcmu-runner.service; static; vendor
  preset: disabled)
    Active: active (running) since ...
```

10.5.2 配置和部署

在现有的 Ceph 集群上创建一个 RADOS 块设备映像。在以下示例中，我们将使用“rbd”存储池中名为“tcmu-lu”的 10G 映像。

创建 RADOS 块设备映像后，请运行 `targetcli`，并确保 tcmu-runner RBD 处理程序（插件）可用：

```
root # targetcli
targetcli shell version 2.1.fb46
```

```
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> ls
o- / ..................................... [...]
  o- backstores ......................... [...]
...
  | o- user:rbd ......... [Storage Objects: 0]
```

为 RBD 映像创建一个后备存储配置项：

```
/> cd backstores/user:rbd
/backstores/user:rbd> create tcmu-lu 10G /rbd/tcmu-lu
Created user-backed storage object tcmu-lu size 10737418240.
```

创建一个 iSCSI 传输配置项。在下面的示例中，目标限定名“iqn.2003-01.org.linux-iscsi.tcmu-gw.x8664:sn.cb3d2a3a”由 `targetcli` 自动生成，用作唯一的 iSCSI 目标标识符：

```
/backstores/user:rbd> cd /iscsi
/iscsi> create
Created target iqn.2003-01.org.linux-iscsi.tcmu-gw.x8664:sn.cb3d2a3a.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
```

针对您要连接到目标的 iSCSI 发起程序创建一个 ACL 项。在下面的示例中，使用了发起程序 IQN“iqn.1998-01.com.vmware:esxi-872c4888”：

```
/iscsi> cd
iqn.2003-01.org.linux-iscsi.tcmu-gw.x8664:sn.cb3d2a3a/tpg1/acls/
/iscsi/iqn.20...a3a/tpg1/acls> create iqn.1998-01.com.vmware:esxi-872c4888
```

最后，将前面创建的 RBD 后备存储配置链接到 iSCSI 目标：

```
/iscsi/iqn.20...a3a/tpg1/acls> cd ../luns
/iscsi/iqn.20...a3a/tpg1/luns> create /backstores/user:rbd/tcmu-lu
Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.1998-01.com.vmware:esxi-872c4888
```

退出外壳以保存现有配置：

```
/iscsi/iqn.20...a3a/tpg1/luns> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json
```

10.5.3 用法

使用前面配置的 IQN 和主机名，从 iSCSI 发起程序（客户端）节点连接到新供应的 iSCSI 目标。

11 安装 CephFS

Ceph 文件系统 (CephFS) 是符合 POSIX 标准的文件系统，它使用 Ceph 存储集群来存储其数据。CephFS 使用与 Ceph 块设备相同的集群系统：Ceph 对象存储及其 S3 和 Swift API 或本机绑定 (`librados`)。

要使用 CephFS，需有一个正在运行的 Ceph 存储集群，并至少要有一台正在运行的 Ceph 元数据服务器。

11.1 支持的 CephFS 方案和指导

借助 SUSE Enterprise Storage，SUSE 引入了对使用扩展和分布式组件 CephFS 的众多方案的正式支持。本节介绍硬性限制，并提供有关建议用例的指导。

支持的 CephFS 部署必须符合以下要求：

- 至少有一台元数据服务器。SUSE 建议部署多个具有 MDS 角色的节点。其中只有一个节点是“主动”节点，其余节点是“被动”节点。从客户端装入 CephFS 时，请记得在 `mount` 命令中指定所有 MDS 节点。
- 在此版本中，CephFS 快照默认未启用，不受支持。
- 客户端基于 SUSE Linux Enterprise Server 12 SP2 或 SP3，使用 `cephfs` 内核模块驱动程序。不支持 FUSE 模块。
- SUSE Enterprise Storage 中不支持 CephFS 配额，因为仅在 FUSE 客户端中实施配额支持。
- CephFS 支持 http://docs.ceph.com/docs/jewel/cephfs/file-layouts/ 中所述的文件布局更改。但是，尽管文件系统可由任何客户端装入，但无法将新数据池添加到现有的 CephFS 文件系统 (`ceph mds add_data_pool`)。只能在文件系统已卸载时添加这些存储池。

11.2 Ceph 元数据服务器

Ceph 元数据服务器 (MDS) 存储 CephFS 的元数据。Ceph 块设备和 Ceph 对象存储不使用 MDS。POSIX 文件系统用户可通过 MDS 执行基本命令（例如 `ls` 或 `find`），因而不会对 Ceph 存储集群施加巨大的负担。

11.2.1 添加元数据服务器

您可以根据第 4.3 节 “集群部署”中所述，在初始集群部署过程中部署 MDS；或者根据《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.1 节 “添加新的集群节点”中所述，将 MDS 添加到已部署的集群。

部署 MDS 后，请在部署 MDS 的服务器的防火墙设置中允许 `Ceph OSD/MDS` 服务：启动 `yast`，导航到 Security and Users（安全性和用户） › Firewall（防火墙） › Allowed Services（允许的服务），然后在 Service to Allow（要允许的服务）下拉菜单中选择 Ceph OSD/MDS。如果不允许在 Ceph MDS 节点中传送完整流量，则即使其他操作可以正常进行，装入文件系统也会失败。

11.2.2 配置元数据服务器

可以通过在 `ceph.conf` 配置文件中插入相关的选项来微调 MDS 的行为。

MDS 快速缓存大小

`mds cache memory limit`
: MDS 将针对其快速缓存实施软内存限制（以字节数为单位）。管理员应使用此项设置取代旧的 `mds cache size` 设置。默认大小为 1 GB。

`mds cache reservation`
: MDS 快速缓存要维护的快速缓存预留（内存或 inode）。当 MDS 开始接近其预留大小时，会撤消客户端状态，直到其快速缓存大小收缩到可恢复预留为止。默认值为 0.05。

有关 MDS 相关配置选项的详细列表，请参见 http://docs.ceph.com/docs/master/cephfs/mds-config-ref/ 。

有关 MDS 日记程序配置选项的详细列表，请参见 http://docs.ceph.com/docs/master/cephfs/journaler/ 。

11.3 CephFS

部署至少包含一台 Ceph 元数据服务器的正常 Ceph 存储集群后，可以创建并装入 Ceph 文件系统。请确保客户端可连接到网络，并具有正确的身份验证密钥环。

11.3.1 创建 CephFS

CephFS 至少需要两个 RADOS 存储池：一个用于存储数据，另一个用于存储元数据。配置这些存储池时，可以考虑：

- 对元数据池使用较高的复制级别，因为丢失此池中的任何数据都可能会导致整个文件系统不可访问。
- 对元数据池使用延迟较低的存储，例如 SSD，因为在客户端上执行文件系统操作时，这样可以改善用户可察觉到的延迟。

在 `policy.cfg` 中指定 `role-mds` 时，会自动创建所需的存储池。在设置元数据服务器之前，可以手动创建存储池 `cephfs_data` 和 `cephfs_metadata`，以手动优化性能。如果这些存储池已存在，DeepSea 将不会创建它们。

有关管理存储池的详细信息，请参见《管理指南》, 第 7 章 “管理存储池”。

要使用默认设置创建两个需要用于 CephFS 的存储池（例如“cephfs_data”和“cephfs_metadata”），请运行以下命令：

```
root # ceph osd pool create cephfs_data pg_num
root # ceph osd pool create cephfs_metadata pg_num
```

可以使用 EC 存储池取代副本池。建议仅针对低性能要求和不经常发生的随机访问（例如冷存储、备份和存档）使用 EC 存储池。EC 存储池中的 CephFS 需要启用 BlueStore，并且必须为存储池设置 `allow_ec_overwrite` 选项。可以运行 `ceph osd pool set ec_pool allow_ec_overwrites true` 来设置此选项。

纠删码会明显增大文件系统操作的开销，尤其是执行小规模更新时。使用纠删码作为容错机制必然会产生这种开销。这种代价抵消了明显减小的存储空间开销。

创建存储池时，可以使用 `ceph fs new` 命令来启用文件系统：

```
root # ceph fs new fs_name metadata_pool_name data_pool_name
```

例如：

```
root # ceph fs new cephfs cephfs_metadata cephfs_data
```

可以通过列出所有可用的 CephFS 来检查是否已创建文件系统：

```
root # ceph fs ls
 name: cephfs, metadata pool: cephfs_metadata, data pools: [cephfs_data]
```

创建文件系统后，MDS 将能够进入主动状态。例如，在单个 MDS 系统中：

```
root # ceph mds stat
e5: 1/1/1 up
```

提示：更多主题

可以在《管理指南》，第 13 章 “集群文件系统”中找到特定任务（例如装入、卸载和高级 CephFS 设置）的更多信息。

11.3.2 MDS 集群大小

一个 CephFS 实例可由多个主动 MDS 守护进程提供支持。指定给 CephFS 实例的所有主动 MDS 守护进程将在彼此之间分发文件系统的目录树，以此来分散并行客户端的负载。要为 CephFS 实例添加主动 MDS 守护进程，需要一个备用待机守护进程。请启动其他守护进程或使用现有待机实例。

以下命令将显示当前主动和被动 MDS 守护进程的数量。

```
root # ceph mds stat
```

以下命令将文件系统实例中的主动 MDS 的数量设置为两个。

```
root # ceph fs set fs_name max_mds 2
```

要在更新前收缩 MDS 集群，需要执行以下两个步骤。首先设置 `max_mds`，以只留一个实例：

```
root # ceph fs set fs_name max_mds 1
```

然后明确停用其他主动 MDS 守护进程：

```
root # ceph mds deactivate fs_name:rank
```

其中 `rank` 为文件系统实例的主动 MDS 守护进程的数量，范围介于 0 到 `max_mds` -1 之间。有关其他信息，请参见 http://docs.ceph.com/docs/luminous/cephfs/multimds/ 。

11.3.3 MDS 集群和更新

在 Ceph 更新期间，文件系统实例上的功能标识可能会发生变化（通常在添加新功能时发生）。不兼容的守护进程（例如旧版本）无法与不兼容的功能集搭配使用，并将拒绝启动。这意味着更新并重启动一个守护进程可能会导致尚未更新的其他所有守护进程都将停止并拒绝启动。出于此原因，我们建议将主动 MDS 集群的大小缩为一个，并在更新 Ceph 之前停止所有待机守护进程。此更新过程的手动步骤如下所述：

1. 使用 `zypper` 更新 Ceph 相关的包。
2. 按上述说明将主动 MDS 集群缩小至 1 个实例，并使用其在所有其他节点上的 `systemd` 单元停止所有待机 MDS 守护进程：

   ```
   root # systemctl stop ceph-mds\*.service ceph-mds.target
   ```

3. 然后才重启动其余一个 MDS 守护进程，以使其使用更新的二进制文件重启动。

   ```
   root # systemctl restart ceph-mds\*.service ceph-mds.target
   ```

4. 重启动所有其他 MDS 守护进程并重设置所需的 `max_mds` 设置。

   ```
   root # systemctl start ceph-mds.target
   ```

如果您使用 DeepSea，则在阶段 0 和 4 更新 `ceph` 包时，它会遵循此过程。当客户端装入 CephFS 实例并且正在进行 I/O 操作时，可能会执行此过程。不过请注意，当主动 MDS 重启动时，会有一个短暂的 I/O 暂停。客户端将自动恢复。

最好在更新 MDS 集群之前尽可能减少 I/O 负载。如此空闲的 MDS 集群将能更快地完成此更新。反之，在一个负载较重具有多个 MDS 守护进程的集群上，必须提前减少负载以防止进行中的 I/O 超出单个 MDS 守护进程的负载能力。

12 安装 NFS Ganesha

使用 NFS Ganesha 可通过 NFS 访问对象网关或 CephFS。SUSE Enterprise Storage 5 支持 NFS 版本 3 和 4。NFS Ganesha 在用户空间而不是内核空间中运行，直接与对象网关或 CephFS 交互。

12.1 准备

12.1.1 一般信息

要成功部署 NFS Ganesha，需要将 `role-ganesha` 添加到 `/srv/pillar/ceph/proposals/policy.cfg`。有关详细信息，请参见第 4.5.1 节 “`policy.cfg` 文件”。要使用 NFS Ganesha，还需要在 `policy.cfg` 中指定 `role-rgw` 或 `role-mds`。

尽管可以在现有的 Ceph 节点上安装并运行 NFS Ganesha 服务器，但建议在能够访问 Ceph 集群的专用主机上运行该服务器。客户端主机通常不是集群的一部分，但需要能够通过网络访问 NFS Ganesha 服务器。

要在完成初始安装后随时启用 NFS Ganesha 服务器，请将 `role-ganesha` 添加到 `policy.cfg`，并至少重新运行 DeepSea 阶段 2 和 4。有关详细信息，请参见第 4.3 节 “集群部署”。

NFS Ganesha 是通过 NFS Ganesha 节点上的 `/etc/ganesha/ganesha.conf` 文件配置的。但是，每次执行 DeepSea 阶段 4，都会重写此文件。因此，建议编辑 Salt 使用的模板，即 Salt Master 上的 `/srv/salt/ceph/ganesha/files/ganesha.conf.j2` 文件。有关配置文件的详细信息，请参见《管理指南》, 第 14 章 “NFS Ganesha：通过 NFS 导出 Ceph 数据”, 第 14.2 节 “配置”。

12.1.2 要求摘要

在执行 DeepSea 阶段 2 和 4 来安装 NFS Ganesha 之前，必须满足以下要求：

- 至少为一个节点指定 `role-ganesha`。
- 对于每个 Minion，只能定义一个 `role-ganesha`。

- NFS Ganesha 需要对象网关或 CephFS 才能正常工作。
- 如果 NFS Ganesha 预期要使用对象网关来连接集群，则需要填充 Salt Master 上的 `/srv/pillar/ceph/rgw.sls`。

12.2 示例安装

此过程提供了一个安装示例，该示例使用 NFS Ganesha 的对象网关和 CephFS 文件系统抽象层 (FSAL)。

1. 先执行 DeepSea 阶段 0 和 1（如果尚未这样做），然后继续执行此过程。

```
root # salt-run state.orch ceph.stage.0
root # salt-run state.orch ceph.stage.1
```

2. 执行 DeepSea 的阶段 1 之后，编辑 `/srv/pillar/ceph/proposals/policy.cfg` 并添加下行

```
role-ganesha/cluster/NODENAME
```

将 `NODENAME` 替换为集群中某个节点的名称。
另外，请确保已指定 `role-mds` 和 `role-rgw`。

3. 创建文件 `/srv/pillar/ceph/rgw.sls` 并插入以下内容：

```
rgw_configurations:
  rgw:
    users:
      - { uid: "demo", name: "Demo", email: "demo@demo.nil" }
      - { uid: "demo1", name: "Demo1", email: "demo1@demo.nil" }
```

系统稍后会将这些用户创建为对象网关用户，并生成 API 密钥。稍后，您可在对象网关节点上运行 `radosgw-admin user list` 列出所有已创建的用户，并运行 `radosgw-admin user info --uid=demo` 获取有关单个用户的详细信息。
DeepSea 确保对象网关和 NFS Ganesha 都能接收 `rgw.sls` 的 `rgw` 段落中所列全部用户的身份凭证。

导出的 NFS 将在文件系统的第一级别上使用这些用户名；在本示例中，将导出路径 `/demo` 和 `/demo1`。

4. 至少执行 DeepSea 的阶段 2 和 4。建议运行中间的阶段 3。

```
root # salt-run state.orch ceph.stage.2
root # salt-run state.orch ceph.stage.3 # optional but recommended
root # salt-run state.orch ceph.stage.4
```

5. 通过从客户端节点装入 NFS 共享来校验 NFS Ganesha 是否正常工作：

```
root # mount -o sync -t nfs GANESHA_NODE:/ /mnt
root # ls /mnt
cephfs  demo  demo1
```

`/mnt` 应包含所有已导出的路径。CephFS 和两个对象网关用户的目录应该存在。对于用户拥有的每个桶，将导出路径 `/mnt/USERNAME/BUCKETNAME`。

12.3 高可用性主动-被动配置

本节提供一个示例来说明如何设置 NFS Ganesha 服务器的双节点主动-被动配置。该设置需要 SUSE Linux Enterprise High Availability Extension。两个节点分别名为 `earth` 和 `mars`。

有关 SUSE Linux Enterprise High Availability Extension 的详细信息，请参见 https://www.suse.com/documentation/sle-ha-12/。

12.3.1 基本安装

在此设置中，`earth` 的 IP 地址为 `192.168.1.1`，`mars` 的地址为 `192.168.1.2`。

此外，使用了两个浮动虚拟 IP 地址，这样无论服务在哪个物理节点上运行，客户端都可连接到服务。`192.168.1.10` 用于通过 Hawk2 进行集群管理，`192.168.2.1` 专门用于 NFS 导出项。这样，以后便可更轻松地应用安全限制。

以下过程介绍示例安装。https://www.suse.com/documentation/sle-ha-12/install-quick/data/install-quick.html 上提供了更多详细信息。

1. 在 Salt Master 上准备 NFS Ganesha 节点：

a. 在 Salt Master 上运行 DeepSea 阶段 0 和 1。

```
root@master # salt-run state.orch ceph.stage.0
root@master # salt-run state.orch ceph.stage.1
```

b. 在 `/srv/pillar/ceph/proposals/policy.cfg` 中为节点 `earth` 和 `mars` 指定 `role-ganesha`：

```
role-ganesha/cluster/earth*.sls
role-ganesha/cluster/mars*.sls
```

c. 在 Salt Master 上运行 DeepSea 阶段 3 和 4。

```
root@master # salt-run state.orch ceph.stage.3
root@master # salt-run state.orch ceph.stage.4
```

2. 在 `earth` 和 `mars` 上注册 SUSE Linux Enterprise High Availability Extension。

```
root # SUSEConnect -r ACTIVATION_CODE -e E_MAIL
```

3. 在两个节点上安装 `ha-cluster-bootstrap`：

```
root # zypper in ha-cluster-bootstrap
```

4. a. 在 `earth` 上初始化集群：

```
root@earth # ha-cluster-init
```

b. 让 `mars` 加入该集群：

```
root@mars # ha-cluster-join -c earth
```

5. 检查集群的状态。您应该会看到两个节点都已添加到集群中：

```
root@earth # crm status
```

6. 在这两个节点上，禁用引导时自动启动 NFS Ganesha 服务的功能：

```
root # systemctl disable nfs-ganesha
```

7. 在 `earth` 上启动 `crm` 外壳：

```
root@earth # crm configure
```

后续命令在 crm 外壳中执行。

8. 在 `earth` 上，运行 crm 外壳来执行以下命令，以便将 NFS Ganesha 守护进程的资源配置为 systemd 资源类型的克隆：

```
crm(live)configure# primitive nfs-ganesha-server systemd:nfs-ganesha \
op monitor interval=30s
crm(live)configure# clone nfs-ganesha-clone nfs-ganesha-server meta
 interleave=true
crm(live)configure# commit
crm(live)configure# status
    2 nodes configured
    2 resources configured

    Online: [ earth mars ]

    Full list of resources:
         Clone Set: nfs-ganesha-clone [nfs-ganesha-server]
         Started:  [ earth mars ]
```

9. 使用 crm 外壳创建一个原始 IPAddr2：

```
crm(live)configure# primitive ganesha-ip IPaddr2 \
params ip=192.168.2.1 cidr_netmask=24 nic=eth0 \
op monitor interval=10 timeout=20

crm(live)# status
Online: [ earth mars  ]
Full list of resources:
 Clone Set: nfs-ganesha-clone [nfs-ganesha-server]
     Started: [ earth mars ]
 ganesha-ip    (ocf::heartbeat:IPaddr2):    Started earth
```

10. 为了在 NFS Ganesha 服务器与浮动虚拟 IP 之间建立关系，我们使用了共置和顺序约束。

```
crm(live)configure# colocation ganesha-ip-with-nfs-ganesha-server inf:
 ganesha-ip nfs-ganesha-clone
crm(live)configure# order ganesha-ip-after-nfs-ganesha-server Mandatory:
 nfs-ganesha-clone ganesha-ip
```

11. 从客户端使用 `mount` 命令，以确保完成集群设置：

```
root # mount -t nfs -v -o sync,nfsvers=4 192.168.2.1:/ /mnt
```

12.3.2 清理资源

如果其中一个节点（例如 `earth`）上发生 NFS Ganesha 故障，请解决问题并清理资源。
如果 NFS Ganesha 在 `mars` 上发生故障，则只有在清理资源之后，该资源才能故障回复到 `earth`。
要清理资源，请执行以下命令：

```
root@earth # crm resource cleanup nfs-ganesha-clone earth
root@earth # crm resource cleanup ganesha-ip earth
```

12.3.3 设置 Ping 资源

有时，服务器可能由于网络问题而无法访问客户端。Ping 资源可以检测并缓解此问题。配置此资源的操作是可选的。

1. 定义 ping 资源：

```
crm(live)configure# primitive ganesha-ping ocf:pacemaker:ping \
        params name=ping dampen=3s multiplier=100
 host_list="CLIENT1 CLIENT2" \
        op monitor interval=60 timeout=60 \
        op start interval=0 timeout=60 \
        op stop interval=0 timeout=60
```

`host_list` 是以空格分隔的 IP 地址列表。系统将会定期 ping 这些 IP 地址，以检查网络中断问题。如果某个客户端必须始终能够访问 NFS 服务器，请将该客户端添加到 `host_list`。

2. 创建克隆资源：

```
crm(live)configure# clone ganesha-ping-clone ganesha-ping \
        meta interleave=true
```

3. 以下命令会创建 NFS Ganesha 服务的约束。当 `host_list` 不可访问时，此约束会强制服务转移到另一节点。

```
crm(live)configure# location nfs-ganesha-server-with-ganesha-ping
        nfs-ganesha-clone \
        rule -inf: not_defined ping or ping lte 0
```

12.3.4 NFS Ganesha HA 和 DeepSea

DeepSea 不支持配置 NFS Ganesha HA。为了防止在配置 NFS Ganesha HA 之后 DeepSea 发生故障，请从 DeepSea 阶段 4 中排除 NFS Ganesha 服务的启动和停止操作：

1. 将 `/srv/salt/ceph/ganesha/default.sls` 复制到 `/srv/salt/ceph/ganesha/ha.sls`。

2. 从 `/srv/salt/ceph/ganesha/ha.sls` 中删除 `.service` 项，使文件内容如下所示：

```
include:
- .keyring
- .install
- .configure
```

3. 将下行添加到 `/srv/pillar/ceph/stack/global.yml`：

```
ganesha_init: ha
```

12.4 更多信息

更多信息可以在《管理指南》, 第 14 章 “NFS Ganesha：通过 NFS 导出 Ceph 数据”中找到。

13 通过 Samba 导出 CephFS

本章介绍如何通过 Samba/CIFS 共享导出 CephFS。可在 Windows* 客户端中使用 Samba 共享。

警告：技术预览

从 SUSE Enterprise Storage 5 开始，导出 Samba 共享被视为一项技术预览，不再受支持。

13.1 示例安装

导出 CephFS 是一项预览技术，不支持该功能。要导出 Samba 共享，需要在一个集群节点上手动安装 Samba 并对其进行配置。可以通过 CTDB 和 SUSE Linux Enterprise High Availability Extension 提供故障转移功能。

1. 请确保集群中已存在一个正常工作的 CephFS。有关详细信息，请参见第 11 章 “安装 CephFS”。
2. 在 Salt Master 上创建一个特定于 Samba 网关的密钥环，并将其复制到 Samba 网关节点：

   ```
   root@master # ceph auth get-or-create client.samba.gw mon 'allow r' \
       osd 'allow *' mds 'allow *' -o ceph.client.samba.gw.keyring
   root@master # scp ceph.client.samba.gw.keyring SAMBA_NODE:/etc/ceph/
   ```

   将 `SAMBA_NODE` 替换为 Samba 网关节点的名称。
3. 在 Samba 网关节点上执行以下步骤。在 Samba 网关节点上安装 Samba 守护进程：

   ```
   root # zypper in samba samba-ceph
   ```
4. 编辑 `/etc/samba/smb.conf` 并添加以下段落：

   ```
   [SHARE_NAME]
   ```

```
        path = /
        vfs objects = ceph
        ceph:config_file = /etc/ceph/ceph.conf
        ceph: user_id = samba.gw
        read only = no
```

5. 启动并启用 Samba 守护进程：

```
root # systemctl start smb.service
root # systemctl enable smb.service
root # systemctl start nmb.service
root # systemctl enable nmb.service
```

13.2 高可用性配置

本节提供一个示例来说明如何设置 Samba 服务器的双节点高可用性配置。该设置需要 SUSE Linux Enterprise High Availability Extension。两个节点分别名为 `earth` (`192.168.1.1`) 和 `mars` (`192.168.1.2`)。

有关 SUSE Linux Enterprise High Availability Extension 的详细信息，请参见 https://www.suse.com/documentation/sle-ha-12/ 。

此外，使用两个浮动虚拟 IP 地址可让客户端连接到服务，不管该服务在哪个物理节点上运行均如此。`192.168.1.10` 用于通过 Hawk2 进行集群管理，`192.168.2.1` 专门用于 CIFS 导出。这样，以后便可更轻松地应用安全限制。

以下过程介绍示例安装。https://www.suse.com/documentation/sle-ha-12/install-quick/data/install-quick.html 上提供了更多详细信息。

1. 在 Salt Master 上创建一个特定于 Samba 网关的密钥环，并将其复制到上述两个节点：

```
root@master # ceph auth get-or-create client.samba.gw mon 'allow r' \
    osd 'allow *' mds 'allow *' -o ceph.client.samba.gw.keyring
root@master # scp ceph.client.samba.gw.keyring earth:/etc/ceph/
root@master # scp ceph.client.samba.gw.keyring mars:/etc/ceph/
```

2. 准备好 `earth` 和 `mars`，以托管 Samba 服务：

a. 在继续下一步之前，请确保已安装以下包： `ctdb`、 `tdb-tools` 和 `samba`（smb 和 nmb 资源需要）。

```
root # zypper in ctdb tdb-tools samba samba-ceph
```

b. 确保服务 `ctdb`、 `smb` 和 `nmb` 已停止且未启用：

```
root # systemctl disable ctdb
root # systemctl disable smb
root # systemctl disable nmb
root # systemctl stop smb
root # systemctl stop nmb
```

c. 在所有节点上打开防火墙的端口 `4379`。这是为了使 CTDB 能够与其他集群节点通讯。

d. 在共享文件系统上为 CTDB 锁定创建一个目录：

```
root # mkdir -p /srv/samba/
```

3. 在 `earth` 上创建 Samba 的配置文件。这些文件稍后将自动同步到 `mars`。

a. 在 `/etc/ctdb/nodes` 中插入包含集群中每个节点的所有私用 IP 地址的所有节点：

```
192.168.1.1
192.168.1.2
```

b. 配置 Samba。在 `/etc/samba/smb.conf` 的 `[global]` 部分中添加以下行。使用所选的主机名取代“CTDB-SERVER”（集群中的所有节点将显示为一个此名称的大节点，以方便操作）：

```
[global]
    netbios name = CTDB-SERVER
    clustering = yes
    idmap config * : backend = tdb2
    passdb backend = tdbsam
```

```
    ctdbd socket = /var/lib/ctdb/ctdb.socket
```

有关 `csync2` 的详细信息，请参见 https://www.suse.com/documentation/sle-ha-12/singlehtml/book_sleha/book_sleha.html#pro.ha.installation.setup.csync2.start 。

4. 安装并引导 SUSE Linux Enterprise High Availability 集群。

   a. 在 `earth` 和 `mars` 上注册 SUSE Linux Enterprise High Availability Extension。

   ```
   root@earth # SUSEConnect -r ACTIVATION_CODE -e E_MAIL
   ```

   ```
   root@mars # SUSEConnect -r ACTIVATION_CODE -e E_MAIL
   ```

   b. 在两个节点上安装 `ha-cluster-bootstrap` ：

   ```
   root@earth # zypper in ha-cluster-bootstrap
   ```

   ```
   root@mars # zypper in ha-cluster-bootstrap
   ```

   c. 在 `earth` 上初始化集群：

   ```
   root@earth # ha-cluster-init
   ```

   d. 让 `mars` 加入该集群：

   ```
   root@mars # ha-cluster-join -c earth
   ```

5. 检查集群的状态。您应该会看到两个节点都已添加到集群中：

```
root@earth # crm status
2 nodes configured
1 resource configured

Online: [ earth mars ]

Full list of resources:
```

```
 admin-ip       (ocf::heartbeat:IPaddr2):       Started earth
```

6. 在 earth 上执行以下命令，以配置 CTDB 资源：

```
root@earth # crm configure
crm(live)configure# primitive ctdb ocf:heartbeat:CTDB params \
    ctdb_manages_winbind="false" \
    ctdb_manages_samba="false" \
    ctdb_recovery_lock="!/usr/lib64/ctdb/ctdb_mutex_ceph_rados_helper
        ceph client.samba.gw cephfs_metadata ctdb-mutex"
    ctdb_socket="/var/lib/ctdb/ctdb.socket" \
        op monitor interval="10" timeout="20" \
        op start interval="0" timeout="90" \
        op stop interval="0" timeout="100"
crm(live)configure# primitive nmb systemd:nmb \
    op start timeout="60" interval="0" \
    op stop timeout="60" interval="0" \
    op monitor interval="60" timeout="60"
crm(live)configure# primitive smb systemd:smb \
    op start timeout="60" interval="0" \
    op stop timeout="60" interval="0" \
    op monitor interval="60" timeout="60"
crm(live)configure# group g-ctdb ctdb nmb smb
crm(live)configure# clone cl-ctdb g-ctdb meta interleave="true"
crm(live)configure# commit
```

配置选项 ctdb_recovery_lock 中的二进制文件 /usr/lib64/ctdb/ctdb_mutex_ceph_rados_helper 依序包含参数 CLUSTER_NAME CEPHX_USER CEPH_POOL CEPHX_USER。

7. 添加集群 IP 地址：

```
crm(live)configure# primitive ip ocf:heartbeat:IPaddr2 params
 ip=192.168.2.1 \
    unique_clone_address="true" \
    op monitor interval="60" \
    meta resource-stickiness="0"
crm(live)configure# clone cl-ip ip \
    meta interleave="true" clone-node-max="2" globally-unique="true"
```

```
crm(live)configure# colocation col-with-ctdb 0: cl-ip cl-ctdb
crm(live)configure# order o-with-ctdb 0: cl-ip cl-ctdb
crm(live)configure# commit
```

如果 `unique_clone_address` 设置为 `true`，IPaddr2 资源代理将向指定的地址添加一个克隆 ID，从而导致出现三个不同的 IP 地址。这些地址通常是不需要的，但有助于实现负载平衡。有关此主题的更多信息，请参见https://www.suse.com/documentation/sle-ha-12/book_sleha/data/cha_ha_lb.html 。

8. 检查结果：

```
root@earth # crm status
Clone Set: base-clone [dlm]
     Started: [ factory-1 ]
     Stopped: [ factory-0 ]
 Clone Set: cl-ctdb [g-ctdb]
     Started: [ factory-1 ]
     Started: [ factory-0 ]
 Clone Set: cl-ip [ip] (unique)
     ip:0       (ocf:heartbeat:IPaddr2):       Started factory-0
     ip:1       (ocf:heartbeat:IPaddr2):       Started factory-1
```

9. 从客户端计算机进行测试。在 Linux 客户端上运行以下命令，以检查能否从系统复制文件以及将文件复制到系统：

```
root # smbclient //192.168.2.1/myshare
```

A 文档更新

本章列出了本文档自 SUSE Enterprise Storage 4 初始版本发布以来的内容更改。可以在 https://www.suse.com/documentation/ses-4/book_storage_admin/data/ap_adm_docupdate.html 中找到适用于先前版本的集群部署的相关更改。

文档在以下日期进行了更新：

- 第 A.1 节 “2018 年 10 月（SUSE Enterprise Storage 5.5 发布）”
- 第 A.2 节 “2017 年 11 月（维护性更新）”
- 第 A.3 节 “2017 年 10 月（SUSE Enterprise Storage 5 发布）”

A.1 2018 年 10 月（SUSE Enterprise Storage 5.5 发布）

一般更新

- 添加了第 3 章 “Ceph 管理节点 HA 设置” (Fate#325622)。
- 第 4.5.1.5 节 “部署加密的 OSD”和第 5.3 节 “升级期间加密 OSD”中介绍了部署和升级期间加密 OSD (Fate#321665)。

错误修复

- 第 9.1.1.2 节 “创建存储池（可选）”中介绍了需要复制的非数据对象网关池 (https://bugzilla.suse.com/show_bug.cgi?id=1095743)。
- 所有节点的 FQDN 必须可解析为公共网络 IP。请参见第 4.3 节 “集群部署”(https://bugzilla.suse.com/show_bug.cgi?id=1067113)。
- 第 4 章 “使用 DeepSea/Salt 部署”中添加了有关共享多个角色的提示 (https://bugzilla.suse.com/show_bug.cgi?id=1093824)。
- 添加了第 2.4 节 “元数据服务器节点”(https://bugzilla.suse.com/show_bug.cgi?id=1047230)。

- 为 openATTIC 手动编辑 `policy.cfg` (https://bugzilla.suse.com/show_bug.cgi?id=1073331)。
- 第 2.2 节 “监视器节点”中推荐使用 SSD 上的 /var/lib/ceph (https://bugzilla.suse.com/show_bug.cgi?id=1056322)。
- 添加了第 1.4 节 “BlueStore”和第 2.1.3 节 “BlueStore 的 WAL 和 DB 设备的建议大小” (https://bugzilla.suse.com/show_bug.cgi?id=1072502)。
- 第 4.5.1.5 节 “部署加密的 OSD”中扩展了加密 OSD 的部署 (https://bugzilla.suse.com/show_bug.cgi?id=1093003)。
- 步骤 13中将要删除的字节数增加至 4M (https://bugzilla.suse.com/show_bug.cgi?id=1093331)。
- 在第 4.3 节 “集群部署”中，防火墙中断了 DeepSea 阶段 (https://bugzilla.suse.com/show_bug.cgi?id=1090683)。
- 第 4.3 节 “集群部署”中添加了储存库列表 (https://bugzilla.suse.com/show_bug.cgi?id=1088170)。
- 第 5.5 节 “从 SUSE Enterprise Storage 4（`ceph-deploy` 部署）升级到版本 5”和第 5.6 节 “从 SUSE Enterprise Storage 4（Crowbar 部署）升级到版本 5”中添加了使用 `zypper` 第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5” 手动添加储存库的指导信息 (https://bugzilla.suse.com/show_bug.cgi?id=1073308)。
- 第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中添加了升级储存库的列表以及针对 DeepSea 的 `upgrade_init` 选项的简短说明 (https://bugzilla.suse.com/show_bug.cgi?id=1073372)。
- 添加了第 7.1.5 节 “在阶段 0 期间禁用更新和重引导”(https://bugzilla.suse.com/show_bug.cgi?id=1081524)。
- 修复了过程 5.2 “要对 Salt Master 节点应用的步骤”中的提示 (https://bugzilla.suse.com/show_bug.cgi?id=1084307)。
- 添加了第 2.12 节 “SUSE Enterprise Storage 以及其他 SUSE 产品”(https://bugzilla.suse.com/show_bug.cgi?id=1089717)。

- 《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.11 节 “自定义 ceph.conf 文件”中添加了一条有关对象网关配置部分的注释 (https://bugzilla.suse.com/show_bug.cgi?id=1089300)。
- 将 WAL/DB 片段添加到第 2.1.2 节 “最小磁盘大小” (https://bugzilla.suse.com/show_bug.cgi?id=1057797)。
- 会以动态方式计算 MON 的公用地址 (https://bugzilla.suse.com/show_bug.cgi?id=1089151)。
- 修复了第 5.5 节 “从 SUSE Enterprise Storage 4（ceph-deploy 部署）升级到版本 5”中的密钥环位置 (https://bugzilla.suse.com/show_bug.cgi?id=1073368)。
- 第 4.5.1.2 节 “角色指定”中提供了若干个帮助程序片段 (https://bugzilla.suse.com/show_bug.cgi?id=1061629)。
- 过程 5.2 “要对 Salt Master 节点应用的步骤”中导入了自定义 ceph.conf (https://bugzilla.suse.com/show_bug.cgi?id=1085443)。
- 第 2.1.1 节 “最低要求”中更新了为 BlueStore 部署推荐的 RAM 值 (https://bugzilla.suse.com/show_bug.cgi?id=1076385)。
- 在第 5.5 节 “从 SUSE Enterprise Storage 4（ceph-deploy 部署）升级到版本 5”和第 5.6 节 “从 SUSE Enterprise Storage 4（Crowbar 部署）升级到版本 5”中的导入命令后添加了升级 iSCSI 网关的手动步骤 (https://bugzilla.suse.com/show_bug.cgi?id=1073327)。
- 在第 10.4 节 “安装和配置”中，iSCSI 网关部署方式更新为 DeepSea (https://bugzilla.suse.com/show_bug.cgi?id=1073327)。
- 不支持 CephFS 配额。请参见第 11.1 节 “支持的 CephFS 方案和指导”(https://bugzilla.suse.com/show_bug.cgi?id=1077269)。
- 清零步骤中包含 9 个以上的分区，请参见步骤 13。(https://bugzilla.suse.com/show_bug.cgi?id=1050230)。

A.2 2017 年 11 月（维护性更新）

一般更新

- 添加了第 11.3.2 节 “MDS 集群大小”和第 11.3.3 节 “MDS 集群和更新”。

错误修复

- 第 4.3 节 “集群部署”的步骤 13 中增强了磁盘擦除策略 (https://bugzilla.suse.com/show_bug.cgi?id=1073897)。
- 第 5.4.1 节 “将 OSD 迁移到 BlueStore”中添加了有关解除安全措施的提示 (https://bugzilla.suse.com/show_bug.cgi?id=1073720)。
- 引用了过程 4.1 “运行部署阶段”中的第 4.2.2.1 节 “匹配 Minion 名称”以及《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.1 节 “添加新的集群节点”，来统一信息来源 (https://bugzilla.suse.com/show_bug.cgi?id=1073374)。
- 在第 5.2 节 “一般升级过程”中，仅更新 Salt Master 和 Minion，而不是更新所有包。因此，将会使用 `salt target state.apply ceph.updates.salt` 替换 `salt target state.apply ceph.updates` (https://bugzilla.suse.com/show_bug.cgi?id=1073373)。
- 添加了第 5.6 节 “从 SUSE Enterprise Storage 4（Crowbar 部署）升级到版本 5”（https://bugzilla.suse.com/show_bug.cgi?id=1073317 和 https://bugzilla.suse.com/show_bug.cgi?id=1073701）。
- 在第 4.2.2 节 “定位 Minion”中，已使用 `target` 替换“*”并扩展了定位介绍 (https://bugzilla.suse.com/show_bug.cgi?id=1068956)。
- 第 4.3 节 “集群部署”中添加了 Salt Minion 指纹校验 (https://bugzilla.suse.com/show_bug.cgi?id=1064045)。
- 《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.8 节 “通过 Salt 实现自动安装”中删除了复制示例 refactor.conf 文件的建议 (https://bugzilla.suse.com/show_bug.cgi?id=1065926)。
- 修复了过程 4.1 “运行部署阶段”中的网络配置 YAML 文件路径 (https://bugzilla.suse.com/show_bug.cgi?id=1067730)。

- 在过程 5.2 “要对 Salt Master 节点应用的步骤”中校验集群布局 (https://bugzilla.suse.com/show_bug.cgi?id=1067189)。
- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”和过程 5.2 “要对 Salt Master 节点应用的步骤”中添加了 `ceph osd set sortbitwise` (https://bugzilla.suse.com/show_bug.cgi?id=1067146)。
- `osd crush location` 现已弃用，重要：软件要求中以不同的方式对 `ceph.conf` 进行自定义 (https://bugzilla.suse.com/show_bug.cgi?id=1067381)。
- 已在第 4.5.1.2 节 “角色指定”中将“role-master”纠正为“role-admin”(https://bugzilla.suse.com/show_bug.cgi?id=1064056)。
- 修复了过程 4.1 “运行部署阶段”中的 `cluster.yml` 路径 (https://bugzilla.suse.com/show_bug.cgi?id=1066711)。
- 添加了第 12.3.4 节 “NFS Ganesha HA 和 DeepSea”(https://bugzilla.suse.com/show_bug.cgi?id=1058313)。
- 重新添加了第 2.7.2 节 “不同子网中的监视器节点”(https://bugzilla.suse.com/show_bug.cgi?id=1050115)。
- 文件 `deepsea_minions.sls` 只能包含一个 `deepsea_minions` 项。请参见过程 4.1 “运行部署阶段”(https://bugzilla.suse.com/show_bug.cgi?id=1065403)。
- 更改了第 4.3 节 “集群部署”的第一个过程中的步骤顺序 (https://bugzilla.suse.com/show_bug.cgi?id=1064770)。
- 澄清了第 5.4.1 节 “将 OSD 迁移到 BlueStore”的内容 (https://bugzilla.suse.com/show_bug.cgi?id=1063250)。

A.3 2017 年 10 月（SUSE Enterprise Storage 5 发布）

一般更新

- 添加了第 5.7 节 “从 SUSE Enterprise Storage 3 升级到版本 5” (Fate#323072)。
- 由于 DeepSea 的推出，删除了已过时的 `Crowbar` 安装工具。
- 由于 DeepSea 的推出，删除了已过时的 `ceph-deploy` 工具。
- 更新了第 2 章 “硬件要求和建议”（https://bugzilla.suse.com/show_bug.cgi?id=1029544 和 https://bugzilla.suse.com/show_bug.cgi?id=1042283 ）。
- 更新了第 12 章 “安装 NFS Ganesha”（https://bugzilla.suse.com/show_bug.cgi?id=1036495、https://bugzilla.suse.com/show_bug.cgi?id=1031444、FATE#322464 ）。
- 配置的 DeepSea 命名纲要已更改。请参见第 4.5.1.4 节 “配置指定”(https://bugzilla.suse.com/show_bug.cgi?id=1046108)。
- 可在 EC 存储池中使用 CephFS，请参见第 11.3.1 节 “创建 CephFS”(FATE#321617)。

错误修复

- 添加了第 10.5 节 “使用 `tcmu-runner` 导出 RADOS 块设备映像”(https://bugzilla.suse.com/show_bug.cgi?id=1064467)。
- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中改进了升级过程，在其中包含了 openATTIC 角色 (https://bugzilla.suse.com/show_bug.cgi?id=1064621)。
- 在步骤 4 中添加了过程 4.1 “运行部署阶段”的参考 (https://bugzilla.suse.com/show_bug.cgi?id=1064276)。
- 在过程 5.2 “要对 Salt Master 节点应用的步骤”中修改了升级过程（https://bugzilla.suse.com/show_bug.cgi?id=1061608 和 https://bugzilla.suse.com/show_bug.cgi?id=1048959 ）。
- 在《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.11 节 “自定义 `ceph.conf` 文件”中添加了 `rgw.conf` (https://bugzilla.suse.com/show_bug.cgi?id=1062109)。
- 在第 4.3 节 “集群部署”中将 DeepSea 安装移到了接近最后的步骤 (https://bugzilla.suse.com/show_bug.cgi?id=1056292)。

- 添加了第 4.5.1.5 节 “部署加密的 OSD”(https://bugzilla.suse.com/show_bug.cgi?id=1061751)。
- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中更新并简化了升级过程 (https://bugzilla.suse.com/show_bug.cgi?id=1059362)。
- 在重要：软件要求中指出升级之前需要检查 DeepSea 版本 (https://bugzilla.suse.com/show_bug.cgi?id=1059331)。
- 在第 7.1 节 “使用自定义的配置文件”中指出需要对自定义 .sls 文件加上 `custom-` 前缀 (https://bugzilla.suse.com/show_bug.cgi?id=1048568)。
- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中添加了一条有关密钥功能不匹配的注释 (https://bugzilla.suse.com/show_bug.cgi?id=1054186)。
- 在第 4.3 节 “集群部署”中合并了冗余列表项目 (https://bugzilla.suse.com/show_bug.cgi?id=1055140)。
- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中添加了一条有关集群升级可能需要花费很长时间的注释 (https://bugzilla.suse.com/show_bug.cgi?id=1054079)。
- 必须使用 `deepsea_minions:` 指定 Salt Minion 的目标 (https://bugzilla.suse.com/show_bug.cgi?id=1054229)。
- 在过程 5.2 “要对 Salt Master 节点应用的步骤”中的导入步骤后插入了运行阶段 1 (https://bugzilla.suse.com/show_bug.cgi?id=1054155)。
- 添加了《管理指南》, 第 1 章 “Salt 集群管理”, 第 1.11 节 “自定义 `ceph.conf` 文件”(https://bugzilla.suse.com/show_bug.cgi?id=1052806)。
- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中添加了缺少的步骤 (https://bugzilla.suse.com/show_bug.cgi?id=1052597)。
- 在第 12.2 节 “示例安装”中纠正了 `radosgw-admin` 命令语法 (https://bugzilla.suse.com/show_bug.cgi?id=1052698)。
- 在过程 5.1 “要对所有集群节点应用的步骤（包括 Calamari 节点）”中指出，升级期间“salt”不是 Salt Master 的必需主机名 (https://bugzilla.suse.com/show_bug.cgi?id=1052907)。

- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”的“重要说明”段落中改善了措辞和句式 (https://bugzilla.suse.com/show_bug.cgi?id=1052147)。
- 在过程 5.2 “要对 Salt Master 节点应用的步骤”中添加了导入期间手动指定角色的注释 (https://bugzilla.suse.com/show_bug.cgi?id=1050554)。
- 添加了第 5.4.1 节 “将 OSD 迁移到 BlueStore”(https://bugzilla.suse.com/show_bug.cgi?id=1052210)。
- 在过程 5.2 “要对 Salt Master 节点应用的步骤”中详细解释了 `salt-run populate.engulf_existing_cluster` (https://bugzilla.suse.com/show_bug.cgi?id=1051258)。
- 在第 4.5.1.7 节 “`policy.cfg` 文件示例”中添加了 openATTIC 角色 (https://bugzilla.suse.com/show_bug.cgi?id=1052076)。
- 在第 4.5.1.7 节 “`policy.cfg` 文件示例”中纠正了 `profile-default` 路径 (https://bugzilla.suse.com/show_bug.cgi?id=1051760)。
- 将之前的小节拆编成新章第 7 章 “自定义默认配置” (https://bugzilla.suse.com/show_bug.cgi?id=1050238)。
- 参考DeepSea 阶段说明中的第 1.2.3 节 “Ceph 节点和守护进程”，以提供最新的 Ceph 服务列表 (https://bugzilla.suse.com/show_bug.cgi?id=1050221)。
- 在第 4 章 “使用 DeepSea/Salt 部署”中改善了 Salt Master 说明和措辞 (https://bugzilla.suse.com/show_bug.cgi?id=1050214)。
- 在第 1.2.3 节 “Ceph 节点和守护进程”中添加了可选节点角色说明 (https://bugzilla.suse.com/show_bug.cgi?id=1050085)。
- 更新了一般升级过程（https://bugzilla.suse.com/show_bug.cgi?id=1048436、https://bugzilla.suse.com/show_bug.cgi?id=1048959 和 https://bugzilla.suse.com/show_bug.cgi?id=104i7085）。
- 添加了新的 DeepSea 角色“Ceph Manager”(https://bugzilla.suse.com/show_bug.cgi?id=1047472)。
- 添加了第 5.5 节 “从 SUSE Enterprise Storage 4（`ceph-deploy` 部署）升级到版本 5”(https://bugzilla.suse.com/show_bug.cgi?id=1048436)。

- 在 DeepSea 阶段说明中将阶段 0 指定为完全可选的阶段 (https://bugzilla.suse.com/show_bug.cgi?id=1045845)。
- 在第 9.1.1 节 “对象网关配置”中更新了默认存储池列表 (https://bugzilla.suse.com/show_bug.cgi?id=1034039)。
- 现在，在第 9 章 “Ceph Object Gateway”中添加了 DeepSea 所要部署的对象网关的“重要说明”片段 (https://bugzilla.suse.com/show_bug.cgi?id=1044928)。
- 在第 4.3 节 “集群部署”中纠正了外壳脚本 (https://bugzilla.suse.com/show_bug.cgi?id=1044684)。
- 第 5.2 节 “一般升级过程”中添加了“设置 `require-osd-release luminous` 标识”(https://bugzilla.suse.com/show_bug.cgi?id=1040750)。
- 在第 4.5.1.7 节 “`policy.cfg` 文件示例”中添加了示例 `policy.cfg` 的注释 (https://bugzilla.suse.com/show_bug.cgi?id=1042691)。
- 在第 4.3 节 “集群部署”中改进了用于擦除 OSD 磁盘的命令 (https://bugzilla.suse.com/show_bug.cgi?id=1042074)。
- 在第 4.3 节 “集群部署”中删除了有关在 Salt Master 上安装 `salt-minion` 的建议 (https://bugzilla.suse.com/show_bug.cgi?id=1041590)。
- 在第 4.3 节 “集群部署”和第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中提到禁用 AppArmor (https://bugzilla.suse.com/show_bug.cgi?id=1039852)。
- 在第 4.3 节 “集群部署”中添加了防火墙建议 (https://bugzilla.suse.com/show_bug.cgi?id=1039344)。
- 在第 7.1 节 “使用自定义的配置文件”中删除了 openATTIC `systemd` 命令行的 XML-RPC 参考 (https://bugzilla.suse.com/show_bug.cgi?id=1037371)。
- 在第 4.3 节 “集群部署”中纠正了 YAML 语法 (https://bugzilla.suse.com/show_bug.cgi?id=1035498)。
- 在第 4.5.1.2 节 “角色指定”中添加了“ganesha”角色说明 (https://bugzilla.suse.com/show_bug.cgi?id=1037365)。

- 在第 4.5.1 节 “`policy.cfg` 文件”中澄清并改进了句式 (https://bugzilla.suse.com/show_bug.cgi?id=1037360)。
- 在第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中添加了从 SUSE Enterprise Storage 4 到版本 5 的升级过程 (https://bugzilla.suse.com/show_bug.cgi?id=1036266)。
- 在第 4.3 节 “集群部署”中将术语“供应”替换成了“准备”（https://bugzilla.suse.com/show_bug.cgi?id=1036400 和 https://bugzilla.suse.com/show_bug.cgi?id=1036492）。
- 在第 4.5.1.6 节 “项目过滤”中添加了有关高级方法的警告 (https://bugzilla.suse.com/show_bug.cgi?id=1036278)。
- 在第 4.5.1.7 节 “`policy.cfg` 文件示例”中替换了冗余 `role-admin` 指定 (https://bugzilla.suse.com/show_bug.cgi?id=1036506)。
- 改进了 DeepSea 阶段说明和第 4.3 节 “集群部署”(https://bugzilla.suse.com/show_bug.cgi?id=1036278)。
- 在第 7.1 节 “使用自定义的配置文件”中添加了部署步骤的修改内容 (https://bugzilla.suse.com/show_bug.cgi?id=1026782)。
- 根据 (https://bugzilla.suse.com/show_bug.cgi?id=1020920) 的建议澄清并改进了第 4 章 “使用 DeepSea/Salt 部署”。
- 在第 7.1 节 “使用自定义的配置文件”中建议启用自定义 openATTIC 服务 (https://bugzilla.suse.com/show_bug.cgi?id=989349)。
- 已将网络建议移到第 2 章 “硬件要求和建议”，并包含了第 2.7.1 节 “将专用网添加到正在运行的集群”(https://bugzilla.suse.com/show_bug.cgi?id=1026569)。